Facilities for Series 200, 300 and 500
HP-UX Concepts and Tutorials

HP 9000 Computers

HP Part Number 97089-90081

HEWLETT
PACKARD

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

NOTICE

The information contained in this document is subject to change without notice.

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

WARRANTY

A copy of the specific warranty terms applicable to your Hewlett-Packard product and replacement parts can be obtained from your local Sales and Service Office.

Restricted Rights Legend

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

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

HEWLETT
PACKARD

Customer Note

Hewlett-Packard is in the process of changing the color of our documentation binders. In order to accomplish this changeover we are placing two spine inserts with this manual. Please use the insert that matches the binders you receive.

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

Printed in U.S.A.

Printing History

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

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

November 1985...Edition 1 – Vol. 7: Facilities for Series 200, 300, and 500
July 1986...Update
November 1986...Edition 2 – Update incorporated

Table of Contents

The Series 200 ITE as System Console

Overview of Article 1
Description of the ITE 2
Keyboard Requirements 2
Description of a System Console 3
Additional Considerations 4
Using the Keyboard 6
Character Entry Group 7
Numeric Group 8
Display Control Group 8
Edit Group 13
Function Key Group 14
System Control Group 22
The Display 24
Memory Addressing Scheme 24
Cursor Sensing 26
Cursor Positioning 27
Screen Relative Addressing 27
Absolute Addressing 28
Cursor Relative Addressing 29
Combining Absolute and Relative Addressing 30
Display Enhancements 31
Raster Control 32
Accessing Color 32
Configuring the ITE 37
Configuration Function Keys 37
Terminal Configuration Menu 38
ITE Escape Sequences 46
Sequence Types 46
Keyboard Diagrams for Other Languages 52
HP 98203B Keyboards 52

The Series 200 ITE as System Console

Overview of Article

This article provides you with a description of the ITE as system console for Series 200 computers. If you have a Series 200 computer and you are using it as a terminal connected to an HP-UX system, you need to read the "Terminal Emulator Manual" for the particular terminal your computer is to emulate.

The topics this article covers are as follows:

- Description of the ITE
- Using the Keyboard
- The Display
- ITE Configuration
- ITE Escape Sequences

NOTE

This article **does not** cover the HP 46020A keyboard that is used by the Model 217 and 237 computers. For information on this keyboard read the article, "The Series 300 ITE as System Console" found in the manual, *HP-UX Concepts and Tutorials, Vol. 7.*

Description of the ITE

The Internal Terminal Emulator (ITE) consist of "device driver" code contained in the HP-UX kernel and associated with the built-in keyboard and monitor on a Series 200 computer with memory management. Memory management means the computer contains a central processing unit designed to run the HP-UX system. Your particular keyboard and monitor combination are considered to be an ITE if they meet one requirement given in the section, "Keyboard Requirements" and one requirement given in the section, "Monitor Requirements".

Keyboard Requirements

The keyboard you are using is considered to be an ITE keyboard if it is an:

- HP Model 226 or Model 236 built-in keyboard,
- HP 98203B (Large Keyboard) connected to a Keyboard/HP-IB Interface card (HP part number 09920-66533 or 09920-66534),
- HP 46020A keyboard connected to a Model 217 or Model 237. The use of this keyboard is covered in the *HP-UX Concepts and Tutorials, Vol. 7* article, "The Series 300 ITE as System Console".

Monitor Requirements

The monitor you are using is considered to be an ITE monitor if it is an:

- HP Model 226 or Model 236 built-in monitor,
- HP 82912A or HP 82913A monitor connected to an HP 98204A video board,
- HP 35721A/B/C or HP 35731A/B monitor connected to an HP 98204B video board,
- HP 98781A high resolution monochromatic (black and white) monitor used with a Model 237.

Description of a System Console

The **system console** is a keyboard and monitor (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 boot ROM and HP-UX operating system assign the system console function according to a prioritized search sequence. HP-UX's search for a system console terminates as soon as one of the following conditions is met:

- A built-in serial interface card, HP 98626A, HP 98628A[1], HP 98642A[1], or HP 98644A RS-232C serial interface is present with the "remote bit"[2] set. If more than one is present, the one with the lowest select code is used. In the case of the HP 98642A (4-channel multiplexer), port 1 is used.

- A Model 237 is present or an HP 98700H Display Station is present with its HP 98287A Display Station Interface card configured for "internal" control space. Note that if the HP 98700H display station's display interface card (HP 98287A) is configured for "external" control, it is never chosen ("external" bit-mapped displays have a specific select code address and "internal" bit-mapped displays **do not**). There are two things you should note about the "internal" HP 98700H Display Station:

  - it **is not** supported as a boot device on the Series 200 Revision 3.0 or 4.0 boot ROM. In this case, boot ROM messages would be displayed on the next console found, but HP-UX messages would be displayed on the HP 98700H Display Station.

  - it **cannot** exist "internally" with a Model 237 because the display addresses of both devices collide.

- An HP 98204A/B compatibility video interface or an internal alpha display (e.g. Model 236 display) is present.

[1] The HP 98628A Datacomm Interface Card and the HP 98642A 4-channel Multiplexer Interface with their "remote bit" set are not supported as remote consoles by the Revision A boot ROM; however, it is supported as the system console by the HP-UX operating system. Therefore, when an HP 98628A or HP 98642A card is used and has its "remote bit" set, the boot ROM sends messages to the next console found, but HP-UX sends its messages to the ITE (terminal) associated with the HP 98628A or HP 98642A card.

[2] 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 card the remote bit is set by setting a switch on the board as described in that board's installation manual.

- A built-in serial interface, HP 98626A, HP 98628A, HP 98642A, or HP 98644A RS-232C serial interface is present without the "remote bit" set. If more than one is present, the one with the lowest select code is used. In the case of the HP 98642A (4-channel multiplexer), port 1 is used. The boot ROM **does not** recognize the serial interface card as console when this condition is met; however, HP-UX does.

If none of the above conditions are met, no system console exists. While the boot ROM tolerates this, HP-UX will not.

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 card or an HP 98642A RS-232C Multiplexer card as its display, HP-UX must be installed using either the computer's ITE hardware, an HP 98626A Serial Interface card, or an HP 98644A Serial Interface card.

Additional Considerations

This section contains information on bit settings for the HP 98626A RS-232C Serial Interface Card's Line Control Switch Pack.

The boot ROM requires that the Line Control Switch Pack settings on the HP 98626A RS-232C Serial Interface card be set to the same setting as your ITE (terminal). HP-UX resets these values to system defaults on log in. These values are as follows:

StopBits	should be set to 1.
BaudRate	should be set to 9600 bps.
Parity/DataBits	should be set to 0's/7.
Enq/Ack	should be set to NO.
Pace (Handshake)	should be set to XON/XOFF.

To make the above settings on your HP 98626A Serial Interface card, set your cards Line Control Switch Pack (U-2) switches as follows:

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

Additional settings for Handshake Type bits 6 and 7 of the Line Control Switch Pack, **not** described in the HP 98626A installation manual, are:

Bit 6	Bit 7	Handshake Type
0	0	ENQ/ACK
1	0	XON/XOFF
0	1	NO HANDSHAKE
1	1	NO HANDSHAKE

Note that other switch settings for the Line Control Switch Pack on the HP 98626A card are defined in the installation manual supplied with that card.

Using the Keyboard

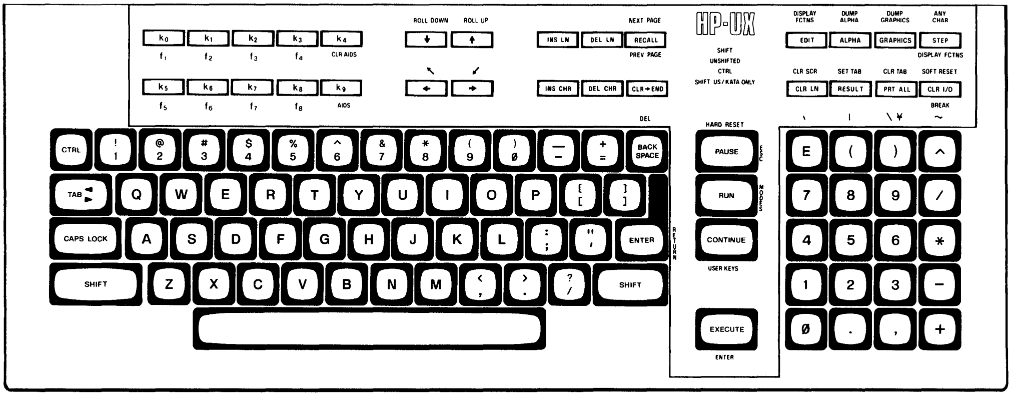
The Series 200 computer keyboard is divided into six functional groups:

- Character Entry Group,
- Numeric Group,
- Display Control Group,
- Edit Group,
- Function Key Group,
- System Control Group.

The remainder of this section covers these key groups.

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.



US ASCII Keyboard and Overlay

If you are using the HP 46020A keyboard with a Model 217 or 237 computer, you can find an explanation of its keys in the article, "The Series 300 ITE as System Console" found in the manual, *HP-UX Concepts and Tutorials, Vol. 7.*

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 uppercase (caps lock on) , this key will give you lowercase letters.

CAPS LOCK — sets the unshifted keyboard to either uppercase (the power-on default) or lowercase (normal typewriter operation) letters.

ENTER RETURN — sends the ReturnDef sequence to the computer (the default is to send $^{\mathrm{C}}{}_{\mathrm{R}}$). 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.

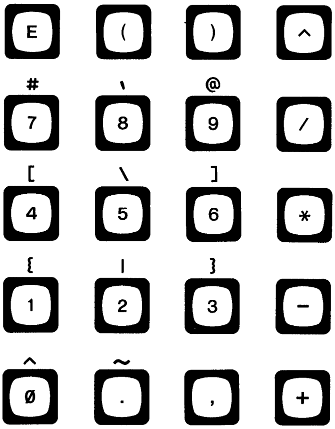
CTRL — provides access to the standard ASCII control characters.

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 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 computers 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 following diagram.



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.

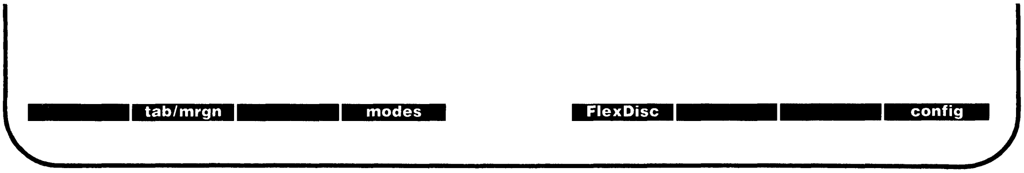
SET TAB
shift—RESULT
sets tab stops. To set a tab stop move the cursor to the desired location, hold the SHIFT key down, and press **SET TAB**.

CLR TAB
PRT ALL
deletes individual 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**.

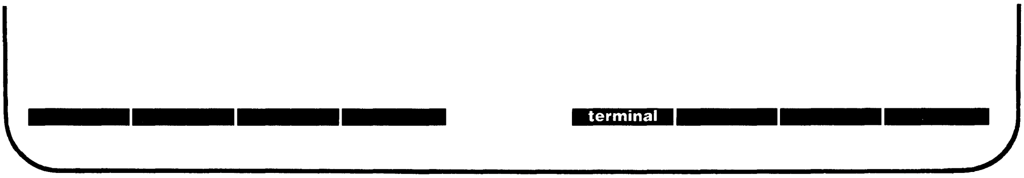
Cursor Control

To use the knob (cursor control wheel) and arrow keys, either have the **transmit functions mode** disabled or be in a program or command, such as **vi** or **ex**, which understand and enable these keys. The **transmit functions mode** specifies whether escape code functions are executed locally at the terminal (ITE) or transmitted to the 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; however, continuously holding an arrow key down or rapidly spinning the knob is discouraged[3]. To disable the **transmit functions mode**:

press: **f9** (labeled **AIDS** on the overlay)



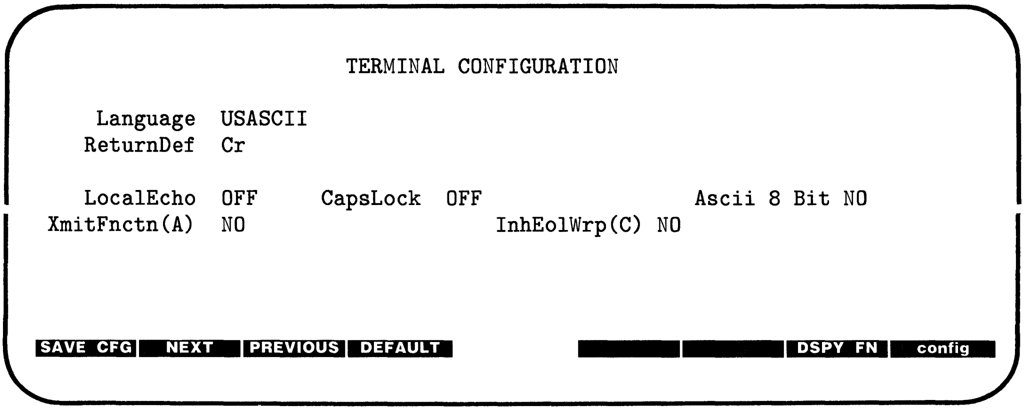
press: **f8** (config)



[3] Use of the knob or arrow keys **is not** recommended in the *vi* editor. On busy systems an escape character created by the knob or arrow keys, while in the *vi* editor, can be lost and misinterpreted by the system. This may result in the loss of data on a given line.

press: **f5** (terminal)

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



```
                              TERMINAL CONFIGURATION

       Language  USASCII
      ReturnDef  Cr

     LocalEcho  OFF      CapsLock  OFF                   Ascii 8 Bit NO
  XmitFnctn(A)  NO                      InhEolWrp(C) NO
```

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

press: **f8** (**config**)

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

press: **f2** (**NEXT**)

This turns off the transmit function mode.

press: **f1 (SAVE CFG)**

This returns you to the shell 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.

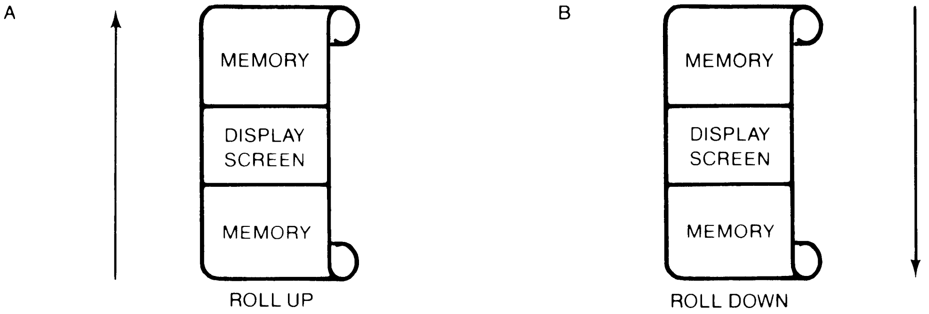
Another method used to move the cursor is the **knob**. Rotating the knob either clockwise or counter-clockwise moves the cursor horizontally. Rotating the knob while pushing the [SHIFT] key moves the cursor vertically until the top or bottom of the display screen is reached, at that time the screen display rolls up or down in display to the next page or previous page. Note that use of the knob **is not** recommended in the *vi* editor.

The [RECALL] key has been redefined to allow you to move to the next or previous page.

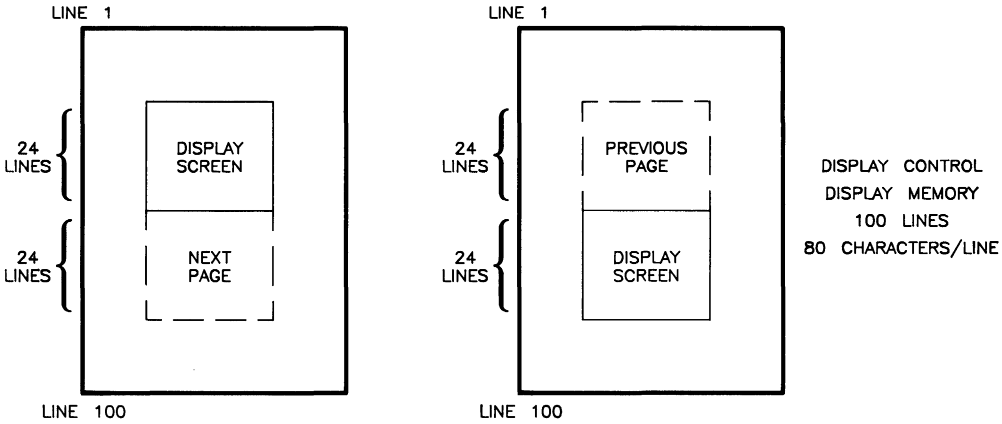
[RECALL]
PREV PAGE moves you back a page within your text.

NEXT PAGE
shift—[RECALL] moves you forward a page within your text.

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 lines of data (23 lines for a Model 226 and 47 lines for a Model 237 high resolution monitor). The current page is that sequence of lines which appears on the screen at any given time. The **previous page** is the preceding 23/24/47 lines in display memory. The **next page** is the succeeding 23/24/47 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.

NOTE

Using these keys while in the *vi* editor without an *.exrc* file can cause loss of important data.

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

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

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

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

CLR LN — performs the same function as CLR→END.

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

DEL CHR — deletes the character at the cursor.

DEL
shift—BACK SPACE — 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 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 the 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] left apostrophe.

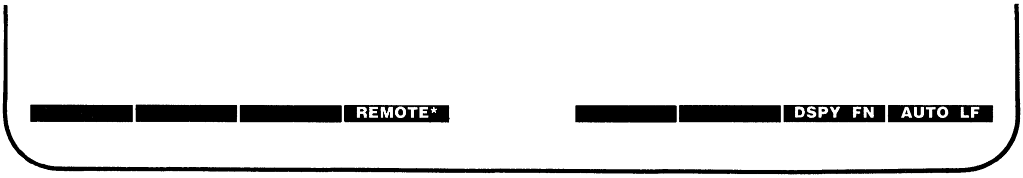
| shift—[(] vertical bar.

\ shift—[)] backslash[4].

~ shift—[^] tilde.

The remainder of this section covers the relationship of the eight function keys, **f1** through **f8**, 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:



[4] This same key sequence returns the yen (¥) character when the ITE is configured to use the KATAKANA language. See later section of this article entitled, “Configuring the ITE”.

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.

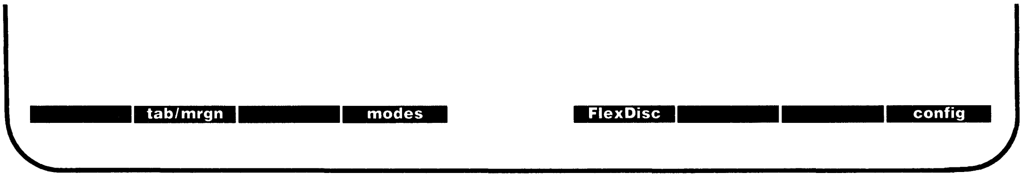
When the **remote mode** is enabled and a key is pressed, the terminal transmits the associated ASCII code to HP-UX. In **local mode** (remote mode is disabled), when a character key is pressed, the associated character is displayed at the current cursor position on the screen and nothing is transmitted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

When the **auto line feed mode** (**AUTO LF**) 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.

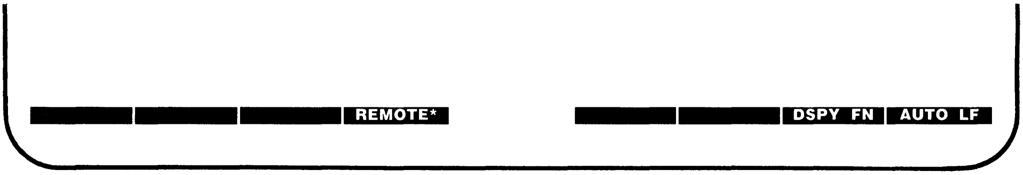
When the **display functions mode** is enabled, the ITE displays ASCII control codes, and escape sequences but does not execute them (except for ^C_R).

k9 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.

`modes` allows access to the remote mode, auto line feed mode, and display functions mode. These modes were previously described in this section. Pressing the function key **f4** or the RUN key labeled **MODES** on the keyboard overlay, you will obtain the following function key labels:



`FlexDisc` pressing this softkey (this only applies to the Model 226 and Model 236 computers) 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 described 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; however, it does return a beep.

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

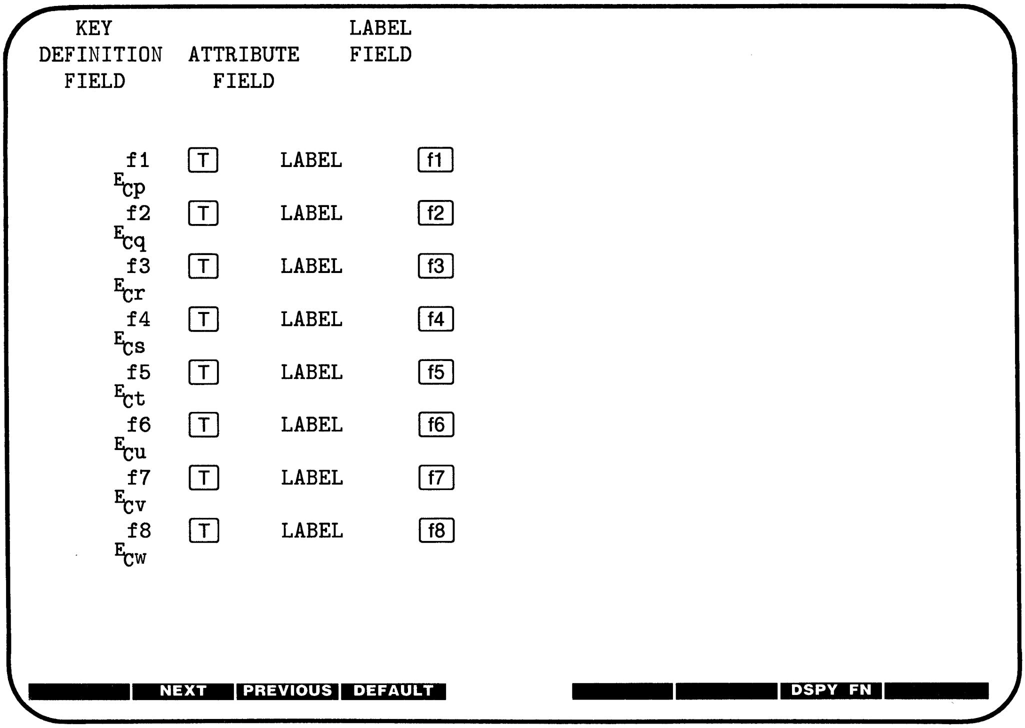
- You can assign to each key a string of ASCII alphanumeric characters and/or control codes (such as carriage return or line feed).
- You can specify each key's operational attribute: whether its key definition is to be executed locally at the terminal, transmitted to the computer, or both.
- You can assign to each key an alphanumeric label (up to 8 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 and CONTINUE (**USER KEYS**) keys simultaneously. The following user keys menu will appear:



```
     KEY                         LABEL
 DEFINITION  ATTRIBUTE    FIELD
   FIELD        FIELD


       f1    T        LABEL       f1
      Ecp
       f2    T        LABEL       f2
      Ecq
       f3    T        LABEL       f3
      Ecr
       f4    T        LABEL       f4
      Ecs
       f5    T        LABEL       f5
      Ect
       f6    T        LABEL       f6
      Ecu
       f7    T        LABEL       f7
      Ecv
       f8    T        LABEL       f8
      Ecw
```

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 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 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**) on Model 226, Model 236, and HP 46020A keyboards.

The ENTER (**RETURN**) key can be used to include carriage return (␍) codes (with display functions mode enabled) in key definitions. If auto line feed mode is also enabled, the ENTER (**RETURN**) key will generate a ␍ ␊, otherwise it is considered a cursor movement key. Note that CTRL-J generates a ␊ code.

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 k9 (**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><key><label length><string length><label><string>

where:

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

$<$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 **f5** user key. The user key (**f5**) is to have attribute "N". Note that $^{E}_{C}$&jB is the escape sequence used in this example to turn on the user key labels.

$^{E}_{C}$&f0a5k5d3LloginTOM$^{E}_{C}$&jB

After issuing the foregoing escape sequence from your program to the terminal, the **f5** portion of the user keys menu is as follows:

```
f5 N LABEL login
TOM
```

If the transmit only attribute (2) is designated, the particular user key will have no effect unless the terminal is in remote.

Controlling the Function Key Labels Programmatically

From a program executing in a host computer, you can control the 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 to 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:

$^{E}{}_{C}$&j@ Enables the user keys and remove all key labels from the screen.

$^{E}{}_{C}$&jA Enables the modes key.

$^{E}{}_{C}$&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. Pressing the ALPHA key toggles the ALPHA display and pressing the GRAPHICS key toggles the GRAPHICS display. Both of these keys may be enabled at the same time.

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
DISPLAY FCTNS
enables **display functions mode** as defined in the function key group. Pressing the **DISPLAY FCTNS** key again cancels the **display functions mode**.

CLR LN
defined in the Edit Group.

CLR SCR
shift—CLR LN
clears the entire alpha portion of the screen display starting from the cursor position.

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 the **CLR TAB** key or the softkey labeled **CLR TABS**.

CLR TAB
shift—PRT ALL

clears a tab previously set at the current cursor position.

SOFT RESET
shift—CLR I/O

does the following:

- Sounds the computer's beeper.
- Disables display functions mode (if enabled).

The data on the screen, all terminal operating modes (except display functions mode), and all active configuration parameters are unchanged. Note that **SOFT RESET** also:

- Restores default color maps.
- Rewrites the whole screen.

HARD RESET
shift—PAUSE

has the same effect as restarting the ITE. 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 (Model 236 only).

CLR I/O
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 "Bourne Shell Programming" in your *HP-UX Concepts and Tutorials*.

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 Series 200 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 consists of the escape character (ASCII 27) followed by at least two subsequent characters.

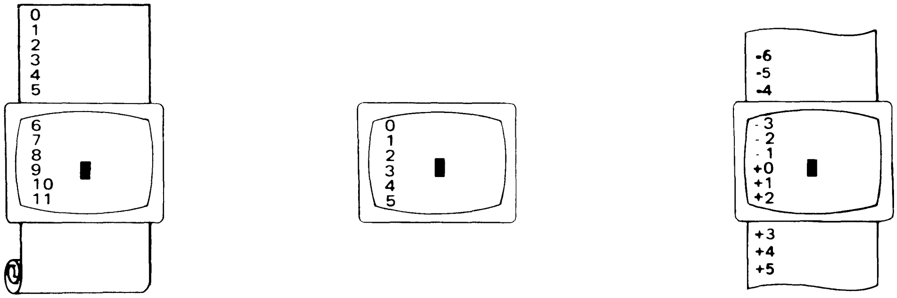
Memory Addressing Scheme

Display memory positions can be addressed using absolute or relative coordinate values. On a Series 200 Medium Resolution monitor, display memory is made up of 80 columns (0 thru 79) and up to four 24-line pages (100 lines of scrolling) with 80 characters per line. A Model 226 has a display memory made up of 50 columns (0 thru 49) and up to four 23-line pages (100 lines of scrolling) with 50 characters per line. On Series 200 High Resolution monitors, display memory is made up of 128 columns (0 thru 127) and up to two 47-line pages (100 lines of scrolling) with 128 characters per line. The types of addressing available are absolute (memory relative), screen relative, and cursor relative.

Row Addressing

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

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



a.) Absolute: row 6

b.) Screen Relative: row 0

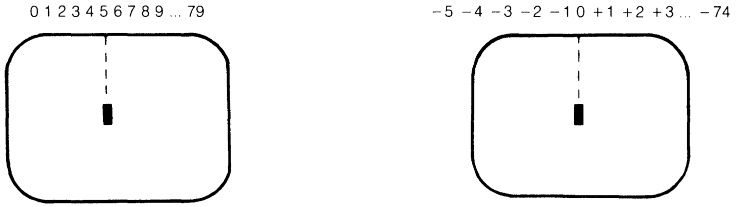
c.) Cursor Relative: row -3

Row Addressing

Column Addressing

Column addressing is accomplished in a manner similar to row addressing. There is no difference between screen 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 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.



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 $^{E}_{C}$a to the terminal, the terminal returns to the program an escape sequence of the form $^{E}_{C}$&axxxcyyyR$^{C}_{R}$, where xxx is the absolute column number and yyy is the absolute row number of the current cursor position. You will later see that this escape sequence is identical to the escape sequence for an absolute move of the cursor.

Relative Sensing

When a program sends the escape sequence $^{E}_{C}$‘, the terminal returns to the program an escape sequence of the form $^{E}_{C}$&a xxxcyyyY$^{C}_{R}$ where xxx is the column number of the cursor and yyy is row position of the cursor relative to screen row 0. This escape sequence is identical to the escape sequence for a relative move of the cursor (discussed later in this article).

Cursor Positioning

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

Screen Relative Addressing

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

$^{E}{}_{C}$&a<column number> c <row number>Y

$^{E}{}_{C}$&a<row number> y <column number>C

$^{E}{}_{C}$&a<column number>C

$^{E}{}_{C}$&a<row number>Y

where:

<column number> is a decimal number specifying the screen column to which you wish to move the cursor. Zero specifies the left-most column.

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

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

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

Example

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

$^{E}{}_{C}$&a6y19C

Absolute Addressing

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

$^{E}_{C}$&a<column number> c <row number>R

$^{E}_{C}$&a<row number> r <column number>C

$^{E}_{C}$&a<column number>C

$^{E}_{C}$&a<row number>R

where:

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

<row number> is a decimal number (0-99) 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, 99 specifies the last.

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:

$^{E}_{C}$&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:

$^{E}_{C}$&a ±<column number> c ±<row number>R

$^{E}_{C}$&a ±<row number> r ±<column number>C

$^{E}_{C}$&a ±<column number>C

$^{E}_{C}$&a ±<row number>R

where:

<column number> is a decimal number 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 against a white background.
- Underline Video - characters are underscored.

NOTE

The Model 226 computer does not provide display enhancements. The Model 220 computer with an HP 98204A composite video card does not provide display enhancements.

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:

^{E}c&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:

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

Enhancement Character

Note that the escape sequence for "end enhancement" (^{E}c&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:

$^{E}_{C}$*dE Restores the alphanumeric display; enables writing to the alphanumeric display (this returns the screen, on bit-mapped displays).

$^{E}_{C}$*dF Disables further writing to the alphanumeric display (this inhibits any further output with bit-mapped displays).

Accessing Color

The monitor used for accessing color is the built-in color monitor of the Model 236. To access color on a Series 200 computer, you must understand some simple terms.

Color pair - two colors which define the foreground color (color of the characters) and the background color, respectively.

For Model 236 computers at least one of the color pairs 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. Note that this is not true for a bit-mapped color display.

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.

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:

$^{E}_{C}$&v <n> <parameter>

where:

<n> is the pen number,

<parameter> is a single character that specifies the action.

To select a pre-defined pen number, the necessary <parameter> is S. Thus,

$^{E}_{C}$&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 "ITE" uses the last notation specified, or RGB notation at power-up). To select a notation type, use the $^{E}_{C}$&v escape sequence used above:

$^{E}_{C}$&v <n> <parameter>

where:

<n> is 0 (for RGB) or 1 (for HSL),

<parameter> is a single character that specifies the action.

To select a notation, the necessary <parameter> is M. Thus, the sequence

$^{E}_{C}$&v 1M

selects HSL notation. It does nothing more.

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

a specifies red (hue) used in the foreground.

b specifies green (saturation) used in the foreground.

c specifies blue (luminosity) used in the foreground.

x specifies red (hue) used in the background.

y specifies green (saturation) used in the background.

z specifies 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 a 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:

$^{E}{}_{C}$&v <0|1>m <n>a <n>b <n>c <n>x <n>y <n>z <pen#>I where either a 0 or a 1 precedes the m parameter (selecting either RGB or HSL notation, respectively), and n is one of the legal values from the tables. <pen#> is an integer in the range 0 thru 7 which, precedes the **i** parameter, 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

```
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 Model 236 computers 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 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

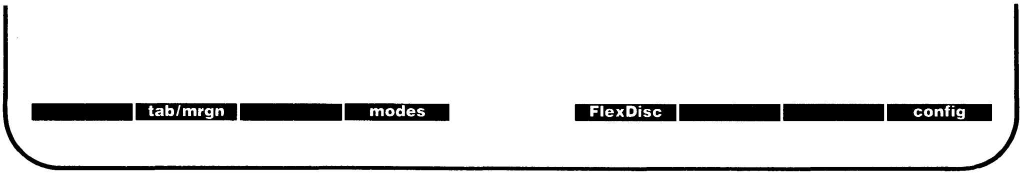
Supplying neither a foreground nor a background color when defining a color pair causes both the foreground and background to be black.

Configuring the ITE

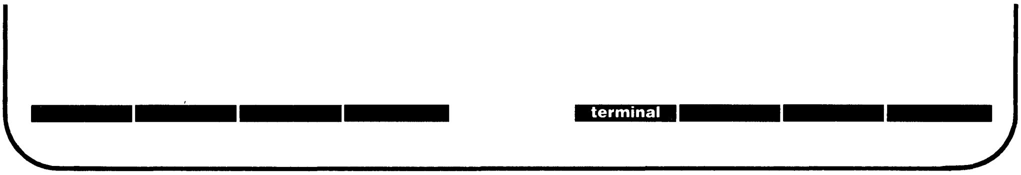
The Internal Terminal Emulator is designed so that the various ITE characteristics can be configured quickly by displaying configure "menus" on the screen and then using system function keys to change the content of these menus.

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:



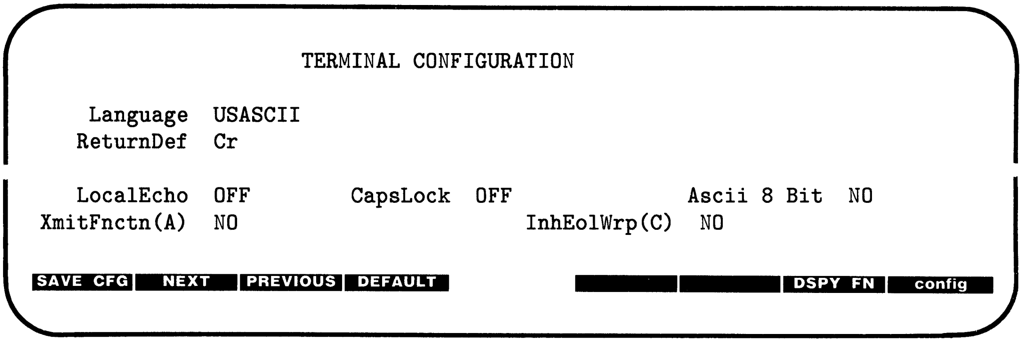
The function keys f2 (tab/mrgn), f4 (modes) and f5 (FlexDisc) were covered in the section "Using the Keyboard" along with a brief discussion on the function key f8 (config). When you press f8 (config) a new softkey display appears at the bottom of your screen.



Pressing f5 (terminal) fills the display with the **Terminal Configuration Menu** which is covered next.

Terminal Configuration Menu

After pressing [f5] (terminal) your display should look like this:



```
                         TERMINAL CONFIGURATION

     Language  USASCII
    ReturnDef  Cr

    LocalEcho  OFF        CapsLock  OFF              Ascii 8 Bit  NO
 XmitFnctn(A)  NO                          InhEolWrp(C)  NO
```

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.

`LocalEcho` is enabled using this escape sequence:

$^{E}_{C}$&k1L

When this field is enabled characters entered through the keyboard are both displayed on the screen and transmitted to the host computer. The default condition for `LocalEcho` is disabled or OFF, because HP-UX will echo the character's itself. The following escape sequence disables `LocalEcho`:

$^{E}_{C}$&k0L

`CapsLock` determines whether the ITE generates the full 128-characters ASCII set or only Teletype-compatible codes. The following escape sequence enables (turns ON) `LocalEcho`:

$^{E}{}_{C}$`&k1C`

When `CapsLock` is enabled 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 key for generating ˜ and ' is disabled.

The following escape sequence disables (turns OFF) `CapsLock`:

$^{E}{}_{C}$`&k0C`

When `CapsLock` is disabled the ITE generates the full 128-character ASCII set of codes. Note that the default condition for `CapsLock` is disabled or OFF.

`XmitFnctn(A)` determines whether escape code functions are executed at the ITE and transmitted to the host computer. The following escape sequence enters YES in the `XmitFnctn` field:

$^{E}{}_{C}$`&s1A`

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.

The following escape sequence enters NO in the `XmitFnctn` field:

$^{E}{}_{C}$`&s0A`

When NO is enter in this field 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.

The default condition for this field is NO.

`InhEolWrp(C)` designates whether or not the end-of-line wrap is inhibited. The following escape sequence enters NO in the `InhEolWrp(C)` field:

$^{E}{}_{C}$`&s0C`

With NO in the `InhEolWrp(C)` field, 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).

The following escape sequence enters YES in the `InhEolWrp(C)` field:

$^{E}{}_{C}$`&s1C`

With YES in the `InhEolWrp(C)` field, 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).

The default condition for this field is NO.

`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[5] layout with mutes)

FRANCAIS qwM (French QWERTY layout with mutes)

FRANCAIS az (French AZERTY[5] 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.

[5] 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 except on the Models 217 and 237 which use the HP 46020A keyboard described in the article, "The Series 300 ITE as System Console" found in the manual, *HP-UX Concepts and Tutorials Vol. 7.*

Series 200 computers support three character sets which contain the special characters associated with all of the international languages. These character sets are: Extended Roman, Roman 8 and US/KATAKANA.

The HP 98204A video interface board **does not** provide the Roman 8 character set. This interface board provides the Extended Roman and the US/KATAKANA Character sets. The only difference between the Extended Roman and Roman 8 character sets are six additional characters included with the Roman 8 character set. These characters have been shaded in on the Roman 8 character set chart provided in this section.

To get the Roman 8 character set you must have an HP 98204B video interface board with its CHAR SELECT switch set to the **1** or ON position. If on this same board you want the KATAKANA character set, you need to set the CHAR SELECT switch to the 0 or the OFF position. Note that there is no CHAR SELECT switch on the HP 98204A board.

The following charts show the character sets previously mentioned. The 8-bit code can be accessed by configuring the field **ASCII 8 Bit** to **YES**.

NOTE

Several languages are available for both the HP 98203A and HP 46020B keyboards. The default ITE language is determined by the language of the keyboard.

ROMAN8 CHARACTER SET
(USASCII PLUS ROMAN EXTENSION)

				b8	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
				b7	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
				b6	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
				b5	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
b4	b3	b2	b1		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	NUL	DLE	SP	0	@	P	‘	p				¯	â	Å	Á	Þ
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q			À	Ý	ê	î	Ã	þ
0	0	1	0	2	STX	DC2	"	2	B	R	b	r			Â	ý	ô	Ø	ã	•
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s			È	°	û	Æ	Ð	µ
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t			Ê	Ç	á	å	đ	¶
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u			Ë	ç	é	í	Í	ˡo
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v			Î	Ñ	ó	ø	Ì	—
0	1	1	1	7	BEL	ETB	’	7	G	W	g	w			Ï	ñ	ú	æ	Ó	¼
1	0	0	0	8	BS	CAN	(	8	H	X	h	x			´	¡	à	Ä	Ò	½
1	0	0	1	9	HT	EM	)	9	I	Y	i	y			`	¿	è	ì	Õ	ª
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z			ˆ	¤	ò	Ö	õ	º
1	0	1	1	11	VT	ESC	+	;	K	[	k	{			¨	£	ù	Ü	Š	«
1	1	0	0	12	FF	FS	,	<	L	\	l	|			˜	¥	ä	É	š	■
1	1	0	1	13	CR	GS	-	=	M	]	m	}			Ù	§	ë	ï	Ú	»
1	1	1	0	14	SO	RS	.	>	N	∧	n	~			Û	ƒ	ö	β	Ÿ	±
1	1	1	1	15	SI	US	/	?	O	_	o	DEL			₤	¢	ü	Ô	ÿ	

Roman 8 Character Set

To use the Roman 8 or Extended Roman Character Set, configure your TERMINAL CONFIGURATION menu to the language you want, ASCII 8 Bit should be set to YES and your terminal (ITE) should be in the remote mode. Note that your terminal does not necessarily have to have REMOTE enabled, but if it is not, then the HP-UX environment must be set up to handle 8-bit characters. All characters for the various languages mentioned in the menu are now available to you except KATAKANA.

KANA8 CHARACTER SET
(JISCII PLUS KATAKANA)

				b_8	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
				b_7	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
				b_6	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
				b_5	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
b_4	b_3	b_2	b_1		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	NUL	DLE	SP	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	ﾞ		
1	1	1	1	15	SI	US	/	?	O	_	o	DEL	▨	▨	ｯ	ｿ	ﾏ	ﾟ		▨

US/KATAKANA Character Set

To use the US/KATAKANA Character Set, configure your TERMINAL CONFIGURATION menu to the language KATAKANA, ASCII 8 Bit should be set to YES and your terminal (ITE) should be in the remote mode. To type US characters, press $\boxed{\text{CTRL}}$ - $\boxed{,}$. If you want KATAKANA characters, press $\boxed{\text{CTRL}}$ - $\boxed{.}$.

For the French keyboard layouts, the AZERTY[6] 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 (ˆ and ¨ on the French layout and ' on the Spanish). If the mutes are enabled, those keystrokes will NOT move the cursor or generate a character. If you then type an applicable vowel, the vowel will appear in the same character position with the accent and the cursor then will move to the next column. If you type any character other than an applicable vowel or a space, however, the character will be displayed and the accent key ignored. If you type a space after a mute accent, the accent alone will appear in that space.

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

Values: YES ($^{E}_{C}$`&k 1I`) = 8-bit codes.
NO ($^{E}_{C}$`&k 0I`) = Standard 7-bit codes.

Changing the Fields

To change the fields in the Terminal Configuration Menu, 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 `USASCII` is displayed in the `Language` field, pressing **NEXT** or f2 would change the field to `VLAAMS`.

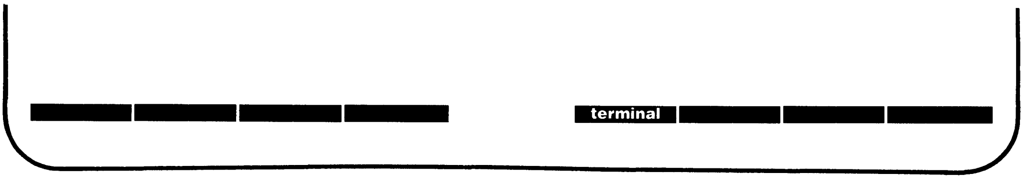
[6] The AZERTY characters can be obtained only through a software configuration of the keyboard. Physical AZERTY keyboards or hardware are not available on Series 200 computers except on the Models 217 and 237 which use the HP 46020A keyboard described in the article, "The Series 300 ITE as System Console" found in the manual, *HP-UX Concepts and Tutorials Vol. 7*.

PREVIOUS changes the setting of the field you are presently in to the previous setting in that field. For example, with FRANCAIS qzM displayed in the Language field, pressing **PREVIOUS** or f3 would change the field back to USASCII.

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 which 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 enables 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 HP-UX through use of escape sequences. The effect is identical to using non-ASCII keys on the ITE keyboard. Escape sequences consist of the escape character followed by one or more visible (non-control) ASCII characters.

The sequences listed in this section are recognized and executed by the ITE whether they are received from the data communication link or from the keyboard (in remote mode), 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 (␛) 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 character in order to define the operation. Absolute and relative cursor addressing are examples of operations that require longer control sequences.

Escape Sequences for ITE Control

Escape Code	Function
${}^{E}{}_{C}$1	Set tab
${}^{E}{}_{C}$2	Clear tab
${}^{E}{}_{C}$3	Clear all tabs
${}^{E}{}_{C}$4	NOT IMPLEMENTED (Set left margin)
${}^{E}{}_{C}$5	NOT IMPLEMENTED (Set right margin)
${}^{E}{}_{C}$9	NOT IMPLEMENTED (Clear all margins)
${}^{E}{}_{C}$@	NOT IMPLEMENTED (Makes the terminal program wait approximately one second.)
${}^{E}{}_{C}$A	Cursor up
${}^{E}{}_{C}$B	Cursor down
${}^{E}{}_{C}$C	Cursor right
${}^{E}{}_{C}$D	Cursor left
${}^{E}{}_{C}$E	Hard reset (power on reset of ITE)
${}^{E}{}_{C}$F	Cursor home down
${}^{E}{}_{C}$G	Move cursor to the left margin
${}^{E}{}_{C}$H	Cursor home up
${}^{E}{}_{C}$I	Horizontal tab
${}^{E}{}_{C}$J	Clear screen from cursor to the end of memory.
${}^{E}{}_{C}$K	Clear line from cursor to end of line
${}^{E}{}_{C}$L	Insert line
${}^{E}{}_{C}$M	Delete line
${}^{E}{}_{C}$P	Delete character
${}^{E}{}_{C}$Q	Start insert character mode
${}^{E}{}_{C}$R	End insert character mode
${}^{E}{}_{C}$S	Roll up
${}^{E}{}_{C}$T	Roll down
${}^{E}{}_{C}$U	Next page
${}^{E}{}_{C}$V	Previous page

Escape Sequences for ITE Control (continued)

Escape Code	Function
$^{E}_{C}$W	NOT IMPLEMENTED (Format mode on)
$^{E}_{C}$X	NOT IMPLEMENTED (Format mode off)
$^{E}_{C}$Y	Enables display functions mode
$^{E}_{C}$Z	Disables display functions mode
$^{E}_{C}$[	NOT IMPLEMENTED (Start unprotected field)
$^{E}_{C}$]	NOT IMPLEMENTED (End unprotected/transmit-only field)
$^{E}_{C}$^	NOT IMPLEMENTED (Primary terminal status request)
$^{E}_{C}$‘	Sense cursor position(relative)
$^{E}_{C}$a	Sense cursor position (absolute)
$^{E}_{C}$b	NOT IMPLEMENTED (Unlock keyboard)
$^{E}_{C}$c	NOT IMPLEMENTED (Lock keyboard)
$^{E}_{C}$d	NOT IMPLEMENTED (Transmit a block of text to computer)
$^{E}_{C}$f	NOT IMPLEMENTED (Modem disconnect)
$^{E}_{C}$g	Soft reset (of ITE)
$^{E}_{C}$h	Cursor home up.
$^{E}_{C}$i	Back tab
$^{E}_{C}$j	NOT IMPLEMENTED (Begin User Key Definition mode)
$^{E}_{C}$k	NOT IMPLEMENTED (End User Keys Definition mode)
$^{E}_{C}$l	NOT IMPLEMENTED (Begin Memory Lock mode)
$^{E}_{C}$m	NOT IMPLEMENTED (End Memory Lock Mode)
$^{E}_{C}$p	Default value for softkey **f1**
$^{E}_{C}$q	Default value for softkey **f2**
$^{E}_{C}$r	Default value for softkey **f3**
$^{E}_{C}$s	Default value for softkey **f4**
$^{E}_{C}$t	Default value for softkey **f5**
$^{E}_{C}$u	Default value for softkey **f6**
$^{E}_{C}$v	Default value for softkey **f7**
$^{E}_{C}$w	Default value for softkey **f8**
$^{E}_{C}$z	NOT IMPLEMENTED (Initiate terminal self test)
$^{E}_{C}$~	NOT IMPLEMENTED (Secondary terminal status request)

Extended Escape Sequences

Escape Code Sequence	Function
^E_C&a <col>c <row>Y	Moves the cursor to column "col" and screen row "row" on the screen (screen relative addressing). The row or column values may be used by themselves in this escape sequence (e.g ^E_C&a <col> C or ^E_C&a <row> Y). The value you choose to leave out remains as previously set.
^E_C&a <col>c <row>R	Moves the cursor to column "col" and row "row" in memory (absolute addressing). The row or column values may be used by themselves in this escape sequence (e.g ^E_C&a <col> C or ^E_C&a <row> R). The value you choose to leave out remains as previously set.
^E_C&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. The row or column values may be used by themselves in this escape sequence (e.g ^E_C&a ±<col> C or ^E_C&a ±<row> Y). The value you choose to leave out remains as previously set.
^E_C&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. The row or column values may be used by themselves in this escape sequence (e.g ^E_C&a ±<col> C or ^E_C&a ±<row> R). The value you choose to leave out remains as previously set.
^E_C&q0L	NOT IMPLEMENTED (Unlock configuration)
^E_C&q1L	NOT IMPLEMENTED (Lock configuration)
^E_C&d	<char> Selects the display enhancement indicated by <char> to begin at the present cursor position. <char> can be @ or A thru O.

Extended Escape Sequences (continued)

Escape Code Sequence	Function
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	ASCII 8 Bits 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
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

Extended Escape Sequences (continued)

Escape Code Sequence	Function
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	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.
EC&v n<parameter>	Read the section in this article "Accessing Color".

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.

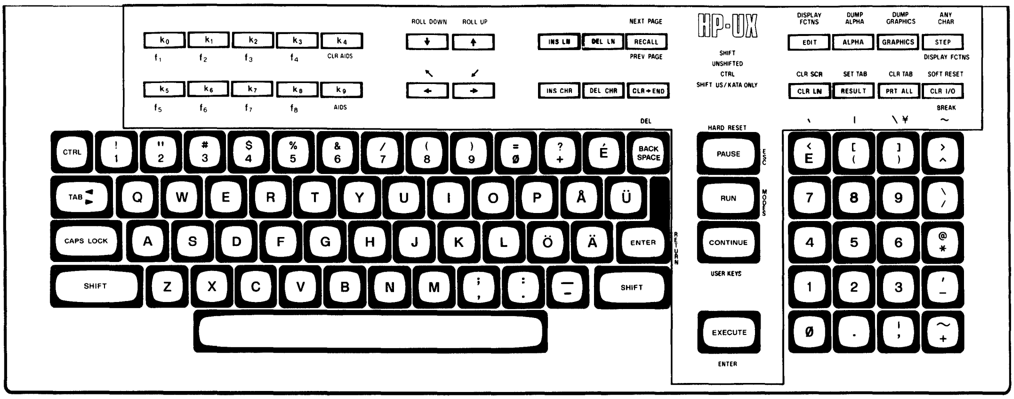
Keyboard Diagrams for Other Languages

This section shows diagrams of the types of HP 98203B keyboards which are available to the Series 200 user. The keyboard option you now have can be configured to any of the languages by use of the TERMINAL CONFIGURATION menu previously mentioned in this article.

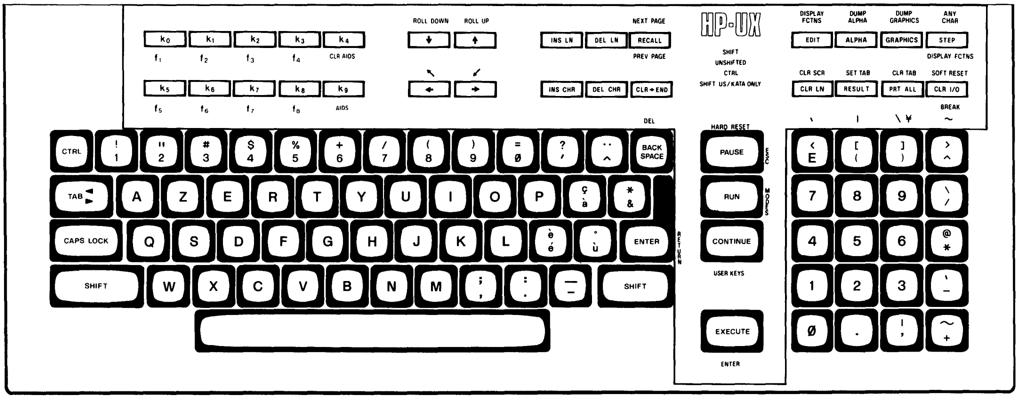
To change to another keyboard language, use the terminal configuration menu and select the **Language** field you want. Then change the **ASCII 8 Bit** field to YES. Next, save the changed configuration menu. Note that the languages accessed through the terminal configuration menu are not available in 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 or purchase a keyboard appropriate to the language you are using.

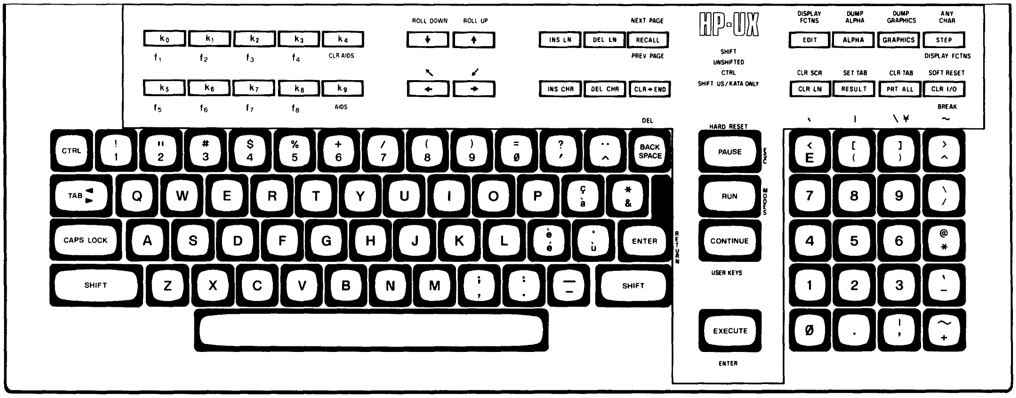
HP 98203B Keyboards



SVENSK/SUOMI (Swedish/Finnish)

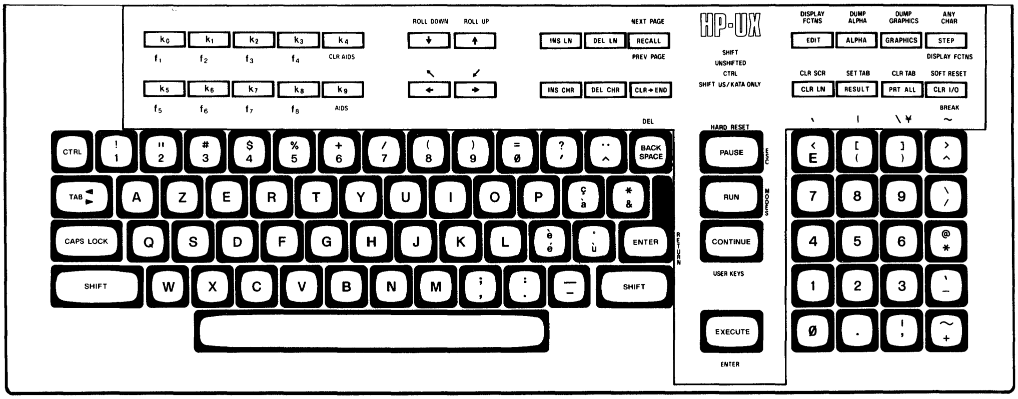


FRANCAIS azM (French AZERTY[7] layout with mutes is not available)

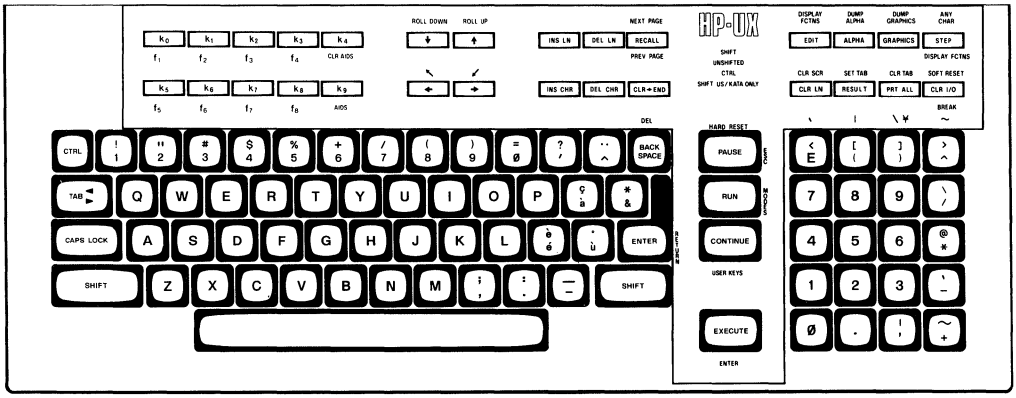


FRANCAIS qwM (French QWERTY layout with mutes)

[7] 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 except on the Models 217 and 237 which use the HP 46020A keyboard described in the article, "The Series 300 ITE as System Console" found in the manual, *HP-UX Concepts and Tutorials Vol. 7.*

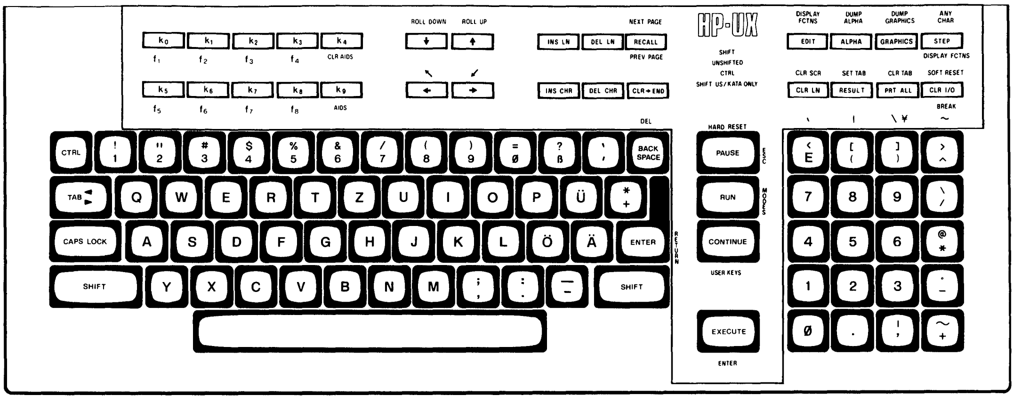


FRANCAIS az (French AZERTY[8] layout is not available)

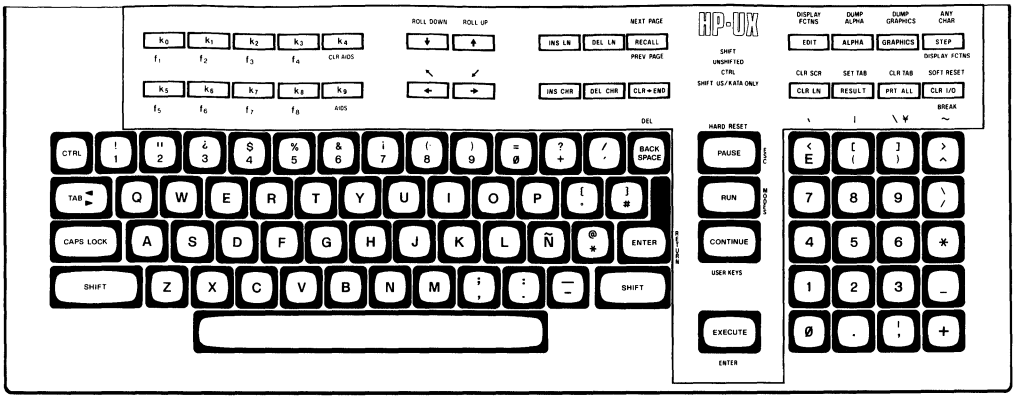


FRANCAIS qw (French QWERTY layout)

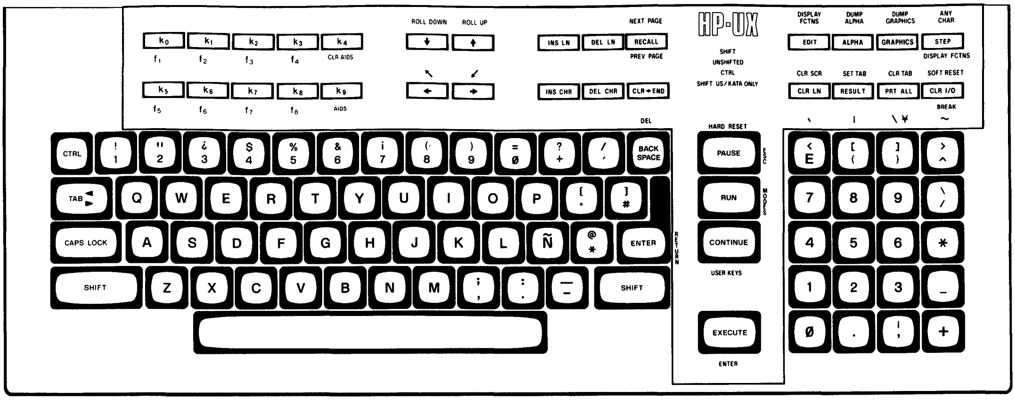
[8] The AZERTY characters can be obtained only through a software configuration of the keyboard. Physical AZERTY keyboards or hardware are not available on Series 200 computers except on the Models 217 and 237 which use the HP 46020A keyboard described in the article, "The Series 300 ITE as System Console" found in the manual, *HP-UX Concepts and Tutorials Vol. 7*.



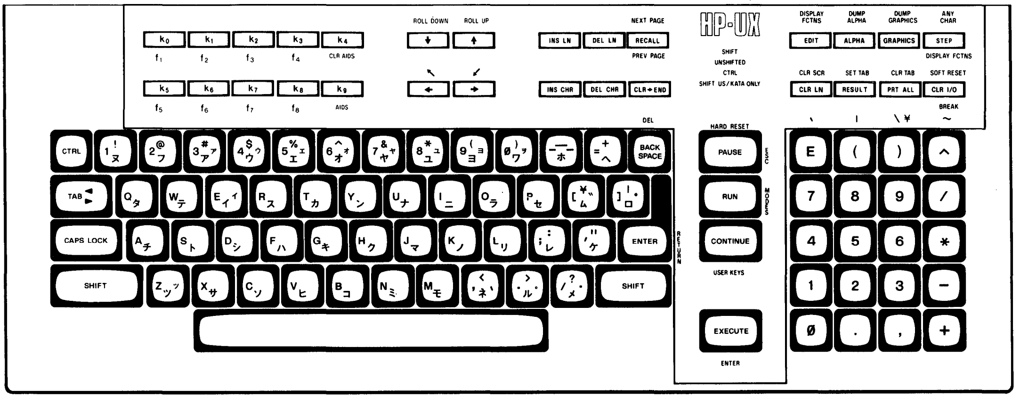
DEUTSCH (German)



ESPANOL M (Spanish with mutes)



ESPANOL (Spanish)



KATAKANA (Japanese)

Index

a

Absolute Addressing 28, 30

Absolute Sensing 26

Accessing Color 32

AIDS (k9) 15, 19

ALPHA 22

ANY CHAR (shift-STEP) 22

ASCII 8 Bit field 43, 44, 52

Attribute field 19

AUTO LF 14, 16

Auto line feed mode 16

AZERTY 40

b

BACK SPACE 7

BaudRate 4

BREAK (CLR I/O) 23

c

CAPS LOCK 7

CapsLock field 39

Character Entry Group 6, 7

CLR AIDS (k4) 17, 21

CLR→END 14

CLR LN 14, 22

CLR SCR (shift-CLR LN) 22

CLR TAB 8, 16

CLR TABS 16

CLR TAB (shift-PRT ALL) 23

Color pair 32, 33

Column Addressing 25

Config 17, 45

Configuration menus 37

CTRL 7

Cursor Control 9

Cursor Positioning ... 27
Cursor Relative Addressing ... 29, 30
Cursor Sensing ... 26

d

DEFAULT ... 45
DEL CHR ... 14
DEL LN ... 14
DEL (shift-BACK SPACE) ... 14
Display Control Group ... 6, 8
Display Enhancement ... 31
DISPLAY FCTNS (shift-EDIT) ... 22
DISPLAY FCTNS (STEP) ... 22
Display function mode ... 16
DSPY FN ... 14, 45
DUMP ALPHA (shift-ALPHA) ... 22
DUMP GRAPHICS (shift-GRAPHICS) ... 22

e

EDIT ... 22
Edit Group ... 6, 13
Enq/Ack ... 4
ENTER (EXECUTE) ... 17
Escape Sequences ... 46, 47, 48
ESC (PAUSE) ... 22
Exrc ... 13
Extended escape sequences ... 24, 46, 49, 50, 51

f

FlexDisc ... 17
Function Key Group ... 6, 14
Function Key Labels Programmatically, Controlling the ... 21

g

GRAPHICS ... 22

h

HARD RESET (shift-PAUSE) 23
HOME 13
HP:
  35721A/B/C 2
  35731A/B 2
  46020A 7
  82912A 2
  82913A 2
  98203A 52
  98203B 2
  98204A 2, 3
  98204B 2, 3
  98287A 3
  98626A 3, 4, 5
  98628A 3, 4
  98642A 3, 4
  98644A 3, 4
  98700H 3
  98781A 2
HSL (Hue-Saturation-Luminosity) 33, 36

i

InhEolWrp(C) field 40
INS CHR 14
INS LN 14
Internal Terminal Emulator 2, 24, 37
ITE 1, 2
ITE Escape Sequences 46

k

KATAKANA 41, 42, 43
Keyboard Diagrams:
  French (AZERTY layout) 54
  French (AZERTY layout with mutes) 53
  French (QWERTY layout) 54
  French (QWERTY layout with mutes) 53
  German 55
  Katakana (JASCII) 56

Spanish ... 56
Spanish (with mutes) ... 55
Swedish/Finnish ... 52

Keyboard Groups:
Character Entry ... 6, 7
Display Control ... 6, 8
Edit ... 6, 13
Function Key ... 6, 14
Numeric ... 6, 8
System Control ... 6, 22

Keyboard Requirements ... 2

Keyboards:
American (USASCII) ... 40
French (AZERTY layout) ... 40, 54
French (AZERTY layout with mutes) ... 40, 53
French (QWERTY layout) ... 40, 54
French (QWERTY layout with mutes) ... 40, 53
German ... 40, 55
Katakana (JASCII) ... 40, 56
Spanish ... 40, 56
Spanish (with mutes) ... 40, 55
Swedish/Finnish ... 40, 52

Key definition field ... 19

Knob ... 11

l

Label field ... 19

Language field ... 40, 52

Line Control Switch Pack ... 4, 5

LocalEcho field ... 38

Local mode ... 16

m

Modes ... 17

MODES (RUN) ... 15, 19, 21

Monitor Requirements ... 2

n

NEXT 44
Next page 12
NEXT PAGE (shift-[RECALL]) 11
Numeric Group 6, 8
Numeric key pad 8
Numeric key pad and US/KATA keys 15

o

Overlay color-code 6

p

Pace (Handshake) 4
Parity/DataBits 4
Pen# 32
Pen Definitions 33
Pen, Selecting a 33
PREVIOUS 45
Previous page 12
PREV PAGE ([RECALL]) 11
[PRT ALL] 22

q

QWERTY 40

r

Raster Control 32
Relative Addressing 27, 30
Relative Sensing 26
REMOTE 13, 14
Remote mode 16
[RESULT] 22
ReturnDef field 38
RETURN ([ENTER]) 7
RGB (Red-Green-Blue) 33, 35, 36
ROLL UP (shift-[↑]) 11
ROLL DOWN (shift-[↓]) 11
Roman 8 .... 41, 42
Roman 8 Character Set 42
Row Addressing 24

s

SAVE CFG 44
Screen Relative Addressing 27, 30
Selecting a Pen 33
SET TAB 16
SET TAB (shift-[RESULT]) 8, 22
[SHIFT] 7
SOFT RESET (shift-[CLR I/O]) 23
Special (device) file 3
Stop Bits 4
System Console 3
System Control Group 6

t

[TAB] 7
Tab/mrgn 16
Terminal Configuration Menu 37, 38
Terminal Configuration Softkeys:
  Config 45
  DEFAULT 45
  DSPY FN 45
  NEXT 44
  PREVIOUS 45
  SAVE CFG 44
Transmit functions mode 9

u

User Key Field:
  Attribute 19
  Key Definition 19
  Label 19
USER KEYS ([CONTINUE]) 15, 17, 18, 19, 21
User Keys Locally, Defining 18
User keys mode 17
User Keys Programmatically, Defining 20
US/KATAKANA 41, 43
US/KATAKANA Character Set 43
US/KATA keys, Numeric key pad and 15

x

XmitFnctn(A) field 10, 39

Table of Contents

The Series 300 ITE as System Console

Overview of Article 1
Description of the ITE 2
Keyboard Requirements 2
Description of a System Console 3
Additional Considerations 4
Using the Keyboard 5
Character Entry Group 6
Numeric Group 9
Display Control Group 9
Edit Group 12
Function Key Group 13
System Control Group 20
The Display 22
Memory Addressing Scheme 22
Cursor Sensing 23
Cursor Positioning 24
Screen Relative Addressing 24
Absolute Addressing 25
Cursor Relative Addressing 27
Combining Absolute and Relative Addressing 28
Display Enhancements 29
Raster Control 30
Accessing Color 30
Configuring the ITE 35
Configuration Function Keys 35
Terminal Configuration Menu 36
ITE Escape Sequences 44
Sequence Types 44
Keyboard Diagrams for Other Languages 50
HP 46020A Keyboards 51

The Series 300 ITE as System Console

Overview of Article

This article provides you with a description of the ITE as system console for Series 300 computers. If you have a Series 200 computer and you are using it as a terminal connected to an HP-UX system, you need to read the "Terminal Emulator Manual" for the particular terminal your computer is to emulate.

The topics this article covers are as follows:

- Description of the ITE
- Using the Keyboard
- The Display
- ITE Configuration
- ITE Escape Sequences

Description of the ITE

The Internal Terminal Emulator (ITE) consist of "device driver" code contained in the HP-UX kernel and associated with the keyboard and monitor on a Series 300 computer. Your particular keyboard and monitor combination are considered to be an ITE if they meet one requirement given in the section, "Keyboard Requirements" and one requirement given in the section, "Monitor Requirements".

Keyboard Requirements

The keyboard you are using is considered to be an ITE keyboard if it is an HP 46020A (HP-HIL Keyboard) connected to the computer mainframe's HP-HIL interface port.

Monitor Requirements

The monitor you are using is considered to be an ITE monitor if it is an:

- HP 35721A/B/C or HP 35731A/B monitor connected to a built-in Model 310 video board,
- HP 98782 and 98700H monitor connected to an HP 98287A video board which has been configured for "internal" control space,
- HP 35731A/B monitor connected to an HP 98542 or HP 98546 video board,
- HP 35741A/B monitor connected to an HP 98543 or HP 98546 video board,
- HP 98781A monitor connected to an HP 98544 video board,
- HP 98782A monitor connected to an HP 98545 video board.

Description of a System Console

The **system console** is a keyboard and monitor (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 boot ROM and HP-UX operating system assign the system console function according to a prioritized search sequence. HP-UX's search for a system console terminates as soon as one of the following conditions is met:

- A built-in serial interface, HP 98626A, HP 98628A[1], HP 98642A, or HP 98644A RS-232C serial interface is present with the "remote bit"[2] set. If more than one serial interface card with its "remote bit" set is present, the one with the lowest select code is used. In the case of the HP 98642A (4-channel multiplexer), port 1 is used.

- An "internal" bit-mapped display is present. This is true if a Model 310 built-in video output, HP 98542A, HP 98543A, HP 98544A or HP 98545A board is present. It is also true if an HP 98700H display station with its display interface card (HP 98287A) configured for "internal" control space is present. Note that if the HP 98700H display station's display interface card (HP 98287A) is configured for "external" control, it is never chosen ("external" bit-mapped displays have a specific select code address and "internal" bit-mapped displays **do not**).

- An HP 98546A compatibility video interface is present.

- A built-in serial interface, HP 98626A, HP 98628A, HP 98642A, or HP 98644A RS-232C serial interface is present without the "remote bit" set. If more than one is present, the one with the lowest select code is used. In the case of the HP 98642A (4-channel multiplexer), port 1 is used. The boot ROM **does not** recognize the serial interface card as console when this condition is met; however, HP-UX does.

If none of the above conditions are met, no system console exists. While boot ROM tolerates this, HP-UX will not.

[1] The HP 98628A Datacomm Interface Card with its "remote bit" set is not supported as a remote console by the Revision A boot ROM; however, it is supported as the system console by the HP-UX operating system. Therefore, when an HP 98628A card is used and has its "remote bit" set, the boot ROM sends messages to the next console found, but HP-UX sends its messages to the ITE (terminal) associated with the HP 98628A card.

[2] 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, HP 98642A and HP 98646A interface cards the remote bit is set by setting a switch on the board as described in each board's installation manual.

Additional Considerations

This section contains information on bit settings for the HP 98626A RS-232C Serial Interface Card's Line Control Switch Pack.

The boot ROM requires that the Line Control Switch Pack settings on the HP 98626A RS-232C Serial Interface card be set to the same setting as your remote terminal. HP-UX resets these values to system defaults on log in. These values are as follows:

Stop Bits	should be set to 1.
BaudRate	should be set to 9600 bps.
Parity/DataBits	should be set to 0's/7.
Enq/Ack	should be set to NO.
Pace (Handshake)	should be set to XON/XOFF.

To make the above settings on your HP 98626A Serial Interface card, set your cards Line Control Switch Pack (U-2) switches as follows:

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

Additional settings for Handshake Type bits 6 and 7 of the Line Control Switch Pack are:

Bit 6	Bit 7	Handshake Type
0	0	ENQ/ACK
1	0	XON/XOFF
0	1	NO HANDSHAKE
1	1	NO HANDSHAKE

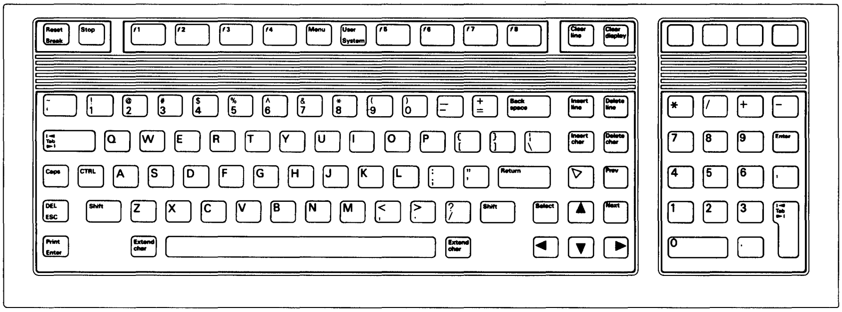
Note that the other switch settings for the Line Control Switch Pack on the HP 98626A card are defined in the installation manual supplied with that card.

Using the Keyboard

This section of the article covers the Series 300 ITE System Console keyboard. This keyboard is divided into five functional groups:

- Character Entry Group,
- Numeric Group,
- Display Control Group,
- Edit Group,
- Function Key Group,
- System Control Group.

The remainder of this section covers these key groups using the HP 46020A keyboard.



HP 46020A Keyboard

Character Entry Group

The character entry keys are arranged like a typewriter, but have some added features. This is the largest group on the keyboard, comprising 60 keys capable of generating 96 characters; including upper and lower-case alphabetic characters, numeric characters, space, punctuation, and some commercial symbols. Several data control keys are also available. The character indicated on the bottom-half of the keycap is the standard (or base) character generated by the key. The character indicated on the top-half of the keycap is the shifted character.

Shift
gives you uppercase letters when you are typing in lowercase (caps lock off). When you are typing in uppercase (caps lock on) , this key will give you lowercase letters.

Caps
sets the unshifted keyboard to either uppercase (the power-on default) or lowercase (normal typewriter operation) letters.

Return
sends the ReturnDef sequence to the computer (the default is to send ␍). 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.

DEL
ESC
generates the Escape control character. This key pressed on its own equals ESC. The escape character (followed by one or more alphanumeric characters) initiates terminal controls similar to those normally available to the computer; see the section of this article, "ITE Escape Sequences". For example, to home up the cursor (left margin of first line), press ESC and the capital H.

DEL is used to generate the Delete character. This key is pressed when Shift is pressed to obtain the Delete character. In Local Mode, this character is displayed as an inverse video square. In Remote Mode, the displayed character and its use depends on the application program.

Print
is not implemented on ITEs.

Enter
The Enter key is not implemented on ITEs.

Select
is not implemented on ITE's.

Extend char — pressed in conjunction with another key accesses the Roman 8 character set. The complete Roman 8 character set can be found in the section, "Configuring the ITE". Your computer may not be able to display all of these characters; when the key for an undisplayable character is pressed, an `hp` or regular alphabetic character will appear on the screen in its place.

To use the Extend char key for displaying Roman 8 and KATAKANA characters, you must configure the HP-UX environment on your ITE so that:

- parity generation and detection are disabled,
- the character size is 8 bits,
- input characters are not stripped to seven bits.

The terminal configuration menu must also have its **ASCII 8 Bit** field set to YES and the **Language** field set to KATAKANA for KATAKANA characters or any other language except KATAKANA for Roman 8 characters. If your ITE's monitor is connected to an HP 98546A video compatibility interface card, then you should make sure that the CHAR SELECT switch on that card is set to 1 for Roman 8 characters and 0 for KATAKANA characters.

The following step-by-step procedure explains how to set up your ITE environment to deal with 8-bit characters:

1. Execute the following *stty(1)* command to prevent input characters from being stripped to seven bits, to use 8-bit characters, and to disable parity checking.

   ```
   stty -parenb cs8 -istrip Return
   ```

2. Enter the Terminal Configuration menu by pressing:

   a. System key,

   b. then function key f8 (**config**),

   c. then function key f5 (**terminal**).

3. The cursor should be in the **Language** field and any language except KATAKANA should be displayed. If the language displayed is KATAKANA press the function key f2 (**NEXT CHOICE**) until the language you need appears in that field.

4. Tab to the **ASCII 8 Bit** field and press function key [f2] (**NEXT CHOICE**) until YES appears in that field. Next press function key [f1] (SAVE CONFIG).
5. Return to the system mode by pressing the [System] key.

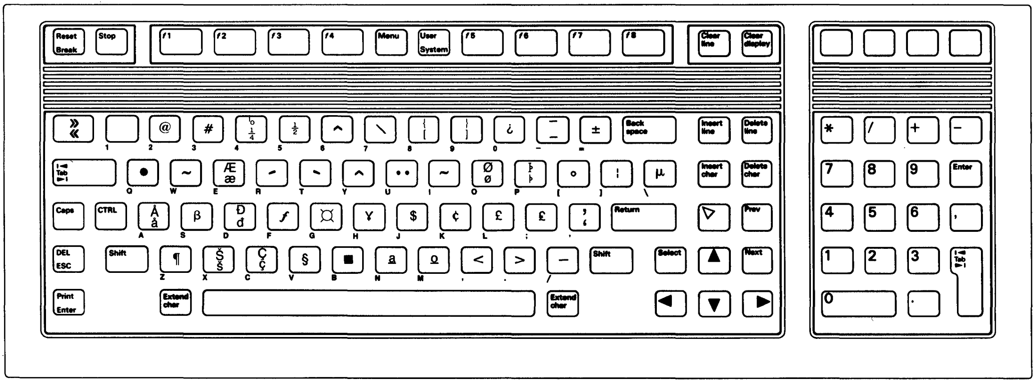
Your terminal or computer is now configured for use with the [Extend char] key. To use this key, simply hold it down and press the key associated with the Roman 8 character you wish to type. Note that this same key works with the KATAKANA characters if the **Language** field of your terminal configuration menu is set to KATAKANA instead of USASCII. To enable the KATAKANA character set press the [Extend char] key to the right of the space bar once. With the KATAKANA character set enabled, any key you press will produce KATAKANA characters. To disable the KATAKANA character set, press the [Extend char] to the left of the space bar. Pressing this [Extend char] key enables the USASCII character set and any key you press will return a USASCII character.

Refer to the figure "Diacritic Characters for the HP46020A", and type the following character using the [Extend char] key (your terminal configuration menu's **Language** field should be configured for USASCII characters):

æ

To do this hold down [Extend char] and press the unshifted [E] key.

The [Extend char] key can also be use to access the diacritical marks shown in the diagram below. The terminal configuration menu should still have the **Language** field set to any non-KATAKANA language and the **ASCII 8 Bit** field set to YES. You should also be in the local mode. Note that even though you use two keystrokes to produce diacritical combinations only one byte (character) is sent to the computer.



Diacritic Characters for the HP 46020A

Some accented characters do not appear on a keycap but are formed by combining two characters; normally a diacritic mark plus an alphabetic character (e.g., a circumflex plus a vowel). The ITE handles these characters as follows:

1. A character is typed that corresponds to the first part of a diacritical combination (for the configured keyboard type). The cursor will not move and no character will be displayed.
2. The next character is typed:
   a. If it forms a valid character combination (one giving a valid diacritical combination) the combination is displayed and the cursor moves to the next character position.
   b. If it forms an invalid character combination (other than space), the previous keystroke is ignored and the most recently typed character is displayed.
   c. If it is a space, the first character is displayed on the screen, and the cursor moves to the next character position.

Numeric 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. Again note that the [Enter] key **is not implemented** on the ITE keyboard.

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 function keys.

`SET TAB`	sets tab stops.
`CLR TAB`	deletes current tab stop.
`CLR TABS`	deletes all tab stops.

These are function keys are [f2], [f3], and [f4] respectively. To obtain these system-defined keys, press the [System] key. One of the function key labels which appears at the bottom of the display is the **tab/mrgn** function key, [f2]. Press this function key and you will receive a set of function key labels containing **SET TAB**, **CLR TAB**, and **CLR TABS**.

Cursor Control

The arrow keys allow you to move the cursor horizontally and vertically on the display. These keys are enabled when you are in the **vi** editor; however, **they are not recommended for use while in the *vi*** editor[3].

▲ 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.

While in the shell environment (not in the **vi** editor), these same keys used in conjunction with the Shift key enable you to scroll up and down through the screen display to the next or previous page. They also allow you to move the cursor to its upper or lower home position. These keys are listed as follows:

shift- ▼ scrolls the screen display downward.

shift- ▲ scrolls the screen display upward.

↖ moves the cursor to its upper home position.

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

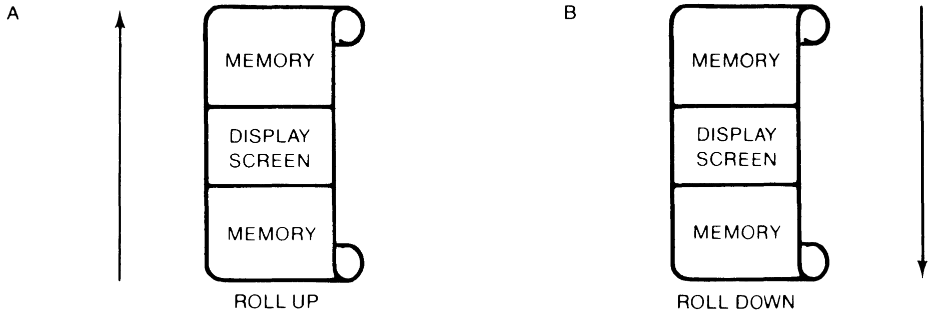
Another way to move to the next or previous page is to use the following keys:

Next moves you forward a page within your text.

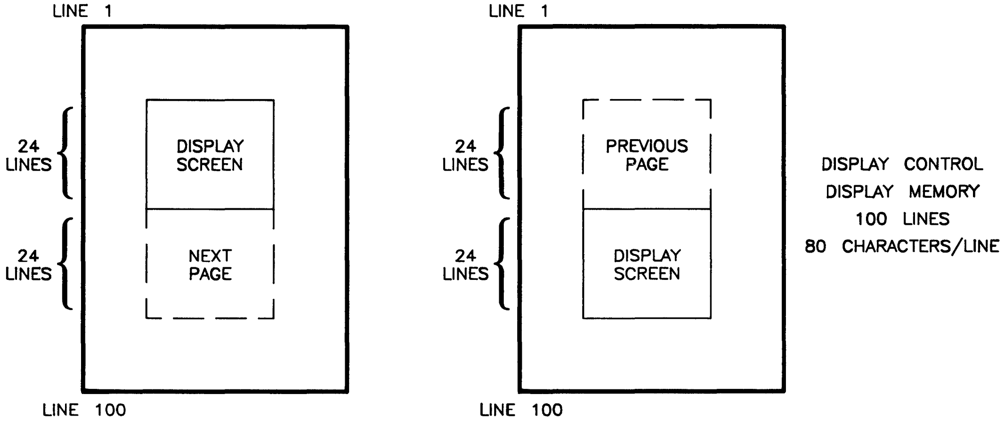
Prev moves you back a page within your text.

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-lines (47-lines for a high resolution monitor) 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.

[3] On busy systems an escape character created by these keys can be lost and misinterpreted by the system. This may result in the loss of data on a given line.



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.

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

Clear line — clears the characters in a line from the present cursor position to the end of the line.

Insert line — 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.

Delete line — 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.

Insert char — sets the insert mode, allowing you to insert characters to the left of the cursor.

Delete char — deletes the character at the current cursor location.

Clear display — clears all the display memory (and screen) from the current cursor position to the end of the memory.

Function Key Group

On the upper part of your keyboard is a set of eight function keys. 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. This section of the article covers how these keys have been assigned by the system and how you can define them. The topics covered are:

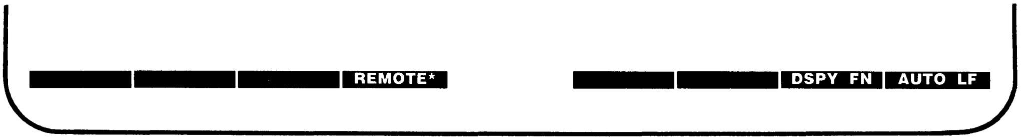
- Modes of Operation,
- System-Defined Keys,
- Defining User Keys Locally,
- Defining User Keys Programmatically,
- Controlling Function Key Labels Programmatically.

Modes of Operation

The HP 46020A keyboard has two modes of operation: **system mode** and **user mode**. The mode is set with the System/User key.

The mode of operation for HP 46020A keyboards defines the functions of the keys f1 through f8 that run across the top of the keyboard. In the system mode for Internal Terminal Emulators (ITEs), these keys are given definitions by the HP-UX system. In user mode, the keys are given definitions that you (or some application program) provide.

The HP 46020A keyboard is set to the **modes** function after the HP-UX system is booted. Note that anytime after boot up the **modes** function softkeys can be displayed by pressing the System key and then the f4 (**modes**) function key. The softkeys associated with this function are as follows:



The asterisk in the **REMOTE** softkey label shown above indicates that the function named is active. In order to use any terminal or computer keyboard with HP-UX you must have this softkey active as seen above. If the softkey is not active just press the the function key f4 once and it will become active.

Definitions for the **mode** function keys are as follows:

REMOTE when active, puts the terminal in communication with a computer. When the asterisks is absent from the label, the terminal operates in Local mode. Characters typed on an ITE in LOCAL mode are merely displayed and are not sent to any user process in HP-UX.

DSPY FN when active, does the following:

- In local mode, it displays control codes and escape sequences but does not execute them. For example, if you press the ◄ key the terminal displays $^{E}_{C}$D on the screen but does not perform the "cursor left" function.
- In remote mode, it transmits control codes and escape sequences but does not execute them locally. For example, it you press Shift and ▲ (roll up) the terminal transmits an $^{E}_{C}$S but does not perform the "roll up" function. If local echo is enabled (ON) then the $^{E}_{C}$S is also displayed on the screen.

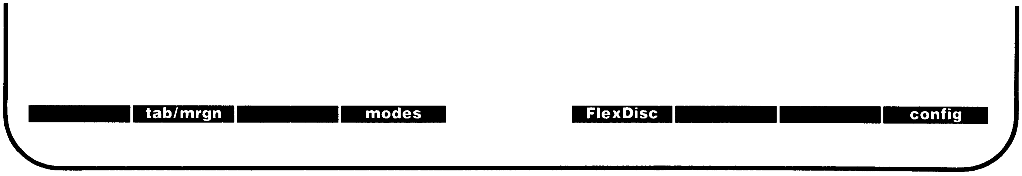
There are two exceptions to the foregoing description:

- When you press the **DSPY FN** function key, the $^{E}_{C}$Z (which disables display functions mode) or $^{E}_{C}$Y (which enables display functions mode) is executed but is not transmitted or displayed.
- A $^{C}_{R}$ (or $^{CR}{}_{LF}$ if auto line feed mode is enabled) is transmitted, and (if echoed) it is executed and displayed and the ITE also performs a line feed.

AUTO LF when active, causes a carriage-return ($^{C}_{R}$) linefeed ($^{L}_{F}$) to be generated whenever you press Return key (instead of just $^{C}_{R}$).

The definitions of the keys f1 through f8 can be removed or displayed along the bottom of the screen with the Menu key. The functions of the System, User and Menu keys are described below.

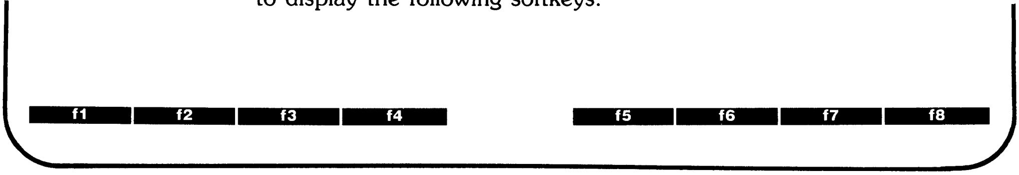
System System turns on system mode, giving keys f1 through f8 (as shown below) their HP-UX system definitions (see the "System-defined Keys" section that follows).



The softkey labels shown above are what you see on the ITE.

User (shift-System)

User turns on the user mode, giving keys f1 through f8 definitions provided by an application program (see the "User-defined Keys" section that follows). The User key is used in conjunction with the Shift key to display the following softkeys:



Menu

The actual definitions (labels) for the system-defined and user-defined keys can be removed or displayed at the bottom of the screen. Menu, CTRL–Menu, System and User (Shift–System) are used to control the display of these labels. CTRL–Menu calls up the users key display so you can define your softkeys f1 through f8. User (Shift–System) displays the user defined softkeys. System displays the System defined softkeys. Menu when toggled turns the softkey labels off and on.

System-Defined Keys

The system-defined keys are the definitions given to f1 through f8 while in **system** mode. Their actual labels are displayed on the screen when System is pressed. Refer to the tables at the back of this appendix for the labels supplied by your particular keyboard.

tab/mrgn displays a set of labels that enable control of margins, tabs, and selection of the start column for data transmission to a host computer.

modes displays labels for controlling display operations.

FlexDisc not implemented.

config displays a set of labels that allow the selection of the terminal configuration menu.

Defining User Keys Locally

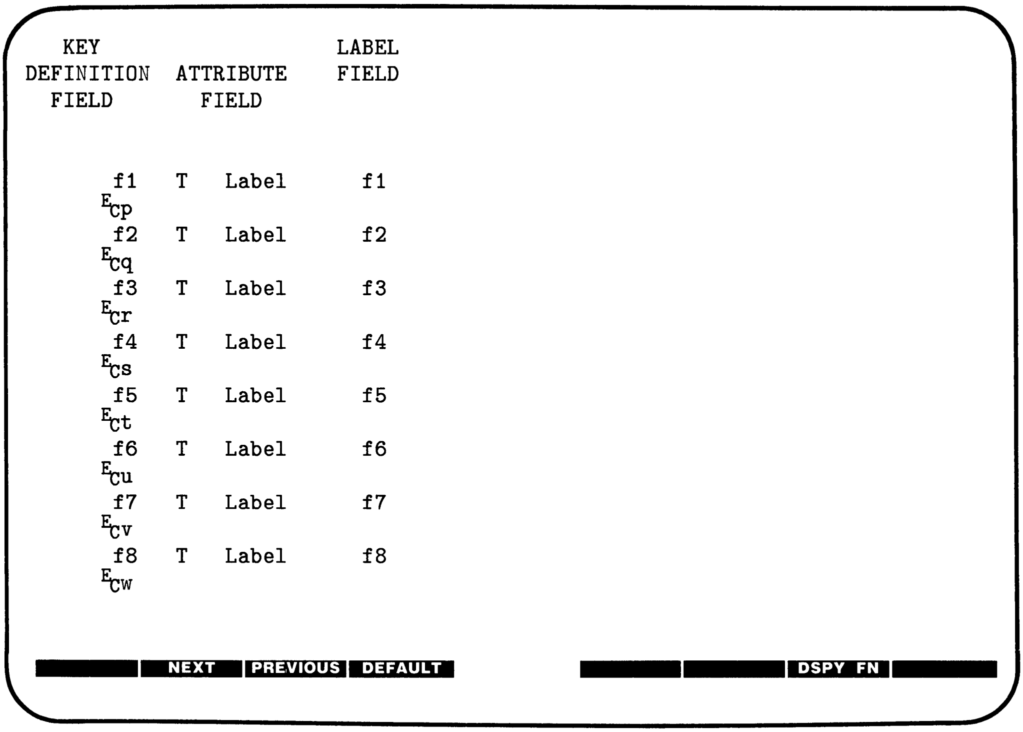
The user-defined keys are the definitions given to [f1] through [f8] while in **user** mode. Their actual labels are displayed on the screen when [User] or [Shift]-[System] is pressed.

As their name suggests, user-defined keys may be defined by you, the user, or by an application program you execute.

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 terminal operation. These escape sequences can be found in the section of this article entitled, "ITE Escape Sequences".

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), simultaneously press the [CTRL]–[Menu] keys. The following user keys menu will appear:



```
   KEY                    LABEL
DEFINITION  ATTRIBUTE     FIELD
  FIELD       FIELD


      f1    T   Label       f1
     ␛p
      f2    T   Label       f2
     ␛q
      f3    T   Label       f3
     ␛r
      f4    T   Label       f4
     ␛s
      f5    T   Label       f5
     ␛t
      f6    T   Label       f6
     ␛u
      f7    T   Label       f7
     ␛v
      f8    T   Label       f8
     ␛w
```

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 along with a trailing ␍.

N Treated in the same manner as the alphanumeric keys. If the terminal or computer is in local mode, the content of the key is executed locally. If the terminal or computer is in the remote mode and **LocalEcho** is disabled (OFF), the content of the key is transmitted to the host computer. If the terminal or computer is in the remote mode and local echo is enabled (ON), the content of the key is both transmitted to the host computer and executed locally.

Since the alphanumeric keys are disabled (ITE only) 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 eight-character field to the right of the word "Label" allows you to supply the user key's label. When the terminal is in user key's 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].

The [Return] key can be used to include carriage return ($^{C}{}_{R}$) codes (with display functions mode enabled) in key definitions. If auto line feed mode is also enabled, the [Return] key will generate a $^{CR}{}_{LF}$, otherwise it is considered a cursor movement key.

When the user keys menu is displayed on the screen you may use the [Insert char], [Delete char], [Clear line] and arrow keys for editing the contents of the label and key definition fields.

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:

␛&f <attribute><key><label length><string length><label><string>

where:

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

<key> = 1-8k : f1-f8, (1 is the default) respectively

<label length> = 0-16d (0 is the default and only the first 8 characters are displayed) <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 ␍ and ␊, and explicit escape sequence characters) in the label string must not exceed 16 (only the first 8 characters are displayed on the label), and in the definition string must not exceed 80.

Example: Assign login as the label and TOM as the definition for the f5 user key. This function key is to have attribute "N". Note that ␛&jB is the escape sequence used in this example to turn on the user key labels.

```
␛&f0a5k5d3LloginTOM␛&jB
```

After issuing the foregoing escape sequence from your program to the terminal, the f5 portion of the user keys menu is as follows:

```
f5 N LABEL login
TOM
```

If the transmit only attribute (2) is designated, the particular user key will have no effect unless the terminal is in remote.

Controlling the Function Key Labels Programmatically

From a program executing in a host computer, you can control the 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 toggling Menu).

- You can enable the mode selection keys (this is the equivalent of pressing the System key. Then pressing the softkey labeled **modes**).

- You can enable the user keys (this is the equivalent of simultaneously pressing the Shift and User keys).

The escape sequences are as follows:

$^E{}_C$`&j@` Removes all key labels from the screen.

$^E{}_C$`&jA` Enables the modes key.

$^E{}_C$`&jB` Enables the user keys.

System Control Group

These keys are located in the upper-left corner of your keyboard. They control system functions related to display operations.

Reset

Reset provides a **SOFT RESET** that does the following:

- Sounds the computer's beeper.
- Disables display functions mode (if enabled).

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

Pressing the Reset key (**SOFT RESET**), is also good for restoring the the ITE after using the screen for graphics.

The ITE can be returned to the power-on state by holding down the Shift and CTRL keys and pressing Reset. This is also called a HARD RESET.

Shift-CTRL-Reset
(HARD RESET)

has the same effect as restarting the ITE. 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

Break creates an interrupt signal (SIGINT) which is sent to all processes within your process group. 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, "Bourne Shell Programming" found in your *HP-UX Concepts and Tutorials, Vol. 4 and 5.*

Stop

Stop is the same as using CTRL-S to suspend the display and CTRL-Q to continue the display. Pressing the Stop key once suspends the display and pressing it again continues the display.

The Display

The Internal Terminal Emulator's (ITE's) display has many features of its own, video highlights (such as inverse video), raster control, cursor sensing and addressing, and color highlight control (for Series 300 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 consists of the "escape" (ASCII character 27) character followed by at least two subsequent characters.

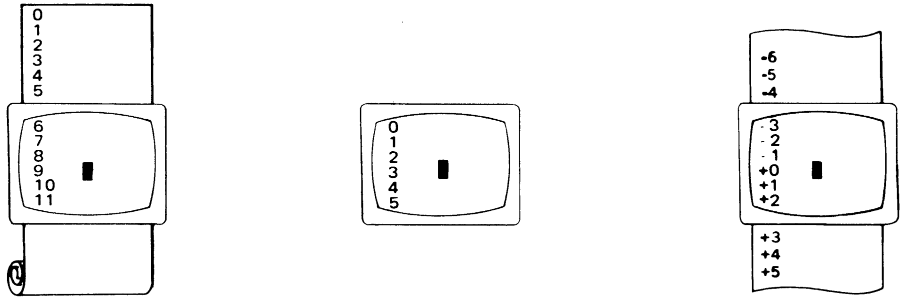
Memory Addressing Scheme

Display memory positions can be addressed using absolute or relative coordinate values. On a Series 300 Medium Resolution monitor, display memory is made up of 80 columns (0 thru 79) and up to four 24 line pages (100 lines of scrolling) with 80 characters per line. On Series 300 High Resolution monitors, display memory is made up of 128 columns (0 thru 127) and up to two 47 line pages (100 lines of scrolling) with 128 characters per line. The types of addressing available are absolute (memory relative), screen relative, and cursor relative.

Row Addressing

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

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



a.) Absolute: row 6

b.) Screen Relative: row 0

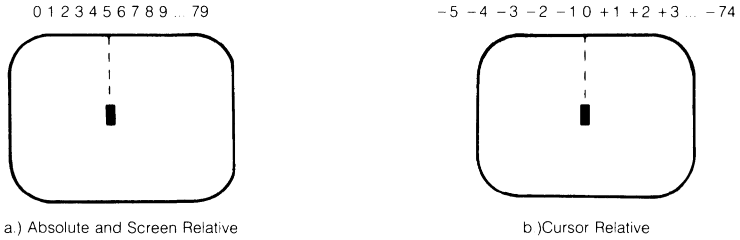
c.) Cursor Relative: row -3

Row Addressing

Column Addressing

Column addressing is accomplished in a manner similar to row addressing. There is no difference between screen 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 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.



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

Absolute Sensing

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

Relative Sensing

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

Cursor Positioning

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

Screen Relative Addressing

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

$^{E}_{C}$&a<column number> c <row number>Y

$^{E}_{C}$&a<row number> y <column number>C

$^{E}_{C}$&a<column number>C

$^{E}_{C}$&a<row number>Y

where:

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

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

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

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

Example

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

$^{\mathrm{E}}{}_{\mathrm{C}}$&a6y19C

Absolute Addressing

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

$^{\mathrm{E}}{}_{\mathrm{C}}$&a<column number> c <row number>R

$^{\mathrm{E}}{}_{\mathrm{C}}$&a<row number> r <column number>C

$^{\mathrm{E}}{}_{\mathrm{C}}$&a<column number>C

$^{\mathrm{E}}{}_{\mathrm{C}}$&a<row number>R

where:

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

<row number> is a decimal number (0-99) 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.

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:

$^{E}_{C}$&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:

${}^{E}_{C}$&a ± <column number> c ± <row number>R

${}^{E}_{C}$&a ± <row number> r ± <column number>C

${}^{E}_{C}$&a ± <column number>C

${}^{E}_{C}$&a ± <row number>R

where:

<column number> is a decimal number 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:

$^{E}_{C}$&a+15c-25R

Combining Absolute and Relative Addressing

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

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

$^{E}_{C}$&a69c+18R

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

$^{E}_{C}$&a-15c 3Y

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

$^{E}_{C}$&a9c 47R

Display Enhancements

The ITE 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.

NOTE

The Series 300 ITE **does not** provide Half-bright and Blinking enhancements.

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.

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_C&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:

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

Enhancement Character

Note that the escape sequence for "end enhancement" (^E_C&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:

$^{E}_{C}$`*dE` restores the alphanumeric display; enables writing to the alphanumeric display.

$^{E}_{C}$`*dF` disables further writing to the alphanumeric display.

Accessing Color

The monitors used for accessing color are the:

- HP 35741 connected to an HP 98543A color video board,
- HP 98782A connected to an HP 98545A color video board,
- HP 98782A plus 98700H connected to an HP 98287A color video board with option 700.

To access color on a Series 300 computer, you must understand some simple terms:

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 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.

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:

$^{E}_{C}$&v <n> <parameter>

where:

<n> is the pen number,

<parameter> is a single character that specifies the action.

To select a pre-defined pen number, the necessary <parameter> is S. Thus,

$^{E}_{C}$&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 "ITE" uses the last notation specified, or RGB notation at power-up). To select a notation type, use the $^{E}_{C}$&v escape sequence used above:

$^{E}_{C}$&v <n> <parameter>

where:

<n> is 0 (for RGB) or 1 (for HSL),

<parameter> is a single character that specifies the action.

To select a notation, the necessary <parameter> is M. Thus, the sequence

$^{E}_{C}$&v 1M

selects HSL notation. It does nothing more.

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

a specifies red (hue) used in the foreground.

b specifies green (saturation) used in the foreground.

c specifies blue (luminosity) used in the foreground.

x specifies red (hue) used in the background.

y specifies green (saturation) used in the background.

z specifies 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)

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

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

HSL Definition Algorithm

	parm. 1	parm. 2	parm. 3
COLOR SELECTED	H RANGE	S RANGE	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

	parm.1	parm. 2	parm. 3
COLOR SELECTED	RED RANGE	GREEN RANGE	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:

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

where:

<0|1> precede the `m` parameter to select either RGB or HSL notation. RGB is selected when 0 precedes the `m` parameter and HSL is selected when 1 precedes the `m` parameter.

<n> is one of the legal values selected from either the HSL or RGB table. The table selected is determined by the value preceding the `m` parameter, as explained above.

<pen#> is an integer in the range 0 thru 7 which, precedes 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. Note that the `a`, `b`, `c`, `x`, `y`, and `z` parameters are defined in the section, "Changing Pen Definitions".

Examples

```
␛&v 0m 1a 0b 0c 0x 1y 0z 5I
```

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

```
␛&v 0m 1a 1y 5I
```

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

```
␛&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.

```
␛&v 5I
```

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

NOTE

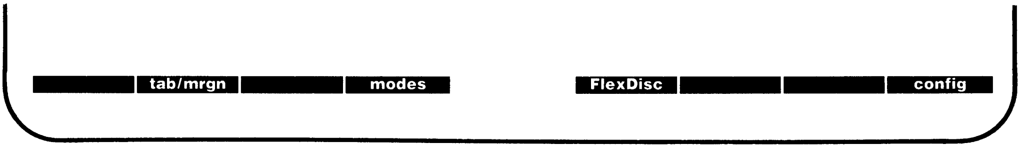
Supplying neither a foreground nor a background color when defining a color pair causes both the foreground and background to be black.

Configuring the ITE

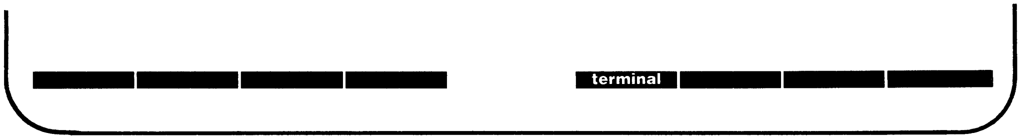
The Internal Terminal Emulator is designed so that the various ITE characteristics can be configured quickly by displaying configure "menus" on the screen and then using system function keys to change the content of these menus.

Configuration Function Keys

To gain access to the configuration menus through the keyboard, press the System key. This causes the following softkey display to appear at the bottom of the screen:



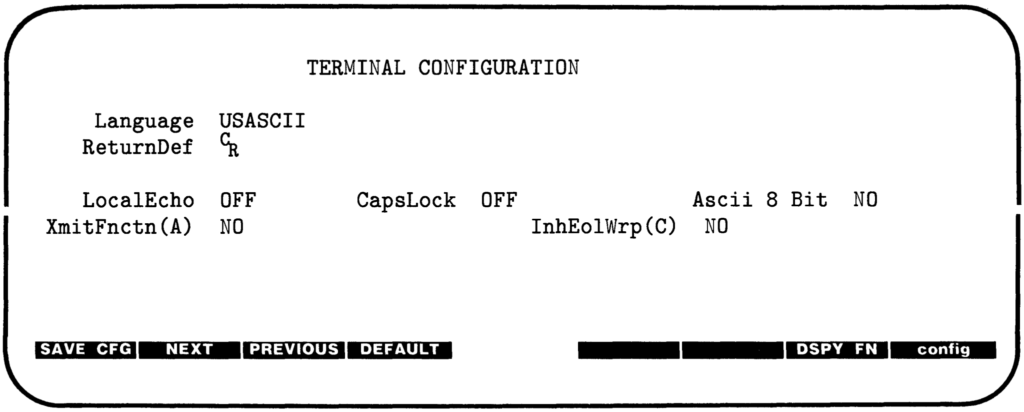
The function keys f2 (tab/mrgn), f4 (modes) and f5 (FlexDisc which is not implemented) were covered in the section "Using the Internal Terminal Emulator" along with a brief discussion on the function key f8 (config). When you press f8 (config) a new softkey display appears at the bottom of your screen.



Pressing f5 (terminal) fills the display with the **Terminal Configuration Menu** which is covered next.

Terminal Configuration Menu

After pressing f5 (terminal) your display should look this:



```
                      TERMINAL CONFIGURATION

     Language  USASCII
    ReturnDef  ␍

    LocalEcho  OFF        CapsLock  OFF              Ascii 8 Bit  NO
 XmitFnctn(A)  NO                        InhEolWrp(C)  NO
```

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 RETURN key. The default definition is an ASCII ␍. 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: ␍ space

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

LocalEcho is enabled using this escape sequence:

$^{E}{}_{C}$&k1L

When this field is enabled characters entered through the keyboard are both displayed on the screen and transmitted to the host computer. The default condition for LocalEcho is disabled or OFF, because HP-UX will echo the character's itself. The following escape sequence disables LocalEcho:

$^{E}{}_{C}$&k0L

CapsLock determines whether the ITE generates the full 128-characters ASCII set or only Teletype-compatible codes. The following escape sequence enables (turns ON) LocalEcho:

$^{E}{}_{C}$&k1C

When CapsLock is enabled 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 key for generating ˜ and ‘ is disabled.

The following escape sequence disables (turns OFF) CapsLock:

$^{E}{}_{C}$&k0C

When CapsLock is disabled the ITE generates the full 128-character ASCII set of codes. Note that the default condition for CapsLock is disabled or OFF.

`XmitFnctn(A)` determines whether escape code functions are executed at the ITE and transmitted to the host computer. The following escape sequence enters YES in the `XmitFnctn` field:

$^{E}_{C}$`&s1A`

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.

The following escape sequence enters NO in the `XmitFnctn` field:

$^{E}_{C}$`&s0A`

When NO is enter in this field 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.

The default condition for this field is NO.

`InhEolWrp(C)` designates whether or not the end-of-line wrap is inhibited. The following escape sequence enters NO in the `InhEolWrp(C)` field:

$^{E}_{C}$`&s0C`

With NO in the `InhEolWrp(C)` field, 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).

The following escape sequence enters YES in the `InhEolWrp(C)` field:

$^{E}_{C}$`&s1C`

With YES in the `InhEolWrp(C)` field, 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).

The default condition for this field is NO.

Language

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

American USASCII Keyboard "USASCII"

English (UK) Keyboard "UK"

Canadian (English) Keyboard "ENGLISH CANADIAN"

Katakana/JASCII Keyboard "KATAKANA"

Swedish Keyboard "SVENSK"

Norwegian Keyboard "NORSK"

French Keyboard "FRANCAIS"

German Keyboard "DEUTSCH"

Spanish (European) Keyboard "ESPANOL EUR."

Canadian (French) Keyboard "CANADIAN FRANCAIS"

Italian Keyboard "ITALIANA"

Dutch Keyboard "NEDERLANDS"

Finnish Keyboard "SUOMI"

Danish Keyboard "DANSK"

Swiss (German) Keyboard "SCHWEIZ-DEUTSCH"

Swiss (French) Keyboard "SUISSE ROMAND"

Swiss (German) Keyboard "SCHWEIZ-DEUTSCH*"

Swiss (French) Keyboard "SUISSE ROMAND*"

Spanish (Latin American) Keyboard "ESPANOL LAT"

Flemish Keyboard "VLAAMS"

Diagrams of the above keyboards can be found at the end of this article.

Series 300 computers support two character sets which contain the special characters associated with all of the international languages. These character sets are: Roman 8 and KATAKANA. The following charts show these character sets.

ROMAN8 CHARACTER SET
(USASCII PLUS ROMAN EXTENSION)

				b_8	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
				b_7	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
				b_6	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
				b_5	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
b_4	b_3	b_2	b_1		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	NUL	DLE	SP	0	@	P	‘	p				¯	â	Å	Á	Þ
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q			À	Ý	ê	Î	Ã	þ
0	0	1	0	2	STX	DC2	"	2	B	R	b	r			Â	ý	ô	Ø	ã	•
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s			È	°	û	Æ	Đ	µ
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t			Ê	Ç	á	å	đ	¶
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u			Ë	ç	é	í	Í	Io
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v			Î	Ñ	ó	ø	Ì	—
0	1	1	1	7	BEL	ETB	’	7	G	W	g	w			Ï	ñ	ú	æ	Ó	¼
1	0	0	0	8	BS	CAN	(	8	H	X	h	x			´	¡	à	Ä	Ò	½
1	0	0	1	9	HT	EM	)	9	I	Y	i	y			`	¿	è	ì	Õ	ª
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z			^	¤	ò	Ö	õ	º
1	0	1	1	11	VT	ESC	+	;	K	[	k	{			¨	£	ù	Ü	Š	«
1	1	0	0	12	FF	FS	,	<	L	\	l	|			˜	¥	ä	É	š	■
1	1	0	1	13	CR	GS	-	=	M	]	m	}			Ù	§	ë	ï	Ú	»
1	1	1	0	14	SO	RS	.	>	N	^	n	~			Û	ƒ	ö	ß	Ÿ	±
1	1	1	1	15	SI	US	/	?	O	_	o	DEL			₤	¢	ü	Ô	ÿ	

Roman 8 Character Set

To use the Roman 8 Character Set, read the section in this article entitled, "Character Entry Group". The explanation for the [Extend char] key in this key group covers how to access both the Roman 8 and KATAKANA character sets. Note, if your ITE monitor is connect to an HP 98546A video compatibility interface card, then you need to set the CHAR SELECT switch on this card to 1 for the Roman 8 character set and 0 for the KATAKANA character set.

The characters shown in the shaded areas on the "KATAKANA Character Set" chart **are not** provided on the medium resolution versions of the Model 310 and 320 computers.

KANA8 CHARACTER SET
(JISCII PLUS KATAKANA)

				b_8	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
				b_7	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
				b_6	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
				b_5	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
b_4	b_3	b_2	b_1		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	NUL	DLE	SP	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	ﾞ		
1	1	1	1	15	SI	US	/	?	O	_	o	DEL	▨	▨	ｯ	ｿ	ﾏ	ﾟ		▨

KATAKANA Character Set

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

The following escape sequence enters YES in the ASCII 8 Bit field:

${}^{E}_{C}$&k1I

The following escape sequence enters NO in the ASCII 8 Bit field:

${}^{E}_{C}$&k0I

Changing the Fields

To change the fields in the Terminal Configuration Menu, 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.

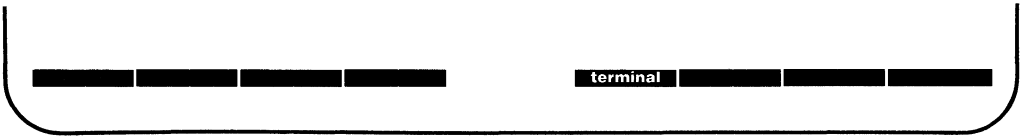
NEXT changes the setting of the field you are presently in to the next setting in that field. For example, if USASCII is displayed in the Language field, pressing **NEXT** or f2 would change the field to VLAAMS.

PREVIOUS changes the setting of the field you are presently in to the previous setting in that field. For example, with VLAAMS displayed in the Language field, pressing **PREVIOUS** or f3 would change the field back to USASCII.

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 which 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 enables 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:



To get back to the System softkey display from this state, press the System key.

ITE Escape Sequences

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

The sequences listed in this section are recognized and executed by the ITE whether they are received from the data communication link or from the keyboard (in remote mode), 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 ($^{C}_{R}$), escape ($^{E}_{C}$) 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 character in order to define the operation. Absolute and relative cursor addressing are examples of operations that require longer control sequences.

Escape Sequences for ITE Control

Escape Code	Function
$^{E}_{C}$1	Set tab
$^{E}_{C}$2	Clear tab
$^{E}_{C}$3	Clear all tabs
$^{E}_{C}$4	NOT IMPLEMENTED (Set left margin)
$^{E}_{C}$5	NOT IMPLEMENTED (Set right margin)
$^{E}_{C}$9	NOT IMPLEMENTED (Clear all margins)
$^{E}_{C}$@	NOT IMPLEMENTED (Makes the terminal program wait approximately one second.)
$^{E}_{C}$A	Cursor up
$^{E}_{C}$B	Cursor down
$^{E}_{C}$C	Cursor right
$^{E}_{C}$D	Cursor left
$^{E}_{C}$E	Hard reset (power on reset of ITE)
$^{E}_{C}$F	Cursor home down
$^{E}_{C}$G	Move cursor to the left margin
$^{E}_{C}$H	Cursor home up
$^{E}_{C}$I	Horizontal tab
$^{E}_{C}$J	Clear screen from cursor to the end of memory.
$^{E}_{C}$K	Clear line from cursor to end of line
$^{E}_{C}$L	Insert line
$^{E}_{C}$M	Delete line
$^{E}_{C}$P	Delete character
$^{E}_{C}$Q	Start insert character mode
$^{E}_{C}$R	End insert character mode
$^{E}_{C}$S	Roll up
$^{E}_{C}$T	Roll down
$^{E}_{C}$U	Next page
$^{E}_{C}$V	Previous page

Escape Sequences for ITE Control (continued)

Escape Code	Function
␛W	NOT IMPLEMENTED (Format mode on)
␛X	NOT IMPLEMENTED (Format mode off)
␛Y	Enables display functions mode
␛Z	Disables display functions mode
␛[	NOT IMPLEMENTED (Start unprotected field)
␛]	NOT IMPLEMENTED (End unprotected/transmit-only field)
␛^	NOT IMPLEMENTED (Primary terminal status request)
␛`	Sense cursor position(relative)
␛a	Sense cursor position (absolute)
␛b	NOT IMPLEMENTED (Unlock keyboard)
␛c	NOT IMPLEMENTED (Lock keyboard)
␛d	NOT IMPLEMENTED (Transmit a block of text to computer)
␛f	NOT IMPLEMENTED (Modem disconnect)
␛g	Soft reset (of ITE)
␛h	Cursor home up.
␛i	Back tab
␛j	NOT IMPLEMENTED (Begin User Key Definition mode)
␛k	NOT IMPLEMENTED (End User Keys Definition mode)
␛l	NOT IMPLEMENTED (Begin Memory Lock mode)
␛m	NOT IMPLEMENTED (End Memory Lock Mode)
␛p	Default value for softkey **f1**
␛q	Default value for softkey **f2**
␛r	Default value for softkey **f3**
␛s	Default value for softkey **f4**
␛t	Default value for softkey **f5**
␛u	Default value for softkey **f6**
␛v	Default value for softkey **f7**
␛w	Default value for softkey **f8**
␛z	NOT IMPLEMENTED (Initiate terminal self test)
␛~	NOT IMPLEMENTED (Secondary terminal status request)

Extended Escape Sequences

Escape Code Sequence	Function
␛&a <col>c <row>Y	Moves the cursor to column "col" and screen row "row" on the screen (screen relative addressing). The row or column values may be used by themselves in this escape sequence (e.g ␛&a <col> C or ␛&a <row> Y). The value you choose to leave out remains as previously set.
␛&a <col>c <row>R	Moves the cursor to column "col" and row "row" in memory (absolute addressing). The row or column values may be used by themselves in this escape sequence (e.g ␛&a <col> C or ␛&a <row> R). The value you choose to leave out remains as previously set.
␛&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. The row or column values may be used by themselves in this escape sequence (e.g ␛&a ±<col> C or ␛&a ±<row> Y). The value you choose to leave out remains as previously set.
␛&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. The row or column values may be used by themselves in this escape sequence (e.g ␛&a ±<col> C or ␛&a ±<row> R). The value you choose to leave out remains as previously set.
␛&q0L	NOT IMPLEMENTED (Unlock configuration)
␛&q1L	NOT IMPLEMENTED (Lock configuration)
␛&d	<char> Selects the display enhancement indicated by <char> to begin at the present cursor position. <char> can be @ or A thru O.

Extended Escape Sequences (continued)

Escape Code Sequence	Function
^E_C&k<x>A	AUTO LF enable: x=1; disable: x=0
^E_C&k<x>B	NOT IMPLEMENTED (BLOCK enable: x=1; disable: x=0)
^E_C&k<x>C	Caps Lock enable: x=1; disable: x=0
^E_C&k<x>I	ASCII 8 Bits enable: x=1; disable: x=0
^E_C&k<x>J	NOT IMPLEMENTED (FrameRate 50 Hz : x=1; 60 Hz : x=0)
^E_C&k<x>L	LocalEcho enable: x=1; disable: x=0
^E_C&k<x>M	NOT IMPLEMENTED (MODIFY ALL enable: x=1; disable: x=0)
^E_C&k<x>N	NOT IMPLEMENTED (SPOWLatch enable: x=1; disable: x=0)
^E_C&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
^E_C&k<x>R	REMOTE enable: x=1; disable: x=0
^E_C&s<x>A	xmitFnctn(A) enable: x=1; disable: x=0
^E_C&s<x>B	NOT IMPLEMENTED (SPOW(B) enable: x=1; disable: x=0)
^E_C&s<x>C	InhEolWrp(C) enable: x=1; disable: x=0
^E_C&s<x>D	NOT IMPLEMENTED (Line/Page(D) enable: x=1; disable: x=0)
^E_C&s<x>G	NOT IMPLEMENTED (InfHndShk(G) enable: x=1; disable: x=0)
^E_C&s<x>H	NOT IMPLEMENTED (Inh DC2(H) enable: x=1; disable: x=0)
^E_C&w 12F	Turns on the display window (top 24 rows)
^E_C&w 13F	Turns off the display window

Extended Escape Sequences (continued)

Escape Code Sequence	Function
^E_C*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.
^E_C* s <parameter>	Read device I.D. Status is parameter number 1.
^E_C&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.
^E_C&f <attribute>a <key>k	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.
^E_C&v n<parameter>	Read the section in this article "Accessing Color".

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

Keyboard Diagrams for Other Languages

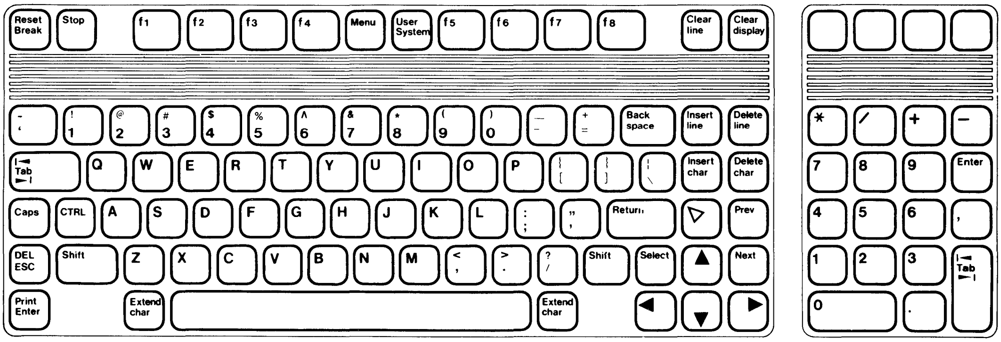
This section shows diagrams of the types of HP 46020A keyboards which are available to the Series 300 user. The keyboard option you now have can be configured to any of the languages by use of the TERMINAL CONFIGURATION menu previously mentioned in this article.

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. Also notice that, if you are running */bin/sh* or */bin/csh*, you must use *stty* to enable your shell environment to use all 8 bits of the code. For more information on your keyboard, read the appropriate manual sent with your system.

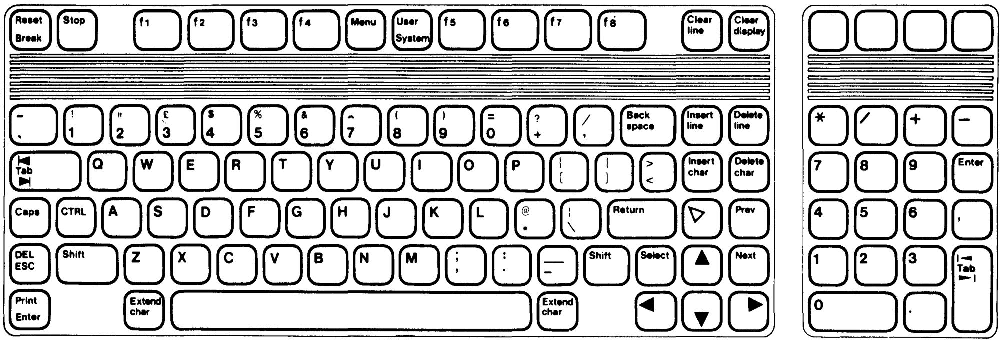
You will notice as you look at the keyboards that a majority of the character key and numeric key labels have changed. To use your keyboard effectively in any one of these languages, you will have to re-label the key caps or purchase a keyboard appropriate to the language you are using.

HP 46020A Keyboards

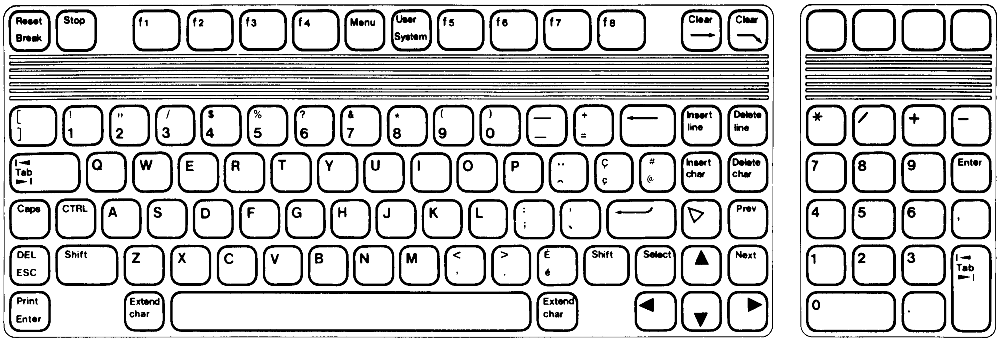
These keyboards are labeled according to the language they are to be used with.



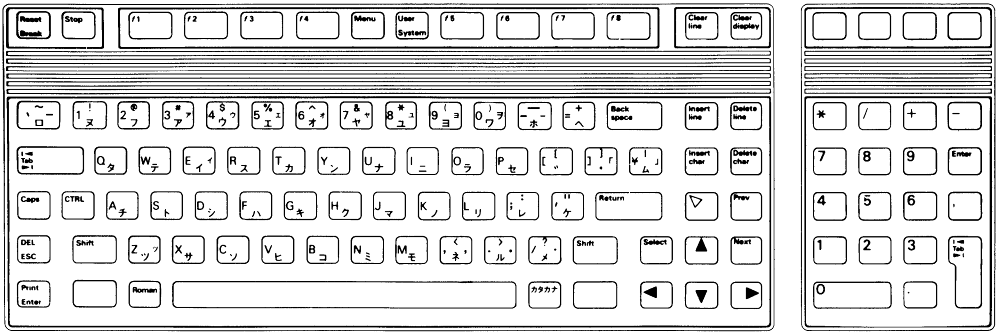
American USASCII Keyboard "USASCII"



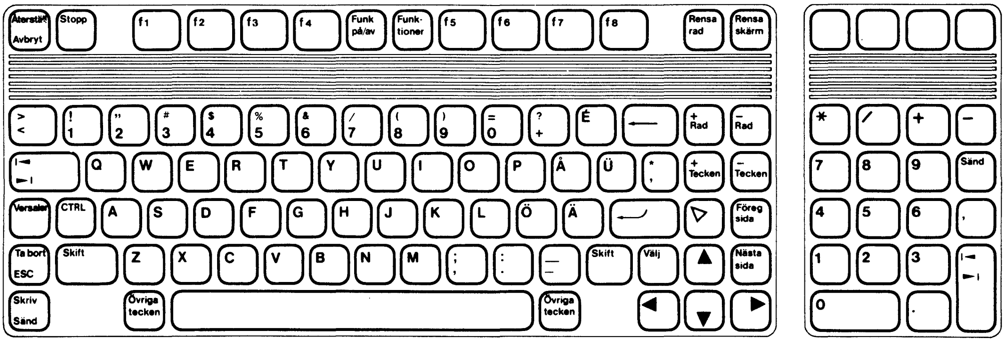
English (UK) Keyboard "UK"



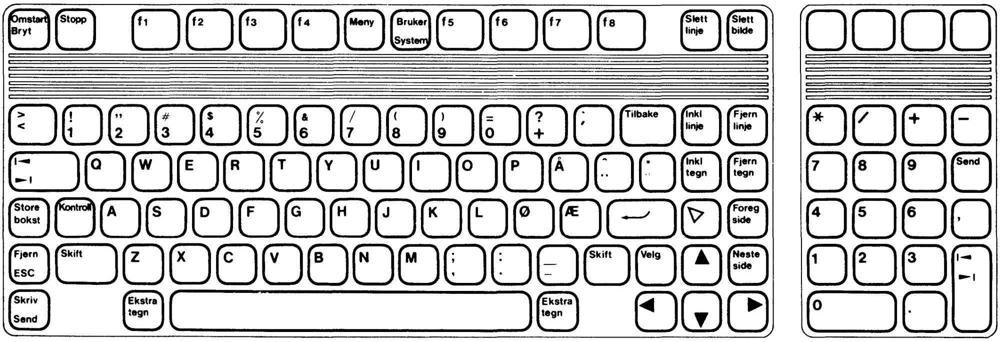
Canadian (English) Keyboard “ENGLISH CANADIAN”



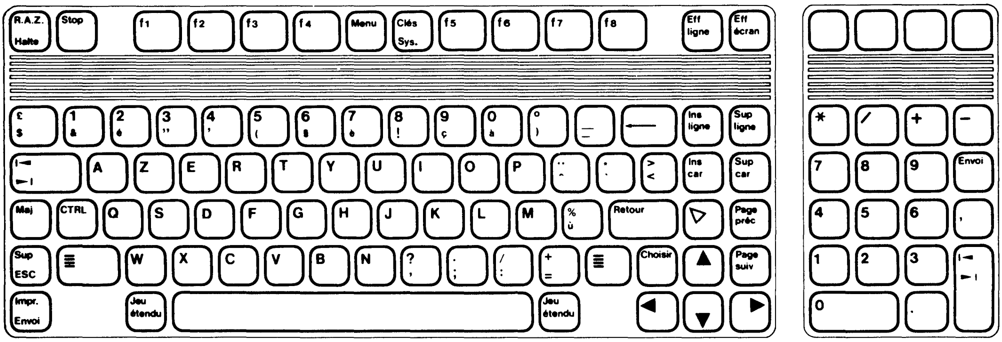
Katakana/JASCII Keyboard “KATAKANA”



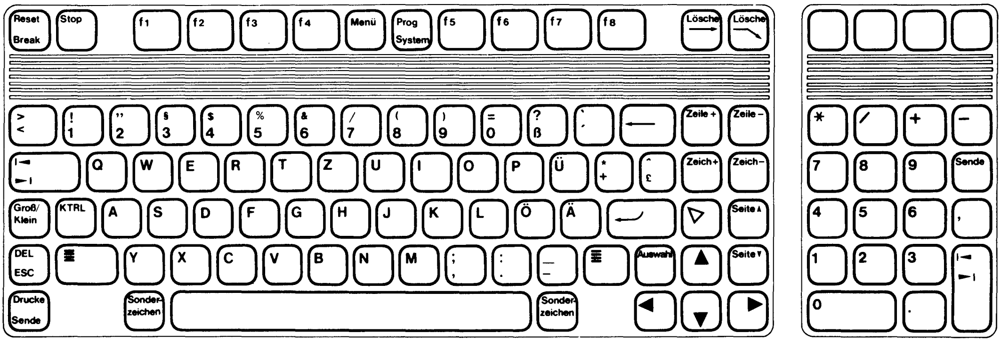
Swedish Keyboard “SVENSK”



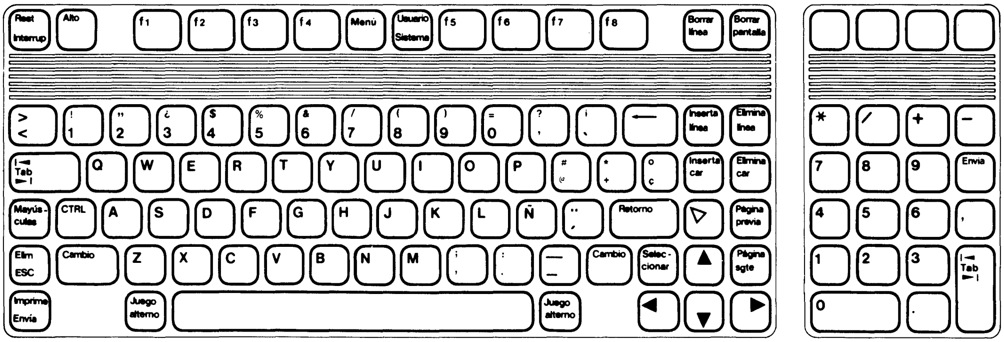
Norwegian Keyboard “NORSK”



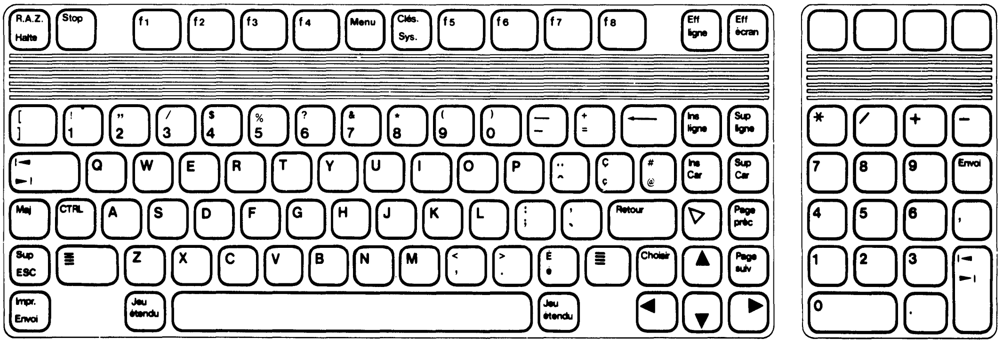
French Keyboard “FRANCAIS”



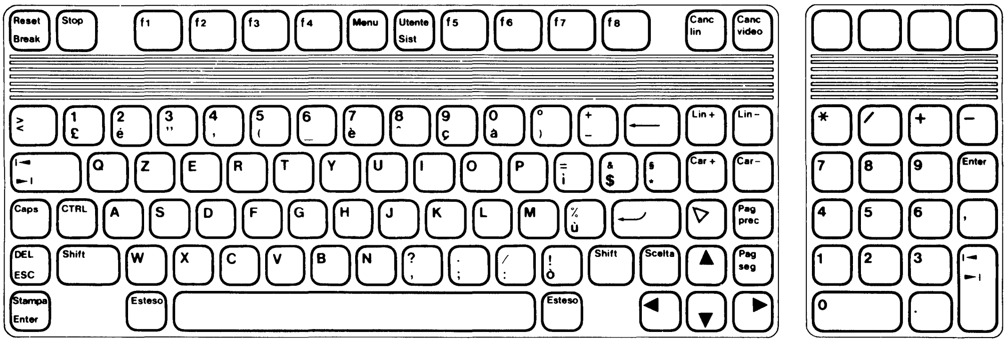
German Keyboard “DEUTSCH”



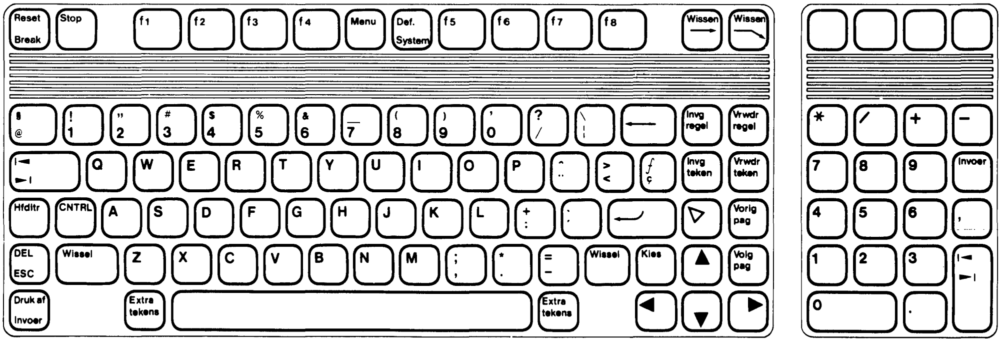
Spanish (European) Keyboard "ESPANOL EUR."



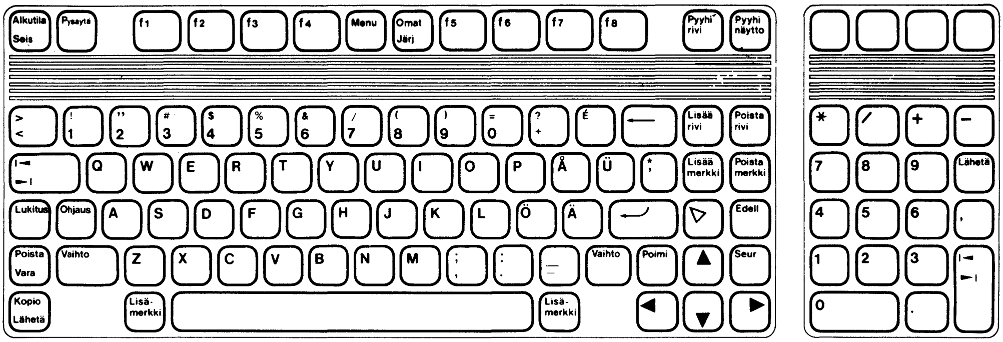
Canadian (French) Keyboard "CANADIAN FRANCAIS"



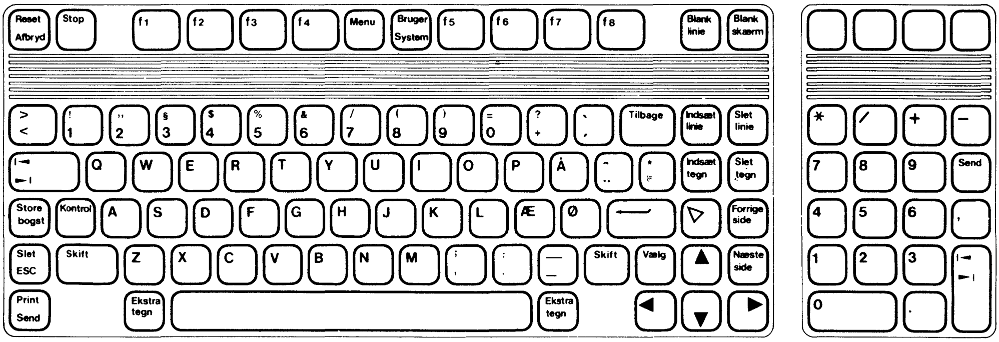
Italian Keyboard "ITALIANA"



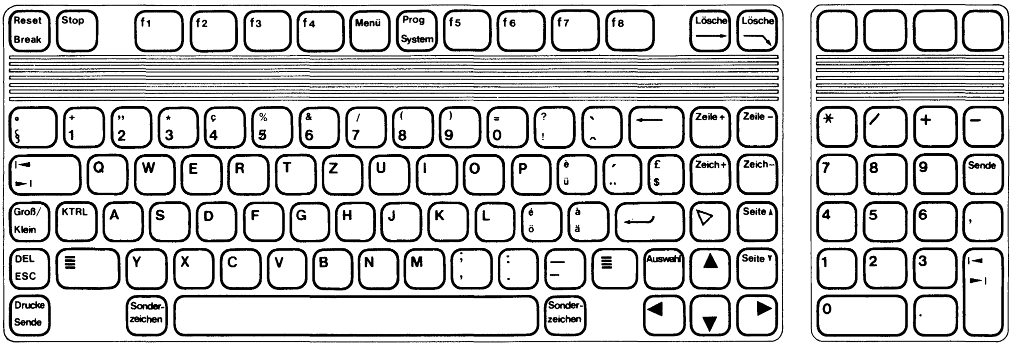
Dutch Keyboard "NEDERLANDS"



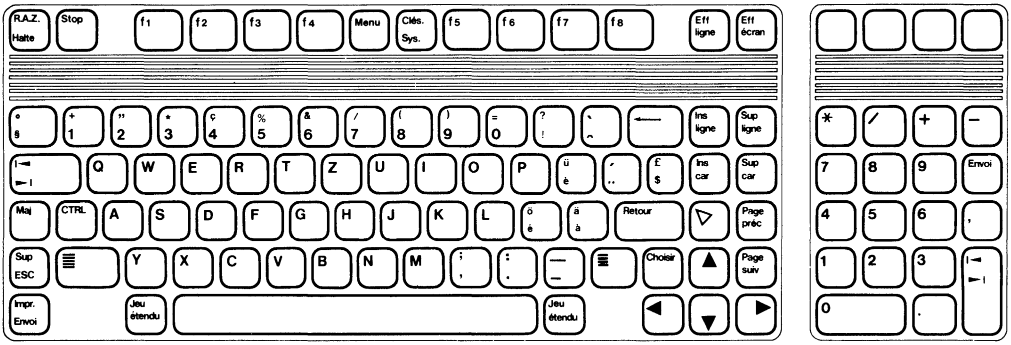
Finnish Keyboard "SUOMI"



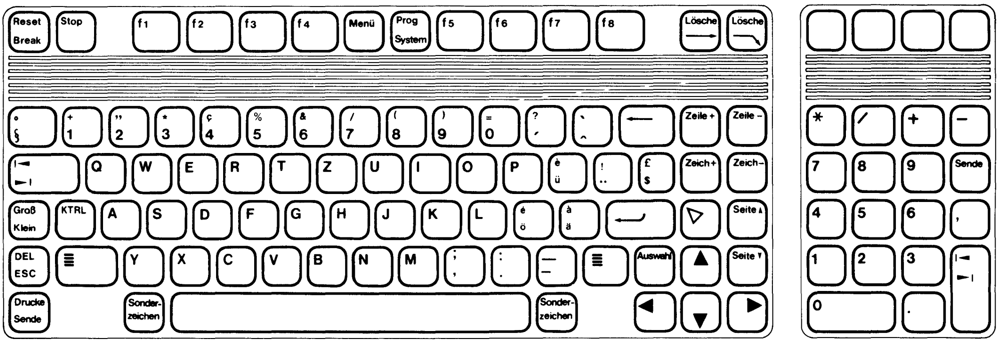
Danish Keyboard "DANSK"



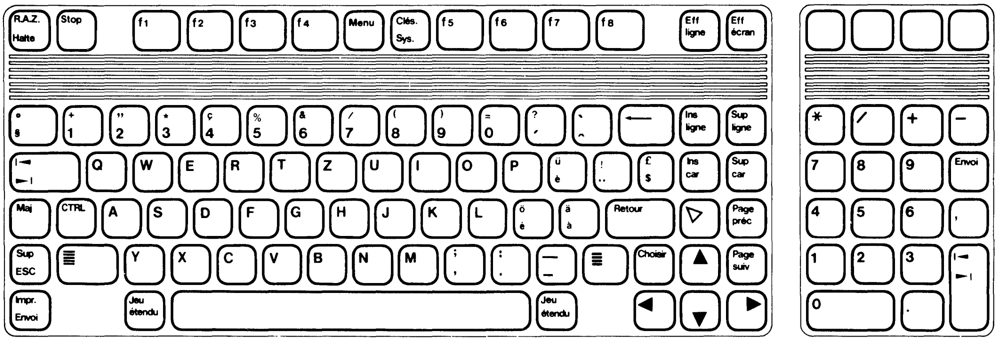
Swiss (German) Keyboard "SCHWEIZ-DEUTSCH"



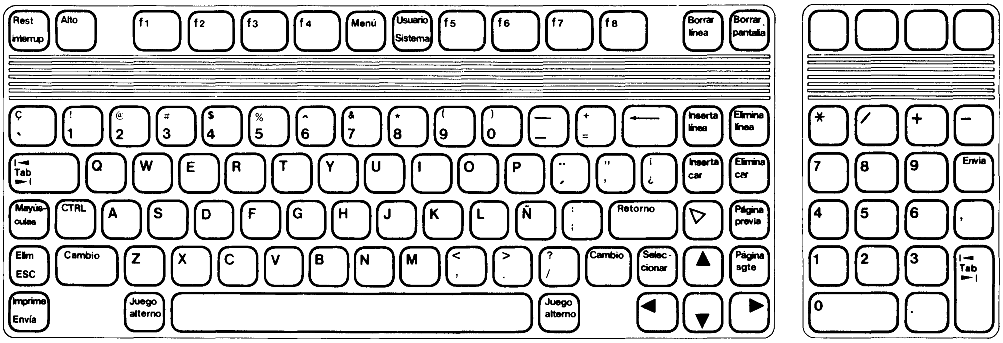
Swiss (French) Keyboard "SUISSE ROMAND"



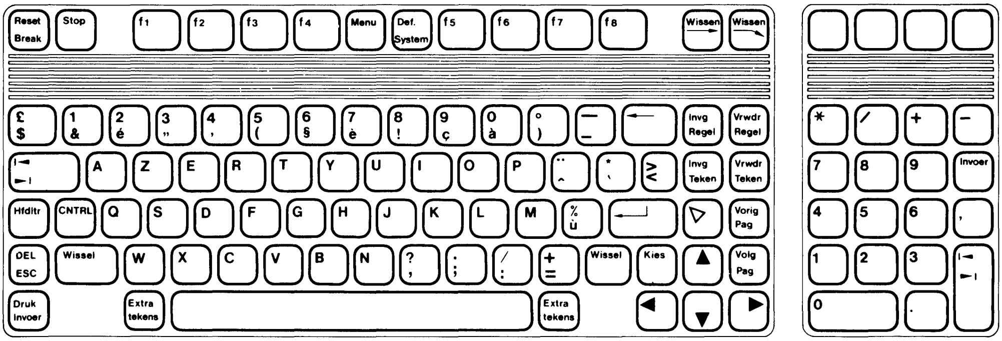
Swiss (German) Keyboard "SCHWEIZ-DEUTSCH*"



Swiss (French) Keyboard "SUISSE ROMAND*"



Spanish (Latin American) Keyboard "ESPANOL LAT"



Flemish Keyboard "VLAAMS"

Notes

Index

a

Absolute Addressing 25, 28
Absolute Sensing 23
Accessing Color 30
ASCII 8 Bit field 7, 8, 43, 51
Attribute Field 17
AUTO LF 14

b

Back space 6
BaudRate 4
Break 21

c

Caps 6
CapsLock field 37
Character Entry Group 5, 6
Clear display 12
Clear line 12, 18
CLR TAB 9
CLR TABS 9
Color pair 30, 31
Column Addressing 23
Config 15, 44
CTRL 6
Cursor Control 10
Cursor Positioning 24
Cursor Relative Addressing 27
Cursor Sensing 23

d

DEFAULT 43
Defining User Keys 19
DEL 6
Delete char 12, 18

Delete line 12
Diacritical Combinations 8, 9
Diacritic Characters 8
Display Control Group 5, 9
Display Enhancement 29
DSPY FN 14, 43

e

Edit Group 5, 12
Enq/Ack 4
Enter 6, 9
ESC 6
Escape Sequences 45, 46, 47
Exrc 12
Extend char 7, 8
Extended escape sequences 22, 45, 48, 49, 50

f

FlexDisc 15
Function Key Group 5, 13
Function Key Labels 20

h

HARD RESET 21
HOME 12
HP:
- 35721A/B/C 2
- 35731A/B 2
- 35741A 30
- 35741A/B 2
- 46020A 5, 8, 13, 51
- 98542A 2, 3
- 98543A 2, 3, 30
- 98544A 2, 3
- 98545A 2, 3, 30
- 98546A 2, 3
- 98626A 3, 4
- 98628A 3
- 98642A 3
- 98644A 3

98700H 2, 30
98781A 2
98782A 2, 30
HSL (Hue-Saturation-Luminosity) 31, 34

i

InhEolWrp(C) field 38
Insert char 12, 18
Insert line 12
Internal Terminal Emulator 2, 22, 35, 45
ITE 1, 2, 45
ITE Escape Sequences 45

k

KATAKANA 7, 39, 41
KATAKANA Character Set 42
Keyboard Diagrams:
American (USASCII) 52
Canadian (English) 53
Canadian (French) 56
Danish 58
Dutch 57
English (UK) 52
Finnish 58
Flemish 61
French 55
German 55
Italian 57
Katakana (JASCII) 53
Norwegian 54
Spanish (European) 56
Spanish (Latin American) 61
Swedish 54
Swiss (French) 59, 60
Swiss (German) 59, 60

Keyboard Groups:
- Character Entry 5, 6
- Display Control 5, 9
- Edit 5, 12
- Function Key 5, 13
- Numeric 5, 9
- System Control 5

Keyboard Requirements 2

Keyboards:
- American (USASCII) 39, 52
- Canadian (English) 39, 53
- Canadian (French) 39, 56
- Danish 39, 58
- Dutch 39, 57
- English (UK) 39, 52
- Finnish 39, 58
- Flemish 39, 61
- French 39, 55
- German 39, 55
- Italian 39, 57
- Katakana (JASCII) 39, 53
- Norwegian 39, 54
- Spanish (European) 39, 56
- Spanish (Latin American) 39, 61
- Swedish 39, 54
- Swiss (French) 39, 59, 60
- Swiss (German) 39, 59, 60

Key Definition Field 18

l

Label Field 18

Language field 7, 8, 39, 51

Line Control Switch Pack 4

LocalEcho field 18, 37

m

[Menu] 14, 15, 20

Modes 15

Monitor Requirements 2

n

NEXT . . . 43
Next . . . 10
Next page . . . 10
Numeric Group . . . 5, 9

p

Pace (Handshake) . . . 4
Parity/DataBits . . . 4
Pen# . . . 30
Pen Definitions . . . 31
Pen, Selecting a . . . 31
Prev . . . 10
PREVIOUS . . . 43
Previous page . . . 10
Print . . . 6

r

Raster Control . . . 30
Relative Addressing . . . 28
Relative Addressing, Cursor . . . 27
Relative Addressing, Screen . . . 24
Relative Sensing . . . 24
REMOTE . . . 13, 14
Reset . . . 20, 21
Return . . . 6, 18
ReturnDef field . . . 36
RGB (Red-Green-Blue) . . . 31, 33, 34
Roman 8 . . . 7, 39, 41
Roman 8 Character Set . . . 40
Row Addressing . . . 22

s

SAVE CFG . . . 43
Screen Relative Addressing . . . 24
Select . . . 6
Selecting a Pen . . . 31
SET TAB . . . 9
Shift . . . 6
SOFT RESET . . . 20

Special (device) file 3
Stop 21
Stop Bits 4
Stty 7, 51
System 7, 9, 13, 14, 15, 20
System Console 3
System Control Group 5
System-Defined Keys:
  Config 15
  FlexDisc 15
  Modes 15
  Tab/mrgn 15
System mode 13

t

Tab 6
Tab/mrgn 15
Terminal Configuration Menu 35, 36
Terminal Configuration Softkeys:
  Config 44
  DEFAULT 43
  DSPY FN 43
  NEXT 43
  PREVIOUS 43
  SAVE CFG 43

u

User 14, 15, 20
User-defined keys 16
User Key Field:
  Attribute 17
  Key Definition 18
  Label 18
User keys 16
User Keys, Defining 19
User mode 13

x

XmitFnctn(A) field 38

Table of Contents

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

The Keyboard....2
- Alphanumeric Group....2
- Numeric Pad Group....2
- Display Control Group....2
  - Setting and Clearing Margins....2
  - Setting and Clearing Tab Stops....3
  - Cursor Control....3
- Edit Group....6
  - Insert Character Mode....7
- Function Key Group....7
  - Modes....7
  - AIDS....8
  - User Keys....9
  - User-definable Keys....9

The Display....11
- Memory Addressing Scheme....11
  - Row Addressing....11
  - Column Addressing....11
- Cursor Sensing....12
  - Absolute Sensing....12
  - Relative Sensing....12
- Cursor Positioning....12
- Screen Relative Addressing....13
  - Example....13
- Absolute Addressing....13
  - Example....14
- Cursor Relative Addressing....14
  - Example....15
- Combining Absolute and Relative Addressing....15
- Display Enhancements....15
- Raster Control....16
- Accessing Color....16
  - Color Pair....16
  - Pen#....17
  - Selecting a Pen (Color Pair)....17
  - Changing Pen Definitions....17
  - Examples....19

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

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

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

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

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

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

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

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

The Keyboard

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

Alphanumeric Group

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

Numeric Pad Group

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

Display Control Group

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

Setting and Clearing Margins

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

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

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

\E4 - set the left margin at the current cursor location.

\E5 - set the right margin at the current cursor location.

\E9 - clear both margins; by default the left margin becomes 1, the right margin becomes 80.

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

Setting and Clearing Tab Stops

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

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

\E1 - set a tab stop at the current cursor position.

\E2 - clear a tab stop previously set at the current cursor position.

\E3 - clear all tab stops currently set. Note that this feature is available only from softkeys (as are the margin functions described above).

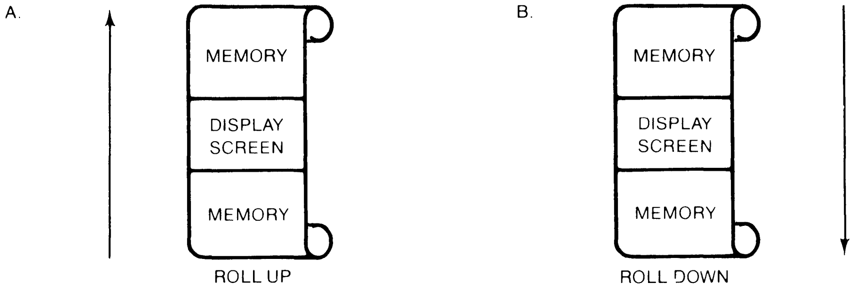
Cursor Control

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

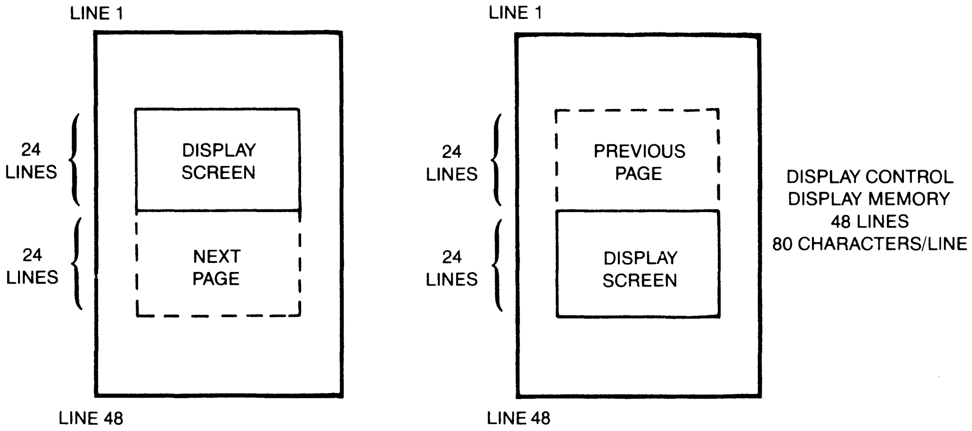
Key	Escape Sequence	Feature
↑	\EA	Move the cursor up one row in the current column position. Holding the key down causes the cursor to move continuously, row by row, until the key is released. When the cursor is in the top row of the screen, moving the cursor up actually moves the cursor to the same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

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.

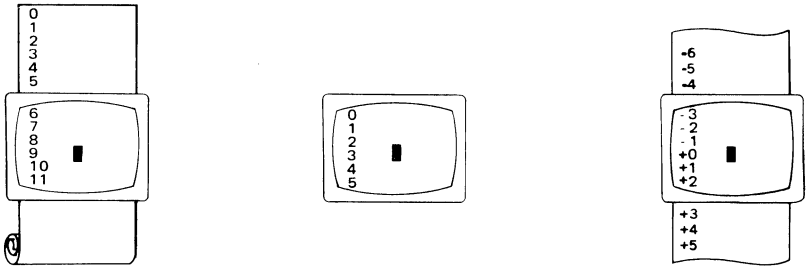
Memory Addressing Scheme

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



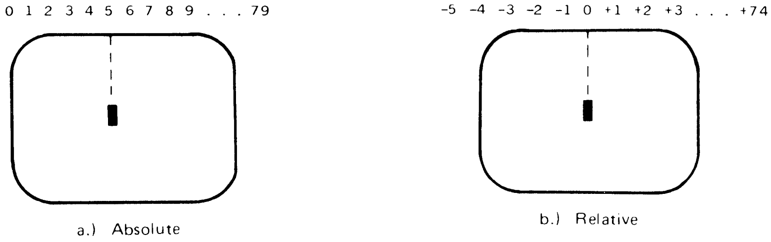
a.) Absolute: row 6   b.) Screen Relative: row 0   c.) Cursor Relative: row -3

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.



a.) Absolute

b.) 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. (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.

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

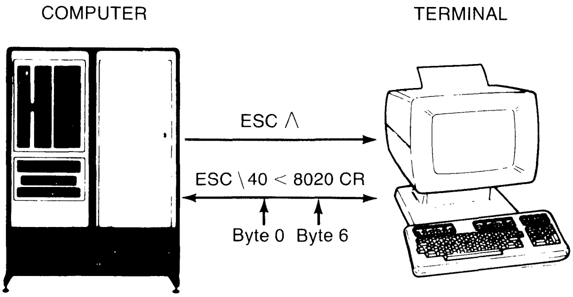
Terminal status is made up of 14 status bytes (bytes 0 through 13) containing information such as display memory size, switch settings, configuration menu settings, and terminal errors. There are two terminal status requests: primary and secondary. Each returns a set of seven status bytes.

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.

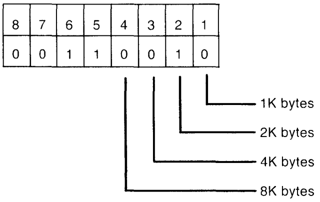


BYTE	ASCII	BINARY	STATUS	
0	4	0011 0100	4K bytes of display memory (2 pages of 24 lines)	
1	0	0011 0000	Function key transmission disabled	Configuration Straps A-H
			Cursor wraparound disabled	
2	<	0011 1100		
3	8	0011 1000	Auto line feed disabled	
			Terminal sends secondary status	
4	0	0011 0000	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

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

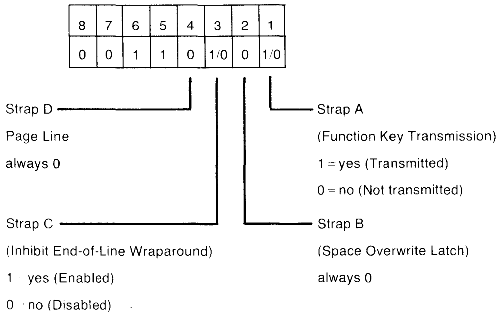
2K bytes

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

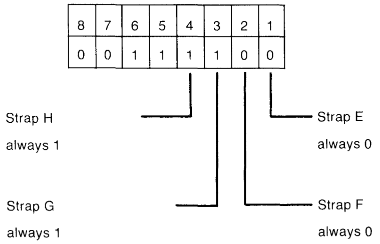
Strap D
Page Line
always 0

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

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

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

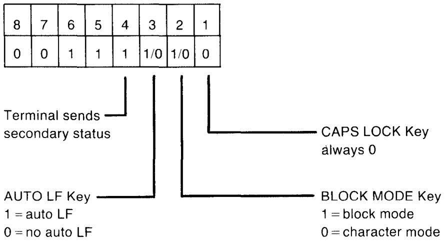
Strap H
always 1

Strap E
always 0

Strap G
always 1

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

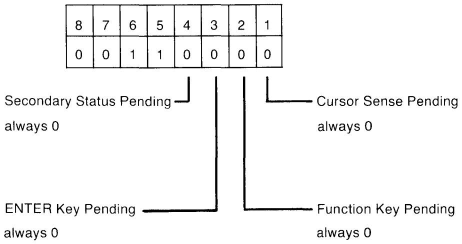
Terminal sends
secondary status

CAPS LOCK Key
always 0

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

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

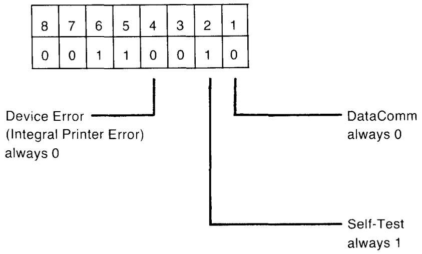
Secondary Status Pending
always 0

Cursor Sense Pending
always 0

ENTER Key Pending
always 0

Function Key Pending
always 0

BYTE 5 ERROR FLAGS



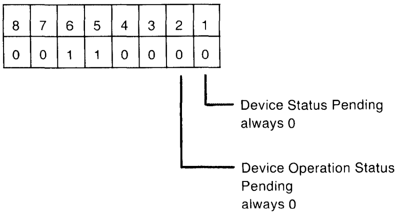
8	7	6	5	4	3	2	1
0	0	1	1	0	0	1	0

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

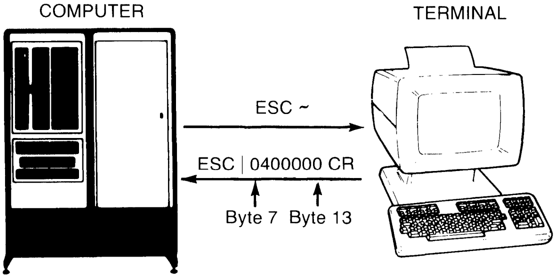
(tracks "S", "F", or "U" completion codes associated with ^{E}c&p device control sequences.)

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

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 (

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

Table of Contents

HP 98700H Graphics Display as a "Terminal"

The Keyboard 2

Alphanumeric Group 2

Numeric Pad Group 2

Display Control Group 2

Edit Group 6

Function Key Group 8

Other Keys 14

The Display 15

Memory Addressing Scheme 15

Cursor Sensing 17

Cursor Positioning 18

Screen Relative Addressing 18

Absolute Addressing 19

Cursor Relative Addressing 20

Combining Absolute and Relative Addressing 21

Display Enhancements 21

Accessing Color 22

Additional Notes About the ITE and the HP 98700H 27

Color Map 27

Screen Windowing 28

Interaction with Starbase and HP Windows/9000 28

Multiple Keyboards 29

Native Language Support 29

Setting up the HP 98700H as a Terminal 31

Controlling Configuration and Status 31

Reconfiguring the Terminal 31

Sending Terminal Status 33

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.

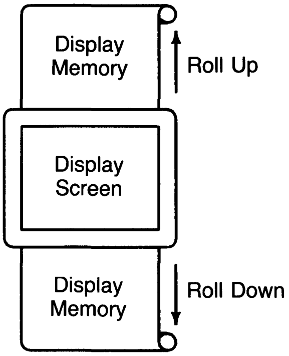


Figure 1. The "Roll" Data Functions

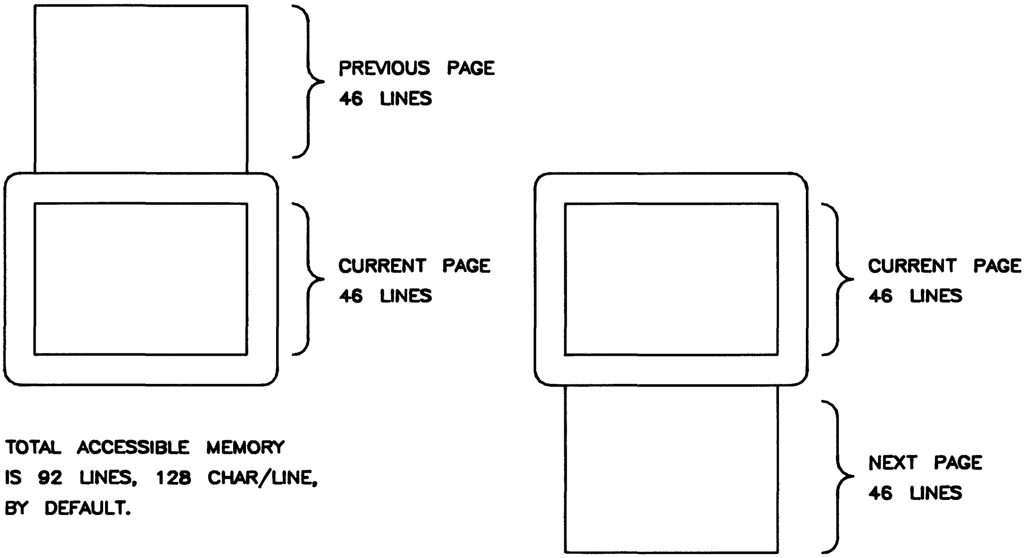


Figure 2: Previous Page and Next Page Concepts

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 characters 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

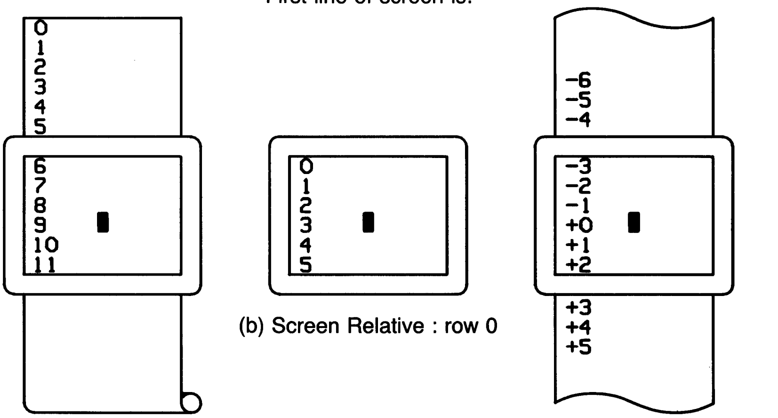
Display memory positions can be addressed using absolute or relative coordinate values. Display memory is made up of 128 columns (0 - 127) and any number of 46 line pages (specified by the HP-UX configuration). As shipped to you, the display memory has 92 lines (0 - 91) of 128 characters (2 screens). 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
```

First line of screen is:



(a) Absolute : row 6

(b) Screen Relative : row 0

(c) 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.

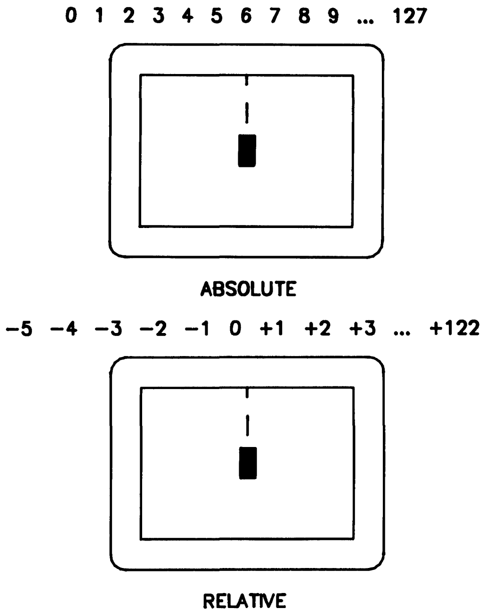


Figure 4. Column Addressing

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

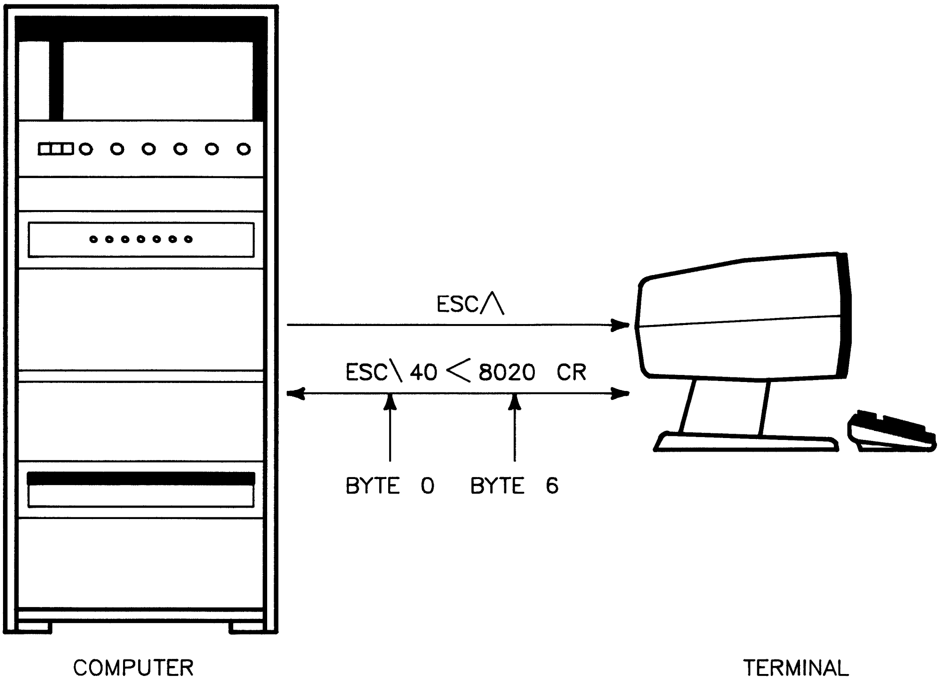
Terminal status is made up of 14 status bytes (bytes 0 through 13) containing information such as display memory size, switch settings, configuration menu settings, and terminal errors. There are two terminal status requests: primary and secondary. Each returns a set of seven status bytes.

Primary Terminal Status

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

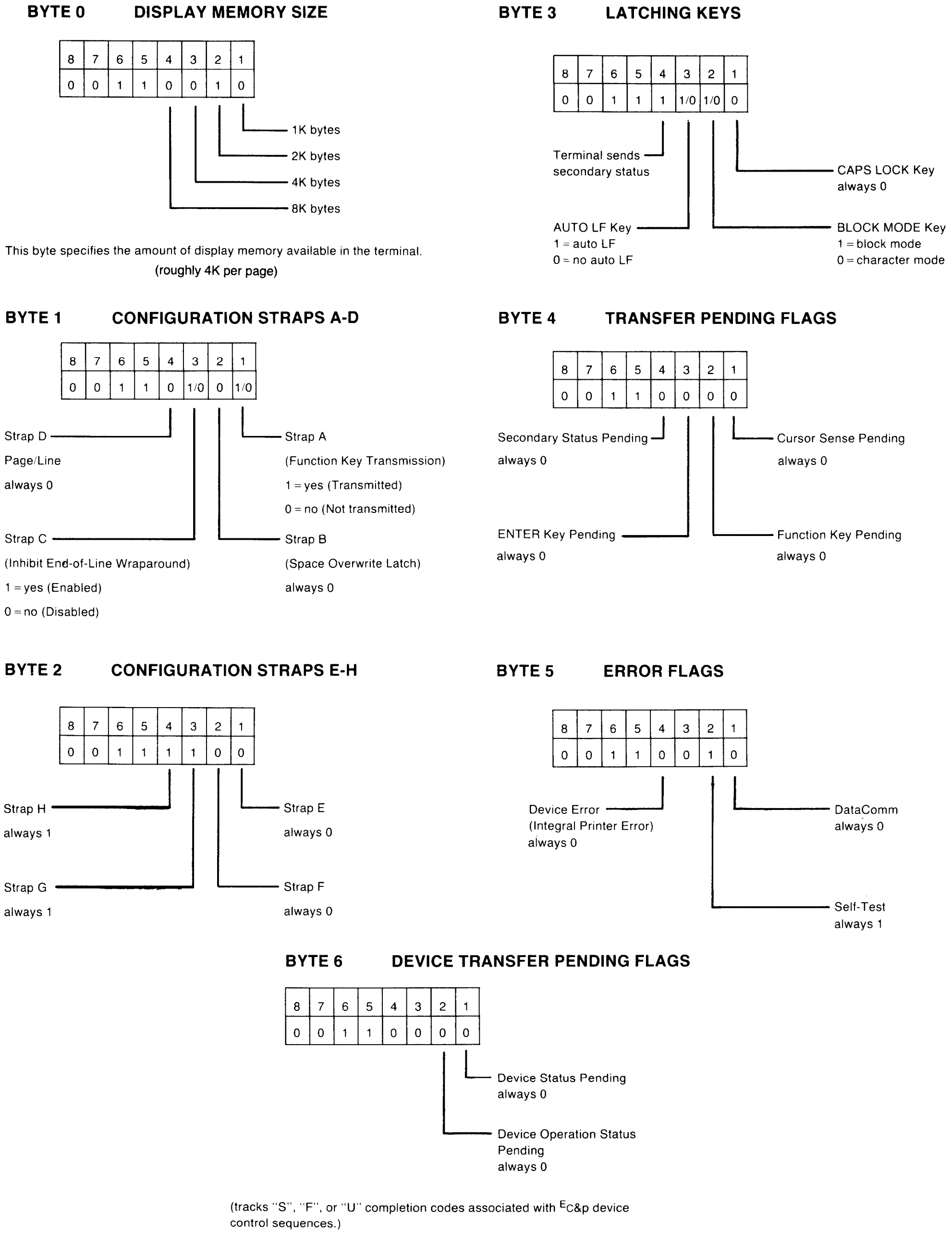
```
\E^
```

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	Configuration Straps A-H
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		

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

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

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 E
always 0

Strap G
always 1

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

CAPS LOCK Key
always 0

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

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

Cursor Sense Pending
always 0

ENTER Key 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.)

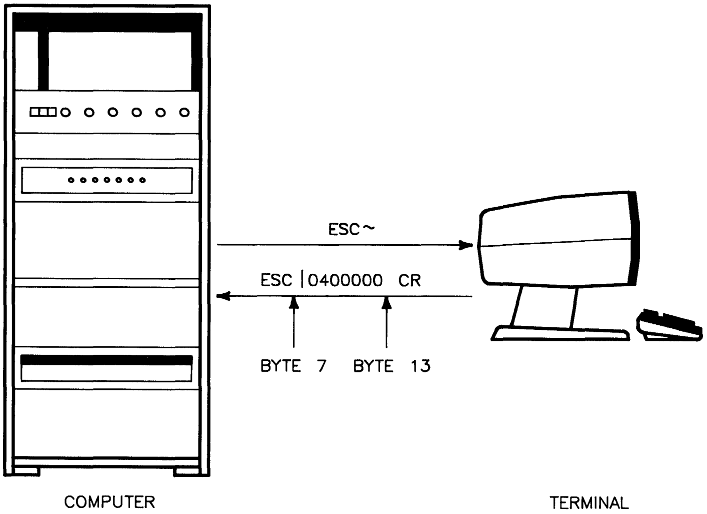
Figure 6. Primary Status Bytes

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

Chapter 1: Using Aterm

Manual Overview 1
Where Do I Start? 2
If You Are a User 2
If You Are the System Administrator 2
Asynchronous Terminal Emulator 3
Invoking aterm 3
Local Commands 5
Configuration Commands 5
Emulator Configuration Display 7
Emulator Status Display 8
Serial Interface Status Display 8
Emulator Input Diversion 9
Emulator Output Diversion 10
Data Link Break 11
Terminating the Emulator 12
Await Prompt Condition 12
Local Shell Commands 13
Example: A Script File to Log In to a Remote Computer 13
Diagnostic Messages (aterm) 15
Problems 19
Bad Connections 19

Chapter 2: Emulator Configuration

Data Communication Links 21
Interface Card Installation 22
Connecting Interface Cables 24
Modem Connections 24
Direct Connections 25
Making a Special Connector 31
Software Configuration 33
Loading the SERIAL.opt Driver 33
Creating an ASI or MUX Device File 34
Preparing the Configuration File 35
Emulator Modes and Handshaking 38
Example Configuration Files (aterm) 41

Appendix A: Series 200/300/500 Character Codes 43

Using Aterm 1

The asynchronous terminal emulator (*aterm*) program, available **only** on HP 9000 Series 500 HP-UX computers, emulates a CRT display terminal while interacting with a remote host computer system, whether the host is running HP-UX or some other operating system. While *aterm* can only be activated on a Series 500 computer that is running HP-UX, the host computer can use any operating system that is capable of communicating with user terminals through an asynchronous (direct or modem) terminal connection.

Aterm and *cu* have similarities as well as some differences. While *cu* is the preferred terminal emulator program for most users, there are cases where *aterm* may be more suitable for certain applications.

Aterm requires a previously prepared configuration file that must be specified when the *aterm* command is invoked from HP-UX. The configuration file specifies the data-communication parameters (such as baud rate, bits per character, parity, and such) that are to be used while passing information between *aterm* and the remote computer.

NOTE

Aterm is available **only** on Series 500 computers.

Manual Overview

This tutorial includes two chapters:

- Chapter 1 contains an overview of *aterm* and discusses emulator operation. It is oriented toward the needs of typical system users.
- Chapter 2 provides hardware and software installation information that is generally most useful to the System Administrator.

Where Do I Start?

If You Are a User

If you are a user, and your System Administrator has set up the hardware and software configuration on your Series 500 computer, refer to the terminal emulator operation topics discussed in the remainder of this chapter.

If You Are the System Administrator

If you are System Administrator, you need to perform the asynchronous terminal emulator (*aterm* program) tasks related to hardware and software installation that are explained in Chapter 2. Chapter 1 contains some helpful tips for use when troubleshooting erroneous operating results.

Asynchronous Terminal Emulator

Invoking *aterm*

The emulator is invoked much like any other HP-UX command. It occupies one executable file, normally located in the directory `/bin`. To run *aterm*, type:

```
aterm cfile
```

where, `cfile` is the name of your previously prepared configuration file. If the configuration 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

~* **RETURN**

to obtain a listing of the parameters used. If you need to change emulator configuration values, you can use local commands that are described in the next section.

If you have enabled 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 the connection is not completed within that time limit, *aterm* aborts.

Occasionally, you may be using *aterm* to respond to incoming calls that connect through an auto-answer modem. If so, you must be prepared to recognize the arrival of the incoming call (such as listening to an audible ringer). When the call arrives, invoke *aterm*. If *aterm* cannot successfully complete the connection within one minute, the message:

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

is displayed and you must repeat the process to make the connection. If the computer that initiated the incoming call has its own time limit for call completion, *aterm* must also complete the connection before that time limit expires or the calling computer may abort the call, causing the connection to fail.

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

```
aterm: connected
```

Using *aterm* for File Transfers

To transmit a file (*trans1* in this example) to a remote computer with *aterm* operating as a non-interactive background process, use standard shell input-diversion notation as in this example:

```
aterm fig3000 <trans1 &
```

Aterm transmits the file *trans1* (which could contain a concatenated series of commands and/or data files for the remote computer) to the remote computer, and uses the configuration file *fig3000* to set up the datacomm link that carries the transmission(s). The amperesand causes HP-UX to run *aterm* as a background process, leaving the keyboard free for other uses while *aterm* is running.

Note that the redirection from *trans1* is a shell input diversion, **not** an emulator input diversion (emulator input diversions are described in the section, "Local Commands" later in this chapter).

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 that redirects output to a specidied file. Diagonostics, while they normally appear on the terminal emulator screen, 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

Local commands are commands that are interpreted by the terminal emulator without being passed on to the host computer. The emulator passes some of these commands to the local HP-UX system for execution. *Aterm* treats any input line that begins with a tilde (~) as a local command, and interprets it accordingly, whether the input is from the keyboard or from a command file. The default local-command escape character is ~, but it can be redefined to a different character by changing the local-command escape character definition in the configuration file specified when invoking *aterm*.

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

The examples in this chapter are based on the assumtion that the default local-command escape character ~ is being used. If you are using an alternate character, substitute the alternate character for ~.

To correct typing errors that occur when entering a local command, use **BACK SPACE** to relocate the cursor, then retype the line, beginning ast the first incorrect character. **ESC-M** cannot be used to clear a line of text.

Configuration Commands

Most *aterm* configuration parameters can be changed to new values without terminating *aterm* or disconnecting the datacomm link. The only parameters that cannot be changed while *aterm* is running are device address, host name, modem connection, and switched service. This feature is useful if you do not know the exact communication format being used by the host computer. Communication parameters such as input and output modes, echo, and terminator sequences can be modified at will, and you can see the results of each modification immediately.

To change configuration while *aterm* is running, use the local-command escape character followed by the *aterm* configuration escape character, an asterisk (*), and the new configuration command. For example, to change the data rate to 9600, type:

```
~*dr 9600 RETURN
```

If the attempted change is not valid, a diagnostic is displayed. Equivalent configuration commands can be placed in a script file if you need to run the emulator as a background process and change configuration while *aterm* is running. Local configuration commands are commonly used to change prompt handshake and prompt timeout without program termination. An example is provided later in this chapter.

The following list shows the recognized configuration command names:

da Serial device file name (no default);

hn Name of remote computer system (no default);

db Number of data bits per character: 5, 6, 7, or 8 (default = 7);

sb Number of stop bits per character: 1, 1.5, or 2 (default = 1);

pa Character parity: none (n), odd (o), even (e), zero (0), or one (1) (default = o);

dr Rate for data sent and received: 50, 75, 110, 134.5, 150, 300, 600, 1200, 1800, 2400, 3600, 4800, 9600, or 19 200 baud (default = 2400 baud);

mc Modem control: enabled (+) for full-duplex modem, or disabled (−) for full-duplex hard-wired connection (default = −);

ss Switched service: auto-answer (a) or manual originate (o) (default = o);

ga Gap: number of character transmission times to delay between successive output characters; values range from 0 to 254 (default = 0);

ec Echo: enabled (+) if the host computer echos characters sent by the emulator, disabled (−) otherwise (default = −);

te Terminal ENQ/ACK: enabled (+) or disabled (−) (default = +);

he Host ENQ/ACK: enabled (+) or disabled (−) (default = −);

tx Terminal XON/XOFF: enabled (+) or disabled (−) (default = −);

hx Host XON/XOFF: enabled (+) or disabled (−) (default = −);

im Input mode: block (b), character (c), or line (l) (default = b);

om Output mode: character (c) or line (l) (default = 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 (default = −);

pt Prompt timeout: number of seconds to allow for receipt of a prompt sequence during an input diversion; values range from 1 to 600, with 0 disabling the timeout altogether (default = 0);

st Single text terminators: list of characters, any of which terminates a line sent by the host computer when the emulator is in input line mode; up to eight characters may be specified (default = no characters);

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 (default = carriage-return/line-feed);

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 (default = DC1);

bl Beginning of line: character to be prefixed to each line sent to the host computer (default = none);

el End of line: one or two characters to be postfixed to each line sent to the host computer (default = carriage-return);

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 the standard input (default = ~).

Note that *aterm* does not support block or format modes.

Emulator Configuration Display

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

```
~*
```

produces a formatted display of the current configuration on the standard output that is similar to the following example:

```
+-----------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>                       !
+---------------------------------------------------------------+
```

The formatted display information can be diverted from the display to a file or another program such as a line-printer spooling program. For example:

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

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

Emulator Status Display

The **?** emulator configuration command character produces a current emulator status listing. Syntax is:

```
~?
```

As before, the formatted display is sent to standard output, but it can be diverted by redirection or to a pipe. Here is an example of part of a typical display produced by the status command:

```
+------------------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                                 !
!                                                               !
+---------------------------------------------------------------+
```

Serial Interface Status Display

To produce an Asynchronous Serial Interface (ASI card) Status listing, use the command:

```
~/
```

Aterm inputs information from the interface status registers on the ASI card, formats it, then sends the formatted display to standard output. As with previous displays, the information can be redirected to a file or piped to a spooler. This information is useful to service personnel when diagnosing emulator malfunctions. Here is a typical example:

```
+---------------------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 CTRL—J
```

where **CTRL—J** sends a line-feed character to the remote computer (**RETURN** cannot be used in this case to transmit the line feed).

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 it 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 acceptable 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.

Problems

This section discusses those problems you are most likely to encounter while using *aterm*.

Bad Connections

Bad connections are by far the most common problem encountered when using *aterm*. Direct and modem connections both present occasional difficulties when contacting remote systems.

If you are using a modem link, check the modems and make sure they are compatible. Using the *cu* command, interactively try to call the other system over the problem line.

Notes

Emulator Configuration 2

This chapter discusses how to set up data communication links between computers, and explains software configuration procedures that must be completed before *aterm* can be used.

Installing data communication links involves the following steps:

1. Determine the type of connection that is needed, and what interfaces are appropriate; that is: should a direct or modem connection be used; does the connection require an asynchronous serial interface (ASI) card, or can a multiplexer (MUX) be installed instead for more efficient use of I/O slots.
2. Install interface(s).
3. Install modem(s) if a modem connection is being used.
4. Install interface cables as appropriate for the type of connection being used.
5. Build and install any necesary adapters (this usually applies only in the case of direct connections).

Data Communication Links

The data communication (datacomm) link is a direct or modem data path between the terminal or computer and another computer that is acting as host. Direct connections consist of cables and adapter wiring directly between the two computers, and are commonly used over short distances where noise is not a problem (about 17 metres, 50 feet, is considered the maximum recommended length of a direct connection, although direct connections have been successfully used over much longer distances when there is little electrical or magnetic noise present.

Modem connections are used for longer distances, and telephone lines usually provide the data path. A modem (MOdulator-DEModulator) is used to convert the digital data from the computer into a form that can be handled by standard telephone circuits. This method of transmission is much less sensitive to external noise effects than direct connections.

The type of datacomm link being used affects the available choice of interfaces for the connection. *Aterm* supports only the HP 27128A ASI card and the HP 27130A/B 8-channel MUX card on the terminal computer. Only the ASI card can be used with modem connections. The 8-channel MUX card is usually preferred for direct connections because it can support eight datacomm links on a single I/O select code, whereas the ASI card supports only a single link. *Aterm* does not support 6-channel MUX cards.

Be Careful

when planning datacomm connections between systems, and especially between HP-UX (or similar) systems that support *uucp* and *cu* facilities. Wiring configurations that work well for *aterm* may not be suitable for other facilities, and configurations that are necessary for other facilities (such as links that must support active *getty*s at both ends simultaneously) may be incompatible with *aterm*'s capabilities and limitations. If you are concerned about potential compatibility conflicts, compare the datacomm link discussions in this tutorial and the tutorial(s) related to other HP-UX datacomm facilities being used.

Interface Card Installation

Only two interface models can be used with *aterm*. If a modem connection is being used, an ASI card must be installed in the *aterm* computer. If you are using a direct connection, either interface will work, but the 8-channel MUX is preferred because it can be connected to several terminals while consuming only one I/O slot in the computer.

CAUTION

Interface cards contain components that are sensitive to damage from electrostatic discharge. Be sure to follow special handling procedures for static sensitive devices when configuring and installing the card. Keep the card in its special conductive plastic bag while unpacking the card from its shipping container and setting the configuration switches.

Card Configuration

The ASI card has a cluster of card configuration switches. Set switches S2 and S8 in their DOWN position, and the remaining switches UP. No other configuration of the ASI card is needed. Software overrides in *aterm* configure the card as needed for correct operation.

There are no configuration switches on the 8-channel MUX card.

Card Installation

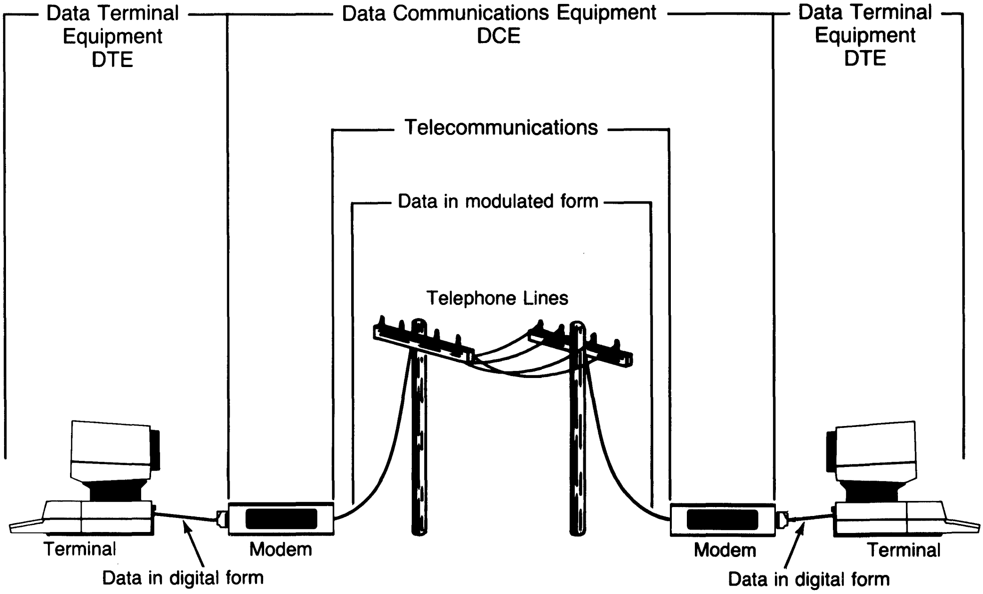
- Be sure the power to your computer is OFF.
- Hold the interface card by the two extractors on the outside corners of the card and slide it into the I/O slot on the computer. Push the card into the internal backplane connector until it is fully seated.
- Install and connect the DTE (male) ASI card modem cable (HP 27128A Opt. 001) or MUX cable and connector assembly to the interface. The conductive grommets on the interface cable must be seated correctly in the computer frame for proper radio-frequency interference shielding. Refer to computer installation manuals for correct procedures. Connecting the cable or MUX connector box to other cables and equipment are discussed in the next topic.

Connecting Interface Cables

Modem Connections

If the distance between your computer 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 is recommended.

Modem connections for Series 500 computers require an HP 27128A Asynchronous Serial Interface (ASI) card whose firmware revision number is 27128-80005 or greater. Only the DTE (male) modem cable (HP 27128A Opt. 001) can be used to connect the ASI card to the modem. Other cables are not suitable for connecting the ASI card to modems.



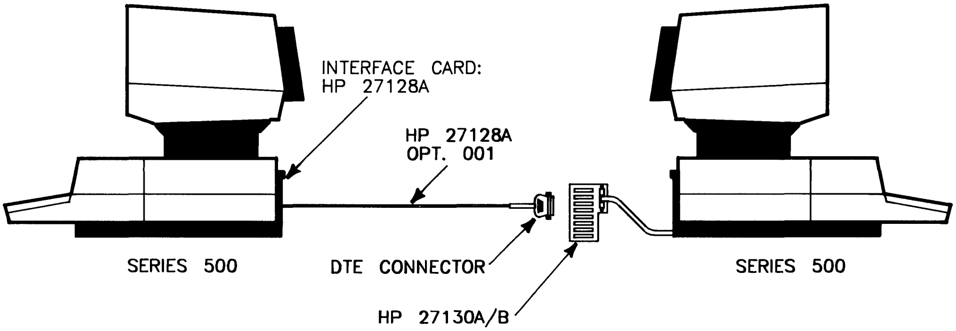
A Typical Modem Connection

Modem connections on Series 500 computers can handle both incoming and outgoing calls over single line. No special modification is necessary; simply connect the DTE (male) cable from the interface to the modem.

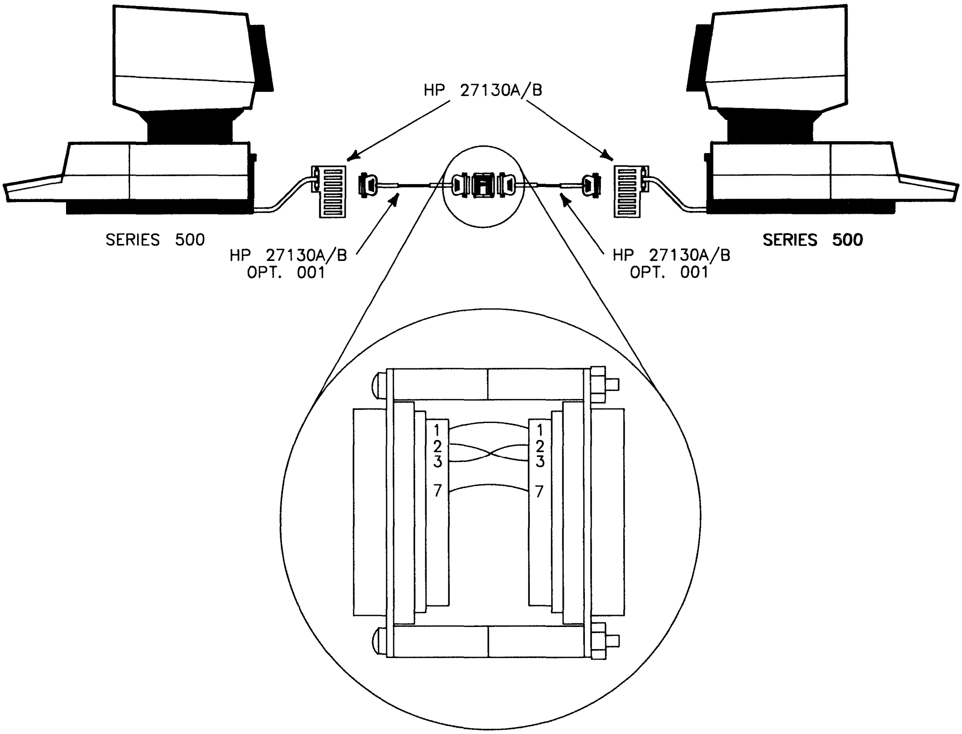
During software configuration, the *mknod* command is used to associate a special (device) file with the interface card on a specified select code, and a flagging mechanism assigns the line as either an incoming or outgoing port. Procedures are explained under the topic heading, "Creating a Device File", in the Software Configuration section of this chapter.

Direct Connections

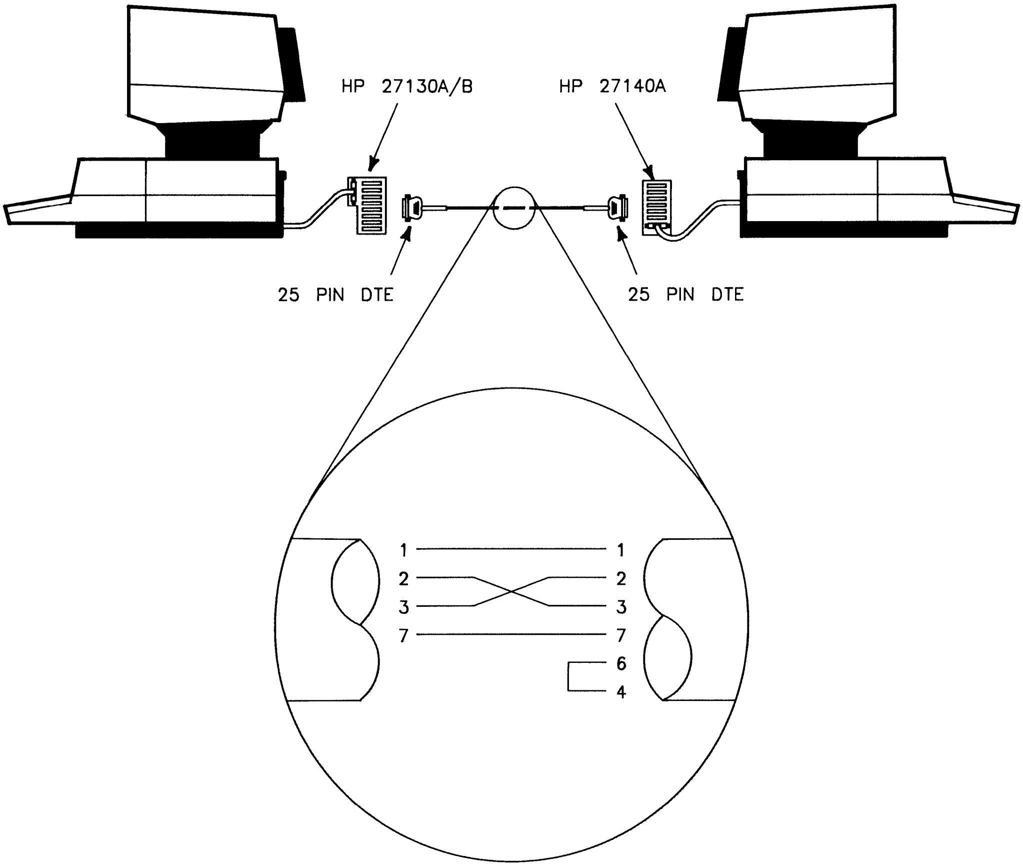
Direct connections can be made between two ASI cards, or between an ASI card and any other compatible interface on a Series 500, Series 200 or other host computer. This section provides examples for each of these combinations, so you should have no difficulty matching one to your needs. Note that two DTE (male) cable ends cannot be connected together unless an appropriate adapter is inserted between them. Exploded views show adapter wiring when an adapter is needed. Construction notes for a typical adapter follow the examples. Note that *aterm* cannot be run on any computer containing a 6-channel MUX card. **The *aterm* computer must contain an 8-channel MUX or ASI card.**



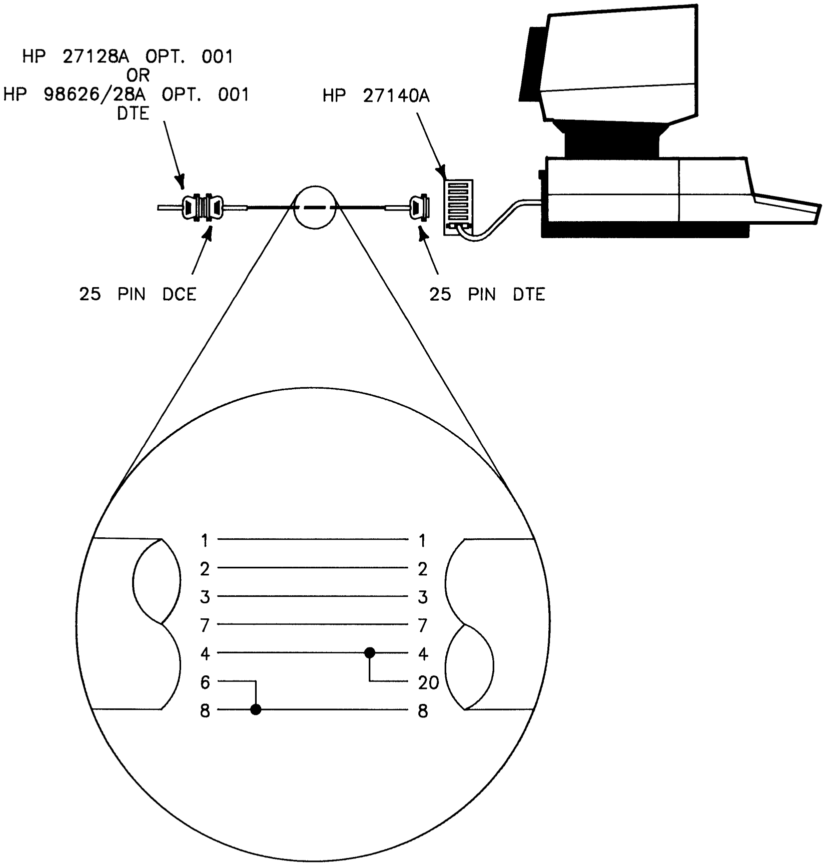
Series 500 MUX to Series 500 ASI



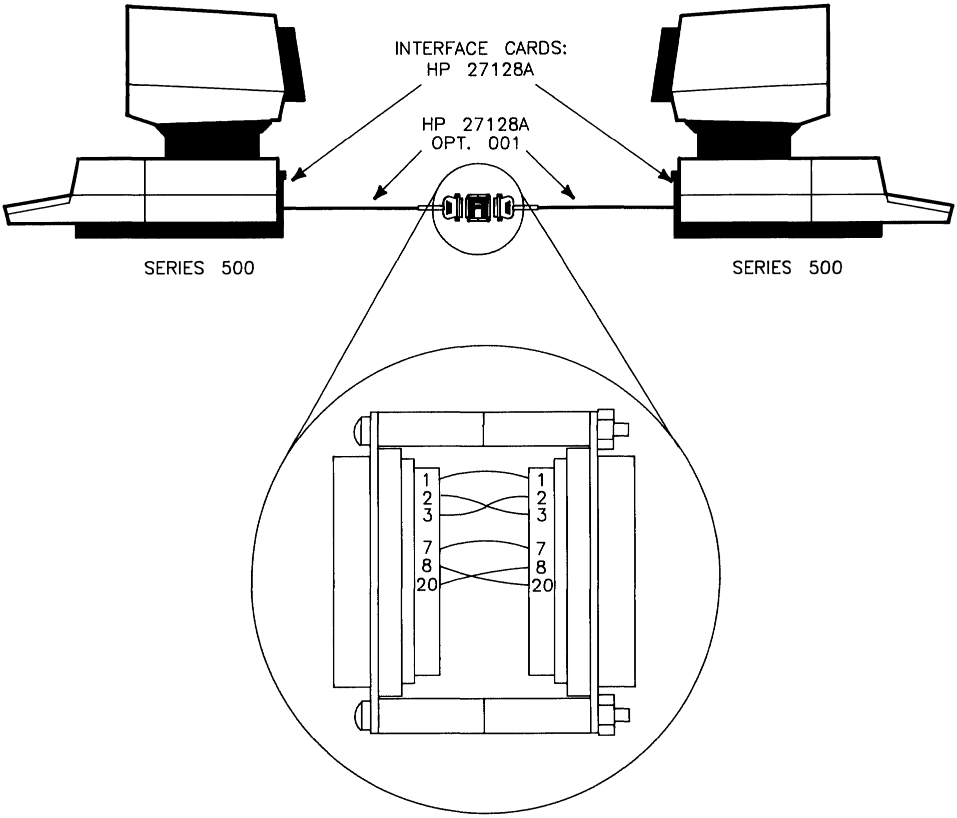
Series 500 MUX to Series 500 MUX



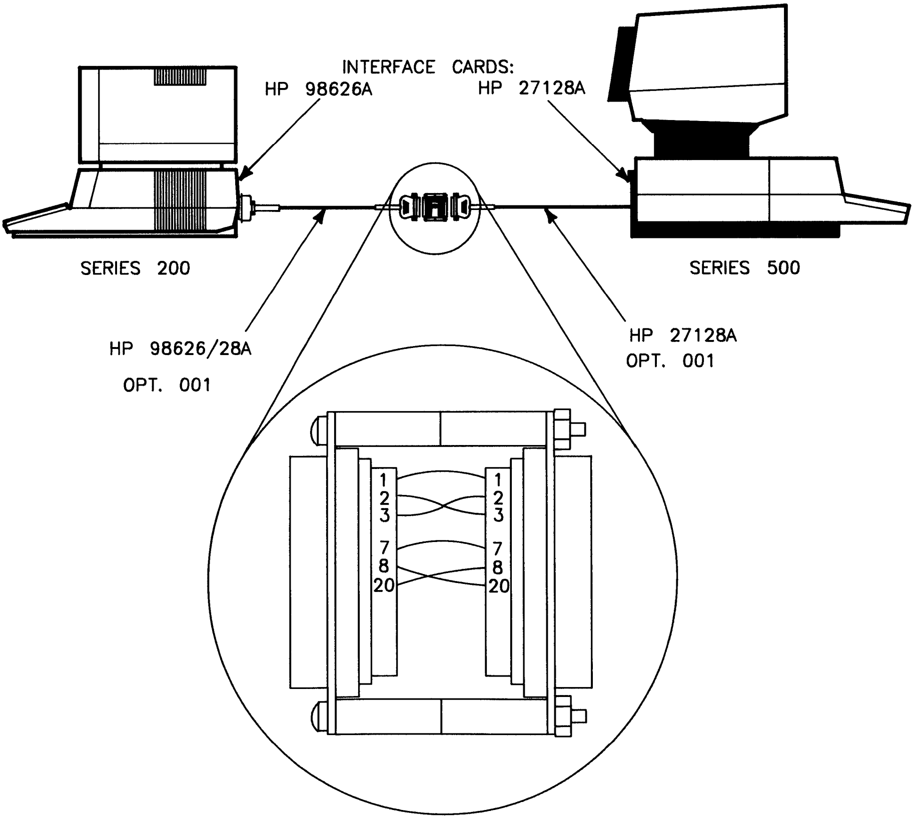
Series 500 MUX to Series 500 6-channel MUX



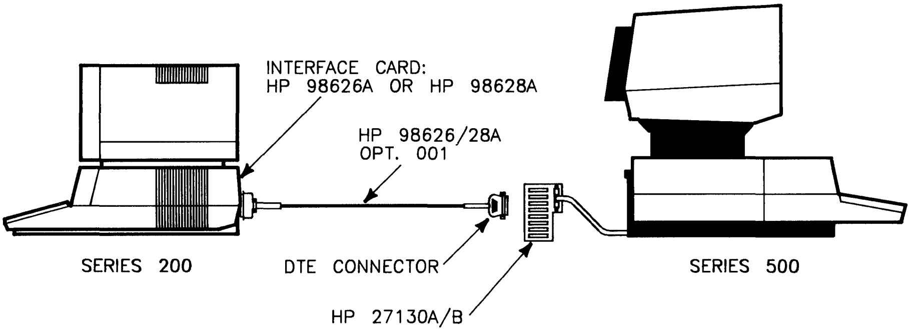
Series 500 ASI to Series 500 6-channel MUX



Series 500 ASI to Series 500 ASI



Series 500 ASI to Series 200/300



Series 500 MUX to Series 200/300

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

- Series 200/300 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 200/300 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.

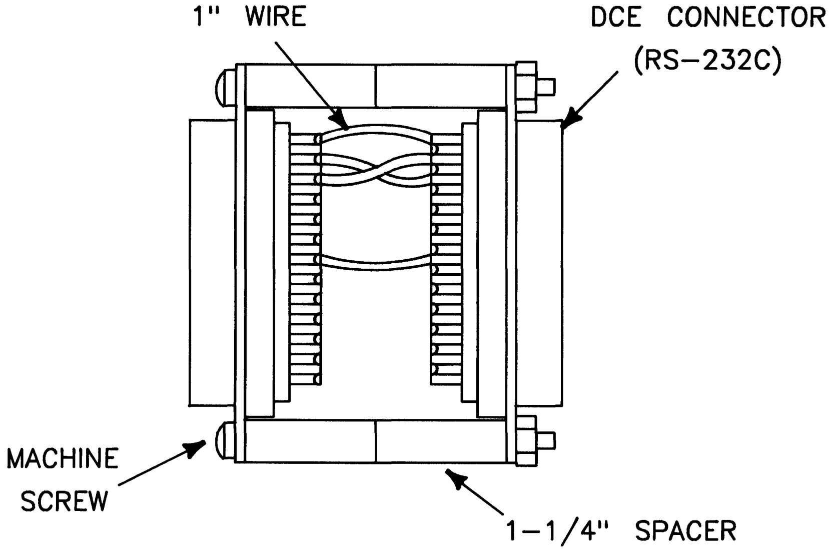
Making a Special Connector

The "special connector" is not a part numbered item that 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 or equivalent) connectors,
- Two 1½-inch long (4 cm) machine screws (HP part number: 2200-0125 or equivalent) 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 conncector.),

- Four 0.625-inch metal (HP part number: 0380-0010) spacers or two 1.25-inch spacers to match the screw length and allow sufficient extra for the nuts and washers,
- Eight 1-inch (2.5 cm) long pieces of 24 gauge insulated hook-up wire stripped about 1/4 inch (6 mm) at each end for soldering.

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



Special Connector

Wiring diagrams on preceding pages in this chapter provide a useful wiring guide for the adapter connector. The exact cross-wiring required may vary, depending on the type of remote system being accessed.

Software Configuration

This section discusses the required software configuration for supporting *aterm* on Series 500 computers. *Aterm* runs only on Series 500 computers. To use *aterm* the System Administrator must:

- Load the SERIAL.opt driver in the boot area of the system disc unless tests show that the SERIAL.opt driver is already there.
- Create a device file.
- Prepare a configuration file.
- Install the *aterm* program.

Loading the SERIAL.opt Driver

The SERIAL.opt driver is not normally included when HP-UX is first installed. Use the following procedure to install it:

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

   ```
   osck -v /dev/hd RETURN
   ```

   If the SERIAL.opt driver has been installed, skip the remaining steps 2, 3, and 4, and continue with device file creation.

2. Determine which HP-UX system software model number you are using from this list (for example, 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

2. Type:

   ```
   oscp -a /system/970xxA/SERIAL.opt /dev/hd RETURN
   ```

   The -a option appends to an existing operating system from a list of ordinary files, and puts the result in the operating system boot area of the system disc. File /system/970xxA/SERIAL.opt contains the driver being copied to the boot area. xx in the driver file pathname shown represents two digits needed to complete the correct model number taken from the list in Step 1. For example, if the system model number is *97070A*, xx corresponds to the digits *70*. /dev/hd represents the name of the special (device) file associated with the system disc.

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

   ```
   osck -v /dev/hd RETURN
   ```

4. To activate the *SERIAL.opt* driver, reboot the system by typing:

   ```
   /etc/reboot -r RETURN
   ```

Error 6 results if you attempt to use *aterm* without the *SERIAL.opt* driver (19) being installed and active.

Creating an ASI or MUX Device File

To configure the emulator, you need to know the name of the special (device) file associated with the ASI (HP 27128A) or 8-channel MUX (HP 27130A/B) card on your system. The name is defined when the system administrator creates the device file by executing the *mknod* command:

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

where:

pathname — is the name of the device file you are creating including the directory path to the file.

mode — is **c** to specify a character-mode device.

driver — is **19**, the number that identifies the general-purpose SERIAL-GSG driver used with the ASI and 8-channel MUX in *aterm* applications.

0x — indicates that all following digits are to be interpreted as hexadecimal.

Sc — is the select code of the ASI or MUX card in hexadecimal notation (use two digits with a leading zero if needed).

Ad — is two hexadecimal digits representing the port number on the MUX with a leading zero. Valid values are 00 through 07. Use 00 for ASI cards.

U is a single-digit hexadecimal value specifying a secondary address, such as a device unit number. This value must be 0 for *aterm*.

V is a single-digit hexadecimal value specifying a secondary address, such as the volume number in a multi-volume disc drive. This value must be 0 for *aterm*.

For example:

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

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

Preparing the Configuration File

Before running *aterm*, you must prepare a configuration file. *Aterm* reads this file in order to establish the correct operating conditions for communicating with the host computer. The configuration file is a normal HP-UX text file, so you can use any text editor to create and update it.

Each configuration command occupies one text line in 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 can precede the command name.

The following table lists all recognized configuration command names with other accompanying information:

- The name of the parameter whose definition or value is altered by the command.
- Description of the parameter, and acceptable arguments or values that can be used with or are required by the given command.
- The default value (if any) that is used if the command is not included in the configuration file.

The only required configuration command that does not have a default value is the special (device) file name that is associated with the ASI or MUX card.

The range of acceptable values for given parameters in the table and some of the behavioral characteristics of *aterm* depend on the type of interface being used. For example, modem control and switched service commands are meaningless if you are using a MUX card, and the emulator **Break** key works only with an ASI card. Refer to the Diagnostic Messages section at the end of Chapter 1 for additional information.

The arguments associated with 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 whose binary values fall in the range of 0 through 37 octal. To express an ASCII control character as an argument to such commands, use the circumflex notation convention. For example, **^M** is equivalent to a carriage return whose octal code value is 015 octal. See the appendix heading, "HP 9000 Character Codes" for a list of circomflex-notation codes. To use a circumflex character as an argument to a command instead of part of a control character representation sequence, precede it with a backslash (\ ).

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 baud.	2400
mc	Modem Control	Enables (+) or disables modem handshake. If enabled, switched service command can be used to select auto-answer or manual originate by monitoring the carrier detect signal from the modem. If disabled, carrier detect is ignored.	–
ss	Switched service	Auto-answer (a) or manual originate (o); relevant for modem lines only.	o
ga	Gap	Specifies the time delay between successive characters in a data transmission as a multiple of the average time required to send a single character. Acceptable values are 0 through 254.	0
ec	Echo	Enabled (+) if the host computer echoes characters sent by the emulator, disabled (–) otherwise.	+

Name	Parameter	Description and Acceptable Values	Default
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.	–
im	Input mode	Method for accepting and processing input data from the host computer; buffered (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

Name	Parameter	Description and Acceptable Values	Default
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 to 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

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" OR "Y"
  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 9600
pa e
db 7
ec +
```

Series 200/300/500 Character Codes

A

The following tables list the characters available for the internal printer and display of the Model 520 computer. The representation for a Series 200/300/500 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 the character is a control character, it may not print or display unless the DISPLAY FUNCTIONS mode of the device is enabled. ASCII control characters can be generated by simultaneously pressing **CTRL** and the key listed in the **CTRL Char** column of the table.

Table A-1. ASCII Character Codes

Gra- phic	CTRL Char	ASCII Character Abbreviation, Description	Binary	Oct	CHR$ Dec	Hex
␀	@	NUL,Null	00000000	000	000	00
␁	A	SOH,Start Of Header	00000001	001	001	01
␂	B	STX,Start Of Text	00000010	002	002	02
␃	C	ETX,End Of Text	00000011	003	003	03
␄	D	EOT,End Of Transmission	00000100	004	004	04
␅	E	ENQ,Enquiry	00000101	005	005	05
␆	F	ACK,Acknowledge	00000110	006	006	06
␇	G	BEL,Bell,attention signal	00000111	007	007	07
␈	H	BS,Backspace	00001000	010	008	08
␉	I	HT,Horizontal Tab	00001001	011	009	09
␊	J	LF,Line-feed	00001010	012	010	0A
␋	K	VT,Vertical Tab	00001011	013	011	0B
␌	L	FF,Form-feed	00001100	014	012	0C
␍	M	CR,Carriage-return	00001101	015	013	0D
␎	N	SO,Shift Out	00001110	016	014	0E
␏	O	SI,Shift In	00001111	017	015	0F
␐	P	DLE,Data Link Escape	00010000	020	016	10
␑	Q	DC1,Device Control 1,X-ON	00010001	021	017	11
␒	R	DC2,Device Control 2	00010010	022	018	12
␓	S	DC3,Device Control 3,X-OFF	00010011	023	019	13
␔	T	DC4,Device Control 4	00010100	024	020	14
␕	U	NAK,Negative Acknowledge	00010101	025	021	15
␖	V	SYN,Synchronous idle	00010110	026	022	16
␗	W	ETB,End Transmission Block	00010111	027	023	17
␘	X	CAN,Cancel	00011000	030	024	18
␙	Y	EOM,End Of Medium	00011001	031	025	19
␚	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 A-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 A-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

Index

a

ASCII Character codes . . . 43-45
Auto-answer modem . . . 3
Await prompt . . . 12

b

BREAK . . . 11
Break, Data Link . . . 11
Buffered input mode . . . 39

c

Cables, datacomm . . . 24-32
Cables, interface . . . 24-32
Changing configuration . . . 5, 6, 7
Character input mode . . . 39
Commands, configuration . . . 5, 6, 7
Commands, configuration file . . . 36-38
Commands, local . . . 5
Configuration, changing . . . 5, 6, 7
Configuration commands . . . 5, 6, 7
Configuration display . . . 7
Configuration file . . . 1, 3
Configuration file commands . . . 36-38
Configuration file example . . . 41, 42
Configuration file, preparing . . . 35
Configuration, interface card . . . 22
Configuration, software . . . 33
Connection timeout . . . 3
Connection to remote computer . . . 3
Connections, bad . . . 19
Correcting typing errors . . . 5

d

Data Link Break 11
Datacomm cables 24-32
Datacomm link installation 21-32
Device Files 34
Diagnostic messages 15-19
Display, configuration 7
Display, status 8
Diversions 4, 9, 10

e

Emulator commands 5
Emulator Modes 38
End of File 12
Error diversions 4
Error messages 15-19
Errors, typing 5
Escape to shell 13
Example:
  Login script 13

f

File, end of 12
File transfers 4
Files, Device Special 34

h

Handshaking 38
Handshaking, prompt 40

i

Input diversions 4, 9, 12
Input modes 38, 39
Installing datacomm links 21-32
Installing interface cards 22-23

Interface cables 24-32
Interface card installation 22-23
Invocation of *aterm* 3

l

Line input mode 38
Line output mode 40
Local commands 5
Local shell commands 13
Local shell termination 13

m

Messages, diagnostic 15-19
mknod 34

o

Output diversions 4, 10
Output modes 39, 40

p

Preparing configuration file 35
Problems 19
Prompt 12
Prompt Handshaking 40

s

Serial Interface status 8
SERIAL.opt Driver 33
Shell escape command 13
Special connector 31-32
Special Files 34
Status display 8
Status, Serial Interface 8

t

Terminating diversions 9, 10

Terminating local shell 13

Terminating the Emulator 12

Tilde character 5

Timeout, connection 3

Troubleshooting 19

Typing errors 5

w

Wait for prompt 2

Table of Contents

Assembler Reference Manual

Program Instructions 1

Program Statement Format 1

Symbols 2

Local Labels 2

Opcodes 2

Size Suffixes 2

Expressions 3

Pseudo-Instruction Syntax And Semantics 4

Interfacing Assembly Routines 5

Linking 5

Calling Conventions 6

Language Dependencies 9

Conversion From The Pascal Language System (PLS) 11

Additions 11

Differences 12

Assembler Reference Manual

Program Instructions

Program Statement Format

Assembly programs are written with one instruction allowed per line. Mnemonic operation codes (opcodes) and register symbols must be written in lowercase. Uppercase and lowercase characters cannot be used interchangeably (the assembler is case sensitive). The number of spaces used to separate elements within an instruction line are not significant (one blank or multiple successive blanks are treated identically).

If a label is present, it must start in the first column of the instruction line. The opcode must start in column two or later (at least one blank must separate label, if present, and opcode). Blanks are not permitted within the operand field. The first blank encountered after the start of the operand field begins the comment field. Here is an example of how typical instruction lines are structured. The first line of numbers identify column positions, and are not part of a program.

```
12345678901234567890123456789012345678901234567890

Label     move a1,a2             comment field
```

An asterisk (*) in column one indicates that the entire line is a comment:

```
12345678901234567890123456789012345678901234567890

*
*         These are comments.
*
```

Symbols

Symbols must begin with an alphabetic character, but can contain letters, numbers, @, $ and _. Symbols can contain any number of characters, but each program statement must be entirely contained on one line.

When the asterisk (*) is not located in the first column of the statement line (comment field), it is interpreted as a symbol having the value of the program counter. Normally, it appears only in the operand field of the program statement unless the statement is a comment.

Register symbols refer to the pre-defined registers **a0** through **a7**, **d0** through **d7**, **sp**, **pc**, **ccr**, and **sr**.

Local Labels

A local label has the form <*digit*>$. A local label can be used to label any machine instruction. Any number of occurrences of the same local label is allowed within an assembly source file. When a local label is referenced, the reference refers to the nearest declaration of the local label.

Opcodes

Most opcodes and their syntaxes are defined in the microprocessor manual shipped with your system. Size suffixes are allowed only for those operations which include a size field in the instruction and for the conditional branch `bcc`. In addition to the opcodes listed in the manual, the Series 200/300 assembler recognizes some variants. For the `bcc` instruction, the form `jcc` can be used. Also, `jbsr` can be substituted for `bsr`. In such cases, the assembler decides the appropriate size for the instruction. No size suffix is allowed.

Size Suffixes

Size suffixes are used in the language to specify the size of the operand in the instruction, including addressable locations and registers. All instructions that can operate on more than one data size assume the default size of word (16 bits) unless a size suffix is used. Size suffixes can also be appended to address register specifications when used in indexed addressing. Operand sizes are defined as follows:

Suffix	Data Unit	Bits
b	byte	8
w	word	16
l	long	32

Expressions

Expressions are evaluated in left-to-right order, and parentheses are permitted. Symbols referring to defined labels are permitted in expressions. The value of these symbols is their relative value within the assembled code. The only operations that can be performed on these symbols are addition and subtraction. One label can be subtracted from another, the result being an absolute value. A label can be added to an absolute value but not to another symbol. Recognized operators include:

Operator	Operation
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
!	Bitwise OR
&	Bitwise AND
^	Bitwise eXclusive OR
<	Shift left
>	Shift right

Pseudo-Instruction Syntax And Semantics

The following commands are direct instructions to the assembler, and do not result in instruction codes to the microprocessor. They are used to create data structures, control program flow and form, and support other programming functions that are not directly related to step-by-step machine instructions. A complete list of microprocessor commands can be found in the microprocessor manual that was shipped with your system.

align <name>,<modulus>

Create a global symbol of type **align**. When the loader sees this symbol it will create a hole beginning at symbol <*name*> whose size will be such that the next symbol will be aligned on a <modulus> boundary.

asciz '<string>'

Put a null terminated <string> into the code at this point.

bss

Put the following assembly into the uninitialized data segment.

comm <name>,<size>

Create a global symbol <name>, put it in the bss segment with size <size>.

data

Place the following assembly in the initialized data segment.

dc[.b|.w|.l]
<expr>|'<string>'[,<expr>|'<string>']

Place the list of expressions <expr> or strings <string> into the code at this point. Size suffixes can be used to specify the units of storage into which the values will be placed. Default is word. In the case of string literals, the amount of storage needed will be determined by the assembler and each character will be assigned into a unit.

ds[.b|.w|.l] <expr>

The units of space are specified by the size suffix. The number of units is determined by the expression.

equ <expr>

Assigns the value and attributes of the expression to the label.

even

Forces even word alignment.

globl <name>[,<name>]...

Declares the list of names to be global symbols.

include "<name>"|<<name>>

Specifies a file to be merged into the assembly at the point where the instruction is located. The file will be searched for according to the conventions of C (see *cc*(1) in the *HP-UX Reference*).

text

Place the following assembly in the code segment.

Interfacing Assembly Routines

Effective use of the assembler requires the ability to interface correctly with higher-level languages supported on Series 200/300 HP-UX computers.

Linking

For a symbol to be known externally, it must be declared in a `globl` statement. It is not necessary that a symbol that has been defined externally be declared within a module; if a referenced symbol is not defined within the module, it is assumed to be externally defined. However, it is generally recommended that all external symbols be declared in a `globl` statement, to avoid confusing global and local symbols.

Calling Conventions

All languages currently supported on Series 200/300 follow certain conventions with regard to subroutine calls. These conventions must be followed correctly in order to call or be called by a higher level language. Calling conventions are summarized as follows:

- Parameters are pushed in reverse order and taken off in the same order as the procedure call;
- The calling routine pops the parameters from the stack upon return;
- The called routine saves and restores the registers it uses (except d0, d1, a0, a1);
- Function results are generally returned in d0, d1;
- `tst.b` required for all stack space used plus that required for the link of any routine called; and
- `link/unlk` instructions are used to allocate local data space and to reference parameters.

These conventions are more easily understood when an example is provided. Consider the following C program which, in turn, produces assembly language output from the C compiler:

```
main()
{
        test(1,2);
}
test(i,j)
register int i, j;
{
        int k;
        k = i + j;
        return k;
}
```

The resulting Assembly Language output (line numbers have been added on the left side for use in the ensuing explanation):

```
 1:           data
 2:           text
 3:           globl   _main
 4:     _main
 5:           link    a6,#-__F1
 6:           tst.b   -__M1-8(a7)
 7:           movem.l #__S1,-__F1(a6)
 8:           move.l  #2,-(sp)
 9:           move.l  #1,-(sp)
10:           jbsr    _test
11:           addq    #8,sp
12:           jra     L12
```

```
13:     L12   unlk     a6
14:           rts
15:     __F1  equ      0
16:     __S1  equ      0
17:     __M1  equ      0
18:           data
19:           text
20:           globl    _test
21:     _test
22:           link     a6,#-__F2
23:           tst.b    -__M2-8(a7)
24:           movem.l #__S2,-__F2(a6)
25:           move.l   8(a6),d7
26:           move.l   12(a6),d6
27:           move.l   d7,d0
28:           add.l    d6,d0
29:           move.l   d0,-4(a6)
30:           move.l   -4(a6),d0
31:           jra      L14
32:           jra      L14
33:     L14   movem.l -__F2(a6),#192
34:           unlk     a6
35:           rts
36:     __F2  equ      12
37:     __S2  equ      192
38:     __M2  equ      0
39:           data
```

Note that when parameters are pushed by the calling routine (_main), the second parameter is pushed first then the first parameter is pushed (lines 8 and 9). When the called routine (_test) accesses the parameters (lines 25 and 26), it finds the first parameter first on the stack followed by the second parameter (line 25 accesses the first parameter and line 26 accesses the second).

Also note that the stack is popped upon return from the subroutine (line 11) and not by the subroutine itself. Since the called routine makes use of d6 and d7, it pushes those registers on the stack (line 24) and then pops them (line 33) before it returns.

The function result is placed in d0 before returning (line 30). Had the function returned a double-precision, floating-point number, that number would have been placed in d0 and d1.

A `tst.b` instruction (line 23) is needed before any use of stack space by any assembly-language routine in order to ensure enough stack space for the routine. If the test fails, the operating system can detect the problem and allocate more stack space for the process. If the test is not performed, the program may die unnecessarily with a segmentation violation. The minimum amount of space that must be specified in the test is the sum of:

- The amount of space taken by the link instruction;
- The greatest amount of space used for any parameters that may be pushed;
- The constant 8 to account for subroutine jumps and the link which that routine may do.

C and other high-level languages use link and unlk instructions (lines 22, 34) in all routines. The link instruction is used to allocate local data space and to allow a constant reference point for accessing parameters. The following illustration shows what happens when the link instruction on line 22 is executed.

Before the link:

8(sp)	value of j
4(sp)	value of i
(sp)	return address

After the link:

12(a6)	value of j	
8(a6)	value of i	
4(a6)	return address	
(a6)	old (a6)	
–4(a6)	value of k	(sp)

Note how the parameter `i` is accessed on line 25. On line 29 the local variable `k` is set. The link instruction is not necessary in assembly language code. If it is not there, however, the routine will not show up in a stack backtrace from **adb**. If a `link` instruction is included, an `unlk` must be executed before returning.

Language Dependencies

C

In C, all variables and functions declared by the user are prefixed with an underbar. Thus, a variable named test in C would be known as _test at the assembly language level. All global variables can be accessed through this name by using a long absolute addressing mode. C always pushes a four-byte quantity on the stack for pointers and any form of integer (char, short, long). C always pushes eight bytes for a floating-point number (floating-point numbers are converted to double-precision).

Fortran

Fortran uses the same naming convention as C, and externals can be accessed in the same fashion. Fortran always pushes the address of its parameter for user-defined functions.

Pascal

In Pascal, any exported user-defined function is prefixed by the module name surrounded by underbars. Thus, for Pascal, a function named funk in module test would be known as _test_funk in the resulting assembly-language code. If a procedure is declared as external as in:

```
procedure proc;  external;
```

all calls to proc emit a reference to _proc.

Global variables are accessed as 32-bit absolute, relative to the global base. The global variable i1 would be accessed as

```
move.l  test+0x4,d0
```

in this example:

```
Pascal [Rev 2.1Ma  4/19/83] test.p                      Page 1

     1:D         0 $list 'test.l',tables$
     2:D         0 program test;
     3:D         1 var
     4:D      8  1   i1,i2: integer;
     5:D         1 procedure p;
     6:D         2   var
     7:D     -4  2     j: integer;
     8:C         2   begin

Dump of P
 j                 var lev= 0d2 addr=-00000004 local

P dump complete
```

```
    9:C        2   end;
   10:C        1 begin

Dump of TEST
 i1                var lev= 0d1 addr=00000004 longabs globalbase = test
 i2                var lev= 0d1 addr=00000000 longabs globalbase = test
 p                 proc lev= 0d1 entry: 00000000

 test              proc lev= 0d0 entry: 00000012


TEST dump complete

   11:C        1 end.
```

Pascal always pushes a four-byte quantity on the stack for pointers and integers. For a user-defined function, any parameter greater than four bytes is passed as an address.

The *HP-UX Reference* manual pages for these compilers should be consulted for further information. Assembly listings can be generated by C and Fortran, and can be consulted to get valuable information. The only current means for looking at the code generated by Pascal is through the *adb* debugger.

Conversion From The Pascal Language System (PLS)

A translator (**atrans**) is provided to assist in converting from PLS assembly language to HP-UX assembly language syntax. All code to be ported should be run through the translator first. Lines requiring human intervention will be noted by the translator. Tasks performed by the translator are explained in the *HP-UX Reference*, *atrans*(1) manual page.

atrans cannot detect or alter parameter passing conventions that are pushed in reversed order on PLS.

as assumes `rorg 0` for all assemblies. **as** does not generate relative references to external symbols; all external references are absolute. As such, code size can increase when being ported from PLS to HP-UX.

as does not support Pascal modules.

as accepts the same syntax as the PLS assembler for all machine instructions with these exceptions:

Additions:

- **as** accepts `jcc` where `cc` is a condition code accepted by `bcc`. In this case, **as** decides the length of the instruction required.
- **as** accepts a greater number of operators for expressions. Parentheses are permitted within expressions.
- **as** accepts an immediate operand for the register list in a `movem` instruction. Needed for compiler.
- **as** allows numeric value for displacement as in `12(pc,d6)`. Needed for compiler.
- **as** accepts <digit>**$** to specify a local label.

Differences:

- **as** is a case-sensitive assembler. All opcodes and register names must be listed in lowercase.
- **as** accepts `(pc)` to specify pc-relative references. This is the only way to specify pc-relative.
- The PLS assembler assumes pc with index in some cases for a parameter of the form `8(a0)`. **as** does not.

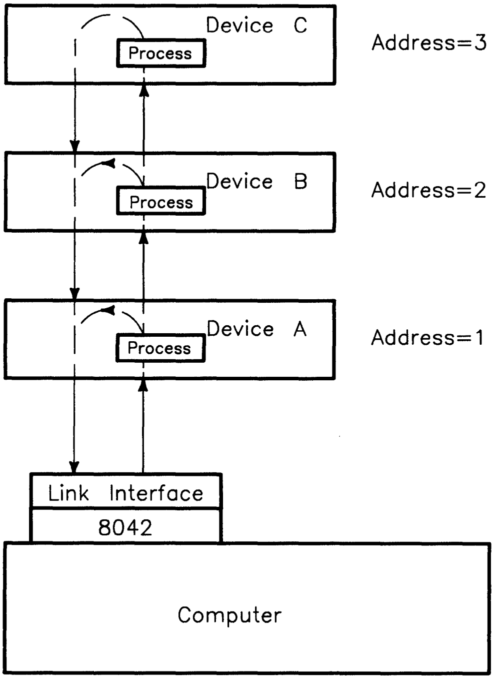
The greatest differences occur in supported pseudo-instructions. The only PLS pseudo-instructions that are supported are `dc`, `ds`, `equ`, and `include`. The translator handles some other pseudo-instructions, but many will have to be handled by hand.

Using HP-HIL Devices

The Interface to HP-HIL Devices

This article describes communication with the HP-HIL interface, and other functions provided by the HP-HIL peripheral processor (8042). It is primarily a description of the enhancements added to handle the HP-HIL interface. This interface is capable of supporting up to seven devices, such as graphics input, system ID Module, and other peripherals generally related to human input, as well as the system keyboard.

Before launching into a discussion of the workings of the HP-HIL interface, a general overview needs to be presented. HP-HIL stands for "Hewlett-Packard Human Interface Link". The following diagram illustrates the basic components.



Hewlett-Packard Human Interface Link

HP-HIL initialization takes place in the following manner. The peripheral devices in the HP-HIL link are powered up when the computer is turned on. Next, because of the "linkback mode" each device is recognized by the computer as the "last" device in the link. The "linkback mode" is where the computer sends out a signal and the HP-HIL device sends that same signal back to the computer through the return side of the device. Each device in the link is checked in this manner until there are no additional devices to check in the link. Note that the computer **does not** know the type of each device in the link; it just knows that there is a device at that location in the link.

To further explain the HP-HIL initialization process, the following example is given. Assume the computer has sent out a signal, looking for the first device on the link. In our previously shown diagram, it would find Device A. Being the first device on the link, it's address is considered to be 1. The computer then instructs Device A to *exit* linkback mode; that is, send the signals through to a possible next device. The computer then attempts to contact the second device on the link. Device B responds and is assigned address 2. The computer now knows that there are at least two devices on the link. The computer commands Device B to exit linkback mode, and attempts to contact the next device. Successful, the computer now knows about Device C. As our diagram illustrates, Device C is the last device on the link, so the process proceeds differently at this point.

The computer instructs Device C to exit linkback mode, and attempts to contact (nonexistent) Device D. Since it is not there, a timeout occurs, and the computer deduces that Device C is the last device on the link. Therefore, it instructs Device C to once again enter linkback mode, and the link is configured.

The link can deal with a maximum of seven devices at any one time (see the Note in the section, "A Few HP-HIL Devices"). If there are eight or more devices physically connected, the devices after number seven are not found.

As the above discussion indicates, the address of a particular device is merely its topological order of placement along the link. In the above diagram, Device A has address 1, B has address 2, and C has address 3. This is only a result of their physical order of connection. If Device C had been connected between Devices A and B, Device A would still have been address 1, but Device C would be address 2, and B would be address 3. The type of device is irrelevant to the address assigned to it.

After the link is operational, and during subsequent link operations, each device looks at the data being sent down the link. If a device notices that the destination address associated with the link data is the same as that device's address, that device receives and acts on the data. Otherwise, the data is merely shuttled along to the next device.

A Few HP-HIL Devices

This section provides a brief description of a few of the HP-UX supported HP-HIL devices. You can make use of these devices by writing special programs to control them in the C Language, FORTRAN, or Pascal.

Before you can use an HP-HIL device, your terminal or computer must meet the following requirements:

- It should have a built-in HP-HIL interface present. This is the case for the HP 150-II, HP 150-III, HP Models 217, 237, 310 and 320, Vectra Personal Computer, Integral Personal Computer, and Model 550 with an HP 98700H Graphics Display Station. The HP 2393 and HP 2397 terminals also have built-in HP-HIL interfaces.
- If you are using a Model 220, it should have an HP 9920 Option 535 (HP 09920-66535, HP-HIL Keyboard/HP-IB Interface) card inserted in its backplane.

The following is a list of HP-HIL devices supported by the HP-UX system. This list also provides the maximum current which each device uses.

NOTE

The total current your HP-HIL link can use before it stops working is 750 milliamps. This current limit is true for all HP-UX computers except the Integral PC and HP 98700A/H which have a total current limit of 520 milliamps.

When determining the total current used by your HP-HIL link, you should note that the current limits listed in this section are maximum current limits. The typical current used by each HP-HIL device is approximately two-thirds of this value, so when calculating the total current used by your HP-HIL link you need to take two-thirds of the sum of the maximum current values.

To determine the total current which your HP-HIL link draws when connected to the HP-HIL interface, add up the maximum current used by each device in the link and multiply the result by two-thirds. Again use the following list to determine the maximum current each device uses.

- **HP 35723A (HP-HIL/Touch Bezel)** — This module is a screen bezel which is placed over the bezel of the HP 35731 (medium resolution black and white monitor) and HP 35741 (medium resolution color monitor) 12-inch video monitors. It can be programmed to select various functions by simply touching the screen. Note that this device is simply a lower resolution digitizer. The maximum current this device uses is 200 milliamps.

- **HP 46021A (HP-HIL Keyboard)** — This keyboard has alphabetic and numeric keys similar to those on a typewriter. Note that this keyboard replaces the HP 46020A keyboard and that all of the keys in each key group of the HP 46021A function the same as those of the HP^{4}6020A. The key groups you will find on this keyboard are as follows:

  - **Character Entry Group** — allows alphabetic and numeric characters, as well as mathematical and commercial signs to be entered. It also contains data control keys such as [Back space] and [Return].

  - **Numeric Group** — provides for rapid entry of numeric data.

  - **Display Control Group** — controls the location of the cursor on the display.

  - **Edit Group** — allows data to be inserted in and deleted from the display.

  - **Function Key Group** — provides you with system defined function key labels, as well as with user defined function key labels.

  - **System Control Group** — controls system functions related to display operations, such as using the [Stop] key to suspend the display.

  The maximum current this device uses is 100 milliamps.

- **HP 46060A (Two-Button Mouse)** — This module is a relative graphics input device for some graphics programs. It is commonly used to move the cursor to any position on the CRT (display) without using arrow keys or an HP 46083A (Knob). The maximum current this device uses is 200 milliamps.

- **HP 46080A (Extension Module)** — The Extension Module allows you to increase the distance between any HP-HIL device by eight feet. Note that the HP-HIL link is capable of handling seven addresses and that the Extension Module **does not** occupy one of these addresses. The maximum current this device uses is 25 milliamps.

- **HP 46081A (HP-HIL/Audio Extension)** — The HP 46081A Module allows a separation of 2.4 meters (8 feet) between the host computer and another HP-HIL device. This module also contains a speaker. Note that the HP-HIL link is capable of handling seven addresses and that the Audio Extension **does not** occupy one of these address. The maximum current that this device uses is 25 milliamps.

- **HP 46082A/B (HP-HIL/Audio Remote Extension)** — The HP 46082A Module allows a separation of 15 meters (49.2 feet) between the host computer and the graphics display station, and the HP 46082B Module allows a separation of 30 meters (98.4 feet). The Audio Remote Extension module also contains a speaker. Note that the HP-HIL link is capable of handling seven addresses and that the Audio Remote Extension module **does not** occupy one of these address. Each extension is a pair of modules plus a length of special link cable and RGB coax cables. The maximum current that each of these devices use is 50 milliamps.

- **HP 46083A (Rotary Control Knob)** — This module provides the additional feature of a rotary control knob to your system. Note that a switch is provided which toggles the knob from X-axis data to Y-axis data. The maximum current this device uses is 110 milliamps.

- **HP 46084A (HP-HIL ID Module)** — The HP 46084A Module is an HP-HIL device that returns an identification number for identifying you as the computer user. The identification number is unique to your particular ID Module. This allows application programs to use the ID Module to control access to program functions, data bases, and networks. Note that the identification number is the product/exchange and serial numbers returned in a packed format as explained in the section of this article, "Report Security Code". The maximum current this device uses is 60 milliamps.

- **HP 46085A (Control Dials)** — This module provides nine user-definable knobs. These knobs can be software defined to provide zooming, panning, rotation, horizontal and vertical motion, color translation, and menu control when using graphics. Each knob on the HP 46085A can be defined in software to do functions other than those mentioned. Note that the HP 46085A Module occupies 3 addresses on the link. Each horizontal row corresponds to one address (bottom to top). The maximum current this device uses is 320 milliamps.

- **HP 46086A (Function Box)** — This module provides 32 function keys to select software-defined functions. A status LED, which is controlled by software, provides an indication of when the device is sending or receiving data. This device uses a non-standard keycode set (Keycode Set 2) which is shown below. The maximum current this device uses is 80 milliamps.

Keycode Set 2 for the Function Box

(press value/release value)

	0/1	2/3	4/5	6/7	
8/9	10/11	12/13	14/15	16/17	18/19
20/21	22/23	24/25	26/27	28/29	30/31
32/33	34/35	36/37	38/39	40/41	42/43
44/45	46/47	48/49	50/51	52/53	54/55
	56/57	58/59	60/61	61/63	

- **HP 46087A (A-size Digitizer)** — The A-size Digitizer provides the capability to enter data from an ISO A4 or ANSI A-size drawing, or free-hand graphics input. It uses either a pen-like stylus or optionally an HP 46089A (Four-Button Cursor). The maximum current this device uses is 200 milliamps.

- **HP 46088A (B-size Digitizer)** — The B-size Digitizer provides the capability to enter data from an ISO A3 or ANSI B-size drawing, or free-hand graphics input. It uses either a pen-like stylus or optionally an HP 46089A (Four-Button Cursor). The maximum current this device uses is 200 milliamps.

- **HP 46089A (Four-Button Cursor)** — This device is a four switch puck that may be used on the A or B-size Digitizer in place of the stylus. This device **does not** take an address space, and it **does not** use any additional current.

- **HP 46094A (HP-HIL/ Quadrature Port)** — This device provides the capability for interfacing an off-the-shelf 3 Button Mouse, Trackball or any other device which provides an output in quadrature to the HP-HIL link. The maximum current this device uses is 200 milliamps.

- **HP 92916A (Bar-Code Reader)** — This module reads all standard bar-codes using a wand as the input device. It provides you with an effective and reliable alternative to the time consuming keyboard for data entry. Note that HP-UX supports this device in the keyboard mode, where the input from the device looks like keycodes. The keycodes which can be read by the Bar-Code Reader are: 3 of 9, Interleaved 2 out of 5, UPC/EAN, and Codabars USD-4 and ABC. The maximum current this device uses is 200 milliamps.

For more information on these devices, call your local HP Sales or Service Representatives.

Using HP-HIL Devices

This section gives a procedure for creating special (device) files for your HP-HIL devices, provides programs for identifying HP-HIL devices, and includes tables for interpreting data for the sample programs. This section also includes a discussion of the commands (opcodes) used in the macros located in the file */usr/include/sys/hilioctl.h*.

Note

HP-HIL devices can be added to or removed from the HP-HIL link without effecting the HP-UX operating system while it is running. However, if you are running an application which requires the use of that particular device and you:

- remove the device from the link, or
- open the link to the device, or
- open the link to add a new device

your application might not recognize the change and as a result it will not work as expected. An HP-HIL device can be added anywhere in the HP-HIL link provided it is not a non-extendable device (e.g. HP 46060A, Two-button Mouse), in this case the device can only be added to the end of the link.

A Few Terms

The following terms will be used throughout this article:

- A **special (device) file** is a file associated with an I/O device. Special (device) files are read and written just like "ordinary files" (a type of HP-UX file containing either a program, text or data), but requests to read or write result in activation of a driver of the associated device. Entries for each file normally reside in the */dev* directory. In this article, you will find that a special (device) file is referred to as a device file or a special file.

- A **macro** is a command which contains a set of instructions to be performed. The term **macro** was derived from the word **macroinstruction**.

- A **data frame** is the way information travels through the HP-HIL link. It consists of 15 bits of information which include: start (1 bit), stop (1 bit), command (1 bit), parity (1 bit), address (3 bits), and data (8 bits). The frame is transmitted around the link at the rate of 10 micro-seconds per bit, or 150 micro-seconds per frame.

Bit Format for Data Frame

start	address	command	opcode/data byte	parity	stop
1	1 2 3	1	1 2 3 4 5 6 7 8	1	1

- A **command (opcode)** in this article is an operational code used in a lower level programming language (assembly) to perform an operation, such as incrementation, inversion or multiplication, on one or more operands. This is the definition for the term **opcode**; however, in this article it will be used as the definition for the term **command**.

- A **path name** is a sequence of directories and "ordinary files" (HP-UX files containing either programs, text or data) separated by /'s which map out a path leading to a destination file.

- A **select code** is part of an address used for devices; a number determined by a setting on an interface card to which a peripheral device is connected. Multiple peripherals connected to the same interface card share the same select code.

Creating a Special Device File for HP-HIL Devices

Each device on the HP-HIL link has a unique address based on its position in the link (e.g. the first addressable HP-HIL device is address 1 and so on). To access a device, you must first create a special (device) file using the *mknod* command. The format for using this command is as follows:

```
   /etc/mknod /dev/device_name c XX 0xSS0GA0
```

where:

`device_name` is some name you give to identify the HP-HIL address (e.g. `hil1`) for which you are creating a device file.

`c` specifies the character mode rather than the block mode.

`XX` is the major (driver) number used with the device you are creating. Series 200/300 computers use major (driver) number 24 for communicating with HP-HIL devices. Series 500 computers use major (driver) number 42 for communicating with HP-HIL devices.

`SS` is the select code of the device. Series 500 uses select code 0xff and Series 200/300 uses select code 0x00.

`G` is the slot number minus 4 (on the Series 500) for the HP 98288 Display Station Buffer associated with the HP 98700 Display Station. The slot numbers used for the Display Station Buffer on the Series 500 are slots 4 through 7. If you inserted your Display Station Buffer in slot 4, the value of `G` is 0. This field is always 0 on the Series 200/300 computers.

`A` is the HP-HIL address of the device you wish to talk with. The address ranges from one to seven. The first device which can be assigned an address in the HP-HIL link gets address number one; the second addressable device gets address number two and so on. An addressable device is a device which is not an extension device, such as the HP 46080A (Extension Module) and the HP 46081A (HP-HIL/Audio Extension).

You may need to create a special (device) file for the 8042 driver so you can talk to the timer, talk to the beeper, or change the keyboard repeat rate. To do this, use the following form of the *mknod* command:

```
/etc/mknod /dev/device_name c XX 0xSS0G00
```

`device_name` is some name you give to identify the HP-HIL address (e.g. `rhil`) for which you are creating a device file.

`c` specifies the character mode.

`XX` is the major (driver) number used with the device you are creating. Series 200/300 computers use major (driver) number 23 for communicating with the 8042 driver. Series 500 computers use major (driver) number 41 for communicating with the 8042 driver.

`SS` is the select code of the device. Series 500 uses select code 0xff and Series 200/300 uses select code 0x00.

`G` is the slot number minus 4 (on the Series 500) for the HP 98288 Display Station Buffer associated with the HP 98700 Display Station. The slot numbers used for the Display Station Buffer on the Series 500 are slots 4 through 7. If you inserted your Display Station Buffer in slot 4, the value of `G` is 0. This field is always 0 on the Series 200/300 computers.

Additional Considerations

When you execute a long listing of the */dev* file, you will find that the device files for HP-HIL devices on Series 200/300 computer are as follows:

```
crw-rw-rw-   1 root     root       24 0x000010 Oct 29 09:02 hil1
crw-rw-rw-   2 root     root       24 0x000020 May 22  1985 hil2
crw-rw-rw-   1 root     root       24 0x000030 May 22  1985 hil3
crw-rw-rw-   1 root     root       24 0x000040 May 22  1985 hil4
crw-rw-rw-   1 root     root       24 0x000050 May 22  1985 hil5
crw-rw-rw-   1 root     root       24 0x000060 May 22  1985 hil6
crw-rw-rw-   1 root     root       24 0x000070 May 22  1985 hil7
crw-rw-rw-   1 root     root       25 0x000080 May 22  1985 hilkbd
```

Device files for HP-HIL devices on Series 500 computers are listed as follows:

```
crw-rw-rw-   1 root     root      42 0xff0010 Oct 29 09:02 hil1
crw-rw-rw-   2 root     root      42 0xff0020 May 22  1985 hil2
crw-rw-rw-   1 root     root      42 0xff0030 May 22  1985 hil3
crw-rw-rw-   1 root     root      42 0xff0040 May 22  1985 hil4
crw-rw-rw-   1 root     root      42 0xff0050 May 22  1985 hil5
crw-rw-rw-   1 root     root      42 0xff0060 May 22  1985 hil6
crw-rw-rw-   1 root     root      42 0xff0070 May 22  1985 hil7
crw-rw-rw-   1 root     root      43 0xff0080 May 22  1985 hilkbd
```

Note that the last device file (*hilkdb*) listed has a different major (driver) number. This is the device file for the HP-HIL "cooked" keyboard driver. The HP-HIL "cooked" keyboard driver does not require a new keyboard; it simply provides a protocol conversion for using your present HP-HIL keyboard. This protocol only recognizes the down stroke of a key when it is pressed (not both up and down keystrokes). Using this protocol conversion with programs that trap individual keystrokes (by reading from the HP-HIL interface) makes the application programs more compact, because they are keeping track of fewer keystrokes.

Note that the keyboard sends a set of data which consist of a four byte time stamp, one byte that contains status information as follows:

- 1000xxxx — both Shift and CTRL have been pressed,
- 1001xxxx — only CTRL has been pressed,
- 1010xxxx — only Shift has been pressed,
- 1011xxxx — Shift and CTRL have not been pressed,

and a final byte that contains a keycode taken from the table below. The following is a table of the keycodes for the HP-HIL "cooked" keyboard driver.

Keycodes for the HP-HIL "Cooked" Keyboard Driver

Keycodes in		Key Label	
hex	decimal	Unshifted	Shift
00	0	Unused	
01	1	~	,
02	2	|	\
03	3	ESC	DEL
04	4	Unused	
05	5	Break	Reset
06	6	Stop	
07	7	Select	
08	8	Enter[1]	
09	9	Tab[1]	
0A	10	(Blank 1)[1]	
0B	11	(Blank 2)[1]	
0C	12	(Blank 3)[1]	
0D	13	(Blank 4)[1]	
0E	14	↖	
0F	15	Prev	
10	16	Next	
11	17	Enter	Print
12	18	Extend char (left)	
13	19	Extend char (right)	
14	20	System	User
15	21	Menu	
16	22	Clear line	
17	23	Clear display	
18	24	Caps	
19	25	Tab	
1A	26	Unused	

[1] Numeric keypad key.

Keycodes for the HP-HIL "Cooked" Keyboard Driver (continued)

Keycodes in		Key Label	
hex	decimal	Unshifted	Shift
1B	27	f1	
1C	28	f2	
1D	29	f5	
1E	30	f6	
1F	31	f7	
20	32	f3	
21	33	f4	
22	34	▼	
23	35	▲	
24	36	f8	
25	37	Unused	
26	38	◄	
27	39	►	
28	40	Insert line	
29	41	Delete line	
2A	42	Unused	
2B	43	Insert char	
2C	44	Delete char	
2D	45	Unused	
2E	46	Back space	
2F	47	Unused	
30	48	Unused	
31	49	Unused	
32	50	Unused	
33	51	Unused	
34	52	Unused	
35	53	Unused	

Keycodes for the HP-HIL "Cooked" Keyboard Driver (continued)

Keycodes in		Key Label	
hex	decimal	Unshifted	Shift
36	54	Unused	
37	55	Unused	
38	56	Unused	
39	57	Enter	
3A	58	Unused	
3B	59	Unused	
3C	60	0[2]	
3D	61	.[2]	
3E	62	,[2]	
3F	63	+[2]	
40	64	1[2]	
41	65	2[2]	
42	66	3[2]	
43	67	-[2]	
44	68	4[2]	
45	69	5[2]	
46	70	6[2]	
47	71	*[2]	
48	72	7[2]	
49	73	8[2]	
4A	74	9[2]	
4B	75	/[2]	
4C	76	Unused	
4D	77	Unused	
4E	78	Unused	
4F	79	Unused	
50	80	1	!
51	81	2	@

[2] Numeric keypad key.

Keycodes for the HP-HIL "Cooked" Keyboard Driver (continued)

Keycodes in		Key Label	
hex	decimal	Unshifted	Shift
52	82	3	#
53	83	4	$
54	84	5	%
55	85	6	^
56	86	7	&
57	87	8	*
58	88	9	(
59	89	0	)
5A	90	-	_
5B	91	=	+
5C	92	[	{
5D	93	]	}
5E	94	;	:
5F	95	’	"
60	96	,	<
61	97	.	>
62	98	/	?
63	99	(Space)	
64	100	O	
65	101	P	
66	102	K	
67	103	L	
68	104	Q	
69	105	W	
6A	106	E	
6B	107	R	
6C	108	T	
6D	109	Y	
6E	110	U	

Keycodes for the HP-HIL "Cooked" Keyboard Driver (continued)

Keycodes in		Key Label	
hex	decimal	Unshifted	Shift
6F	111	I	
70	112	A	
71	113	S	
72	114	D	
73	115	F	
74	116	G	
75	117	H	
76	118	J	
77	119	M	
78	120	Z	
79	121	X	
7A	122	C	
7B	123	V	
7C	124	B	
7D	125	N	
7E	126	Unused	
7F	127	Unused	

Communicating with HP-HIL Devices

This section provides sample programs which can be used to identify and describe all of the HP-HIL devices supported by HP-UX and it explains the data read by the programs.

In order to use the programs in this section , you need to type them in using the *vi* editor or HP-UX editor of your choice (**the line numbers are not part of the program**). Next, compile the programs using either the C Language compiler command *cc*, the FORTRAN compiler command *fc*, or the Pascal compiler command *pc*. Finally type the file name *a.out* and press the Return key. The programs will return the device status and 5 hexadecimal values of the Describe Record. They next wait for you to move the Two-Button Mouse around on your desk or type in something from the HP 46021A keyboard before returning any data. Note that this same process and programs may be used to test all the HP-HIL devices supported on the HP-UX operating system.

Sample C Language Program

The sample program covered in this section is used to open a link to the device located at link address 1, it requests that an Identify and Describe be performed on the device, reads some data, and then closes the link to the device. To communicate with an HP-HIL device from the C Language, these intrinsics are used: *open, close, read,* and *ioctl.*

```
#include <sys/hilioctl.h>
main()
{
   int fd, status, index, bytes_read, done;
   unsigned char describe[11], buffer[10]

   /*Open the device file for the first device on the link.*/

   fd = open("/dev/hil1", 0);

   for (index = 0; index < 12; index++)
   {
      describe[index] = ' ';
   }

   /*This ioctl system call requests a describe record from
     the device.  The describe record contains information
     describing the amount and types of data that can be
     returned by the device.*/

   status = ioctl(fd, HILID, &describe[0]);
   printf("status %d    \n",status);
   printf("describe record\n");

   for (index=0; index<12; index++)
   {
      printf("      %x\n", describe[index]);
   }
   printf("\n", ' ');

   /*Read at least 21 bytes of data from the device.*/

   done = 0;
   while (done < 21)
   {
      bytes_read = read(fd, buffer, 1);
      done += bytes_read;

      for (index = 0; index < bytes_read; index++)
      {
         printf("      %x\n", buffer[index]);
      }
   }
   close(fd);
}
```

The following is an explanation of the program:

Line 1 provides the *include* file (`sys/hilioctl.h`) which contains a list of macros that execute specific functions when used within a C program. The `HILID` macro (**line 21**) which is called in the above program executes the identify and describe function. Note that the macro `HILID` can be found in the file *hilioctl.h*. The path name for the *hilioctl.h* file is */usr/include/sys/hilioctl.h*. If you execute the *cat* or *more* command on this path name, you will receive a screen listing of this file's contents.

Lines 4 and 5 declare the variables to be used in the program.

Line 9 opens the file `/dev/hil1` for reading.

Lines 11 through **14** initialize the describe array.

Line 21 uses the system call `ioctl` to call the macro `HILID` which when executed returns a describe array in the argument `describe[0]`. A function return value is assigned to the variable `status`.

Line 22 prints the value for `status` and **Line 23** prints the column header for the `describe record` listing.

Lines 25 through **28** are a *for link* which list the contents of the `describe` array. The contents listed are the device ID, the describe record header, and the I/O descriptor byte.

Line 29 prints a blank and executes a carriage return.

Line 33 initializes the variable `done`.

Lines 34 through **43** are a *while link* which reads 21 bytes of data from the the device one byte at a time and then list this data on the standard output (CRT). The data returned is the number of bytes in each packet, a time stamp, and a poll record header and it's parameters.

Line 44 closes the file `/dev/hil1`.

The results of using this program to read data from a Two-Button Mouse are similar to:

```
status  0
describe record
      68
      12
      C2
      1E
      2
      0
      0
      0
      0
      0
      0
      0

      8
      2
      82
      D9
      D3
      2
      1
      0
      8
      2
      82
      D9
      DB
      2
      1
      1
      8
      2
      82
      D9
      DD
```

An explanation for these results can be found in the section of this article entitled, "Description of the Data Returned by the Programs".

Sample Pascal Program

The sample programs covered in this section are used to open a link to the device located at link address 1, they requests that an Identify and Describe be performed on the device, read some data, and then close the link to the device. To communicate with an HP-HIL device from Pascal, the HP-HIL device file is opened by executing a Pascal RESET. There is also the need to use the Pascal ALIAS directive to make function and procedure calls to the C Language commands: *ioctl* and *sprintf*. Other Pascal directives used are: $standard_level 'HP_MODCAL'$ and $type_coercion 'NONCOMPATIBLE'$ for the Series 500 Pascal program only and $sysprog on$ for the Series 200/300 program only.

The differences in the two programs provided in this section are in the directives used and the preappending of the underscore to Series 200/300 ALIAS command calls.

Series 200/300 Program Listing

```
$sysprog on$

program hildev (input, output);

const
   maxstr = 255;
   maxdes = 12;
   loopcount = 21;
   hil_id = 1074554883;
   format = '     %X'#0;

type
   anystr = packed array [1..maxstr] of char;
   des_array = packed array[ 1..maxdes ] of char;

var
   device_f                  : file of char;
   buf                       : char;
   describe                  : des_array;
   index, status             : integer;
   format_str                : packed array[ 1..8] of char;
   line                      : anystr;

function ioctl $alias '_ioctl'$ (fd, hilid : integer;
                                 var des : des_array): integer; external;

```

```
procedure sprintf $alias '_sprintf'$ (anyvar str, format : anystr;
                                     num : integer); external;

begin
  (*Open the device file for the first device on the link.*)

  reset (device_f, '/dev/hil1');
  for index := 1 to maxdes do describe[index] := chr(0);
  format_str := format;

  (*This ioctl system call requests a describe record from
    the device.  The describe record contains information
    describing the amount and types of data that can be
    returned by the device.*)

  status := ioctl(3, hil_id, describe);
  writeln ('status = ', status:2);
  writeln ('describe record');
  for index := 1 to maxdes do
    begin
      sprintf (line, format_str, ord(describe[index]));
      writeln (line);
    end;
  writeln;

  (*Read at least 21 bytes of data from the device.*)

  index := 0;
  while index < loopcount do
    begin
      read (device_f, buf);
      sprintf (line, format_str, ord(buf));
      writeln (line);
      index := index + 1;
    end;
  close(device_f);
end.
```

Series 500 Program Listing

```
$standard_level 'HP_MODCAL'$
$type_coercion 'NONCOMPATIBLE'$
program hildev (input, output);

const
   maxstr = 255;
   maxdes = 12;
   loopcount = 21;
   hil_id = 1074554883;
   format = '     %X'#0;

type
   anystr = packed array [1..maxstr] of char;
   des_array = packed array[ 1..maxdes ] of char;

var
   device_f                   : file of char;
   buf                        : char;
   describe                   : des_array;
   index, status              : integer;
   format_str                 : packed array[ 1..8] of char;
   line                       : anystr;

function ioctl $alias 'ioctl'$ (fd, hilid : integer;
                                var des : des_array): integer; external;

```

```
27 procedure sprintf $alias 'sprintf'$ (anyvar str, format : anystr;
28                                      num : integer); external;
29
30 begin
31   (*Open the device file for the first device on the link.*)
32
33   reset (device_f, '/dev/hil1');
34   for index := 1 to maxdes do describe[index] := chr(0);
35   format_str := format;
36
37   (*This ioctl system call requests a describe record from
38     the device.  The describe record contains information
39     describing the amount and types of data that can be
40     returned by the device.*)
41
42   status := ioctl(3, hil_id, describe);
43   writeln ('status = ', status:2);
44   writeln ('describe record');
45   for index := 1 to maxdes do
46     begin
47       sprintf (line, format_str, ord(describe[index]));
48       writeln (line);
49     end;
50   writeln;
51
52   (*Read at least 21 bytes of data from the device.*)
53
54   index := 0;
55   while index < loopcount do
56     begin
57       read (device_f, buf);
58       sprintf (line, format_str, ord(buf));
59       writeln (line);
60       index := index + 1;
61     end;
62   close(device_f);
63 end.
```

The following is an explanation of the program:

Line 1 $sysprog on$ is the Series 200 Pascal directive which allows you to use ANYVAR. The ANYVAR parameter specifier in a function or procedure relaxes type compatibility checking when the routine is called. The Series 500 directives which allow you to do this are $standard_level$ and $type_coercion$.

Lines 5 through **10** are the constants defined as follows:

- `maxstr` is the maximum string length for a packed array of characters.
- `maxdes` is the maximum string length for a packed array of characters containing the describe record.
- `loopcount` is the number of times the while link of **lines 55** through **61** is to be executed.
- `hil_id` is the decimal value of the HILID command. The decimal values for the other HP-HIL commands can be found in a table in the section, "Identify and Describe Command (HILID)". Note that it is only necessary to use these decimal values of the HP-HIL commands when using the FORTRAN and Pascal programming languages.
- `format` is the character string used to format the output of the *sprintf* command. Note that `#0` is the end of line character in Pascal.

Lines 13 through **14** are the type declarations for the program. They are defined as follows:

- `anystr` is a packed array of characters. This packed array of characters is used to declare the variables `line`, `str`, and `format`.
- `des_array` is the packed array of characters used to declare the variable `describe`.

Lines 17 through **22** are a list of the variables declared for this program. They are defined as follows:

- `device_f` is the file name assigned to the device file */dev/hil1*.
- `buf` is the character variable returned from reading the `device_f` file.
- `describe` is the packed array of characters which is assigned the describe record. The describe record is explained in the section, "Identify and Describe Command (HILID)".
- `index` is a link control variable.
- `status` is the variable which is assigned the status of the opened file `device_f` (`/dev/hil1`) after executing the HILID command.
- `format_str` is the packed array of characters assigned the value of the constant called `format ('    %X'#0)`.
- `line` is the packed array of characters that is assigned the value of the character string that is returned when the *sprintf* command is executed.

Line 24 is a function which references the external HP-UX command *ioctl*. Note that the string parameter in the ALIAS directive has an under score preappended to it. This is only true for Series 200/300 computers.

Line 27 is a procedure which references the external HP-UX command *sprintf*. Note that the string parameter in the ALIAS directive has an under score preappended to it. This is only true for Series 200/300 computers.

Line 33 opens the device file */dev/hil1* and assigns it the name `device_f`.

Line 34 initializes the packed array `describe`.

Line 35 assigns the constant `format` to the packed array of characters `format_str`.

Line 42 executes the function `ioctl` and assigns its status value to the variable `status`.

Line 43 causes the variable `status` to be output to the display.

Line 44 outputs the describe record label.

Lines 45 through **49** are a for link which causes the describe record to be sent to the display or standard output (CRT).

Line 54 initializes the count variable `index` to zero.

Lines 55 through **61** are a while link which causes the data read from the device file `device_f` to be sent to the display (CRT).

Line 62 closes the device file `device_f`.

The results from using this program to read data from a Two-Button Mouse are similar to:

```
status  0
describe record
      68
      12
      C2
      1E
      2
      0
      0
      0
      0
      0
      0
      0

      8
      2
      82
      D9
      D3
      2
      1
      0
      8
      2
      82
      D9
      DB
      2
      1
      1
      8
      2
      82
      D9
      DD
```

An explanation for these results can be found in the section of this article entitled, "Description of the Data Returned by the Programs".

Sample FORTRAN Program

The sample program covered in this section is used to open a link to the device located at link address 1, it requests an identify and describe be performed on the device, reads some data, and then closes the link to the device. To communicate with an HP-HIL device from FORTRAN, the HP-HIL device file is opened by executing a FORTRAN OPEN statement. There is also the need to use the FORTRAN ALIAS directive to make calls to the C Language commands: *ioctl* and *read*.

The program is as follows:

```
 1        program hildev
 2 $alias ioctl='ioctl'(%val,%val,%ref)
 3 $alias read='read'(%val,%ref,%val)
 4        integer fd, istatus, index, HILID, count
 5        parameter (HILID = 1074554883)
 6        character*12 describe
 7        character*1 ch1
 8        open(unit=10,file="/dev/hil1")       ! open for first hil device
 9        fd=fnum(10)                          ! get the file descriptor
10 C                                           ! for unit 10 (/dev/hil1)
11        count = 0                            ! initialize count variable
12        do index = 1, 12                     ! initialize character array
13          describe(index:index) = ' '        ! describe
14        end do
15        istatus=ioctl(fd,HILID,describe)     ! check the status
16        print*,"status = ",istatus           ! output the status
17        print*,"describe record"             ! label describe record
18        do 110 index = 1,12
19          write(6,100) describe(index:index)
20 100      format(' ',6X,Z2.1)                ! output the describe
21 110    continue                             ! record
22        write(6,'()')
23
24        do while (count .LT. 21)             ! read 21 bytes of data
25          call read (fd, ch1, 1)             ! from the device
26          write(6,'(7X,Z2.1)') ch1
27          count = count + 1
28        end do
29        end
```

The following is an explanation of the program:

Line 1 is the `program` statement which defines the the name of the program. The program name for this program is `hildev`.

Lines 2 and **3** are ALIAS directives used to assign internal function names to the external HP-UX commands *ioctl* and *read*.

Line 4 declares the variables contained in it to be integers. These variables are defined as follows:

- `fd` is file descriptor associated with unit number 10.
- `istatus` is the variable assigned the status value returned by the `ioctl` command.
- `index` is the control variable for the DO link in **lines 18** through **21** of the program.
- `HILID` is the variable associated with the Identify and Describe Command. This variable is assigned the value shown in the PARAMETER statement of **line 5**. The decimal code values for the various HP-HIL commands can be found in the section of this article entitled, "Identify and Describe Command (HILID)".
- `count` is the count variable for the DO WHILE link of **lines 23** through **27**.

Lines 6 and **7** are character variables defined as follows:

- `describe` is a string of characters twelve characters long. This variable is assigned the describe record when the `ioctl` function is called.
- `ch1` is a one character string. This variable is assigned the data received when a call to the `read` command is made in the DO WHILE link of **lines 23** through **27**.

Line 8 opens the device file */dev/hil1*.

Line 9 assigns the value of unit number 10 to `fd`.

Line 11 initializes the count variable to zero.

Lines 12 through **14** form a DO link which initializes each character of `describe` to blank.

Line 15 assigns the value of the `ioctl` function call to the variable `istatus`.

Line 16 displays the value assigned to `istatus` on the standard output (CRT).

Line 17 displays the describe record label on the standard output.

Lines 18 through **21** display the contents of the character string `describe` on the standard output.

Lines 24 through **28** read the data from the device file */dev/hil1* and display it on the standard output.

The results from using this program to read data from a Two-Button Mouse are similar to:

```
status  0
describe record
      68
      12
      C2
      1E
      2
       0
       0
       0
       0
       0
       0
       0

       8
       2
      82
      D9
      D3
       2
       1
       0
       8
       2
      82
      D9
      DB
       2
       1
       1
       8
       2
      82
      D9
      DD
```

An explanation for these results can be found in the section of this article entitled, "Description of the Data Returned by the Programs".

Description of the Data Returned by the Programs

This section of the article provides you with an interpretation of the data obtained from running any one of the previously discussed programs. To understand how to interpret the Identify and Describe Command data returned by these programs, read the section of this article entitled, "Identify and Describe Command (HILID)".

If you used the previously discussed programs to identify and describe an HP 46060A (Two-button Mouse), the first set of data returned to you is created by lines 21 through 29 of the C language program, 41 through 48 of the Pascal program (42 through 49 for the Series 500 Pascal program) , and 15 through 21 of the FORTRAN program. The data looks similar to the following:

```
status    0
describe record
      68
      12
      c2
      1e
      2
      0
      0
      0
      0
      0
      0
      0
```

where the status returned is 0 which indicates that you can communicate with the HP-HIL device. If you receive a -1, it means you are not able to communicate with the HP-HIL device. The remaining data created by the *for link* of lines 25 through 28 is explained as follows:

68 is the device ID for an HP 46060A (Two-button Mouse).

12 is the describe record header which supplies some of the parameters of the device and provides an indication of how much additional information is to follow this parameter. The 8 bit character string for the hexadecimal value 12 is: 00010010. Reading this bit string from right to left you find that bit 0 is 0 and bit 1 is 1. This says that the header uses 16 bits to describe the resolution of the device, and axes X and Y will be reported. Bit 4 is set because the I/O descriptor byte is to appear later on in the Describe Record.

c2 is the lower-byte resolution.

1e is the higher-byte resolution. To determine the total counts per meter multiply the higher byte by 256 and add the result to the lower byte read above. Your total counts per meter reading should be 7 874.

2 is the I/O descriptor byte. The 2 reading indicates the buttons for which the device reports keycodes. The data here indicates that keycodes are reported for buttons one and two.

Note that the zeros following the I/O descriptor byte (2) do not provide any information. In the case of the FORTRAN sample program blanks were returned. These zeros and blanks fill in the data locations of the remaining seven empty bytes of data which may be obtained from the Identify and Describe Command depending on the device file you are reading. The Identify and Describe Command can return up to 12 bytes of data. These 12 bytes of data are as follows:

Device ID byte
Describe Record Header
Number of counts per centimeter (meter) low byte
Number of counts per centimeter (meter) high byte
Maximum count of X-axis low byte
Maximum count of X-axis high byte
Maximum count of Y-axis low byte
Maximum count of Y-axis high byte
Maximum count of Z-axis low byte
Maximum count of Z-axis high byte
I/O descriptor byte
Identify and describe command

The interpretation of these bytes of data can be found in the section, "Identify and Describe Command" in this article.

The second set of data received from the previously mentioned program looks similar to the following display. Note that leading zeroes of the two digit hexadecimal values have been omitted.

```
8
2
82
D9
D3
2
1
0
8
2
82
D9
DB
2
1
1
8
82
D9
DD
```

where:

8 is the number of bytes contained in the packet which was read including the length byte. Eight bytes of data were read in this packet.

2 82 D9 D3 is a time stamp in tens of milliseconds since power-up. The time stamp since power-up for this packet of data is 42 129 875 milliseconds.

2 is the poll record header. This header indicates to the System the type and quantity of information to follow, as well as reporting simple status information. The 8 bit character string for the hexadecimal value of 02 is: 00000010. This shows bit 0 is 0 and bit 1 is 1, which indicates that the coordinate axes the device is reporting are: X and Y.

1 is the X coordinate relative position of 1. Note that this reading is how much the X coordinate has moved since the last poll.

0 is the Y coordinate relative position of zero (0). Note that this reading is how much the Y coordinate has moved since the last poll variable.

8 is the number of bytes in the second packet of data sent. Eight bytes of data were read in this packet.

2 82 D9 DB is a time stamp in tens of milliseconds since power-up. The time stamp since power-up for this packet of data is 42 129 883 milliseconds.

2 is the poll record header. This header indicates to the System the type and quantity of information to follow, as well as it reports simple status information. The 8 bit character string for the hexadecimal value of 02 is: 00000010. This shows bit 0 is 0 and bit 1 is 1, which indicates that the coordinate axes the device is reporting are: X and Y.

0 is the X coordinate position of one (1).

1 is the Y coordinate position of one (1).

8 is the number of bytes in the second packet of data sent. Eight bytes of data were read in this packet.

2 82 D9 DD is a time stamp in tens of milliseconds since power-up. The time stamp since power-up for this packet of data is 42 129 885 milliseconds.

The last three bytes of data that were not shown in the previous data listing were truncated by the program. These bytes of data if read would return the Poll Record Header, X-axis coordinate position and the Y-axis coordinate position.

If you were using an HP 46021A (Keyboard) instead of an HP 46060A (Two-button Mouse), then the second set of data read from the device would look similar to the following display. Note that leading zeroes of the two digit hexadecimal values have been omitted. It should also be noted that the describe record header returned for the HP 46021A is the hexadecimal value `DF`, which indicates that it is an Extended Keyboard. The trailing zeros again should be ignored, as they do not provide any information.

```
7
A2
7D
37
B6
40
F2
7
A2
7D
37
BA
40
F3
7
A2
7D
49
20
40
F2
```

where:

`7` is the number of bytes contained in the packet which was read. Seven bytes of data were read in this packet.

`A2 7D 37 B6` is a time stamp in tens of milliseconds since power-up. The time stamp since power-up for this packet of data is 2 726 115 254 milliseconds (A2 7D 37 B6 hexadecimal = 2 726 115 254 decimal).

`40` is the poll record header. This header indicates to the system the type and quantity of information to follow, as well as reporting simple status information. The 8 bit character string for the hexadecimal value of `40` is: 0100 0000. Bit 6 is 1 (rightmost bit is bit 0), which indicates that the information read is up to 8 bytes and that the data produced can be interpreted by using Keycode Set 1 (Keycode Set 1 is provided at the end of this article).

`f2` is the space bar key which has been pressed down.

`7` is the number of bytes contained in the second packet which was read. Seven bytes of data were read in this packet.

`A2 7D 37 BA` is a time stamp in tens of milliseconds since power-up. The time stamp since power-up for this packet of data is 2 726 115 258 milliseconds (A2 7D 37 BA hexadecimal = 2 726 115 258 decimal).

`40` is the poll record header. This header indicates to the system the type and quantity of information to follow, as well as reporting simple status information. The 8 bit character string for the hexadecimal value of `40` is: 0100 0000. Bit 6 is 1 (rightmost bit is bit 0), which indicates that the information read is up to 8 bytes and that the data produced can be interpreted by using Keycode Set 1 (Keycode Set 1 is provided at the end of this article).

`f3` is the space bar key which has been released.

`7` is the number of bytes contained in the packet which was read. Seven bytes of data were read in this packet.

`A2 7D 49 20` is a time stamp in tens of milliseconds since power-up. The time stamp since power-up for this packet of data is 2 726 119 712 milliseconds (A2 7D 49 20 hexadecimal = 2 726 119 712 decimal).

`40` is the poll record header. This header indicates to the system the type and quantity of information to follow, as well as reporting simple status information. The 8 bit character string for the hexadecimal value of `40` is: 0100 0000. Bit 6 is 1 (rightmost bit is bit 0), which indicates that the information read is up to 8 bytes and that the data produced can be interpreted by using Keycode Set 1 (Keycode Set 1 is provided at the end of this article).

`f2` is the **space bar** key which has been pressed down.

HP-HIL Commands

This section contains a descriptions of the various HP-HIL commands (opcodes), including their usage, effects, and format when transmitted and received. Note that the term **command** as it relates to opcode in this article is explained in the section, "A Few Terms". The commands covered in this section are used within the macros located in the file */usr/include/sys/hilioctl.h*. As stated earlier these macros are called by using them as parameters in an *ioctl* command. For example:

```
ioctl(fd, HILID, &describe[0])
```

The following is a table of the macros included in the file */usr/include/sys/hilioctl.h*. This table contains the hexadecimal commands (opcodes) for these macros, as well as a brief description of each macro. A detailed description of the commands is provided in the sections which follow this table.

HP-HIL Macros

Macro	Opcode (hexadecimal)	Description
HILID	03h	Identify and Describe
HILPST	05h	Perform Self Test
HILRR	06h	Read Register
HILWR	07h	Write Register
HILRN	30h	Report Name
HILRS	31h	Report Status
HILED	32h	Extended Describe
HILSC	33h	Report Security Code
HILDKR	3Dh	Disable Keyswitch Auto Repeat
HILER1	3Eh	Enable Keyswitch Auto Repeat, cursor key repeat rate = 1/30 second. This is based on a system poll rate of 1/60 second.
HILER2	3Fh	Enable Keyswitch Auto Repeat, cursor key repeat rate = 1/60 second. This is based on a system poll rate of 1/60 second.
HILP1..HILP7	40h..46h	Prompt 1 through Prompt 7
HILP	47h	Prompt (General Purpose)
HILA1..HILA7	48h..4Eh	Acknowledge 1 through Acknowledge 7
HILA	4Fh	Acknowledge (General Purpose)

The macro names on the previous page cannot be used directly as parameters for an *ioctl* system call from Pascal or FORTRAN. However, if you assign the decimal value of the code to a declared integer variable and then use it as a parameter to an *ioctl* command, you can use the macro within a Pascal or FORTRAN program. The following table provides you with a listing of these macros in their decimal form.

HP-HIL Macros and Their Decimal Equivalent

Macros	Decimal Equivalent
HILID	1074554883
HILPST	1073833989
HILRR	-1073649658
HILWR	-2147391481
HILRN	1074817072
HILRS	1074817073
HILED	1074817074
HILSC	1074817075
HILDKR	-2147456963
HILER1	-2147456962
HILER2	-2147456961
HILP1	-2147456960
HILP2	-2147456959
HILP3	-2147456958
HILP4	-2147456957
HILP5	-2147456956
HILP6	-2147456955
HILP7	-2147456954
HILP	-2147456953
HILA1	-2147456952
HILA2	-2147456951
HILA3	-2147456950
HILA4	-2147456949
HILA5	-2147456948
HILA6	-2147456947
HILA7	-2147456946
HILA	-2147456945

Identify and Describe Command (HILID)

This command is used to determine the type of devices that are connected to the HP-HIL link, as well as the characteristics of these devices. Each device responds to the Identify and Describe command with a series of data bytes which are referred to as the Identify and Describe Record.

The Identify and Describe Record varies in length from 2 to 11 bytes. This record contains the following data format:

- Device ID Byte,
- Describe Record Header,
- I/O Descriptor Byte.

The remainder of this section will cover how to interpret each segment of the Identify and Describe Record.

Device ID Byte — is used to identify the general class of device and its nationality in the case of a keyboard or keypad. Since the sample program in this appendix only returns a two digit hexadecimal value as a device ID, you need a table which can be used to interpret what this value means. The following is a table of HP-HIL devices and the hexadecimal values which correspond to these devices.

HP-HIL Device Identification Codes

Device Type	ID Range (hex)	Device Description
	E0...FFh	Standard Keyboard (85-87 keys)
Keyboard	C0...DFh	Extended Keyboard (107-109 keys)
	A0...BFh	Compressed Keyboard (91-93 keys)
	98...9Fh	Undefined
	90...97h	Graphics Tablet and Digitizer
Absolute	8C...8Fh	Touchscreen
Positioners	88...8Bh	Touch-pad
	80...87h	Undefined
	70...7Fh	Undefined
Relative	6C...6Fh	Undefined
Positioners	68...6Bh	Mouse
	60...67h	Generic Quadrature Devices (e.g. Control Dials, Quad Port, etc.)
	5C...5Fh	Barcode Reader
Character	50...5Bh	Undefined
Entry	40...4Fh	Undefined
	30...3Fh	32-button Module and ID Module
Other	2C...2Fh	Undefined
	20...2Bh	Undefined

If the HP-HIL device you are using with the sample program is a keyboard or keypad, then you can to use the following table to determine its nationality.

HP-HIL Keyboard Nationality Codes

Lower 5 bits of Device ID (hex)	Nationality of Keyboard or Keypad
00...02h	Undefined
03h	Swiss/French
04...06	Undefined
07h	Canadian/English
08...0Ah	Undefined
0Bh	Italian
0Ch	Undefined
0Dh	Dutch
0Eh	Swedish
0Fh	German
10...12h	Undefined
13h	Spanish
14h	Undefined
15h	Belgian (Flemish)
16h	Finnish
17h	United Kingdom
18h	French/Canadian
19h	Swiss/German
1Ah	Norwegian
1Bh	French
1Ch	Danish
1Dh	Katakana
1Eh	Latin American/Spanish
1Fh	United States

To use the above table, assume the hexadecimal ID number returned is `DF`. Use the "Device Identification Codes" table to determine the type of device being used. Reading down the second column of this table you find that the device corresponding to `DF` is an Extended Keyboard (107-109 keys). You next need to determine its nationality since it is a keyboard. To do this, use the "Keyboard Nationality Codes" table and the lower 5 bits of the hexadecimal value `DF`. In the case of this example the lower 5 bits are `11111` binary or `1f` hexadecimal. In the table `1F` corresponds to the United States; therefore, the keyboard is designed for use in the United States.

Describe Record Header — is used to supply additional information about the axes used by the device if the device is intended to return coordinates, provides information about the I/O Descriptor Byte, and gives information about additional commands used by the device. The Describe Record Header is the second hexadecimal value read by the sample program in this article and each of the 8 bits of this value represents an important piece of data. The following table can be used to interpret this data.

Description of Describe Record Header

Bit 7	Set if the device contains two independent sets of coordinate axes. An example of two independent sets of HP-HIL devices is a device with two joysticks connected to it. Each of the joysticks have their own set of coordinate axes. Note that currently joysticks are not supported on HP products. It is assumed, however, that both sets of coordinate axes share common characteristics as identified in the remainder of the record. Default (clear) indicates a maximum of one set of axes.
Bit 6	Set if the device is to return absolute positional data (unsigned integers). Default (clear) indicates relative data (2's complement).
Bit 5	Set i the device returns all positional information at 16-bits/axis. Default (clear) is 8-bits/axis.
Bit 4	Set if the I/O Descriptor Byte is to follow later in the Identify and Describe Record. Default (clear) indicates that the device has no buttons, no proximity detection, and no prompt/acknowledge functionality, with no I/O Descriptor Byte to follow.
Bit 3	Set if the device supports the Extended Describe command. This command is covered later on in this article. Default (clear) indicates that the Extended Describe command is not supported.
Bit 2	Set if the device supports the Report Security Code command. This command is covered later on in this article. Default (clear) indicates that the Report Security Code command is not supported.
Bits 1 and 0	Bits 1 and 0 indicate the coordinate axes the device reports. If nonzero, then following the header will be 16 bits or 2 bytes of data describing the resolution of the device in counts per centimeter if Bit 5 is set and counts per meter if Bit 5 is clear. If the device is an absolute positioner, there will be an additional 16 bits/axis detailing the extent of each coordinate axis. For example, if the HP-HIL device has X, Y, and Z axes then there will be a maximum count/axis given for the lower and higher bytes of each axis. This is true no matter if the data is being report in 8 bit or 16 bit format. To determine the number of axes used by an HP-HIL device, use the following table:

bit 1	bit 0	Axes Reported
0	0	None
0	1	X
1	0	X and Y
1	1	X, Y, and Z

You now have the information necessary for interpreting a Describe Record Header. An example using this Identify and Describe Record parameter can be found in the section of this article entitled, "Description of the Sample Program's Data".

I/O Descriptor Byte — is used to indicate the buttons the device reports keycodes for, whether the device has proximity detection, and what Prompt/Acknowledge functions, if any, are implemented in the device. Proximity detection is a way of determining whether the stylus is in contact with the X and Y axis sensing device. Note that Prompt and Acknowledge are treated as a set, and no device may indicate support of any particular Prompt or Acknowledge without also supporting its counterpart. If none of the above features are implemented, the I/O Descriptor Byte is not transmitted. The following is the description of this byte:

Description of Bits for I/O Descriptor Byte

Bit 7	Set if the device implements the general purpose Prompt and Acknowledge functions. Generally speaking, a Prompt is an audio or visual indication to you that the HP-UX system is ready for some form of input, and Acknowledge is an indication to you that the input has been received by the HP-UX system. Default (clear) implies these functions are not implemented.
Bits 6, 5, and 4	Indicate specific Prompt/Acknowledges (Prompt 1..7 and Acknowledge 1..7) implemented in the device. Default (clear) indicates none. Use the following table to determine Prompt/Acknowledges:

Bit 6	Bit 5	Bit 4	Prompt/Acks. Implemented
0	0	0	None
0	0	1	1
0	1	0	1 and 2
0	1	1	1, 2, and 3
1	0	0	1 .. 4
1	0	1	1 .. 5
1	1	0	1 .. 6
1	1	1	1 .. 7

Description of Bits for I/O Descriptor Byte (continued)

Bit 3 Set if the device reports the Proximity In/Out keycodes. Proximity In/Out keycodes describe the stylus or pointers position as it moves into or out of contact with an X and Y axis sensing device (e.g. digitizer or touchscreen). Default (clear) indicates no proximity detection. Proximity detection is a way of determining whether the stylus is in contact with the X and Y axis sensing device.

Bits 2, 1, and 0 Indicate the buttons for which the device reports keycodes. A button report table is given below.

Button Report

Bit 2	Bit 1	Bit 0	Buttons Reported
0	0	0	None
0	0	1	1
0	1	0	1 and 2
0	1	1	1, 2, and 3
1	0	0	1 .. 4
1	0	1	1 .. 5
1	1	0	1 .. 6
1	1	1	1 .. 7

You now have the information necessary for interpreting an I/O Descriptor Byte. An example using this Identify and Describe Record parameter can be found in the section of this article entitled, “Description of the Sample Program’s Data”.

In the description of the Sample Program found in this article, you should recall that the first *for link* returned the Identify and Describe Record data and the *while link* and its internal *for link* returned data read when using the device. To interpret the latter set of data, you need to know how to read the information included with the Poll Record Header. The Poll Record Header returns information on X, Y, and Z coordinate positions, and keycodes sets being used. The Poll Record Header bits are assigned as follows:

Description of the Poll Record Header

Bit 7 Set if the device is reporting data from the second set of coordinate axes. Default (clear) indicates data from set 1.

Bits 6 and 5 Based on the value of these bits, you can determine the data type. The following is a table for determining the data type:

Bit 6	Bit 5	Data Type
0	0	No additional data
0	1	Up to 8 bytes ASCII 8-bit data
1	0	Up to 8 bytes Keycode Set 1 data
1	1	Up to 8 bytes Keycode Set 2 data

Bit 4 Not implemented. Default is clear.

Bit 3 Set indicates request for status check. Default (clear) indicates status unchanged.

Bit 2 Set indicates device ready for data. Default (clear) indicates not ready for data transfer at this time.

Bits 1 and 0 Indicate the coordinate axes the device is reporting. The following table provides information for determining which axes are to be reported.

Bit 1	Bit 0	Axes Reported
0	0	None
0	1	X
1	0	X and Y
1	1	X, Y, and Z

The Poll Record Header is followed by device data. If the device indicated that it would report coordinate information as 16-bits/axis in the Describe Record Header, then for each axis reported there is a lower byte and then a higher byte coordinate. Otherwise, the higher byte is not transmitted. In general, the Poll Record format indicates the maximum data which can be reported. Note that most devices only transmit a subset each time.

Once the positional data has been given, the next set of data provided by the Poll Record is 8 bytes of Keycode Set 1 or ASCII (8-bit) information, as specified by the Poll Record Header Bits 5 and 6.

Perform Self Test (HILPST)

This command causes the addressed device(s) to perform a self test, returning one data byte indicating the results of the test.

The data byte is 00 (hexadecimal) to indicate a successful test, with non-zero values representing device-specific failures.

Read Register (HILRR)

The Read Register provides a means for interaction with more complex devices via HP-HIL, allowing for types of data transfer not generally supported by the HP-HIL devices.

A device indicates support of the Read Register command in the Extended Describe Record (covered later in this article), it also indicates the specific read registers contained in the device. To perform a register read, the system transmits a data frame containing the address of the register it is to read, followed by the Read Register command. The device, upon receiving the command, transmits the contents of the register indicated by the data frame. The following diagrams illustrate this process.

System sends:

0	Register Address

Device responds:

	Register Data

Devices which do not support the Read Register command discard all data frames.

Write Register (HILWR)

Write Register provides a means of setting the contents of individual registers in devices supporting this advanced feature, as well as providing the means for transmitting a large amount of data to a device at speeds approaching the links maximum of 6 500 bytes/second.

There are two forms of the Write Register command which may be used separately or mixed in the proper order to provide the maximum degree of flexibility. Devices indicate support of either of these two forms (or both) in the Extended Describe Record. To write to individual registers in a device, the system transmits the register address, followed by the intended contents of that register. This is called Write Register Type 1. Several register address/data pairs may be transmitted in sequence.

System sends:

0	Register Address
	Register Data
0	Register Address
	Register Data
0	Register Address
	Register Data
	. . .
0	Register Address
	Register Data

The system may also wish to write several bytes to a single register in sequence, utilizing Write Register Type 2. Type 2 transfers are sent to the device by setting the most significant bit in the byte containing the register address, followed by as many data frames as desired (up to the maximum Write Buffer Length of the device, reported in the Extended Describe Record). Although it is possible to write several bytes to the same register using Type 1 transfers in devices supporting both types (repeating the register address before each byte of data), the intention of the two types of data transfers is quite different, and use of Type 2 is recommended. Write Register Type 2 has the following format:

System Sends:

1	Register Address
	Register Data
	Register Data
	. . .
	Register Data

The two formats may also be mixed for devices supporting both Write Register Type 1 and Type 2. There is, however, the restriction that all writes to individual registers occur first, because once the device detects the most significant bit in the register address byte to be set it assumes all following data frames to be data for that register. The following diagram illustrates the mixing of the two Types.

System sends:

0	Register Address
	Register Data
0	Register Address
	Register Data
	. . .
1	Register Address
	Register Data
	Register Data
	. . .
	Register Data

Devices which do not support the Write Register command will discard all data frames.

Report Name (HILRN)

Report Name is used to request a string of up to 15 characters (8-bit ASCII) which would aid in describing the device to the user. Devices indicate support of the Report Name command in the Extended Describe Record.

Report Status (HILRS)

Report Status is used to extract device-specific status information from devices configured on the link. Devices indicate support of the Report Status command in the Extended Describe Record. Devices supporting the command respond with from 1 to 15 bytes of device-specific status information. Interpretation of the status bytes will necessarily depend upon the device in question.

Extended Describe (HILED)

Support of the Extended Describe command is indicated in the Describe Record header. It provides additional information concerning more advanced device features which may not be required for basic operation.

Devices supporting the Extended Describe command respond with a series of data types referred to as the Extended Describe Record. The record length may vary from 1 to 15 bytes (although only 5 bytes are currently defined). The Extended Describe Record has the following format:

	Extended Describe Record Header
0	Maximum Read Register Supported
0	Maximum Write Register Supported
	Maximum Write Buffer Length Low Byte
	Maximum Write Buffer Length High Byte
	. . .

Devices responding to the Extended Describe command return at least 1 byte of data, the Extended Describe Record Header. Devices supporting Read Register to Write Register will need to report additional information so that their capabilities may be more fully defined. The Extended Describe Record Header both supplies some of the parameters of the device and provides an indication of how much additional information is to follow. The meanings of the individual bits in the Header are as follows:

Description of Extended Describe Record Header

bit 7 and 6	Reserved for future use. Default will be clear.
bit 5	Set if the Report Status command is supported. Default (clear) indicates Report Status not supported.
bit 4	Set if the Report Name command is supported. Default (clear) indicated Report Name not supported.
bit 3	Reserved for future use. Default will be clear.
bit 2	Set if Read Register supported. If set, immediately following the Header is a byte indicating the registers supported for reading in the device. Default is clear, indicating Read Register not supported.
bits 1 and 0	Bit 1 and bit 0 indicate support of the Write Register command. If bit 1 is set, Write Register Type 2 is supported by the device. If bit 0 is set, Write Register Type 1 is supported. If both bits are set, then the device supports both Type 1 and Type 2. If either bit 1 or bit 0 is set, then following in the Record will be information indicating the registers supported for writing in the device. If bit 1 is set, then an additional 16 bits will be returned indicating the maximum number of data bytes which may be written to the device at a time using Write Register Type 2 without data loss.

If the device indicated support for the Read Register command in the Header, then following the Header is a byte indicating the read registers supported by the device. The maximum Read Register supported byte indicates the largest read register address supported. Note that it is assumed that all addresses less than this maximum are also supported. Thus a byte of 0Fh indicates that the device contains 16 read registers, addressed as read registers 0..15. HP-HIL protocol allows for devices containing up to 128 read registers, addressed as 0..127.

If Write Register (Type 1 or Type 2) support is indicated, then next is a byte indicating the write registers supported. The maximum Write Register supported byte indicates the largest write register address supported in the device. It is assumed that all addresses less than the maximum are also supported. Up to 128 write registers, addressed as 0..127, are supported in the HP-HIL protocol.

If Write Register Type 2 is supported, as indicated by bit 1 of the Extended Describe Record Header being set, then following the maximum Write Register supported byte is 16 bits of data indicating the maximum number of bytes which may be transmitted to the device in a Type 2 transfer without overflowing the device's internal buffer. This number, transmitted first low byte, then high byte, represents the buffer length of the device minus 1. Thus a device capable of buffering 1024 bytes of data would transmit a maximum Buffer Length Low Byte of FFh and a maximum Buffer Length High Byte of 03h.

Report Security Code (HILSC)

The Report Security Code command is used to extract a unique identifier from the device. Support of the command is indicated in the Describe Record Header. The security code is a series of from 1 to 15 bytes which uniquely identify the device in question. Similar in purpose to a serial number, it may also contain information related to user identity, network address, or other information which is unique to a particular user or environment.

The only data transmitted by the ID Module is in response to the Report Security Command and a self test command.

The Report Security command invokes a data response according to the HP-HIL specifications. The following information applies to any device that supports the report security command.

The data format consists of a one byte header and eight bytes of binary data. The eight data bytes are the packed product and serial numbers of the HP-HIL device. In the case where ID Module is an exchange module signified by a ten digit part number, the five digit prefix number remains the same and the product number letter is replaced by the least significant digit of the part number.

The product, exchange and serial number formats are:

Header	:	H	(1 byte header)
Product number is	:	DDDDDA	(5 digits and 1 ASCII character)
Exchange part number	:	DDDDDd	(5 digits and 1 ASCII character)
Serial number is	:	YYWW@NNNNN	(9 digits and 1 ASCII character)

where:

H is the data header.

DDDDD is the product number (e.g. 46084).

A is the product number alpha character.

d is the least significant numeric character of the exchange number.

YY is the year code (year less 60).

WW is the week code (0 to 51).

@ is the serial country of manufacturing code.

NNNNN is the serial suffix (0 to 99 999)

The header byte is transmitted before the eight data bytes. The header's purpose is to allow for other data formats, however none are currently implemented.

The five digits of the product or exchange part prefix number are converted to a two byte binary number and the high order bit of a third byte. The remaining lower seven bits of the third byte contain the ASCII character. In products where two alpha characters are used in the product number, only the first character is used in the data format. The order of the bytes have been arranged to transmit the least significant byte of the number first.

In a similar manner, the nine digits of the serial number are converted to a four byte binary number. The country code of manufacturing is in the last byte to be transmitted and is an ASCII character.

The Report Security data bytes are transmitted starting with byte 1 and going through byte 9. Bits are numbered starting with bit 0 at the right most position of the byte (least significant bit) and going through bit 7 (most significant bit), left most position.

The report security data format is:

Byte	Bit	Description
1	7 - 0	The first byte is the header containing the number 10 hexadecimal for the following format. The general scheme for the header is: Bits 7 - 4 are assigned as format variations, where format 1 is the only assignment. Bits 3 - 0 are undefined, but set to zero.
2 3 4	7 - 0 7 - 0 7	The second and third bytes and the 7th bit of the fourth byte represent the 5 digits of the product or exchange part number DDDDD in binary form. The least significant bit is bit 0 of byte two.
4	6 - 0	The least significant seven bits of byte four represent the product letter or the least significant digit of the exchange number numeric character. The character is the US ASCII 7 bit representation of the character.
5 6 7 8	7 - 0 7 - 0 7 - 0 5 - 0	The fifth, sixth, seventh bytes and the six least significant bits of byte eight represent the 9 digits of the serial number YYWWNNNNN in binary form, without the alpha character. The least significant bit is bit 0 of byte 5.
8	7 - 6	The two most significant bits of byte eight are reserved for future use and are set to zero.
9	6 - 0	The least significant seven bits of byte 9 represent the serial number letter. The character is the US ASCII 7 bit representation of the character.
9	7	The most significant bit of byte nine is reserved for future use and is set to zero.

Sample of Report Security Format for A Product Module

The following information is returned upon receiving a Report Security command for a Product Module. The data is based on the data format described in the last section. Byte 1 is the first byte sent from the module to the host.

The sample results given are based on the product number 46084A and serial number 2519A00001. The serial number corresponds with the year of 1985, week 19, and serial number suffix 00001. Note that by adding 60 to the above serial numbers first two digits you get the year 85.

Byte No.	Data (hex)	Description
1	10	Header
2	04	Part of product number 46084
3	B4	Part of product number 46084
4	41	Product letter "A" and part of product number 46084
5	61	Part of serial number
6	B0	Part of serial number
7	03	Part of serial number
8	0F	Part of serial number
9	41	Country of Manufacturing Code

Sample of Report Security Format for An Exchange Module

The following information is returned upon receiving a Report Security command for an Exchange Module. The data is based on the data format described in the section, "Report Security Code (HILSC)". Byte 1 is the first byte sent from the module to the host.

The sample results given are based on the exchange number 46084-69901 and serial number 2519A00001. The serial number corresponds with the year of 1985, week 19, and serial number suffix 00001. Note that by adding 60 to the above serial numbers first two digits you get the year 85.

Byte No.	Data (hex)	Description
1	10	Header
2	04	Part of product number 46084
3	B4	Part of product number 46084
4	31	US ASCII character "1" which is part of the product number 460841
5	61	Part of serial number
6	B0	Part of serial number
7	03	Part of serial number
8	0F	Part of serial number
9	41	Country of Manufacturing Code

Since the sample is an exchange module, the exchange part number transmitted is 460841. Byte 4 is the hexadecimal value of 31 which represent the US ASCII character "1". Note, the prefix number 46084 does not change from the sample of the product module and the character "1" is really an ASCII character. When the number is displayed, the character string "-6990" should be inserted into the part number at the appropriate place.

Sample Report Security Program

This program returns information upon receiving a Report Security command for a Product Module. The data is based on the data format described at the beginning of the section, "Report Security Code (HILSC)". Byte 1 is the first byte sent from the module to the host.

The sample results given are based on the product number 46084A and serial number 2519A00093. The serial number corresponds with the year of 1985, week 19, and serial number suffix 00093. Note that by adding 60 to the above serial numbers first two digits you get the year 85.

The sample program is as follows:

```
#include <sys/hilioctl.h>
#define BUF_SIZ      15
#define MAX_DEV      '7'

unsigned char   dev_name[] = { "/dev/hil1" };

main()
{
  extern int    errno;
  int           index, hil_fd, product_no, serial_no,
                serial_hi, serial_lo;
  unsigned char hil_info[BUF_SIZ], product_let, country;

  while (dev_name[8] <= MAX_DEV)           /* device search link */
  {
     printf("\n");
     if((hil_fd = open(dev_name,0)) > 0)
     {                                     /* open device file. */
       printf("%s -",dev_name);            /* print device file name. */
       if(ioctl(hil_fd,HILID,hil_info) < 0)
         printf("ioctl error %d\n",errno); /* invalid ioctl call. */
       else
       {
         for(index = 0; index < BUF_SIZ; printf(" %.2X",hil_info[index++]))
         ;                                 /* print id desc info. */
         printf("\n");
         if((hil_info[1] & ~(~04)) == 04)
         {                                 /* if dev reports sec code. */
           printf("     ID NUMBER: ");
           if(ioctl(hil_fd,HILSC,hil_info) < 0) /* return sec code. */
             printf("ioctl error %d\n",errno); /* invalid ioctl call. */
           else
           {
             for(index = 0; index < BUF_SIZ - 6; printf(" %.2X",
                 hil_info[index++]))       /* print security code. */
             ;
             printf("\n");
             product_no = hil_info[1] | hil_info [2] << 8 |
                          (hil_info[3] >> 7) << 16;
             product_let = hil_info[3] & 0x7f;
             serial_no = hil_info[4] | hil_info[5] << 8 |
                         hil_info[6] << 16 | (hil_info[7] & 0x3f) << 24;
             serial_hi = serial_no/100000;
             serial_lo = serial_no - serial_hi * 100000;
             country   = hil_info[8] & 0x7f;
             printf("     product number = %u%c",product_no,product_let);
             printf("     serial number = %u%c%.5u\n",
                          serial_hi,country,serial_lo);
49            }
50          }
51        }
52       close(hil_fd);
53      }
54      else printf("%s - open error: %d\n",dev_name,errno);
55                                            /* unable to open dev file. */
56      ++(dev_name[8]);                      /* select next device file. */
57    }
58 }
```

The following is an explanation of the program:

Line 1 provides the *include* file (`sys/hilioctl.h`) which contains a list of macros that execute specific functions when used within a C Language program. HILID and HILSC are two macros which are included in this file and are used in this program. The path name for the *hilioctl.h* file is */usr/include/sys/hilioctl.h*. If you execute the *cat* or *more* command on this path name, you will receive a screen listing of this file's contents.

Lines 2 and **3** define the constants BUF_SIZ and MAX_DEV. BUF_SIZ is assigned the value of 15 and MAX_DEV is assigned the value of 7.

Line 5 assigns the character string "`/dev/hil1`" to the unsigned character array `dev_name`.

Lines 9 through **12** declare the variables used in the program. These variables are defined as follows:

- `errno` is the external variable that receives the error number if there is an invalid *ioctl* call made.
- `index` is the control variable used in the *for loops* of **lines 25** and **35**.
- `hil_fd` is the file descriptor used in the *ioctl* parameter fields.
- `product_no` is the product number of the ID Module being used.
- `serial_no` is the serial number of the ID Module being used.
- `serial_hi` is the higher order set of bits being manipulated to interpret the serial number.
- `serial_lo` is the lower order set of bits being manipulated to interpret the serial number.
- `hil_info` is the unsigned character array which is assigned information return from the HP-HIL device.

- `product_let` is the product letter suffixed to the product number of the HP-HIL ID Module.
- `country` is the unsigned character which designates the country of the manufacturing code for the HP-HIL ID Module.

Line 14 is a HP-HIL device search link (while link) which tries to open an HP-HIL device file. If the device file can be opened it continues with the remainder of the while link; otherwise, it prints an error message (**line 54**) and increments the eighth element (**line 56**) of the character string `dev_name` and searches for a new device.

Line 17 is an IF construct which tries to open the file `dev_name`. If the file can be opened the variable `hil_id` is assigned the file descriptor returned from opening the file. If the `dev_name` can not be opened then an error message (**line 54**) is displayed and the eighth element of the character array `dev_name` is incremented by one (**line 56**). The while link then searches for the next HP-HIL device.

Line 19 prints the device file name.

Line 20 checks for a valid *ioctl* call. If it is valid the program continues from the `else` statement with the next set of instructions; otherwise, an error message is displayed (**line 21**).

Line 24 is a *for link* which displays the describe record information after the label created by *line 19*.

Line 27 test to see if the device is an ID Module if it is not an ID Module then nothing is printed and the *while link* searches for the next device. However, if the device is an ID Module then the program continues by making an *ioctl* call using the macro HILSC.

Line 29 displays the label `ID NUMBER`.

Line 30 test for a valid *ioctl* call. If the call made is invalid then an error message is displayed (**line 31**). If the call made was valid then the program continues.

Lines 34 and **35** are a *for link* which displays the hexadecimal value of the Security Code after the `ID NUMBER` label.

Lines 37 through **48** make up the remainder of the *if-then-else* statement. Their purpose is to unscramble the Security Code information by performing several bit manipulations on the data. The result of all this bit manipulation is a readable product number and serial number being displayed.

A sample display would appear as follows assuming that an ID Module is the first device in the HP-HIL link, an Extended Keyboard is the second device in the HP-HIL link, and a Two-button Mouse is the third device in the HP-HIL link. Note that the remaining devices shown in the display indicate an open error. This is because these devices did not exist in the HP-HIL link.

```
/dev/hil1 -34 04 00 00 00 00 00 00 00 00 00 00 00 00 00
     ID NUMBER: 10 04 B4 41 BD B0 03 0F 41
     product number = 46084A   serial number = 2519A00093

/dev/hil2 -DF 00 00 00 00 00 00 00 00 00 00 00 00 00 00

/dev/hil3 -68 12 C2 1E 02 00 00 00 00 00 00 00 00 00 00

/dev/hil4 - open error: 6

/dev/hil5 - open error: 6

/dev/hil6 - open error: 6

/dev/hil7 - open error: 6
```

The hexadecimal information displayed after each of the device file names shown above can be interpreted by reading the section in the article entitled, "Identify and Describe Command (HILID)".

Disable Keyswitch Auto-repeat (HILDKR)

This command is used to disable the "repeating keys" feature in the addressed device, reducing returned data to one report per keyswitch transition.

Enable Keyswitch Auto-repeat 1 and 2 (HILER1 and HILER2)

These two commands are used to enable the "repeating keys" feature in the addressed device (if the feature is supported). Generally keys repeat at the rate of one report every 1/30 of a second (based on a system poll rate of 1/60 of a second). Some keys, termed "Modifier" keys, will not repeat, while based on the opcode of the Enable Keyswitch Auto-repeat command the Cursor Keys (cursor left, right, up and down) will repeat at either 1/30 of a second interval (opcode 3Eh) or 1/60 of a second intervals (opcode 3Fh). Most keys "repeat" by generating repeated down transitions corresponding to the key position being repeated, although repeating cursor keys report a keycode of 02h.

Prompt 1 through Prompt 7 (HILP1 through HILP7)

These commands are used to provide an audio or visual stimulus to the user, perhaps indicating that the system is ready for a particular type of input. Although intended to be directly associated with Acknowledge 1 through Acknowledge 7 and Button 1 through Button 7, this association is not a requirement.

The Prompts supported by the device are indicated in the Describe Record, and all unsupported Prompts will be treated the same as other unsupported commands.

Prompt (HILP)

Intended as a general-purpose stimulus to the user, Prompt is not intended to be associated with a particular Button as are Prompt 1 through Prompt 7. A device indicates support of Prompt in the Describe Record.

Acknowledge 1 through Acknowledge 7 (HILA1 through HILA7)

These commands, though similar to the Prompt 1 through Prompt 7 commands, are intended to provide an audio or visual response to the user, and are generally directly associated with the corresponding Prompt and Button of the same number, although this is not a requirement. Unsupported Acknowledge commands are ignored. Since there is no explicit "Prompt Off" function provided, this functionality may be part of the Acknowledge definition for a particular device, if required.

Acknowledge (HILA)

This command is similar to Prompt, however, Acknowledge is not associated with any particular Button, but merely as a general purpose audio or visual response to the user.

Keycode Set 1

This section provides the table, "Keycode Set 1".

Keycode Set 1

Keycode for Transition (hex)		United States Legend	
Down	**Up**	**Unshifted**	**Shifted**
00h	01h	5	+ Char
02h	03h	<Repeat Cursor>[3]	
04h	05h	Extend Char (right)	
06h	07h	Extend Char (left)	
08h	09h	Shift (right)	
0Ah	0Bh	Shift (left)	
0Ch	0Dh	CTRL	
0Eh	0Fh	Break	Reset
10h	11h	4 (keypad)	
12h	13h	8 (keypad)	
14h	15h	5 (keypad)	
16h	17h	9 (keypad)	
18h	19h	6 (keypad)	
1Ah	1Bh	7 (keypad)	
1Ch	1Dh	, (keypad)	
1Eh	1Fh	Enter (keypad)	
20h	21h	1 (keypad)	
22h	23h	/ (keypad)	
24h	25h	2 (keypad)	
26h	27h	+ (keypad)	
28h	29h	3 (keypad)	
2Ah	2Bh	* (keypad)	
2Ch	2Dh	0 (keypad)	
2Eh	2Fh	- (keypad)	
30h	31h	B	
32h	33h	V	
34h	35h	C	
36h	37h	X	
38h	39h	Z	
3Ah	3Bh	<NOT LOADED>[4] (left)	
3Ch	3Dh	<NOT USED>	
3Eh	3Fh	ESC	

[3] Keycode 02h reserved for Cursor Repeat.

[4] NOT LOADED key positions are located next to the Extend Char keys, below the Shift keys, and are generally covered by non-operative filler keys.

Keycode Set 1 (continued)

Keycode for Transition (hex)		United States Legend	
Down	Up	Unshifted	Shifted
40h	41h	6	– Char
42h	43h	<blank/f10> (keypad)	
44h	45h	3 (keypad)	Prev
46h	47h	<blank/f11> (keypad)	
48h	49h	. (keypad)	
4Ah	4Bh	<blank/f9> (keypad)	
4Ch	4Dh	Tab >| (keypad)	|<
4Eh	4Fh	<blank/f12> (keypad)	
50h	51h	H	
52h	53h	G	
54h	55h	F	
56h	57h	D	
58h	59h	S	
5Ah	5Bh	A	
5Ch	5Dh	<NOT USED>	
5EH	5Fh	Caps	
60h	61h	U	
62h	63h	Y	
64h	65h	T	
66h	67h	R	
68h	69h	E	
6Ah	6Bh	W	
6Ch	6Dh	Q	
6Eh	6Fh	Tab >|	|<
70h	71h	7	&
72h	73h	6	^
74h	75h	5	%
76h	77h	4	$
78h	79h	3	#
7Ah	7Bh	2	@
7Ch	7Dh	1	!
7Eh	7Fh	‘	~

Keycode Set 1 (continued)

Keycode for Transition (hex)		United States Legend	
Down	Up	Unshifted	Shifted
80h	81h	<BUTTON 1>	
82h	83h	<BUTTON 2>	
84h	85h	<BUTTON 3>	
86h	87h	<BUTTON 4>	
88h	89h	<BUTTON 5>	
8Ah	8Bh	<BUTTON 6>	
8Ch	8Dh	<BUTTON 7>	
8Eh	8Fh	<PROXIMITY IN/OUT>	
90h	91h	Menu	
92h	93h	f4	
94h	95h	f3	
96h	97h	f2	
98h	99h	f1	
9Ah	9Bh	8	+ Line
9Ch	9Dh	Stop	
9Eh	9Fh	Enter	Print
A0h	A1h	System	User
A2h	A3h	f5	
A4h	A5h	f6	
A6h	A7h	f7	
A8h	A9h	f8	
AAh	ABh	9	– Line
ACh	ADh	Clear line	
AEh	AFh	Clear display	
B0h	B1h	8	*
B2h	B3h	9	(
B4h	B5h	0	)
B6h	B7h	-	_
B8h	B9h	=	+
BAh	BBh	Back space	
BCh	BDh	Insert line	
BEh	BFh	Delete line	

Keycode Set 1 (continued)

Keycode for Transition (hex)		United States Legend	
Down	Up	Unshifted	Shifted
C0h	C1h	I	
C2h	C3h	O	
C4h	C5h	P	
C6h	C7h	[	{
C8h	C9h	]	}
CAh	CBh	\	|
CCh	CDh	Insert char	
CEh	CFh	Delete char	
D0h	D1h	J	
D2h	D3h	K	
D4h	D5h	L	
D6h	D7h	;	:
D8h	D9h	’	“
DAh	DBh	Return	
DCh	DDh	<home cursor>	
DEh	DFh	Prev	
E0h	E1h	M	
E2h	E3h	,	<
E4h	E5h	.	>
E6h	E7h	/	?
E8h	E9h	<NOT USED>	
EAh	EBh	Select	
ECh	EDh	<NOT USED>	
EEh	EFh	Next	
F0h	F1h	N	
F2h	F3h	<space bar>	
F4h	F5h	.	
F6h	F7h	<NOT LOADED>[5] (right)	
F8h	F9h	<left cursor>	
FAh	FBh	<down cursor>	
FCh	FDh	<up cursor>	
FEh	FFh	<right cursor>	

[5] NOT LOADED key positions are located next to the Extend Char keys, below the Shift keys, and are generally covered by non-operative filler keys.

Index

a

Acknowledge 1 through Acknowledge 7 (HILA1 through HILA7) 61
Acknowledge (HILA) 61
ALIAS directive 21, 27, 29, 30
A-size Digitizer 6

b

Bar-Code Reader 7
B-size Digitizer 6

c

Cat 58
Cc 16
C Language 16, 17
C Language Program, Explanation 19
C Language Program, Sample 17
Close 17
Command 37
Command (opcode) 8
Communicating with HP-HIL Devices 16
Control Dials 5
Cooked keyboard 11

d

Data, Description of Sample Programs 32
Data frame 8
Describe Record Header 39, 42
Description of Sample Programs Data 32
Device ID 32, 40
Device ID Byte 39, 40
Device Identification Codes, HP-HIL 40
Directive, ALIAS 21
Disable Keyswitch Auto-repeat (HILDKR) 60
Driver, 8042 10

e

Enable Keyswitch Auto-repeat 1 and 2 (HILER1 and HILER2) 60
Exchange Module 56
Exchange Module, Report Security Format for An 56
Extended Describe (HILED) 50
Extended Describe Record 47, 48, 51
Extension Module 4

f

Fc 16
FORTRAN 16
FORTRAN Program, Explanation 29
FORTRAN Program, Sample 29
Four-Button Cursor 6
Function Box 6

h

HILA 37, 38, 61
HILA1..HILA7 .... 37, 38, 61
HILDKR 37, 38, 60
HILED 37, 38
HILER1 .... 37, 38, 60
HILER2 .... 37, 38, 60
HILID 19, 26, 30, 32, 37, 38, 39, 58
Hilkdb 10, 11
HILP 37, 38, 61
HILP1..HILP7 .... 37, 38, 61
HILPST 37, 38, 46
HILRN 37, 38, 50
HILRR 37, 38, 46
HILRS 37, 38, 50
HILSC 37, 38, 52, 58
HILWR 37, 38, 47
HP:
  150II 3
  150III 3
  2393 3
  2397 3
  35723A (HP-HIL/Touch Bezel) 4
  46021A 16

46021A (HP-HIL Keyboard) 4
46060A (Two-Button Mouse) 4, 32
46080A (Extension Module) 4
46081A (HP-HIL/Audio Extension) 5
46082A/B (HP-HIL/Audio Remote Extension) 5
46083A (Rotary Control Knob) 5
46084A (HP-HIL ID Module) 5
46085A (Control Dials) 5
46086A (Function Box) 6
46087A (A-size Digitizer) 6
46088A (B-size Digitizer) 6
46089A (Four-Button Cursor) 6
46094A (HP-HIL/Quadrature Port) 6
92916A (Bar-Code Reader) 7
98700H 3
9920 3

HP-HIL 1

HP-HIL:

Audio Extension 5
Audio Remote Extension 5
Commands 37
Cooked keyboard driver 11
Device Identification Codes 40
Devices 3
ID Module 5
Interface 1
Keyboard 4
Keyboard Nationality Codes 41
Macros 37
Macros and Their Decimal Equivalent 38
Peripheral processor 1
Quadrature Port 6
Touch Bezel 4

HP-HIL Devices, Communicating with 16

HP-HIL devices, Using 7

i

Identification Codes, HP-HIL Device 40

Identify and Describe Command (HILID) 39

ID Module 52

Include file 19, 58

In/Out Keycodes, Proximity 44
Integral Personal Computer 3
Ioctl 17, 21, 27, 29, 30, 37, 58, 59
I/O Descriptor Byte 33, 39, 43

k

Keyboard, Cooked 11
Keyboard Nationality Codes, HP-HIL 41
Keycode Set 1 .... 36, 62, 63, 64, 65
Keycodes for the HP-HIL "Cooked" Keyboard Driver 12, 13, 14, 15, 16
Keycodes, Proximity In/Out 44

l

Linkback mode 2

m

Macro 8
Macroinstruction 8
Macros, HP-HIL 37
Mknod 9, 10
More 58

n

Nationality Codes, HP-HIL Keyboard 41

o

Open 17

p

Pascal 16
Pascal Program, Explanation 25
Pascal Program, Sample 21
Path name 8
Pc 16
Perform Self Test (HILPST) 46
Poll Record Header 35, 45
Product Module 55
Product Module, Report Security Format for A 55
Prompt 1 through Prompt 7 (HILP1 through HILP7) 61

Prompt/Acknowledge function 43
Prompt (HILP) 61
Proximity detection 43
Proximity In/Out Keycodes 44

r

Read 17, 29, 30
Read Register (HILRR) 46
Report Name (HILRN) 50
Report Security Code (HILSC) 52
Report Security Data Format 54
Report Security Format for An Exchange Module, Sample 56
Report Security Format for A Product Module, Sample 55
Report Security Program, Sample 56
Report Status (HILRS) 50
RESET (Pascal) 21
Rotary Control Knob 5

s

Sample Programs Data, Description of 32
Security Data Format, Report 54
Select code 8
Special (device) files 7, 8
Special (device) files, Creating 9
Sprintf 21, 26, 27
STANDARD_LEVEL 21, 25
SYSPROG ON 21, 25

t

Two-Button Mouse 4, 16, 32
TYPE_COERCION 21, 25

v

Vectra Personal Computer 3

w

Write Register (HILWR) 47
Write Register Type 1 .... 48, 49, 52
Write Register Type 2 .... 48, 49, 52

MANUAL COMMENT CARD

HP-UX Concepts and Tutorials

Vol. 7: Facilities for Series 200, 300, and 500

for HP 9000 Computers

Manual Reorder No. 97089-90081

Name: ______________________________

Company: ______________________________

Address: ______________________________

____

Phone No: ______________

Please note the latest printing date from the Printing History (page ii) of this manual and any applicable update(s); so we know which material you are commenting on ______________.

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

HEWLETT
PACKARD

HP Part Number
97089-90081
Printed in U.S.A. 11/86

97089-90626
For Internal Use Only