AGIOS FUNCTION CALL REFERENCE

SECTION 8

This section is a reference to the Alpha Graphics Input/Output System (AGIOS) set of function calls on the HP 150.

AGIOS is a facility that lets you use system routines to perform tasks on the HP 150 keyboard and display. They let you perform text and graphics mode operations on your display, let you define and use softkeys (function keys), and let you perform all touch screen operations.

This manual contains information elsewhere pertinent to *using* the AGIOS function set. Section 5, System Software, provides an introduction to the Alpha Graphics I/O System and includes a simple example of an AGIOS function call. Section 7, Programming the HP 150, describes a general AGIOS caller function for the C language. This function provides a relatively easy way for C programs (and programs in other languages with suitable modifications to the *agios* function) to invoke any AGIOS function. Section 7 also includes many examples of AGIOS calls from the C language.

CONTENTS

SYNTAX USED IN THE AGIOS FUNCTION CALLS

Each AGIOS function call is explained in detail on the succeeding pages. In order to clarify the information that you need to supply, a standard notation is used. Parentheses and positional notation is used as follows:

PARM	Indicates a single byte parameter.
(PARM1,PARM2)	Indicates two single byte parameters with parm1 in the high byte and parm2 in the low byte.
(,PARM)	Indicates a single byte parameter in the low order byte of the word. The high byte is ignored.
(PARM,)	Indicates a single byte parameter in the high order byte of the word. The low byte is ignored.
(PARM)	Indicates a word (16 bit) parameter.
((PARM))	Indicates a double word parameter. Usually the first word is a data segment address and the second word is an offset address.

Where applicable, the AGIOS call name is followed by the corresponding escape sequence. For example, one entry later in this section is:

DEFINE TOUCH FIELD (ESC - z g)

This indicates that the escape sequence which corresponds to the Define Touch Field AGIOS call is ESC - z g.

BATCH FUNCTION CALL

A special function call is available which lets you execute a sequence of function calls automatically. Using this function call is especially convenient when you frequently perform the same set of AGIOS function calls.

To "batch" function calls, you set up the sequence of AGIOS function calls in a ***command buffer***. Then you issue the following batch function call using the command buffer as one of its parameters.

(0, 0)	Function code
(BUFFER LENGTH)	COMMAND BUFFER length (byte count).
((COMMAND BUFFER))	A pointer to the buffer containing the AGIOS function calls. The function calls are defined consecutively. Use the same parameter format as specified in this section for the individual function calls.

A batch call is aborted when any of the function calls in the batch causes an error condition. Additionally, you cannot nest batch function calls (include them in the batch).

Example:

This example clears the alpha display by homeing up first, then clearing the display. Refer to the "H" and "J" options of the "Execute Two Character Sequence" function call.

The command buffer looks like this:

Contents-->	*16*	*0*	*H*	*16*	*0*	*J*
Byte -->	*+0*	*+1*	*+2*	*+3*	*+4*	*+5*

The assembler routine that sets up the command buffer and executes a batch call might look like this:

```
CLS     PUSH DS                 ; Save DS on Stack
        POP  CMDSEG             ; And Store it in Batch Buffer
;
        MOV  AX,4403H           ; I/O Control Write
        MOV  BX,1               ; Console Handle
        MOV  CX,8               ; Batch Buffer Length
        MOV  DX,offset BATCH    ; Batch Command Buffer
        INT  21H
        RET
;
CMDBUF DB   16,0,'H',16,0,'J'
;
BATCH   DB   0,0                ; AGIOS Batch Command
BUFLEN  DW   6                  ; CMDBUF Len 6 Here
CMDOFF  DW   CMDBUF             ; CMDOFF equates CMDBUF
CMDSEG  DW   0                  ; Data Segment Dummy Value
```

VIDEO INTRINSICS

The video intrinsics are a set of functions that may be used to update the state of the display. With the exception of the Write Line function, all intrinsics operate on a pre-defined subset area of the 24 by 80 character display. All row and column values are relative to zero. The upper left corner of the display is (0,0) and the lower right corner is (23,79).

A null data buffer pointer (segment = 0FFFFH) will suppress the update operation for that data type. There is a one to one correspondence between the position of a data byte in its buffer and the character position that it will affect in the pre-defined update area, starting at the upper left corner, incrementing column position first and then row position.

The ASCII data consists of the 8 bit HP Standard ASCII character code. The character set data consists of character set code characters as follows:

CHARACTER CODE:	*CHARACTER SET SELECTED:*
@	*Normal Roman*
A	*Line Drawing*
B	*Bold Face Roman*
C	*Italic Roman*
D	*Math*
SPACE	*No Change*

The enhancement data consists of enhancement code characters as follows:

security off:	*@*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*
security on :	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*[*	**	*]*		
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*

(A space, 20H, indicates no change to the current state.)

DEFINE AREA

This function specifies the area to be operated upon by subsequent area update operations.

(0, 1)	Function Code.
(LR-ROW,LR-COL)	Defines the lower right corner of the area to be operated upon.
(UL-ROW,UL-COL)	Defines the upper left corner of the area to be operated upon.
((PREV-COORD))	A pointer to a buffer where the previous area coordinates are returned. The format of the returned data is the same as the two input coordinates. This buffer pointer may be null.

WRITE AREA

This function writes data into the pre-defined display area.

(0, 2)	Function Code
(DATA LENGTH)	Length of the data buffers.
((ENH POINTER))	A double word pointer to a buffer of enhancement code characters to be used to change the enhancement state of the update area.
((CHAR SET))	A double word pointer to a buffer of character set code characters to be used to change the character set state of the update area.
((ASCII POINTER))	A double word pointer to the buffer of ASCII data to be written into the update area.

CLEAR AREA

This function clears the most recently defined area to ASCII blanks (20H), with no enhancements set.

(0, 3)	Function Code

ENHANCE AREA

This function sets the enhancement state of the entire pre- defined display area to the specified enhancement.

(0, 4)	Function Code
(,ENHANCEMENT)	An enhancement code character.

READ AREA

This function reads data from the pre-defined display area.

(0, 5)	Function Code
(DATA LENGTH)	Length of the data buffers.
((ENH POINTER))	A double word pointer to the buffer that the enhancement data will be read into.
((CHAR SET))	A double word pointer to a buffer that the character set data will be read into.
((ASCII POINTER))	A double word pointer to the buffer that the ASCII data will be read into.

SHIFT AREA

This function shifts the data in the pre-defined display area. Enhancements and character set are shifted with the ASCII data. Data shifted off an edge of the update area is lost.

(0, 6)	Function Code
(DATA LENGTH)	Length of the data buffer.
((ENH POINTER))	A double word pointer to a buffer of enhancement codes to be used to enhance the remaining unshifted area.
((CHAR SET))	A double word pointer to a buffer of character set code characters to be used to change the character set state of the remaining unshifted area.
((ASCII POINTER))	A double word pointer to a buffer of ASCII data to be written into the remaining unshifted area.
(DIRECTION,DIST)	DIRECTION: 0 = up 1 = down 2 = left 3 = right DIST: The number of rows/columns to shift the current video data.

WRITE LINE

This function writes a single row (or part of a row) in the workspace. Unlike Write Area, this intrinsic ignores the area bounds set by Define Area. If the position and length of the data are defined such that the right workspace boundary is violated, that portion of the data exceeding the boundary is ignored. No line wrap occurs.

(0, 7)	Function Code
(WKSP-ROW,WKSP-COL)	Defines the workspace relative position at which the data will be written.
(DATA LENGTH)	Length of the data buffers.
((ENH POINTER))	A double word pointer to the buffer of enhancement data to be written at the designated position.
((CHAR SET))	A double word pointer to a buffer of character set code characters to be used to change the character set state of the update area.
((ASCII POINTER))	A double word pointer to the buffer of ASCII data to be written at the designated position.

APPLICATION SOFTKEYS

This section gives the AGIOS function calls that support Application Softkeys. You can access these softkeys by typing [Shift] [User System]. You can use ASCII and Extended Roman characters within Application Softkey labels.

UPDATE SOFTKEY LABEL

This function will update a softkey label and enhancement.

(0, 8)	Function Code
(,NUMBER)	Softkey Number (the softkey number is from 1 to 8 inclusive)
((DATA))	A double word pointer to the buffer of ASCII data to be written into the label area. (16 bytes.)
(,TOP ENH)	Enhancement code for the top half of the label.
(,BOT ENH)	Enhancement code for the bottom half of the label.

READ SOFTKEY LABEL

This function gets the softkey number specified by the caller and then returns the softkey label and the enhancement code characters in two buffers.

(0, 9)	Function Code
(,NUMBER)	Softkey Number
((DATA))	A double word pointer to the buffer which is for the ASCII data.
((ENHANCEMENTS))	A double word pointer to the buffer which is for the enhancement code characters.

DISPLAY SOFTKEY LABELS

This function displays the application softkey labels in the softkey window.

(0,11)	Function Code

CONTROL FUNCTIONS

EXECUTE TWO CHARACTER SEQUENCE (ESC CHAR)

(0,16) Function Code

OP-CHAR The character equivalent of a 2 character escape sequence. Any operation characters are valid except for those that return data. The following list defines some of the most common ones.

0	:	*Dump Alpha to Printer*	*U*	:	*Next Page*
1	:	*Set tab*	*V*	:	*Previous Page*
2	:	*Clear tab*	*W*	:	*Format Mode On*
3	:	*Clear all tabs*	*X*	:	*Format Mode Off*
4	:	*Set left margin*	*Y*	:	*Display Functions On*
5	:	*Set right margin*	*Z*	:	*Display Functions Off*
9	:	*Clear margins*	*[*	:	*Start Unprotected Field*
@	:	*Delay one second*	*]*	:	*End Protected Field*
A	:	*Cursor up*	*b*	:	*Enable Keyboard*
B	:	*Cursor down*	*c*	:	*Disable Keyboard*
C	:	*Cursor right*	*f*	:	*Modem Disconnect*
D	:	*Cursor left*	*g*	:	*Soft Reset Terminal*
E	:	*Reset terminal*	*h*	:	*Home Up*
F	:	*Home down*	*i*	:	*Back Tab*
G	:	*Return*	*j*	:	*Display Softkey Def Menu*
H	:	*Home up*	*k*	:	*Exit Softkey Def Menu*
I	:	*Tab*	*l*	:	*Memory Lock On*
J	:	*Clear display*	*m*	:	*Memory Lock Off*
K	:	*Clear line*	*p*	:	*Default [f1] Value*
L	:	*Insert Line*	*q*	:	*Default [f2] Value*
L	:	*Delete Line*	*r*	:	*Default [f3] Value*
P	:	*Delete Character w/o Wrap*	*s*	:	*Default [f4] Value*
Q	:	*Insert Character w/o Wrap*	*t*	:	*Default [f5] Value*
R	:	*Insert Character Off*	*u*	:	*Default [f6] Value*
S	:	*Roll Up*	*v*	:	*Default [f7] Value*
T	:	*Roll Down*	*w*	:	*Default [f8] Value*

POSITION CURSOR (ESC & a)

(0,17)	Function Code
MODE	*Bit 0:* *1 = window row address* *0 = workspace row address* *Bit 1:* *1 = relative row address* *0 = absolute row address* *Bit 2:* *1 = negative row address* *0 = positive row address* *Bit 3:* *1 = row address is valid* *0 = row address is not valid* *Bit 4:* *1 = window column address* *0 = workspace column address* *Bit 5:* *1 = relative column address* *0 = absolute column address* *Bit 6:* *1 = negative column address* *0 = positive column address* *Bit 7:* *1 = column address is valid* *0 = column address is not valid*
(COLUMN)	An unsigned integer
(ROW)	An unsigned integer

DEFINE ENHANCEMENTS (ESC & d)

(0,18)	Function Code
SEC	SEC: 1 = on, 0 = off.
ENH	ENH: the ESC &d code character (@..O)

CURSOR SENSE ABSOLUTE (ESC a)

(0,19)	Function Code
((BUFFER))	A pointer to a buffer where two words are returned. The first word is the column number in binary and the second word is the row number in binary.

CURSOR SENSE RELATIVE (ESC `)

(0,20)	Function Code
((BUFFER))	A pointer to a buffer where two words are returned. The first word is the column number number in binary and the second word is the row number in binary.

SET CURSOR TYPE

This function sets the alpha cursor type.

(0,21)	Function Code
(,TYPE)	Alpha cursor type: 0 = underscore 1 = inverse cell.

READ CURSOR TYPE

This function reads the alpha cursor type.

(0,22)	Function Code
((BUFFER))	A pointer to a word location where alpha cursor type data is stored.

READ TERMINAL CONFIGURATION

This function reads the current terminal configurations.

(0,24)	Function Code
((BUFFER))	A word pointer to the buffer where the current configuration is returned.

When this function is complete, BUFFER contains:

(,RRRRRTSP)	R = reserved bits, T = set if touch screen off, S = set if softkeys on, P = set if remote port 2.
(KEYBOARD LANGUAGE)	
(STRING LANGUAGE)	
(OP SYS DEVICE)	Bits 0-2: addr. 0-7. Bits 3-15: dev. 0 = HP-IB, 1 = Accsy.

TOUCH SCREEN FUNCTIONS

The touch features on the HP 150 can be programmed in a variety of ways. The two general types of touch operations are "Field" operations and "Row/Column" operations. These two types can be intermixed.

Several of the touch screen function calls in this section assume keycode mode. Refer to "Keycode Modes" in Section 7 for information on this mode.

FIELD OPERATIONS

There are four types of touch fields you can define. They are:

ASCII Fields:

This mode is very similar to the User-Definable Softkeys (see Section 4). A buffer of ASCII characters is associated with a touch field. A response string of 0 to 80 ASCII characters is obtained by consecutive keyboard input operations. The first input obtains the first ASCII byte, and the second input obtains the second ASCII byte, etc. The response string is generated when the field is touched and should be indistinguishable from the typing of the same string from the keyboard. Auto-repeat is performed.

Keycode Fields:

Keycode fields require that you be in keycode mode (see "Keycodes", Section 7). The two data words of the response string are treated as a keycode and a qualfier and are processed by the regular keyboard routines. The final response to touch depends on the state and mode of keyboard processing. Touch simulates typing on the keyboard. Releasing simulates releasing your finger from the key. Auto-repeat is performed.

Toggle Fields:

The touch field is defined as a toggle switch. Touching the area toggles the field on and off. Whenever the field is touched, sensing information is passed to the application. The sensing information consists of three data bytes. The data is obtained by three consecutive keyboard input operations. The qualifier word of each data byte returned to the application has the touch screen ID. The three data bytes of sensing information are:

01H - toggle on field report opcode
d1 - response string first byte
d2 - response string second byte

02H - toggle off field report opcode
d1 - response string first byte
d2 - response string second byte

Normal Fields:

This type of touch field senses touch and/or release. The sensing information consists of three data bytes. The data can be obtained by three consecutive keyboard input operations. The qualifier word of each data byte returned to the application has the touch screen ID. Auto-repeat is performed. The three data bytes of sensing information are:

05H - field touched report opcode
d1 - response string first byte
d2 - response string second byte

06H - field released report opcode
d1 - response string first byte
d2 - response string second byte

Touch fields can overlap. If they do, then the most recent definition for a character cell takes precedent.

ROW COLUMN OPERATIONS

This type of touch operation returns the row and column position when a touch occurs. The row and column position are returned byte-by-byte using the keyboard input function of the operating system. Three data bytes are returned. The qualifier word returned with each byte of data has the touch screen ID. The data bytes for row/column operations are:

03H - row column touch report opcode
row - touched row number in binary
col - touched column number in binary

The data bytes for release report of row/column are:

04H - row column release report opcode
row - touched row number in binary
col - touched column number in binary

DEFINE TOUCH FIELD (ESC - z g)

(0,32)	Function code
((STRING))	Pointer to response string. Points to 2 words for keycode field the first word is the qualifier and the second word is the keycode. Points to 2 bytes for toggle or normal field, 0 - 80 bytes for ASCII field.
(LENGTH)	Response string length
(ATTRIBUTE,MODE)	Touch ATTRIBUTE: 1= ASCII field 2= Keycode field 3= Toggle field 4= Normal field Reporting MODE: 1= Report when touched 2= Report when released 3= Report both touch and release
(ON-ENH,OFF-ENH)	Enhancements of the field for on and off state for toggle field. Also enhancements of the field when touched and released for normal, ASCII, and keycode fields.
(CURSOR,BEEP)	CURSOR: 0 = do not position cursor 1 = position cursor on touch BEEP: 0 = do not beep 1 = beep on touch
(LR-ROW,LR-COL)	Row and column of the lower right corner of the touch field.
(UL-ROW,UL-COL)	Row and column of the upper left corner of the touch field.

DEFINE SOFTKEY FIELD (ESC - z s)

This function defines one of the eight softkey label areas as a touch field. These fields when touched produce the same response as if the corresponding function key is typed. The default is all softkey touch fields are on.

(0,33)	Function code
(MODE,KEY)	KEY (Softkey number): 1-8 MODE: 1 = on, 0 = off.

DELETE TOUCH FIELD (ESC - z d)

Deletes the touch field with upper left corner at <row> <col>. Nothing happens if there is no touch field there. The row and column are screen relative coordinates.

(0,34)	Function code
(UL-ROW,UL-COL)	Row and column position of the field to be deleted. (0FFH,0FFH) deletes all fields.

TOUCH SCREEN RESET (ESC - z j)

Resets all fields to off.

(0,35)	Function code

SET TOUCH REPORTING MODES (ESC - z n)

This function determines if, and how, touch is reported to your application by the HP 150 terminal.

(0,36) Function code

(,SCREEN-MODE) Touch Field and Row/Col sensing:

- 0 - Disable reporting.
- 1 - Enable sensing for row/column position. Touch fields are inactive.
- 2 - Enable sensing for touch fields only. Row/column sensing is inactive.
- 3 - Enable sensing for both row/column and touch fields. Row/column sensing occurs for areas not defined as touch fields.
- 4 - Toggles touch screen on and off.
- 10-14 - Same as 0-4, but causes escape sequence reports to be sent. This form is used ONLY by the system parser.

(,TOUCH-MODE) Sense touch or touch-release: (used with Row/Col sensing only)

- 1 - Report on Touch
- 2 - Report on Release
- 3 - Report on both Touch and Release

KEYBOARD INTERCEPT

Keyboard Intercept functions let you gain more control over the use of the keyboard. Each of the keyboard keys can be individually set to normal processing or one of the special processing modes. It should be noted that the keycodes for [f1] through [f12] are valid only when the application softkey labels are displayed on the screen with AGIOS function call (0,11).

There is no explicit "get keycode and qualifier" function call. The standard operating system console input function returns the normal ASCII code and also keycodes. The qualifiers are also returned if keycode mode is on.

The qualifier word is composed of the following bit values:

Bit	Value
15-8	Input Device ID: 0C0H = keyboard 080H = touch screen 000H = terminal internal
7	Special key. If set, the data is a non-ASCII keycode.
6	Reserved.
5	Left extend char - set when down.
4	Right extend char - set when down.
3	Control - set when down.
2	Left shift - set when down.
1	Right shift - set when down.
0	Repeating key when set.

To properly use the key intercept functions, the application program should first put the operating system's console input device into raw mode. This will allow keycodes to be passed through without interpretation by the operating system. Keycode mode should then be turned on. Finally each of the keys on the keyboard can be set to the desired mode of operation. For more information see "Keyboard Interfacing" in Section 7 of this manual.

DEFINE KEY CHARACTERISTICS

This function lets you alter characteristics of any of the special keys. Specifically, you can:

- Process the key normally (Same as on HP 2623 terminal)
- Intercept the key and pass a keycode to the application for processing
- Ignore the key when it is pressed
- Beep when the key is pressed in combination with the above characteristics

(0,40) Function Code

(CHARACTERISTICS) Key Characteristics

Bit:	Action: 0	beep
1	intercept	
2	ignore	
3-15	reserved	

If bit 1 and 2 are both set, the key is treated as an intercept key. When both are zero, the key resumes normal functioning.

(KEYCODE) Keycode from table on previous page. A value of 0FEH will set all special keys to the specified characteristics.

GET KEY CHARACTERISTICS

This function returns the characteristics of a key to the caller.

(0,41) Function Code

((BUFFER)) Pointer to a buffer where the key's characteristics will be returned.

(KEYCODE) Keycode

PUT KEY

This function lets you specify direct the terminal to process the keycode normally. Use this function when you wish to 'process' a keypress which was read in when intercept mode was active. For normal ASCII keys, you would simply 'echo' the character in place of using this function.

(0,42) Function Code

(QUALIFIER) Qualifier

Bit: Interpretation:

15-8 = Input Device ID (0C0H = keyboard)
7 = Special key. Must be set.
6 = Reserved
5 = Left extend char, set when down
4 = Right extend char, set when down
3 = Control, set when down
2 = Left shift, set when down
1 = Right shift, set when down
0 = Not used

(KEYCODE) Keycode

KEYCODE ON/OFF

This function turns the keycode mode of the console device on and off. If keycode mode is off, each key hit on the keyboard returns one byte of data when the console input device is read. If keycode mode is on, each key press returns four bytes of data. The first two bytes form a word of qualifiers and the next two bytes form a word of key data. See '*Keycode Modes*' in Section 7 for more detail.

(0,43) Function Code

MODE Keycode mode:

1 = on
0 = off

KEYCODE STATUS

This function returns the on/off status of a keycode.

(0,44)	Function Code
((BUFFER))	A pointer to a byte location where the keycode on/off status is returned.

READ KEYPAD STATUS

This function returns the status of whether the extended keypad is in numeric or graphics mode.

(0,44)	Function Code
((BUFFER))	A pointer to a byte location where the keypad status is returned.

When this function is complete, BUFFER contains 0 if numeric mode is set, and 1 if graphics mode is set.

DISPLAY CONTROL (ESC * d)

CLEAR GRAPHICS MEMORY (ESC * d a)

This function clears graphics display memory to 0. The complete displayable graphics area of 512 x 390 dots are cleared.

(4, 1) Function Code

SET GRAPHICS MEMORY (ESC * d b)

This function sets graphics display memory to 1. The complete displayable graphics area of 512 x 390 dots are set.

(4, 2) Function Code

TURN ON GRAPHICS DISPLAY (ESC * d c)

This function turns on the graphics display. The data in graphics memory is not affected.

(4, 3) Function Code

TURN OFF GRAPHICS DISPLAY (ESC * d d)

This function turns off the graphics display. The data in graphics memory is not affected.

(4, 4) Function Code

TURN ON ALPHANUMERIC DISPLAY (ESC * d e)

This function turns on the alphanumeric display and alphanumeric cursor. The data in alphanumeric memory is not affected.

(4, 5) Function Code

TURN OFF ALPHANUMERIC DISPLAY (ESC * d f)

This function turns off the alphanumeric display. The data in alphanumeric memory is not affected.

(4, 6) Function Code

TURN ON GRAPHICS CURSOR (ESC * d k)

This function turns on the graphics cursor. The data in graphics memory is not affected.

(4, 7) Function Code

TURN OFF GRAPHICS CURSOR (ESC * d l)

This function turns off the graphics cursor. The data in graphics memory is not affected.

(4, 8) Function Code

TURN ON RUBBER BAND LINE (ESC * d m)

This function turns on the rubber band line and graphics cursor.

(4, 9) Function Code

TURN OFF RUBBER BAND LINE (ESC * d n)

This function turns off the rubber band line.

(4,10) Function Code

MOVE GRAPHICS CURSOR ABSOLUTE (ESC * d <x,y> o)

This function moves the graphics cursor to the specified location. The move occurs even if the cursor is turned off.

(4,11) Function Code

(X-COORD) The X coordinate of the new cursor position expressed as an absolute number in the range of plus and minus 16383.

(Y-COORD) The Y coordinate of the new cursor position expressed as an absolute number in the range of plus and minus 16383.

MOVE GRAPHICS CURSOR INCREMENTAL (ESC * d <x,y> p)

This function moves the graphics cursor to the specified location. The move occurs even if the cursor is turned off.

(4,12) Function Code

(X-COORD) The X coordinates of the new cursor position expressed as a number that is relative to the current cursor position. Its range extends from -32768 to +32767.

(Y-COORD) The Y coordinate of the new cursor position expressed as a number that is relative to the current cursor position. Its range extends from -32768 to +32767.

TURN ON ALPHANUMERIC CURSOR (ESC * d q)

This function turns on the alphanumeric cursor. The data in alphanumeric memory is not affected.

(4,13) Function Code

TURN OFF ALPHANUMERIC CURSOR (ESC * d r)

This function turns off the alphanumeric cursor. The data in alphanumeric memory is not affected.

(4,14) Function Code

TURN ON GRAPHICS TEXT MODE (ESC * d s)

This function turns on graphics text mode. Characters that normally go to the alphanumeric display will be drawn on the graphics display.

(4,15) Function Code

TURN OFF GRAPHICS TEXT MODE (ESC * d t)

This function turns off graphics text mode.

(4,16) Function Code

VECTOR DRAWING MODE (ESC * m)

SELECT DRAWING MODE (ESC * m <mode> a)

This function selects the vector drawing mode.

(4,17) Function Code

(MODE) Drawing Mode:

0 = Graphics memory not changed
1 = Clear mode
2 = Set mode
3 = Complement mode
4 = Jam mode

SELECT LINE TYPE (ESC * m <type> b)

This function selects the vector line type.

(4,18) Function Code

(TYPE) Line Type:

1 = _____
2 = User defined line pattern
3 = Current area fill pattern
4 = — - — - — - — - — - — -
5 = ____ ____ ____ ____ ____
6 = — — — — — — — —
7 = ...
8 = —·—·—·—·—·—·—·—·—·
9 = --- --- --- --- --- --- --- --- --- ---
10= __ _ _ __ _ - - — - - — - —
11= Point plot

DEFINE LINE PATTERN AND SCALE (ESC * m <pattern><scale> c)

This function defines a user line pattern and scale.

(4,19)	Function Code
(,PATTERN)	The line pattern expressed as an eight bit binary number.
(SCALE)	The line scale expressed as a binary number from 1 to 16.

DEFINE AREA FILL PATTERN (ESC * m <pattern> d)

This function defines a user area fill pattern of 8 by 8 screen dots.

(4,20)	Function Code
(,DATA ROW1) (,DATA ROW2) (,DATA ROW3) (,DATA ROW2) (,DATA ROW4) (,DATA ROW5) (,DATA ROW6) (,DATA ROW7) (,DATA ROW8)	You specify all eight rows of the 8 by 8 fill pattern. Each DATA-ROW is an eight-bit byte which defines a particular row of the pattern.

FILL RECTANGULAR AREA, ABSOLUTE (ESC * m <x1,y1,x2,y2> e)

This function fills a rectangular area with the selected line or area fill pattern. The rectangular region is defined by specifying the lower left and upper right coordinates.

(4,21)	Function Code
(LWR LEFT X-COORD)	Each coordinate is in the range of -16384 to +16383.
(LWR LEFT Y-COORD)	
(UPR RIGHT X-COORD)	
(UPR RIGHT Y-COORD)	

FILL RECTANGULAR AREA, RELOCATABLE (ESC * m <x1,y1,x2,y2> f)

This function fills a rectangular area with the selected line or area fill pattern. The rectangular region is defined by specifying the lower left and upper right coordinates.

(4,22)	Function Code
(LWR LEFT X-COORD)	Each value is in the range from -32768 to +32767.
(LWR LEFT Y-COORD)	
(UPR RIGHT X-COORD)	
(UPR RIGHT Y-COORD)	

SELECT POLYGONAL FILL PATTERN (ESC * m <pattern> g)

This function selects a pattern for polygonal and rectangular area fill.

(4,23) Function Code

(PATTERN) Area Fill Pattern:

1 = Solid fill pattern
2 = User-defined fill pattern
3-10 = Pre-defined fill pattern

SELECT BOUNDARY PEN (ESC * m <pen> h)

This function selects the pen to be used to draw the boundary of a filled polygon. The actual value is not significant in a black and white system. This function turns on boundary drawing with a solid line pattern. The drawing of each edge of the boundary can be individually controlled.

(4,24) Function Code

(PEN) Boundary Pen Number

NO POLYGON BOUNDARY (ESC * m h)

This function turns off drawing of boundary around a polygon.

(4,25) Function Code

SET RELOCATABLE ORIGIN (ESC * m <x,y> j)

This function sets the relocatable origin to the specified absolute location.

(4,26)	Function Code
(X-COORD)	The X coordinate is the new relocatable origin expressed as an absolute number in the range of -16384 to +16383.
(Y-COORD)	The Y coordinate is the new relocatable origin expressed as an absolute number in the range of -16384 to +16383.

SET RELOCATABLE ORIGIN TO PEN POSITION (ESC * m k)

This function sets the relocatable origin to the current pen position.

(4,27)	Function Code

SET RELOCATABLE ORIGIN TO CURSOR POSITION (ESC * m l)

This function sets the relocatable origin to the current cursor position.

(4,28)	Function Code

GRAPHICS TEXT (ESC *)

The HP 150 offers a comprehensive graphics character set in read-only memory (*ROM*). This standard character set is used by all graphics text operations. However, you do have the ability to create custom characters of your own design and to use these singly or to replace the entire built-in character set.

SET GRAPHICS TEXT SIZE (ESC * m <size> m)

This function sets the graphics text size. The vector lists that define the current character set are scaled using this text size.

(4,29)	Function Code
(X-SCALE)	The X coordinate scale factor for text characters. The format is a 16 bit number with the radix point between bits 7 and 8: Bits 1-8 = integer Bits 9-16 = fraction
(Y-SCALE)	The Y coordinate scale factor for text characters. The format is a 16 bit number with the radix point between bits 7 and 8: Bits 1-8 = integer Bits 9-16 = fraction

SET GRAPHICS TEXT ORIENTATION (ESC * m <orientation> n)

This function selects the graphics text orientation. This also changes the direction of line feed, carriage return, and backspace. The desired orientation is specified by a number defined as:

(4,30) Function Code

(ORIENTATION) Graphics Text Orientation:

1 = Normal
2 = Rotate 90 degrees counterclockwise
3 = Rotate 180 degrees counterclockwise
4 = Rotate 270 degrees counterclockwise

TURN ON TEXT SLANT (ESC * m o)

This function turns on the 26.57 degree slant of graphics text characters.

(4,31) Function Code

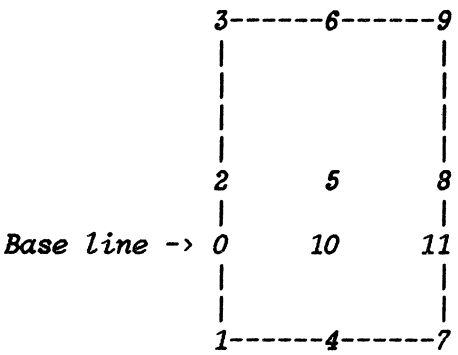
TURN OFF TEXT SLANT (ESC * m p)

This function turns off 26.57 degrees slant of graphics text characters.

(4,32) Function Code

SET GRAPHICS TEXT ORIGIN (ESC * m <0-9> q)

This function sets the graphics text origin to one of twelve positions of text justification. The positions are shown in this figure:



(4,33) Function Code

(ORIGIN) Graphics Text Origin:

A number from 0 to 11.

GRAPHICS TEXT LABEL (ESC * l <text>)

This function outputs a string of graphics characters.

(4,34) Function Code

((TEXT)) Segment and offset address of a string of characters. The string must be terminated by CR, LF, CR LF, or LF CR.

DEFINE USER CHARACTER SET

This function lets you re-define the entire graphics character set. All subsequent graphics text operations will use this character set. This includes text size, orientation, slant, and justification.

(4,35)	Function Code
((TABLE))	Segment and offset address of the table that points to the vector lists of characters.

SELECT DEFAULT CHARACTER SET

This function sets the character set to the default set maintained by the system. The cell size is 7 x 10.

(4,36)	Function Code

OUTPUT SINGLE TEXT CHARACTER

This function outputs a single graphics character defined by a vector list. All current graphics text operations such as size and orientation apply.

(4,37)	Function Code
((CHARACTER))	Segment and offset address of the vector list of a single character.

SET GRAPHICS DEFAULT (ESC * m r)

This function sets the graphics parameters to their default values.

(4,38) Function Code

The defaults affected by this call are:

- Pen down
- Line type 1
- User-defined line pattern solid
- User-defined area fill pattern solid
- Boundary pen off
- Drawing mode set
- Relocatable origin 0,0
- Text size 1
- Text origin 1
- Text slant off
- Text orientation 1
- Graphics text off
- Graphics display on
- Alphanumeric display on
- Graphics cursor off
- Alphanumeric cursor on
- Rubber band line off
- Graphics cursor position 0,0

SET PICTURE DEFINITION DEFAULTS (ESC * m 1 r)

This function sets the picture definition parameters to their default values.

(4,72)	Function Code
(RESET LEVEL)	The level of graphics reset. On the HP 150 the value '1' is the only supported level.

The picture defaults are:

Pen down
Line type 1
User-defined line pattern solid
User-defined area fill pattern solid
Boundary pen off
Drawing mode set
Text size 1
Text origin 1
Text slant off
Text orientation 1
Graphics text off

GRAPHICS HARD RESET (ESC * w r)

Sets the graphics parameters to their power on state.

(4,73)	Function Code

GRAPHICS PLOTTING (ESC * p)

LIFT PEN (ESC * p a)

This function lifts the pen.

(4,39)	Function Code

VECTOR MOVE (ESC * p a <x,y>)

This function lifts the pen and moves the pen to the new coordinate position. The pen is lowered at the end of the operation.

There are three ways to specify the new coordinate position:

(4,40)	Function Code
(X-COORD) (Y-COORD)	The X and Y numbers give an absolute coordinate position in the range from -16384 to +16383.
(4,41)	Function Code
(X-COORD) (Y-COORD)	The X and Y numbers give an incremental coordinate position in the range from -32768 to +32767.
(4,42)	Function Code
(X-coord) (Y-coord)	The X and Y numbers give a relocatable coordinate position in the range from -32768 to +32767.

LOWER PEN (ESC * p b)

This function lowers the pen.

(4,43)	Function Code

VECTOR DRAW (ESC * p b <x,y>)

This function lowers the pen and draws a vector to the new coordinate position. The pen is lowered at the end of the operation.

There are three ways to specify the new vector coordinates:

(4,44)	Function Code
(X-COORD) (Y-COORD)	The X and Y numbers give the absolute coordinates of the vector position. They are in the range from -16384 to +16383.
(4,45)	Function Code
(X-COORD) (Y-COORD)	The X and Y numbers give the incremental coordinates of the vector position. They are in the range from -32768 to +32767.
(4,46)	Function Code
(X-COORD) (Y-COORD)	The X and Y numbers give the relocatable coordinates of the vector position. They are in the range from -32768 to +32767.

PLOT TO CURSOR POSITION (ESC * p c)

This function moves the pen from its current position to the current cursor position if the pen is up. A draw is performed from the current pen position to the current cursor position if the pen is down.

(4,47) Function Code

POINT PLOT (ESC * p d)

This function draws a dot at the current pen position and then lifts the pen.

(4,48) Function Code

SET RELOCATABLE ORIGIN TO PEN POSITION (ESC * p e)

This function sets the relocatable origin to the current pen position.

(4,49) Function Code

START POLYGONAL AREA FILL (ESC * p s)

This function starts polygonal area fill. The boundary pen is lowered with this function.

(4,50) Function Code

TERMINATE POLYGONAL AREA FILL (ESC * p t)

This function terminates the polygon definition and fills the polygon.

(4,51) Function Code

POLYGON MOVE

This function closes the polygon defined up to this point and moves the pen to the new coordinate position to start a new polygon.

There are three ways to specify the new coordinate position:

(4,52) Function Code

(X-COORD)
(Y-COORD) The X and Y numbers give the absolute coordinates of the new position. They are in the range from -16384 to +16383.

(4,53) Function Code

(X-COORD)
(Y-COORD) The X and Y numbers give the incremental coordinates of the new position. They are in the range from -32767 to +32767.

(4,54) Function Code

(X-COORD) (Y-COORD)	The X and Y numbers give the relocatable coordinates of the new position. They are in the range from -32768 to +32767.

POLYGON DRAW

This function defines the edge of a polygon from the current pen position to the new coordinate position.

There are three ways that you can specify the new coordinate position:

(4,55)	Function Code
(X-COORD) (Y-COORD)	The X and Y numbers give the absolute coordinates of the new position. They are in the range from -16384 to +16383.
(4,56)	Function Code
(X-COORD) (Y-COORD)	The X and Y numbers give the incremental coordinates of the new position. They are in the range from -32768 to +32767.
(4,57)	Function Code
(X-COORD) (Y-COORD)	The X and Y numbers give the relocatable coordinates of the new position. They are the range from -32768 to +32767.

LIFT BOUNDARY PEN (ESC * p u)

This function lifts the polygon boundary pen. Undrawn edges of the polygon are not drawn. This remains in effect until the boundary pen is lowered.

(4,58)	Function Code

LOWER BOUNDARY PEN (ESC * p v)

This function lowers the polygon boundary pen. If a boundary pen has been specified, undrawn edges of the polygon are drawn with a solid line pattern. This remains in effect until the boundary pen is lifted.

(4,59) Function Code

GRAPHICS STATUS (ESC * s)

READ DEVICE ID (ESC * s 1)

This function returns the device id of the HP 150.

(4,60)	Function Code
((BUFFER))	Segment and offset address of the buffer to be used for returned device data.

When this function is complete, BUFFER contains an ASCII string that identifies the device.

READ PEN POSITION (ESC * s 2)

This function returns the current position and the state of the pen.

(4,61)	Function Code
((BUFFER))	Segment and offset address of the buffer to be used for returned pen status data.

When this function is complete, BUFFER contains:

(X-COORD) (Y-COORD)	The binary X and Y coordinates of the current pen position
(STATE)	0 = pen lifted, 1 = pen lowered

READ CURSOR POSITION (ESC * s 3)

This function returns the current position of the cursor.

(4,62)	Function Code
((BUFFER))	Segment and offset address of the buffer to be used for the returned cursor data.

When this function is complete, BUFFER contains:

(X-COORD) (Y-COORD)	The X and Y coordinates of current cursor position.

READ CURSOR POSITION, WAIT FOR KEY (ESC * s 4)

This function returns the current position of the cursor, but lets the user move it on the display first. The user can type any ASCII key or the SELECT key on the keyboard to move the cursor. As soon as one of these characters is typed, the cursor coordinates are returned to the program.

(4,63)	Function Code
((BUFFER))	Segment and offset address of the buffer to be used for returned cursor position.

When this function is complete, BUFFER contains:

(X-COORD) (Y-COORD)	The X and Y coordinates of current cursor position.
(CODE)	The character code of the key that was typed.

READ DISPLAY SIZE (ESC * s 5)

This function returns the number of displayable units and also the number of units in millimeters.

(4,64)	Function Code
((BUFFER))	Segment and offset address of the buffer to be used for the returned displayable size and unit data.

When this function is complete, BUFFER contains:

(X-LWR-LEFT) (Y-LWR-LEFT) (X-UPR-RIGHT) (Y-UPR-RIGHT)	The X and Y coordinates of the maximum display size.
(X-MM) (Y-MM)	The X and Y dimensions in dots / millimeters.

READ GRAPHICS SETTINGS (ESC * s 6)

This function returns information about the current graphics settings in effect.

(4,65)	Function Code
((BUFFER))	Segment and offset address of the buffer to be used for returned graphics settings.

When this function is complete, BUFFER contains the settings in consecutive words:

(CLEAR DISPLAY)
(NUMBER OF PENS)
(RESERVED)
(RESERVED)
(AREA SHADING)
(RESERVED)
(RESERVED)
(DYNAMIC MODIFICATION)
(GRAPHICS CHARACTER SIZE)
(GRAPHICS CHARACTER ANGLES)

(GRAPHICS CHARACTER SLANT)
(DOT-DASH LINE PATTERN)
(RESERVED)
(RESERVED)
(RESERVED)
(RESERVED)

READ GRAPHICS TEXT STATUS (ESC * s 7)

This function returns the current attributes of graphics text.

(4,66)	Function Code
((BUFFER))	Segment and offset address of the buffer to be used for the returned graphics attributes.

When this function is complete, BUFFER contains:

(X SIZE) (Y SIZE)	The character cell size.
(ORIGIN)	The text origin.
(ANGLE)	The text orientation.
(SLANT)	The character slant.

READ ZOOM STATUS (ESC * s 8)

This function returns the terminal's zoom setting.

(4,67)	Function Code
((BUFFER))	Segment and offset address of the buffer to be used for the returned zoom setting.

When this function is complete, BUFFER contains:

(ZOOM SIZE)	1 - 16 (the HP 150 always returns 1).
(ZOOM ON/OFF)	0 = off, 1 = on (the HP 150 always returns 0).

READ RELOCATABLE ORIGIN (ESC * s 9)

This function returns the current relocatable origin.

(4,68)	Function Code
((BUFFER))	Segment and offset address of the buffer to be used for the returned origin.

When this function is complete, BUFFER contains:

(X-COORD) (Y-COORD)	The X and Y coordinates of the current relocatable origin.

READ RESET STATUS (ESC * s 10)

This function returns information on whether the terminal has executed a full reset since the last time reset status was checked.

(4,69)	Function Code
((BUFFER))	Segment and offset address of the buffer to be used for the returned reset status.

When this function is complete, BUFFER contains:

(RESET STATUS)
(RESERVED)
(RESERVED)
(RESERVED)
(RESERVED)
(RESERVED)
(RESERVED)

(RESERVED)

READ AREA SHADING (ESC * s 11)

This function returns information on the area shading capability of the terminal.

(4,70)	Function Code
((BUFFER))	Segment and offset address of the buffer to be used for the returned shading data.

When this function is complete, BUFFER contains:

(CAPABILITIES)	The area shading capabilities.
(WIDTH) (HEIGHT)	The area shading pattern size.

READ DYNAMICS (ESC * s 12)

This function returns information on the terminal's dynamic graphics capabilities.

(4,71)	Function Code
((BUFFER))	Segment and offset address of the buffer to be used for the returned dynamic graphics data.

When this function is complete, BUFFER contains:

(SELECTIVE-ERASE-CAPABILITIES)

(COMPLEMENT-CAPABILITIES)

READ EXTENDED SCREEN DIMENSIONS

This function provides information about the alpha and graphics screen size plus the relationship between the two.

(0,74)	Function Code
((BUFFER))	Segment and offset address of the buffer to be used for the returned screen size data.

When this function is complete, BUFFER contains:

(X-PIXELS)	512 graphics display size in pixels.
(Y-PIXELS)	390
(ROWS)	27 alpha display size.
(COLUMNS)	80
(X-MM)	160 graphics display size in mm.
(Y-MM)	120
(ROW-MM)	150 alpha display size in mm.
(COL-MM)	116
(DELTA-X)	10 graphics origin minus alpha origin in mm.
(DELTA-Y)	4