THE WATZMAN/HUG H19 ROM

Heath Users' Group Part Number 885-4600

Heath Users' Group
Hilltop Road
St. Joseph, Michigan 49085

THE WATZMAN/HUG H19
FIRMWARE: HUG P/N 885-4600

The WATZMAN/HUG H19 is replacement software for the code and keyboard encoder ROMs in the H19, H19A, or Z19 terminal, or the Terminal Logic board in an H89, H89A, Z89, or Z90. It can greatly add to the capabilities and usefulness of your terminal. It is supplied as ROMS ready to install, and is also available as source code on disks (HUG P/N 885-1221[-37]).

FEATURES

The WATZMAN/HUG H19 adds several new features to the H19 terminal. These include:

** Greater baud rate selection. Rates from 75 to 38,400 may be selected by software or switch settings. Note: Some units may not operate correctly at 38,400 baud.

** On screen digital 24 hour clock. A digital time display is maintained on the 25th line that does not interfere with terminal operation or use of the 25th line. The clock is controlled by escape codes. It can be set and read, and the display can be turned on or off.

** Native mode keyboard operation. In this mode, each key produces a unique 8-bit code instead of the 2-code escape sequences produced by some keys.

** Added transmit features. In addition to the ability to transmit a page and transmit the 25th line, the WATZMAN/HUG H19 can transmit the current (cursor) line and transmit the character at the cursor. The single character transmit feature sends the character as an 8-bit code that allows the user to determine whether it is a graphic and/or reverse video character. In the other transmit modes, the proper escape sequences are included in the transmitted string if there are graphics or reverse video characters (this is an undocumented feature of the original H19 firmware).

** Software selectable blinking or steady cursor, in underline or block format.

** Software selectable half or full duplex operation. This makes the H19 more useful in certain remote communication applications.

** Software selectable handshaking. You can select either the normal X-on X-off handshaking or hardware handshaking via RTS (pin 4).

** Switch selectable 7 or 8 bit word length.

** Transparent mode. In this mode, each 8-bit code from 0 to 255 produces a unique visible character.

** Upward compatibility. The escape codes for all of the new features except 134.5 baud select, handshaking select, and transparent mode are in accordance with Heath/Zenith standards, so most of the features are compatible with similar features on newer Heath/Zenith products, including the ZT-1 and Z100 series computers.

INSTALLING THE NEW ROMS

WARNING: Unplug your H19 or H89 while changing ROMS.

Installation of the new ROMS may require changing some soldered wire jumpers on the Terminal Logic Board (TLB). If, after reading this section, you determine that you have to change any jumpers, you should remove the TLB. Otherwise, you may be able to change the ROMS with it in place, but if you have an H89, the CPU board will have to be removed.

The instructions below are divided into two sections: one for older models (H19, H89), and one for newer models (H19A, H89A). If you are not sure which kind you have, the newer models may have a metal shield fastened to the back of the TLB board (which must be removed if you have to change any jumpers), and will have a video driver circuit board mounted "piggy back" on the main video board on the bottom of the cabinet. On older models, the video circuitry is all on one board in the cabinet bottom.

At some of the jumper locations, the existing jumpers may be in the form of printed circuit traces connecting the holes instead of wire jumpers. The traces will have to be cut if any changes are to be made, but it is unlikely that you will have to change any of those jumpers.

The Keyboard ROM (885-4600A)

H19/H89 Instructions

A. Remove the IC at U430.

B. The following jumpers should be installed at JP30, JP31, and JP32. (NOTE: Some TLB's will not have JP31 or JP32.)

JP30 1-2, 4-5, 6-7
JP31 2-3
JP32 2-3

C. Insert the new ROM at U430.

H19A/H89A Instructions

A. Remove the IC at U445.

B. The following jumpers should be installed at JP11.

JP11 1-2, 4-5, 6-7

C. Insert the new ROM at U445.

The Code ROM (885-4600B)

H19/H89 Instructions

A. Remove the IC's at U422 and U423 (if any).

B. The jumpers at JP20, JP21, JP22, and JP23 should be installed as follows.

JP20 2-3
JP21 1-2, 4-5
JP22 1-2, 4-5, 6-7
JP23 all open

C. Insert the MYH19 ROM at U422.

H19A/H89A Instructions

A. Remove the IC's at U436 (if any) and U437.

B. The jumpers at JP2, JP3, JP4, and JP5 should be installed as follows.

JP2 all open
JP3 1-2, 4-5, 6-7
JP4 2-3
JP5 1-2, 4-5

C. Insert the MYH19 ROM at U437.

After you have installed the ROMS and made any jumper changes necessary,. replace the TLB if you removed it. If you have an H89, S401 on the TLB will be hidden when you replace your CPU board. Because of this, you should review the next section on S401 settings and make any required changes before you replace the CPU board.

SWITCH S401 SETTINGS

If you had S401 on the TLB set to the standard settings recommended in your H19 or H89 operation manual, and you wish continue standard operation, then you will not have to change S401. If you want to change the baud rate, parity, word size, or duplex mode, refer to the chart that follows.

S401 Configuration Chart

Functions	Select baud rate				Parity Bit Type		Word Size	Duplex Mode
Selection	0	1	2	3	4	5	6	7
Half Duplex	.	.	.	.	.	.	.	0
* Full Duplex	.	.	.	.	.	.	.	1
7-bit Word	.	.	.	.	.	.	1	.
* 8-bit Word	.	.	.	.	.	.	0	.
Even Parity	.	.	.	.	1	1	.	.
Odd Parity	.	.	.	.	1	0	.	.
* Parity Disabled	.	.	.	.	0	.	.	.
Baud Rate Settings								
75	0	0	0	0	.	.	.	.
110	1	0	0	0	.	.	.	.
150	0	1	0	0	.	.	.	.
300	1	1	0	0	.	.	.	.
600	0	0	1	0	.	.	.	.
1200	1	0	1	0	.	.	.	.
1800	0	1	1	0	.	.	.	.
2000	1	1	1	0	.	.	.	.
2400	0	0	0	1	.	.	.	.
3600	1	0	0	1	.	.	.	.
4800	0	1	0	1	.	.	.	.
7200	1	1	0	1	.	.	.	.
* 9600	0	0	1	1	.	.	.	.
19200	1	0	1	1	.	.	.	.
38400	0	1	1	1	.	.	.	.
134.5	1	1	1	1	.	.	.	.

0 = Switch Up
1 = Switch Down
. = Switch does not apply to this situation
* = Standard Configuration

USING THE NEW FEATURES

Here are instructions and examples that will help you understand and use some of the new features of the WATZMAN/HUG H19. A complete summary of escape codes is included following this section. (In this discussion, escape sequences are shown within the text with spaces between the characters for clarity. In actual use, there should be no spaces, as can be seen in the BASIC program examples. All programs are in Microsoft (TM) BASIC for CP/M (R), with any changes required for HDOS included.)

** The Clock

As soon as you turn on your H19 after installing the new ROMS, you can see the digital clock in the lower right corner of the screen. You can easily set the clock from the keyboard as follows. First, depress the Off Line key (it should be down), then press ESC and capital X (shift-X or just X with Caps Lock on). Then enter two digits for the hour, two for the minutes, and two for the seconds. You will see the time display on the screen change as you type in the digits. You may type a colon between hours and minutes, and between minutes and seconds, but it is not necessary. When you have typed in 6 digits, press RETURN to start the clock. You may want to set the time to a few seconds ahead of the actual time, then press RETURN when your watch or clock catches up to the displayed time.

A program can set the clock in the same way as it is done manually by sending ESC, X, 6 digits, and RETURN. The BASIC line

```
PRINT CHR$(27);"X120000"
```

would set the clock to 12 noon. You can set the clock to 000000 if you want to do elapsed timing. You can read the clock in a program by sending ESC e. You can also turn off the display (the clock keeps going internally) by sending ESC d, and turn it back on with ESC c. Below is a program in Microsoft BASIC that illustrates reading the clock in a program.

```
10 REM CLOCK DEMONSTRATION PROGRAM
20 WIDTH 255:PRINT CHR$(27)"E"CHR$(27)"x5";
30 IF INKEY$<>"" THEN PRINT CHR$(27)"y5": STOP
40 PRINT CHR$(27)"e";:REM  COMMAND TERMINAL TO SEND TIME
50 T$=INPUT$(7):REM  READ TIME
60 P$=" A.M."
70 H=VAL(LEFT$(T$,2)):M=VAL(MID$(T$,3,2)):S=VAL(RIGHT$(T$,3))
80 IF S1=S THEN 30
90 S1=S
100 IF H=12 THEN P$=" P.M."
110 IF H>12 THEN P$=" P.M.":H=H-12
120 PRINT CHR$(27)"Y+,";:IF S/2=INT(S/2) THEN PRINT CHR$(27)"p";
130 PRINT " THE TIME IS";H;"HOUR;:IF H<>1 THEN PRINT "S";
140 PRINT ",";M;"MINUTE";:IF M<>1 THEN PRINT "S";
150 PRINT ", AND";S;"SECOND";:IF S<>1 THEN PRINT "S";
160 PRINT P$" "CHR$(27)"q    ":PRINT:GOTO 30
```

If you are using HDOS, leave out line 30. That line allows the CP/M user to stop the program by typing any key. HDOS users can stop it with Control-C, and type PRINT CHR$(27)"y5" to restore the cursor.

Note: There are no ANSI codes for the clock.

** The Transmit Current Line Feature

If you send ESC ^ (ESC [1 p in ANSI), the line on which the cursor resides will be transmitted to your computer. If there are graphics or reverse video characters in the line, appropriate escape sequences will be sent. The following Microsoft BASIC program illustrates this.

```
10 PRINT "TRANSMIT CURRENT LINE DEMONSTRATION NO. 1":PRINT
20 E$=CHR$(27)
30 PRINT "THIS IS A TEST LINE "E$"Ffaad"E$"G";
40 PRINT E$"^";:L$=INPUT$(80):REM  TRANSMIT LINE AND READ IT
50 PRINT:PRINT L$:REM  PRINT TRANSMITTED LINE
```

When you run this program, you will see two lines containing the words "THIS IS A TEST LINE" followed by graphic characters. The second line is the one transmitted by the H19, and it contains the escape sequences necessary to reproduce the graphics. If you type PRINT L$ after running the program, you can see the line again.

One use of the Transmit Current Line feature is to dump the contents of the screen to a printer. The Transmit Page feature (included in the original H19) can also be used to do it, but it sends so many characters at once that it is impossible to write the screen dump program in a slow language such as BASIC. But with the Transmit Line feature, you can get the screen one line at a time. Here is a screen dump program in Microsoft BASIC.

```
10 REM  TRANSMIT CURRENT LINE DEMONSTRATION NO. 2
20 E$=CHR$(27)
30 PRINT E$"j";:REM  SAVE CURSOR
40 PRINT E$"H";:REM  HOME CURSOR
50 FOR I=1 TO 24:PRINT E$"^";:REM  TRANSMIT LINE
60 L$=INPUT$(80):REM  GET LINE
70 LPRINT L$:REM  PRINT LINE
80 IF I<24 THEN PRINT:REM  MOVE CURSOR DOWN
90 NEXT I:PRINT E$"k";:REM  RESTORE CURSOR
```

In HDOS, add line 5 OPEN "O",1,"LP:", line 100 CLOSE #1, and change the LPRINT in line 70 to PRINT #1,. This program can help you in developing other programs by allowing you to save portions of the programs on a printer (or in a disk file with HDOS) while you work. Just renumber the program to start at a high number, replace the REM line at the beginning with STOP, then MERGE it to the end of the program you are developing. Then, when you want to print what is on the screen, just GOTO the first line after

the STOP (enter GOTO nnn in the command mode). You can also use this program as a subroutine within another program for printing duplicates of information on the screen.

Note: You might think that if you are in the ANSI mode you could do a screen dump of graphics to an H25 printer and have it print the information as graphics (the H25 uses ANSI codes). Unfortunately, the H25 requires a terminator character (CR, line feed, reverse line feed, etc.) after an escape sequence, so it will not "see" the ones coming in the screen dump, and it will print the arguments following the escape characters along with the regular printable characters. It is actually best to use the Heath mode when dumping to an H25, because it will ignore all Heath codes in the dump.

** The Transmit Character Feature

When you send ESC _ (ESC [2 p in ANSI) to the terminal, it transmits whatever character is at the cursor. If the character is a graphic character, it is sent as a control character. For example, if the graphic character whose ASCII value is equal to the letter b is at the cursor, ESC _ would return the ASCII value of Control-B. If the character is in reverse video, the character is sent with the high bit set. The BASIC program below illustrates the Transmit Character feature.

```
10 PRINT "TRANSMIT CHARACTER AT CURSOR DEMONSTRATION":PRINT
20 PRINT "USE THE ARROW KEYS TO MOVE THE CURSOR TO THE CHARACTER"
30 PRINT "YOU WISH TO TRANSMIT, THEN PRESS THE '.' KEY TWICE."
40 E$=CHR$(27):PRINT E$"x6";:REM  SHIFT KEYPAD
50 PRINT:PRINT "TEST LINE "E$"Fih~r"E$"G"E$"pX"E$"q";
60 C$=INPUT$(2):IF C$=".." THEN 80
70 PRINT C$;:GOTO 50
80 PRINT E$CHR$(95);:REM  TRANSMIT CHARACTER
90 C$=INPUT$(1):C=ASC(C$):REM  GET CHARACTER
100 PRINT E$"E":PRINT "THE CHARACTER WAS ";
110 IF C>127 THEN PRINT E$"p";
120 IF (C AND 127) < 32 THEN PRINT E$"F";:C=C+96:C$=CHR$(C)
130 PRINT C$E$"q"E$"G"E$"y6":REM   CLEAN UP
```

When this program runs, it prints a test line containing graphic characters. You can use the arrow keys to move the cursor to any character on any line, and then when you press "." twice, the program will print the character that was where you left the cursor. If it is a graphics character, it will be printed as such. The program is also designed to print a reverse video character in reverse video, but it does not because the operating system, whether HDOS or CP/M, removes the 8th bit from incoming characters. The only way to get around that is to access the port directly. To do it in the above program, first change line 60 to

```
60 C$=INPUT$(2):IF C$=".." THEN 75
```

Then add line 75:

```
75 OUT &0351,0   (if you have an H89 or H8 with H8-4)

75 OUT &0373,&025  (if you have an H8 with H8-5)
```

Change line 90 to:

```
90 C=INP(&0350):C$=CHR$(C)  (use &0372 with H8-5)
```

And add line 140:

```
140 OUT &0351,1  (if H8-5 then OUT &0373,&027)
```

Now if you move the cursor over a reverse video character, the program will receive it correctly.

** The Native Keyboard and Transparent Modes

When you send ESC x ? to the terminal (ESC [> 15 h in ANSI), it enters the native keyboard mode. In this mode, each key produces a unique 8-bit code. The function keys produce the letters that would normally follow an escape, with the high bit set. For example, f1 produces S with the high bit set. The keypad keys produce control characters. The following BASIC program demonstrates the native keyboard mode.

```
10 PRINT "NATIVE MODE KEY TEST"
20 E$=CHR$(27):PRINT E$"x?":REM  SET NATIVE MODE
30 OUT &0351,0:REM  IF H8-5 USE OUT &0373,&025
40 GOSUB 80:IF C=ASC("X") THEN 70
50 PRINT "THE VALUE OF THAT KEY WAS";C;"OR ";HEX$(C);" IN HEX"
60 GOTO 40
70 OUT &0351,1:PRINT E$"y?":STOP:REM  USE &0373,&027 FOR H8-5
80 REM  INPUT A CHARACTER DIRECTLY FROM PORT
90 C=INP(&0350):REM  GET PREVIOUS CHARACTER  (USE &0372 FOR H8-5)
100 IF INP(&0350)=C THEN 100:REM  LOOK FOR CHANGE
110 C=INP(&0350):RETURN
```

This program will print the decimal and hex values produced when you type a key. Type X to exit the program.

When you send ESC x @ to the terminal (ESC [> 16 h in ANSI), it enters the transparent mode. In this mode, each 8-bit code from 0 to 255 produces a unique character on the screen. ONCE SET, THE TRANSPARENT MODE CAN ONLY BE CLEARED WITH A TERMINAL RESET (SHIFT-RESET). The terminal will not respond to escape sequences while in the transparent mode.

You can use the transparent mode in conjunction with the native keyboard (set native mode first) mode to test the keyboard with the terminal off line. Each key will produce a unique display on the screen. For example, the f1 key, which normally sends ESC S, will make a reverse video S on the screen. The keypad keys will produce graphic characters. You may want to set the terminal to

wrap at the end of lines (ESC v) before you start the test.

Below is a BASIC program that demonstrates the transparent mode. It sends all codes from 0 to 255 in sequence 5 times. If you are using an H8-5, change 232 in the program to 250. You will have to reset the terminal after running the program.

```
10 REM  TRANSPARENT MODE DEMONSTRATION
20 PRINT CHR$(27)"E"CHR$(27)"v"CHR$(27)"x@";
30 FOR K=1 TO 5
40 FOR I=0 TO 255:OUT 232,I:FOR J=1 TO 5:NEXT J:NEXT I
50 FOR I=1 TO 64:OUT 232,32:FOR J=1 TO 5:NEXT J:NEXT I
60 NEXT K
```

** Using Higher Baud Rates

The WATZMAN/HUG H19 can support 19,200 and 38,400 baud, but some systems may require minor hardware modifications to function properly at these higher rates, and some may not support 38,400 baud correctly after the mods. You should also know that Heath/Zenith CP/M 2.2.03 running on a 2 MHZ computer can only send characters to the terminal at a maximum rate of about 11,000 baud regardless of the terminal baud rate. This means that you may not notice an increase in screen speed at the higher baud rates in CP/M.

To test higher baud rates in CP/M, run CONFIGUR and answer N to "Standard System?". Then select option A and enter the new baud rate and the port number as usual. Then type Y to accept the new baud rate, and X to write it to memory (it will not be written to the disk, so the next time you boot up, the old baud rate will be set). Then go off line and enter the escape sequence to set up the new baud rate, and go back on line. Now you can test the new rate with graphic games, word processors, etc.

If your computer functions normally at the new baud rate, you may want to run CONFIGUR again and write it permanently to the disk (option Y instead of X), and set S401 to the new rate. If you have an H89, you should only set S401 for either 9600 or 19,200 baud, and if you select 19,200 baud, you will have to set S501-6 on the CPU board to 1.

To test 19,200 baud in HDOS, prepare a new system disk and boot up on it. Before you type spaces to set the disk's baud rate, go off line and enter the escape sequence for 19,200 baud. Then go on line and type spaces and continue with the boot procedure. HDOS does not normally support 38,400 baud, but you can patch an HDOS 2.0 system disk to run at that rate. It should be a system disk that has been booted up at least once at another baud rate. Using DUMP (HUG #885-1062) if it is a hard sector disk, or UDUMP (885-8004) for other disks, examine address 05 at track 0, sector 0. If the disk was previously operated at 9600 baud, the value there will be 0C. Patch it to 06 for 19,200 baud, or 03 for 38,400 baud. HDOS can send characters at a faster rate than CP/M, so you will notice more of an improvement in screen speed.

If your system does not work correctly at 38,400 baud, you may want to try running your TLB at 3 MHZ. To do it on an H19 or H89, remove the jumper at JP10 2-5 on the TLB (it may be a trace that you will have to cut) and install a jumper at JP10 1-5. On an H19A or H89A, there are no jumpers to select different processor speeds, but if you are capable of delicate electronic work, you can make the modification by cutting a trace and installing a wire. First locate U429 on the component side of the TLB board. You will see a trace going up from pin 1, and another trace to the right of it coming from under the IC and going on to U426, passing under C451. Cut this trace somewhere between U429 and U426. Then install an insulated wire jumper on the solder side of the board from U427 pin 13 to U426 pin 3. This completes the 3 MHZ modification.

After these modifications, a system may still not work properly at 38,400 baud. A good test is to list a long text file on the terminal. In CP/M, use

```
A>PIP CON:=d:FILENAME.TYP
```

instead of TYPE d:FILENAME.TYP so the file will be listed more rapidly. Watch carefully for any spurious characters (such as graphics characters where there shouldn't be any). If the display is acceptable to you, then you can leave your terminal at 38,400 baud.

ESCAPE CODES FOR THE WATZMAN/HUG H19

The following list is divided into two sections. The first section contains Heath/Zenith escape codes supported by the WATZMAN/HUG H19, and the second contains ANSI codes supported by it.

The first column in each list contains the character or characters that follow the escape character, and the second contains the function of the sequence. The symbol * indicates a sequence that is not part of the standard Heath/Zenith or ANSI codes.

Section 1. Heath/Zenith Escape Sequences

Sequence	Function
#	Transmit page
/K	Response to ESC Z (VT52 identify)
<	Enter ANSI mode
=	Enter alternate keypad mode
>	Exit alternate keypad mode
?<parameter>	Responses to keypad in alternate mode
?M	ENTER
?n	.
?p	0
?q	1
?r	2
?s	3
?t	4
?u	5
?v	6
?w	7
?x	8
?y	9
@	Enter insert character mode
A	Cursor up
B	Cursor down
C	Cursor right
D	Cursor left
E	Clear screen (except 25th line) SHIFT-ERASE
F	Enter graphics mode
G	Exit graphics mode
H	Move cursor to home position
I	Reverse index (reverse scroll)
J	Erase from cursor position to end of screen Response from ERASE key (unshifted)
K	Erase from cursor position to end of line
L	Insert a line at cursor position
M	Delete line at cursor position
N	Delete character at cursor
O	Exit insert character mode

	Function key responses
P	BLUE
Q	RED
R	GRAY (WHITE)
S	f1
T	f2
U	f3
V	f4
W	f5
XnnnnnnnRET	Set clock. nnnnnn = numbers
Y<line#><col#>	Cursor addressing. Line# = SP-8 Col# = SP-o
Z	Identify as VT52. Response: ESC./ K
[	Enter hold screen mode
\	Exit hold screen mode
]	Transmit 25th line
^	Transmit current (cursor) line
_	Transmit character at cursor
b	Erase from beginning of display to cursor
c	Enable clock display
d	Disable clock display
e	Send time to host
j	Save current cursor position
k	Restore current cursor position
l	Erase entire line
n	Cursor position report (ESC Y <line#><col#>)
o	Erase from beginning of line to cursor
p	Enter reverse video mode
q	Exit reverse video mode
r<parameter>	Select baud rate
r@	75 baud
rA	110 baud
rB	150 baud
rC	300 baud
rD	600 baud
rE	1200 baud
rF	1800 baud
rG	2000 baud
rH	2400 baud
rI	3600 baud
rJ	4800 baud
rK	7200 baud
rL	9600 baud
rM	19200 baud
rN	38400 baud
rO *	134.5 baud
t	Enter keypad shifted mode
u	Exit keypad shifted mode
v	Enter wrap at end of line mode
w	Exit wrap at end of line mode

x<parameter>	Heath/Zenith set modes
x1	Enable 25th line
x2	Disable keyboad click
x3	Enter hold screen mode
x4	Set block cursor
x5	Disable cursor
x6	Enter keypad shifted mode
x7	Enter keypad alternate mode
x8	Enable auto line feed on carriage return
x9	Enable auto carriage return on line feed
x;	Set non-blinking cursor
x= *	Set hardware handshaking
x>	Enable half duplex
x?	Enable native keyboard mode
x@ *	Enable transparent mode
y<parameter>	Heath/Zenith reset modes
y1	Disable 25th line
y2	Enable keyboard click
y3	Exit hold screen mode
y4	Set underline cursor
y5	Enable cursor
y6	Exit keypad shifted mode
y7	Exit keypad alternate mode
y8	Disable auto line feed on carriage return
y9	Disable auto carriage return on line feed
y;	Set blinking cursor
y= *	Set software handshaking
y>	Disable half duplex
y?	Disable native keyboard mode
z	Reset to power up configuration
(	Enable keyboard
)	Disable keyboard

Section 2. ANSI Escape Sequences

Secquence	Function
=	Enter keypad alternate mode
>	Exit keypad alternate mode
M	Reverse index (reverse scroll)
O<parameter>	Function key responses
OP	BLUE (DEC F1)
OQ	RED (DEC F2)
OR	GRAY (DEC F3)
OS	f1 (DEC F4)
OT	f2
OU	f3
OV	f4
OW	f5
O<parameter>	Responses to keypad in alternate mode
OM	ENTER
On	.
Op	0
Oq	1
Or	2
Os	3
Ot	4
Ou	5
Ov	6
Ow	7
Ox	8
Oy	9
[>(parameters)h	Set mode #1 Multiple parameters may be used, separated by semicolons.
[>1h	Enable 25th line
[>2h	Disable keyboard click
[>3h	Enter hold screen mode
[>4h	Set block cursor
[>5h	Disable cursor
[>6h	Enter keypad shifted mode
[>7h	Enter keypad alternate mode
[>8h	Enable auto line feed on carriage return
[>9h	Enable auto carriage return on line feed
[>11h	Set non-blinking cursor
[>13h	Set hardware handshaking
[>14h	Enable half duplex
[>15h	Enable native keyboard mode
[>16h	Enable transparent mode

[>(parameters)l	Reset mode #1 (l = small L)
[>1l	Disable 25th line
[>2l	Enable keyboard click
[>3l	Exit hold screen mode
[>4l	Set underline cursor
[>5l	Enable cursor
[>6l	Exit keypad shifted mode
[>7l	Exit keypad alternate mode
[>8l	Disable auto line feed on carriage return
[>9l	Disable auto carriage return on line feed
[>11l	Set blinking cursor
[>13l	Set software handshaking
[>14l	Disable half duplex
[>15l	Disable native keyboard mode
[?(parameters)h	Set mode #2
[?2h	Enter Heath/Zenith mode
[?7h	Enter wrap at end of line mode
[?(parameter)l	Reset mode #2 (l = small L)
[?7l	Exit wrap at end of line mode
	In the following sequences, the parameters are optional repetition factors.
[<parameter>A	Cursor up <n times>
[<parameter>B	Cursor down <n times>
[<parameter>C	Cursor forward <n times>
[<parameter>D	Cursor backward <n times>
[<line#>;<col#>H	Postition cursor
[H	Home cursor
[<parameters>J	Erase in display
[0J	Erase from cursor to end of screen
[1J	Erase from beginning of screen to cursor
[2J	Clear screen
[<parameters>K	Erase in line
[0K	Erase from cursor to end of line
[1K	Erase from beginning of line to cursor
[2K	Clear line
[<parameter>L	Insert a line at cursor <n times>
[<parameter>M	Delete line at cursor <n times>
[<parameter>P	Delete character at cursor <n times>
[<line#>;<col#>R	Response to cursor position report
[<line#>;<col#>f	Position cursor
[f	Home cursor
[<parameters>h	Set mode #3
[2h	Disable keyboard input
[4h	Enter insert character mode
[20h	Enable auto carriage return on line feed

[<parameters>l	Reset mode #3 (l = small L)
[2l	Enable keyboard input
[4l	Exit insert character mode
[20l	Disable auto carriage return on line feed
[<parameters>m	Select graphic rendition
[0m	Disable reverse video
[7m	Enable reverse video
[10m	Enter graphics mode
[11m	Exit graphics mode
[6n	Report cursor position
[<parameter>p	Transmit functions
[p	Transmit page
[0p	Transmit page
[1p	Transmit current line
[2p	Transmit character at cursor
[q *	Transmit 25th line
[<parameter>r *	Set baud rate
[0r	75 baud
[1r	110 baud
[2r	150 baud
[3r	300 baud
[4r	600 baud
[5r	1200 baud
[6r	1800 baud
[7r	2000 baud
[8r	2400 baud
[9r	3600 baud
[10r	4800 baud
[11r	7200 baud
[12r	9600 baud
[13r	19200 baud
[14r	38400 baud
[15r	134.5 baud
[s	Save cursor position
[u	Restore cursor position
[z	Reset to power-up configuration

RESPONSES FROM FUNCTION AND KEYPAD KEYS WHILE IN THE NATIVE KEYBOARD MODE

While the native keyboard mode is set, the function and keypad keys produce unique 8-bit codes instead of escape sequences. The codes are listed below in hex, octal, and decimal.

Key	Hex	Oct.	Dec.	Explanation
RED	D0	320	208	Letter P plus high bit
BLUE	D1	321	209	Letter Q plus high bit
GRAY	D2	322	210	Letter R plus high bit
f1	D3	323	211	Letter S plus high bit
f2	D4	324	212	Letter T plus high bit
f3	D5	325	213	Letter U plus high bit
f4	D6	326	214	Letter V plus high bit
f5	D7	327	215	Letter W plus high bit
ERASE	CA	312	202	Letter J plus high bit
SHIFT-ERASE	C5	305	197	Letter E plus high bit
Keypad keys unshifted				
0	80	200	128	Control-@ plus high bit
1	81	201	129	Control-A plus high bit
2	82	202	130	Control-B plus high bit
3	83	203	131	Control-C plus high bit
4	84	204	132	Control-D plus high bit
5	85	205	133	Control-E plus high bit
6	86	206	134	Control-F plus high bit
7	87	207	135	Control-G plus high bit
8	88	210	136	Control-H plus high bit
9	89	211	137	Control-I plus high bit
.	8A	212	138	Control-J plus high bit
ENTER	8B	213	139	Control-K plus high bit
Keypad keys shifted				
0	90	220	144	Control-P plus high bit
1	91	221	145	Control-Q plus high bit
2	92	222	146	Control-R plus high bit
3	93	223	147	Control-S plus high bit
4	94	224	148	Control-T plus high bit
5	95	225	149	Control-U plus high bit
6	96	226	150	Control-V plus high bit
7	97	227	151	Control-W plus high bit
8	98	230	152	Control-X plus high bit
9	99	231	153	Control-Y plus high bit
.	9A	232	154	Control-Z plus high bit
ENTER	9B	233	155	ESC plus high bit