HEWLETT PACKARD

HP-UX Concepts and Tutorials

Vol. 5: Data Communications

HP-UX Concepts and Tutorials
Vol. 5: Data Communications

Manual Reorder No. 97089-90060

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

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

July 1984...First Edition – Part numbered 97089-90004 was 4 volumes and was shi
with HP-UX 4.0 on Series 500 Computers and with HP-UX 2.1, 2.2, 2.3, and 2.
Series 200 Computers. Each volume did not have an individual part number. '
was obsoleted in April, 1985 and replaced with Manual Kit #97070-87903 w
includes:

	Title	Manual P/N	Binder P/N
Vol. 1:	Text Processing and Formatting	97089-90020	9282-1023
Vol. 2:	Programming Environment	97089-90030	9282-1023
Vol. 3:	Software Development Tools	97089-90040	9282-1023
Vol. 4:	Shells and Miscellaneous Tools	97089-90050	9282-1023
Vol. 5:	Data Communications	97089-90060	9282-1023
Vol. 6:	Graphics	97089-90070	9282-1023

April 1985...Edition 1 – Volume 5: Data Communications

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 HP computers. Others were written at Bell Laboratories or University of California at Berkeley and have been tailored for HP computers.

HP-UX Concepts and Tutorials has six volumes:

- Volume 1: Text Processing and Formatting
- Volume 2: Programming Environment
- Volume 3: Software Development Tools
- Volume 4: Shells and Miscellaneous Tools
- Volume 5: Data Communications
- Volume 6: Graphics

This is "Vol. 5: Data Communications" and the articles it includes are:

Warranty Statement

Hewlett-Packard products are warranted against defects in materials and workmanship. For Hewlett-Packard computer system products sold in the U.S.A. and Canada, this warranty applies for ninety (90) days from the date of shipment.* 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 neccessitated 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.

NO OTHER WARRANTY IS EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. HEWLETT-PACKARED SHALL NOT BE LIABLE FOR CONSEQUENTIAL DAMAGES.

HP 9000 Series 200

For the HP 9000 Series 200 family, the following special requirements apply. The Model 216 computer comes with a 90-day, Return-to-HP warranty during which time HP will repair your Model 216, however, the computer must be shipped to an HP Repair Center.

All other Series 200 computers come with a 90-Day On-Site warranty during which time HP will travel to your site and repair any defects. The following minimum configuration of equipment is necessary to run the appropriate HP diagnostic programs: 1) .5 Mbyte RAM; 2) HP-compatible 3.5″ or 5.25″ disc drive for loading system functional tests, or a system install device for HP-UX installations; 3) system console consisting of a keyboard and video display to allow interaction with the CPU and to report the results of the diagnostics.

To order or to obtain additional information on HP support services and service contracts, call the HP Support Services Telemarketing Center at (800) 835-4747 or your local HP Sales and Support office.

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

Table of Contents

Using the System Console with HP9000 Series 200 Computers

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" set[1] is installed in the computer. If this condition is met and an ITE is present, the ITE is assigned the special (device) file `/dev/ttyOO` and is considered to be the first non-system console terminal connected to HP-UX. (If no ITE is present, `/dev/ttyOO` 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.*

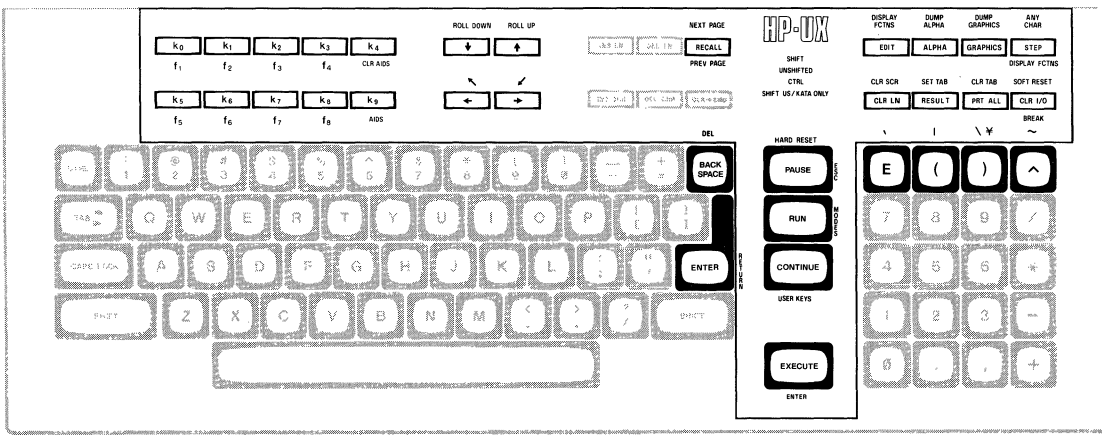
The keyboard overlay supplied with your Series 200 computer allows you to convert it for use as an HP-UX system console keyboard. The key labels on the overlay are color-coded as follows:

- 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.

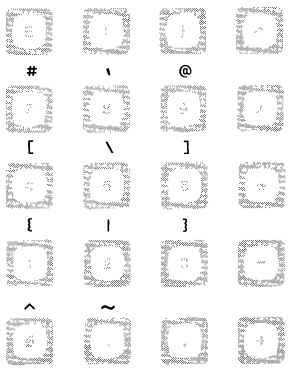
CTRL provides access to the standard ASCII control characters.

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.

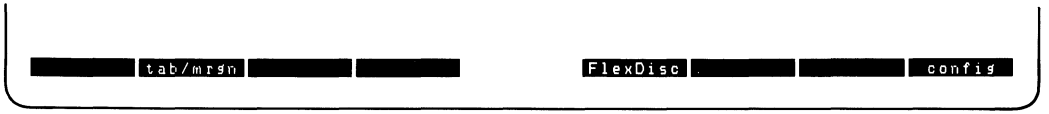
SET TAB
RESULT — sets tab stops. To set a tab stop move the cursor to the desired location, hold the SHIFT key down, and press SET TAB.

CLR TAB
PRT ALL — 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.

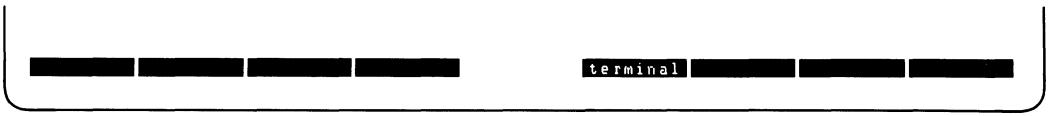
Cursor Control

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)

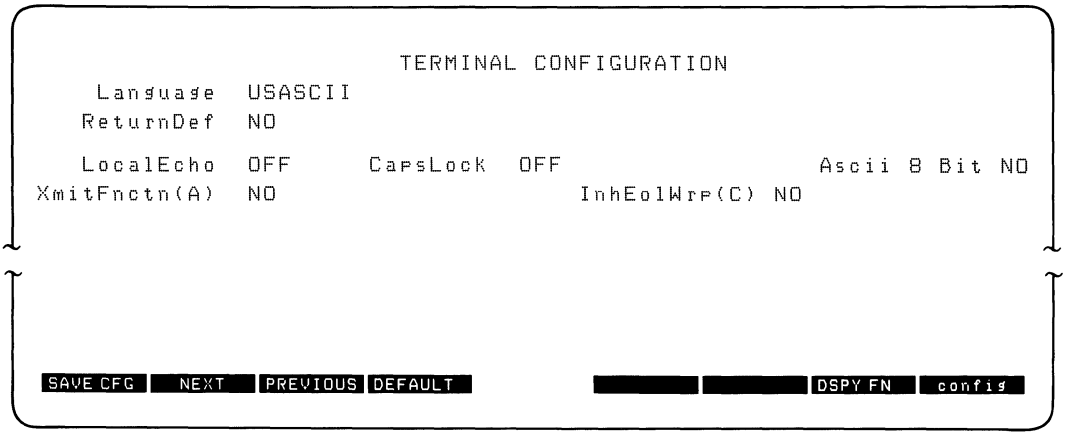


press: k8 (config)



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 **XmitFnctn(A)**.



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 **XmitFnctn(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.

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.

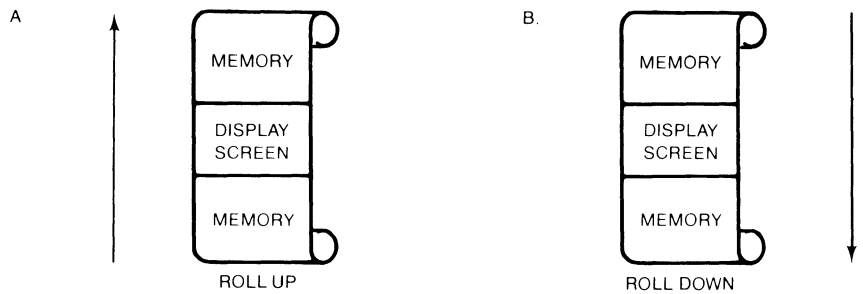
ROLL UP
shift- ↑ scrolls the screen display upward.

shift- ← moves the cursor to its upper home position.

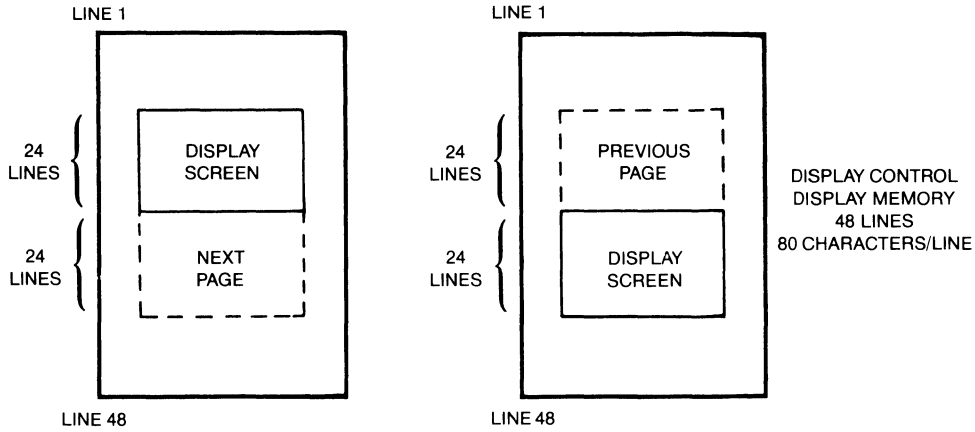
shift- → moves the cursor to its lower home position.

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 (PREV PAGE) — moves you back a page within your text.

shift - RECALL (NEXT PAGE) — moves you forward a page within your text.

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.

shift - BACK SPACE (DEL) — sends the DEL character to the computer.

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.

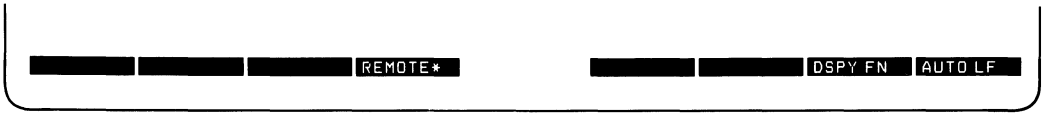
shift-(— vertical bar.

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 MODES — 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.

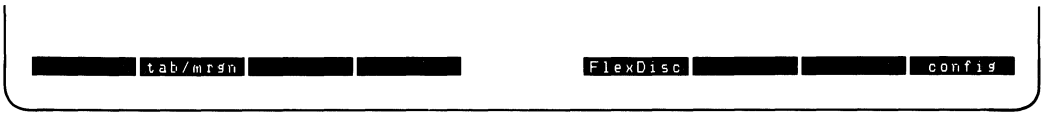
When the **remote mode** is enabled and a key is pressed, the terminal transmits the associated ASCII code to HP-UX. In **local mode** (remote mode is disabled), when a character key is pressed, the associated character is displayed at the current cursor position on the screen (nothing is transmitted 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 CRTL 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 [k4] — clears the screen display of the function key labels. The user function keys, however, are still enabled.

[EXECUTE] ENTER — not implemented.

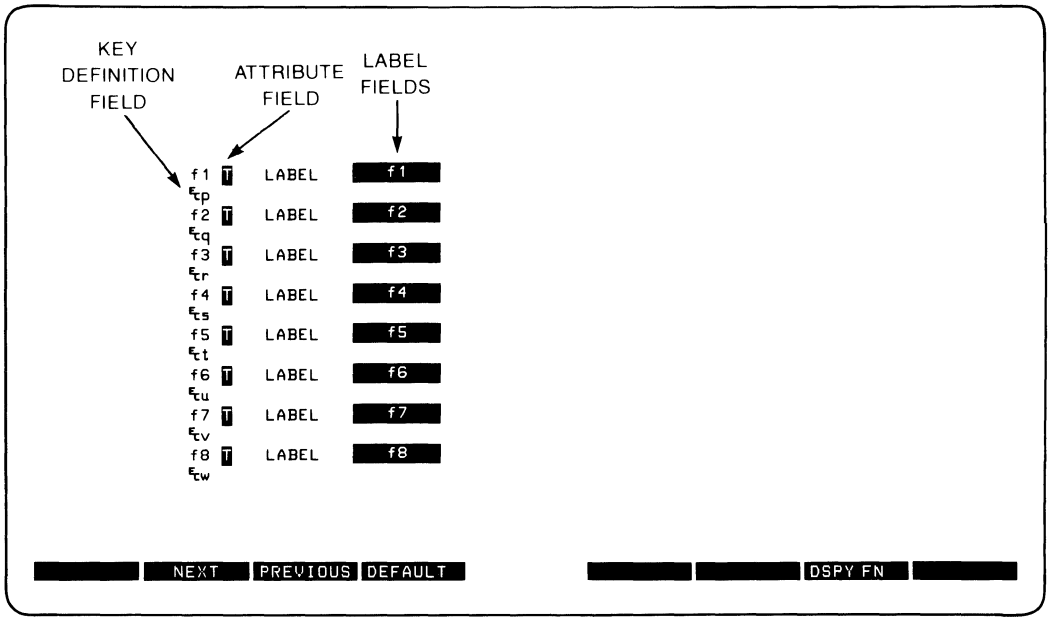
[CONTINUE] USER KEYS — displays eight function keys which can be defined either locally by the user or remotely by a program executed in a host computer. By "defined" it is meant:

- 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:



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 [f2] (NEXT) or [f3] (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 [f7] (DSPY FN) for the Model 236 and for the Model 226 use function key [f1].

The [ENTER] (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 [ENTER] (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 [INS CHR], [DEL CHR] and [CLR LN] keys for editing the contents of the label and key definition fields.

When you are finished defining all the desired keys, press the [k4] (AIDS), [RUN] (MODES) or [CONTINUE] (USER KEYS) key (in all three cases the user keys menu disappears from the screen). When you press [CONTINUE] (USER KEYS), the defined user key labels are displayed across the bottom of the screen and the [f1] through [f8] 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:

^{E}c&f <attribute><label length><string length><label><string>

where:

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

<key> = 1-8k : 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 ^{C}R and ^{L}F, 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 k5 user key. The key is to have attribute "N".

^{E}c&f5k5d4Llogin TOM$^{C}R$$^{E}c$&jB

After issuing the foregoing escape sequence from your program to the terminal, the k5 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 ^{E}c&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 k4 (CLR AIDS)).
- You can enable the mode selection keys (this is the equivalent of pressing the RUN (MODES) key).
- You can enable the user keys (this is the equivalent of pressing the CONTINUE (USER KEYS) key).

The escape sequences are as follows:

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

ESC&JA Enables the modes key.

ESC&JB 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
Graphics ON Alpha OFF	Press Graphics	No change in display
Graphics ON Alpha OFF	Press Alpha	Both displays on screen
Graphics ON Alpha 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 — does the following:

- 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.

The data on the screen, all terminal operating modes (except display functions mode), and all active configuration parameters are unchanged.

HARD RESET
shift-PAUSE — has the same effect as turning the computer's power off and then back on.

A hard reset does the following:

- 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:
  - 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 *HP-UX Reference*. To learn how to use this signal in a shell script read "Shell Programming" in your *HP-UX Selected Articles*.

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.

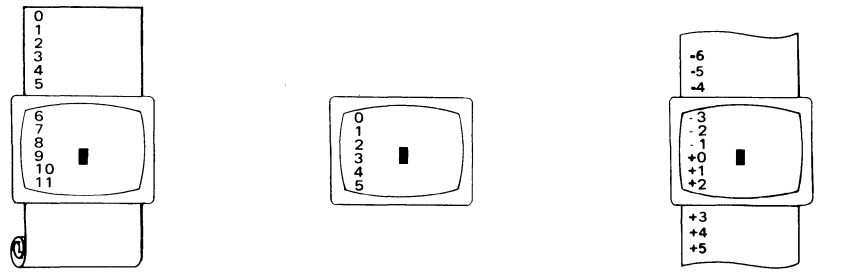
Memory Addressing Scheme

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



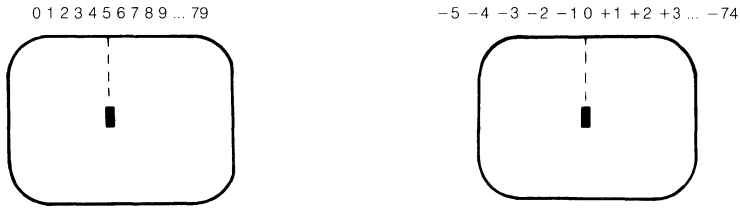
a.) Absolute: row 6 b.) Screen Relative: **row 0** c.) Cursor Relative: **row −3**

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.



a.) Absolute and Screen Relative

b.)Cursor Relative

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 ESC a to the terminal, the terminal returns to the program an escape sequence of the form ESC & a xxx c yyy R CR, 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 ESC `, the terminal returns to the program an escape sequence of the form ESC & a xxx c yyy Y CR 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:

```
Ec&a<column number> c <row number>Y
Ec&a<row number> y <column number>C
Ec&a<column number>C
Ec&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:

```
Ec&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:

```
Ec&a<column number> c <row number>R
Ec&a<row number> r <column number>C
Ec&a<column number>C
Ec&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 (lines/page) X num_page (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:

```
Ec&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:

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

```
Ec&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:

```
Ec&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:

```
Ec&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:

```
Ec&a9c 47R
```

Display Enhancements

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

- Inverse Video - black characters are displayed agains 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 vedio 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:

Ec&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" (Ec&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:

Ec*dE Turn on the alphanumeric display; enable writing to the alphanumeric display.

Ec*dF 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 Ec&v escape sequence used above:

Ec&v n<parameter>

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

Ec&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:

```
Ec&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 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

```
Ec&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

```
Ec&v 0m 1a 1y 5I
```

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

```
Ec&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.

```
Ec&v 5I
```

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

Note

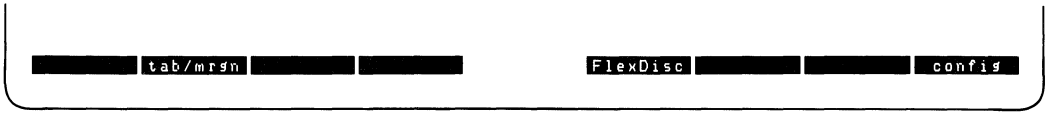
Supplying neither a foreground nor a background color when defining a color pair causes both the foreground and background to be black.

Configuring the ITE

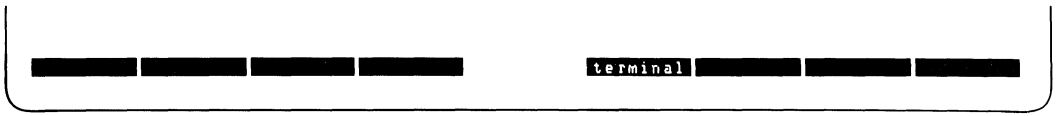
The Internal Terminal Emulator is designed so that the various ITE characteristics can be configured quickly by displaying configure "menus" on the screen and then using system function keys to change the content of these menus.

Configuration Function Keys

To gain access to the configuration menus through the keyboard, press the function key [k9] (labeled AIDS on the system console overlay). This causes the following softkey display to appear at the bottom of the screen:



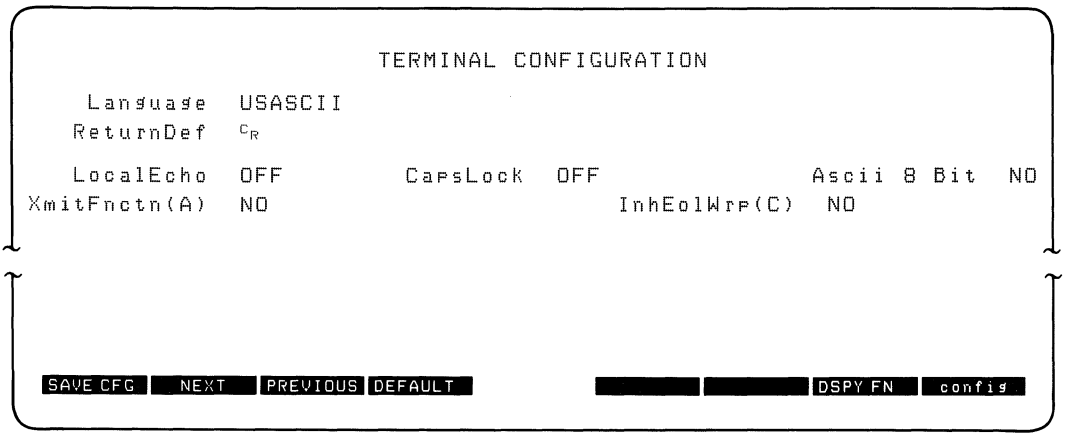
The function keys [f2] (tab/mrgn) and [f5] (FlexDisc) were covered in the section "Using the Internal Terminal Emulator" along with a brief discussion on the function key [k8] (config). When you press [f8] (config) a new softkey display appears at the bottom of your screen.



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` 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.

Default: $^{C}_{R}$ space

`LocalEcho` specifies whether characters entered through the keyboard are both displayed on the screen and transmitted to the host computer.

ON (`E`c`&k 1L`)

Characters entered through the keyboard are both displayed on the screen and transmitted to the host computer.

`CapsLock` determines whether the ITE generates the full 128-characters ASCII set or only Teletype-compatible codes.

ON (`E`c`&k 1C`)

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.

OFF (`E`c`&k 0C`)

The ITE generates the full 128-character ASCII set of codes.

Default: `OFF`

`XmitFnctn(A)` determines whether the escape code functions are executed at the ITE and transmitted to the host computer.

YES (`E`c`&s 1A`)

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.

NO (`E`c`&s 0A`)

The escape code sequences for the major function keys are executed locally but NOT transmitted to the host computer.

Note that display functions will emit `E`c Z and `E`c Y to a host computer.

Default: `NO`

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

NO (Ec&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 (Ec&s1C)

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 AZERTY[1] layout with mutes)
FRANCAIS qwM (French QWERTY layout with mutes)
FRANCAIS az (French AZERTY[1] layout)
FRANCAIS qw (French QWERTY layout)
DEUTSCH (German)
ESPANOL M (Spanish with mutes)
ESPANOL (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 7 6 5 / ROW BIT 4 3 2 1		0 0 0 0	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
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0 0 0 0	0	NU	DL	(SP)	0	@	P	`	p				—	à	À		
0 0 0 1	1	SH	D1	!	1	A	Q	a	q					ê	Î		
0 0 1 0	2	SX	D2	"	2	B	R	b	r					ô	Ø		
0 0 1 1	3	EX	D3	#	3	C	S	c	s				°	û	Æ		
0 1 0 0	4	ET	D4	$	4	D	T	d	t					á	à		
0 1 0 1	5	EQ	NK	%	5	E	U	e	u				ç	é	í		
0 1 1 0	6	AK	SY	&	6	F	V	f	v				Ñ	ó	ø		
0 1 1 1	7	AL	EB	´	7	G	W	g	w				ñ	ú	æ		
1 0 0 0	8	BS	CN	(	8	H	X	h	x			´	¡	à	Ä		
1 0 0 1	9	HT	EM	)	9	I	Y	i	y			`	¿	è	Ì		
1 0 1 0	10	LF	SB	*	:	J	Z	j	z			^	¤	ò	Ö		
1 0 1 1	11	VT	EC	+	;	K	[	k	{			¨	£	ù	Ü		
1 1 0 0	12	FF	FS	,	<	L	\	l	|			~		ä	É		
1 1 0 1	13	CR	GS	-	=	M	]	m	}				§	ë	ï		
1 1 1 0	14	SO	RS	.	>	N	^	n	~					ö	ß		
1 1 1 1	15	SI	US	/	?	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 romote mode. All characters for the various languages mentioned in the menu are now available to you except KATAKANA.

COL BIT 8 7 6 5 / ROW BIT 4 3 2 1		0 0 0 0	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
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0 0 0 0	0	NUL	DLE		0	@	P	`	p				ー	タ	ミ		
0 0 0 1	1	SOH	DC1	!	1	A	Q	a	q			｡	ア	チ	ム		
0 0 1 0	2	STX	DC2	"	2	B	R	b	r			｢	イ	ツ	メ		
0 0 1 1	3	ETX	DC3	#	3	C	S	c	s			｣	ウ	テ	モ		
0 1 0 0	4	EOT	DC4	$	4	D	T	d	t			、	エ	ト	ヤ		
0 1 0 1	5	ENQ	NAK	%	5	E	U	e	u			・	オ	ナ	ユ		
0 1 1 0	6	ACK	SYN	&	6	F	V	f	v			ヲ	カ	ニ	ヨ		
0 1 1 1	7	BEL	ETB	'	7	G	W	g	w			ァ	キ	ヌ	ラ		
1 0 0 0	8	BS	CAN	(	8	H	X	h	x			ィ	ク	ネ	リ		
1 0 0 1	9	HT	EM	)	9	I	Y	i	y			ゥ	ケ	ノ	ル		
1 0 1 0	10	LF	SUB	*	:	J	Z	j	z			ェ	コ	ハ	レ		
1 0 1 1	11	VT	ESC	+	;	K	[	k	{			ォ	サ	ヒ	ロ		
1 1 0 0	12	FF	FS	,	<	L	¥	l	|			ャ	シ	フ	ワ		
1 1 0 1	13	CR	GS	-	=	M	]	m	}			ュ	ス	ヘ	ン		
1 1 1 0	14	SO	RS	.	>	N	^	n	~			ョ	セ	ホ	゛		
1 1 1 1	15	SI	US	/	?	O	_	o	｡			ッ	ソ	マ	゜		K

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 romote mode. To type Japanese ASCII characters, press CTRL - ('). If you want KATAKANA characters, press CTRL - (').

For the French keyboard layouts, the AZERTY[2] 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` transmitts from full set of 8-bit codes when enabled (YES) and transmits only codes less than 128 when disabled (NO).

Values: YES (`Ec&k 1I`) = 8-bit codes.
NO (`Ec&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.

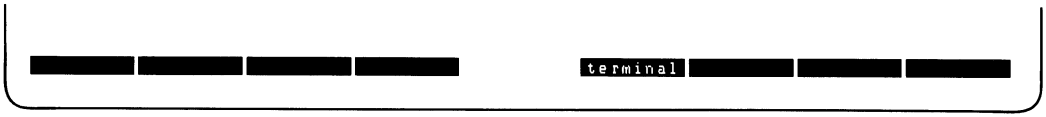
NEXT — changes the setting of the field you are presently in to the next setting in that field. For example, if you press [TAB] until you are located at the field **LocalEcho OFF** and then press [f2] the **LocalEcho** field changes to **ON**.

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 communcation 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
Ec1	Set tab
Ec2	Clear tab
Ec3	Clear all tabs
Ec4	NOT IMPLEMENTED (Set left margin)
Ec5	NOT IMPLEMENTED (Set right margin)
Ec9	NOT IMPLEMENTED (Clear all margins)
Ec@	NOT IMPLEMENTED (Makes the terminal program wait approximately one second.)
EcA	Cursor up
EcB	Cursor down
EcC	Cursor right
EcD	Cursor left
EcE	Hard reset (power on reset of ITE)
EcF	Cursor home down
EcG	Move cursor to the left margin
EcH	Cursor home up
EcI	Horizontal tab
EcJ	Clear screen from cursor to the end of memory.

Escape Sequences for ITE Control (continued)

Escape Code	Function
^[cK	Clear line from cursor to end of line
^[cL	Insert line
^[cM	Delete line
^[cP	Delete character
^[cQ	Start insert character mode
^[cR	End insert character mode
^[cS	Roll up
^[cT	Roll down
^[cU	Next page
^[cV	Previous page
^[cW	NOT IMPLEMENTED (Format mode on)
^[cX	NOT IMPLEMENTED (Format mode off)
^[cY	Enables display functions mode
^[cZ	Disables display functions mode
^[c[	NOT IMPLEMENTED (Start unprotected field)
^[c]	NOT IMPLEMENTED (End unprotected/transmit-only field)
^[c^	NOT IMPLEMENTED (Primary terminal status request)
^[c`	Sense cursor position(relative)
^[ca	Sense cursor position (absolute)
^[cb	NOT IMPLEMENTED (Unlock keyboard)
^[cc	NOT IMPLEMENTED (Lock keyboard)
^[cd	NOT IMPLEMENTED (Transmit a block of text to computer)
^[cf	NOT IMPLEMENTED (Modem disconnect)
^[cg	Soft reset (of ITE)
^[ch	Cursor home up.
^[ci	Backtab
^[cJ	NOT IMPLEMENTED (Begin User Key Definition mode)
^[cK	NOT IMPLEMENTED (End User Keys Definition mode)
^[cl	NOT IMPLEMENTED (Begin Memory Lock mode)
^[cm	NOT IMPLEMENTED (End Memory Lock Mode)
^[cp	Default value for softkey f1
^[cq	Default value for softkey f2
^[cr	Default value for softkey f3
^[cs	Default value for softkey f4
^[ct	Default value for softkey f5
^[cu	Default value for softkey f6
^[cv	Default value for softkey f7
^[cw	Default value for softkey f8
^[cz	NOT IMPLEMENTED (Initiate terminal self test)
^[c~	NOT IMPLEMENTED (Secondary terminal status request)

Extended Escape Sequences

Escape Code Sequence	Function
Ec&a <col>c <row>Y	Moves the cursor to column "col" and screen row "row" or the screen (screen relative addressing).
Ec&a <col>c <row>R	Moves the cursor to column "col" and row "row" in memory (absolute addressing).
Ec&a ±<col>c ±<row>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.
Ec&a ±<col>c ±<row>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.
Ec&¬0L	NOT IMPLEMENTED (Unlock configuration)
Ec&¬1L	NOT IMPLEMENTED (Lock configuration)
Ec&d<char>	Selects the display enhancement indicated by <char> to begin at the present cursor position. <char> can be @ or A thru O.
Ec&k<x>A	AUTO LF enable: x = 1; disable: x = 0
Ec&k<x>B	NOT IMPLEMENTED (BLOCK enable: x = 1; disable: x = 0)
Ec&k<x>C	Caps Lock enable: x = 1; disable: x = 0
Ec&k<x>I	ASCII8Bits enable: x = 1; disable: x = 0
Ec&k<x>J	NOT IMPLEMENTED (FrameRate 50 Hz : x = 1; 60 Hz : x = 0)
Ec&k<x>L	LocalEcho enable: x = 1; disable: x = 0
Ec&k<x>M	NOT IMPLEMENTED (MODIFY ALL enable: x = 1; disable: x = 0)
Ec&k<x>N	NOT IMPLEMENTED (SPOWLatch enable: x = 1; disable: x = 0)
Ec&k<x>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
Ec&k<x>R	REMOTE enable: x = 1; disable: x = 0

Extended Escape Sequences (continued)

Escape Code Sequence	Function
Ec&s<x>A	xmitFnctn(A) enable: x=1; disable: x=0
Ec&s<x>B	NOT IMPLEMENTED (SPOW(B) enable: x=1; disable: x=0)
Ec&s<x>C	InhEolWrp(C) enable: x=1; disable: x=0
Ec&s<x>D	NOT IMPLEMENTED (Line/Page(D) enable: x=1; disable: x=0)
Ec&s<x>G	NOT IMPLEMENTED (InfHndShk(G) enable: x=1; disable: x=0)
Ec&s<x>H	NOT IMPLEMENTED (Inh DC2(H) enable: x=1; disable: x=0)
Ec&w 12F	Turns on the display window (top 24 rows)
Ec&w 13F	Turns off the display window
Ec*d <parameters>	List of <parameters> 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.
Ec* s <parameter> ^	Read device I.D. Status is parameter number 1.
Ec&j <x>	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.
Ec&f <attribute>a <key>k <label length>d <string length>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.

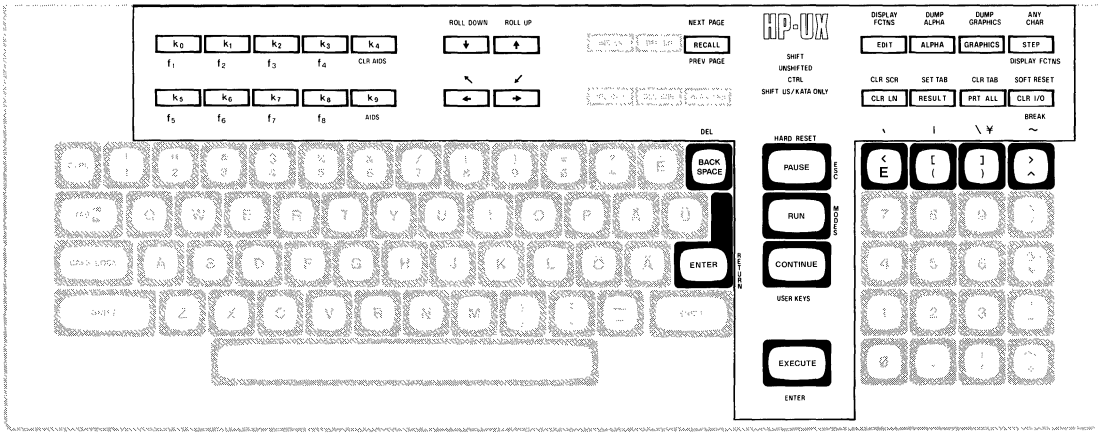
If an escape sequence is not recognized, the terminal ignores subsequent characters until ASCII decimal characters 0 thru 31 or 64 thru 95 is received, terminating the sequence. Note that Ec will terminate the old sequence and start a new one.

Keyboard Diagrams for Other Languages

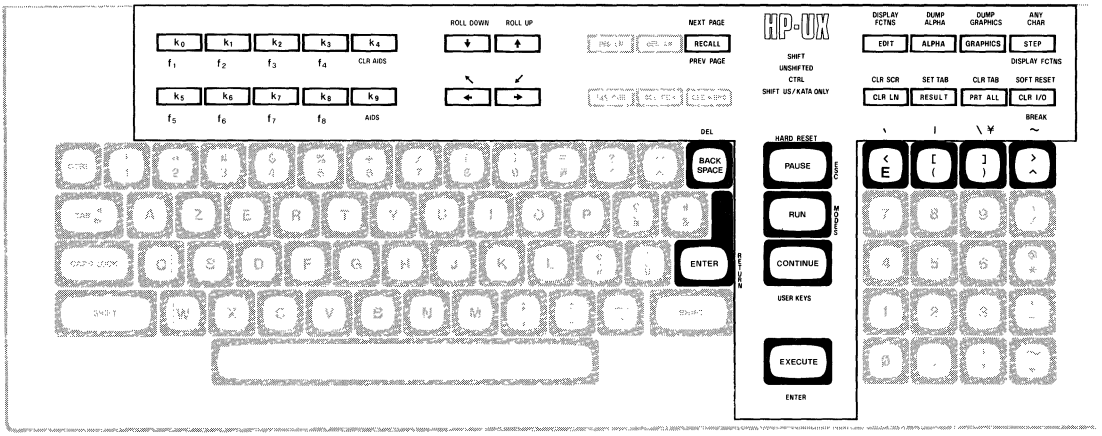
This section shows diagrams of the types of keyboards which are available to the Series 200 user. The keyboard option you now have can be configured to any one of the languages by use of the TERMINAL CONFIGURATION menu previously mentioned in this article.

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.

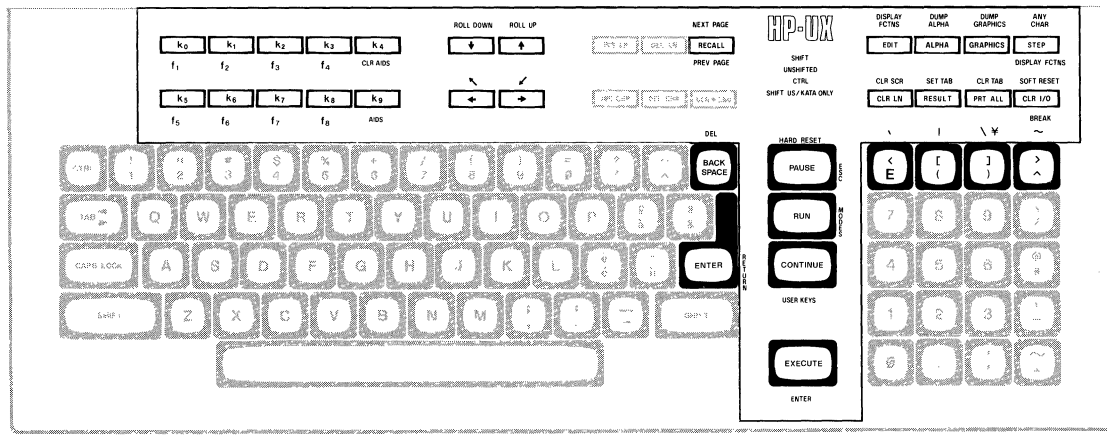
You will notice as you look at the keyboards that a majority of the character key and numeric key labels have changed. To use your keyboard effectively in anyone of these languages, you will have to re-label the key caps.



SVENSK/SUOMI (Swedish/Finnish)

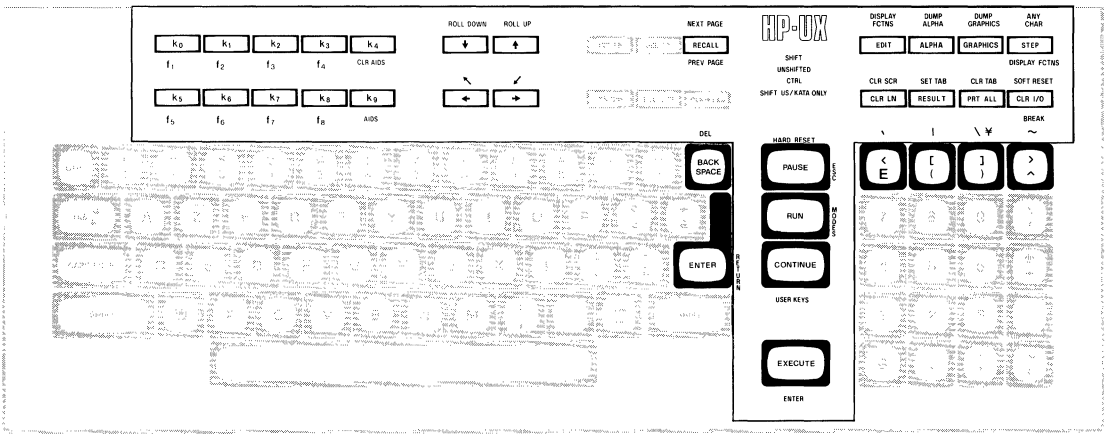


FRANCAIS azM (French AZERTY[3] layout with mutes is not available)

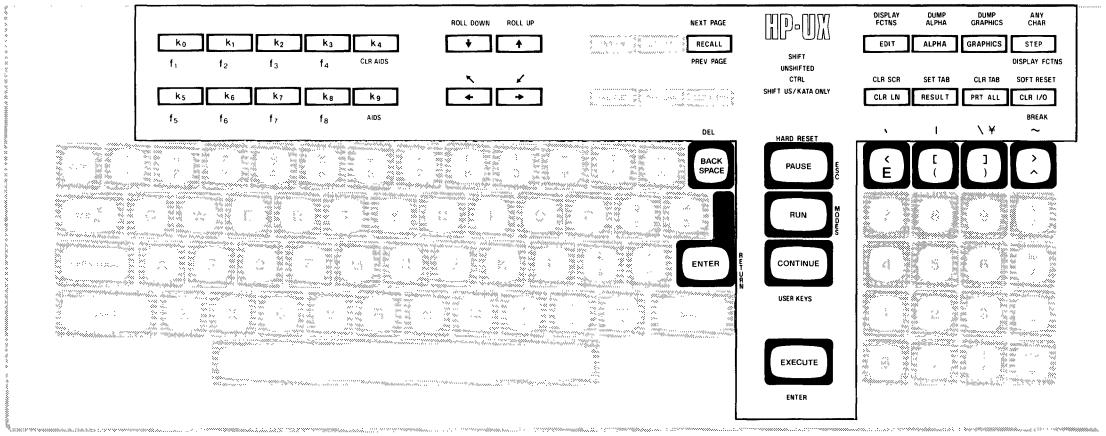


FRANCAIS qwM (French QWERTY layout with mutes)

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.

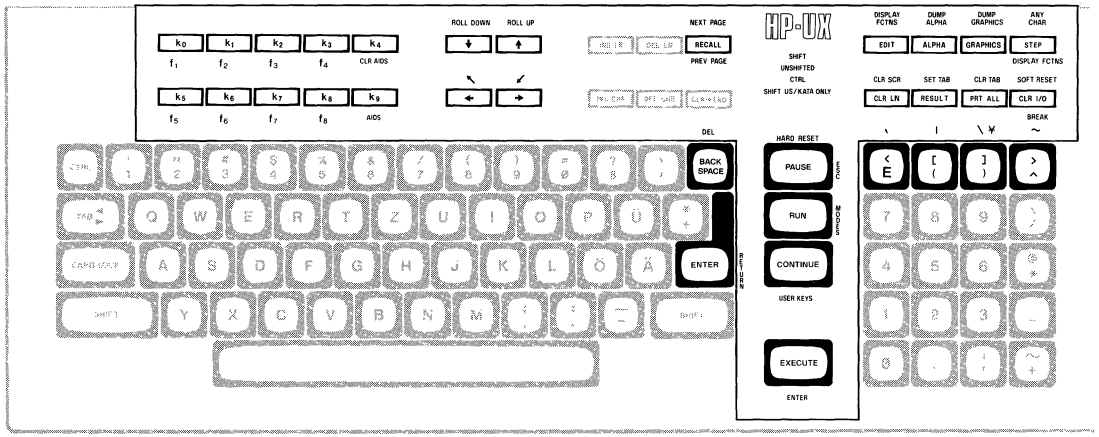


FRANCAIS az (French AZERTY[4] layout is not available)

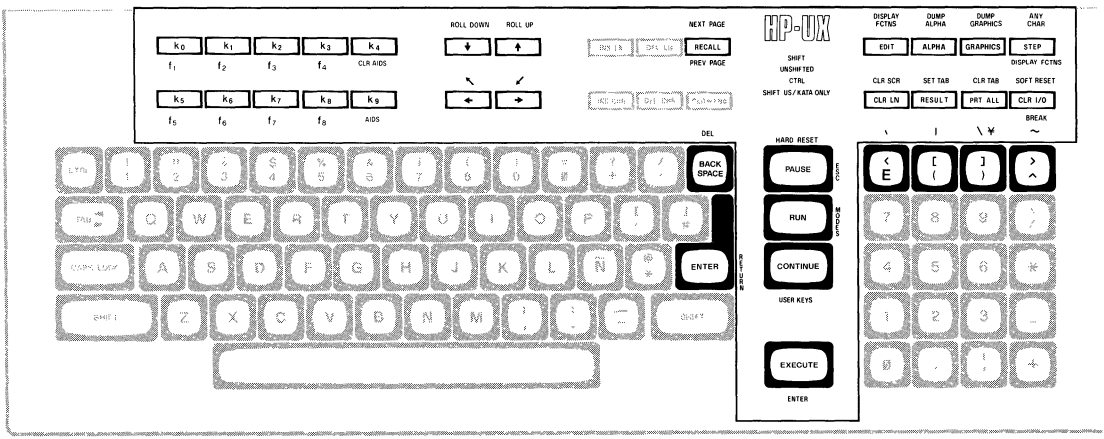


FRANCAIS qw (French QWERTY layout)

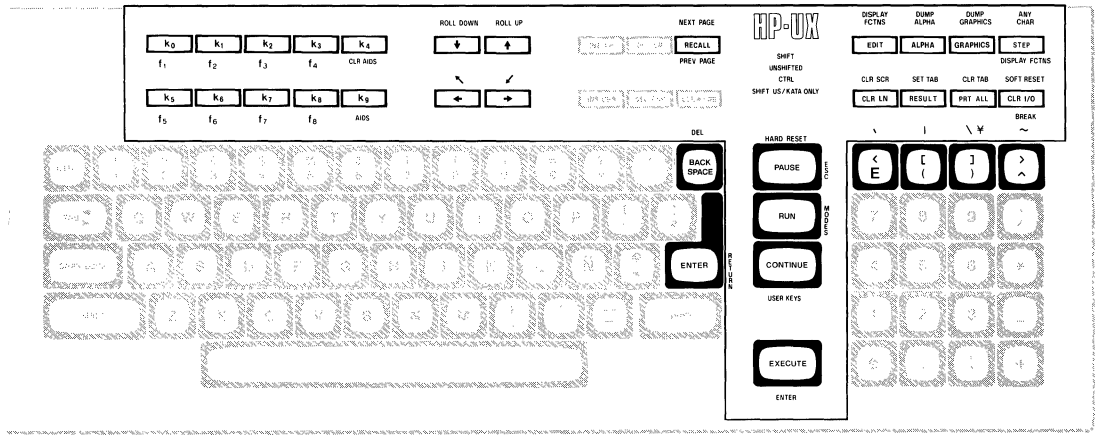
4 The AZERTY characters can be obtained only through a software configuration of the keyboard. Physical AZERTY keyboards or hardware are not available on Series 200 computers.



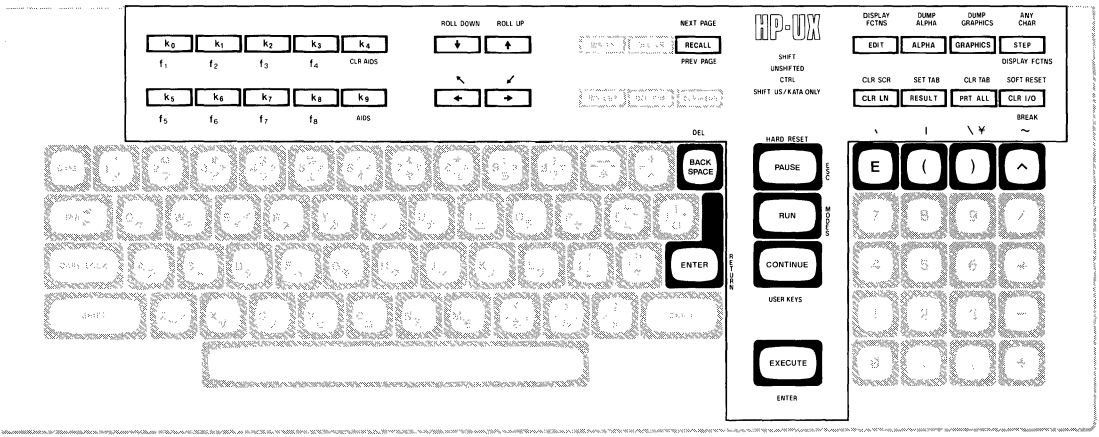
DEUTSCH (German)



ESPANOL M (Spanish with mutes)



ESPANOL (Spanish)



KATAKANA (Japanese)

Notes

Table of Contents

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).

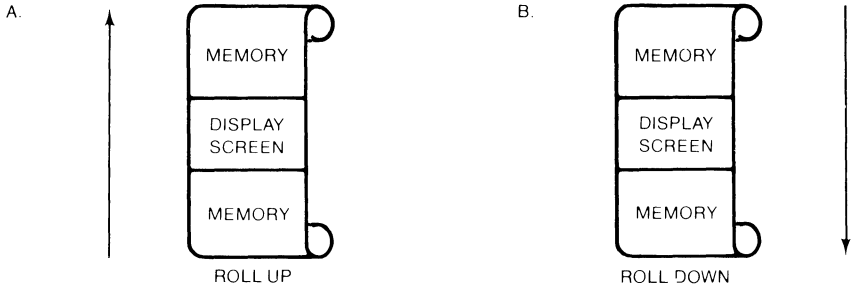
Cursor Control

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

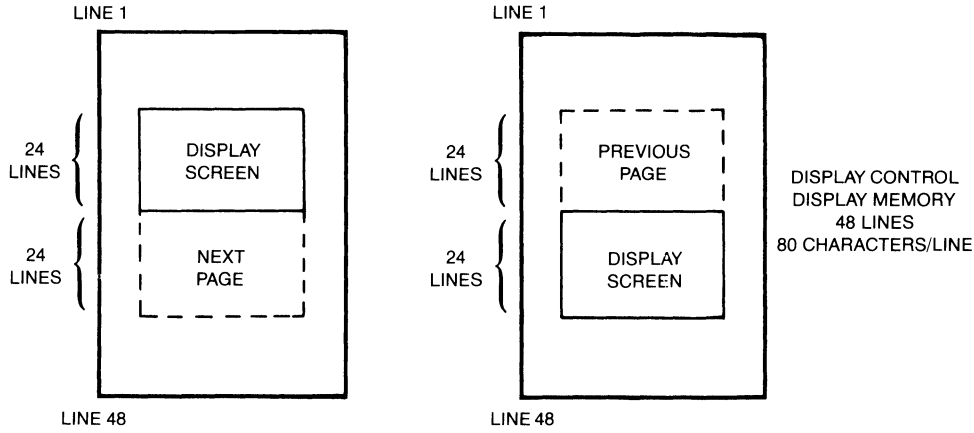
Key	Escape Sequence	Feature
[↑]	\EA	Move the cursor up one row in the current column position. Holding the key down causes the cursor to move continuously, row by row, until the key is released. When the cursor is in the top row of the screen, moving the cursor up actually moves the cursor to the same 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.

Key	Escape Sequence	Feature
none	\EG	Move the cursor to the left margin.
TAB	\EI	Move the cursor forward to the next tab stop.
SHIFT TAB	\Ei	Move the cursor backwards to the previous tab stop.
ROLL ↑	\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.
ROLL ↓	\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.
SHIFT ROLL ↑	\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. The cursor is placed at the left margin, at the top row of the display.
SHIFT ROLL ↓	\EV	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. The cursor is placed at the left margin, at the top row of the display.

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

Accross 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 MODE			DISPLAY FUNCT	AUTO 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 paramater 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 16) through (7 23), respectively.

Controlling Function Key Labels Programmatically

From an application program you can control the function key labels display by using the following escape sequences:

\E&j@ Disable the function keys and remove all key labels from the screen. Note: If a function key is hit while the terminal is in remote, the function key is transmitted whether or not function key labels are displayed on the screen.

\E&jA Enable the mode selection keys.

\E&jB Enable the user-defined labels.

\E&jR Enable screen labels.

\E&jS Disable screen labels.

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.

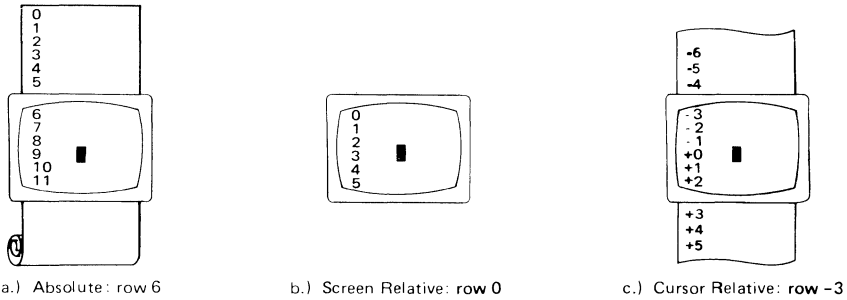
Memory Addressing Scheme

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 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



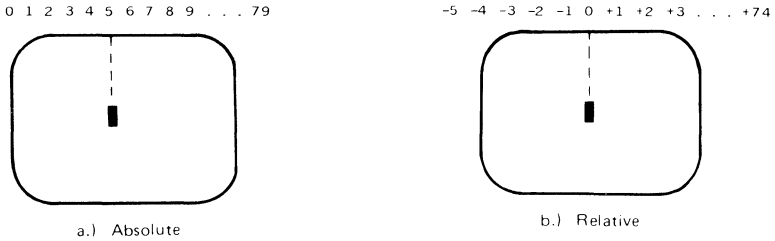
a.) Absolute: row 6 b.) Screen Relative: row 0 c.) Cursor Relative: row -3

Row Addressing

Column Addressing

Column addressing is accomplished in a manner similar to row addressing. There is no difference between screen 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 (lines/page) X num_page (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 5I

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 5I

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 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 520 will reject the background specification of pen 6, and will use black instead.)

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

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 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. 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 specifes 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 ↑) and (ROLL ↓) 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

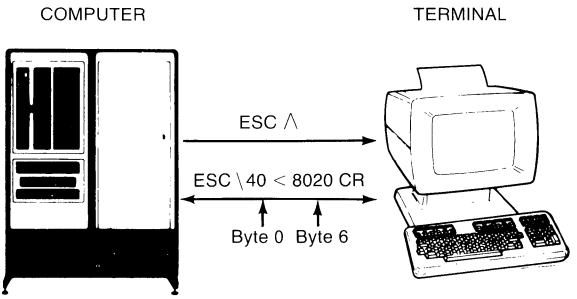
Terminal status is made up of 14 status bytes (bytes 0 through 13) containing information such as display memory size, switch settings, configuration menu settings, and terminal errors. There are two terminal status requests: primary and secondary. Each returns a set of seven status bytes.

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	Auto line feed disabled 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

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

- 1K bytes
- 2K bytes
- 4K bytes
- 8K bytes

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

BYTE 1 CONFIGURATION STRAPS A-D

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

- Strap D — Page Line — always 0
- Strap C — (Inhibit End-of-Line Wraparound) — 1 = yes (Enabled) — 0 = no (Disabled)
- Strap B — (Space Overwrite Latch) — always 0
- Strap A — (Function Key Transmission) — 1 = yes (Transmitted) — 0 = no (Not transmitted)

BYTE 2 CONFIGURATION STRAPS E-H

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

- Strap H — always 1
- Strap G — always 1
- Strap F — always 0
- Strap E — always 0

BYTE 3 LATCHING KEYS

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

- Terminal sends secondary status
- AUTO LF Key — 1 = auto LF — 0 = no auto LF
- CAPS LOCK Key — always 0
- BLOCK MODE Key — 1 = block mode — 0 = character mode

BYTE 4 TRANSFER PENDING FLAGS

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

- Secondary Status Pending — always 0
- ENTER Key Pending — always 0
- Cursor Sense Pending — always 0
- Function Key Pending — always 0

BYTE 5 ERROR FLAGS

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

- Device Error (Integral Printer Error) — always 0
- DataComm — always 0
- Self-Test — always 1

BYTE 6 DEVICE TRANSFER PENDING FLAGS

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

- Device Status Pending — always 0
- Device Operation Status Pending — always 0

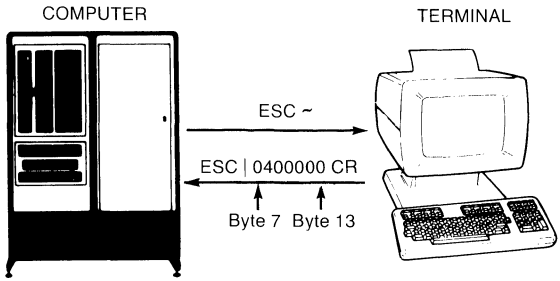
(tracks "S", "F", or "U" completion codes associated with ^{E}C&p device control sequences.)

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

- 1K bytes
- 2K bytes
- 4K bytes
- 8K bytes

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

- always 0
- 1 = I O firmware installed, integral printer present
- 0 = No APL Firmfare
- 0 = Non-programmable terminal
- 1 = Terminal identifies self

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

- Strap M
- Strap J (Auto Terminate)
  - 1 = yes (Enabled)
  - 0 = no (Disabled)
- Strap K (Clear Terminator)
  - 1 = yes (Enabled)
  - 0 = no (Disabled)
- Strap L (Self-Test Inhibit)
  - 1 = yes (Inhibit test)
  - 0 = no (Allow test)

Straps J-M do not apply to the terminal.

BYTE 10 KEYBOARD INTERFACE KEYS

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

- Switch R always 0
- Switch N Printer (Escape Code Transfer) always 0
- Switch P Compatibility Mode (Scaled) always 0
- Switch Q Compatibility Mode (Unscaled) always 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

- Strap V
- Strap S
- Strap U
- Strap T

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

- Strap Z
- Strap W (Data Comm Te
  - 1 = yes (Inhibit)
  - 0 = no (Allow)
- Strap Y
- Strap X

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

HP 98700H Graphics Display as a "Terminal"

The HP 98700H Graphics Display Station can be used as a standalone computer, as a terminal to other computers, or as system console to an HP 9000 Model 550 running HP-UX. This article describes the HP 98700H as a terminal and as system console for the Series 550. It also discusses the methods of accessing the "terminal's" features: from the keyboard and from a program or command (via escape sequences).

The software that makes the HP 98700H appear as a system console is the ITE (Internal Terminal Emulator). The ITE consists of "device driver" code contained in the HP-UX kernel. See the section on "ITE and the HP 98700H" later in this article for more information.

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 not be connected via a modem.
- must have a device file named */dev/console.*

Each HP-UX system must have a system console. When HP-UX is run on the HP 98700H, the computer's keyboard and display act as the system console.

The display portion of the "terminal" consists of a display screen and display memory. The display cursor (an inverse video square 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 (see section on "Memory Addressing Scheme" later in this article). 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 keyboard is divided into major functional groups: the alphanumeric group, the numeric pad group, the disply 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 upper case 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 codes for such features are provided in the sections that follow.

Cursor Control

Several keys exist on the keyboard for changing the location of the cursor. Table 1 describes the keys, their associated escape sequence, and their function.

Table 1. Cursor Control Keys

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.
▶	\EH or \Eh	Home up: moves the cursor to the left margin in the 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 ine in display memory appears immediately above the cursor position.
none	\EG	Move the cursor to the left margin.

Table 1. Cursor Control Keys (continued)

Key	Escape Sequence	Feature
Tab	\EI	Move the cursor forward to the next tab stop.
Shift Tab	\Ei	Move the cursor backwards to the previous tab stop.
Shift ▲	\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.
Shift ▼	\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.
Next	\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. The cursor is placed at the left margin, at the top row of the display.
Prev	\EV	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. The cursor is placed at the left margin, at the top row of the display.

The data in display memory can be accessed (displayed on the screen) in blocks that are known as "pages". A page consists of 46 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 46 lines in display memory. The **next page** is the succeeding 46 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.

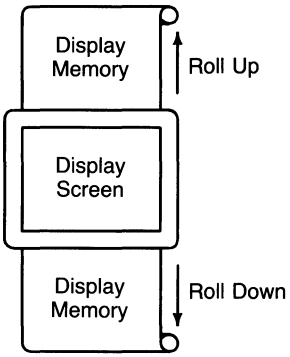


Figure 1. The "Roll" Data Functions

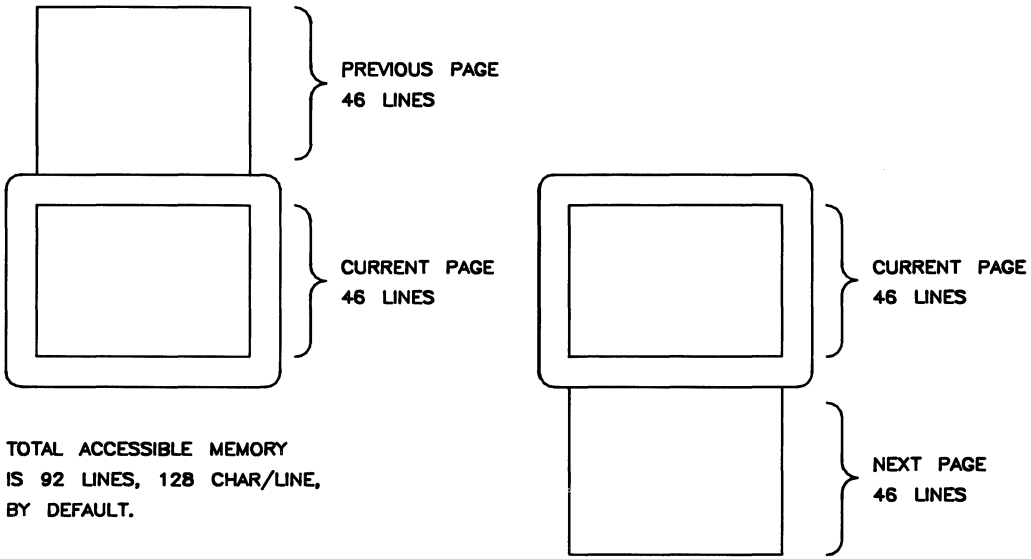


Figure 2: Previous Page and Next Page Concepts

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 edit keys and escape sequences shown in Table 2 may be used.

Table 2. Edit Keys

Key	Escape Sequence	Function
Clear display	\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.
Insert line	\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.
Delete line	\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.
Delete char	\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.
Insert char	\EQ	Turn on the insert character mode (see the description that follows).
Insert char	\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 of the keyboard are 10 keys labeled f1 through f4, Menu, User, f5 through f8. The functions performed by the function keys (f1 through f8) change dynamically as you use the terminal but are basically determined by the Menu and User keys. At any given time the applicable function labels for these keys appear across the bottom of the display screen.

The following table gives the escape sequences to enable/disable some of the function keys and the user-defined keys:

Table 3. Function Key Enable/Disable

Escape Sequence	Function
\E & j @	Turn off labels
\E & j A	Turn on MODES labels
\E & j B	Turn on user-defined labels
\E & j R	enable Menu, System, and User keys
\E & j S	disable Menu, System, and User keys

SYSTEM

The "SYSTEM" key accesses general terminal control functions. When you press the "SYSTEM" key the eight function keys are redefined to two functional keys: "margins/tabs" (f2) and "modes" (f4). Lower case letters indicate that further menus are available; upper case letters describe functions to be performed. Pressing one of these 2 keys causes another menu to be displayed as described below. The other 6 function keys have no effect.

margins/tabs - When the "MARGINS/TABS" function 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 later.

Note that the "MARGINS/TABS" function 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.

- 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 function keys or with the following 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 128.
  ```

  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

  Some of the eight function keys accessed with the "SYSTEM" key are used 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 these features are available only from softkeys (as are the margin functions described above.)

modes - When you press the "MODES" key [f4] 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):

[f1]	[f2]	[f3]	[f4]	[f5]	[f6]	[f7]	[f8]
			REMOTE MODE			DISPLAY FUNCT	AUTO 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 [Return]
- By simultaneously pressing the keys [CTRL] and [M]
- By pressing any of the user keys [f1] through [f8] (provided that a carriage-return code is included in the particular key definition). See section on "User" keys later in this article.

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 [Shift][▲], 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 he 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 char-
            acters are displayed
```

MENU

The "MENU" key is used to turn on and off the label display while in any label menu. Pressing the "MENU" key once will turn off the display, pressing it again will return the label display to the previous mode.

NOTE

While the label display is off the user key definitions are in effect regardless of the previous mode.

USER

When you press the USER key (Shift System) the eight function keys display the user defined key labels. In the section "User-definable Keys" the function of the user keys and the procedure for defining them is described.

User-definable Keys - The eight function keys f1 through f8, 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 (when the "USER" key is pressed), is displayed across the bottom of the screen.

The definition of each user key may contain up to 80 characters (alphanumeric character, 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 the fields are defined as follows:

`<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 keyboard (they are labeled "f1" through "f8").

`<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 though 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 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 `f1` through `f8`, respectively. These escape sequences have no special meaning to the terminal or to HP-UX.

Other Keys

There are a few keys that do not fall into any particular functional key group. They are the "RESET", "STOP", and "PRINT" keys.

RESET

After running graphics programs or exiting HP Windows/9000, the HP 98700H may be left in an incorrect state for the ITE to run. The correct state may be restored by pressing the "RESET" key (`Shift` `Break`). This will reset the color map (see section on "Other Features" later in this article), control of the display, and keyboard auto-repeat delay and auto-repeat period to the normal ITE state. The contents of the display memory are *not* changed. This key is also useful if the HP 98700H has been powered down and you wish to make it useable again. Note that the HP 98700H *must* have been originally powered up with or before the series 500 in order to be used at all.

PRINT

The "PRINT" key (Shift Enter) is useful with the "RESET" key or by itself. It causes the entire contents of the screen to be regenerated, along with the function-key labels, to the state last known by the ITE. Nothing else is changed. This key is useful if the screen (frame buffer) has been cleared or changed by another process for some reason.

STOP

Pressing the "STOP" key will suspend output to the screen until it is pressed again. Note that this key performs the same function as the CTRL S (suspend output) and CTRL Q (resume output) sequence, but no ASCII codes are generated. It may be used along with "CTRL-S" and "CTRL-Q". For example, you may stop the output with CTRL S and resume it with the "STOP" key.

The Display

The "terminal's" display has may features of its own, such as video highlights (inverse video), raster control, cursor sensing and addressing, and color highlight control. 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 128 columns (0 - 127) and any number of 46 line pages (specified by the HP-UX configuration). As shipped to you, the display memory has 92 lines (0 - 91) of 128 characters (2 screens). The amount of display memory can be determined from byte 0 of the primary terminal status (discussed in the section entitled "Sending Terminal Status", later in this article). The types of addressing available are absolute (memory relative), screen relative, and cursor relative.

Row Addressing

Figure 3 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
```

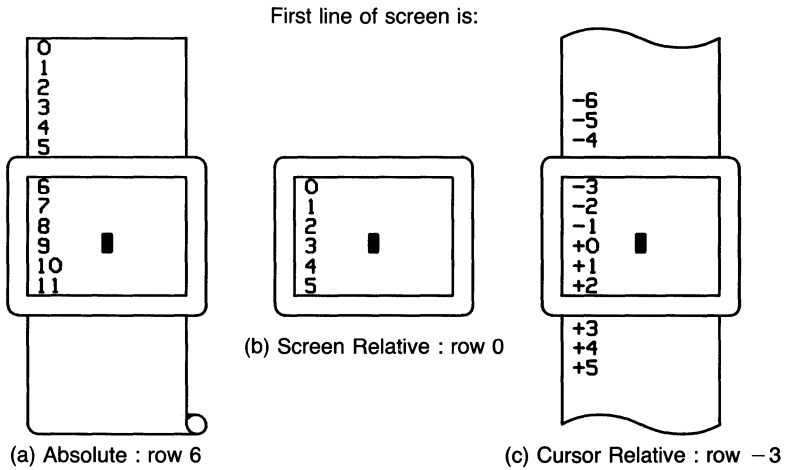


Figure 3. Row Addressing

Column Addressing

Column addressing is accomplished in a manner similar to row addressing. There is no difference between screen and cursor relative column addressing. Figure 4 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 columns 0 and 127, while the relative column addressing example shows limits of -5 and 122. The cursor cannot be wrapped around from column 0 to 127 by specifying large negative values for relative column positions.

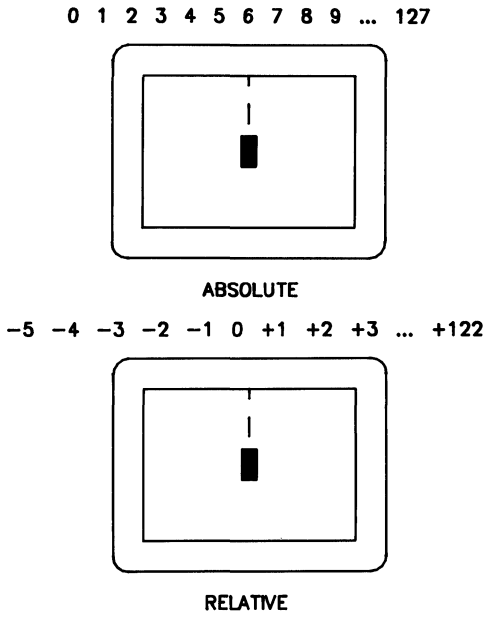


Figure 4. Column Addressing

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 the fields are defined as follows:

<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 - 45) to which you wsh to move the cursor. Zero specifies the top row of the screen; 45 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 46 rows and 128 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 the fields are defined as follows:

<column number> is a decimal number (0 - 127) specifying the column coordiate (within display memory) of the character at which you want the cursor positioned. Zero specifies the first (leftmost) column in display memory, 127 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:

```
[46 (lines/page) × num_page (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 91.

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 bondaries 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.

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 the fields are defined as follows:

<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.

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 follows 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 - foreground and background colors are exchanged (see below).
- Underline Video - Characters are underscored.

NOTE

The half bright and blinking display enhancements are not implemented on this terminal. Selecting the half bright enhancement on the HP 98700H causes the terminal to select pen 3 (See the section "Accessing Color" later in this article). Selecting the blinking enhancement has no effect.

The display enhancements are used on a field basis. The field cannot span more than one line. The field scrolls wth 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.

Table 4. 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.

Accessing Color

Color pair -
two colors which define the foreground color (color of the characters) and the background color, respectively. 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 in Table 5 (re-defining a color pair is described later).

Table 5. Color Pairs

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	black

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. Table 6 gives the values needed to define the eight principle colors.

Table 6. Eight Principle Color Values

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), an n is one of the legal values from the table above. <pen#> is an integer in the range 0 - 7 which, when combined wth 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 5I
```

This example re-defines pen 5 to specify red characters on a green background. This example is equivalent to:

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

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 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 backgrond (RGB notation). This example illustrates how multiple pens can be defined on a single line using different notations.

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

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.

```
\E&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. This is like typing on a typewriter without paper or ribbon; you can't see what is written.

Additional Notes About the ITE and the HP 98700H

The following section contains information that is essential if you are using certain features of HP-UX, but is not necessary reading for using the HP 98700H as a "normal" terminal.

Color Map

Features of the ITE deal mainly with lower level hardware control and interaction with the Starbase graphics system. A brief explanation of the basic operation of the HP 98700H will help clarify these features.

The HP 98700H is a color-mapped system of up to eight planes. It has internal memory, called the frame buffer, which directly controls what is displayed on the color monitor. Each pixel (the smallest displayable unit) on the display monitor is controlled by one byte of the frame buffer memory. The eight planes correspond to the width (8 bits or 1 byte) of the frame buffer.

The color map takes each byte of the frame buffer and from it generates a combination of red, green, and blue and sends it to the monitor. Since 1 byte may take on 256 different values, there are 256 different colors that may be visible on the screen at any one time. Each pixel may be assigned any one of these 256 (including black) colors. Any one of over 16 million colors may be generated by the color map, but only 256 are available at any one time.

The ITE uses only 3 of the 8 planes available since only 8 (2^3) colors need to be available for the "terminal". The other planes are disabled and turned off by the ITE; the information in them is not displayed and remains unchanged while the ITE is running.

Screen Windowing

The ITE may be configured to use only the lower part of the display monitor screen and not touch the rest. This might be useful when interacting with graphics programs. The escape sequence to accomplish this is:

```
\E&w2f <starting screen line> U
```

where `<starting screen line>` is the line number (0 through 45) of the screen where you want the display to start. If no number is entered, 0 is used as the default, giving a full-screen display. The width of the displayed area is not changeable; it is always 128 characters. Note that the display memory size is not changed by this escape sequence; only the effective page (display screen) size is changed. The next-page and previous-page keys will work based on this new page size. If a number outside the range of 0 to 45 is entered, the entire escape sequence is ignored.

Interaction with Starbase and HP Windows/9000

While running HP Windows/9000 the ITE is automatically disabled from receiving any keystrokes. After exiting HP Windows/9000 or running graphics programs, the "RESET" and "PRINT" keys may be needed to restore correct operation to the ITE.

Note that a limited interaction other than screen windowing may be safely accomplished with the ITE and the Starbase graphics system. Since the ITE only uses 3 out of the possible 8 planes of the frame buffer it is possible for alpha and graphics information to coexist independently on the screen. This requires that the color map be set up with some care. See the *Starbase Device Driver Library, PN98592-90010* and *HP-UX Reference, Vol.4, PN09000-90008*, commands *inquire_color_table, define_color_table* and *inquire_sizes* for further information on setting up the color map.

Multiple Keyboards

Multiple keyboards, or HIL devices which look like keyboards, may be connected to the HP-HIL and run with the ITE. The data from the keyboards is logically "or"ed together; there is no way to distinguish between different keyboards. The effective language of the keyboard set is the language of the first keyboard on the loop when the model 550 was powered up.

CAUTION

Opening driver 43 (the HP 98700H HP-HIL cooked keyboard driver) directly from a user program is dangerous. It will "steal" keystrokes from the ITE. This could cause an inability to exit from the program, locking your session up.

Native Language Support

The ITE currently has limited support for native languages.

Character Sets

Output of the complete ROMAN8 character set is supported.

The Katakana (KANA8) character set is not available.

The extended character set (accessed by the "EXTEND CHAR" key) is not available.

Native Language Keyboards

All of the local language keyboards are supported by the ITE with the exception of KATAKANA which is supported only in ROMAN mode. Muting, however, is not available. Typing an umlaut followed by an "a", for example, will produce the two separate characters instead of an "umlaut a" character.

Other

Note also that the ITE is always in 8-bit mode; 7-bit substitution is never done. This means that 8 bits of character code data are always generated. The *istrip* option of *stty(1)* may make it appear that only 7-bit data is being generated.

The keyboard connected to the system self-identifies itself as being a particular language. There is no way to change the effective language of a keyboard from the ITE.

Since the extended character set is not supported, a method of accessing the ASCII characters which are often substituted on local language keyboards is needed. This is done by shifting the keypad keys as shown in Table 7.

Table 7. Extended ASCII Characters

KEYS SHIFTED	KEYS NORMAL
|	*
\	/
,	+
‘	-
[	7
]	8
{	9
}	ENTER
^	4
#	5
~	6
@	,
<	1
>	2

These shifted keypad characters are available on all non-USASCII, non-Katakana keyboards.

Setting up the HP 98700H as a Terminal

To set up the HP 98700H as a terminal you must make a device file using *mknod* and modify *inittab.*

To make a device file use the command:

```
mknod /dev/ite1 c 29 0xff0100
```

where */dev/ite1* is the device file ,*c* is the *mknod* option designating */dev/ite1* as a character file, *29* is the driver file (ITE is driver #29), and the 01 portion of the hex number corresponds to the slot number of the graphics display buffer card (*HP 98288A*). The graphics display buffer card can be installed in any of slots 04 - 07. The number used in the *mknod* command is the slot# - 4 (00 - 03). Installation of this board *must* be performed by a Hewlett-Packard engineer or a Hewlett-Packard trained Customer Engineer. The engineer should refer to *HP9050 Hardware Support Document, PN 09050-90038.*

Modify *inittab* as described in the HP-UX Reference section 5 (*inittab(5)*).

Controlling Configuration and Status

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

Reconfiguring the Terminal

The terminal allows you to reset a few of its configuration parameters via escape sequences. These parameters and their escape sequences are listed in Table 8.

Table 8. Reconfiguration Escape Sequences

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 = 0.
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 = 0.
Remote Mode	\E&k nR	When n is 0, the remote mode is off. When n is 1, the remote mode is on. Default = 1.
Caps Mode	\E&k nP	When caps mode is enabled, all unshifted alphabetic keys generate uppercase letters and all shifted alphabetic keys generate lowercase lettes. 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 = 0. 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 executed both at the terminal and transmitted to HP-UX. When x = 1, the escape code sequences generated by control keys such as Shift ▲ and Shift ▼ 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: x = 0

Sending Terminal Status

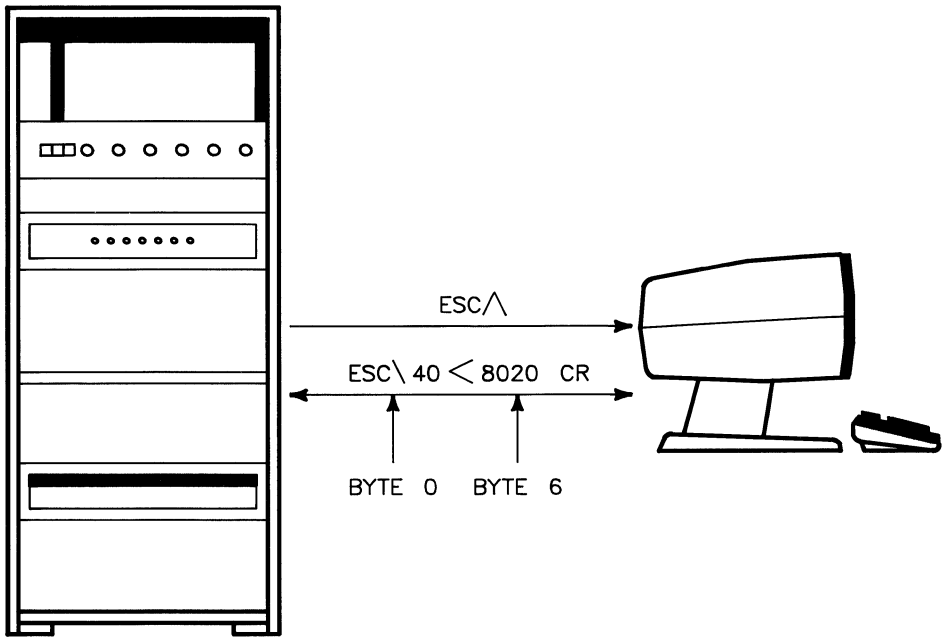
Terminal status is made up of 14 status bytes (bytes 0 through 13) containing information such as display memory size, switch settings, configuration menu settings, and terminal errors. There are two terminal status requests: primary and secondary. Each returns a set of seven status bytes.

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 Figure 5.



BYTE	ASCII	BINARY	STATUS
0	4	0011 0100	8K bytes of display memory (2 pages of 46 lines)
1	0	0011 0000	Function key transmission disabled
			Cursor wraparound disabled
2	<	0011 1100	
3	8	0011 1000	Auto line feed disabled
			Terminal sends secondary status
4	0	0011 0000	
5	2	0011 0010	Last Self-Test ok
6	0	0011 0000	

Configuration Straps A-H

Figure 5. Primary Terminal Status Example

BYTE 0 DISPLAY MEMORY SIZE

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

1K bytes
2K bytes
4K bytes
8K bytes

This byte specifies the amount of display memory available in the terminal.
(roughly 4K per page)

BYTE 1 CONFIGURATION STRAPS A-D

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

Strap D
Page/Line
always 0

Strap C
(Inhibit End-of-Line Wraparound)
1 = yes (Enabled)
0 = no (Disabled)

Strap A
(Function Key Transmission)
1 = yes (Transmitted)
0 = no (Not transmitted)

Strap B
(Space Overwrite Latch)
always 0

BYTE 2 CONFIGURATION STRAPS E-H

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

Strap H
always 1

Strap G
always 1

Strap E
always 0

Strap F
always 0

BYTE 3 LATCHING KEYS

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

Terminal sends
secondary status

AUTO LF Key
1 = auto LF
0 = no auto LF

CAPS LOCK Key
always 0

BLOCK MODE Key
1 = block mode
0 = character mode

BYTE 4 TRANSFER PENDING FLAGS

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

Secondary Status Pending
always 0

ENTER Key Pending
always 0

Cursor Sense Pending
always 0

Function Key Pending
always 0

BYTE 5 ERROR FLAGS

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

Device Error
(Integral Printer Error)
always 0

DataComm
always 0

Self-Test
always 1

BYTE 6 DEVICE TRANSFER PENDING FLAGS

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

Device Status Pending
always 0

Device Operation Status
Pending
always 0

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

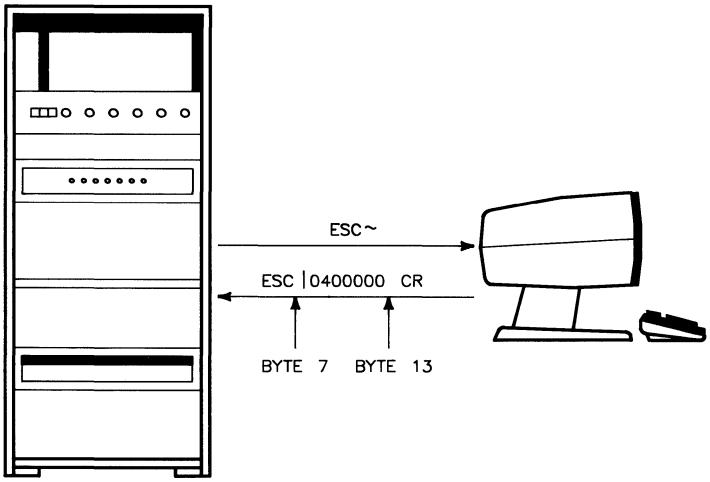
Figure 6. Primary Status Bytes

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 Figure 7.



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	

Figure 7. Secondary Terminal Status Example

BYTE 7 Buffer Memory (always zero)

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

- 1K bytes
- 2K bytes
- 4K bytes
- 8K bytes

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

- 0 = Non-programmable terminal
- 1 = Terminal identifies self
- always 0
- 1 = I/O firmware installed, integral printer present
- 0 = No APL Firmfare

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

- Strap M
- Strap J (Auto Terminate)
  - 1 = yes (Enabled)
  - 0 = no (Disabled)
- Strap K (Clear Terminator)
  - 1 = yes (Enabled)
  - 0 = no (Disabled)
- Strap L (Self-Test Inhibit)
  - 1 = yes (Inhibit test)
  - 0 = no (Allow test)

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

- Switch R always 0
- Switch N Printer (Escape Code Transfer) always 0
- Switch P Compatibility Mode (Scaled) always 0
- Switch Q Compatibility Mode (Unscaled) always 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

- Strap V
- Strap S
- Strap U
- Strap T

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

- Strap Z
- Strap W (Data Comm Test)
  - 1 = yes (Inhibit)
  - 0 = no (Allow)
- Strap Y
- Strap X

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

Figure 8. Secondary Status Bytes

Notes

Table of Contents

Mailx: Mail Handler

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 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
```

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 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
```

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 seelect 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 **DEL**s (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.

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
```

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 *rmsgs* to *rmsgs −s* type:

```
alias rmsgs "! rmsgs -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.

copy (abbr: *co*) The *copy* command is identical to *save* except that copied messages are not marked for deletion when you quit.

delete (abbr: *d*) Deletes a list of messages. Deleted messages can be reclaimed with the *undelete* command.

dp or *dt* *dpt* or *dt* deletes the current message and prints the next message. It is useful for quickly reading and disposing of mail. Displays `At EOF` if no messages remaining.

edit (abbr: *e*) The *edit* command provides editing capabilities for individual messages. It takes a list of messages as described under the *type* command and processes each by writing the message into a file *message*<x> (where <x> 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. *Mailx* then reads the message back and removes the file. *Edit can be abbreviated to e.*

else Marks the end of the **then** part of an *if* statement and the beginning of the part to take effect if the condition of the *if* statement is false.

endif Marks the end of an *if* statement.

exit (abbr: *ex* or *x*) Leaves *mailx* without updating the system mailbox or the file your were reading. Thus, if you accidentally delete several messages, you can use *exit* to avoid scrambling your mailbox.

file (abbr: *fi*) Identical to *folder.*

folders List the names of the folders in your folder directory.

folder (abbr: *fo*) The *folder* 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>	*Name*'s system mailbox
&	Your *~/mbox* 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 your 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 **DEL**s (or **RUBOUT**s), *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*.

alias	An alternative name for a person or list of people.
flag	An option, given on the command line of *mailx*, prefaced with a −. For example, *−f* is a flag.
header field	At the beginning of a message, a line containing information that is part of the structure of the message. Popular header fields include `to`, `cc`, and `subject`.
mail	A collection of messages. Often used as in the phrase, "Have you read your mail?"
mailbox	The place where your mail is stored, typically in the directory */usr/mail*.
message	A single letter from someone, initially stored in your *mailbox*.
message list	A string used in *mailx* command mode to describe a sequence of messages.
option	A piece of special purpose information used to tailor *mailx* to your taste. Options are specified with the *set* 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
!	Single command escape to shell
-	Back up to previous message
Print	(**P**) Type message with ignored fields
Reply	(**R**) Reply to author of message only
Type	(**T**) Type message with ignored fields
alias	(**a**) Define an alias as a set of user names
alternates	(**alt**) List other names you are known by
chdir	(**cd**) Change working directory, home by default
copy	(**co**) Copy a message to a file or folder
delete	(**d**) Delete a list of messages
dp or *dt*	Delete current message, type next message
endif	End of conditional statement; see *if*
edit	(**e**) Edit a list of messages
else	Start of else part of conditional; see *if*
exit	(**ex** or **x**) Leave mail without changing anything
file	(**fi**) Interrogate/change current mail file
folder	(**fo**) Same as *file*
folders	List the folders in your folder directory
from	(**f**) List headers of a list of messages
headers	(**h**) List current window of messages
help	Print brief summary of *mailx* commands
hold	(**ho**) Same as *preserve*
if	Conditional execution of *mailx* commands
ignore	Set/examine list of ignored header fields
list	List valid *mailx* commands
local	List other names for the local host
mailx	(**m**) Send mail to specified names
mbox	Arrange to save a list of messages in *mbox*
next	(**n**, **+**, or **RETURN**) Go to next message and type it
preserve	Arrange to leave list of messages in system mailbox
print	Print specified messages without *ignored* headers
quit	(**q**) Leave *mailx*; update system mailbox and *mbox* as appropriate
reply	(**r**) Compose a reply to a message

Command	Description
save	(**s**) Append messages, headers included, on a file
set	(**se**) Set binary or valued options
shell	(**sh**) Invoke an interactive shell
source	(**so**) Reads *mailx* commands from a file.
top	Print first so many (5 by default) lines of list of messages
type	(**t**) Print messages
unalias	Remove one or more *alias* groups
undelete	(**u**) Undelete list of messages
unset	Undo the operation of a *set*
visual	(**v**) Invoke visual editor on a list of messages
write	(**w**) Append messages to a file, don't include headers
xit	(**x**) Synonym for *exit*
z	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 ~e and *edit*
SHELL	valued	Pathname of shell for *shell*, ~!, and *!*
VISUAL	valued	Pathname of screen editor for ~*v* and *visual*
append	binary	Always append messages to end of *mbox*
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 *delete*
crt	valued	Minimum number of lines before using *more*
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 ~
folder	valued	Directory to store folders in
hold	binary	Hold messages in system mailbox by default
ignore	binary	Ignore **DEL** (or **RUBOUT**) 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 *saved* 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 *dead.letter*
quiet	binary	Suppress printing of *mailx* version and message numbers
record	valued	File to save all outgoing mail in
screen	valued	Size of window of message headers for *z*, etc.
sendmail	valued	Choose alternate mail delivery system
toplines	valued	Number of lines to print in *top*
verbose	binary	Invoke sendmail with the *-v* 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 *dead.letter* 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 **DEL** (or **RUBOUT**)
~r	filename	Read file into message
~s	string	Set Subject: field to *string*
~t	name ...	Add names to To: field
~v		Invoke screen editor on message
~w	filename	Write message on file
~	command	Pipe message through *command*
~~	string	Quote a ~ in front of *string*

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 */usr/lib/Mail.rc*
−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.

Table of Contents

Overview 1

This manual covers both the asynchronous terminal emulator program and the *uucp* facility programs.

The asynchronous terminal emulator (*aterm*) program enables your HP 9000 Series 500 to emulate a terminal on another computer system. Your computer must use the HP-UX operating system; the operating system for the terminal you emulate is **not** restricted. After specifying communications parameters, the emulator appears as any other HP-UX command. Once the emulator is invoked with these parameters, you can use your terminal as though it were a terminal on the other system.

NOTE

The *aterm* command is **only** available on Series 500 computers. In this manual, you will find section headings which have information relevant to the *aterm* command followed by (**aterm**) in parentheses. If you **are not** using *aterm* then you may skip that section of the manual. For example,

Asynchronous Terminal Emulator (aterm)

The HP-UX *uucp* facility is a set of programs which exchange information between UNIX[1] and non-UNIX systems. Note that the program *cu* which is part of the *uucp* facility allows you to exchange information with non-UNIX systems. *Uucp* programs can be used to transfer files and commands to and from a remote system, to transfer files from a remote system to another remote system and to send and receive mail. You can also forward mail and files through intermediate nodes. All systems involved must use an HP-UX or UNIX operating system, be on the *uucp* network and have the *uucp* facility installed. The network consists of workstations connected with either direct or modem connections. The *uucp* facility is easy to use, fast, reliable and cost-effective.

[1] UNIX is a trademark of AT&T Bell Laboratories.

Overview of Programs

The HP-UX asynchronous terminal emulator program (*aterm*) enables your computer to become an interactive terminal emulating a terminal on any other computer system. The emulator allows you to tailor the communications format, including such parameters as baud rate, word length and parity. The emulator also provides file transfer capability to and from other computers. This program runs **only** on the Series 500 computers.

There are three main programs in the *uucp* facility: *cu*, *uucp* and *uux*, as well as many auxiliary programs: *uuclean*, *uulog*, *uuname*, *uupick*, *uustat*, *uusub*, and *uuto*. The *uucp* and *uux* programs operate in the background mode leaving your terminal free for other uses. Both your local system and the remote system must use an HP-UX or UNIX operating system except when you are using *cu* which allows you to use a non-UNIX operating system. The rest of this section is a brief overview of the three main *uucp* programs.

Cu is an acronym for *call UNIX*. With the *cu* program you can interactively log onto any other UNIX or non-UNIX system. *Cu* provides a way for you to check your communications link and transfer ASCII files, but implements no error checking.

Uucp is an acronym for *UNIX-to-UNIX copy program.* With the *uucp* program you can have the source_file and/or the destination_file reside on remote systems. To specify remote source or destination files you simply include the remote system name with the file name.

The information necessary to contact the remote system as well as the security access information is kept in a set of files on both systems. Once you have placed this information in the appropriate files, the *uucp* facility can automatically establish the remote connection and protect your data files from unauthorized use.

All data transferred is checked for errors and re-transmitted should an error occur. This makes the *uucp* facility a reliable method of information exchange.

The *uux* command is the acronym for *UNIX-to-UNIX execution.* With *uux* you can only execute those commands which the remote system gives you permission to execute. The remote system has a list of these commands in its *L.cmds* file.

Although the *mail* command is a local HP-UX command, mail can also be used with the *uucp* facility. You can send mail to remote systems or forward mail through several remote systems to the final destination system.

Manual Overview

This manual contains ten chapters covering these areas:

- an introduction and overview;
- hardware configuration;
- software configuration;
- using the asynchronous terminal emulator program;
- the uucp file system;
- the uucp demons;
- using the uucp commands;
- the X.25 Network;
- uucp log, status, mail and clean-up information;
- possible uucp problems and solutions.

Chapter 1: Overview

This chapter gives an overview of the programs covered in this manual, as well as a brief description of each chapter included in this manual. You are also provided with directions on how to use this manual as a system administrator or system user.

Chapter 2: Hardware Configuration

This chapter describes the hardware installation steps which must be taken by the System Administrator to configure the Series 500 computer for the asynchronous terminal emulator program (*aterm*) and for the *uucp* facility programs. It also includes installation steps which must be taken by the system administrator to configure the Series 200 computer for the *uucp* facility programs.

Chapter 3: Software Configuration

This chapter outlines the software configuration tasks the system administrator needs to perform after the hardware is installed:

- setting up your device and configuration files for the asynchronous terminal emulator program on the Series 500 computer;
- setting up the uucp software configuration.

Chapter 4: Asynchronous Terminal Emulator (aterm)

This chapter discusses emulator modes, handshaking, local commands and provides many examples of using the asynchronous terminal emulator on your Series 500 computers.

Chapter 5: Uucp File System

This chapter describes the files which are used by *uucp* facilities to implement remote communication.

Some of these files are created automatically in the */usr/spool/uucp* directory by the *uucp* programs as they carry out the transfer of information. If you look in this directory you can see the status of the transfer by looking at these files. Understanding the file structure also helps in case you have a problem in transferring data.

The system administrator must edit many of these files (Software Configuration chapter) to specify:

- how your Series 200/500 can contact each remote system;
- how, when and if systems on the uucp network can contact you;
- access permission restrictions for files and commands.

Chapter 6: Uucp Demons

The *uucp* demons are the programs that do the work of the *uucp* facility. These demons are automatically invoked when the *uucp* or *uux* programs are operating in the background mode. You can also invoke them interactively.

Chapter 7: Using the Uucp Commands

This chapter illustrates the use of the *uucp* facility programs to:

- send files and commands to a remote system;
- receive files and commands from a remote system;
- send and receive mail;
- monitor status, log and access information;
- clean up old or unwanted files.

Chapter 8: The X.25 Network

This chapter provides you with a brief discussion on what the X.25 Network is and it explains how to set up *uucp* for X.25 communications.

Chapter 9: Log, Status and Cleanup Information

This chapter discusses how the system logs information about each transaction, how you can check on job or system status and how you can clean up old or unwanted files.

Chapter 10: Problems and Solutions

The most frequently encountered problems, their solutions and debugging information are presented in this chapter.

Where Do You Start?

If You Are a User

If your System Administrator has set up the hardware and software configurations on your Series 200 or 500 computer:

- for the asynchronous terminal emulator (Series 500 *only*) go to the chapter, "Asynchronous Terminal Emulator";
- for the *uucp* facility go to the chapter, "Using the uucp Commands".

If You Are the System Administrator

Aterm (aterm)

If you are the Series 500 System Administrator you need to perform the asynchronous terminal emulator (*aterm* program) tasks listed in the chapters:

- Hardware Configuration;
- Software Configuration.

When these configurations are set up, the system can execute the emulator program as described in the chapter, "Asynchronous Terminal Emulator".

Uucp Facility

If you are the Series 200/500 System Administrator for the *uucp* facility, the first thing you should do to get started is to contact each remote system you want to communicate with and obtain the following information:

- the remote node name;
- whether this will be a direct (hardwired) or a modem (telephone) connection;
- the times the remote system will permit communications;
- the remote system telephone number;
- the data rate;
- the remote system login name and password if any.

You need to incorporate this information into your files to establish connection and protection specifications.

The next tasks you need to perform are to set up your hardware and software configurations as described in the chapters, "Hardware Configuration", and "Software Configuration".

At this point your system can respond to the *uucp* facility commands. The features described in the chapter, "Status, Log and Cleanup Information" should be started as soon as possible to monitor the activities of *uucp* and to keep your file storage area free of old or unwanted data. These features should be periodically implemented.

Read the complete manual for a better understanding of the asynchronous terminal emulator (*aterm*) and the *uucp* facility processes. You should be familiar with the HP-UX file and command system. The *HP-UX Concepts and Tutorials* volumes 1 through 6, *HP-UX Reference* manual, and the HP-UX Series 200/500 *System Administrator Manual*, provide more information about the HP-UX system.

Additional Networks

This section is intended to introduce you to HP networks other than the ones found in this manual. It provides a brief explanation of each network and gives references to documentation for these networks. The additional networks are as follows:

- RJE
- LAN
- HP AdvanceNet

The RJE Network

The RJE Emulator package enables your HP 9000 HP-UX workstation to communicate with remote computers and peripherals that support IBM 2780/3780 Remote Job Entry (RJE) data transmission protocols. The emulator can also communicate with other IBM 2780/3780 compatible devices for file transfer.

The supported features of the RJE Emulator include:

- Binary synchronous communications with EBCDIC transmission codes,
- Data transmission at up to 19 200 bits per second,
- Space compression and expansion in 3780 mode only, thereby raising the effective throughput rate,
- Transparent mode, which allows all possible EBCDIC combinations to be used as data,
- Full or half-duplex operation,
- Programmable modem timeout value,
- RJE/send subsystem adapted from System III UNIX MRJE,
- Tracing of data,
- Print formatting utility using IBM print conventions,
- Auto answer or manual originate,
- MSV2 protocol.

The following IBM 2780/3780 features are not supported:

- Interactive mode (Instead of ACK, a message is transmitted.),
- Multipoint transmission,
- 6-bit transcode,
- Bell messages,
- Hardwired connections.

For more information on RJE networking, read the *RJE User's Guide.*

The Local Area Network

A Local Area Network (LAN) is a way of connecting multiple systems together in a limited geographical area. For example, a local area network can consist of systems attached to a single length of cable. Or, a Local Area Network can be formed by connecting a central system to each of the other systems in the building. Cabling of the Private Branch Exchange (PBX) telephone systems can also be used to interconnect devices within a site. PBX cabling forms a unique data transmission network unlike other LAN configurations; therefore, references to LANs in this manual do not include PBX LANs unless directly stated.

A LAN is not simply a connection of hardware; software controls the interaction and transmission of data between systems on the network. There are many different kinds of Local Area Networks; however, there are a few main characteristics they all share. These characteristics are:

- Limited geographic coverage,
- Single organization ownership,
- High data rate.

For more information on the LAN network, read the *NS/9000, LAN User's Guide* and the *NS/9000, LAN Node Manager's Guide.*

The HP AdvanceNet

HP AdvanceNet is an HP networking strategy which offers the following:

- Size Alternatives and growth paths,
- Easy-to-use network,
- Compatibility with:

  Industry standards,

  De facto standards.

Because HP AdvanceNet is based on industry standards, it is able to provide communication between HP computers and other vendors' computer products which are also standards based. Links currently defined up through Level 3 of the Open System Interconnection (OSI) model provide a common protocol for simplified communication in a multi-vendor environment.

HP AdvanceNet provides system-to-system communication capability within or between HP product lines, such as the HP 1000, HP 3000, and HP 9000 multi-user computer systems. These communication capabilities range from user-level services (for example, virtual terminal and file transfer) to physical links (such as point-to-point, PBX, X.25, and satellite).

For more information on HP AdvanceNet, read portions of the LAN documentation mentioned in the previous section which include HP AdvanceNet. Also read information on HP AdvanceNet found in these manuals:

- *NS/1000 User/Programmer Reference Manual,*
- *NS/3000 User/Programmer Reference Manual.*

Hardware Configuration 2

This chapter discusses the steps which must be taken to configure the hardware for the asynchronous terminal emulator program and the uucp facility programs.

Asynchronous Terminal Emulator HP 9000 Series 500 (aterm)

To execute the asynchronous terminal emulator on your Series 500 computer, you need the HP 27128A Asynchronous Serial Interface (ASI) card. This card should be installed according to the directions in the *ASI Installation* manual. Configuration switch settings on the ASI card are overridden by software. The asynchronous terminal emulator can also be used with an HP 27130A/B (8-channel multiplexer). Note that you **cannot** use an HP 27140A (6-channel multiplexer) with the asynchronous terminal emulator or any other I/O card not mentioned in this paragraph.

The male RS-232C connector cable is recommended for connecting to the host computer or modem. If you must connect to another male cable, use a modem eliminator cable such as HP 13232U. A female connector for the ASI card is also available.

Uucp Facility

Uucp communications always assume the presence of standard RS-232C modem signals.

Inserting the Series 500 ASI Card

The HP 27128A (ASI) card contains the necessary firmware for modem signaling. If you are using a modem connection, follow the directions in the installation manual shipped with the card.

Before you insert the ASI card:

- be sure the power to your computer is off;
- set switches two and eight of node address switches to the on (down) position.
- you must use the male cable (HP 27128A Opt. 001).

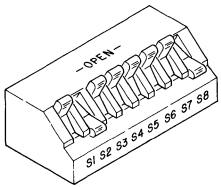


Figure 2-1. Switch Settings

Inserting Other Series 500 Cards

The HP 27130A/B interface supports up to 8 EIA RS-232C-compatible devices. It consists of an interface card and an RS-232C connection panel. It is the recommended interface for direct connection of terminals, providing slightly higher performance than the HP 27140A Modem MUX at a slightly lower per-port cost. The HP 27130A/B can also support terminal clusters up to 100m distant with a customer-fabricated cable. Note that the HP 27130A/B will not work for modem connections.

Before you insert the HP 27130A/B card:

- be sure that the power to your computer is off,
- read the *HP 9000 Series 500 Configuration Information and Order Guide* for card configuration, direct connection, and cabling information.

The HP 27140A interface supports up to 6 EIA RS-232C/CCITT-V.22- compatible devices. It consists of an interface card and an RS-232C connection panel. The HP 27140A is the recommended interface when one or more ports will be connected to a modem or directly connected to another HP 27140A interface.

Before you insert the HP 27140A card:

- be sure that the power to your computer is off,
- read the *HP 9000 Series 500 Configuration Information and Order Guide* for card configuration, direct connection, and cabling information.

HP-UX Series 200

The HP 98626A, HP 98644A, or HP 98628A interface card can be used to implement the *uucp* facilities with your Series 200 computer. Before you insert the interface card:

- turn the computer's power off;
- set all U3 switches on the HP 98626A interface card to 1. If you are using the HP 98628A interface card, you need to set two switches on the row of 8 switches labeled DEFAULTS on the interface card. The two switch settings for the HP 98628A interface card are as follows: set switch 4 to 1, and set switch 5 to 0.
- set switches 1, 4, and 8 to 1 and the remaining switches to 0 on the HP 98644A interface card. Also cut or remove the remote jumper on the HP 98644A interface card.

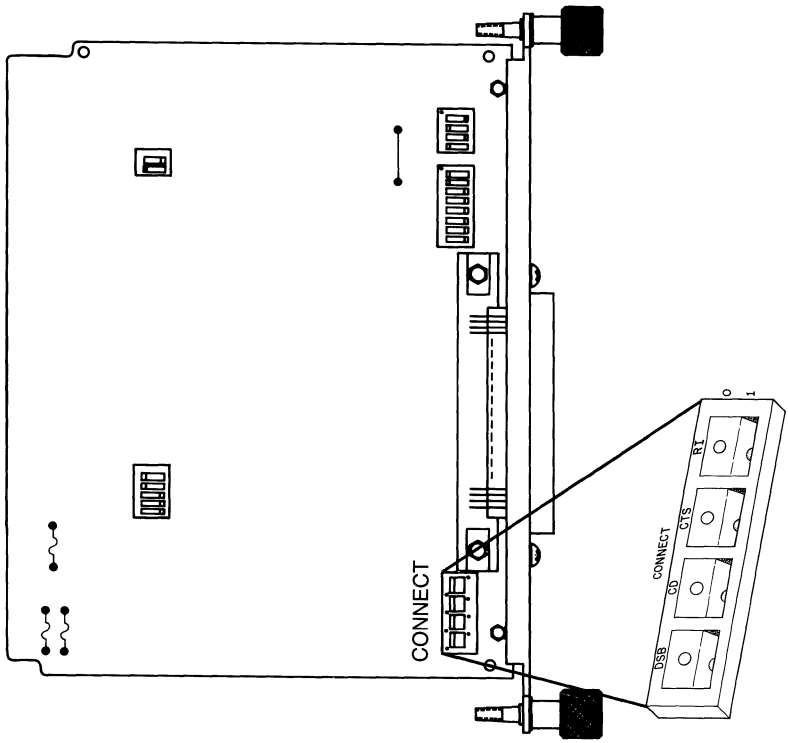


Figure 2-2. HP 98626/8 Switch Settings

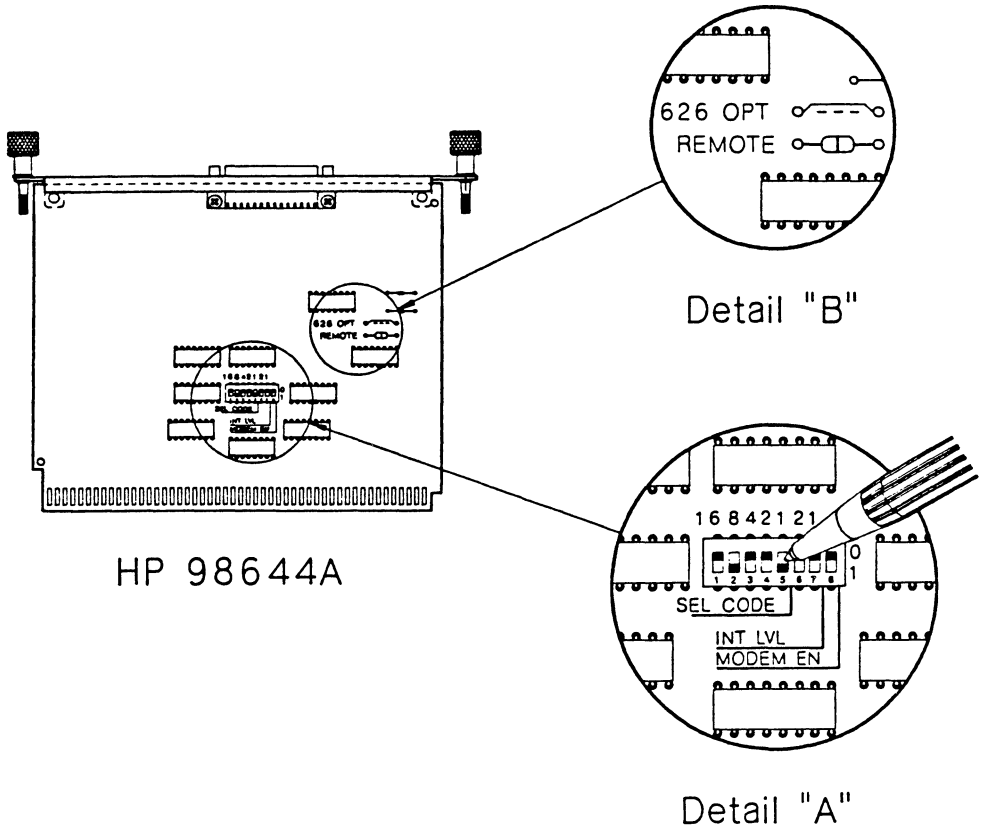


Figure 2-3. HP 98644 Switch Settings

Modem Connections

If the distance between your system and the remote system is more than about 15 meters (50 feet) or if noise on a direct connection line becomes a problem, a modem connection should be used.

Modem connections for the Series 500 computers require the HP 27128A Asynchronous Serial Interface (ASI) card whose firmware revision number is 27128-80005 or greater. Set switches 2 and 8 down for modem control. The HP 27140A (6-Channel Multiplexer) can also be used with the Series 500 for modem connection. There are no switch settings on this card which need to be made for modem use. For information on installing this card, read the installation manual for your HP 27140A interface.

Modem connections for the Series 200 computers require either the HP 98626A or HP 98644A serial interface card or the HP 98628A datacomm interface card. Set all three switches to the "CONNECT" position. The HP 98642A (4-Channel Multiplexer) can also be used with the Series 200 for modem connection. There are no switch settings on this card which need to be made for modem use. For information on installing this card, read the installation manual for your HP 98642A card.

For Series 500 computers, use a modem cable (HP 27128A Opt. 001) to connect the ASI card to the modem and use the HP 92219Q cable for connecting an HP 27140A card to a modem. For Series 200 computers, use a modem cable with male end (HP 98626/28A Opt. 001) to connect either the serial or datacomm interface card. Note that you **cannot** use an interface card with a DCE (female) end since this does not carry through modem signals.

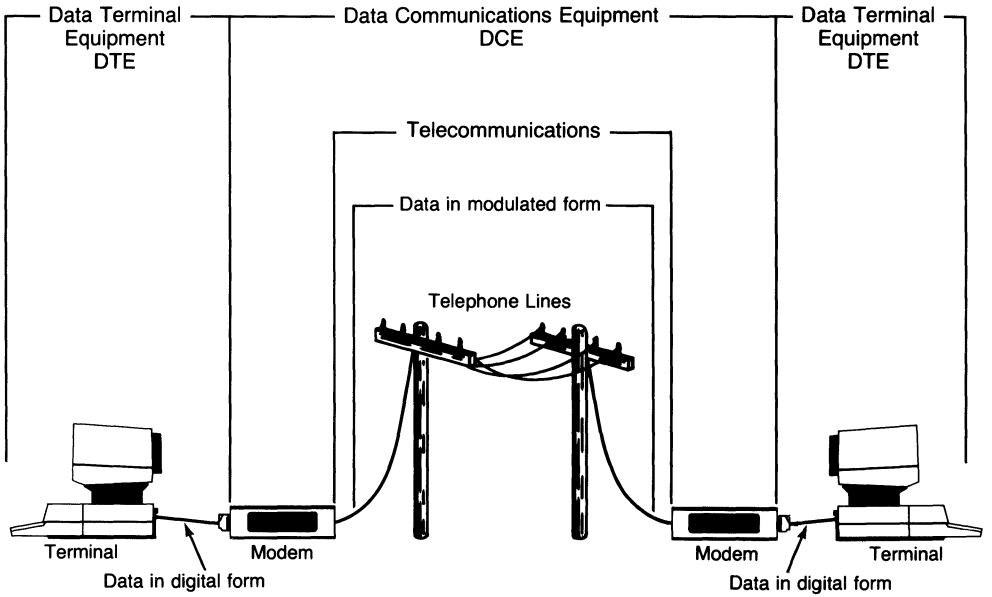


Figure 2-4. Modem Network

Modem connections for Series 200 and 500 computers may use the same line for both incoming and outgoing calls. No special modification is necessary; the DTE (male) cable end can be connected to the modem. When you use the *mknod* command to associate a special (device) file with the interface card on a specified select code, a flagging mechanism assigns the line as either an incoming or outgoing port. Refer to the section, "Creating a Device File", in the "Software Configuration" chapter of this manual for more information.

Many companies sell modem devices which are compatible with the *uucp* facility. Contact your nearest HP Sales Office for further information.

Direct Connection

Direct connections can be made between two interface cards, two multiplexers, or a multiplexer and an interface card. These direct connections also apply to connecting an interface card or multiplexer of a Series 200 computer to an interface card or multiplexer of a Series 500 computer. This section provides examples for each of these connections. Note that two DTE (male) cable ends cannot be connected together. In some case, you **must** connect each DTE cable end to a DCE RS-232C connector and modify the pin connections. Refer to the diagrams provide in this section for your particular direct connection configuration. Note that the "special connector" mentioned in this section can be designed using the instructions found in the last section of this chapter.

The following diagrams show the direct connections between two systems with exploded views to show the necessary wiring modifications:

- This is an example of directly connecting a Series 200 computer with either an HP 98626A or HP 98628A interface card to a Series 500 computer with an HP 27130A/B (8-channel multiplexer).

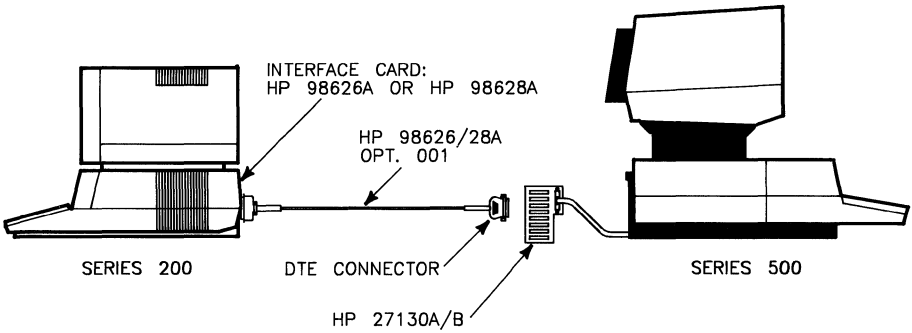


Figure 2-5. Series 200 to Series 500 MUX

- This is an example of directly connecting a Series 200 computer with either an HP 98626A or HP 98628A interface card to a Series 500 computer with an ASI (HP 27128A) card. The first figure shows a direct connection for communication in one direction, and the second figure shows a direct connection for communication in both directions.

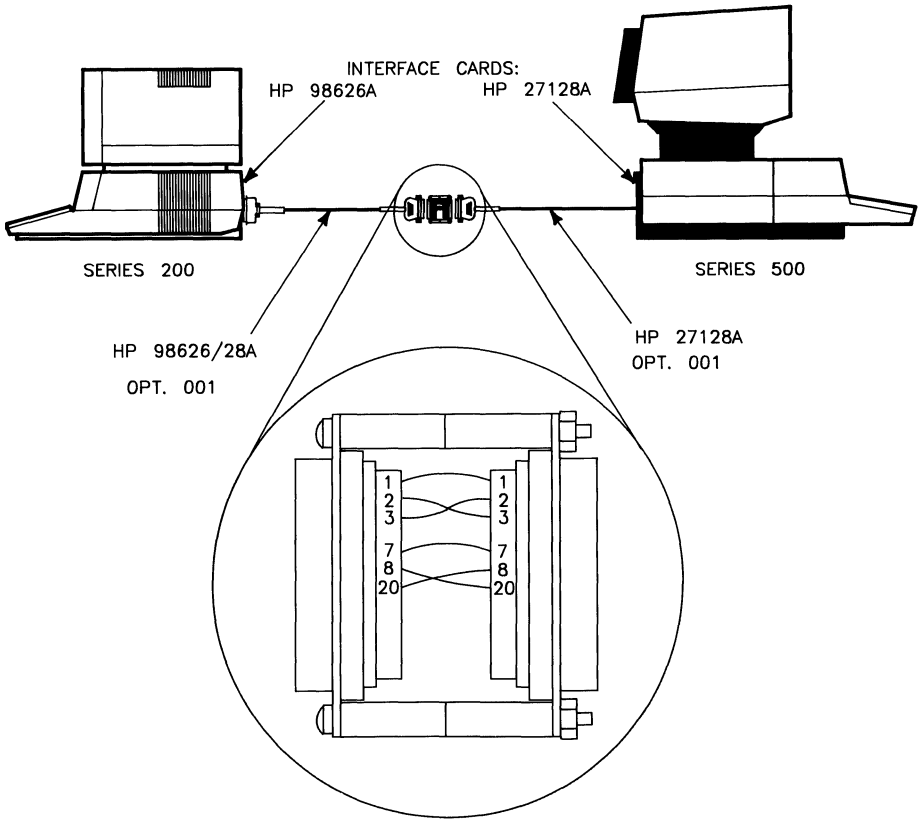


Figure 2-6. Series 200 to Series 500 (unidirectional)

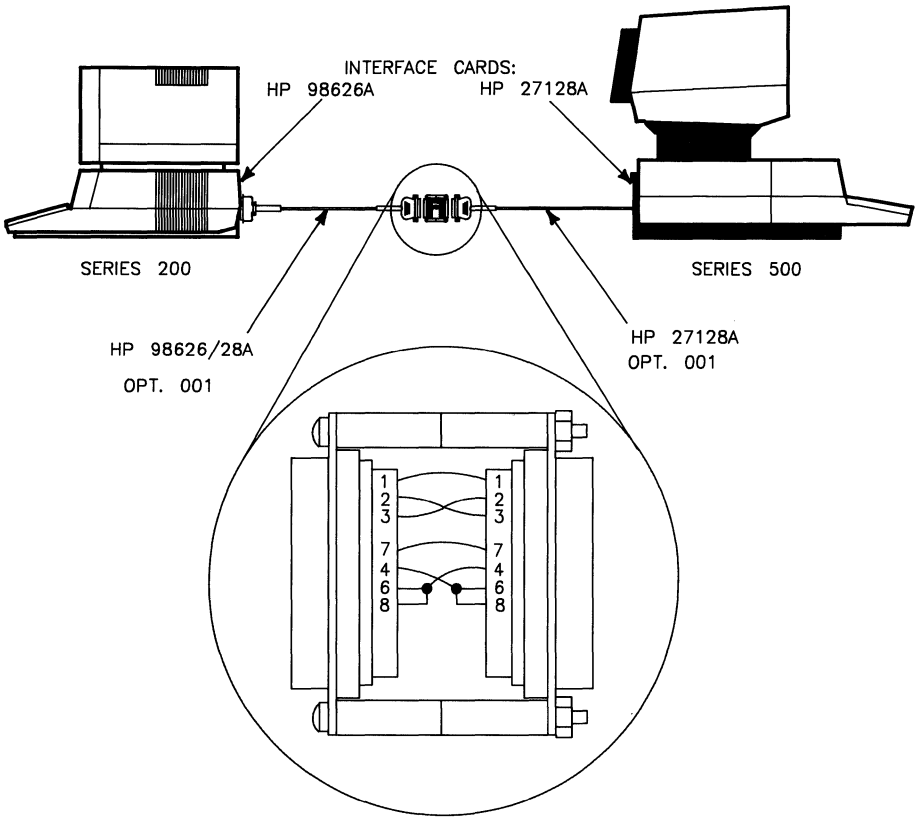


Figure 2-7. Series 200 to Series 500 (bidirectional)

- This is an example of directly connecting a Series 200 computer with either an HP 98626A or HP 98628A interface card to a Series 200 computer with either an HP 98626A or HP 98628A interface card.

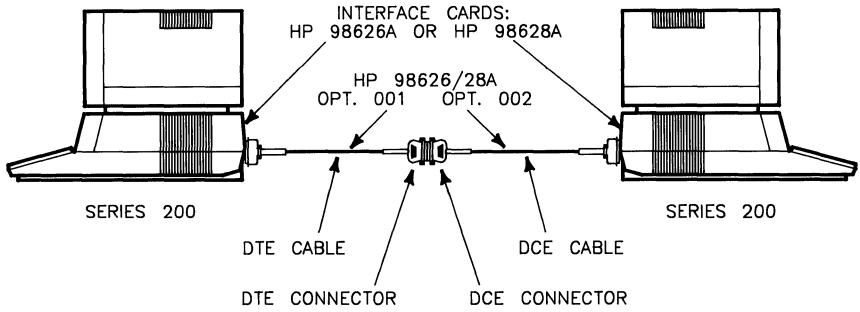


Figure 2-8. Series 200 to Series 200

- This is an example of directly connecting a Series 500 computer with an ASI (HP 27128A) card to a Series 500 computer with an HP 27130A/B (8-channel multiplexer).

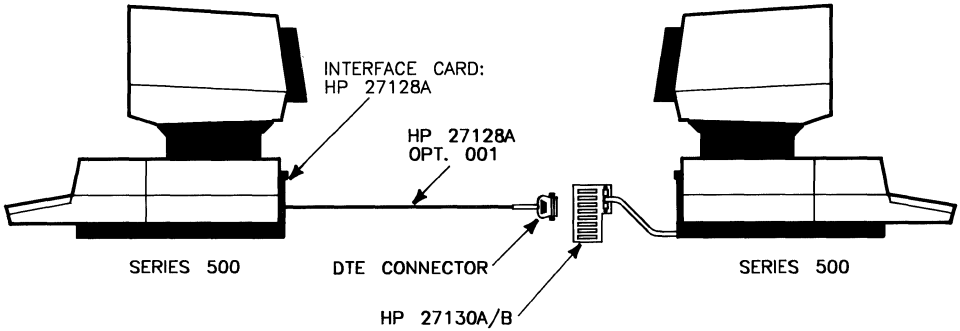


Figure 2-9. Series 500 to Series 500 MUX

- This is an example of directly connecting a Series 500 computer with an HP 27130A/B (8-channel multiplexer) to another Series 500 computer with an HP 27130A/B (8-channel multiplexer).

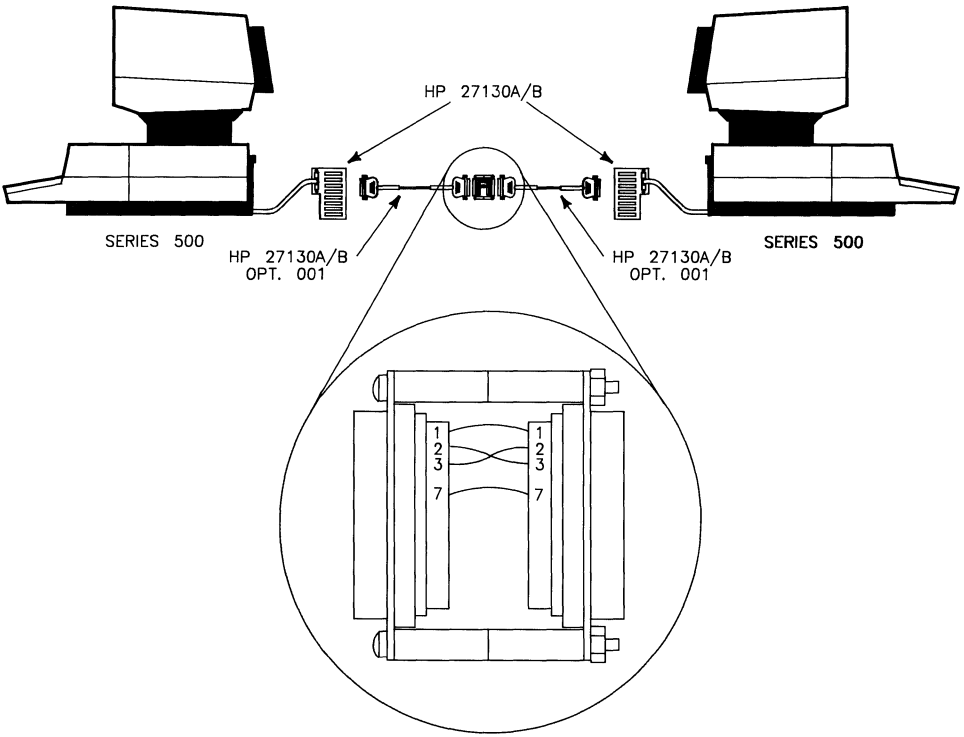


Figure 2-10. Series 500 MUX or Series 500 MUX

- This is an example of directly connecting a Series 500 computer with an ASI (HP 27128A) interface card to another Series 500 computer with an ASI (HP 27128A) card. The first figure shows a direct connection for communication in one direction, and the second figure shows a direct connection for communication in both directions.

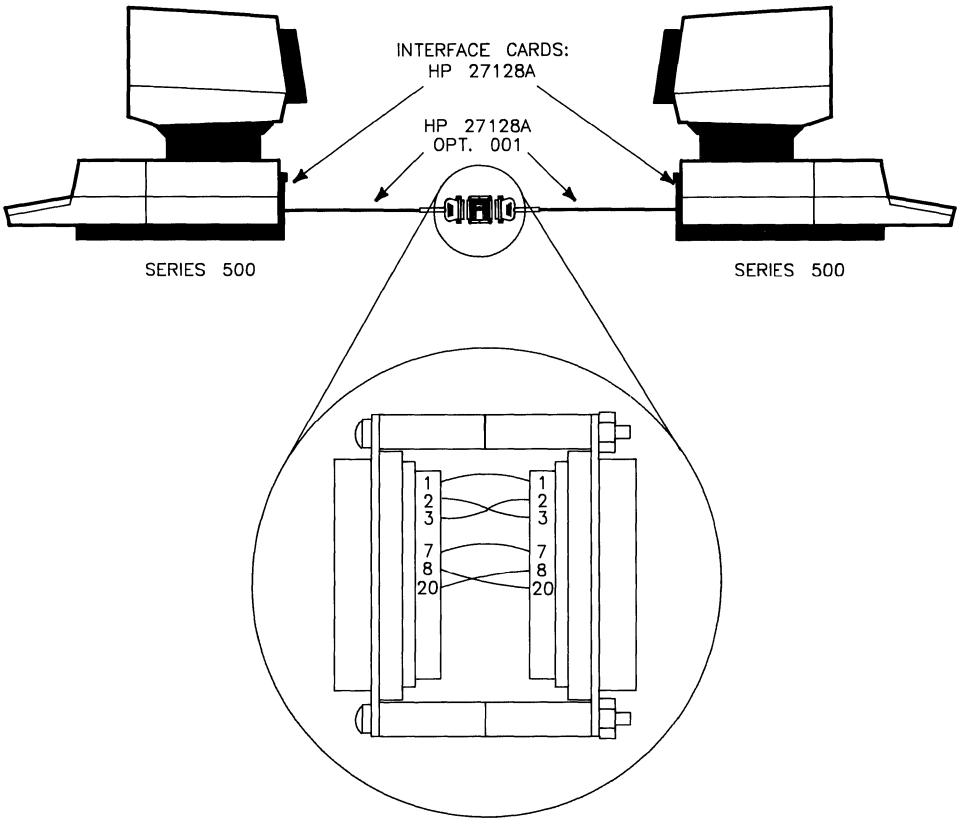


Figure 2-11. Series 500 to Series 500 (unidirectional)

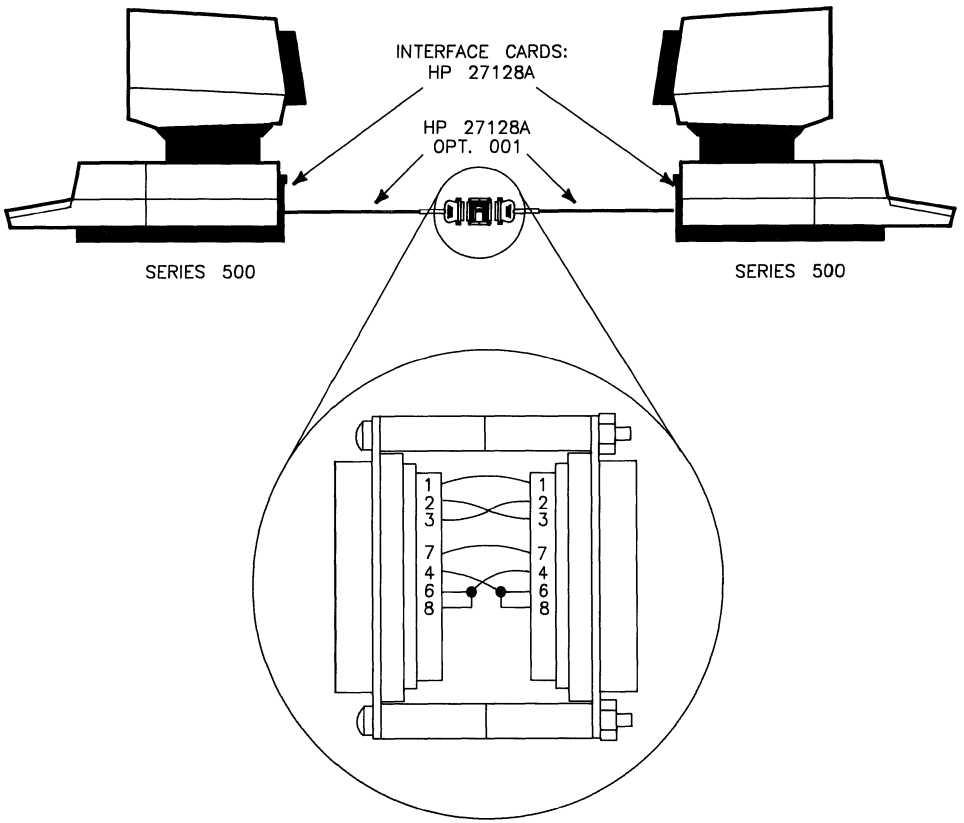


Figure 2-12. Series 500 to Series 500 (bidirectional)

- The direct connections explained here are between an HP serial interface card and an HP 27140A (6-channel modem multiplexer), between an HP 27130A/B (8-channel multiplexer) and HP 27140A, and between an HP 27140A and another HP 27140A. Note that the following diagrams show direct connection for transmission in only one direction where one device is ACTIVE (sends information) and the other device is PASSIVE (waits for the information).

The diagrams given in the following three figures **do not** use the special connection as shown in previous examples. The direct conections shown here are made by the use of a cable and connectors as labeled in the diagrams. It is up to you to make the cable. The wires on the ends of this cable are connected to the pins on the connectors as shown.

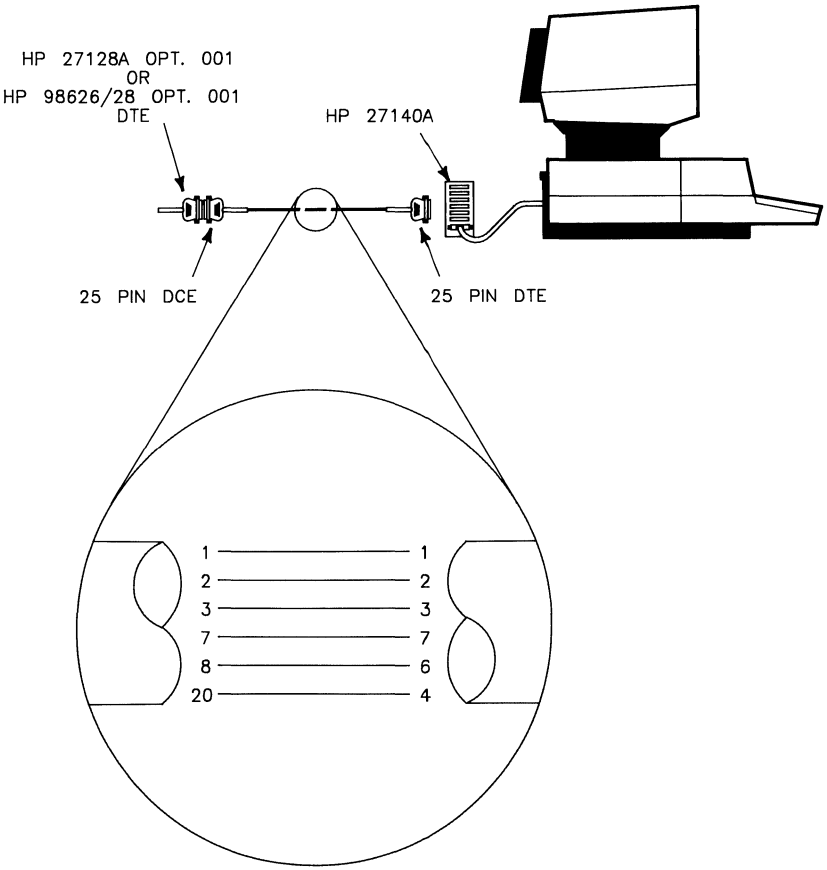


Figure 2-13. Serial Interface card to HP 27140A

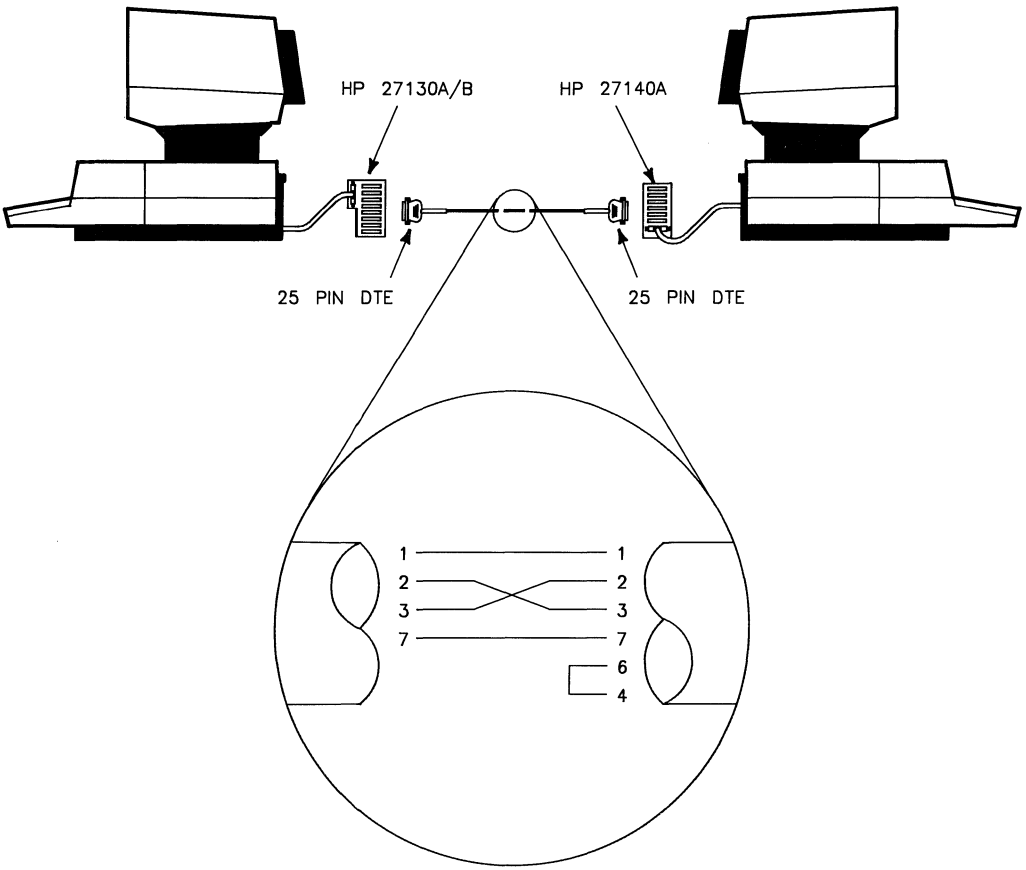


Figure 2-14. HP 27130A/B to HP 27140A

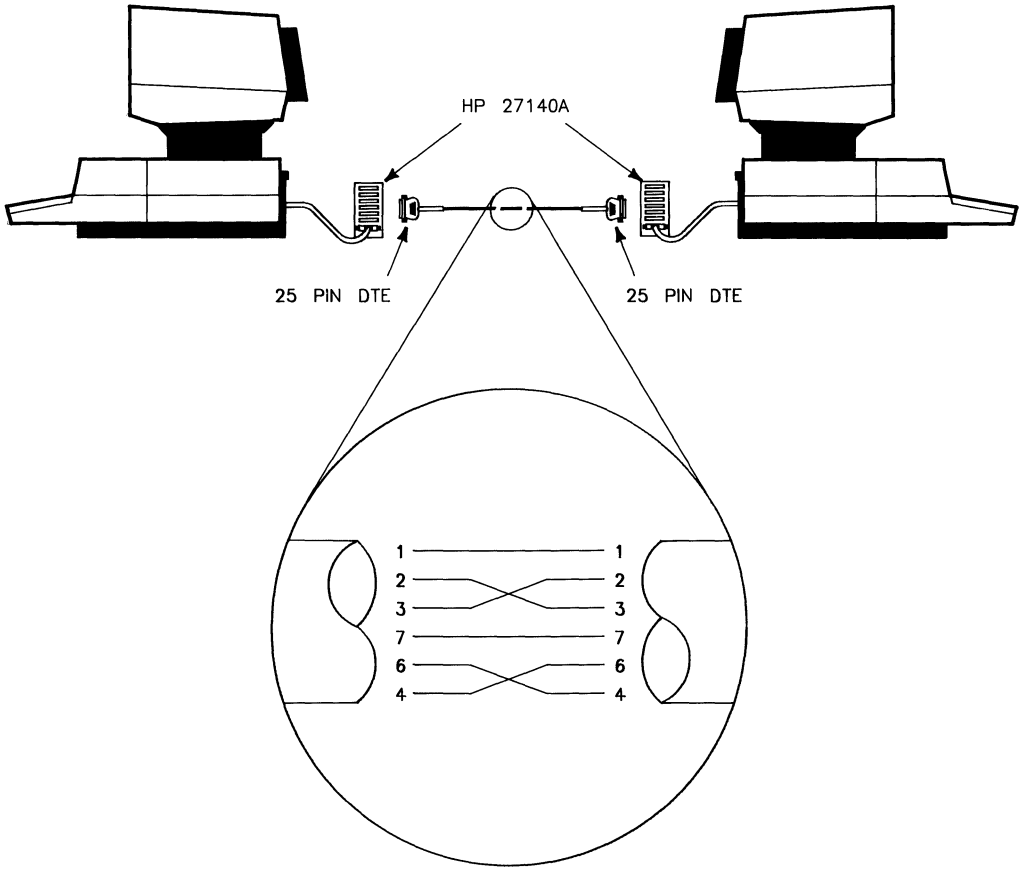


Figure 2-15. HP 27140A to HP 27140A

- The direct connections explained here are between an HP serial interface card and an HP 27140A (6-channel modem multiplexer), and between an HP 27140A and another HP 27140A. Note that the following diagrams show direct connection for transmission in two directions where either device may be ACTIVE (sends information) or PASSIVE (waits for the information).

  The following two diagrams shown **do not** use the special connector as given in previous examples. The direct conections shown here are made by the use of cables and connectors as labeled in the diagrams. The wires on the ends of the cables are connected to the pins on the connectors as shown.

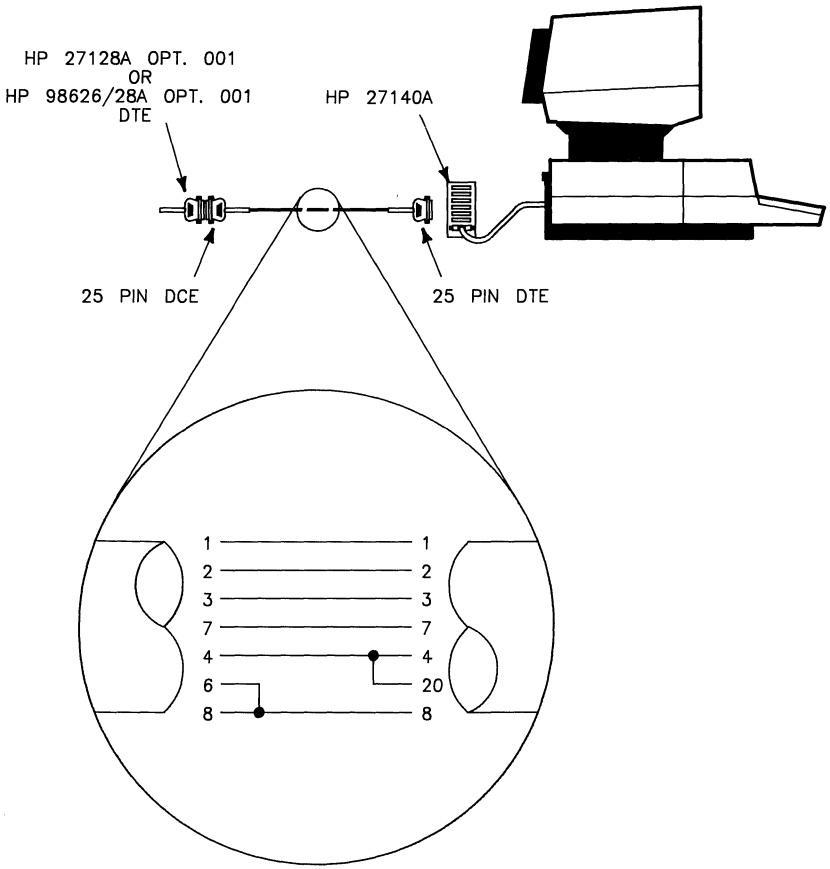


Figure 2-16. Serial Interface card to HP 27140A

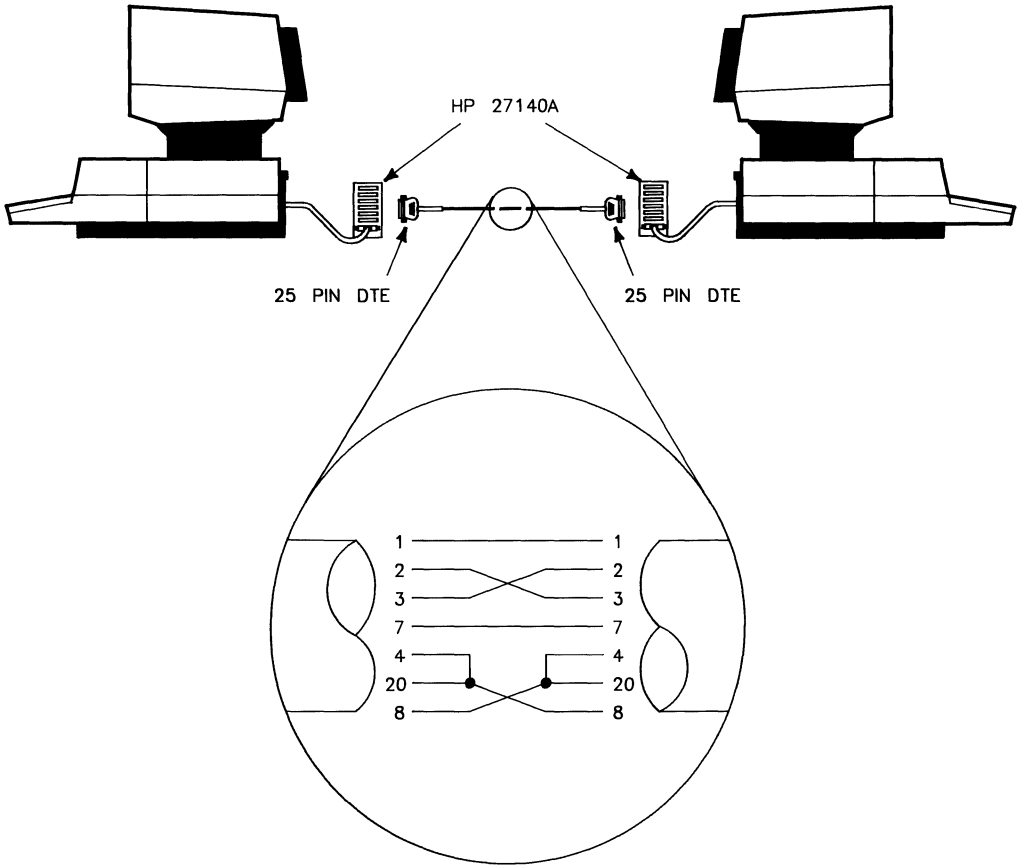


Figure 2-17. HP 27140A to HP 27140A

You should be aware of these additional considerations when making direct connection:

- Series 200 computers can use an HP 98642A (4-channel multiplexer) in place of an HP 98626 or HP 98628 interface card; however, you must treat the modem port as an HP 98626A or HP 98628A connection and you must treat HP 27130A/B (8-channel multiplexer) ports as multiplexers with 3-wire cables.
- Series 200 computers making direct connections using an HP 98644A card should treat this card as if it were an HP 98626A card.
- *Aterm* will not work with an HP 27140A (6-channel multiplexer), on any port that has a *getty* on it, or with any direct connection active in both directions.

The "Getty Entries" section of the chapter, "Software Configuration" describes how to make each line coming into your HP 9000 an ACTIVE (you can initiate calls) or a PASSIVE (you must wait to be contacted) line.

Making a Special Connector

The "special connector" is not a part numbered item which can be purchased. This connector has to be made using special parts. A list of the necessary parts is given as follows:

- Two DCE (25-pin RS-232C female connectors HP part number: 1251-0063) connectors,
- Two 1½ inch long machine screws (HP part number: 2200-0125) with nuts and lock washers (Note, these machine screws should be of the proper size to fit through the holes on both sides of the connectors and should not be so long that they keep you from plugging in the connector.),
- Four .625 inch metal (HP part number: 0380-0010) spacers,
- 8 - 1 inch long pieces of 24 gauge electrical wire (note the wires should have ¼ of an inch of insulation stripped from each end).

The "special connector" should look like the following example when put together:

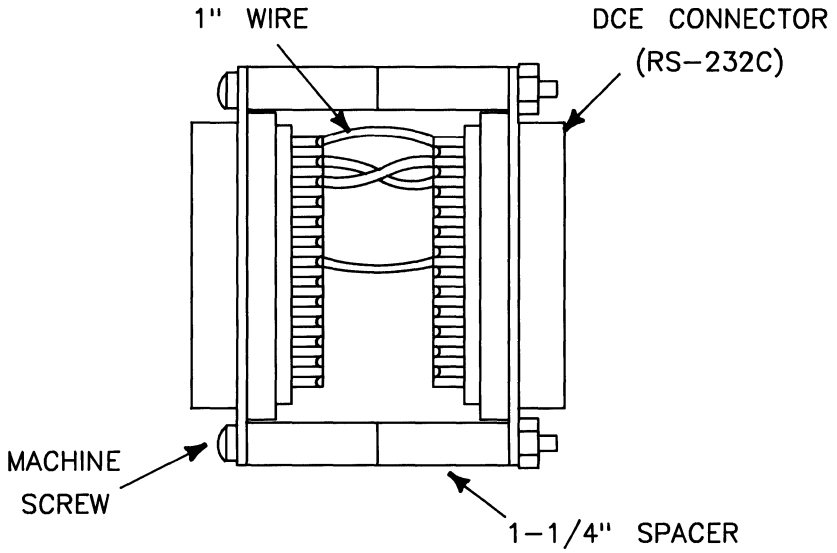


Figure 2-18. Special Connector

Wiring diagrams have been given on previous pages in this chapter to help you wire your "special connector". Note that the *HP 9000 Series 500 Configuration Information and Order Guide* provides an alternate method for making direct connections with other systems. To find the appropriate information in this guide, read the section, "Uucp Connection" found in, "Appendix I".

NOTES

Software Configuration 3

This chapter discusses the software configuration necessary for your Series 200/500 computer to use the asynchronous terminal emulator program (*aterm*) and the *uucp* facility programs.

Asynchronous Terminal Emulator HP 9000 Series 500

The asynchronous terminal emulator program (*aterm*) runs only on the Series 500 computer. To use *aterm* the System Administrator must:

- test to see if the SERIAL.opt driver is already there and if it is not then load the SERIAL.opt driver in the boot area of the system disc;
- create a device file;
- prepare a configuration file;
- install the *aterm* program.

Loading the SERIAL.opt Driver (aterm)

The SERIAL.opt driver is not installed when HP-UX is installed. Use the following procedure to install this driver:

1. Before installing the SERIAL.opt driver, test to see that it has not been previously installed by typing:

   ```
   osck -v /dev/sys_disc Return
   ```

   If the SERIAL.opt driver has been installed, skip the remainder of the section and proceed with the next one.

2. Determine which HP 9000 Series 500 system model number you are using from this list (e.g. 97078C):

   97070C Model 520 single-user system
   97078C Model 520 multi-user system for 32 users
   97079C Model 530/540/550 single-user system
   97080C Model 520 multi-user system for 16 users
   97088C Model 530/540/550 multi-user system for 32 users
   97089C Model 530/540/550 multi-user system for 16 users

3. Type:

   ```
   oscp -a /system/970xxA/SERIAL.opt /dev/sys_disc Return
   ```

   where `-a` says append to an existing operating system from a list of ordinary files, and put the resulting system in the boot area. The file `/system/970xxA/SERIAL.opt` is the driver being copied to the boot area of the operating system. The `xx` in this file is a two digit number taken from the last two digits of the system model number given in the above list of model numbers. For example, in the system model number *97070A* the `xx` would be the digits *70*. The special file `/dev/sys_disc` identifies the system disc.

4. Verify that the file has been copied into the boot area by typing:

   ```
   osck -v /dev/sys_disc Return
   ```

5. Re-boot your system to make the *SERIAL.opt* driver active.

Error 6 is generated if you attempt to use the *aterm* program without *SERIAL.opt* driver (19) being active.

Creating an ASI Device File (aterm)

To configure the emulator you need to know the special (device) file name associated with the Asynchronous Serial Interface (ASI or HP 27128A) card or the 8-channel multiplexer (HP 27130A/B). This name varies from system to system. The system administrator must create this device file by executing the following *mknod* command:

```
mknod pathname mode driver 0xScAdUV
```

where:

`pathname`	is the name of the file device you are creating.
`mode`	must be **c** to specify character mode device.
`driver`	must be the ASI or the 8-channel multiplexer driver number 19: the general purpose SERIAL GSG-IO driver.
`0x`	indicates that the digits following should be interpreted as hexadecimal.
`Sc`	is the select code of the ASI card in hexadeciamal notation.
`Ad`	are the hexadecimal digits representing the port number.
`U`	a single digit hexadecimal value specifying a secondary address, such as the unit number of the device. This value must be 0 for *aterm*.
`V`	a single digit hexadecimal value specifying a secondary address, such as the volume number in a multi-volume drive. This value must be 0 for *aterm*.

If you type:

```
mknod /dev/asi4 c 19 0x040000 Return
```

the device file named /dev/asi4 is associated with select code 4, bus address 0.

Preparing the Configuration File (aterm)

Before running the emulator, you need to prepare a configuration file. The emulator reads this file in order to establish the correct operating conditions for communications with the host computer system. Since the configuration file is a normal HP-UX text file, you may use any text editor to create and update this file.

Each configuration command occupies one text line of the file and consists of a two-character command name, followed by one or more blanks and the command argument value. No leading blanks or other characters are allowed in front of the command name.

The following table lists the recognized configuration command names. For each command, the table entry identifies the parameter, lists acceptable values, and specifies the default value. If you omit a given command from the configuration file, then the emulator assumes the default value. The only required parameter that does not have a default value is the device file name of the ASI (*da* parameter).

The range of acceptable values for the parameters in the table is dependent on the device you are using. If you are using a multiplexer many of these values are restricted, i.e., Break has no effect. Refer to the Diagnostic Messages section at the end of the "Asynchronous Terminal Emulator" chapter.

The arguments of each of the last six commands in the table (beginning with `st`) consist of one or more literal characters. These characters are often ASCII control characters (codes in the range of 0 to 37 octal). To express an ASCII control character as an argument, use the up arrow convention. For example, **^M** is equivalent to CR (carriage return, code 015). See the appendix, "HP 9000 Character Codes" for a list of up arrow codes. To use the up arrow itself as an argument, precede it with a backslash.

Table 3-1. Configuration Command Names

Name	Parameter	Description and Acceptable Values	Default
da	Device address	Device file name of the ASI (unquoted sequence of non-blanks).	No default
hn	Host name	Name of host computer system, for documentation only (unquoted sequence of non-blanks).	No default
db	Data bits	Number of data bits per character: 5,6,7,or 8. Note that the parity bit is in addition to data bits when parity is even or odd, but is included in the data bit count when parity is set to zero or one.	7
sb	Stop bits	Number of stop bits per character: 1, 1.5, or 2.	1
pa	Parity	Character parity: none (n), odd (o), even (e), zero (0), or one (1). The parity bit on incoming data is stripped and ignored if parity is zero or one.	o
dr	Data rate	Rate for data sent and received; valid rates are 50, 75, 110, 134.5, 150, 300, 600, 1200, 1800, 2400, 3600, 4800, 7200, 9600, or 19200 bits/second.	2400
mc	Modem Control	Enabled (+) allows auto answer or manual originate. Disabled (-) causes direct connection to modem with prompt for dialing.	-
ss	Switched service	Auto-answer (a) or manual originate (o); relevant for modem lines only.	o
ga	Gap	Number of character transmission times to delay between successive output characters (0 to 254).	0
ec	Echo	Enabled (+) if the host computer echos characters sent by the emulator, disabled (-) otherwise.	+
te	Terminal ENQ/ACK	Handshake for data sent by the host computer; if enabled the emulator responds with ACK when ENQ is received and the data buffer is available; enabled (+) or disabled (-).	+
he	Host ENQ/ACK	Handshake for data sent by the emulator; if enabled the emulator sends ENQ and waits for ACK from the host computer: enabled (+) or disabled (-).	-
tx	Terminal XON/XOFF	Handshake for data sent by the host system. If enabled, the emulator sends XOFF to suspend transmission an XON to resume: enabled (+) or disabled (-).	-
hx	Host XON/XOFF	Handshake for data sent by the emulator; if enabled the emulator suspends transmission when XOFF is received and resumes when XON is received.	-

Table 3-1. Configuration Command Names (continued)

Name	Parameter	Description and Acceptable Values	Default
im	Input mode	Method for accepting and processing input data from the host computer; block (b), character (c), or line (l).	b
om	Output mode	Method for sending data to the host computer; character (c) or line (l).	c
ph	Prompt handshake	If enabled (+), the emulator waits for the prompt sequence before sending each line of data during an input diversion; if disabled (-), no wait is performed.	-
pt	Prompt timeout	Number of seconds to allow for receipt of a prompt sequence during an input diversion. If prompt handshake is disabled, then the emulator always waits this length of time (1 to 600, 0 disables timeout).	0
st	Single text terminators	List of characters, any of which terminates a line sent by the host computer when the emulator is input line mode; up to eight characters may be specified.	None
dt	Double text terminator	A pair of characters which together terminate a line sent by the host computer when the emulator is in input line mode.	CR LF
ps	Prompt sequence	One or two characters which terminate a line sent by the host computer when the emulator is in input line mode, and which satisfy the prompt handshake if enabled.	DC1
bl	Beginning of line	Character prefixed to each line sent to the host computer.	None
el	End of line	One of two characters to be postfixed to each line sent to the host computer.	CR
es	Local command character	Character which designates a local command to be interpreted by the emulator if it comes at the beginning of a line read from standard input.	-

Emulator Modes and Handshaking (aterm)

Input Modes

The input mode specifies the method used by the emulator for accepting and processing input data from the host computer. The mode you select, character, buffered, or line, affects emulator appearance and efficiency.

- **Line input mode** causes the emulator to accept data from the host system one line at a time. In this mode, you must specify single and double text terminator characters and one or two prompt sequence characters sent by the host computer to mark the end of a line. All incoming data is stored in a private buffer on the HP 27128A (ASI) card until one of these characters is received.

  Line input mode is the most efficient mode for the terminal emulator, and should be used whenever possible if efficiency is a concern. Not every host computer system guarantees the transmission of required terminator characters, which may prove to be a problem if visual prompts are to be used. For example, the host computer may transmit:

  ```
  IF IT IS OKAY TO CLEAR RESPOND "YES"
  CLEAR?
  ```

  The cursor may be left immediately after the question mark (this is known as a visible prompt). If the host computer has no defined prompt sequence, it does not notify the emulator that a partial line has been transmitted and needs to be displayed. Since there is no end-of-line character, the emulator does not display the second line.

  The HP 3000 is one computer that utilizes a prompt sequence to solicit input from interactive terminals. The character used for a prompt is DC1.

- **Character input mode** processes data as it is received, one character at a time. Character input mode requires a great deal of interaction time between the HP 27128A (ASI) card and the main processor, and is also limited by delays caused by some CRT models. These limitations may not be apparent when using a 300 bit-per-second modem connection, but they can restrict throughtput and cause the emulator to fall behind and lose data if one of the terminal handshakes is not enabled.

- **Buffered input mode** behaves much like character input mode, except that if the emulator begins to fall behind the transmitted data, the ASI buffer begins to fill. When the emulator reads the contents of the ASI buffer, the entire contents of the buffer are read, rather than just one character. This requires only one interaction between the ASI and the main processor. If there is no data in the buffer, the main processor waits for the buffer to fill.

NOTE

The enable prompt timeout command is only effective for buffered input mode. It does not work for line or character input modes.

Be sure to define a prompt sequence and enable a prompt handshake to prevent data loss.

Output Modes

There are two output modes available to communication with the host computer.

- **Character output mode** passes the data from the terminal keyboard to the host computer one character at a time. All data characters, including the back space character, are sent to the host computer without any processing by the terminal emulator. If you have configured echoing as enabled (the host computer echos data sent to it), then local echoing is suppressed. In this mode, when you type a character, it will not appear on the CRT until it has gone to the host and back again.
- **Line output mode** transfers entire lines of data from the keyboard to the host. You must press `Return` to finish the line and send it to the host computer. In this mode, you can edit a line before sending it to the host computer.

The emulator is more efficient when configured for the line output mode, but for normal typing the difference is not significant. If the host computer has awkward editing conventions (such as not recognizing `Back space`), you may prefer to use line output mode. If you require character-by-character interaction with the host computer, you should use the character output mode.

Prompt Handshaking

Several parameters in the configuration file control the flow of data from the terminal to the host system. The values chosen for these parameters will affect how the emulator performs as a background process, using a shell input re-direction to provide a script file to the host system.

Some host systems provide a fixed prompt sequence. The HP 3000 provides a DC1 character to ask for the next data or command line from the connected terminal. For such a host, you must specify the correct prompt character and enable the prompt hankshake.

Other host systems provide a high-performance path for accepting data and commands from a terminal. For example, the host system may provide a large amount of buffering ("type-ahead") combined with fast input processing compared to the data rate of the link. In such cases, especially if a host flow control (DC1/DC3 or ENQ/ACK handshake) is available, you may find that you can disable the prompt handshake without losing data.

For some host systems, you may need to experimentally develop the correct sequence of prompt control configuration commands to run the emulator as a background process. The commands you need to include in your script file are described in the section, "Local Commands" found in the chapter, "Asynchronous Terminal Emulator".

Example Configuration Files (aterm)

To prepare a configuration file, you must first prepare a list of the host computer's communication parameters. For this example, the host computer is a VAX 11/750 and connection is to be made through a modem. The communication requirements for this example computer are:

8 data bits
zero parity
300 bits/second data rate
Connection through a modem, manual originate
No terminal or host ENQ/ACK
Terminal XON/XOFF handshake

All of the other communications parameters are the default values, and **do not** need to be changed. The ASI device file name of the HP 27128A (ASI) card is `/dev/asi3`.

The configuration file for these requirements is:

```
da /dev/asi3
hn VAX_modem_connection
dr 300
db 8
pa 0
mc +
te -
tx +
```

An example configuration file for the HP 1000 is:

```
dr 1200
hn NAME_phone_number
pa 0
db 8
da /dev/asi04
he -
te +
mc -
om c
im b
e1
ec +
```

An example configuration file for the HP 3000 is:

```
da /dev/muxb2
ec +
hn hp3000_name
```

An example configuration file for a UNIX system is:

```
da /dev/asi04
dr 1200
pa e
db 7
ec +
```

An example configuration file for a multiplexer is:

```
da /dev/muxb1
dr 1200
pa e
db 7
ec +
```

Uucp Programs

This section discusses the software steps the System Administrator must take to use a Series 200/500 computer as a node on the *uucp* network. Basically you must specify how and which other nodes on the network can contact you and how and which other nodes you can contact.

Seven main areas are covered:

- describing the boot and login processes;
- creating a device file;
- naming your node;
- *uucp* login;
- setting up a *getty* entry;
- installing the *uucp* programs;
- editing the necessary files.

General Startup Information

This section describes the boot and login processes. The tasks you need to perform for each file and command mentioned are discussed in later sections of this chapter.

Loading the Operating System

You must have the HP-UX operating system loaded in the boot area of your systems hard disc. Refer to the HP-UX *System Administrator Manual* for your particular computer for information about loading an operating system.

Loading Optional Drivers

This section uses the HP 27140A (6-channel modem multiplexer) to explain how to load an optional driver. This same procedure may be used to load other necessary drivers with the exception of Series 200 serial communication drivers which are loaded when the system is loaded.

The HP 27140.opt driver is not installed when HP-UX is installed. If you need to use this driver, use the following procedure to install it:

1. Before installing the HP27140.opt driver, test to see that it has not been previously installed by typing:

   ```
   osck -v /dev/sys_disc Return
   ```

This list the drivers which have already been installed with your system. If the HP27140.opt driver has been installed, skip the remainder of this procedure and continue reading in this section. If the driver has not been installed continue with the steps in this procedure.

2. Determine which HP 9000 Series 500 system model number you are using from this list (e.g., 97078C):

   97070C Model 520 single-user system
   97078C Model 520 multi-user system for 32 users
   97079C Model 530/540/550 single-user system
   97080C Model 520 multi-user system for 16 users
   97088C Model 530/540/550 multi-user system for 32 users
   97089C Model 530/540/550 multi-user system for 16 users

3. Type:

   ```
   oscp -a /system/970xxA/HP27140.opt /dev/sys_disc Return
   ```

   where `-a` says append to an existing operating system from a list of ordinary files, and put the resulting system in the boot area. The file `/system/970xxA/HP27140.opt` is the driver being copied to the boot area of the operating system. The `xx` in this file is a two digit number taken from the last two digits of the system model number given in the above list of model numbers. For example, in the system model number *97070A* the `xx` would be the digits *70*. The special file `/dev/sys_disc` identifies the system disc.

4. Verify that the file has been copied into the boot area by typing:

   ```
   osck -v /dev/sys_disc Return
   ```

5. Re-boot your system to make the *HP27140.opt* driver active.

Note, to use this same procedure to load any other drivers that you might need for your particular system application, just change the system model number (e.g. 97070C) and the driver name (e.g., HP 27130 opt).

Final Sequence of Events Once the System is Loaded

Once the HP-UX operating system configuration is loaded, its initialization process begins and executes the file `/etc/init`. The init program reads the `/etc/inittab` file to find the all incoming ports. You need to add an entry, called a `getty` (get terminal) entry, in this `inittab` file for each incoming *uucp* line. The operating system can then regularly check all incoming lines to see if another system is trying to communicate with you. The `getty` entries are therefore needed only if this line is used PASSIVEly or ACTIVEly by your local system. The `getty` entry specifies the special file name of your dial-in line as well as its communication speed. The getty command also causes the "`login`: " prompt to be sent to the calling computer when a connection is established.

The *init* program then invokes the */etc/rc* shell script. This script causes many things to happen, among them setting your system's node name and executing the */etc/cron* program, which runs commands on a scheduled basis. You should use the */etc/cron* program to regularly compact and clean up some of the files used by the *uucp* facility, for example, files used to log transactions (see the chapter, "Uucp Facility Demons").

Creating a TTY Device File

A device file must exist in the `/dev` directory to associate each device connected to your Series 200/500 computer with a special file name. This is done with the *mknod* command. Note that the process for creating a device file in the section of this chapter entitled "Creating an ASI Device File" may be used for creating `ttyd`, `cul` (modem), `cua`(autodial) device files.

To create a `tty`, `cul`, or `cua` device file, make an entry of the form:

```
mknod /dev/name c 31 0xScAdnn
```

where:

- The file `/dev/name` is either `/dev/ttydXX`, `/dev/culXX`, or `/dev/cua`. "XX" are characters used to differentiate between the device files for the various communication ports (for example, ttyd01 and ttyd02). Note that the file naming convention `ttydXX` is for dial up lines and the file naming convention `ttyXX` is for hardwire connections.
- The characters `Sc` identify the 2-digit hexadecimal select code of the communication port's interface.
- If the communication port is connected via an HP 27130A/B (8-channel multiplexer) or HP 27140A (6-channel modem multiplexer), the characters `Ad` specify a 2-digit hexadecimal port number on the multiplexer. If it is connected via an HP 27128A Asynchronous Serial Interface, the characters `Ad` should both be zero.
- The characters `nn` represent one of the following for the HP 27128A card and the HP 27140A modem multiplexer. For the HP 27130A/B, all lines are `00` because it has no modem capabilities.

  00 for the incoming device file. 01 for the outgoing device file. 02 for the incoming device file if the modem obeys CCITT protocols. 03 for the outgoing device file if the modem obeys CCITT protocols.

Note that `ttyd`, `tty`, `cua`, and `cul` files use driver:

31 when using the HP 27128A serial interface card or the HP 27130A/B 8-channel multiplexer.

29 when using the HP 27140A 6-channel modem multiplexer.

1 when using serial interface cards on a Series 200 computer.

To illustrate, suppose you have inserted an HP 27128A Asynchronous Serial Interface into your computer at select code 2. You next connect a modem to it so that this port may be used as a dial-in and dial-out port. The following *mknod* commands create the necessary device files:

```
mknod /dev/ttyd02 c 31 0x020000     #incoming device file
mknod /dev/cul02 c 31 0x020001      #outgoing device file
mknod /dev/cua02 c 31 0x020001      #autodial device file
```

Note that, depending on whether or not you have the directory */etc* set up in your directory path, you would execute the *mknod* command with the */etc* prefix. For example:

```
/etc/mknod /dev/ttyd02 c 31 0x020000
```

Check to be sure that you have both read and write access to the */dev* device file for lines used as out going ports. The *ll* command lists file characteristics, for example:

```
ll /dev/file_name
```

displays the following information:

```
crw-rw-rw- 1 root other 31 0x020000 May 13 14:37 /dev/file_name
```

showing read/write access for everyone.

Change the protection to read/write access if necessary with the *chmod* command, for example:

```
chmod 666 /dev/file_name
```

where:

`666` is the HP-UX code for read/write access.

Note that the *uucp* `/dev` files `cul` and `cua` are normally readable and writeable by everyone.

Remove any old entries from the */dev* directory which have the same select code as the cards you now intend to use for ingoing or outgoing *uucp* calls.

For more information on the *mknod* command, read the chapter, "The System Administrator's Toolbox" in your *System Administrator Manual.* For more information on the *chmod* command read *chmod(1)* in your *HP-UX Reference.*

Naming Your Node

Every node on the *uucp* network must have a unique nodename. Note that the Local Area Network (LAN) nodenames are independent of the *uucp* nodenames.

Since nodenames are frequently typed, a carefully planned convention can help all users identify and remember system nodenames. Try to avoid extraneous characters such as hyphens, numbers or upper-case letters.

To determine if your node has a name and if so what the nodename is, type:

```
uname -n
```

or

`uuname -l` (if uucp has been installed)

or

```
hostname
```

There are two ways to name your node:

1. use the *hostname* command in the system script */etc/rc* when you want this node name to come up with every change of state or power-up;
2. execute the *hostname* command as the superuser (`root`) when you want the name to last until you change your state or power-down your system.

Using the */etc/rc* system script is the recommended procedure.

Uucp Login

When you installed your HP-UX system, this entry was made in */etc/passwd* to provide a login for *uucp*:

```
uucp::5:1::/usr/lib/uucp:/bin/sh
```

You need to change this to allow a remote system to contact you and initiate the process *uucico* which takes care of any work requested.

This login should have a password for obvious security reasons. An example of the initial entry prior to setting the password is:

```
uucpln::5:5::/usr/spool/uucppublic:/usr/lib/uucp/uucico
```

where:

`uucpln`	is the *uucp* login name and is restricted to a maximum of eight characters;
`::5:5::`	are conventions used by HP-UX to designate the user and group access restrictions (They have nothing to do with the mode bits.);
`/usr/spool/uucppublic`	is a public area normally accessible to everyone and used here as the login directory;
`/usr/lib/uucp/uucico`	is a program which must be started when you login as a uucp user.

Getty Entries

You need a *getty* entry in your */etc/inittab* file for each line which is used as a *uucp* login port for your Series 200/500 computer.

The following is an example of the format used for the *getty* entries in */etc/inittab*:

```
/etc/getty [-h] [-t timeout] line [speed]
```

where:

`-h`	forces a hangup on the line by setting the speed to zero before setting the speed to the default or specified speed.
`-t timeout`	specifies that `getty` should exit if the open on the line succeeds and no one types anything in the specified number of seconds.
`line`	is the name of a `tty` line in `/dev` to which `getty` is to attach itself.
`speed`	is a label to a `speed` and `tty` definition in the file `/etc/gettydefs`.

An example for a direct connection at 9600 baud is:

```
/etc/getty tty02 5
```

An example for a modem connection at 1200/300 baud is:

```
/etc/getty ttyd02 3
```

Editing the Library Files for Uucp

Before you read this section, you need to have a basic understanding of where various files and directories are located in the HP-UX file system. The specific directories and files you should be aware of are listed in the following diagram:

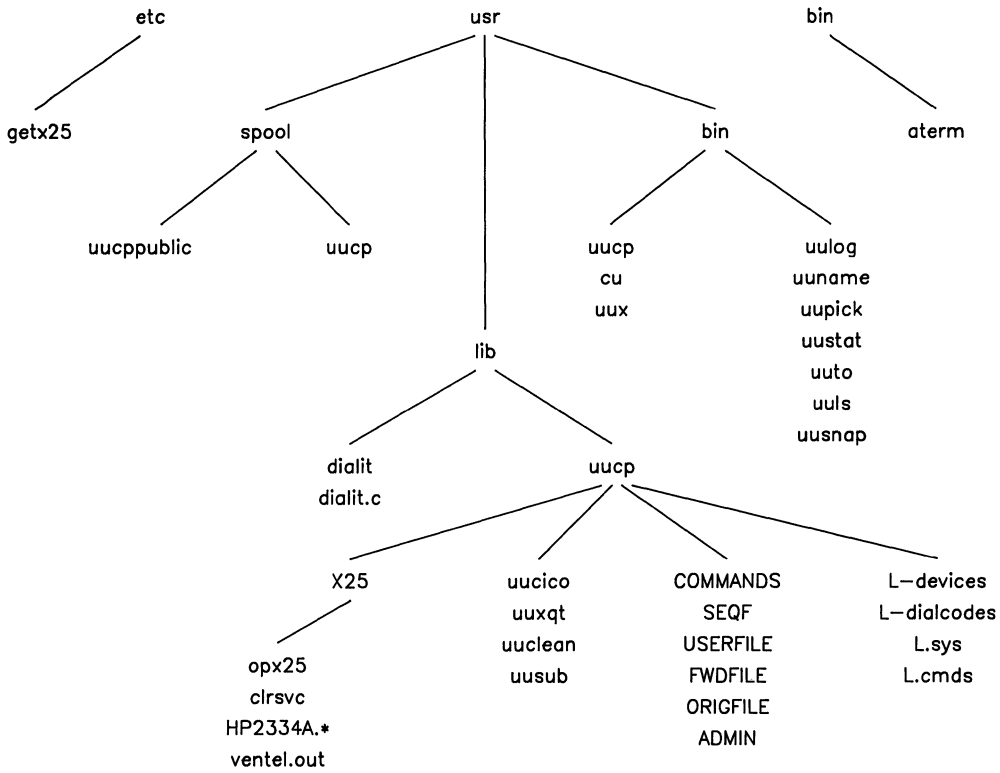


Figure 3-1. Files/Directories Installed

The directory */usr/lib/uucp* contains program modules which the *uucp* commands need to use. These program modules initiate and carry on all communications with remote systems and perform the remote execution of commands.

You must edit the following files:

- `FWDFILE` — to provide a list of systems your system can forward to or through.
- `ORIGFILE` — to provide a list of systems which may forward to or through your system.
- `L.sys` — to specify the remote system parameters;
- `L-devices` — to provide a list of valid devices;
- `dialit.c` — to modify the C source code for a dialing routine *Dialit.c* then must be compiled and directed to the *dialit* file.;
- `L-dialcodes` — if you use special characters in the modem phone number;
- `USERFILE` — if you want security protection for a file(s);
- `L.cmds` — if you want to list the commands a remote system can execute with uux on your local system.

The system administrator must edit these library files before any *uucp* communications take place. The "Library Files" section of the chapter, "Uucp File System" in this manual discusses how you place information specific for your needs into the files. Reading this entire chapter can provide you with a better understanding of the *uucp* facility processes and can help you should problems occur.

Additional Uucp Information

The chapter, "Status, Log and Cleanup Information" in this manual discusses the methods by which you can monitor status and log information, as well as the ways you can rid your file system of old or unwanted files. It is recommended that you begin implementing these features as soon as possible.

Configurations Complete

When you (the System Administrator) have completed the steps outlined below your Series 200/500 computer is completely configured to use the asynchronous terminal emulator (*aterm*) and the *uucp* facilities.

The *aterm* program requires:

- installing the hardware;
- adding the *SERIAL.opt* driver to the boot area;
- creating a device file;
- creating a configuration file;
- using optinstall to install the aterm program.

The uucp facility programs require:

- installing the hardware;
- adding any necessary drivers to the boot area;
- creating a device file;
- naming your node;
- providing a uucp login entry;
- specifying a getty entry;
- using optinstall to install the uucp file structure;
- configuring the uucp library files.

Asynchronous Terminal Emulator

4

Invoking the Emulator (aterm)

The emulator appears as any other HP-UX command. It occupies one executable file, normally located in the `/bin` directory. If you wish to run it, type: `aterm cfile`

NOTE

This program does not function in conjuction with the *tee* program.

Here, `cfile` is the name of your previously prepared configuration file. If this file name is omitted or cannot be opened or read by the emulator, the emulator aborts and displays a diagnostic message. If an improper command or argument is detected in the configuration file, the emulator issues a diagnostic message and continues running. If this happens, check the running configuration by typing the local command `~*` and then Return to see a listing of the parameters. You can change values in the emulator configuration with local commands described in the next section.

If you have enabled the modem handshake, the message:

```
aterm: waiting for modem connection
```

is displayed. If you are originating the call, you have one minute to make a modem connection to the host. If you do not, the program is aborted. If you are receiving a call on an auto-answer modem, you have one minute from the time a call is received to establish a connection. If you do not make a connection within a minute, the message:

```
aterm: can't complete modem connection
```

is displayed and you must start the program again to make a connection.

When the emulator has completed initialization and is ready to accept commands or data, the emulator displays:

```
aterm: connected
```

To invoke the emulator as a background process, use the standard shell input diversion notation. For example,

```
aterm fig3000 <trans1 &
```

causes the file `trans1` with commands to be sent to the host computer. This file can also contain several local commands. Note that this is a shell input diversion, and **not** an emulator input diversion (emulator input diversions are described in the section, "Local Commands" in this chapter). The HP-UX system starts the emulator as a background process and displays the process number. This leaves the keyboard free to execute other commands.

In the above example, any data sent to standard output appears on your screen. One way to suppress this is to specify a shell output diversion to a file. Diagonostics normally appear on the terminal emulator screen, but they can be redirected to another file or can be included in an output diversion file.

To divert the error messages to a separate file, use the command:

```
aterm cfile > output_file 2>errfile
```
(for Bourne Shell)

```
(aterm cfile > output_file) >& errfile
```
(for C Shell)

This diverts the emulator output to a file named `output_file` and any error messages to a file named `errfile`. To include any error messages in the output file, use the command:

```
aterm cfile > output_file 2>&1
```
(for Bourne Shell)

```
aterm cfile >& output_file
```
(for C Shell)

Local Commands (aterm)

Local commands are commands that are interpreted by the terminal emulator without being passed on to the host computer. The emulator can pass some of these commands to the local HP-UX system for execution. You must specify a local command with the local command character at the beginning of an input line. This can be done either from the keyboard or from a command file. The default local escape character is ~. You can change the local command character by changing the local escape character in the configuration file.

To send a line to the host computer beginning with ~ or any other local command character, begin the line with two local escape characters (~~). The emulator removes one of the characters and sends the rest of the line to the host computer.

In the examples in this chapter, it is assumed that you are using the default local command character ~. If you choose to change the local command character to some other character, substitute the new character for ~.

When entering a local command, you may use Back space to correct one or more incorrect characters. The command ESC M does not work to clear a line of text.

Configuration Commands

You can modify all the configuration parameters of the terminal emulator except device address, host name, modem connection, or switched service without terminating the emulator or disconnecting the data link. This feature is useful if you do not know the exact communication format of the host computer. You can vary parameters such as input and output modes, echo, and terminator sequences and see the results of the modification immediately.

The local emulator command character for a configuration command is *. As an example, if you wish to change the data rate to 9600, enter:

```
~*dr 9600
```

If the attempted change is invalid, a diagnostic is displayed. You may need to use configuration commands in a script file if you run the emulator as a background process. A good use of the local configuration commands is to change the prompt handshake and the prompt timeout. An example is given at the end of this chapter.

Emulator Configuration Display

If you enter a local configuration command without an argument, the emulator displays the current configuration. If you enter:

```
~*
```

the emulator writes a formatted display of the current configuration to the standard output. An example display is:

```
+-----------C-O-N-F-I-G-U-R-A-T-I-O-N---D-I-S-P-L-A-Y-----------+
!  DEVICE FILE NAME: /dev/asi04                                  !
!  HOST SYSTEM NAME: hp3000                                      !
!  DATA RATE: 1200   DATA BITS: 8   STOP BITS: 1   PARITY: ZERO  !
!  TERMINAL ENQ/ACK: ON        HOST ENQ/ACK: OFF                 !
!  TERMINAL X-ON/X-OFF: PFF    HOST X-ON/X-OFF: OFF              !
!  MODEM CONTROLS: NO          SWITCHED SERVICE: ORIGINATE       !
!  INPUT MODE: BUFFERED        OUTPUT MODE: CHAR    GAP:  0      !
!  HOST ECHO: YES              SIGNAL RATE: HIGH                 !
!  NO SINGLE TEXT TERMINATOR CHARACTERS                          !
!  DOUBLE TEXT TERMINATOR SEQUENCE: < 15> < 12>                  !
!  PROMPT CHARACTER: < 21>         HANDSHAKE: DISABLED           !
!  PROMPT TIMEOUT:  0                                            !
!  NO BEGINNING OF LINE OUTPUT PREFIX                            !
!  OUTPUT POSTFIX FOR END OF LINE:  < 15>                        !
!  LOCAL COMMAND ESCAPE CHARACTER:  <176>                        !
+---------------------------------------------------------------+
```

You can divert the information from the display to a file. For example,

```
~* >/dev/lp
```

writes the formatted configuration display to the file `/dev/lp`, which could be a printer on the HP-UX system.

Emulator Status Display

The ? command character produces a listing of the current emulator status. The format is:

```
~? [> dfile]
```

The characters in the bracket can optionally be used to write the status display to a destination file. An example display produced by this command is:

```
+-----------------S-T-A-T-U-S---D-I-S-P-L-A-Y-----------------+
!                                                             !
!  VERSION:  QA RELEASE X.11, 8 OCTOBER 1982                  !
!  STANDARD INPUT FROM KEYBOARD                               !
!                                                             !
+-------------------------------------------------------------+
```

ASI Status Display

To produce a listing of the ASI status, use the command:

```
~/ [> dfile]
```

The information displayed is in the form that it is read from the ASI card. This information reflects the status of registers on the interface card, and may be useful to HP service personnel if they must diagnose problems with the emulator. An example listing is:

```
+--------------------A-S-I---S-T-A-T-U-S--------------------+
!  SWITCHES:  BEH   MODE CONTROL: 80H      END COUNT:   200  !
!  ALERT1 READ MODE: ENABLED                                !
!  STRIPPING:   0H      HANDSHAKE: 1H    ENQ/ACK TIMER:   5  !
!  DATA BITS:   3      STOP BITS:  0          PARITY:     3  !
!  BAUD RATE:   9      XMSN MODE:  2   ENQ/ACK COUNTER:  80  !
!  GAP TIMER:   0    BREAK TIMER:  4    INACT. TIMER:     0  !
!  MODEM CONNECTION TIMER:  60   CONNECTION: ANSWER         !
!  SINGLE TEXT TERMINATOR CHARACTERS:                       !
!       < 12>                                               !
!  DOUBLE TEXT TERMINATOR SEQUENCE: < 15> < 12>             !
!  PROMPT SEQUENCE:  < 21> <  0>                            !
+-----------------------------------------------------------+
```

Emulator Input Diversion

The emulator can read the contents of a data file and send them to the host computer with an emulator input diversion command. Note that this is different from a shell input diversion, which is specified when you invoke the emulator. A shell input diversion file may contain local commands, just as if you typed them into the emulator interactively from the keyboard. An emulator input diversion file is read from the beginning to the end of the file, regardless of the file's contents. This means that commands preceded by the local command character (default ~) and read from the input diversion file are passed on to the host computer without being executed.

The emulator input diversion command must be:

```
~< [:] sfile
```

The brackets contain the optional character :. This character causes a silent input diversion. If this character is used, the emulator does not display the data read from the file to standard output as it is sent to the host computer. Omit this character if you want to monitor the data file transfer.

The `sfile` is the source file that is to be read for the input diversion. If the emulator is not able to open or read this file, a diagnostic is displayed and the diversion is terminated.

The input diversion can be manually terminated by entering ~<. If there is no diversion occurring when the command is entered, there is no effect. You must usually send an appropriate command to the host computer before you can start the emulator input diversion. For example, if you wish to use the FCOPY utility on an HP 3000 computer, type:

```
RUN FCOPY.PUB.SYS
FROM=;TO=DESTFILE;NEW
~<:  listf.c
:EOD
```

In this example, these commands invoke the HP 3000 FCOPY.

`RUN FCOPY.PUB.SYS`	invokes the file copy utility.
`FROM=;TO=DESTFILE;NEW`	specifies that data is to be copied to the new permanent file `DESTFILE`.
`~<: listf.c`	is the local emulator command that causes the contents of the file `listf.c` to be sent to the HP 3000. The use of the character : indicates that it is to be a silent diversion (not displayed on the CRT).
`:EOD`	is an FCOPY end of file indicator that completes the FCOPY command.

NOTE

The ~< command is the only command that may be typed on the keyboard during an emulator input diversion without risking loss of data. Note that executing the ~< command in this manner stops emulator input diversion.

Emulator Output Diversion

You can transfer files from the host computer to the local computer with an emulator output diversion command. The command is:

```
~>[>][:] dfile
```

The character > in brackets can be used to append the data to an existing file. If you omit this character, the emulator overwrites an existing file. The : character can be used to specify a silent output diversion. If this character is used, the emulator does not write data from the host computer to standard output as it is received. If you want to monitor the progress of the output diversion, omit this character.

You can terminate the output diversion at any time by entering the command :

```
~>
```

To use the emulator output diversion to transfer a data file from the host computer, you usually need to enter some command to the host computer that causes the contents of remote file to be copied or listed to the data communications line. This command should normally be entered after you have started the emulator output diversion. For example, you could use the FCOPY utility on the HP 3000:

```
RUN FCOPY.PUB.SYS
~> fileone
FROM=M23JUNE;TO=
~>
```

In this example:

`RUN FCOPY.PUB.SYS`	invokes the FCOPY utility.
`~> fileone`	starts the emulator output diversion.
`FROM=M23JUNE;TO=`	is an FCOPY command that causes the contents of the file `M23JUNE` to be copied to the standard output file, which in the case of a login terminal is the data communications line.
`~>`	terminates the output diversion.

NOTE

Since you are diverting the output of the emulator to a file, the command to make the host computer list a file is also included in the output file. Remove this line from the file with an editor.

Data Link Break

Since Break on the emulator keyboard is ignored by the terminal emulator, a separate key sequence is defined to send a data link break to the host computer. The data link break is the local command:

```
~#
```

You must press Return to send a data link break, but only the data link break character is sent to the host computer.

Terminating the Emulator

It is recommended that you terminate the emulator with a local command. If you are using the default local command character of ~, then the command is:

```
~.
```

If you type CTRL D to signify the end of a file, the emulator sends the control-D character to the host computer. However, the terminal emulator performs normal termination procedures when an end of file is encountered during a shell input diversion operation.

If you stop the emulator, or it terminates in an unexpected manner, the local keyboard may be left in an undefined state. This is more likely to happen if the emulator is configured for host echo disabled and character mode output. If this happens, characters may still be accepted by the HP-UX system, even though they are not displayed. To correct this situation, type the following command:

```
stty sane Return
```

Await Prompt Condition

When you invoke the emulator as a background process using a shell input file diversion, you may develop extensive script files to direct the emulator's automatic operation. When you have specified prompt conditions (either handshake or timeout), the emulator waits for the prompt condition to be met after the transmission of each data or command line to the host computer.

The emulator accepts and acts on the first line in the script file without waiting for the prompt condition. After each local command (which does not send data to the host computer), the emulator accepts and acts upon the next line of the script file without waiting for the prompt condition. The command to send a data link break (`~#`) is not considered to be sending data to the host computer.

To make the emulator wait for a prompt condition without sending any data, use the command:

```
~@
```

This command does not send data to the host, but does cause the emulator to wait for either the prompt sequence to be received or the prompt timeout to occur (according to the current configuration). While the emulator is waiting for the prompt condition, incoming data from the remote computer is accepted and processed.

Local Shell Commands

You can execute a local HP-UX shell command with the emulator running. The command is:

```
~! command
```

This causes the local HP-UX system to execute a command. For example, if you want to check the contents of your local directory, type:

```
~! ls -l
```

You can also enter a series of local HP-UX local shell commands by typing `~!` without any commands. You may then type in a series of commands to the local HP-UX shell. While local HP-UX commands are executing, the keyboard and CRT are connected directly to standard input and output. To terminate a local shell, or to terminate a local HP-UX command that reads from the keyboard, enter a line that contains only CTRL D. Normal operation resumes as soon as the local command or shell terminates. This is indicated by the prompt `!` from the emulator.

Example: A Script File to Log In to a Remote Computer

The following script file is used to log the terminal emulator in to a remote VAX computer system.

```
~*ph -
~*pt 5
~@
~*dr 1200
~#
~@ bob
~*ps ?
~*pt 0
~*ph + betty
```

When this script file is read by the emulator, the following sequence occurs:

`~*ph -`	Prompt handshake is disabled.
`~*pt 5`	Prompt timeout is enabled and set to five seconds.
`~@`	The emulator is told to wait for a prompt.
`~*dr 1200`	The emulator data rate is set to 1200.
`~#`	A break character is sent to the remote computer. The VAX recognizes this as a signal to increase the data rate to 1200.
`~@`	The emulator is told to wait for the prompt at the end of the VAX login message. This prompt must already be specified in the configuration file.
`bob`	After the prompt is received, the user name `bob` is sent to the host computer.
`~*ps ?`	The prompt character is changed to `?`.
`~*pt 0`	Prompt timeout is set to 0 seconds.
`~*ph +`	Prompt handshaking is enabled to respond to the character `?`, the last character of the line `Password?` from the VAX.
`betty`	After the new prompt `?` is received, the password `betty` is sent to the VAX. The emulator should now be logged on to the host computer.

Diagnostic Messages (aterm)

This section lists the diagnostic messages generated by the *aterm* facility followed by their interpretations. Refer to the *HP-UX Reference* manual for error number (n) information.

`aterm: ASI busy`

Another aterm process is currently communicating with the device you want.

`aterm: begin input diversion`

The input diversion command was successfully processed.

`aterm: begin output diversion`

The output diversion command was successfully processed.

`aterm: can't complete modem connection`

This diagnostic appears only if you have a full-duplex modem connection. One cause is an open link timeout, which can occur on a switched service connection if the incoming call (in the case of an autodial modem) is not completed within one minute.

`aterm: can't open <file_name>`

The config file or device file could not be opened.

`aterm: can't open pipe`

A system error prevented aterm from establishing a link between its two processes.

`aterm: can't reconfigure device address`

The **da** parameter may appear only in the config file.

`aterm: can't reconfigure host name`

The **hn** parameter may appear only in the config file.

`aterm: can't reconfigure modem control`

The **mc** parameter may appear only in the config file.

`aterm: can't reconfigure service type`

The **ss** parameter may appear only in the config file.

`aterm: connected`

The emulator has completed the initialization process and is now ready to accept data and commands.

`aterm: data piping terminated`

The pipe to the aterm co-process was unexpectedly closed.

`aterm: device file name too long`

The name you used for your device file contained more than 31 characters.

`aterm: end of file`

The end of file was encountered during an input diversion. The diversion is terminated.

`aterm: error` **n** during input diversion
`aterm: error` **n** during output diversion

Error **n** indicates the error that occurred.

`aterm: error reading configuration file`

Either the config file does not exist or is not accessible.

`aterm: error` **n** reading remote line
`aterm: error` **n** writing remote line

Line error **n** occurred.

`aterm: error reading stdin (keyboard)`

There is a serious problem with stdin.

`aterm: gen i/o read reg` **n** `failed, errno` **e**
`aterm: gen i/o write reg` **n** `failed, errno` **e**
`aterm: gen i/o read str reg` **n** `failed, errno` **e**
`aterm: gen i/o write str reg` **n** `failed, errno` **e**

The device you specified is probably incorrect. You specified a driver other than 19 with mknod.

`aterm: host system name too long`

The name used for the host system contained more than 31 characters.

`aterm: input diversion terminated`

You manually terminated an input diversion.

`aterm: invalid data rate`

The only acceptable values are: 50, 75, 110, 134.5, 150, 300, 600, 900, 1200, 1800, 2400, 3600, 4800, 7200, 9200 and 19200.

`aterm: invalid host echo`

Only the '+' or '−' is valid.

`aterm: invalid input mode`

Only 'l', 'c' or 'b' is valid.

`aterm: invalid local character set`

The only permitted values are: `usa, f, d, gb, e, s, i, dk and ec.`

`aterm: invalid modem control`

Only '+' or '−' is valid.

`aterm: invalid number of data bits`

The only acceptable values are: 5, 6, 7 and 8.

`aterm: invalid number of stop bits`

The only acceptable values are 1, 1.5 and 2.

`aterm: invalid output character gap`

The only acceptable values are in the range 0 through 254.

`aterm: invalid output mode`

Only 'c' and 'l' are valid parameters.

`aterm: invalid parity`

The only accepable values are: n, 0, 1, o and e.

`aterm: invalid prompt handshake`

Only the '+' or '−' is valid.

`aterm: invalid prompt timeout`

The prompt timeout must be a number in the range 0 through 600.

`aterm: invalid remote character set`

The only permitted values are: `usa, f, d, gb, e, s, i, dk, and ec.`

`aterm: invalid service type`

Only 'a' or 'o' is valid.

`aterm: invalid signal rate selector`

Only the '+' or '−' is valid.

`aterm: invalid terminal/host ENQ/ACK (XON/XOFF)`

Only '+' or '−' may be specified.

`aterm: keyboard data ignored during input diversion`

The emulator accepts only local commands during an input diversion. These local commands include display status, terminate diversion and change communication parameters. If you attempt to send normal data from the keyboard to the host computer during an input diversion, the emulator ignores the data and displays this message.

`aterm: keyboard data ignored while awaiting prompt condition`

This is a warning of an error in transferring data, i.e., unrecognized handshake, involving the loss of data.

`aterm: MUX busy`

Another aterm process is using the MUX.

`aterm: open of <file_name> failed, error n`

Aterm could not open the specified file. For example, if you should get an `Error 6` this usually indicates that the serial driver is not installed as described in the section, "Asynchronous Terminal Emulator" of the chapter, "Software Configuration" in this manual.

`aterm: output diversion terminated`

You manually terminated output diversion.

`aterm: read pipe error`

An error occurred communicating with the co-process.

`aterm: terminated`

The emulator recognized a termination condition.

`aterm: unknown command name`

Aterm does not recognize the command you entered.

`aterm: use '~~...' to send line starting with '~'`

The emulator uses this diagnostic in response to an unrecognized local command. If you intend to send a line starting with the local escape character to the host computer, you should re-enter the line, remembering to repeat the first character.

`aterm: waiting for modem connection`

The emulator issues this message to indicate that an open link is being attempted on a connection you have configured as a full-duplex modem. If you specified the switched service type as manual call or originate, you have one minute to complete the connection. If you specified auto-answer, then the one minute interval does not begin until an incoming telephone call is received.

`usage: aterm cfile`

You omitted the configuration file when you invoked the emulator or have included additional unrecognized options.

Uucp File System 5

There are three HP-UX directories which contain the files used to implement the *uucp* facilities:

- */usr/spool/uucp* (the spool directory for background processing);
- */usr/bin* (the directory with executable command files);
- */usr/lib/uucp* (the library of programs which only uucp uses and configuration files).

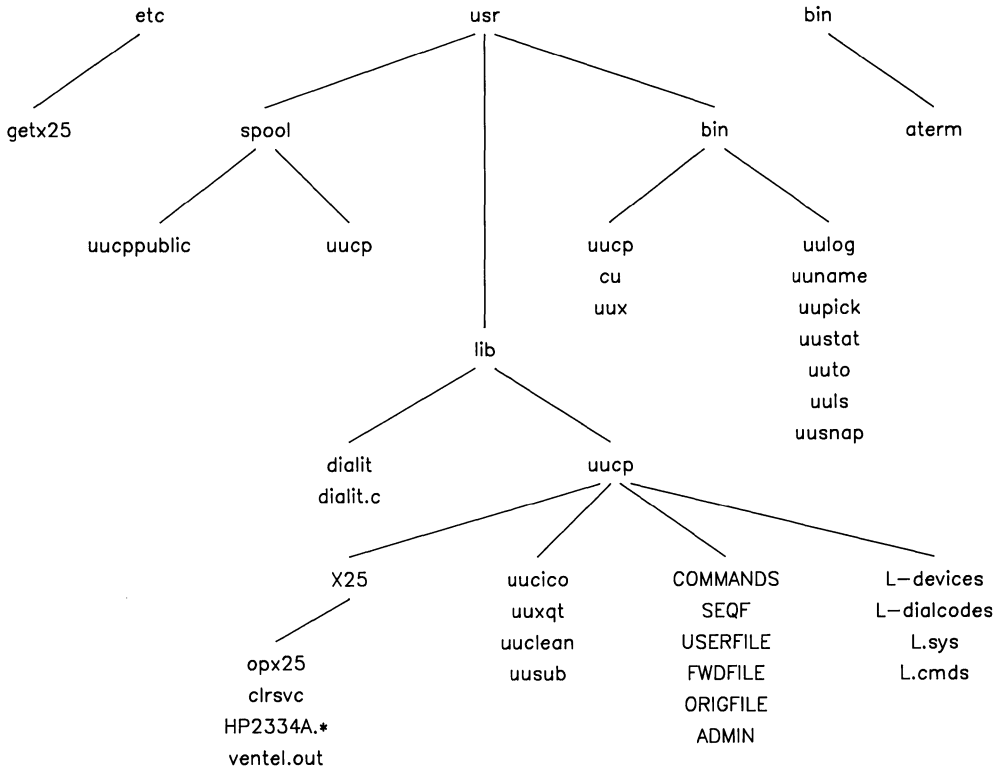


Figure 5-1. Files in Directories

Examples of uucp Data Transfer

The next three examples illustrate the dynamics of the *uucp* data transfer. **Please refer to these examples when reading about the *uucp* file system.**

uucp source_file dest_file local remote

This example shows the transfer of one file from a local system to a remote system.

1. The local system checks its `USERFILE` to verify that the user has access to the source_file and that this file is readable by everybody.[1]
2. A workfile (`C.`) is setup in local spool directory.
3. Your terminal displays a prompt indicating that you can now perform other operations.
4. *Uucp* checks the local `L.sys` file for information about connecting to the remote system.
5. *Uucp* checks the `L-devices` file to determine whether this is a valid device with connection and speed matches.
6. The remote system and device line are locked using `LCK.` files.
7. If this is a modem device, the `dialit` program executes a dialing routine.
8. *Uucp* attempts to login on remote system according to `L.sys` file information.
9. The remote system checks its `USERFILE` to determine whether your local system should be called back to verify your identity — assumed not required.
10. The local system now sends a request for work (one line of `C.workfile`) to the remote system.
11. The remote system checks its `USERFILE` to verify that your local system can access to the remote dest_file (If the destination file already exists, it must be writeable.).[2]
12. The source_file is sent to a temporary (`TM`) file on remote (If transmission proceeds without error the `TM` file is copied into the remote dest_file.).
13. The local and remote systems disconnect.

---

[1] When the dest_file is on the local system, the local system checks its **USERFILE** to verify that the local user has access permission to the dest_file/path if it already exists and that the dest_file is writeable. A dest_file is created if none exists.

[2] When the source_file is on the remote system the remote system now checks its **USERFILE** to verify that the local system had access permission to the source_file/path and that the source_file is readable by everybody.

uucp -C source_file dest_file local remote

This example illustrates multiple transfers from both the local and remote systems. A **user** on the local system initiates the *uucp* command. This example differs from the first one in that the callback option as well as multiple work orders on both systems are discussed.

1. The local system checks its `USERFILE` to verify that the user has access to the source_file and that this file is readable by everybody.[1]
2. A workfile (`C.`) and a source_file (`D.`) are put on spool directory.
3. The local user gets terminal back, *uucp* now functions in the background mode; the local system is the master.
4. *Uucp* checks the local `L.sys` file for information on how to connect to the remote system.
5. *Uucp* checks the `L.devices` file to determine if this is a valid device and for connect/speed match.
6. *Uucp* locks the remote system and device line using `LCK.` files.
7. If this is a modem device the `dialit` program now executes a dialing routine.
8. *Uucp* now logs into the remote system according to the information in the `L.sys` file; the remote system is the slave, the master waits for slave to acknowledge that the slave is here — "`Shere`".
9. The slave (remote) checks its `USERFILE` to see if the master (local) should be called back to verify its identity. If call back is required, the slave signals the master to hang up. The master disconnects and "dies". The slave changes to the master role and initiates a call to the old master.
10. The master sends a request for work (`S`, `R`, or `X` line of the `C.workfile`) to the slave.
11. The slave checks its `USERFILE` to verify that master's system_name can access to the dest_file (If the destination file already exists, it must be writeable.);[2] If access is not permitted the slave sends a "`NACK`" and the master puts a "Remote access to file/path denied" message in the `LOGFILE`.

---

[1] When the dest_file is on the local system, the local system would check its `USERFILE` to verify that the local user has access permission to the dest_file/path if it already exists and that the dest_file is writeable. A dest_file is created if none exists.

[2] When the source_file is on the remote system, the remote system now checks its `USERFILE` to verify that the local system has access permission to the source_file/path and that the source_file is readable by everybody.

12. The slave sends the acknowledge (`ACK`) message, "`SY`", and master's source_file is sent to the slave's temporary (`TM`) file (The master transfers the bits in 64 byte packets. The slave retrieves the packets, checks their checksum and puts the data in the `TM` file in the `/usr/spool/uucp`. The slave must `ACK` each packet; if the master does not receive the `ACK` from the slave in a specified time interval, the master retransmits the packet. This is done up to five times. If no `ACK` is received the transmission is aborted. If transmission proceeds without error the TM file is copied into the slave's dest_file and the master deletes the `D.*` file if one exists.).

13. The master checks for additional work.

    If there is additional work go to step 10.

14. The master sends a hangup message, "`H`" to the slave who then checks its spool directory for `C.*` files for master.

    If there is work on slave for master, the slave sends a hangup no, "`HN`", message to the master, the master and slave change roles, and go to step 10.

15. The slave sends a hangup yes, "`HY`", message to the master and they both disconnect.

uux command_string

This example illustrates the *uux* command sequence of actions:

1. A workfile (`C.*`) and two data (`D.*`) files are setup in the local spool directory. One data file has an X grade and will become an execution file on the remote system. The other data file(s) contains any file(s) the command requires.

   Steps 2 through 13 are the same as in the second example.

   Note that when a request for work is sent to remote system; the `D.aaXbb` becomes an execution file (`X.*`) on the remote spool directory. Any other data files are also transferred. At this point both systems can disconnect and the execution demon uuxqt starts.

14. The remote system checks its `L.cmds` file to verify that your local system may execute the specified command_string on the remote system.

15. The remote system executes the command.

16. The remote system notifies your local system by mail of the execution status.

Spool Directory

The Public Area

The public area, `/usr/spool/uucppublic`, is the area with general access privileges. This area can be used to receive files from a remote system which does not have access to any specified path on your system. Each time you use the "`sys_name!~/file_name`" as your path name, the system stores "`file_name`" in "`/usr/spool/uucppublic/file_name`" on the remote system "`sys_name`".

The uucp Directory

The `/usr/spool/uucp` directory contains work files, data files, log files and system status files.

At the time of initial installation of your system, this directory is empty. These files are automatically created when the *uucp* facilities are used.

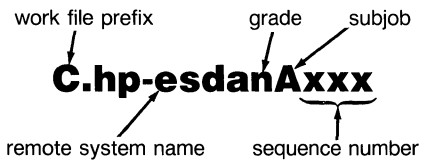
Work Files

Work files have a `C.` prefix and are work orders to copy data files.

When you use *uucp* or *uux* commands or the remote *mail* command these work files are automatically created. A child process of the parent *uucp* scans the spooler directory and in chronological order, taking the oldest work order first, processes whatever is asked for in the background mode.

Workfile names contain the information indicating which systems must be contacted to perform the work requested. The following figure shows the general form of a `C.workfile`.

Workfile Name



where:

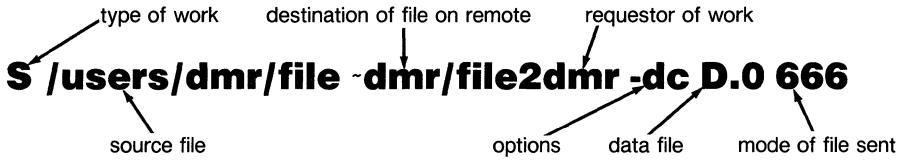
C.	is always the work file prefix;
remote system name	is the remote system to contact (The name may be a maximum of seven characters.);
grade	is a work sequencing mechanism (The higher the grade the sooner the work is done since work files are processed in alphabetical sequence. The highest grade is A and the lowest is z.);
sequence number	is the job number associated with the work file and is assigned by the system (The sequence number is used with the uustat command.).
subjob	is the character used to differentiate among files with the same sequence number. These are subjobs of the request.

You can alter the grade by using the `-g<grade_desired>` option with the *uucp* command. Workfiles created by *uux* command have a grade of "A" since command execution is a higher priority than file transfers.

Whereas the workfile names tell the *uucp* facility which system to contact, the workfile contents tell the facility exactly what work must be done. The contents are broken down into eight fields, as explained below.

The workfile may contain one or more lines of the following form (Only seven fields are shown.):

Workfile Contents



where:

type of work can be the following:

S — end a file from your system to the desired remote system;

R — copy a file from the remote system onto your local system;

X — send a request to the remote system for processing (The remote generates the C file with S lines specifying file(s) for processing.).

source file is the source file of the copy in either direction. The path name on a line with an X type of work does not have an explicit file name since the remote system is sending the file. When you have an R type of work the path name includes a file name on the remote system, but does not necessarily include the whole path name.

destination is the destination of the file copied. R types of work have local destinations, while S or X types of work specify a copy to a local or remote location. Note that the destination is expanded on either the local or remote to the login directory if you use "~".

user is the login name of the user who requested the work.

options is a list of command options and begins with a minus sign (-). If you use the -dc options (these are the the default options) you specify with the "d" that the system make any directories needed on the destination and with "c" that the system use the source file(s) to copy from. A "C" option indicates that a copy of the source file exists as D. file in the /usr/spool/uucp directory.[1]

data file is the data file which is copied. If you chose a "c" option then this data file should be D.0; if you chose the "C" option the file is `D.*`. Note tha a D.0 data file uses the source file current at the time of transfer background may involve a delay) while a `D.*` data file uses the source file current at the time of the request.[1]

mode is the mode of the source file sent in an "S" type of work. This can have read, write and/or execute mode capabilities.

The eighth field only exists if an "n" exists in the option list. This option causes a user on a remote system to receive mail when a file for him has arrived. This field holds the login name of the user who is notified.

[1] If you want to modify a file while an original copy is transmitted across the network, use the -C option. This forces a copy of the file to be queued on the spool directory to await its turn for transmission. The -c option takes a copy of the file only when it is time to transmit that file. The default option is -c.

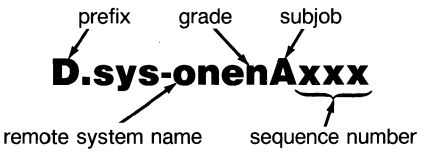
Each work file can contain up to 20 entries (lines beginning with "X", "S", "R" or some combination).

Data Files

Data files begin with the prefix "D.". There are two types of data files: image and execution.

Image Data Files

Image Data File Name



where:

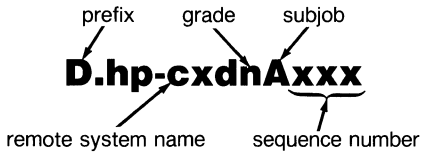
prefix	is always "D.".
system name	is normally the name of the remote system the file is destined to if the `D.*` file is a "copy" of the file on your system (If the data file to become an execution file on the remote then the system name field is your local system and the grade field contains an "X".);
grade	is again the work sequencing number. See grade under work files; (The grade of these files is usually "n" if generated by a uucp command or "B" if generated by a remote mail command.)
sequence number	is a four digit number is a job number associated with the data file.
subjob	is the character used to differentiate among files with the same sequence number. These are subjobs of the request.

The image data file generated with the -c (default) option for the *uucp* command, D.0, does not really exist, since the data is gathered at the time of transfer. The image data file generated with the -C option contains the copy of a file at the time the uucp command was invoked.

Data Execution Files

The data execution files contain the information necessary to execute a command on the remote system. This command is requested locally by a *uux* command or remote mail. The following figure illustrates the data execution file name fields.

Data Execution File Name



where:

prefix	is always "D.".
system name	is the name of the local system where the file was generated.
grade	is always "X" denoting a data file which when transferred to the remote system becomes an execution file.
sequence number	is again the job number associated with the data file.
subjob	is the character used to differentiate among files with the same sequence number. These are subjobs of the request.

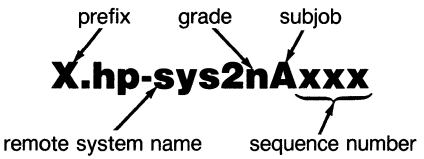
Data execution files become execution files (`X.*`) when they are transferred to the remote system. As with all data files, their contents remain the same, only their name is changed. The next section describes the contents of these files.

Execution Files

The execution files found on the spool directory, */usr/spool/uucp*, are the product of data execution files which were copied from another system. These files are created by the *uucp* program when the request for work is transferred.

The following figure illustrates their fields.

Execution File Name



where:

prefix	is always "X.".
system name	is always the name of the remote system which initiated the transfer.
grade	is X for execute.
sequence number	is the job number associated with this file.
subjob	is the character used to differentiate among files with the same sequence number. These are subjobs of the request.

A typical execution file would contain these five lines:

- a user line (has a "U" prefix);
- a required file line (has a "F" prefix);
- a required standard input information line (has an "I" prefix);
- a required standard output information line (has an "O" prefix);
- a command line (has a "C" prefix).

F,I and O lines are required **only** if their respective files are required for the command execution.

Examples of the general form of the line, a specific example and then a discussion of the line fields is next. The line prefix is not included in the discussion but is shown in the line examples.

User Line

```
U user source_system
U kls hp-dcx
```

where:

`user`	is the user login name for the user on the remote system who issued the command;
`source_system`	is the name of the remote system where this execution file originated.

There should be only one "U" line per execution file.

NOTE

You can route an execution file through intermediate nodes, however the information on its origin is lost. The last intermediate system routed through and the login used for uucp are contained are shown on the user line in this file.

Required File Line

```
F required_file <source>
F D.hp-dccB278
```

where `required_file` has two fields:

- data file name as it should appear in your local */usr/spool/uucp* directory;
- source from which the data file name above was copied.

An execution file may contain zero or more F lines.

Standard Input Information Line

```
I D.file_for_standard_input
I D.hpdccB278
```

where `file_for_standard_input` is used if the original command used the standard input file for its parameters or if a redirection is specified.

If the remote *mail* command used the standard input file for its contents you would use this line.

There may be zero or one of these lines.

Standard Output Information Line

```
O file_for_standard_output system_where_directed
O /dev/lp hpdcdx
```

where:

`file_for_standard_output`	is the file or device which is to be used for output;
`system_where_directed`	is the system name of the file or device used.

NOTE

You can send the output to a different system than the one on which the command is executed by using, for example:

```
uux "sys2!pr sys1!file > sys3!/dev/lp"
```

The execute system is sys2 while the final output system is sys3. The proper access is necessary for both systems: sys1 must have access to the /dev/lp on sys2 and sys2 must have access permission to the /dev/lp on sys3. It is usually more convenient to have the system which executes the command also contain the output file.

Command Line

```
C command arguments
```

where:

`command`	is the command to be executed;
`arguments`	is an optional field and may consist of any options or file names the command supports. For a further discussion of the command parameters refer to the *HP-UX Reference* manual.

The command **must** exist in the remote command security file `/usr/lib/uucp/L.cmds`. The command must also exist in the `/usr/bin` or `/bin` directories unless an explicit and fully qualified path name is given. The system automatically searches the `/usr/bin` and `/bin` directories when you issue a command so you do not have to explicitly give the whole path name for the command.

If you do use an explicit path name for an *uux* command, that path name must exist in the `/usr/lib/uucp/L.cmds` file.

Typical Execution File

An example of the contents of a typical execution file is:

```
U dmr hpcnoa
F D.testsysA1569 DIALLOG
F D.testsysA1570 LOGFILE
F D.testsysA1571 LOG-WEEK
C pr DIALLOG LOGFILE LOG-WEEK
```

Note that the `DIALLOG` source file for the first required file `F` line is also one of the `C` command line arguments.

Lockfiles and Temporary Files

Lockfiles are created to provide you with exclusive access to a system while you are using it. When you want to copy a file to or from a remote system lockfiles lock out any other system from trying to communicate with that remote system. When a log file is being updated, locking that file assures that no overwriting takes place.

Examples of lockfiles follow:

`LCK..hpdsd`	lock call to system hpdsd; conversation in progress;
`LCK.LOG`	lock LOGFILE for making an entry;
`LCK..cu104`	lock /dev/cu104 while conversation is going on;
`LCK.SQ`	lock the SQuence file while the sequence number is being updated.

If one of the *uucp* programs is aborted abnormally, some lockfiles may exist and prevent future communications with *uucp*. Delete them with the */usr/lib/uucp /uuclean* program. This program is discussed in the chapter, "Log, Status and Cleanup Information".

The *uucp* programs use temporary files to hold data being received until the entire transmission has completed without error. The temporary file is then copied into the destination file and then this temporary file is automatically deleted.

Examples of temporary (TM) file names are:

`TM.<pid>.<count>`

`TM.00216.001`

`LTMP.*` (Used and cleaned up internally by uucp programs.)

`dummy`

Files Recording Information

The log summary file, named `LOGFILE`, is used to record all *uucp* connections either local or remote, the files transferred, the completion or failure of the transfers, the success or failure of an autodial attempt, and the status of the *uux* commands. It is also used to record at what time transfers take place.

The `SYSLOG` file is used to record the number of bytes sent or received from a system and the number of seconds used in the transfer. This file is used to report traffic statistics between connections.

The `DIALLOG` file records information about the modem used, the telephone number dialed and the result of the dialing. An example of a `DIALOG` file is in the chapter, "Log, Status and Cleanup Infomation".

The `ERRLOG` file records information about any errors encountered in the communication processes.

Binary Files

The directory `/usr/bin` contains these command files: `cu`, `uucp`, `uux`, `uulog`, `uuname`, `uupick`, `uustat` and `uuto`. (The *aterm* program is in the `/bin` directory.)

Every time you issue one of these commands the system looks in the `/usr/bin` directory for an executable file with the appropriate command name and executes it.

Library Files

When the *uucp* programs were installed in their proper directories, several files which supply remote connection information, were created in the `/usr/lib` directory.

The System Administrator must edit each of these files except `SEQF` to add the information obtained earlier from the remote systems contacted and the information specific for the local system needs.

The directory `/usr/lib/uucp` contains program modules which the *uucp* commands need to use. These program modules initiate and carry on all communications with remote systems and perform the remote execution of commands.

The files, with their total path names, are:

`/usr/lib/uucp/L.cmds` — list of allowed uux commands;

`/usr/lib/uucp/FWDFILE` — list of systems your system can forward through;

`/usr/lib/uucp/ORIGFILE` — list of originating systems that can forward through your system;

`/usr/lib/uucp/SEQF` — keeps track of local sequence numbers;

`/usr/lib/uucp/USERFILE` — gives protection information for local users and remote systems;

`/usr/lib/uucp/L-devices` — defines devices and types of connections possible;

`/usr/lib/uucp/L-dialcodes` — contains strings that are abbreviations for telephone number prefixes;

`/usr/lib/dialit` — contains modem dialing sequences;

`/usr/lib/dialit.c` — C language source file for dialit;

`/usr/lib/uucp/L.sys` — contains information concerning what systems your system knows about and how to make a connection;

L.cmds File

The `L.cmds` file determines which commands can be executed from remote machines using *uux* on your local system. This file can be edited to add new commands or to delete old commands.

This is an example of the contents of a `L.cmds` file:

```
rmail,node1,node2,node3
pr,node1
col
lp,node12
```

limiting the remote execution to these four commands. Note that there are system nodes given with the above commands. This is so different system nodes can be specifically given access to the individual commands. Where there is no system node mentioned that command can be used by all systems with access to your system.

For security reasons, you probably want users on other systems to only be able to send users on your system *mail*; you do not want remote users to read your local system *mail*. The `rmail` parameter permits the execution of the receive mail command on your local system.

Only one command is permitted per line. The complete path name does not have to prefix the command if it resides in `/bin` or `/usr/bin`. If not, the full path name must be given, for example:

```
/BIN/my_command
```

indicates that the command, my_command, found in the `/BIN` directory is a valid command.

All commands found in the `L.cmds` file are executable by all remote systems if they have been assigned special permission to use these commands.

SEQF and SQFILE

The `SEQF` file is used by the *uucp* programs to record the general sequence numbers used in the work file and data file names. You **do not** need to manage this file.

The `SQFILE` was not created automatically when the *uucp* facility was installed; it must be created by the System Administrator. The sequence number records the number of transactions between your local system and a remote system. Each remote system that you want to implement sequence checking with must have an entry in the `SQFILE`. For example, to initiate sequence checking the `SQFILE` on `sys1` has an entry for `sys2` and correspondingly the `SQFILE` on `sys2` has an entry for `sys1`:

```
SQFILE on sys1        SQFILE on sys2

    sys2                  sys1
```

Thereafter each time a transaction is made, the sequence number in the `SQFILE` on both systems is incremented. The `SQFILE` is used for explicit sequence checking between remote systems. These numbers must match before a transaction can be made.

For example, when you attempt to make a connection to a remote system and the sequence check fails, a message similar to the following is given:

```
dmr hpfcla (5/23-9:37-24748) HANDSHAKE FAILED (BAD SEQ)
```

The `BAD SEQ` message indicates a bad sequence number and that the two systems are out of "sync".

Sequence files are used as security measures to ensure that *uucp* transactions are with the specified remote machine. Both machines keep a record of the name of the remote machine, a count of the number of transactions and the time of the most recent transaction. These files are updated after every transaction and compared. If the files on the two machines disagree, the connection is terminated and one of the files must be corrected manually to bring them back into agreement.

USERFILE

The `USERFILE` specifies the type of access permission that is granted to both local users and remote systems; this is the major security tool for the *uucp* facility. **Please encourage** all users to read this section and be sure they understand it.

There are three areas where you can implement security for your system and files. You can:

- ask that a remote system be called back to verify its identity;
- check the login name for a user;
- restrict file access paths to certain systems.

A `USERFILE` line entry has four main fields:

```
<user>,<system_name> [c] <path_name>
```

There must be a blank or tab between the `user,system_name` field, the `c` option if used and each `path_name`.

where:

`user`	associates the access permission for a named user on your local system or is used to determine whether a call back is required for the remote system;
`system_name`	determines the access permissions for a remote system logged into your local system;
`[c]`	is an optional field indicating that the remote system logged in as `user` should be called back (When the remote system tries to log into your local system, the remote system is called back to verify its identity.);
`pathame`	is a list of path names separated by blanks (This is the critical list which gives the `user` or `system_name` access along the specified paths. If the path name ends at a directory, all files and sub-directories in that directory are accessible.).

At the time you install your system `USERFILE` contains:

```
uucp, /
, /
```

which provides unlimited access for all users on your local system and for all remote systems logging into your local system. You can restrict the access by editing this file.

The following example shows the contents of a typical `USERFILE`:

```
dmr,hp-sys1    /users/dmr /dev/null
emd,hp-sys2    /usr/spool/uucppublic /dev/null
kls,hp-sys3 c /
uucp,          /usr/spool/uucppublic /dev/null
,             /usr/spool/uucppublic /dev/null
```

The fourth line gives *uucp* users on your local system and all remote systems not previously specified access to the two specified paths: `/usr/spool/uucppublic` and `/dev/null`. The `/dev/null` is the default input and output file for the *uux* command. The last line gives all users on your local system not previously specified access to the same two paths. You probably want to include these permissions in your `USERFILE`.

The following lists show the searching sequence for `USERFILE`:

USERFILE SEARCH
(remote system)

1. Check the `user` field. In order for a remote system to log onto your system, its *user* field **must** appear in the system's *USERFILE*. It is also recommended that you have the remote system's *system_name* field in your *USERFILE*, but it is not necessary. Note that if the remote system when communicating with your system has the correct *user* field but the *system_name* field was left blank, then there **must not** be a line before it in your *USERFILE* that contains the correct *system_name* field and the incorrect *user* field, as permission for the remote log in will be denied. For example, when a remote system calling you has a *user* field of uucp and a *system_name* field of `remote1` and your *USERFILE* contains the following:

   ```
   test1,remote1 /path_name /path_name
     uucp,     path_name
   ```

   then login permission will be denied to the remote system. In this example, your system tests the remote system's *user* field (uucp) and finds it is incorrect; however, the remote system's *system_name* field (`remote1`) is correct and your system sets a flag and continues to search for the correct *user* field. When the correct *user* field is found and *system_name* field is blank, log in permission is denied to the remote system.

2. Check the `system_name` field for the system access to the path/file for each file transfer request.

   [If no `system_name` match is found, use the first blank or null `system_name` field if and only if that line is not the same line used for a blank user field.]

USERFILE SEARCH
(file on local system)

1. Check the path/file access for local user permission.

   [If no user match is found, use the first blank or null user field.]

2. Check to verify that the path/source_file is readable or the destination_file path is readable and the destination file is writeable.

The `USERFILE` is searched sequentially for the `user`. If the search does not find the `user` the **first** entry encountered with a null user entry is employed. **Be cautious**, the first `user` entry that is null defines the access permission for **all** users who are not specifically named. If you put a null `user` field before some named `user` fields, the search finds the null field first and stops searching; the named users after the null user field are never searched. When a match or null field is found **and** the user is a remote system, the *uucp* program checks to see whether a call back is required. If so all activity stops until the remote system is called and its identity verified. If the user is on your local system, the *uucp* program checks the `USERFILE` for path names that user is granted access to. If no match or null field is found for the user an access denied message is generated.

The next field sequentially checked is the remote `system_name` field. This field specifies access permission for remote systems to any path_names on the match or null system_name line. Note that all users on a specific remote system are restricted to the same path(s). If the remote system_name does not have permission to either read a source file or write a destination file, an access denied message is again generated.

Note that the `user` and the `system_name` fields are divorced from each other. The *uucp* facility starts its search process at the first line of the `USERFILE` for **each** field. For remote systems the user field is only searched for the call back option. The *uucp* program starts at the beginning of the `USERFILE` to search for a `system_name` field to restrict remote systems to the paths specified on that line. The users field for `local users` is combined with the paths described on the specified line to restrict access to that path.

Although there may be several lines with the same system name, the *uucp* program must find either the name of the remote system or a null system name on the system name line before the information transfer is permitted.

Users on a local system must also have permission to access their own files when using *uucp* facility commands!

When the *uucp* program reaches a null or blank user field before finding the user name wanted, that line is used for local user path_name access or remote system call back. If the *uucp* program reaches the same line in its system_name search and finds a blank or null system_name, that line **cannot** be used to grant remote system access to the path_names specified. *Uucp* continues its `system_name` search on succeeding lines. For example:

```
,        /dev/null
,hpsys1 /
```

gives `hpsys1` access to all paths on your local system. It **does not** give all unmatched users on all unmatched systems access to the path `/dev/null`.

When the *uuxqt* demon encounters an `X.*` file in the `/usr/spool/uucp` directory, the `L.cmds` file is checked to determine whether the command can be executed by **all** remote systems. The `USERFILE` is then searched to find the first **null** system field for path access permission.

L-devices File

The `L-devices` file defines the devices and types of connections valid for **both** the *uucp* facility and the *cu* terminal emulator. Each entry describes a connection type, the `/dev` entries for the connection and the speed at which the connection is made.

When this file is automatically created at installation time, the file contains the following lines as examples for you:

```
<type> <cul> <cua> <speed> <proto>
DIR tty04 0 9600
DIR tty12 0 9600
ACUVADIC3450 cul03 cua03 1200
DIR tty09 0 9600
DIR tty09 0 1200
ACUVENTEL212 cul06 cua06 300
```

where:

`type` denotes the type of connection (This may be a direct connection indicated by `DIR` or an auto dial modem connection, automatic calling unit (`ACU`) indicated by the string, `ACUmodem_name`, the name of the autodial modem in use. The maximum length of the string is 19 characters. This modem connection must have an autodial routine in `/usr/lib/dialit.c`);

`cul` is the /dev entry name for the main data line you specified when you used the mknod command (This line is used to transmit all data once the connection is made with the remote system. It is recommended, but not required, that you use an entry_name with a `cul` prefix for dialout lines and tty for dialin lines.);

`cua`	contains a zero, `0`, if the line refers to a direct connection (`DIR`) or contains the name of a `/dev` entry_name if the line refers to a modem connection (`ACUmodem_name`) (It is this line which is passed to the dialit routine.);[1]
`speed`	specifies the speed of the communications line associated with the `L-devices` entry (The *uucp* facility allows speeds of 300, 1200, 2400 4800 and 9600 baud. The speed you choose depends on the restrictions of the remote system.);
`proto`	a single letter specifying the protocol to be used on that line.

You need to to define your own direct and modem connection devices by inserting appropriate entries similar to the examples given.

Order is important when using the *cu* command, for example:

```
cu -ltty09 -s1200 dir
```

causes the system to look at its direct connection nodes for `tty09`. In the above list of `L-devices` contents, the first matching entry specifies 9600 baud, so the line connection is opened at 9600 baud, even though you explicitly stated 1200 baud speed, `-s1200`. The `-s` speed option has no effect with *cu* if a direct connection is specified; once a rate is found it can only be changed using the following command line:

```
~!stty 1200 < /dev/tty09
```

This order restriction does not apply to the *uucp* command.

L-dialcodes File

The L-dialcodes file specifies the telephone prefix translations. Each entry in this file contains two fields: an identifying string and the string number you want substituted when you use the identifying string, for example:

```
boston 131-149
```

When you want to use a modem connection to contact a remote node, your system looks in the L.sys file for the phone number of that remote node. If you have used the string, `boston`, in the phone field for that remote system, your local system then looks in the `L-dialcodes` file for the translation of `boston` into an actual telephone number, `131-149`.

These abbreviations for telephone number prefixes can reduce some time, typing and mistakes.

---

[1] The entries for **<cul>** and **<cua>** may be the same if you have only one physical line connection. You may also have two entry_names in the **/dev** directory with the same select code for dialout lines

Dialit.c

The `dialit.c` file is the C language source of the `dialit` module used by *uucp* to perform the autodialing needed to contact a remote system with a modem connection. This file was installed in the `/usr/lib` directory when you used the *optinstall* command.

Four example dialing routines are supplied for you in `dialit.c`. You may need to edit these routines or modify the USER-SUPPLIED-ROUTINES section for your specific needs. Comment lines in `dialit.c` and the following chart will help you modify these routines. The procedure headers in `dialit.c` contain information about modem configuration.

For this modem	Use this name in the L.sys device field and the L-devices type field
VENTEL 212	ACUVENTEL212
VADIC 3450	ACUVADIC3450
HP 35141A	ACUHP35141A
HP 92205A	ACUHP92205A
HAYES	ACUHAYESSMART
HP37212A	ACUHP37212A

The programs contained within this module are examples of autodialing routines for selected modems currently on the market. The Hewlett-Packard Company makes no claim as to the validity or reliability of this code. These programs are not supported products, but simply examples for our customers. Their compatibility with future products is not guaranteed.

In certain areas, especially those with tone dialing, the numbers may be dialed faster than the PBX can respond to. In this case, you should insert a "-" or any defined pause signal between the numbers dialed.

The following code is an example of part of the `dialit.c` program:

```
static char Uni_id[] = "@(#)14.1";
/* UNISRC_ID: @(#)dialit.c      14.1     83/05/01  */
/*************************************************************
 *    (c) Copyright 1983 Hewlett Packard Co.
 *        ALL RIGHTS RESERVED
 *************************************************************/
/*************************************************************************
 *           * * * * D I S C L A I M E R * * * *
 *  The programs contained with in this module are examples of autodial-
 *  ing routines for selected modems currently on the market. H.P.
 *  makes no claim as to the validity or reliability of the code in this
 *  module. These programs are not supported products, but simply examples
 *  for our customers. Their compatibility with future products is not
 *  guaranteed.
 *************************************************************************/
/*************************************************************************
 *  This module consists of:
 *     main routine - this routine is the main entry point into the module.
 *     The usage of this routine is:
 *               dialit (modemtype) (cua) (phone) (speed)
 *     Where:
 *        modemtype - is the name of a modem know in the Modem structure
 *                    along with a user supplied routine to do the
 *                    autodialing. The standard is for the modemtype to
 *                    be a name of the form:
 *                            ACUmodemname
 *                    such as:
 *                            ACUVENTEL212
 *                    This convention is followed since this is the form
 *                    expected by both uucp and cu which utilize this program
 *                    to perform their autodialing.
 *
 *        cua - This must be the full path name of the /dev entry over which
 *              the auto dial sequence is to be sent to the modem. In the
 *              case of uucp and cu this entry is pulled from the L-devices
 *              file. NOTE: that in the L-devices file the full pathname is
 *              not given. But uucp and cu do expand it before calling this
 *              module.
 *
```

```
*        phone - The phone number to be called by the autodial modem. The
*                phone number may consist of digits, '=' and '-' only.
*                The special characters are mapped to wait for secondary
*                dialtone(if implemented on the modemtype) or 5 second
*                pause respectively.
*
*        speed - This argument is the speed desired for transmission,
*                i.e. 1200,300,etc. The inclusion of this parameter
*                allows you to configure the cua line. If the dial
*                routine is called from cu or uucp the line has already
*                been configured.
*
*     sendstring routine - writes the designated string to the device
*          whose descriptor was sent it.
*
*     await routine - will read from the designated device a sequence of
*          characters until a certian string is recognized or a specific
*          number of characters is read.
*
*     ckoutphone routine - scans the phone string and checks for invalid
*          characters and determins a delay time used for alarm timeout
*          purposes when calling the remote machine.
*
*     map_phone routine - map the characters '=' and '-' which mean wait
*          for a secondary dial tone and pause respectfully to their
*          actual character representation for given modems.
*
*     log_entry - make an entry into the DIALLOG which resides in /usr/spool
*          /uucp.
*
*     make_entry - called by log_entry. makes the actual entry in the logfile.
*
*     prefix - tests a string to determine if it begins with a given prefix.
*
*     mlock - lock the logfile so only one process may write to it at a time.
*
*     remove_lock - remove the logfile lock and allows another process to
*          access the log.
*
*     close_log - cleans up any temporary log files created and closes the
*          log file.
*
```

```
 *   USERSPECIFIED ROUTINES:
 *    these routines are supplied by the users of the uucp package. Each
 *    routine is written for a specific type of autodialer modem and must
 *    have an entry in the Modems structure.
 *
 *   HISTORY OF MODULE:
 *
 *    15 NOV 82 - created initial source code for module including one
 *                modem support routine for the racal vadic 3450 modem.
 *
 *    17 DEC 82 - modules added which support the autodial feature for
 *                the ventel 212 plus autodial modem.
 *
 *    12 JAN 83 - add the mapping of '=' and '-' to respective characters
 *                for a given modem. This conforms to specifying telephone
 *                by the UNIX standard. Also correct the error in the
 *                routine ckoutphone by making the badphone parameter an
 *                address of an integer.
 *
 *    31 JAN 83 - added the modules to update, create and manage the logging
 *                of the autodial information.  Also changed the interface
 *                of this module to reflect the final version's form.
 *
 *    07 MAR 83 - modified the pause between the sending of (cr) to the
 *                ventel to be a busy wait, rather than a sleep. This is
 *                a much more reliable method for modem wake up.
 *
 *    22 APR 83 - added alarm around write to avoid possible hangs.
 ************************************************************************/
#include (stdio.h)
#include (termio.h)
#include (setjmp.h)
#include (sys/types.h)
#include (signal.h)
#include (ctype.h)
#include (fcntl.h)
#include (sys/stat.h)
#include (sys/dir.h)
#include (time.h)
#include (pwd.h)

#define FILENAMESIZE 15
#define MAXFULLNAME  100
#define MAXMSGSIZE 256
```

```
#define SAME     0
#define FALSE    0
#define TRUE    -1
#define FAIL    -1
#define SUCCESS  0
#define MAXRETRIES  3
#define PREFIX "DIAL."
#define LOG_LOCK  "/usr/spool/uucp/LCK..DIAL"
jmp_buf Sjbuf;
char *modemtype;              /* modem name as entered in the L-devices file
                              and L.sys file. */
int alarmtout();
/*************************************************************************
 *  The following structure Modems is used in determining which user
 *  supplied routine is to be used for autodialing given a specific
 *  modem type.  Each user specified routine must have at least one entry
 *  in this structure and each modem type used for autodialing must have
 *  only one entry in the structure.
 *
 *  To add additional modem types and routines simply add them to the
 *  initialization of modem[].
 *************************************************************************/
    int vad3450(),                    /* function name for vadic3450*/
       ventel1212(),                 /* ventel 212+3 function */
       hp35141_autodial(),           /* hp support link modem */
       hayes_smart();                /* hayes smartmodem1200 function */

    struct Modems{
          char *name;                   /* modem name */
          int (*modem_fn)();            /* function to call */
    } modem[] = {
          "ACUVADIC3450",     vad3450,
          "ACUHP35141A",      hp35141_autodial,
          "ACUVENTEL212",     ventel212,
          "ACUHAYESSMART",    hayes_smart,
          "ACUHP92205A",      hayes_smart,
          0,
    };
```

There are lines at the beginning of dialit.c which define the integer functions used and the module functions which can be called by the dialit program.

Locate the lines which define the integer functions and add your modem name, for example:

```
int hays1200;
```

Next define your modem connection by adding its name and function to the modem[] structure, for example:

```
"ACUHAYS1200",    hays120,0
```

Now locate the USER-SUPPLIED-ROUTINES section at the end of this module and add the code necessary for your specific type of modem. Note that the await routine now has a parameter specifying string length.

After you have finished modifying the `dialit.c` source code, you **must** compile the code and store the compiled version in the `dialit` file.

Dialit File

Note that the `dialit` path name does not include the *uucp* sub-directory name.

The `dialit` file contains the object code necessary to implement autodial sequences necessary for the specific modem you are using. When the *uucp* program checks the `L-devices` file and finds that the device for the remote system uses a modem connection, the `dialit` routine is called to perform the autodialing and an entry is made in the `DIALLOG` file.

Refer to the `dialit.c` section above for information on modifying the contents of the `dialit` file.

L.sys File

The `L.sys` file is used by the uucp facility to provide information on which systems can be contacted, when the remote system allows communication, if the connection to the remote system is direct or modem, the speed of the communications and how to log into the remote system.

The following is an example of what a `L.sys` file installed automatically might look like:

```
<sysname> <time>[,<retry>] <dev> <speed> <phone> <logininfo>
sys1 Any,1 tty04 9600 tty04 login:-EOT-login: uucp
sys2 Mo0800-1730,10 tty06 1200 tty06 login:-BREAK-login: access
sys3 Wk0800-0600 tty03 1200 tty03 login: network password: hpdcd
sys4 Any,5 ACUVADIC3450 1200 555-1212 login:uucp
sys5 Any,5 ACUVENTEL212 300 999-7777 login: sys5 ssword: uucp
sys6 Any,5 ACUHP2334A 9600 f/123456789 login: sys6 ssword: h9
```

where:

`sysname` is the remote name of the system whose contact name is contained on the entry line (This name may be up to seven characters. If you have alternate means of communicating with a certain system, you can have more than one entry in the `L.sys` file with the same sysname. This file is searched sequentially to determine if the requested system name matches a sysname. If you are a PASSIVE (receiving calls) system with respect to `sysname`, the remaining five fields are blank. Specifying the name permits queuing work.);

`time` gives the time of day as well as the days sysname allows communication (Days of the week: Su, Mo, Tu, We, Th, Fr, and Sa, Wk for week day or Any for any day of the week, are followed by the times permitted. The time should be a range of times in a 24 hour format, for example:

```
0800-0600
```

allows calls any time except between 6 and 8 a.m. and:

```
Su Mo Tu 0600-2300
```

allows calls Sunday, Monday and Tuesday between 6:00 a.m. and 11:00 p.m.

The default for time of day is all times are permitted.);

`[,retry]` is an optional field indicating the waiting time in minutes between a failed call connection to a remote system and the next retry (The default is wait 5 minutes; specified times that are less than 5 minutes default to 5 minutes.);

Note that this retry time is a **wait** time only; it does not specify that a retry at dialing will be attempted.

dev indicates whether a direct (**DIR**) entry or a modem (**ACUmodemname**) entry must be found in the **L-devices** file (This field is the same as the **<cul>** field of the corresponding **DIR** entry in **L-devices** file.);

speed specifies the speed at which communications take place (The union of the **<dev>**, **<speed>** and **<type>** fields are searched for in the **L-devices** file for the proper entry to use.);

phone is the telephone number of the remote system to call for login (For direct connections the phone field is the same as the dev field. The telephone number may contain a string, such as **boston which is searched for in the L-dialcodes** file and translated into the associated telephone prefix, **999**.

Samples of the permitted characters are:

"0" through "9" "=" wait for secondary dial tone; "-" pause for five seconds.

These are generic examples of the characters; your modem may have different characters which you must map to the above meanings in the dialit file.

A maximum length of the translated telephone number is 29 characters.

Since you have the ability to use *uucp* over two types of communication links, two protocols are provided to give efficient use of these links. The *f-protocol* is used with X.25 (see the chapter, "The X.25 Network") lines and the *g-protocol* is used with regular phone lines. Note that this *f* or *g* protocol character is prefixed to the phone number. For example,

```
f/226-3111
```

which says *f-protocol* is being used and

```
g/226-3111
```

says *g-protocol* is being used. If there is no protocol character used then *uucico* will default to *gf-protocol.*

You can have *uucp* protocol which specifies the use of both *f* and *g* protocol. For example,

```
fg/cul05
```

here, the **fg/** specifies that *f* or *g* protocol may be used, but *f* is given higher priority.

`ogininfo` is a field containing the information necessary for logging into the remote system. This field should contain the prompt you expect to receive from the remote system, followed by a space and then the response you are expected to give. For example, if you expect the prompt: `login` and your response is `sys5`, you then expect the prompt: `password` and your response is `abcxyz`, you would enter:

```
login: sys5 password: abcxyz
```

NOTE

After the connection to a remote system is made, the system may need to be "poked" before the first login message is received; this is most common on direct connections. If you use:

```
login:-EOT-login: sys5 password: abcxyz
```

The "-EOT-", causes a control-D to be sent to the remote system if the login message is not received in 30 seconds. You may also use "-Break-" to send a series of null characters or any string that has dashes on both sides to initiate the poke.

The login information is given as a series of fields and subfields in the format:

```
[expect send]
```

where `expect` is the string expected to be read and `send` is the string to be sent when the expect string is received.

The expect field may be made up of subfields of the form:

```
expect[-send-expect]...
```

where the `send` is sent if the prior `expect` is not successfully read and the `expect` following the `send` is the next expected string. For example, `login--login` expects `login`. If a `login` is received, the program goes on to the next field; if it does not get the `login` it sends `null` followed by a new line, and then expects `login` again.

When `L.sys` was installed, the protection mode, 444, was "readable by everybody". Since this file may contain proprietary information, you can change its mode to 400 so that only the owner, *uucp*, can read the contents. Be sure that *uucp* remains the owner since the *uucp* program needs to read `L.sys` to implement the data transfer.

Note that the **send** string may also contain:

\s	blank
\d	1 second delay
\c	no carriage return on the **send** string
\N	null character
\\	backslash
EOT	two control-D's
BREAK	send a break

For example, you may have a **logininfo** field similar to the following:

```
login:-\d\d\d@\c-login:XYZ ssword: Ply.
```

Uucp Facility Demons 6

The *uucp* demons are programs which perform the operations necessary for the *uucp* facility transfer of information. This chapter discusses the demons and their specific functions.

Uucp Facility Invocation

When you execute a *uucp* command two things happen:

- work files are set up in the `/usr/spool/uucp` directory
- a child process is spawned. The child process invokes the *uucp* communciations demon:

```
/usr/lib/uucp/uucico -r1
```

Uucico is the mnemonic for UNIX to UNIX copy in copy out. The **-r1** option specifies that the *uucico* act in the master role.

Uucico scans the `/usr/spool/uucp` directory for the work file with the highest grade. At least one work file must exist in the spool directory since the *uucp* command that spawned the *uucico* also set up a work file. (Unless *uucico* was started manually as in the example above.)

Next *uucico* examines the system names, either local or remote, which are part of the source and destination file names. If a remote system is specified, *uucico* looks in the `L.sys` file to determine how to contact the remote system. For modem connections, `L.sys` gives *uucico* the type of modem to use, the telephone number, the speed of data transfer and the login information. *Uucico* now lookes in the `/usr/lib/uucp/L-devices` file to determine if this modem device is a valid device. This `L-devices` entry must match the speed in the `L.sys` file and gives the information on which communication line (`/dev/line_entry`) that modem resides.

Uucico checks to see if another *uucp* facility is using the line. If not, *uucico* creates lockfiles in the `/usr/spool/uucp` directory for the line **and** for the remote system it is trying to call. These lockfiles implement a binary semaphore mechanism.

Uucico now spawns a child process which invokes the `/usr/lib/dialit` program to make the actual call to the remote system. Once online or connected, *dialit* returns a status to the parent *uucico* process.

Uucico now uses the login information in the `L.sys` file to attempt to login to the remote system. There must be an entry in the `/etc/passwd` file on the remote system of the form:

```
uucp::5:5::/usr/spool/uucppublic:/usr/lib/uucp/uucico
```

It is important to note two special things about this `/etc/passwd` entry:

1. The login directory for *uucp* purposes **must** be `/usr/spool/uucppublic`.
2. The `/usr/lib/uucp/uucico` demon **must** be invoked instead of the normal shell `/bin/sh`; if this is not done, communications can never take place.

At this point the *uucico* on the system that originated the call is the master since it was invoked with the **-r1** option. Another *uucico* is automatically invoked on the remote system which functions as the slave. The slave sends the master a message "Shere", meaning slave is here.

When the master receives the "Shere" message, as series of messages are sent back and forth to establish the handshake and communications protocol.

Please refer to the second example at the beginning of the chapter, "Uucp File System" for the conditions activating a reversal of master/slave roles.

Once this is complete, the master sends a request and waits for the approval of that request by the slave. If the request is approved, the actual transfer of the file begins. The file is broken down into 64 byte packets which include a checksum to guarantee that the packet is transferred without error. If a packet is not valid upon its arrival, it is re-transmitted up to five times. After the non-successful fifth try, *uucico* assumes a bad connection and breaks off communications.

Communications are re-established when another *uucico* is invoked.

The master continues transmitting requests to the slave until there is no more work for that system. The master then sends the slave a hangup request. When the slave receives the hangup request it scans its spool directory for any work files that have the master's (remote) destination. If none are found, the slave returns a hang-up OK message to the master and communications are broken off. If the slave does have work for the master, a hang-up denial message is sent indicating to the master that the slave has work to send.

At this point the roles of master and slave are switched. The new master starts sending requests to the new slave. When all work has been sent, communications are broken off.

Upon termination of communications, each *uucico* demon (master and slave) spawns a child process and dies. The child process invokes the `/usr/lib/uucp/uuxqt` execution demon. *Uuxqt* scans the `/usr/spool/uucp` directory for any execution files created by the *uucico* transfer. Remember that the initial work file with a grade of "X" becomes an execution file upon transfer. If any execution files are found, *uuxqt* attempts to execute them and then *uuxqt* terminates.

A file is the smallest unit transfered; in case a new connection is made, a packet of information is not re-transmitted, *uucico* attempts to transfer the entire file! Small files provide the best way to ensure a successful transmission. The longer time involved in transferring large files increases the probability of a communication error.

User Invocation

Although the *uucp* demons are normally invoked as a result of the *uux* or *uucp* commands, either you or a *uucp* demon may also initiate them.

Uucico Demon

The syntax to invoke the *uucico* demon is:

```
/usr/lib/uucp/uucico -r1 [-ssystem_name] [-xn] &
```

where:

`-r1`	invokes *uucico* as the master;
`-ssystem_name`	is an optional parameter indicating the node name of the system you want to contact;
`-xn`	is an optional parameter providing debug or error information (n is a digit between 1 and 9; the higher values print more information.).
`&`	is used to execute the process in the background. This prevents your terminal from being tied up.

Uuxqt Demon

The *uuxqt* module performs local command execution of execution (`X.*`) files from remote systems. The syntax needed is:

```
/usr/lib/uucp/uuxqt [-xn] &
```

where:

`-xn`	is the same as for uucico above.
`&`	is used to execute the process in the background. This prevents your terminal from being tied up.

Uudemons

Uudemons are script files which are used to:

- periodically clean up certain files, such as your files which log information;
- communicate with systems which are waiting for you to contact them.

These script files are normally executed by entries in the `/usr/lib/crontab` file.

The *uudemon.hr* script is shipped with your Series 200 or Series 500 system. Note that *uudemon.hr* hourly cleans up old status files and lock files and polls all systems for which you have work pending. If you use the **-ssystem** option the *uudemon*s force a call to the system specified. This is necessary for PASSIVE only systems (systems waiting for contact from another system) which cannot initiate communication with you. You can edit this file by replacing the **-s<nodename>** with the system name you want polled.

Using the Uucp Facility 7

After the *uucp* facility has been installed on your system, you are ready to begin using the *uucp* commands. Your System Administrator (or you) **must** have completed the hardware configuration, the software configuration and edited the neccessary files. Refer to the chapter entitled, "Overview" for a summary of tasks necessary before you use the examples in this chapter.

You must be logged into your local system to use these commands.

Syntax Information

Path Names

Just as you must use unique names to reference files on your local system, unique *system_name!path_name/file_names* are needed to identify files on remote systems. Note that the system name is separated from the complete path name to that file by a ! character, which has the exciting name of "bang"!

An example of a complete path name (fully qualified) to a file is:

```
rem_node!/usr/sys_dir/your_dir/your_file
```

You also have the option of using the ~ character to represent a login directory. If you use `~your_user_name`, the remote system expands this to the `your_user_name` login directory, for example:

```
rem_node!~your_user_name/your_file
```

would be expanded by the remote system to:

```
rem_node!/usr/sys_dir/your_dir/your_file
```

if the login directory for `your_user_name` is `/usr/sys_dir/your_dir`.

If you specify `~/file_name`, the *uucp* facility uses the public area in the spool directory: `uucppublic`.

If you do not use any path name after the system name, the **current local directory** or the remote login directory is used.

The path name syntax for files within the **current local directory** also supports:

*	a "wild card" character indicating zero or more characters;
?	any single character;
[...]	ending character(s) for the file.

Note that the *uucp* command allows you to exchange files with other systems by using remote systems as links into the system you wish to obtain information from or send it to. For example, you might type a command line which looks like this:

```
uucp message node1!node2!node3!/usr/spool/uucppublic/file_name
```

This sends a `message` to the system `node3` and places it in the default directory `/usr/spool/uucppublic` on that system. The `message` is placed into a file with the name of `file_name`. For more information on this read the section in this chapter entitled, "Forwarding through Several Systems".

Option Separators

Square brackets indicate optional parameters; spaces and the - character are **required between each option**.

The Cu Command

The *cu* ("call UNIX") command manages interactive communications with HP-UX or UNIX remote computers, as well as with non-UNIX remote computers. It functions as an asynchronous terminal emulator.

The *cu* command is used interactively to transmit messages to and from remote systems and to transfer ASCII files. You can also use the *cu* command to interactively contact a remote system to verify that the connection is working properly before you invoke the *uucp* or *uux* commands.

Cu tries each line in the `L-devices` file (which specifies acceptable devices and types of connections) until it finds a match with the parameters specified or until it runs out of entries.

Cu Command with a Modem Connection

The syntax for using the *cu* command with a modem connection is:

```
cu [-sspeed] [-aacu] [-q] [-h] [-m] [-t] [-o|-e] tel_num|sysname
```

where:

-sspeed	specifies the transmission speed of 110, 150, 300, 1200, 2400, 4800, or 9600 baud. (The default is 300 baud.);
-aacu	specifies the modem device name to override searching for the first available device with the correct speed;
-q	enables the ENQ/ACK handshake;
-h	emulates local echo (The remote system expects your terminal to be in half-duplex mode.);
-m	ignores modem controls;
-t	is for adding CR to LF on output to remote (for terminals);
-o|-e	designates even or odd parity (The default is no parity.);
tel_num	is the telephone number of the remote system (Only digits, "-" meaning pause and "=" meaning wait for secondary dial tone are allowed.);
sysname	gives the name of a system which appears in *L.sys*.

Some examples of using the cu command with a modem connection:

`cu -s1200 -e -q 555-1212` (outside line, 1200 baud)

`cu -s1200 -o -q 9=555-1212` (inside line, 1200 baud)

if your inside telephone line requires that you dial "9" and then wait for a secondary dial tone before you dial the number.

These messages are displayed on your CRT if the connection is made:

```
autodialing - please wait
Connected
login:  (from remote system)
```

If the autodial failed, the message:

```
Connect failed: autodial failed
```

is generated or in some cases, a reason for the failure is given.

You need to know how to log into the remote system to respond to its `login:` prompt.

Cu with Direct Connection

The syntax for the cu command with a **direct connection** is:

```
cu [-h] [-q] [-m] [-t] [-o|-e] -lline dir|sysname
```

where:

-h	emulates local echo (default is full duplex);
-q	enables the ENQ/ACK handshake;
-m	ignores modem controls;
-t	is for adding CR to LF on output to remote (for terminals);
-o|-e	designates even or odd parity (default is no parity);
-l	specifies the device name and is a mandatory parameter;
dir	is a character string which must be used for a direct connection.
sysname	gives the name of a system which appears in *L.sys*.

With a direct connection the `-sspeed` parameter is ignored. The *cu* facility uses the speed field in the `L-devices` file.

For example if you type:

```
cu -ltty09 -s1200 -m dir
```

and the line in the valid devices file for the direct connection of tty09 is:

```
DIR tty09 0 9600
```

the line specified by tty09 is opened at 9600 baud and **not** 1200 baud. Note that it may be changed to 1200 baud by executing the following:

```
~!stty 1200 < /dev/tty09
```

Some examples of using the *cu* command for a direct connection are:

```
cu -ltty09 dir
cu -h -ltty09 dir (remote expects you to be in half-duplex mode);
cu -o -ltty09 dir (odd parity)
```

The login prompt appears on your CRT if the connection is made properly.

After Connection

Cu reads data from the standard input file and passes it to the remote system when the transmit process is active. *Cu* accepts data from the remote system and passes it to the standard output file when the receive process is active.

Transmitted lines beginning with a "~" have special meanings:

~%cd path	change directory (default is $HOME);
~.	terminate the connection;
~%b	transmit a break character (You can also use "~%break".);
~! [command]	escape to local shell and return by EOT (If you specify a command on this line, the local shell executes the command and returns.);
~& [command]	run the command, but kill the *cu* "receive" process and restore it later.
~$ [command]	run the command locally and send its output to the remote system;

~%take from [to]	copies the file "from" on the remote HP-UX or UNIX system to the file "to" on your local system (If you do not use the optional [to] parameter the file "from" on the remote system is copied to a file with the same name, "from", on your local system and created if none exists.);
~%put from [to]	copies the file "from" on your local system to the file "to" on the remote HP-UX or UNIX system (If you do not use the optional [to] parameter the file "from" on your local system is copied to a file with the same name, "to", on your remote system.);
~%<file	upload file with prompt handshake;
~%setps xy	set prompt sequence to **xy** (default DC1);
~%setpt n	set prompt timeout to **n** seconds;
~%set	tell what the current timeout and prompt are;
~~...	send the line "~..." to the remote system;
~%nostop	toggle the DC3/DC1 input control protocol off and on;
~%>file	begin output diversion to file;
~%>	end any active output diversion.

When you want to transfer the ASCII file, "My_file", which is on your local system to a file named "My_rem_file" on the remote system, type:

```
~%put My_file My_rem_file
```

do not press any key on your keyboard while transferring files with "%take" or "%put". The facility may transmit incorrect data or be left in an unstable state.

Do not terminate the *cu* program while a communication is in process.

Using the uucp Command

The *uucp* command is a background program that is used to transfer files to and from remote HP-UX or UNIX systems. *Uucp* creates work files and data files in the `/usr/spool/uucp` directory for later processing.

General uucp Snytax

The general syntax for the *uucp* command is:

```
uucp [options] source_file(s) destination_file
```

for *uucp* to copy the `source_file(s)` to the destination file.

The following options can be used:

-c	use the source file **when** copying out (rather than making a copy of the source file at the time the command is issued and storing it on the spool directory for later processing — this is the default.);
-C	make a copy of the source file at the time the command is given and store it on the spool directory;
-d	make all necessary directories for the file (This is the default.);
-esys	send the uucp command to the remote system sys (This option is only successful if the remote system allows the *uucp* command in its `L.cmds` file.);
-f	do not create intermediate directories;
-ggrade	request a grade for work sequencing (A grade of "A" specifies "do this work first, "z" specifies last and "n" is the default.);
-m	send mail to the requester when the copy is complete;
-nuser	send mail to notify the user on the remote system that a file was sent;
-r	create the files necessary for the transfer to take place, but do not invoke uucico to call the remote system;

The `source_file(s)` must exist and both that file and its path name must be readable by everyone.

The `destination file` does not have to exist; *uucp* creates a file by that name if none exists. If the destination file already exists, it must be writable by everyone and the path name must be readable and writeable by everyone. Your System Administrator can change the access permissions to files and paths.

For example, if you typed:

```
uucp /usr/sys_dir/user_dir/file1 hpsys1!~uucp/file2
```

this work file is created in /usr/spool/uucp:

```
C.hpsys1n2270
```

whose contents is:

```
S /usr/sys_dir/user_dir/file1 ~uucp/file2 sys_dir -dc D.0 444
```

Do not use the *uucp* command to copy a **local** source file to a **local** destination file.

Sending Files To a Remote System

The following examples send one or several files to a remote system:

```
uucp file1 hpsys1!/users/hpfsd/file2
```

sends `file1` in the current directory to `/users/hpfsd/file2` on remote system `hpsys1`.

```
uucp /usr/all.exp/cmd/file1 hpsys1!/users/hpfsd/file2
```

sends `/usr/all.exp/cmd/file1` on the local system to `/users/hpfsd/file2` on the remote system `hpsys1`.

```
uucp ~kls/file1 hpsys1!~uucp/file2
```

sends `file1` on the login directory for `kls` to `file2` on the remote system, `hpsys1`, in the login directory for *uucp*. Note that the `~` is used to represent either the local login directory or the remote login directory. In this example `~uucp` (after `hpsys1!` in the destination file field) is used not as the *uucp* command, but as a typical name for a remote login directory.

A copy is made of `file1` **when** the *uucico* demon performs the file transfer. Since this is a background operation, the transfer occurs at some time after you invoke the *uucp* command. If you have modified `file1` between the time you invoked the *uucp* command and the time of transfer, the modified version is the one transferred to the remote system.

When you want to send a copy of a file's current contents to a remote system and then continue to modify that file, use the -C option. For example:

```
uucp -C -m ~kls/file1 hpsys1!~uucp/file2
```

makes a copy of `file1` and places it on the spool directory to send at transfer time to `file2` on the remote system.

Receiving Files From Remote Systems

The following examples show how to use the *uucp* command to request that a file on a remote system be copied to your local system.

```
uucp -m hpsys2!~ems/prog ~my_login/BIN/
```

requests the file `prog` from the login directory for `ems` on the remote system, `hpsys2`, be copied into a file of the same name in the `BIN` directory of `my_login` directory of the local system. The **-m** option asks that you receive mail when the copy is complete.

```
uucp hpsys2!*.[ab] my_login
```

fetches all files ending in **a** or **b** in the login directory on the remote system, `hpsys2` and places them in the subdirectory, `my_login` on your local system.

All files in the **current** directory can be sent with a * character or a subset of these files can be sent with `*.[qualifiers]`. You cannot use * to represent entire path names.

Forwarding through Several Systems

In order for forwarding to be possible on your system, you have to have special permissions set up in the following files.

`L.sys` contains information which determines how *uucp* will automatically reach other systems, and whether remote systems will be able to log into your system.

`FWDFILE` is a subset of `L.sys` and it provides a list of the systems through which your system may forward files.

`ORIGFILE` contains a list of originating system nodes which may forward files through your system. For example, if system nodes B, C, and D are part of your *uucp* network and they have included your system node A in their `ORIGFILE` then you can send a copy of `file_name` to system node D by typing,

```
uucp B!C!D!file_name Return
```

The `L.sys` file should be set up as explained in the chapter, "Uucp File System". The subset of `L.sys` called `FWDFILE` should be set up with a list of the system nodes which you have forwarding access through. The `ORIGFILE` should be set up with a list of the system nodes which have forwarding access through you. The system node names are used to identify a given system within a network of systems using *uucp*. The following table shows you how you might set up the permissions within these file for the given network diagram.

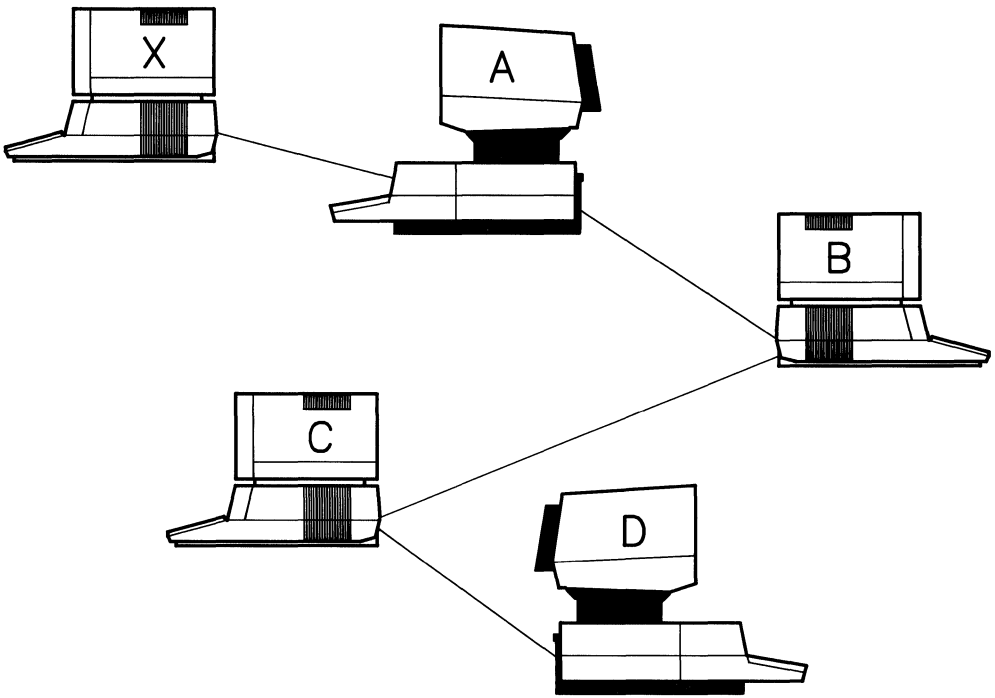


Figure 7-1. Network of Systems Using Uucp

The letter characters used in the following table represent system node names.

Table 7-1. Network System Nodes

File Names	System node names			
	A	B	C	D
L.sys	X, B	A, C	B, D	C
FWDFILE	B	C		
ORIGFILE	A	B		

Analyzing the column containing system node **C**, you will find that systems **B** and **D** can communicate with it using *uucp*. The reason for this is both of these systems were included in system **C**'s `L.sys` file. However, system **C** does not have permission to forward through other systems because it does not contain any system node names in its `FWDFILE` file. System **B** can however forward files through system **C** because it has been included in system **C**s `ORIGFILE` file.

An example of sending a message (file) through a series of remote nodes would look similar to the following:

```
uucp message node1!node2!node3!/usr/spool/uucppublic/file_name
```

Anyone desiring to send a `message` (file) to you from a remote system would use the same format as shown above, where:

`message`	is the file you wish to send.
`node1`	is the node you would be sending your message through and `node2`, `node3`, ..., etc. are the nodes your message will be relayed through before reaching its destination.
`/usr/spool/uucppublic`	is a directory open for writing by everyone. It is in this directory the receiver would find messages from remote systems and the sender of a message would direct his messages.
`file_name`	is the name given to the file which was sent to a remote system and placed in the directory `/usr/spool/uucppublic`.

Uucp Command Errors

If an error occurs in the *uucp* command transfer, a message indicating the problem is generated on the standard output device, normally your CRT.

"1" is the only error number ever generated. This indicates one of the following conditions:

- you have no right to access this file;
- the file does not exist;
- the file may not be copied;
- the system name given is incorrect.

Using the uux Command

The *uux* command gathers zero or more data files from various systems, executes a command on a specified system and then sends the standard output file for that execution to a file on the system on which the command was executed.

General uux Syntax

The general syntax for using the uux command is:

```
uux [options] command_string
```

where options can be:

- **-** uses standard input to get the data for the command;
- **-z** requesting that the remote system be notified by mail only if the command execution failed;
- **-r** creates the files necessary for the transfer to take place, but does not invoke *uucico* to call the remote system.
- **-n** requesting no mail notification for the remote system.

The `command_string` must be enclosed in double quotes ("command_string") if you specify an input or output diversion for the `command_string`. If you did not use quotes, the shell would try to redirect the input/output of the *uux* command. For example, if you type:

```
uux "sys2!pr !ems/cmd/file > sys2!/dev/lp"
```

you request that the `pr` command be executed on remote system `sys2`. Your `ems/cmd/file` on the local system (`sys1`) is printed by `sys2` to its `/dev/lp` file. At the time the *uux* facility is evaluating the command string, the `/dev/lp` file must also be accessible to the system **originating** the *uux* command (`sys1`).

Note two things illustrated in this example:

- all local files must be prefixed with `!`;
- the system for the redirection of the output should be the same system on which the command is executed.

These files are created in the `/usr/spool/uucp` directory:

`C.sys2AAxxx` is the work file which has these content lines:

```
S D.sys2Bxxxx D.sys2BxAxx ems - D.sys2BAxxx 666
S D.sys1XAxxx X.sys1XAxxx ems - D.sys1XAxxx 666
```

`D.sys1XAxxx` an execution grade data file which has these content lines:

```
U ems sys1
F D.sys2BAxxx file
O /dev/lp sys2 O
C pr -n file
```

`D.sys2BAxxx` which has as its contents a copy of the "file" to be printed.

Example

When you want to compile a Pascal program on your local HP-UX system and have the results of that compilation directed to a file, you would use a command_string similar to:

```
pc pas_file.p > pas_com
```

The remote execution of this command string merely includes the *uux* command and name(s) of the local or remote systems, for example:

```
uux "hpsys1!pc !/users/cmd/pas_file.p > hpsys1!~/pas_com"
```

requests that hpsys1 execute the pc command on the Pascal program, `pas_file.p` on the local system and place the results of the compilation on the hpsys1's public area in the file `pas_com`. Note that the system!file the output is redirected to should be the same system which executed the command.

You must have the permission of the remote system to execute each command. The list of all the commands a system permits to be executed by another system are contained in the `L.sys` file. You are notified by mail if the requested command on the remote system was not allowed.

The usual file naming conventions stated in the beginning of this chapter apply to the uux command file names with these exceptions:

- all local files must be prefixed with `!`;
- output files must have their parentheses escaped: `\(output_file\)`.

Using:

```
uux hpsys1!uucp hpsys2!/usr/file1 \(hpsys1!/usr/file2\)
```

sends a *uucp* command to `hpsys1` to copy `file1` from `hpsys2` to `file2` on `hpsys1`. Preceding the left and right parentheses by a \ character tells the shell to interpret the parentheses literally. The parentheses tell the shell not to gather `file2` on `hpsys1` as a data file for the *uucp* command, but rather to use `file2` as the output file.

Uux Error Numbers

The *uux* command has several errors which are printed as error numbers rather than the usual error messages. These error numbers and their interpretations are listed below.

1 You have no right to access a file, the file does not exist or the file is not copiable.

2 The path name cannot be properly expanded.

101 You specified an invalid system name.

102 The size of the parameters given to uux exceeds the maximum length specified in the `BUFSIZ` variable.

Using uuclean, uulog, uuname, uupick, uustat, uusub and uuto

The *uuclean, uulog, uuname, uupick, uusub,* and *uuto* commands are grouped together in this section and discussed in alphabetical order.

Using the uuclean Command

The *uuclean* command scans a directory for files with a specified prefix and deletes those which are older than the specified number of hours. If you have a backlog of jobs that cannot be transmitted to other systems, they should be cleaned-up so that the file space can be reused. You can also have the uuclean command remove lock and status files which are no longer needed.

These cleanup activities can be routinely executed by shell scripts started by the *cron* program. Refer to the chapter, "Log, Status and Cleanup" for a detailed discussion.

The general syntax for the uuclean command is:

```
uuclean [options]
```

where options can be:

-d*directory*	is the directory to scan for files instead of the spool directory (The spool directory is the default.);
-p*pre*	is the file prefix (Up to ten prefixes may be specified. If no **-p** option is specified, **all** files older than the **-n*time*** hours are deleted.);
-n*time*	is the time in hours where files older than *time* are deleted (The default is 72 hours.);
-m	sends mail to the owner of a file when that file is deleted.

An example of using the uuclean command is:

```
/usr/lib/uuclean -pLOG -pLCK -n24
```

This removes all files starting with `LOG`, such as `LOGFILE`s and all files starting with `LCK`, such as `LCKFILE`s older than 24 hours.

The chapter, "Status, Log and Cleanup" contains examples of shell scripts which use the *uudemons* to implement *uuclean* commands.

Using the uulog Command

The *uulog* command displays a summary log of *uucp* and *uux* transactions. If you use the *uulog* command without any options, the information in the temporary log files (`LOG.*`) is appended to the main `LOGFILE`. These `LOG.*` files are created only if `LOGFILE` is locked when the *uucp* facility attempts to make an entry. *Uulog* then gathers information in the `LOGFILE` in the `/usr/spool/uucp` directory and prints this information.

The general syntax for the *uulog* command is:

```
uulog [-ssys_name] [-uuser_name]
```

where:

`-ssys_name`	prints information about work involving system `sys_name`;
`-uuser_name`	prints information about work done for the specified `user_name`.

which displays user system (date time `PID_number`) status action.

For example, if you type:

```
uulog -smit
```

a typical display for the system `mit` would be:

```
john mit (2/14-10:11-15486) SUCCESSFUL (AUTODIAL)
john mit (2/14-10:11-15486) SUCCEEDED (call to mit)
john mit (2/14-10:11-15486) OK (startup)
john mit (2/14-10:11-15486) REQUEST (S D.mitn2236 D.mitn2236 john)
john mit (2/14-10:12-15486) OK (conversation complete)
```

The invocation of the *uulog* command without any parameters appends all temporary log files (`LOG.*`) to the main `LOGFILE`.

Refer to the appendix, "Log Entry Messages" for an alphabetical listing and interpretation of `LOGFILE` messages.

Using the uuname Command

The *uuname* command returns the *uucp* name of your local system or the nodenames of remote systems known to your local system.

The general syntax for uuname is:

```
uuname [-l] [-v]
```

where:

-l returns your local system name.

For example:

```
uuname -l Return
```

returns:

```
My_node_name
```

and:

```
uuname Return
```

returns:

```
mit
csu
hp-sys1
UCLA
ISU
```

-v returns a description for each system listed in the ADMIN file.

These are the names of the systems you can communicate with.

Using the uupick Command

This command should be used with the *uuto* command.

With the *uupick* command you can accept or reject files transmitted to you with the uuto command on another system. Uupick searches `/usr/spool/uucppublic` for files sent to your local system by *uuto*. For each file or directory found *uupick* prints a message about the designated file on the standard output file. *Uupick* then waits for an answer indicating what you want to do with that file, reading the answer line from the standard input file. The cycle is repeated until *uupick* finds no more uuto files destined for you.

The general syntax for uupick is:

```
uupick [-ssys_name]
```

where:

`-ssys_name` searches `/usr/spool/uucppublic` for files sent only from `sys_name`.

When *uupick* is reading from the standard input file to determine the disposition of a file, the following translation table is used:

Table 7-2. Translation Table

You Type	This Meaning
<carriage return>	go on to next entry
d	delete the entry
m[dir]	move the file to the named directory **dir** (The default is the current directory.)[1]
a[dir]	move all files sent from <sys_name> to the named directory **dir** (The default is the current directory.)
p	print the contents of the file
q	quit (stop)
EOT (control d)	quit (stop)
!command	escape to the shell to do **command**
*	print a command summary

[1] Do not use "." to represent the current directory with the **m** or the **a** parameter. Use the default instead.

For example, if you type:

```
uupick Return
```

and `file_name` is being sent from `sys1`, this is printed on the standard output file:

```
from system sys1: file file_name
```

You then could use any of the options listed in the table above.

Using the uustat Command

The uustat command initiates status inquiry for all jobs requested and for job control.

The general syntax for the *uustat* command is:

```
uustat [options]
```

where options could be:

`-chour`	remove the status entries which are older than `hour` hours old (Only the user initiating the *uucp* command or the super-user may invoke this option.);
`-jall`	report the status of all the *uucp* requests;
`-kjob_num`	kill (terminate) the uucp request whose job number is `job_num` (The terminated *uucp* request must belong to the person issuing the uustat command or the super-user. The `job_num` is supplied automatically by the *uucp* facility.);
`-mmachine`	report the status of accessibility of `machine` (If `machine` is **all**, then the status of all machines known to the local *uucp* are displayed.);
`-ohour`	report the status of all *uucp* requests which are older than `hour` hours old;
`-yhour`	report the status of all uucp requests which are younger than `hour` hours old;
`-ssys`	report the status of all *uucp* requests involving system `sys`;
`-uuser`	report the status of all uucp requests issued by `user`;
`-v`	report the *uucp* status verbosely (This option is recommended since without it only the status code for each request is printed.).

Using *uustat* without any options prints all job information in the non-verbose mode for the user you are logged in as.

If you type:

```
uustat -v -jall
```

the following is an example of your display:

```
0923     rmd      hpfcla     04/25-11:00      04/25-13:01
REMOTE ACCESS TO FILE DENIED
COPY FINISHED, JOB DELETED
```

where:

`0923`	is the *uucp* job number;
`rmd`	is the user issuing the uucp command;
`hpfcla`	is the remote system;
`04/25-11:00`	is when the uucp command was first issued;
`04/25-13:01`	is the last status update;
`REMOTE ACCESS TO FILE DENIED`	is part of the status report.
`FINISHED, JOB DELETED`	is the remainder of the status report.

The status report indicates that the work file was deleted without any data file being copied.

Using the uusub Command

The *uusub* command defines the *uucp* subnetwork and monitors the connection and traffic among its members. This command is normally used by the superuser or System Administrator.

The general syntax for using the uusub command is:

```
uusub [options]
```

where options could be:

`-asys`	add `sys` to the subnetwork (Only one system can be added at a time.);
`-csys`	exercise the connection to system `sys` by making a call to that system (`sys` may be **all** for all systems in the subnetwork.);
`-dsys`	delete `sys` from the subnetwork;
`-f`	flush (erase) the connection statistics;
`-l`	report the statistics on connections;
`-r`	report the statistics on traffic amount;
`-uhour`	gather the traffic statistics over the past `hour` hours.

which when executed displays:

```
sys #call #ok time #dev #login #nack #other
```

where:

`sys`	is the remote system name;
`#call`	is the number of times your local system tried to call `sys` since the last flush;
#ok	is the number of successful connections;
time	is the latest successful connect `time`;
`#dev`	is the number of unsuccessful connections because of no available device;
`#login`	is the number of unsuccessful connections because of login failure;
`#nack`	is the number of unsuccessful connections because of no response;
`#other`	is the number of unsuccessful connections because of other reasons.

You can define your subnetwork with the **-a** option, for example:

```
uusub -ahpdcd
uusub -ahprvd
uusub -ahpcnob
uusub -ahpfcld
```

You can then monitor this defined subnetwork by typing:

```
uusub -l
```

an example of the output is:

```
sysname   #call     #ok     latest-oktime     #noacu     #login     #nack   #other

 hpdcd      26        9      (4/25-23:58)        0          0         17       0
 hprvd       0        0      (4/21-15:30)        0          0          0       0
hpcnob      25       24      (4/25-23:57)        0          1          0       0
hpfcld       5        2      (4/25-23:56)        0          0         14       0
```

The meanings of the traffic statistics gathered with the -r option are:

```
sfile   sbyte    rfile   rbyte
```

where:

`sfile`	is the number of of files sent;
`sbyte`	is the number of bytes sent over the period of time indicated in the latest *uusub* command with the `-uhour` option;
`rfile`	is the number of files received;
`rbyte`	is the number of bytes received.

The traffic statistics over the last two hours can be gathered by typing:

```
uusub -u2
```

The traffic statistics must be gathered before they can be reported with the `-r` option. For example, if you now type:

```
uusub -r
```

your output could be:

```
sysname       sfile         sbyte         rfile         rbyte

hpdcd           2             699           0             0
hprvd           0             0             4             584
hpfcla          66            266639        34            140784
```

Using the uuto Command

The *uuto* command sends files to a specified destination using the *uucp* facility. You can use the *uupick* command to "pick" disposition of the files sent with the *uuto* command.

The general syntax for the *uuto* command is:

```
uuto [options] source_file destination
```

where:

`options`	can be either: `-p` — copy the `source_file` into the spool directory before transmission; `-m` — generates mail to the sender when the copy is complete.
`source_file`	is the source file(s);
`destination`	is of the form: `sysname!user` where `sysname` is the name of the remote system and `user` is the user on the remote system you are sending the files to.

The source_file(s) are sent to `/usr/spool/uucppublic` on `sysname`.

If you type:

```
uuto -p -m /users/rmd/file hpsys2!mark
```

this *uucp* command is generated:

```
uucp -d -C -m -nmark /users/rmd/file
          ~/receive/mark/hpsys1/
```

where the destination user is `mark` and `hpsys1` is the system **from** which the file is transferred.

Using the Mail Facility

You can use the *mail* command to send mail messages to other systems. For example, if you type:

```
mail remote_sys!name Return
Meet me in the lunch room. Return
Don't be late again! Return
  CTRL D
```

lines two and three are mailed to `name` on `remote_sys`.

When you specify remote systems with *mail* command the *uucp* facility uses the *uux* command sequence. A work file with an `X` grade and one containing the actual mail message, are set up in the `/usr/spool/uucp` directory.

Note that you do not have to specify the entire path name to the user receiving your mail message.

You can also forward mail through intermediate nodes (read the section in this chapter entitled, "Forwarding through Several System") by using a syntax similar to:

```
mail remsys1!remsys2!remsys3!name Return
(Message typed in here.) Return
  CTRL—D
```

your mail would be forwarded through `remsys1` and `remsys2` before it got to its final destination on `remsys3`.

You can mail entire ASCII files to a user on a remote system by using:

```
mail remote_sys!name <file_name Return
```

NOTES

The X.25 Network 8

This chapter provides you with a brief discussion on what the X.25 Network is and explains how to configure the *uucp* software to work with X.25.

An Explanation of X.25

X.25 is a worldwide standard protocol used in many Public Data Networks (PDN's). Public Data Networks (PDN's) are Packet Switched Networks (PSN's).

Packet Switched Network

Before learning what a Packet Switched Network (PSN) is, you need to know what an X.25 packet is. X.25 packets are defined as serially transmittable strings of information (code) containing the following fields:

ADDRESS	where the packet is destined for.
DATA	the actual information to be transmitted, usually no more than 256 bytes.
CHECKSUM	error detection information.
SEQUENCE	this is to preserve the relative order of packet transmission.
OTHER	other fields which are not covered in this manual.

A Packet Switched Network consists of many nodes, ("stations") each of which knows something about where to send packets (routing information.) The entity that wants to send information combines the information into a packet and transmits it to the nearest station.

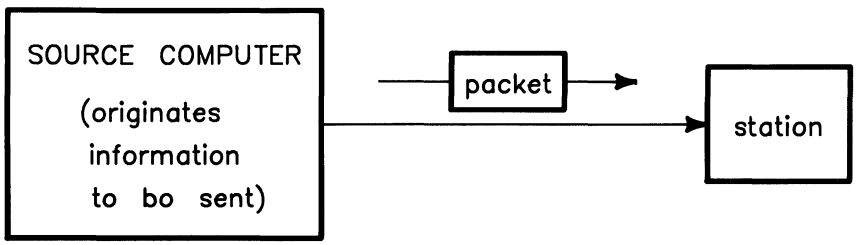


Figure 8-1. Packet Transmission

The station then examines the address information in the packet, and determines automatically where to send it. The station that receives it repeats this process. And it is repeated again at the next station, and again until the packet arrives at its destination.

Consider the following hypothetical stations and interconnections:

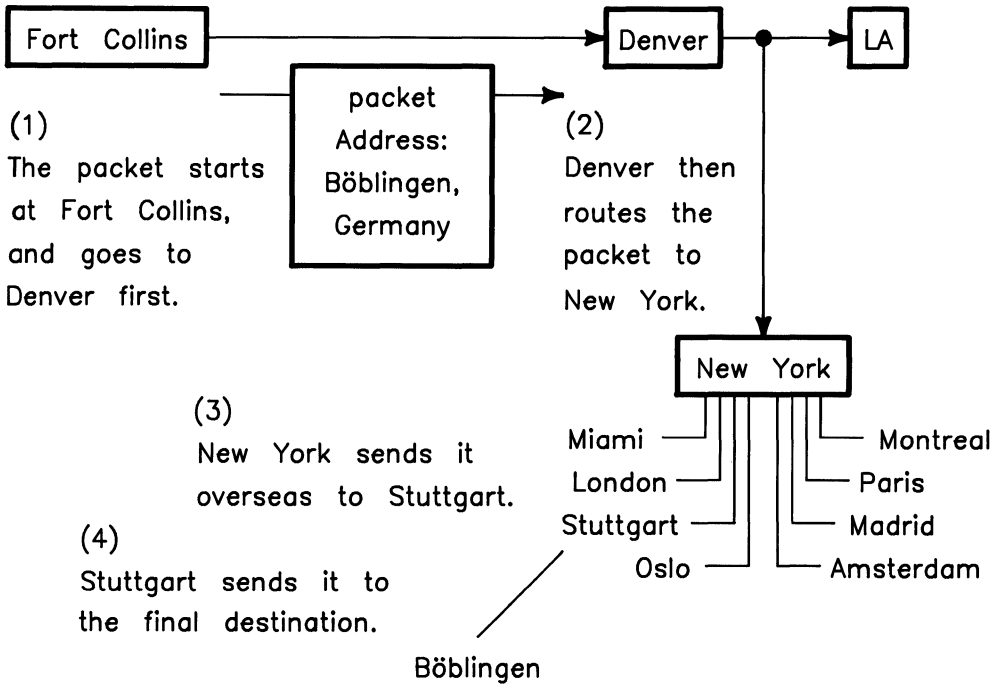


Figure 8-2. Packet Switched Network

In this example, the packet goes to Denver first. The Denver Station determines that the packet should go to New York next, seeing that it is bound for overseas. New York routes it to Stuttgart, and then it arrives at its final destination, Böblingen.

At this point, the receiving computer in Böblingen gets the packet, and extracts the important information from it (packet disassembly.) The address is checked to make sure it arrived at the correct place, and a checksum is computed on the data and checked with the transmitted checksum. If there is an error, retransmission is requested. The sequence number is checked to make sure the packet did not arrive before a packet which preceded it. (Due to a number of factors, it is possible that if two packets are transmitted one after the other, the one transmitted later might arrive earlier.) If everything is **ok**, the data is passed to the receiving computer.

Public Data Network

A Public Data Network (PDN) is a packet switched network that each country has established to handle data traffic. Most countries have only one PDN, while a few have several. Public Data Networks (PDN's) are connected to each other by "gateways". An X.25 gateway is really nothing more than packet switched connections between two stations. For example,

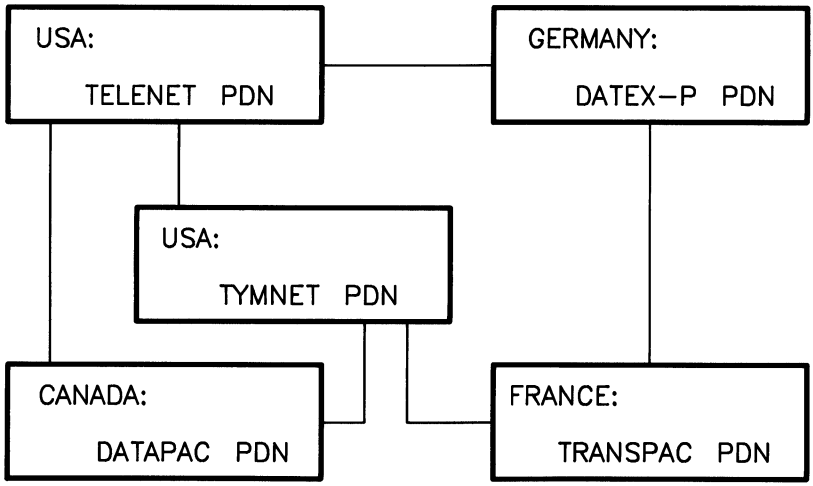


Figure 8-3. Public Data Network

Most civilized countries have their own Public Data Network (PDN) with a "gateway" to the rest of the world. This worldwide conglomeration of PDN's is collectively known as "The X.25 Network".

Configuring uucp for X.25

This section covers topics which help you configure *uucp* for use with the X.25 Network. These topics are as follows:

- Some Prerequisites,
- Installing the HP 2334A,
- Remote and Local Off-line Configuration,
- Preparing for Configuration,
- Configuration Procedure.

The *HP 2334A MULTIMUX Reference and Service Manual* (HP part number: 02334-90001) covers the above topics in detail. If you are using the HP 2334A to connect to the X.25 Network, you need to read the, *HP 2334A MULTIMUX Reference and Service Manual* after reading this manual.

The acronym PAD which stands for Packet Assembler/Disassembler needs to be defined before proceeding with this section of the chapter. A Packet Assembler/Disassembler is a device which takes a message to be transmitted and **assembles** it into transmittable X.25 data packets. Conversely, it receives X.25 packets and **disassembles** them into character streams for transmission to a terminal.

The HP 2334A is the device which provides for the interfacing of your HP-UX system to the X.25 Network. This device must be configured with the proper synchronous X.25 network parameters to enable communication across the synchronous network and the asynchronous protocol (by assigning PAD parameter values) for communication with connected asynchronous devices (or computer ports). These communication parameters must initially be defined off-line (i.e., when not communicating with the synchronous network or devices). They are saved automatically in non-volatile (permanent) memory so that the HP 2334A does not require the configuration process each time the power is turned on.

The off-line configuration (or reconfiguration) of the HP 2334A is normally performed using a terminal which is directly connected to the HP 2334A device port A1. Alternatively the off-line configuration (or recofiguration) may be performed from a remote location by connecting the remote terminal to the HP 2334A device port A1 via a pair of asynchronous modems and a telephone line. Both of these off-line methods are covered in this section of the chapter.

Some Prerequisites

Before you can start configuration of X.25 you need to make sure the following prerequisites have been met.

- The *uucp* facility must already be working on your system.
- You must have an understanding of your *uucp* file system.
- You should know how to use the *uucp* commands.

Information for these prerequisites is covered in the chapters prior to this one. If you haven't read these chapters please do so and return to this chapter when you are done. You also need to use the unpacking list sent with your HP 2334A to verify that you have the necessary hardware to begin the installation of your HP 2334A device.

Installing the HP 2334A

This section covers the necessary steps for installing the HP 2334A. Note that line voltages, power supply settings and mounting the HP 2334A are covered in the manual, *HP 2334A MULTIMUX Reference and Service Manual.* The topics included here are as follows:

- Power-on and CPU Switch Test,
- Connecting Cables to an HP 2334A.

Power-on and CPU Switch Test

The HP 2334A has a power-on self test which is performed automatically at power-on and a CPU switch test which is perform manually by the user of the HP 2334A. The first test is covered in this manual, as the second one is not necessary for getting started on your HP 2334A. The test are as follows:

- An internal "power-on self test" is automatically performed whenever the HP 2334A is switched ON (1), or initialized using the reset button. This test is performed regardless of whether the HP 2334A is off-line or on-line. To observe this test, you need to remove the front panel of the HP 2334A. You can remove the front panel by placing your fingers under the lip on the top part of the front panel and pulling outward. Turn the unit on and observe the LEDs on the CPU card. The LEDs will plink off and on as the self test is being executed. For a description of this test, read the section in this chapter entitled, "Entering the CONFIGURE Mode".

- An off-line CPU Card Switch Test checks the operation of the CPU cards DIP switches. For an explanation of this test, read the section in the, "INSTALLATION" chapter of the, *HP 2334A MULTIMUX Reference and Service Manual* entitled, "CPU CARD SWITCHES TEST".

With the front panel off you need to set switch 8 of the CPU card's DIP switches to the upward position and all other switches should be set to the downward position. Note that the front panel should be left off after you have completed this section.

Connecting Cables to an HP 2334A

The cable connections made in this section are for "local off-line configuration". Local off-line configuration is where you have a terminal directly connected (not through a modem) to the A1 port of your HP 2334A and the HP 2334A connected to the the X.25 Network. The next section in this chapter explains "remote" and "local" off-line configuration.

Before connecting any cables, you need to have the following cards insert in the HP 2334A:

- Synchronous Network Adapter Card
- Modem Control Adapter Card

The Synchronous Network Adapter Card has already been installed in your unit and is located behind the front panel as shown in the diagram below.

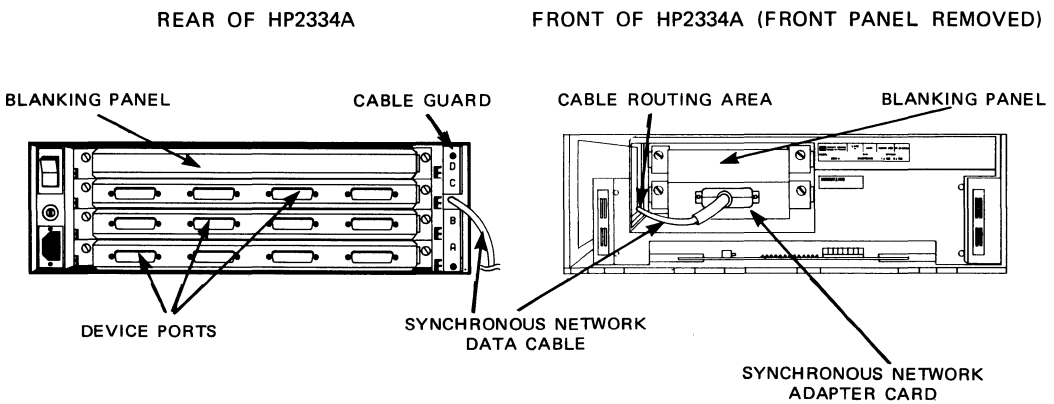


Figure 8-4. HP 2334A to Device and Synchronous Network Connection

The Modem Control Adapter Card (HP40261A) is installed in one of the Device Adapter Card slots located in the backplane of the HP 2334A as shown in the diagram below. Note that the first adapter card is inserted in slot A and the next one is inserted into slot B and so on. Also note that initial adapter cards come installed.

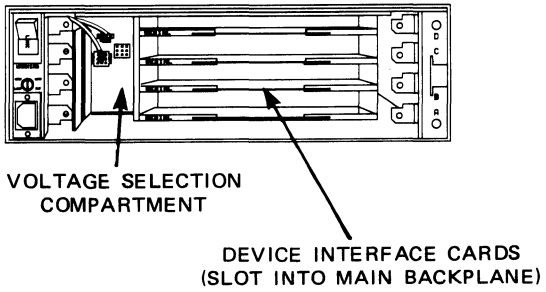


Figure 8-5. Adapter Card Slots

For local off-line configuration, you need to have an HP 40261A card inserted in slot A.

Connect an interactive terminal to port A1. Port A1 is located on the Modem Control Adapter Card which is in slot A of the backplane. The port is labeled number 1 on the Modem Control Adapter Card. The terminal is connect to port A1 by an RS-232C cable with a DTE connector.

The Synchronous Network Cable (HP part number: 02333-60008) is connected to the Synchronous Network Adapter Card using the following procedure:

1. Insure that the HP 2334A is switched OFF (0), then disconnect the power cord.
2. Remove the HP 2334A front panel.
3. On the right-hand side of the backplane (see the diagram, "HP 2334A to Device and Synchronous Network Connection"), remove the cable guard.

NOTE

Do not remove any of the Device Adapter Cards or blanking panels.

4. Pass the connector of the Synchronous Network Cable network into the data cable routing area, as shown in the diagram mentioned in step 3. This cable (part number: 02333-60008) is supplied with the HP 2334A.
5. From the front of the HP 2334A, carefully pull the cable through the data cable routing area and then plug the data cable connector into the Synchronous Network Adapter Card connector (see the diagram mention in step 3) and secure it in position by tightening the two locking screws.
6. Insure that the cable is a loose fit in the routing area. Then replace the front panel.
7. At the rear of the HP 2334A, place the slot in the cable guard over the data cable. Then insert a plastic cable tie (i.e. tie wrap) through the two holes in the cable guard slot and secure the cable guard (this acts as a cable clamp). Replace the cable guard on the HP 2334A backplane and tighten the two cross-head screws to secure it.
8. Replace the power cord and switch ON the HP 2334A as required.

Remote and Local Off-line Configuration

Off-line configuration may be performed remotely by connecting a terminal to port A1 of the HP 2334A via a telephone line and two full duplex, asynchronous modems (at 1200 baud) as shown in the figure below. An operator is required at the HP 2334A location to perform certain simple actions:

- DIP switch setting,
- Power-on/reset,
- Reading the LEDs.

A second telephone line is necessary for conveying verbal instructions.

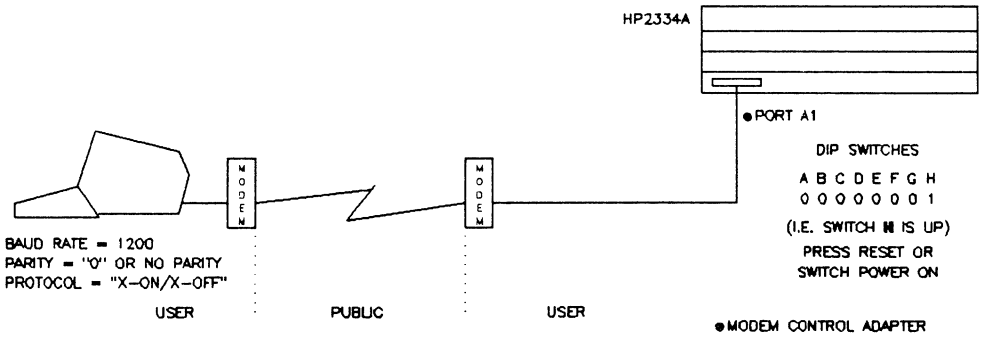


Figure 8-6. Remote Off-line Configuration

Remote off-line configuration is an important feature as it enables, for example, an experienced operator to perform off-line configuration of HP 2334As located at multiple remote sites (branch offices) without leaving the office.

The Modem Control Adapter Card (HP 40261A) provides modem interface ports A1 to A4 as remote configuration requires the use of asynchronous modems between the terminal and device port A1. Once communication is established between the terminal and the HP 2334A, the preparation and configuration procedure is the same as when configuring the HP 2334A locally (see the diagram below).

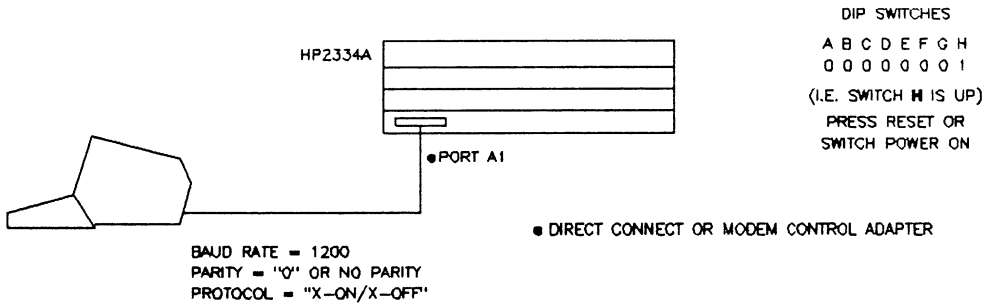


Figure 8-7. Local Off-line Configuration

The procedure for configuring the HP 2334A is explained in the remaining sections of this chapter.

Preparing for Configuration

The HP 2334A should be prepared for off-line configuration as follows (refer to the diagrams in the previous sections for connections and switch settings):

1. Connect an interactive terminal to device port A1. All other asynchronous devices may remain connected as required. The synchronous network may remain connected as required.
2. Set the terminal as follows:
   a. **BAUD RATE** — 1200 baud.
   b. **DATA BITS** — 7.
   c. **PARITY** — set for "0" or no parity.
   d. **X-ON/X-OFF Handshake** — **ENABLED**.
   e. **FULL DUPLEX**.
   f. **REMOTE** mode.
   g. **CHARACTER** mode (**BLOCK** mode **OFF**).
3. Make sure the HP 2334A is set to **CONFIGURE** mode by checking the **CPU** cards switch settings. The **CPU** cards **DIP** switches should be set as follows:

```
DIP Switch: A  B  C  D  E  F  G  H
   Setting: X  X  X  X  0  0  0  1  Where: 0 = DOWN / OFF
                                           1 = UP / ON
                                           X = ON or OFF

Switches A, B, C, and D may be set as required.
```

NOTE

When the HP 2334A is in CONFIGURE MODE **only** port A1 is enabled and it is preconfigured at 1200 baud.

Configuration Procedure

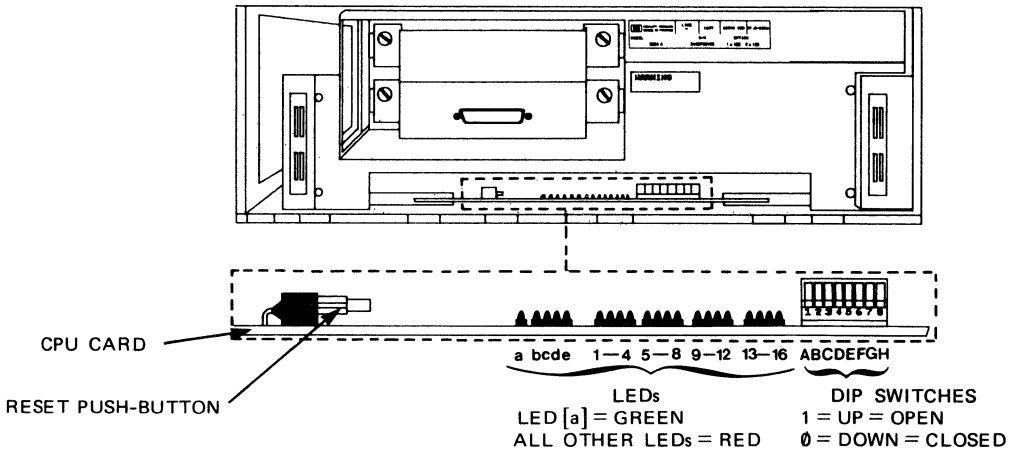
This section covers the following topics:

- Entering the CONFIGURE Mode,
- Sample Off-line Configuration Listing,
- Entering the HSA Command Mode,
- Step-by-step Configuration Procedure for HSA,
- Entering the UDP Command Mode,
- Entering the ASG Command Mode,
- Adding New Entries to the uucp Files,
- Creating New Configuration Files.

Entering the CONFIGURE Mode

To enter **CONFIGURE** mode, simply press the **CPU** card's reset button (or if the HP 2334A is switched OFF, switch it ON). The front panel must be removed to get to the reset button. To remove the front panel, place your fingers in the channel on its upper edge and pull outward. The inside of the HP 2334A looks like the diagram shown below. If you use the ON/OFF switch, it is located on the backside of the HP 2334A.

HP2334A WITH FRONT PANEL REMOVED



NOTE:
1) SWITCHES A, B, C AND D ARE SENSED:
   A) CONTINUOUSLY DURING THE RUN MODE, CONFIGURE MODE, SWITCHES TEST, DEVICE LOOP BACK TEST AND MODEM LOOP BACK TEST.
   B) DURING THE SELF DIAGNOSTIC TESTS, ONLY AT POWER-ON OR AFTER RESET.
2) SWITCHES, E, F, G AND H ARE ONLY SENSED AT POWER-ON OR AFTER RESET.

Figure 8-8. CPU card's Reset Button and LEDs

Once the above process has been executed the HP 2334A goes through a power-on self test that last approximately twenty seconds and causes the following to happen:

1. All of the 21 LEDs shown in the diagram above turn ON (illuminate) for one second, then they turn OFF (extinguish) for one second.
2. Then the LEDs are individually illuminated, starting at LED **a** and going through to LED **16**.

3. The self diagnostic tests are then performed with LEDs **d** or **e** ON (illuminated) to indicate which test routine is active (LEDs **a** and **1** to **16** are OFF). See the diagram below.

```
      LED DISPLAY                    TEST
[a]   [b]   [c]   [d]   [e]
 0     0     0     0     1    - CPU Card Test (10 sec. approx.)
 0     0     0     1     0    - Internal Bus and Device Interface
                                Cards Test (3 sec. approx.)

                               Where: 0 = LED OFF
                                      1 = LED ON
```

   The CPU Card Test is performed first and, if successful, the Internal Bus and Device Interface Cards are then tested.

If your CPU Card Test is **successful**, you may continue with the next section. However, if you had a **failure** or an error was detected, the HP 2334A halts and provides a failure indication as follows:

- LED **a** remains **off** (extinguished).
- LED **d** or **e** remains **on** (illuminated) indicating the failed test routine.
- LEDs **1** to **16** provide a display indicating the type of failure (refer to the chapter "TROUBLESHOOTING").
- LED **b on** indicates an invalid DIP switch setting.

If the HP 2334A halts due to a failure refer to the section, "User Troubleshooting" in the chapter, "OPERATION" or the chapter, "TROUBLESHOOTING". Note that both of these chapters are found in your *HP 2334A MULTIMUX Reference and Service Manual.* If the failure cannot be corrected then contact the nearest HP Sales and Service Office.

Sample Off-line Configuration Listing

The HP 2334A automatically provides the initial off-line configuration listing as shown in this section; however, the listing shown in this section has been filled in with the correct field values for Levels I through III. The remaining fields, with the exception of the "HP 2334 CONFIGURATION X.25 SRA" field, can be filled in after you have read the *HP 2334A MULTIMUX Reference and Service Manual* and you are familiar with the HP 2334A.

The off-line configuration listing is divided into three groups they are as follows:

- The HP 2334A-to-Synchronous Network (X.25) configuration. This includes all the field entries made using the *HSA* command.
- The HP 2334A-to-Device (X.3 parameters) configuration. This configuration is for assigning ports to the various devices which are to be connected to the HP 2334A. The *ASG* command is used to assign PAD or CAS/PAD profiles to the HP 2334A asynchronous ports. Explanations for PAD and CAS/PAD can be found in the chapter, "CONFIGURATION" in the, *HP 2334A MULTIMUX Reference and Service Manual.*
- The hardware installed in the HP 2334A. This lists the ROMs on the CPU card and is followed by a list of the device "interface" and "adapter" card information. Note that no data is displayed if a "device adapter card" is not inserted in the associated slot A, B, C, or D. The "device adapter cards" used with the HP 2334A are identified as follows:

  Direct Connect Adapter Card (HP 40260A)

  Modem Control Adapter Card (HP 40261A)

  Note that the Direct Connect Adapter Card is not implemented for use with your HP 2334A.

The following listing is a sample of what your display would show had you already configured it to the values given. This listing assumes that you are in the United States of America and using TELNET.

```
HSA   02334-80320 . 02334-80330

 HP2334 CONFIGURATION X.25 LEVEL I
      ------------------------------------------------------------------
      -PHYS. LINK  : X.21bis DTE                -LINE SPEED   : 9600
      ------------------------------------------------------------------
 HP2334 CONFIGURATION X.25 LEVEL II
      ------------------------------------------------------------------
      -NETWORK TYPE:  TEL, 12                   -EQUIP. TYPE  : LAP-B DTE
      -FRAME WINDOW:  7                         -TIMER  T1    : 3000  ms
      -RET. CNT N2 :  20                        -I-FRAME      : 131 bytes
      ------------------------------------------------------------------
 HP2334 CONFIGURATION X.25 LEVEL III
      ------------------------------------------------------------------
      -LOCAL ADDRESS : nnnnnnnnnnnnnss
      -WIND. SIZE IN : 2                        -WIND. SIZE OUT: 2
      -THROUGHPUT IN : 9600                     -THROUGHPUT OUT: 9600
      -PACK. SIZE IN : 128                      -PACK. SIZE OUT: 128
      -FIRST PVC     :                          -LAST PVC      :
      -FIRST SVC IN  :                          -LAST SVC IN   :
      -FIRST 2W SVC  : 1                        -LAST 2W SVC   : 64
      -FIRST SVC OUT :                          -LAST SVC OUT  :
      -FIRST POOL PRT: B4                       -LAST POOL PRT : B4
      -D-BIT         : NO
      -PKT. NUMBERING: 8
      -FAC. SUPPORTED:
      -PVC ASSOC. PRT:
      ------------------------------------------------------------------
 HP2334 CONFIGURATION X.25 LUG
      ------------------------------------------------------------------
      -REMOTE ADDRESS
      ------------------------------------------------------------------
 HP2334 CONFIGURATION X.25 SRA
      ------------------------------------------------------------------
      -REMOTE ADDRESS
      ------------------------------------------------------------------
ASG Assignment for each port

    .    1    2    3    4
    D    2    2    2    2
    C    2    2    2    2
    B    2    2    2    2
    A    2    2    2    2
      ------------------------------------------------------------------
CPU    02334-80300 . 02334-80310 .
SC-D               .             .
SC-C               .             .
SC-B               .             .
SC-A   05180-2039  . 05180-2040  . RS232MOD4 ports
```

The following is an explanation of the various sections of the configuration listing:

- The HSA refers to the Synchronous Network Adapter Card which fits into slot "A" of the Network Adapter card cage. This card cage is located behind the front panel of the HP 2334A as seen in the picture below. The two part numbers adjacent to HSA refer to two ROMs located on the CPU card.

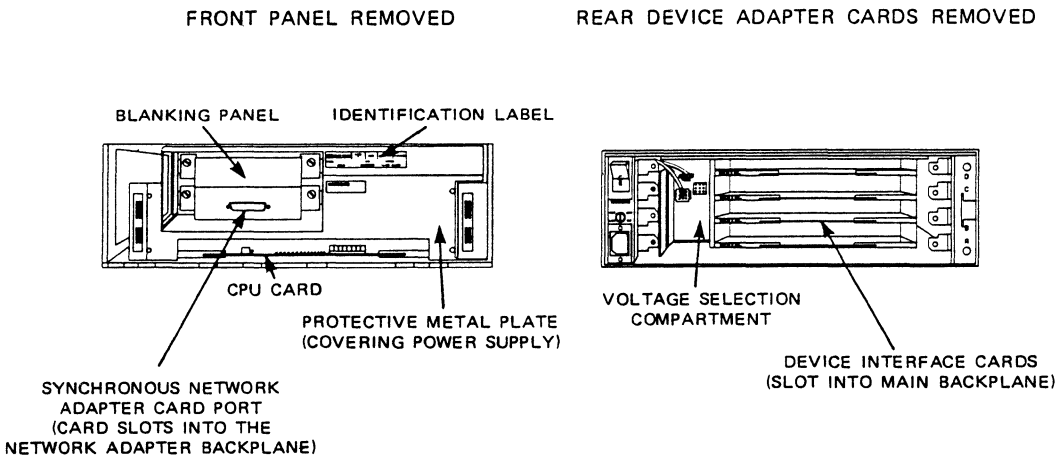


Figure 8-9. Front and Rear View of the HP 2334A

- Parameters associated with the various fields in Levels 1 throught 3 have recommended values assigned to them. These values should be used to configure your HP 2334A for the first time.
- The two part numbers associated with the CPU toward the bottom of the configuration list refer to two more ROMs located on the CPU card.
- The parameters `SC-D`, `SC-C`, `SC-B`, and `SC-A` refer to the Device Adapter Cards inserted in the slots D, C, B, and A respectively. These slots are located on the backplane of the HP 2334A. The values associated with these fields are:

```
The part numbers of the ROMs located on the Device
Interface Cards.


The type of card inserted into the backplane of the
HP 2334A which is the HP 40261A card represented by
this value: RS232MOD 4 ports.
```

Note that the nnnnnnnnnnnnnss values located after various fields in the listing are local network addresses and remote addresses which may be up to 15 digits long the last two digits (ss) being the sub-address of the HP 2334A device ports. For a detailed explanation of this, read the sections, "X.25 Level 3" and "MSG, LUG and SRA CONFIGURATION" found in your, *HP 2334A MULTIMUX Reference and Service Manual.*

Entering the HSA Command Mode

The HP 2334A-to-synchronous network configuration may be defined using the *HSA* command and an HP-UX supported interactive terminal connected to device port A1. The *HSA* command allows the user to configure all the X.25 parameters (Levels 1, 2, and 3), Symbolic Remote Address (SRA) facilities, and Local User Address (LUG) facilities.

Once the configuration listing, which has not been filled in, has been displayed and the user asterisk (*) prompt is obtained, type:

```
HSA Return
```

The *HSA* command you just executed refers to the Synchronous Network Adapter Card mounted in slot "A" of the Network Adapter cage. The following is next displayed as a user prompt:

```
HSA:
```

Step-by-step Configuration Procedure for HSA

This section provides you with a step-by-step procedure for entering the correct values in the fields of the previously shown configuration listing. Note that some of the fields in the configuration listing are preset and are not mentioned in the following procedure. For a detailed explanation of this procedure, you need to read the chapter entitled, "CONFIGURATION" in your *HP 2334A MULTIMUX Reference and Service Manual.*

Note that all commands are entered on the *HSA* command line which is indicated by the HSA: prompt. The step-by-step procedure is as follows:

1. Specify the data transmission rate on the synchronous network connetion, type:

   ```
   LEVEL1 Return
   ```

   The following message is displayed:

   ```
   Line_speed?
   ```

Respond to it by typing:

```
9600 Return
```

2. Enter the fields for Level 2 of the configuration, type:

   ```
   LEVEL2 Return
   ```

   The following message is displayed:

   ```
   NTK_type?
   ```

   If you are in the United States of America, you would respond to the above prompt by typing in the response given below; otherwise, refer to your *HP 2334 MULTIMUX Reference and Service Manual* for the proper response.

   ```
   TEL,12 Return
   ```

   This message is displayed:

   ```
   Frame_window?
   ```

   Respond to it by typing:

   ```
   7 Return
   ```

   This message is displayed:

   ```
   Timer_T1?
   ```

   Respond to it by typing:

   ```
   3000 Return
   ```

3. Enter the values for Level 3 of the configuration, type:

   ```
   LEVEL3 Return
   ```

   The following message is displayed:

   ```
   Lcl Addr.?
   ```

   Respond to it by typing:

   ```
   nnnnnnnnnnnnnss Return
   ```

   where **nnnnnnnnnnnnn** is the local network address which may be be up to 13 digits long and **ss** is the 2 digit sub-address.

The remainder of this message and response sequence is given in a tabular form. To use this table correctly, you should start with the message and response at the top of the table and work your way to the bottom. Note that you will be reading the message in the left column and responding with the prompt in the right column. Press the Return key after typing in your response. There are some cases where you are asked just to press Return without entering a response.

Message Displayed	Response
Thrput in?	9600
Thrput_out?	9600
Wind sz in?	2
Wind sz out?	2
Def/mod vc tbl.?	no
Fst pvc?	Press Return
Fst svc in?	Press Return
Fst 2w svc?	1
Lst 2w svc?	64
Fst svc out?	Press Return
First pool port?	A1
Last pool port?	A1
Neg pk sz?	no
Neg wd sz?	no
Neg thrput?	no
Rev. char. acc.?	no
D-Bit?	no

To verify the field entries you made to your configuration listing, type the following after the HSA: prompt:

```
list Return
```

If a field entry is wrong, you will have to re-enter the command after the HSA: prompt (e.g. level1) which gets you the section containing the field that needs to be changed. Note that there is no way to step to the field you wish to correct. You must re-enter the correct values to all of the fields as the prompts for them appear.

To exit the HSA mode, type: Return

Entering the UDP Command Mode

The UDP primary user command is used to create or modify User Defined Profiles (UDPs). The HP 2334A has several pre-defined BDPs (Basic Defined Profiles) which can be used for many standard applications, bt certain configurations require a special sets of parameters to be defined (e.g. auto-speed, auto-parity or different flow control mechanisms). A good knowledge of the standard X.3 and local parameters is required to avoid creating erroneous UDPs.

To enter the UDP mode, type:

```
udp Return
```

after the * prompt. The following message should appear:

```
Prof number?
```

Respond to this message by typing:

```
2 Return
```

The following message should appear:

```
    -PROFILE : 2                        -FREE SPACE : 43 parameters

    -EXISTING PROFILES : 1,21,31,51,61,71,100,101,121,141

UDP:
```

In response to the UDP: prompt, you **must** type:

```
set 11:12,0:13,14:2 Return
```

this defines profiles for the following "free spaces": 11 and 14 respectively. Note that 0 is a separator and not really a parameter.

To see the newly modified profile listing, type:

```
par? Return
```

The following will appear in your display:

```
PAR 1:1, 2:1, 3:2, 4:0, 5:1, 6:5, 7:21, 8:0, 9:0, 10:0, 11:12,12:1, 13:0, 14:0,
 15:1, 16:8, 17:24, 18:0, 0:13, 1:0, 2:0, 3:0, 4:0, 5:0, 6:0, 7:128, 8:0, 9:0, 1
0:0, 11:0, 12:0, 13:3, 14:2, 15:0, 16:0, 17:0, 18:63, 19:255, 20:0, 21:0, 22:64,
 23:1, 24:0, 25:0
```

To exit the UDP mode, type:

Return

The following will appear:

```
Prof number?
```

Respond to this prompt by typing:

Return

The * should appear in the display. You are now ready to proceed to the next section where you are to enter the ASG mode.

Entering the ASG Command Mode

The ASG primary user command is used to assign PAD or CAS/PAD profiles to the HP 2334A asynchronous ports. Note that a Remote PAD (associated with CAS/PAD) profile is automatically downloaded by a CAS/PAD profile and is not user assigned.

To enter the ASG command mode, type the following after the *:

asg Return

Respond to the ASG: prompt by typing:

list Return

this gives you a listing of the profile assignments of the ports. Your display should look like this:

```
Assignment for each port

.    1    2    3    4

D    1    1    1    1
C    1    1    1    1
B    1    1    1    1
A    1    1    1    1
```

Change all of the port profile assignments to 2 by typing:

a,b,c,d:2 Return

Test to see that the port profiles have been changed type:

```
list Return
```

The display should look like this:

```
Assignment for each port

.    1    2    3    4

D    2    2    2    2
C    2    2    2    2
B    2    2    2    2
A    2    2    2    2
```

Exit the ASG mode by typing:

```
Return
```

The * will appear in the display. Turn your HP 2334A OFF (0) to prepare for the next section.

Adding New Entries to the uucp Files

Before proceeding with this section, you need to remove the front panel cover and set switch 2 on the CPU card to the upward or ON position. All other switches on the CPU card switch packet should be in the down position or OFF.

At this point your HP 2334A has been configured and connected to the X.25 Network. The remaining step-by-step procedure explains how to configure your HP-UX software for use with the X.25 Network.

1. You should have the HP-UX supported terminal which is connected to your HP 2334A set at a baud rate of 2400. Next turn its power on and observe the display the following prompt should appear:

   ```
   @
   ```

   Note that you cannot make a direct connection to the HP 2334A ports through an HP 27130A (8-channel multiplexer) or three of the ports on the HP 98642A (4-channel multiplexer). The port you can use on the HP 98642A is port number 1. You can make direct connection to the HP 2334A ports through an HP 27140A (6-channel multiplexer).

2. All data communication cards should be configured just as if you were going to connect to a modem.

3. Set up the following HP-UX files for uucp use: */dev, /etc/inittab, /etc/passwd, /usr/lib/uucp/L.sys, /usr/lib/uucp/L-devices.* Note that these files are set up in the same manner as was explained in the "Uucp File System" chapter of this manual with the exception of HP 2334A and X.25 naming conventions. The following are examples of how these files should be set up:

   a. You can have up to 16 terminals remotely or locally connected to the ports on the backplane of your HP 2334A. The special (device) file names you give them should be as follows:

      ```
      /dev/x25.n
      ```

      where the `n` is the select code of the computer port.

   b. The */etc/passwd* file should have the following entry already made for *uucp*:

      ```
      uucp:password:5:1::/usr/spool/uucppublic:/usr/lib/uucp/uucico
      ```

      where the `password` is assigned by the system administrator for security purposes.

   c. The */etc/inittab* file should contain entries similar to the following for each incoming port from the HP 2334A:

      ```
      2:00:c:/etc/getx25 x25.0 2 HP2334A
      ```

      where `2` is the state of the HP-UX system. The command to be executed is `/etc/getx25`. The first parameter to the mentioned command is the special (device) file to be executed. The second parameter is the speed indicator (baud rate) for getx25, a value of 2 is common. The final parameter is the name of the PAD device you are connected to: `HP 2334A`.

   d. The */usr/lib/uucp/L.sys* file contains entries similar to the following for each HP 2334A device you are able to communicate with on the X.25 Network:

      ```
      hpbm Any,5 ACUHP2334A 9600 f/45762  login:-EOT-login:uucp Password:xxxx
      ```

   e. The */usr/lib/uucp/L-devices* file contains entries similar to the following:

      ```
      ACUHP2334A x25.0 x25.0 2400
      DIR        x25.0 0     2400
      ```

4. If no changes were made to the files mentioned in step 3 then skip this step. However, if changes were made, enter system state 2 to execute the file changes. To do this, type:

   ```
   init2 Return
   ```

5. To test to see if the *getx25's* entries are there, type:

   ```
   ps -ef Return
   ```

6. Execute the following command line:

   ```
   cu -l<line> -m dir
   ```

   where `<line>` is the device name (i.e. x25.0), without `/dev/`. Your display should display the following:

   ```
   Connected

   ERR

   @
   ```

7. You are now ready to test the your HP 2334A. To do this, type the following:

   ```
   local_network_address Return
   ```

   where `local_network_address` is an address to another unit which you have access to on the X.25 Network. Note that you should receive a COM message indicating a circuit has been established. If you do not receive a COM message, try to reconfigure the HP 2334A. If this does not work, you should call your Local HP Sales or Service Representative for help.

Creating New Configuration Files

After you have successfully completed the steps in the last section and you decide that you want to talk to another kind of PAD, you have to write new configuration files (scripts). They must be placed in */usr/lib/uucp/X25*, and they must be named *.in, *.out, and *.clr, where * is the name of your "modem type" as specified to *uucp*, without the initial ACU. For example,

```
HP2334A.in

HP2334A.out

HP2334A.clr
```

You **do not** have to modify *dialit.c*, since that program now assumes that any unknown modem type is an X.25 PAD, and looks for the appropriate configuration file to control it.

Each configuration file (script) looks something like a shell file, though it's actually interpreted by *opx25*, a program that talks to the PAD and is thus like a telephone operator. You tell what characters to send, and which ones you expect back. Each file has a different purpose:

*.in detect an incoming call *.out make an outgoing call *.clr clear the circuit (hang up)

As stated above, the configuration files are like shell scripts and they are executed using the *opx25* command. The *opx25* command executes HALGOL programs which are scripts used for communicating with devices such as modems and X.25 PADs. The scripts are the configuration files covered in this section.

A configuration file (script) contains lines of the following type:

empty	Lines are ignored.
beginning with /	Lines are ignored (comments).
ID:	Denotes a label. ID is limited to alphanumerics or "_".
send STRING	STRING must be surrounded by " ". The text is sent to the device. Non-printable characters are represented as in C; if you don't know C, just represent each non-printing ASCII char as \DDD, where DDD is the octal ASCII character code. In the *.out file, \# in a string is taken to be the number being dialed.
break	Send a break "character" to the device.
expect NUMBER STRING	Here NUMBER is how many seconds to wait before giving up. 0 means wait forever, but this isn't advised. Whenever STRING appears in the input within the time allotted, the command succeeds. Thus, it isn't necessary to specify the entire string. For example, if you know that the PAD will send several lines followed by an "@" prompt, you could just use "@" as the string. The one exception is in the *.in file, where you need to specify at least the end of the expected input. If you just specify a substring in the middle, the rest of the input remains unread until the logger comes along. The logger reads junk, causing it to repeat its login prompt.
run program args	The program (sleep, date, whatever) is run with the args specified. Don't use "" here. Also, the program is invoked directly (with execp), so wild cards, redirection, etc. are not possible.

error ID	If the most recent expect or run encountered an error, go to the label ID.
exec program args	Like run, but doesn't fork.
echo STRING	Like send, but goes to stderr instead of to the device.
set debug	Sets the program in debug mode. It echoes each line to `/tmp/opx25.log`, as well as giving the result of each expect and run. This can be useful for writing new scripts.
set log	Sends subsequent incoming characters to `/usr/spool/uucp/X25LOG`. This can be used in the `*.in` file as a security measure, since part of the incoming data stream contains the number of the caller. There is a similar feature in *getx25*: it writes the time and the login name into the same logfile.
set numlog	Like "set log", only better in some cases, since it sends only digits to the log file, and not other characters. For the `*.in` file, for example, "set numlog" gives you information about who has called, but in a compact form.
timeout NUMBER	Sets a global timeout value. Each *expect* command uses time in the timeout reservoir; when this time is gone, the program gives up (exit 1). If this command isn't used, there is no global timeout. Also, the global timeout can be reset any time, and a value of 0 turns it off.
exit NUMBER	Exits with this value. 0 is success, anything else is failure. The `*.out` file should observe the convention that an exit value of 1 means you couldn't dial the number, and a value of 2 means that you couldn't get the prompt after dialing the number. You can test configuration files, sort of, by running *opx25* by hand, using the argument "-f" followed by the name of the script file. The program in this case sends to, and expects from, standard output and input, so you can type the input, observe the output, and see messages with the *echo* command.

The content of a configuration file is the HALGOL program (script) you write and place in that file with a file name of your own choosing. These files are executed using the *opx25* command.

Below is a list of the *opx25* command line and its options.

```
opx25 [-fscript] [-cchar] [-onumber] [-inumber] [-nstring] [-d] [-v]
```

where:

`[-fscript]`	causes *opx25* to read the configuration file (`script`) as the input program. If `-f` is not specified then *opx25* reads standard input for the `script`.
`[-cchar]`	causes *opx25* to use `char` as the first character in the input stream instead of actually reading it from the input descriptor. This is useful sometimes when the program that calls *opx25* is forced to read a character but cannot "unread" it.
`[-onumber]`	causes *opx25* to use `number` for the output file descriptor (i.e. the device to use for *send*). The default is 1.
`[-inumber]`	causes *opx25* to use `number` for the input file descriptor (i.e. the device to use for *expect*). The default is 0.
`[-nstring]`	causes *opx25* to save this `string` for use when `\#` is encountered in a *send* command.
`[-d]`	causes *opx25* to turn on the debugging mode.
`[-v]`	causes *opx25* to turn on the verbose mode.

The following configuration file (script) is a sequence that could normally be accomplished by entering a set of PAD commands one at a time; however, to save time, this script was written. The configuration file (script) test a PAD connection to see if a virtual circuit is active at the time you are trying to communicate with it, and if there is a virtual circuit active it will clear it.

```
/ clear the HP2334A
timeout 20
/ ignore garbage
send "\021\021\r"
expectg 2 "**************************************************"

cr:
        send "\r"
        expect 2 "@"
        error brk_clr
        exit 0
brk_clr:
        break
        run sleep 1
        expect 2 "@"
        error dle_clr
        send "CLR\r"
        expect 2 "@"
        error dle_clr
        exit 0

dle_clr:
        send "\020"
        expect 2 "@"
        error cr
        send "CLR\r"
        expect 2 "@"
        error cr
        exit 0
```

Log, Status and Cleanup 9

This chapter discusses the files which contain information about your transactions, the files reflecting system status information, and the programs which compact or delete old or unwanted files. It also contains several shell scripts to implement the clean up operations.

Refer to the appendix, "Log Entry Messages", for an alphabetical listing and interpretation of messages in DIALLOG, LOGFILE and SYSLOG.

Logging Information

The LOGFILE file

A LOGFILE is created and maintained automatically by the *uucp* facility for logging all *uucp* communications and transactions. It is the major source for determining the reason for communications failure. It also keeps track of requests from the local or remote system, the files transferred, the completion or failure of transfers, the success or failure of autodial and the status of *uux* commands. Refer to the *uulog* command in the "Using the Uucp Facility" chapter.

The following example of a weekly log file segments shows the typical messages found in the file. Their interpretations are in the box following the lines from LOGFILE.

```
user system (date-time-PID) log entry
uucp hpfcms (6/5-8:25-23221) OK (startup)
uucp hpfcms (6/5-8:25-23221) OK (conversation complete)
```

remote system called us.

```
uucp hpfclj (6/6-10:46-5183) OK (startup)
uucp hpfclj (6/6-10:46-5183) REQUESTED (S D.hpcnoaB5954 D.hpcnoaB5954 sww)
sww hpfclj (6/6-10:46-5183) COPY (SUCCEEDED)
sww hpfclj (6/6-10:46-5183) REQUESTED (S D.hpfcljX5952 X.hpflcjX5952 sww)
sww hpfclj (6/6-10:46-5183) COPY (SUCCEEDED)
sww hpfclj (6/6-10:46-5183) REQUESTED (S D.hpcnoaB5958 D.hpcnoaB5958 sww)
sww hpfclj (6/6-10:46-5183) COPY (SUCCEEDED)
sww hpfclj (6/6-10:47-5183) REQUESTED (S D.hpfcljX5956 X.hpfcljX5956 sww)
sww hpfclj (6/6-10:47-5183) COPY (SUCCEEDED)
sww hpfclj (6/6-10:47-5183) OK (conversation complete)
```

Remote system `hpfclj` called us. They sent us job `5954`. We sent them `5952`, `5958` and `5956`. Note that the user changed.

```
uucp hpfclj (6/6-13:39-6583) OK (startup)
dmr hpfclj (6/6-13:39-6583) REQUEST (S D.hpfcljB1917 D.hpfcljB1917 dmr)
dmr hpfclj (6/6-13:39-6583) REQUESTED (CY)
dmr hpfclj (6/6-13:39-6583) REQUEST (S D.hpcnoaX1915 X.hpcnoaX1915 dmr)
dmr hpfclj (6/6-13:39-6583) REQUESTED (CY)
dmr hpfclj (6/6-13:39-6583) OK (conversation complete)
```

Remote system `hpfclj` called us. We initiated two copies: `1917` and `1915`.

```
root hpfcld (6/5-6:01-21531) SUCCESSFUL (AUTODIAL)
root hpfcld (6/5-6:01-23531) SUCCEEDED (call to hpfcld)
root hpfcld (6/5-6:01-23531) OK (startup)
root hpfcld (6/5-6:01-23531) OK (conversation complete)
```

We called remote system `hpfcld`. No work requested.

```
root hpfcla (6/5-8:58-24969) FAILED (AUTODIAL)
root hpdcla (6/5-8:58-24969) FAILED (call to hpfcla)
```

The autodial failed to remote system `hpfcla`.

```
root hpdcd (6/5-7:58-24493) FAIL (NO CARRIER DETECTED)
root hpdcd (6/5-7:58-24493) FAILED (call to hpdcd)
```

The autodial to remote system `hpdcd` worked but found no carrier on `hpdcd`.

```
root hpdcd (6/5-8:58-24981) SUCCESSFUL (AUTODIAL)
root hpdcd (6/5-8:58-24981) LOST LINE (LOGIN)
root hpdcd (6/5-8:58-24981) LOST LINE (LOGIN)
root hpdcd (6/5-8:58-24981) FAILED (LOGIN)
root hpdcd (6/5-8:58-24981) FAILED (call to hpdcd)
```

The autodial to remote system `hpdcd` was successful. The login to `hpdcd` failed.

```
root hpfcla (6/5-20:56-127) NO CALL (MAX RECALLS)
root hpfcla (6/5-20:56-127) CAN NOT CALL (SYSTEM STATUS)
```

Local system tried to call `hpfcla`. `STST.*` file indicated that over ten tries were attempted. The dialing try to `hpfcla` was consequently stopped.

```
uucp hpfclj (6/6-10:47-5375) sww XQT (PATH=/bin:/usr/bin;rmail rmd)
uucp hpfclj (6/6-10:47-5375) sww XQT (PATH=/bin:/usr/bin/;rmail dmr)
```

The execution demon *uuxqt* is sending mail from sww on remote system `hpfclj` to `rmd` and `dmr` on the local system.

NOTE

If the communication is initiated from your local system, the time in your `LOGFILE` is that of your local time zone.

If the communication is initiated from a remote system, the time in **your** `LOGFILE` is `EASTERN` time.

The SYSLOG file

The `SYSLOG` keeps track of the number of bytes transferred between systems and the time in seconds it took to complete the transfer. This file is used by the *uusub* command in reporting traffic statistics between various connections. The chapter, "Uucp File System", contains an example of this file.

The DIALLOG file

The `DIALLOG` file is created by the *dialit* module to log information about the modem used, the telephone number dialed and the result of the dialing.

The ownership of the `DIALLOG` file is set at mode 600 (read permission for owner only) by the *dialit* module. Only the owner, *uucp*, should be able to read `DIALLOG` since confidential numbers are listed here.

The following segments are from a typical `DIALOG` file:

```
root ACUVENTEL212 (6/10-6:01-480) SUCCESSFUL (opening of /dev/cua04)
root ACUVENTEL212 (6/10-6:01-480) PHONE OK (phone # 9=226-1111, delay 50 secs)
root ACUVENTEL212 (6/10-6:01-480) MAPPED PHONE - SUCCESS (9&2261111)
root ACUVENTEL212 (6/10-6:01-480) SUCCESS (modem wake up)
root ACUVENTEL212 (6/10-6:01-480) REQUESTED (dial number - 9&2261111)
root ACUVENTEL212 (6/10-6:01-480) ONLINE (remote system)
root ACUVENTEL212 (6/10-6:01-480) SUCCESSFUL (autodial)
```

This section of the `DIALLOG` file indicates that the autodialing sequence was successful.

When the *uucico* demon searched the `L.sys` file for the name of the remote system to contact, a remote system with a modem connection was found. *Uucico* then looked in the `L-devices` file which specified the line parameters to pass to the *dialit* routine. The *dialit* routine not only performed the autodialing sequence, but also made an entry in the `DIALLOG` file.

Status

The `STST` files are the system status files. These files are created in the spool directory by the *uucico* program. They contain information for each remote system, such as *login, dialup* or *retry* time failures. If the *uucp* program is aborted the `STST.*` file remains in the `/usr/spool/uucp` directory. When two systems are conversing, they contain a `TALKING` status.

The form of the file name is:

```
STST.sys_name
```

where `sys_name` is the remote system name.

For ordinary failures, such as *dialup* or *login*, the `STST.*` file prevents repeated tries for about 5 minutes. This is the default time, you can change this time for any system with the `time` field in the `L.sys` file.

When the maximum number of retries (10) is reached, the `STST.*` file must be manually removed before any future attempts to converse can succeed with that remote system. The `STST.*` file is normally deleted by the uudemons after six hours. However you can find information about the transaction in `LOGFILE`.

You can use the uustat command to check on one or all jobs that have been queued. The identification printed when a job is queued is used as a key to query status of the particular job, for example:

```
uustat -j123
```

can return:

```
123 user system 06/08-08:30 06/08-9:46
JOB IS QUEUED
```

You can also use the *uustat* command to check on the status of the last transfer to each system on the network, for example:

```
uustat -mall
```

can return:

```
sys1 06/08-8:50 CONVERSATION SUCCEEDED
```

When sending files to a system that has not been contacted recently, the time of last access to that system can help you determine if the system is not in service.

You can use the *uusub* command to set up and monitor subnetwork statistics. Two files are created by the *uucp* facility: `L_sub` and `R_sub`, which keep track of the defined subnetwork and their corresponding statistics.

Refer to the chapter, "Using the uucp Commands" for more information about *uustat* and *uusub*.

Cleanup

The following shell scripts can be started from the cron program to routinely compact your log files, SYSLOG and LOGFILE, and cleanup any backlog of jobs that were not able to be transmitted to other systems.

Enter the following lines into a file with a name of your choosing:

```
56 6-22 * * * /usr/lib/uucp/uudemon.hr >> /usr/spool/uucp/DEMONLOG 2>&1
0 6 * * * /usr/lib/uucp/uudemon.day >> /usr/spool/uucp/DEMONLOG 2>&1
5 * *1 /usr/lib/uucp/uudemon.wk >> /usr/spool/uucp/DEMONLOG 2>&1
```

Next, type the following:

```
crontab file_name Return
```

```
May 29 15:18 1983  uudemon.hr Page 1


# UNISRC_ID: @(#)uudemon.hr      14.1     83/05/01
#*****************************************************************
#*     (c) Copyright 1983 Hewlett Packard Co.
#*         ALL RIGHTS RESERVED
#*****************************************************************/


#
# This is a example of a demon to run hourly.
# It will clean up bad spool entries an establish
# communications with specified systems. This demon
# normally run at 20 minutes past the hour.

/usr/bin/uulog
/usr/lib/uucp/uuclean -pSTST -pLCK -n6

# the entries below are to contact system you wish contacted on
# an hourly basis
# Uncomment the lines and replace the <nodename> with the proper
# system names of the remotes you wish to contact. Add addition
# entries or delete to match your situation.

#/usr/lib/uucp/uucico -r1 -s<nodename>
#/usr/lib/uucp/uucico -r1 -s<nodename>
#/usr/lib/uucp/uucico -r1 -s<nodename>
```

Note the use of the **-s** system option to force a call to a system which may be PASSIVE (receives calls from other systems) only with respect to you.

```
May 29 15:17 1983  uudemon.day Page 1

# UNISRC_ID: @(#)uudemon.day    14.1    83/05/01
#*****************************************************************
#*    (c) Copyright 1983 Hewlett Packard Co.
#*        ALL RIGHTS RESERVED
#*****************************************************************/


#
# This is am example of a demon which should run daily
# to clean up log system and call those remotes
# you wish once per day. Normally run at 4:00 am.
#
/usr/bin/uulog
cat /usr/spool/uucp/LOGFILE >> /usr/spool/uucp/LOG-WEEK
/usr/lib/uucp/uuclean -pLOGFILE -n0
cat /usr/spool/uucp/SYSLOG >> /usr/spool/uucp/SYS-WEEK
/usr/lib/uucp/uuclean -pSYSLOG -n0
cat /usr/spool/uucp/DIALLOG >> /usr/spool/uucp/DIAL-WEEK
/usr/lib/uucp/uuclean -pDIALLOG -n0
#
# below is the command to callup a system once per
# day. Replace <nodename> by the system name of the
# remote you wish to contact. And uncomment the line.

#/usr/lib/uucp/uucico -r1 -s<nodename>



May 29 15:18 1983  uudemon.wk Page 1

# UNISRC_ID: @(#)uudemon.wk     14.1    83/05/01
#*****************************************************************
#*    (c) Copyright 1983 Hewlett Packard Co.
#*        ALL RIGHTS RESERVED
#*****************************************************************/


#
# This is am example of a demon to be run once per week.
# The entries delete the old weekly log files and clean up
# the /usr/spool/uucp directory.
#
/usr/lib/uucp/uuclean -pTM -pC. -pD. -pLTMP -pLOG. -pdead -pX.
/usr/lib/uucp/uuclean -pLOG-WEEK -pSYS-WEEK -n0
/usr/lib/uucp/uuclean -pDIAL-WEEK -n0
```

Refer to the *uudemons* section of the chapter, "Uucp Facility Demons".

Problems 10

This chapter discusses the most frequent problems you are likely to encounter using the *uucp* facilities.

Bad Connections

A bad connection is the most common problem encountered with the asynchronous terminal emulator (*aterm*) and *uucp* facilities. Both direct and modem connections occasionally have problems in contacting remote systems. If you examine the `LOGFILE` in the `/usr/spool` directory, the remote entry usually points to a bad direct or modem line. You should also check to be sure that you are using compatible modems if that is the type of connection used. Try the *cu* command to interactively attempt to call the other system using the problem line.

When the transaction can only partially complete, the temporary file `TM.*` is not copied into the destination file and remains in your `/usr/spool/uucp` directory.

Out of Space

When the disc containing in the `/usr/spool` directory is out of space work requests cannot be sent or received. This can occur when your system is heavily used or if you have not cleaned up the the non-transmittable files. If your needs do not require that a copy of the file be stored in the spool directory until the transmission is ready to start (the `-C` option in the *uucp* command) use the default *-c* option.

Out-of-date Information

Passwords, logins and phone numbers for remote systems are sometimes changed without your knowing of the change. Be sure that your automatic dialing mechanism does not keep trying to dial an unreachable system.

You should not change the mode (protection) bits on the *uucp* files which are the command modules. For example, the commands need an execution by everybody mode.

Abnormal Termination

DO NOT turn off the power to your computer while *uucp* is running even though *uucp* may be running in the background mode.

Do not press any key on your keyboard while you are using the *cu* command with the "~%take" or "~%put".

NOTES

Glossary

absolute path name - path name beginning with a slash. It indicates that the file's location is given relative to the current root directory, and that the search begins there.

boot - the process of searching for, loading and running an operating system.

boot area - the portion of an SDF mass storage medium containing an executable module. In some implementations this area boots the operating system.

character special file - a special file associated with devices which transfer data character-by-character. Examples are printers, terminals, nine-track magnetic tapes and discs accessed in raw mode.

child - a process spawned via the fork system call.

command - a program which is executed through the shell command interpreter. Arguments following the command name are passed on to the command program. You can write your own command programs, either as compiled programs or as shell scripts written in the shell command language.

command interpreter - a program which reads lines typed at the keyboard or redirected from a file and interprets them as requests to execute other programs. The command interpreter for HP-UX is called the shell.

control character - a member of a character set which may produce some action in a device other than printing or displaying a character. In ASCII code control characters are those in the code range 0 through 31 and 127. Control characters can be generated by simultaneously pressing a displayable character key and CTRL.

crash - a case where the system no longer works either because of its hardware failing or because of an error in the system program itself.

default search path - the sequence of directory prefixes that the shell and other HP-UX commands apply in searching for a file known by an incomplete path name. It is defined by PATH in *environ*. `Login` sets `PATH = :/bin:/usr/bin`, which means that your working directory is the first directory searched, followed by `/bin`, followed by `/usr/bin`. You can redefine the search path by modifying `PATH`, usually done in `/etc/profile` and in `.profile`.

directory - provides the mapping between the names of files and the files themselves. Each user may have a directory of his own files. An HP-UX directory is named and behaves exactly like an ordinary file except that it can be written on only by the file system.

effective group ID - an active process has an effective group ID that is used to determine file access permissions.

effective user ID - an active process has an effective user ID that is used to determine file access permissions and other permissions with respect to system calls.

environment - the set of conditions, such as your working directory, home directory and type of terminal you are using, that exist for any given process.

file - an HP-UX file is simply a stream of bytes. The interpretation of file contents and structure is up to the programs that access the file. file descriptor - a unique identifier of an open file, returned by the file system when a file is opened for reading and writing.

file name - names consisting of up to 14 characters used to name ordinary files, special files or directories.

file system - the directory structure and all associated files.

fork - an HP-UX system call which spawns a new process as an almost exact copy of the current process. This is the only means by which new processes are created in HP-UX.

group - a distinction given to any number of users who may be permitted to access the same set of files. The users in any given group are listed in the "group file": `/etc/group`.

kernel - the most basic part of the HP-UX operating system. The kernel supports the file system, task synchronization, scheduling, communication, I/O, process creation and memory allocation.

library - an archive file containing a set of subroutines and variables which may be accessed by a user program.

link - a directory entry for a file. One physical file may have several links; it may appear in several different directories possibly under different names.

login directory - the directory which becomes the default directory when you first log in. It is specified in **/etc/passwd** on the line beginning with your user name.

login procedure - the procedure required before a user can begin using an HP-UX system. The user must enter a user name and a password (if one exists).

mode - the 16 bits which determine access and execute permission for a file, determine whether to save text image after execution, set effective user and group IDs and indicate file type.

new-line - the character which usually separates lines of characters. It is the same character as line-feed ASCII octal code 12. The terminal driver normally recognizes the carriage-return sent by a terminal as the new-line indicator and translates it into the new-line character.

ordinary file - a type of HP-UX file, created by the user and containing a program or text.

orphan process - a process spawned by a subsequently terminated process. `Init` inherits all orphan processes.

owner - the owner of a file is usually the creator of that file. The ownership of the file can be changed by the super-user or the current owner.

parent directory - a directory's parent directory is the directory one level above it in the file hierarchy.

parent process ID - the process ID of the creator of the process.

password - a string of characters used to verify the identity of a user. Passwords can be associated with users and groups.

path name - a null-terminated character string starting with an optional slash (/) followed by zero or more directory names. These path names are separated by slashes, optionally followed by a file name.

A slash by itself names the root directory.

permission bits - the nine low-order bits of a file's mode. These bits determine read, write and execute permissions for the file owner, group and others.

pipe - an interprocess communication channel used to pass data between two processes.

process - an invocation of a program. There may be several processes all running the same program, but with different data and in different stages of execution.

process group ID - a positive integer identifying the process group each active process belongs to.

process ID - a positive integer identifying each active process in the system.

program - a sequence of instructions to the computer, either in the form of a compiled high-level language or a sequence of shell command language instructions in a text file.

prompt - the character displayed by the shell on the CRT indicating that the previous command has been completed and the system is ready for another command. It is usually a "$" or a "#" but the user can re-define it to be any string.

quit signal - the signal sent by SIGQUIT causing a running program to terminate and generating a file with the core image of the terminated process.

relative path name - the path name not beginning with a slash. It indicates that the file's location is given relative to the current working directory and that the search begins there.

root directory - the highest level directory of the hierarchical file system. In HP-UX the slash (/) character refers to the root directory.

SDF - Structured Directory Format This format provides a structured access to files through the root directory of the volume.

shell - a command language, a command interpreter and a programming language that provides the user interface to the HP-UX operating system.

shell script - a sequence of shell commands and shell programming language constructs, usually stored in a file, for invocation as a user command (program) by the shell.

special file - a file associated with an I/O device. Special files are read and written just like ordinary files, but requests to read and write result in activation of the associated device. Entries for each special file normally reside in the directory `/dev`.

standard error - the default destination file (user's login device) for error and special messages.

standard input - the default source of input data (user's login device) for a program if a source file is not specified.

standard output - the default destination of output data (user's login device) for a program if a destination file is not specified.

super-user - the HP-UX system administrator. This user has access permission to all files and can perform privileged operations.

system call - an HP-UX operating system (kernel) function available to the user through a high-level language.

working directory - the directory in which you currently reside. This is the default directory in which path name searches begin when given a path name does not begin with a slash (/).

NOTES

Series 200/500 Character Codes

B

The following tables list the characters available for the internal printer and display of the Model 520 computer. The representation for a Series 200/500 external printer or terminal may differ. For each character, the tables show its **graphic** representation on the printer, its **description** and its **binary**, **octal**, **decimal** and **hexadecimal** code value.

If it is a control character, it may not print or display unless the DISPLAY FUNCTIONS mode of the device is enabled. The ASCII control characters can be generated by simultaneously pressing CTRL and the key listed in the **CTRL Char** column of the table.

Table B-1. ASCII Character Codes

Graphic	CTRL Char	ASCII Character Abbreviation, Description	Binary	Oct	CHR$ Dec	Hex
^@	@	NUL,Null	00000000	000	000	00
^A	A	SOH,Start Of Header	00000001	001	001	01
^B	B	STX,Start Of Text	00000010	002	002	02
^C	C	ETX,End Of Text	00000011	003	003	03
^D	D	EOT,End Of Transmission	00000100	004	004	04
^E	E	ENQ,Enquiry	00000101	005	005	05
^F	F	ACK,Acknowledge	00000110	006	006	06
^G	G	BEL,Bell,attention signal	00000111	007	007	07
^H	H	BS,Backspace	00001000	010	008	08
^I	I	HT,Horizontal Tab	00001001	011	009	09
^J	J	LF,Line-feed	00001010	012	010	0A
^K	K	VT,Vertical Tab	00001011	013	011	0B
^L	L	FF,Form-feed	00001100	014	012	0C
^M	M	CR,Carriage-return	00001101	015	013	0D
^N	N	SO,Shift Out	00001110	016	014	0E
^O	O	SI,Shift In	00001111	017	015	0F
^P	P	DLE,Data Link Escape	00010000	020	016	10
^Q	Q	DC1,Device Control 1,X-ON	00010001	021	017	11
^R	R	DC2,Device Control 2	00010010	022	018	12
^S	S	DC3,Device Control 3,X-OFF	00010011	023	019	13
^T	T	DC4,Device Control 4	00010100	024	020	14
^U	U	NAK,Negative Acknowledge	00010101	025	021	15
^V	V	SYN,Synchronous idle	00010110	026	022	16
^W	W	ETB,End Transmission Block	00010111	027	023	17
^X	X	CAN,Cancel	00011000	030	024	18
^Y	Y	EOM,End Of Medium	00011001	031	025	19
^Z	Z	SUB,Substitute	00011010	032	026	1A
^[	[	ESC,Escape	00011011	033	027	1B
^\	\	FS,File Separator	00011100	034	028	1C
^]	]	GS,Group Separator	00011101	035	029	1D
^^	^	RS,Record Separator	00011110	036	030	1E
^_	_	US,Unit Separator	00011111	037	031	1F

Table B-1. ASCII Character Codes (continued)

Graphic	ASCII Character Description	Binary	Oct	CHR$ Dec	Hex
	Space, Blank	00100000	040	032	20
!	Exclamation point	00100001	041	033	21
"	Quotation mark	00100010	042	034	22
#	Pound sign, Number sign	00100011	043	035	23
$	Dollar sign	00100100	044	036	24
%	Percent sign	00100101	045	037	25
&	Ampersand	00100110	046	038	26
'	Apostrophe, Acute accent	00100111	047	039	27
(	Opening parenthesis	00101000	050	040	28
)	Closing parenthesis	00101001	051	041	29
*	Asterisk, Star	00101010	052	042	2A
+	Plus	00101011	053	043	2B
,	Comma, Cedilla	00101100	054	044	2C
-	Hyphen, Minus, Dash	00101101	055	045	2D
.	Period, Decimal Point	00101110	056	046	2E
/	Slant, Slash, Solidus	00101111	057	047	2F
0	Zero	00110000	060	048	30
1	One	00110001	061	049	31
2	Two	00110010	062	050	32
3	Three	00110011	063	051	33
4	Four	00110100	064	052	34
5	Five	00110101	065	053	35
6	Six	00110110	066	054	36
7	Seven	00110111	067	055	37
8	Eight	00111000	070	056	38
9	Nine	00111001	071	057	39
:	Colon	00111010	072	058	3A
;	Semicolon	00111011	073	059	3B
<	Less than	00111100	074	060	3C
=	Equals	00111101	075	061	3D
>	Greater than	00111110	076	062	3E
?	Question mark	00111111	077	063	3F
@	Commercial at	01000000	100	064	40
A	Uppercase A	01000001	101	065	41
B	Uppercase B	01000010	102	066	42
C	Uppercase C	01000011	103	067	43
D	Uppercase D	01000100	104	068	44
E	Uppercase E	01000101	105	069	45
F	Uppercase F	01000110	106	070	46
G	Uppercase G	01000111	107	071	47
H	Uppercase H	01001000	110	072	48
I	Uppercase I	01001001	111	073	49
J	Uppercase J	01001010	112	074	4A
K	Uppercase K	01001011	113	075	4B
L	Uppercase L	01001100	114	076	4C
M	Uppercase M	01001101	115	077	4D
N	Uppercase N	01001110	116	078	4E
O	Uppercase O	01001111	117	079	4F

Table B-1. ASCII Character Codes (continued)

Graphic	ASCII Character Description	Binary	Oct	CHR$ Dec	Hex
P	Uppercase P	01010000	120	080	50
Q	Uppercase Q	01010001	121	081	51
R	Uppercase R	01010010	122	082	52
S	Uppercase S	01010011	123	083	53
T	Uppercase T	01010100	124	084	54
U	Uppercase U	01010101	125	085	55
V	Uppercase V	01010110	126	086	56
W	Uppercase W	01010111	127	087	57
X	Uppercase X	01011000	130	088	58
Y	Uppercase Y	01011001	131	089	59
Z	Uppercase Z	01011010	132	090	5A
[	Opening bracket	01011011	133	091	5B
\	Reverse slant, Backslash	01011100	134	092	5C
]	Closing bracket	01011101	135	093	5D
^	Circumflex, Caret, Hat	01011110	136	094	5E
_	Underscore	01011111	137	095	5F
`	Grave accent	01100000	140	096	60
a	Lowercase a	01100001	141	097	61
b	Lowercase b	01100010	142	098	62
c	Lowercase c	01100011	143	099	63
d	Lowercase d	01100100	144	100	64
e	Lowercase e	01100101	145	101	65
f	Lowercase f	01100110	146	102	66
g	Lowercase g	01100111	147	103	67
h	Lowercase h	01101000	150	104	68
i	Lowercase i	01101001	151	105	69
j	Lowercase j	01101010	152	106	6A
k	Lowercase k	01101011	153	107	6B
l	Lowercase l	01101100	154	108	6C
m	Lowercase m	01101101	155	109	6D
n	Lowercase n	01101110	156	110	6E
o	Lowercase o	01101111	157	111	6F
p	Lowercase p	01110000	160	112	70
q	Lowercase q	01110001	161	113	71
r	Lowercase r	01110010	162	114	72
s	Lowercase s	01110011	163	115	73
t	Lowercase t	01110100	164	116	74
u	Lowercase u	01110101	165	117	75
v	Lowercase v	01110110	166	118	76
w	Lowercase w	01110111	167	119	77
x	Lowercase x	01111000	170	120	78
y	Lowercase y	01111001	171	121	79
z	Lowercase z	01111010	172	122	7A
{	Opening (left) brace	01111011	173	123	7B
|	Vertical line	01111100	174	124	7C
}	Closing (right) brace	01111101	175	125	7D
~	Tilde	01111110	176	126	7E
▒	DEL, Delete, Rubout	01111111	177	127	7F

NOTES

Log Entry Messages

C

This appendix provides an alphabetical listing and interpretation of the messages found in the DIALLOG, LOGFILE and SYSLOG files.

/usr/spool/uucp/DIALLOG

Dialit logs dialing status information in the DIALLOG file. Since direct *uucp* connections involve no dialing, they produce no entries in the DIALLOG file. This file grows very quickly. If a collision occurs between two processes wishing to append to the file, one of them starts a DIAL.<pid> file and writes all further messages to that file instead of to DIALLOG. (pid is the process identification number.)

All DIAL* files contain potentially-sensitive information (i.e., phone numbers for other systems) and need to be protected against unauthorized access. Therefore the DIAL* files are owned by user *uucp*; their protection mode is set as unreadable and unwritable by the public when they are created.

Note that the *dialit* routine is user-modifiable by recompiling the source.

Meaning of Entries

The information given here may be inapplicable to a modified version of *dialit*.

The general format for an entry in the DIALLOG file is:

```
user cu_type (month/date-hour:min-pid) message
```

where:

user	is the user who requested the transaction;
cu_type	is the calling unit type, e.g. a device as defined in /ust/lib/uucp/L-devices;
month/date-hour:min	refers to the 24-hour time based on the timezone of the originator process (values can be set by the invoking parent process);

`pid` is the process identifier of the process performing the logging operation (useful for tracing individual process's log entries);

`message` is one of a number of message lines (Refer to the "Message Interpretations" section.).

Sample Entries

These are some sample `DIALLOG` entries.

```
uucp ACUVENTEL (8/24-15:43-8307) SUCCESSFUL (opening of /dev/cul03)
uucp ACUVENTEL (8/24-15:43-8307) PHONE OK (phone # 3524-, delay...)
uucp ACUVENTEL (8/24-15:43-8307) MAPPED PHONE - SUCCESS (3524)
uucp ACUVENTEL (8/24-15:43-8307) SUCCESS (modem wake up)
uucp ACUVENTEL (8/24-15:43-8307) REQUESTED (dial number - 3524)
uucp ACUVENTEL (8/24-15:43-8307) ONLINE (remote system)
uucp ACUVENTEL (8/24-15:43-8307) SUCCESSFUL (autodial)
```

Message Interpretations

`ATTEMPTING (second modem wakeup)` This is logged after the first wakeup attempt fails.

`BAD (phone number - <number>)` The phone number the autodial module used is incorrect.

`ERROR (bad character in phone number <character>)` <character> in phone number is not recognized by dialcodes or dialit module.

`FAILED (autodial)` This message appears as a frequent companion with other `FAILED` messages. The autodial can fail to make a connection for many different reasons: invalid cua device, failure to open the cua special file, an incorrect phone number, no response from the dial prompt, the modem wakeup failed, the attempt to dial failed, the modem type is unknown, slow answer (with carrier), a busy number, or line noise.

`FAILED (connection with remote system)` The autodial module failed to connect to a remote system. This is a secondary entry logged after some other failure.

`FAILED (dial of phone <phone_number>) FAILED (dialing of phone number)` The modem reported dial failure probably due to no answer fast enough (with carrier), a busy number, or line noise.

`FAILED (invalid cua device)` The configuration of the cua device is incorrect. Check the mknod command, the getty entry, the L.sys and the L-devices special file names.

`FAILED (mapping of phone number)` Special characters, such as "=", "-", in the phone number could not be interpreted by the dialcodes module.

`FAILED (modem wake up)` The dialit program could not wake up and synchronize with the modem the first time it tried.

`FAILED (no response from dial prompt)` After waiting for a length of time, there was no response from the dial prompt.

`FAILED (open of cua <device>)` The *uucp* `facility could not open the call unit. Check the mknod` command special file, the getty entry and the `L.sys` and `L-devices` cua field entries.

`FAILED (second wake up)` The dialit program could not wake up and synchronize with the modem the second time it tried.

`MAPPED PHONE - SUCCESS (<mapped phone number>)` The dialit routine succeeded in mapping the phone number as given, containing special characters such as "-" (pause) and "=" (secondary dial tone) separators, into the form the modem understands. This is the form `<mapped phone number>` appears in.

`NOT ATTEMPTING (second wakeup)` This message follows the "FAILED (modem wakeup)" for certain modems.

`NOT KNOWN (modem type specified)` The *dialit* module does not recognize the modem device specified.

`NOT RECEIVED (modem parity message)` The modem is not set for the proper parity.

`ONLINE (remote system)` *Dialit* got a carrier signal from the remote modem.

`PHONE OK (phone # <original phone number>, delay <secs> secs)` Dialit accepted the given phone number as valid, containing "-" (pause) and "=" (secondary dial tone) separators. This is the form `<original phone number>` appears in. `<secs>` is the total computed timeout that dialit allows the modem for dialing the phone number and returning a response.

`REQUESTED (dial phone - <mapped phone number>)` *Dialit* instructed the modem to dial the phone number shown, for the first time. The format of `<mapped phone number>` is the actual form passed to the modem.

`RETRYING (dial number - <mapped phone number>)` *Dialit* instructed the modem to dial the phone number shown, for the second time, after a failure. The format of `<mapped phone number>` is the actual form passed to the modem.

`SUCCESS (modem wake up)` *Dialit* succeeded in resetting and synchronizing with the local modem.

`SUCCESS (modem wake up - second attempt)` *Dialit* succeeded in resetting and synchronizing with the local modem on the second attempt.

`SUCCESSFUL (autodial)` This is the last entry logged for a successful autodial. It means *dialit* terminated and returned "successful".

`SUCCESSFUL (dial phone <phone number>)` The dialit module succeeded in dialing the `<phone number>`.

`SUCCESSFUL (opening of cua device <devicename>)` Dialit managed to open the autodial device `<devicename>` to talk to the modem.

/usr/spool/uucp/LOGFILE

Uucp, uux, uucico, and *uuxqt* log status information here. The file grows very quickly. If a collision occurs between two processes wishing to append to the file, one of them starts a `LOG.<pid>` file and writes all further messages there instead of `LOGFILE`. If *uulog* is invoked with no arguments, it appends all `LOG.*` files to `LOGFILE` and then removes the `LOG.*`. Note: If a `LOG.*` file is active (still in use) when this occurs, the process using it continues to hold the file open and write to it. Since `LOG.*` is unlinked when closed, all information written after the *uulog* executes is lost.

Meaning of Entries

The general format for a `LOGFILE` entry is:

```
user system (month/date-hour:min-pid) message
```

where:

`user`	is the name of the user requesting the transaction;
`system`	is the name of the remote system (may be a null or undefined field if specified by the remote system);
`month/date-hour:min`	is the 24-hour time based on the transaction originator's time zone (The timezone could be set to any value by the invoking parent process but never gets set for transactions initiated by remote systems since their login shell is *uucico.* The times in the file may not be sequential because of old `LOG.*` files appended by *uulog.*);
`pid`	is the process identifier of the logging process (useful for tracing individual process's log entries.);
`message`	is one of a number of message lines (Refer to the "Message Interpretations" section.).

Sample Entries

The following entries are sample entries from a LOGFILE.

```
uucp hp-pcd (8/24-14:34-7710) SUCCESSFUL (AUTODIAL)
uucp hp-pcd (8/24-14:34-7710) SUCCEEDED (call to hp-pcd )
uucp hp-pcd (8/24-14:34-7710) OK (startup)
uucp hp-pcd (8/24-14:34-7710) REQUEST (S D.hp-pcdB1170 ...)
uucp hp-pcd (8/24-14:35-7710) REQUESTED (CY)
uucp hp-pcd (8/24-14:35-7710) REQUEST (S D.hpfclaX1168 ...)
uucp hp-pcd (8/24-14:35-7710) REQUESTED (CY)
uucp hp-pcd (8/24-14:35-7710) OK (conversation complete)
```

Message Interpretations

ACCESS (DENIED) A system tried to access a file for which it did not have file path access permission.

BAD READ (expected <message> got <message>) Transaction terminated abnormally; <message(s)> indicate problem.

CAN NOT CALL (SYSTEM STATUS) A /usr/spool/uucp/STST.<nodename> file still exists for this nodename. The system specified with the [-ssys] option could not be called.

CAUGHT (SIGNAL N) An interrupt, hangup, quit or terminate signal was generated during the uucico operation. N is the signal number.

COPY (FAILED) The system failed to copy a requested file.

COPY (SUCCEEDED) The system succeeded in copying a requested file.

DENIED (CAN'T OPEN) An unauthorized access to a protected file was requested.

DONE (WORK HERE) All local copies are finished.

FAIL (NO CARRIER DETECTED) After an otherwise successful *dialup*, *uucico* checked and found no carrier on a modem line, independent of *dialit*.

FAILED (AUTODIAL) Could not dial another system for some reason; see DIALLOG.

FAILED (CAN'T CREATE TM) The temporary file (used to hold data until the transfer has completed successfully) can not be created.

`FAILED (CAN'T READ DATA)` The input file is protected and can not be opened.

`FAILED (DIALUP ACU write)` An error occurred trying to access the modem.

`FAILED (DIALUP LINE open)` The dial was completed but the line could not be opened.

`FAILED (LOGIN)` Could not successfully negotiate the login sequence specified in the `/usr/lib/uucp/L.sys` file.

`FAILED (call to <nodename>)` Usually a secondary entry, after a different failure entry.

`FAILED (conversation complete)` Usually a secondary entry, after another failure occurred. This could be because a packet of information could not be transferred correctly or the connection had a problem.

`FAILED (startup)` The local and remote systems could not agree on a protocol.

`<file name> XUUCP (DENIED)` A request to copy a protected remote file was denied.

`HANDSHAKE FAILED (BADSEQ)` The remote system sequence number on the local system does not match the local system sequence number on the remote system.

`HANDSHAKE FAILED (CB)` Succeeded in logging in on a remote system, but the other system is set up to call this system back, so the connection failed. Usually the other system then calls back within a short time (usually on a cheaper line or at a higher baud rate).

`LOCKED (call to <nodename>)` A `/usr/spool/uucp/LCK..<nodename>` file already exists for the nodename, due to another conversation already in process or a file left behind due to some sort of abort.

`LOST LINE (LOGIN)` An error occurred during the login process.

`NO (AUTODIAL DEVICE)` There is no available autodial device.
A `/usr/spool/uucp/LCK..<devicename>`
file already exists for every possible device, due to another conversation already in process or a file left behind due to some sort of abort.

`NO (DEVICE)` A device having charactertics matching an entry in the `L-devices` file could not be found.

`NO CALL, MAX RECALLS` The call was attempted ten times without success.

`NO CALL (RETRY TIME NOT REACHED)` A `/usr/spool/uucp/STST.<nodename>` file still exists for this nodename, or, ... and in any case, the retry time specified in `/usr/lib/uucp/L.sys` has not yet been reached.

`NO WORK (<nodename>)` Uucico was initiated for `<nodename>`, but there is no work pending for that system.

`OK (conversation complete)` Normal end of conversation with a remote system.

`OK (startup)` Normal start of conversation with a remote system. If this system is the master, this entry is preceded by other entries; if this system is the slave (it was logged into), this is the first entry for the conversation.

`PERMISSION (DENIED)` An unauthorized access to a protected file was requested.

`PREVIOUS (BADSEQ)` The call to the remote system failed because the remote system's sequence number for the local system did not match the local system's sequence number for the remote system.

`QUE'D (<nodename>)` A copy (*uucp*, not *uux*) operation was queued on the local system, destined for `<nodename>`.

`REQUEST (COPY FAILED <message>)` The message could be any of the following: `<no message>` the reason is not given by remote system can't copy to directory/file - file left in *uucppublic* local access to path denied remote access to path/file denied remote system can't create temporary file system error - illegal *uucp* command generated.

`REQUESTED (CY)` The request for work was completed.

`REQUESTED (file user)` The **user** on a remote system requests the local **file** transferred.

`REQUIRED (CALLBACK)` The remote system called requires that both systems hang up, the remote system then calls the original caller asking the originator to verify its identity.

`REQUEST (S <source filename> <dest filename> <username>)` Start of a transfer from this system to a remote system.

`return_number from system user (MAIL FAIL)` The *mail* command failed and the `return_number` was returned to the sender: `user` on `system`.

`SUCCEEDED (call to <nodename>)` After successful autodial or direct connect, this entry indicates that the local system succeeded in logging in to the remote, but has not (yet) synchronized with the remote *uucico*.

`SUCCESSFUL (AUTODIAL)` This is the first entry for an outbound conversation, where the local system is the master. It means the local system has succeeded in connecting with the remote, but has not (yet) logged in.

`TIMEOUT (AUTODIALER)` The autodial module took longer than timeout value to make a successful connection to a remote system. The timeout value is 30 seconds or five times the length of the phone number, whichever is longer.

`TIMEOUT (DIALUP DN write)` The autodial module took longer than two minutes to make a successful connection to a remote system.

`user XQT (DENIED command)` The **user** tried to execute a **command** on a remote system which was not in the remote system's `L.cmds` file.

`user XQT (path)` The file name which included a path was expanded to the full path name.

`/usr/spool/uucp/LCK.SQ (CAN'T LOCK)` An error occurred five times trying to lock the sequence file.

`WRONG TIME TO CALL (<nodename>)` The `/usr/lib/uucp/L.sys` file does not allow a call to the remote system at this time. This sort of entry appears if the `L.sys` line for the nodename does not parse properly, for reasons ranging from it containing a nodename only (the simplest way to queue work only for a remote site the local system is passive to) up to an error in the syntax given for the legal call times.

`XQT QUE'D (<command line>)` A remote execute (*uux*, not *uucp*) operation was queued on the local system, destined for a remote system nodename.

`<username> XQT (PATH=<pathlist>;<commandline>)` After *uucico* completes a conversation, it starts *uuxqt* to process all `/usr/spool/uucp/X.*` files. One such entry is logged for each `X.*` file processed.

/usr/spool/uucp/SYSLOG

Uucico logs information here about actual bytes transferred. The file grows quickly. In case of a collision some data may be lost.

The general form for a SYSLOG entry is:

```
user system (month/date-hour:min) (seconds)  direction data <bytes> bytes <secs> secs
```

where:

`user`	is the user who requested the transaction (may be a user from the remote system);
`system`	is the name of the remote system (may be null or undefined if specified by the remote system.);
`month/date-hour:min`	is the 24-hour time based on the timezone given by the originator (The timezone could be set to any value by an an invoking parent process. It never gets set for transactions initiated by remote systems, since their login shell is *uucico*.);
`seconds`	is the time in seconds of the current system clock (independent of any timezone and is useful for programs that deal with the data numerically);
`direction`	can be either "sent" or "received";
`bytes`	is the integer number of bytes transferred;
`secs`	is the integer number of elapsed seconds for the transfer.

Index

a

b

c

d

e

f

g

h

i

l

m

n

o

p

r

s

t

u

W

X

Manual Comment Sheet Instruction

If you have any comments or questions regarding this manual, write them on the enclosed comment sheets and place them in the mail. Include page numbers with your comments wherever possible.

If there is a revision number, (found on the Printing History page), include it on the comment sheet. Also include a return address so that we can respond as soon as possible.

The sheets are designed to be folded into thirds along the dotted lines and taped closed. Do not use staples.

Thank you for your time and interest.

Manual Comment Card

If you have any comments or questions regarding this manual, write them on this comment card and place it in the mail. Include page numbers with your comments wherever possible. Enter the last date from the Printing History page on the line above your name. Also include a return address so that we can respond as soon as possible.

HP-UX Concepts and Tutorials
Vol. 5: Data Communications

97089-90060 April 1985

Last Date: _____

(See the Printing History in the front of the manual)

Name: _____

Company: _____

Address: _____

_____

Phone No: _____

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 37 LOVELAND, COLORADO

POSTAGE WILL BE PAID BY ADDRESSEE

Hewlett-Packard Company
Fort Collins Systems Division
Attn: Customer Documentation
3404 East Harmony Road
Fort Collins, Colorado 80525

Manual Comment Card

If you have any comments or questions regarding this manual, write them on this comment card and place it in the mail. Include page numbers with your comments wherever possible. Enter the last date from the Printing History page on the line above your name. Also include a return address so that we can respond as soon as possible.

HP-UX Concepts and Tutorials
Vol. 5: Data Communications

97089-90060 April 1985

Last Date: _____

(See the Printing History in the front of the manual)

Name: _____

Company: _____

Address: _____

_____

Phone No: _____

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 37 LOVELAND, COLORADO

POSTAGE WILL BE PAID BY ADDRESSEE

Hewlett-Packard Company
Fort Collins Systems Division
Attn: Customer Documentation
3404 East Harmony Road
Fort Collins, Colorado 80525

Manual Comment Card

If you have any comments or questions regarding this manual, write them on this comment card and place it in the mail. Include page numbers with your comments wherever possible. Enter the last date from the Printing History page on the line above your name. Also include a return address so that we can respond as soon as possible.

HP-UX Concepts and Tutorials
Vol. 5: Data Communications

97089-90060 April 1985

Last Date: _____

(See the Printing History in the front of the manual)

Name: _____

Company: _____

Address: _____

_____

Phone No: _____

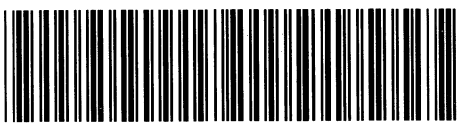
NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 37 LOVELAND, COLORADO

POSTAGE WILL BE PAID BY ADDRESSEE

Hewlett-Packard Company
Fort Collins Systems Division
Attn: Customer Documentation
3404 East Harmony Road
Fort Collins, Colorado 80525



Reorder Number
97089-90060
Printed in U.S.A. 4/85

97089-90604
Mfg. No. Only