	key-input(":0;export DISPLAY") key-input(\015) \n\
Alt ~Ctrl ~Shift<Key>osfCancel:	key-input("DISPLAY=") call-parser(10,"")\
	key-input(":0;export DISPLAY") key-input(\015) \n\
Alt ~Ctrl ~Shift<Key>F12:	key-input("setenv DISPLAY ") call-parser(10,"")\
	key-input(":0") key-input(\015)

```
    :<Key>F2:                      key-input("Ctrl-[OQ")\n        \
    :<Key>F3:                      key-input("Ctrl-[OR")\n        \
    :<Key>osfMenu:                 key-input("Ctrl-[OS")\n

/* Misc. Function Keys (editing keys) */
*term.vt220-auxTrans-miscFkeys:                                   \
    ~Shift <Key>Find:              key-input("Ctrl-[[1~")\n       \
    ~Shift <Key>osfInsert:         key-input("Ctrl-[[2~")\n       \
    ~Shift <Key>DRemove:           key-input("Ctrl-[[3~")\n       \
    ~Shift <Key>osfSelect:         key-input("Ctrl-[[4~")\n       \
    ~Shift <Key>osfPageUp:         key-input("Ctrl-[[5~")\n       \
    ~Shift <Key>osfPageDown:       key-input("Ctrl-[[6~")\n       \
    ~Shift <Key>osfBeginLine:      key-input()\n                  \
    ~Shift <Key>osfEndLine:        key-input()\n                  \
    ~Shift <Key>osfDelete:         key-input()\n

/* IBM -> VT Remap Misc. Function Keys (editing keys) */
*term.vt220-auxTrans-ibmMiscFkeys:                                \
    ~Shift <Key>osfInsert:         key-input("Ctrl-[[1~")\n       \
    ~Shift <Key>osfBeginLine:      key-input("Ctrl-[[2~")\n       \
    ~Shift <Key>osfPageUp:         key-input("Ctrl-[[3~")\n       \
    ~Shift <Key>osfDelete:         key-input("Ctrl-[[4~")\n       \
    ~Shift <Key>osfEndLine:        key-input("Ctrl-[[5~")\n       \
    ~Shift <Key>osfPageDown:       key-input("Ctrl-[[6~")\n

/* Redefine function keys for use with Telnet, to provide DISPLAY variable */
/* osfMenuBar is XK_F10, osfCancel is XK_Escape */

*term.vt220-auxTrans-telnetFkeys:  \
  Alt  ~Ctrl ~Shift<Key>osfMenuBar: key-input("set display/create
                                   /server=0/")\
                                   key-input("trans=tcpip/node=")
                                   call-parser(10,"")\ key-input(\015)
                                   \n\
  Alt  ~Ctrl ~Shift<Key>F11:       key-input("DISPLAY=") call
                                   -parser(10,"")\
```

```
    ~Shift <Key>osfDown:              key-input("Ctrl-[[B")\n          \
    ~Shift <Key>osfUp:                key-input("Ctrl-[[A")\n

/* Function Keys */
*term.vt220-auxTrans-fkeys:                                            \
    :<Key>osfHelp:                    key-input("Ctrl-[[11~")\n        \
    :<Key>F2:                         key-input("Ctrl-[[12~")\n        \
    :<Key>F3:                         key-input("Ctrl-[[13~")\n        \
    :<Key>osfMenu:                    key-input("Ctrl-[[14~")\n        \
    :<Key>F5:                         key-input("Ctrl-[[15~")\n        \
    :<Key>F6:                         key-input("Ctrl-[[17~")\n        \
    :<Key>F7:                         key-input("Ctrl-[[18~")\n        \
    :<Key>F8:                         key-input("Ctrl-[[19~")\n        \
    :<Key>F9:                         key-input("Ctrl-[[20~")\n        \
    :<Key>F10:                        key-input("Ctrl-[[21~")\n        \
    :<Key>F11:                        key-input("Ctrl-[[23~")\n        \
    :<Key>F12:                        key-input("Ctrl-[[24~")\n        \
    :<Key>F13:                        key-input("Ctrl-[[25~")\n        \
    :<Key>F14:                        key-input("Ctrl-[[26~")\n        \
    :<Key>F15:                        key-input("Ctrl-[[28~")\n        \
    :<Key>Help:                       key-input("Ctrl-[[28~")\n        \
    :<Key>F16:                        key-input("Ctrl-[[29~")\n        \
    :<Key>Menu:                       key-input("Ctrl-[[29~")\n        \
    :<Key>F17:                        key-input("Ctrl-[[31~")\n        \
    :<Key>F18:                        key-input("Ctrl-[[32~")\n        \
    :<Key>F19:                        key-input("Ctrl-[[33~")\n        \
    :<Key>F20:                        key-input("Ctrl-[[34~")\n        \
    :<Key>KP_F1:                      key-input("Ctrl-[OP")\n          \
    :<Key>KP_F2:                      key-input("Ctrl-[OQ")\n          \
    :<Key>KP_F3:                      key-input("Ctrl-[OR")\n          \
    :<Key>KP_F4:                      key-input("Ctrl-[OS")\n

/* IBM -> VT Remap Function Keys */
*term.vt220-auxTrans-ibmFkeys:                                         \
    :<Key>osfHelp:                    key-input("Ctrl-[OP")\n          \
```

```
    :<Key>KP_Separator:            key-input("Ctrl-[Ol")\n         \
Alt  <Key>KP_Add:                  key-input("Ctrl-[l")\n          \
    :<Key>osfActivate:             key-input("Ctrl-[OM")\n

/* Numeric Keypad */
*term.vt220-auxTrans-numKeypad:                                    \
    /* Group 1 */                                                  \
    :<Key>KP_0:                    key-input()\n                   \
    :<Key>KP_1:                    key-input()\n                   \
    :<Key>KP_2:                    key-input()\n                   \
    :<Key>KP_3:                    key-input()\n                   \
    :<Key>KP_4:                    key-input()\n                   \
    :<Key>KP_5:                    key-input()\n                   \
    :<Key>KP_6:                    key-input()\n                   \
    :<Key>KP_7:                    key-input()\n                   \
    :<Key>KP_8:                    key-input()\n                   \
    :<Key>KP_9:                    key-input()\n                   \
    /* Group 2 */                                                  \
    :<Key>KP_Decimal:               key-input()\n                  \
    :<Key>KP_Subtract:             key-input()\n                   \
    :<Key>KP_Separator:            key-input()\n                   \
Alt  <Key>KP_Add:                  key-input(",")\n                \
    :<Key>osfActivate:             key-input()\n

/* Application Cursor Keys */
*term.vt220-auxTrans-appCKeys:                                     \
    ~Shift <Key>osfLeft:           key-input("Ctrl-[OD")\n         \
    ~Shift <Key>osfRight:          key-input("Ctrl-[OC")\n         \
    ~Shift <Key>osfDown:           key-input("Ctrl-[OB")\n         \
    ~Shift <Key>osfUp:             key-input("Ctrl-[OA")\n

/* Normal Cursor Keys */
*term.vt220-auxTrans-normCKeys:                                    \
    ~Shift <Key>osfLeft:           key-input("Ctrl-[[D")\n         \
    ~Shift <Key>osfRight:          key-input("Ctrl-[[C")\n         \
```

```
/*  Up                          osfUp                   */
/*  Right                       osfRight                */
/*  Down                        osfDown                 */
/*  End                         osfEndLine              */
/*  Home                        osfBeginLine            */
/*  Prior                       osfPageUp               */
/*  Next                        osfPageDown             */
/*  BackSpace                   osfBackSpace            */
/*  Delete                      osfDelete               */
/*  Insert                      osfInsert               */
/*  F1                          osfHelp                 */
/*  F4                          osfMenu                 */
/*  Shift F8                    osfAddMode              */
/*  F10                         osfMenuBar              */
/*  Select                      osfSelect               */
/*  KP_Enter                    osfActivate             */
/*  Clear                       osfClear                */
/*  Undo                        osfUndo                 */

/* Application Keypad */
*term.vt220-auxTrans-appKeypad:                         \
    /* Group 1 */                                       \
    :<Key>KP_0:                 key-input("Ctrl-[Op")\n \
    :<Key>KP_1:                 key-input("Ctrl-[Oq")\n \
    :<Key>KP_2:                 key-input("Ctrl-[Or")\n \
    :<Key>KP_3:                 key-input("Ctrl-[Os")\n \
    :<Key>KP_4:                 key-input("Ctrl-[Ot")\n \
    :<Key>KP_5:                 key-input("Ctrl-[Ou")\n \
    :<Key>KP_6:                 key-input("Ctrl-[Ov")\n \
    :<Key>KP_7:                 key-input("Ctrl-[Ow")\n \
    :<Key>KP_8:                 key-input("Ctrl-[Ox")\n \
    :<Key>KP_9:                 key-input("Ctrl-[Oy")\n \
    /* Group 2 */                                       \
    :<Key>KP_Decimal:           key-input("Ctrl-[On")\n \
    :<Key>KP_Subtract:          key-input("Ctrl-[Om")\n \
```

```
    :<Key>F8: key-input(“Ctrl-[[19~”)\n \
    ~Shift <Key>F9: key-input(“Ctrl-[[20~”)\n \
    Shift <Key>F9: key-input(“cd $HOME”)\n \
    ~Shift <Key>osfMenuBar: key-input(“Ctrl-[[21~”)\n \
    Shift <Key>osfMenuBar: key-input(“Ctrl-[[22~”)\n \
    :<Key>F11: key-input(“Ctrl-[[23~”)\n \
    :<Key>F12: key-input(“Ctrl-[[24~”)\n \
    :<Key>F13: key-input(“Ctrl-[[25~”)\n \
    :<Key>F14: key-input(“Ctrl-[[26~”)\n \
    :<Key>F15: key-input(“Ctrl-[[28~”)\n \
    :<Key>Help: key-input(“Ctrl-[[28~”)\n \
    :<Key>F16: key-input(“Ctrl-[[29~”)\n \
    :<Key>Menu: key-input(“Ctrl-[[29~”)\n \
    :<Key>F17: key-input(“Ctrl-[[31~”)\n \
    :<Key>F18: key-input(“Ctrl-[[32~”)\n \
    :<Key>F19: key-input(“Ctrl-[[33~”)\n \
    :<Key>F20: key-input(“Ctrl-[[34~”)\n \
    :<Key>KP_F1: key-input(“Ctrl-[OP”)\n \
    :<Key>KP_F2: key-input(“Ctrl-[OQ”)\n \
    :<Key>KP_F3: key-input(“Ctrl-[OR”)\n \
    :<Key>KP_F4: key-input(“Ctrl-[OS”)\n \
```

```
******************************************************************************
   HDSTERM  DEFAULT  TRANSLATION  DEFINITIONS
******************************************************************************
/*  Note:  The default MOTIF key bindings are used, so the following      */
/*  standard X Windows keysyms must be replaced by their MOTIF            */
/*  bindings in any event translation resources:                          */
/*                                                                        */
/*  X KeySym                           MOTIF KeySym                       */
/*  -------------                      ---------------------              */
/*  Escape                             osfCancel                          */
/*  Left                               osfLeft                            */
```

these keys you must refer to them by that name in the translation table. In the special case of F10, which is an accelerator key, you must explicitly disable the key before you redefine it in order to disable the accelerator function, which normally overrides the key translation (an example is given below).

For ordinary keys, you can just redefine them; for instance, the Backspace key on an IBM keyboard, designated as osfBackSpace, often needs to be remapped to the Delete key for DEC applications. Since this key doesn't appear in any keyset group, you can use a normal screen translation, such as (but note the use of the osfBackSpace name for the keysym):

```
HDSTerm*screen.translations:  #override \

         <Key>osfBackSpace:  key-input("\177")
```

If the key you want to redefine does occur in one of these keyset groups, you must use the entire keyset in your translation. This is necessary because your new keyset resource definition **completely replaces** the built-in default for that resource. Therefore, you may not use an "#override" directive and simply list only those keys you want changed. Rather, you must include all keys which have translations defined in the built-in defaults. For keys not found in the special sets, you may use the "#override" directive on the *screen.translations resource, as these are applied at widget initialization.

For example, if you want Shift-F9 to generate "cd $HOME" and F10 to function as a normal function key instead of the osfMenuBar, you would have to replace the entire "*term.vt220-auxTrans-fkeys" resource. In the example below, this definition is added for the Shift-F9 key. Note that the unshifted F9 must now be explicitly prefaced by ~Shift (this means "not Shift") and the beginning colon removed (the colon used with other keys means "ignore modifier keys"). The F10 key first must have the accelerator function disabled by entering a null value, and is then remapped in the normal way. Note the use of the osf key binding names instead of the normal keysyms.

```
! this sets the osfMenuBar key (F10) to a null value

HDSTerm*menuBar.menuAccelerator: ""

HDSTerm*term.vt220-auxTrans-fkeys: \

 ~Shift <Key>osfHelp: key-input(“Ctrl-[[11~”)\n \

 Shift <Key>osfHelp: key-input(“Ctrl-[[25~”)\n \

 :<Key>F2: key-input(“Ctrl-[[12~”)\n \

 :<Key>F3: key-input(“Ctrl-[[13~”)\n \

 :<Key>osfMenu: key-input(“Ctrl-[[14~”)\n \

 :<Key>F5: key-input(“Ctrl-[Ctrl-[[15~”)\n \

 :<Key>F6: key-input(“Ctrl-[[17~”)\n \

 :<Key>F7: key-input(“Ctrl-[[18~”)\n \
```

appKeypad	Map numeric keypad keys to DEC VT220/320 "application mode" strings
numKeypad	Map numeric keypad keys to DEC VT220/320 "numeric mode" strings
appCKeys	Map cursor keys to DEC VT220/320 application mode" strings
normCKeys	Map cursor keys to DEC VT220/320 normal mode strings
fkeys	Map function keys to the strings generated by XTerm
ibmFkeys	Map function keys F1 to F4 to the DEC VT220/320 PF1 to PF4 strings
miscFkeys	Map "editing" keys to their closest equivalent to DEC VT220/320 editing keys, **based on their key legends**
ibmMiscFkeys	Map "editing" keys on IBM keyboard **based on their positional equivalent** on a DEC VT220/320 keyboard
telnetFkeys	Map Alt-F10 to Alt-F12 to generate complete shell commands to set the DISPLAY environment variable to the ViewStation's IP address:0

The key to these translations is the use of the action procedure "key-input". If key-input has a parameter, it is taken to be an ASCII string which should be transmitted when the specified input event occurs. If key-input is used without a parameter, the specified event sends the normal ASCII binding returned by Xlib's XLookupString() function.

The "application" mode vs. "normal" or "numeric" mode sets are applied as a result of either user input (from the Dialog box), or receipt of escape sequences. The "normal" vs. "ibm" sets are applied only when the user selects/deselects IBM->VT Kybd Emulation from the Customize General dialog box. The "Telnet" set is applied if HDSTerm is started from Setup as a Telnet window, or from a shell with the "-e telnet" command-line option.

2.2.9.1 Sample Key Translations

To customize the keyboard translations, you must first determine which keyset the keys you wish to change belong to. Keys that belong to one of the special sets can be redefined as part of the set. Keys that do not belong to a special set can be redefined with normal key override translations.

If the key to be redefined is not in one of the special sets, you can simply use the "*screen.translations" resource. Note that some keys have special definitions in the Motif widget set, such as F1, F4, and F10 (see the following table). If you use

Delete	ESC [4~	DEL (hex 0x7F)
End	ESC [5~	- nothing -
Page Down	ESC [6~	ESC [6~
Keypad "+"	+	+
Alt Keypad "+"	, (or ESC Ol)	, (or ESC Ol)

2.2.8.2 Using DEC VT320 keyboards

Key Pressed	IBM -> VT On	IBM -> VT Off(rec.)
F1	ESC O P	ESC O P
F2	ESC O Q	ESC O Q
F3	ESC O R	ESC O R
F4	ESC O S	ESC O S
PF1	ESC O P	ESC O P
PF2	ESC O Q	ESC O Q
PF3	ESC O R	ESC O R
PF4	ESC O S	ESC O S
Find	ESC [1~	ESC [1~
Insert Here	ESC [1~	ESC [2~
Remove	ESC [3~	ESC [3~
Select	ESC [4~	ESC [4~
Prev	ESC [5~	ESC [5~
Next	ESC [6~	ESC [6~
Keypad ","	, (or ESC Ol)	, (or ESC Ol)

The Numeric keypad sequence depends on whether HDterm is using Numeric or Application mode. In Application mode the sequence is ESC O l, which is uppercase O, not zero, and lowercase L, not one.

Note also that using the Meta key (if defined with xmodmap for your keyboard) sets the high bit (8th bit) on for any ASCII character.

2.2.9 HDSterm Key Translations

Unlike many X Windows applications, the keyboard translations for HDSterm are not applied as a single default resource, which you modify by applying your own "HDSTerm*translations" resource. Instead, the translations are dynamically applied by HDSterm at run time, using several sets of translations. These sets are stored as application sub-resources, using the instance name "*term.vt220-aux-Trans-<set name>".

The following sets of translations are used (see full listings below):

Key-set Name	Translation Application

The "Close Printer" line will empty the printer buffer and close the connection to the printer.

The following VT320 command sequences for printing are supported:

ESC [i - Print screen
ESC [0 i - Print screen
ESC [4 i - Printer controller mode off
ESC [5 i - Printer controller mode on
ESC [? 1 i - Print line which has cursor
ESC [?4 i - Autoprint mode off
ESC [?5 i - Autoprint mode on

If the printer device driver fills its buffer so printing is stopped (usually caused by the printer going Offline or running out of paper), HDSterm pops up a warning window asking you to OK the command to close the printer. If you can fix the problem so the printer can resume its printing and the buffer starts to empty, the warning will disappear. All communications with the printer port are suspended while the warning window is displayed. Clicking on the "OK" or closing the warning window with the window manager closes the printer port and loses the data remaining in the printer buffer.

2.2.8 VT320 Keyboard Mapping

The HDS ViewStation can use different keyboards, though the most common types are IBM 101+ and the DEC VT320 type. These keyboards have different layouts and handle several important keyboard functions in different ways, specifically, the Numeric keypad, the editing keypad, and the function keys. HDSterm, as a VT320 emulation, generally expects these key functions to correspond to DEC keys and key codes.

The HDSterm client allows you to select, from the Customize Window General dialog box, whether your keyboard (IBM or DEC type) sends VT320 sequences or the normal keyboard sequences. The "VT On" selection is recommended for HDSterm for both the IBM and DEC keyboards. The actual selections and their corresponding sequences are shown in the tables below.

2.2.8.1 Using IBM 101+ or 102 keyboards:

Key Pressed	IBM -> VT On (rec.)	IBM -> VT Off
F1	ESC O P	ESC [11~
F2	ESC O Q	ESC [12~
F3	ESC O R	ESC [13~
F4	ESC O S	ESC [14~
Insert	ESC [1~	ESC [2~
Home	ESC [2~	- nothing -
Page Up	ESC [3~	ESC [5~

ESC [41 m	Background red
ESC [42 m	Background green
ESC [43 m	Background yellow
ESC [44 m	Background blue
ESC [45 m	Background magenta
ESC [46 m	Background cyan
ESC [47 m	Background white

Note that ESC [0 m resets all attributes to Off. It also resets the text background and foreground colors to the initial colors selected for the screen widget (default is foreground = black and background = lightsteelblue). These escape sequences are ignored on a monochrome server.

These colors are the ANSI standard selections, but the actual colors can be modified by using HDSterm resources that correspond to the escape sequences above. The HDSterm resources to define the colors are in the following form:

HDSTerm*screen.ansiFg_0 = defines the color for ESC [30 m
HDSTerm*screen.ansiFg_6 = defines the color for ESC [36 m
HDSTerm*screen.ansiBg_1 = defines the color for ESC [41 m
HDSTerm*screen.ansiBg_3 = defines the color for ESC [43 m

and so on.

2.2.7 HDSterm Print Commands

HDSterm has local printing support available from a sub-menu on the Commands pull-down menu. The printer is connected to the ViewStation, either to the parallel or one of the serial ports. This printer location is specified by an HDSterm resource:

```
HDSTerm*screen.printDevice:  /dev/pp0
```

which specifies the parallel port (/dev/pp0 is the default entry). You can also specify either of the serial ports, /dev/tty1 (serial port 1 - RJ11) or /dev/tty2 (serial port 2 - DB25).

The following commands are available from the Print Options sub-menu.

- Print Window
- Printer Logging ON
- Printer Logging OFF
- Close Printer

Clicking on the "Print Window" line sends the current contents of the HDSTerm window to the printer. This corresponds to the VT320 command CSI i.

The "Printer Logging ON" and "Printer Logging OFF" lines refer to the VT320 Autoprint function, corresponding to the CSI ? 4 i and CSI ? 5 i commands, respectively.

- Screen Alignment pattern

ESC # 8	- displays a full screen of E’s

- Bell

^G (0x07)	- ring keyboard bell

- Reset

ESC c	- reset terminal emulation

- Reports

CSI *Ps* n	- Device Status Report
Ps = 5	- Operating status
Ps = 6	- Request cursor position
Ps = 7	- Request screen size
CSI 0 c	(or CSI c)- Terminal ID
CSI *Ps* x	- Request Terminal Parameters
^E (0x05)	- Send answerback message

- Set Window, Icon Label

ESC] 0 ; <*label*> ^G	- Set window and icon labels to <*label*>

- Local Printing

ESC [i	- Print screen
ESC [0 i	- Print screen
ESC [4 i	- Printer controller mode off
ESC [5 i	- Printer controller mode on
ESC [? 1 i	- Print line which has cursor
ESC [?4 i	- Autoprint mode off
ESC [?5 i	- Autoprint mode on

- ANSI Color Support

HDSterm supports escape sequences to change foreground and background colors, as follows:

ESC [30 m	Foreground black
ESC [31 m	Foreground red
ESC [32 m	Foreground green
ESC [33 m	Foreground yellow
ESC [34 m	Foreground blue
ESC [35 m	Foreground magenta
ESC [36 m	Foreground cyan
ESC [37 m	Foreground white
ESC [40 m	Background black

Ps = 4	- display underlined
Ps = 5	- display blinking
Ps = 7	- display reverse image
Ps = 22	- display normal intensity
Ps = 24	- display not underlined
Ps = 25	- display not blinking
Ps = 27	- display positive image

- Double width/height text.

ESC # 3	- double height line top half
ESC # 4	- double height line bottom half
ESC # 5	- single width single height line
ESC # 6	- double width single height line

- Editing

CSI *Pn* L	- insert Pn lines
CSI *Pn* M	- delete Pn lines
CSI *Pn* @	- insert Pn blank characters with attributes set to normal
CSI *Pn* P	- delete Pn characters
CSI *Pn* X	- erase Pn characters
CSI K	- erase to the end of the line including cursor
CSI 0 K	- erase to the end of the line including cursor
CSI 1 K	- erase form the beginning of the line including cursor
CSI 2 K	- erase the complete line
CSI J	- erase from cursor to end of screen
CSI 0 J	- erase from cursor to end of screen
CSI 1 J	- erase from beginning of screen to cursor
CSI 2 J	- erase the complete display

- Scrolling Margins

CSI *Pt* ; *Pb* r	- Sets top and bottom margins of the scrolling region to Pt and Pb
CSI r	- Sets top and bottom margins of the scrolling region to the first and last displayed lines

CSI *Pn* B	- move cursor down Pn lines
CSI *Pn* C	- move cursor right Pn columns
CSI *Pn* D	- move cursor left Pn columns
CSI *Pl* ; *Pc* H	- move cursor to line Pl, columns Pc
CSI *Pl* ; *Pc* f	- move cursor to line Pl, columns Pc
IND (ESC D)	- move cursor down one line, scrolled
RI (ESC M)	- move cursor up one line, scrolled
NEL (ESC E)	- move cursor down one line and to the first column, scrolled
^H (0x08)	- move cursor left one column
^J (0x0a)	- perform "newline" function, which is either the same as IND or NEL, depending on the current linefeed/newline mode.
^K (0x0b)	- vertical tab, behaves as ^J above
^L (0x0c)	- form feed, behaves as ^J above
^M (0x0d)	- move cursor to first column of current line
ESC 7	- save states: cursor postion graphic rendition character set shift state wrap flag origin mode (not supported) selective erase (not supported)
ESC 8	- restore saved states

- TAB Stops

HTS (ESC H)	- Set tab stop at the current column
CSI g	- Clear tab at current column
CSI 0 g	- Clear tab at current column
CSI 3 g	- Clear all tab stops
^I (0x09)	- Move cursor right to next tab stop

- Character Rendition and Attributes

CSI *Ps* ; *Ps* ; ... m	- Select character rendition with
Ps = 0	- everything off
Ps = 1	- display bold

ESC + 0	- DEC spec. graphics -> G3

- Invoking Character Sets Using Locking Shifts

SI (Ctrl O)	- Invoke G0 into GL (default)
SO (Ctrl N)	- Invoke G1 into GL
ESC ~	- Invoke G1 into GR
ESC n	- Invoke G2 into GL
ESC }	- Invoke G2 into GR (default)
ESC o	- Invoke G3 into GL
ESC |	- Invoke G3 into GR

- Invoking G2 or G3 into GL For a Single Character Using Single Shifts

SS2 (ESC N)	- Single Shift G2
SS3 (ESC O)	- Single Shift G3

- Terminal Modes

ANSI modes:	
CSI *Ps* ; *Ps* ; ... l	- Reset mode
CSI *Ps* ; *Ps* ; ... h	- Set mode
Ps = 4	- replace mode / insert mode
Ps = 20	- linefeed mode / newline mode
DEC private modes:	
CSI ? *Ps* ; *Ps* ; ... h	- Set mode
CSI ? *Ps* ; *Ps* ; ... l	- Reset mode
Ps = 1	- cursor keys generate "application control" /
ANSI cursor control sequences	
Ps = 3	- 132 / 80 columns per line
Ps = 5	- black chars on white / white chars on black
Ps = 6	- cursor positioning relative to scroll area / global
Ps = 7	- wrap / don't wrap at the end of the line
Ps = 8	- keyboard autorepeat on / off
Ps = 25	- cursor visible / invisible
ESC =	- select application keypad mode
ESC >	- select numeric keypad mode

- Cursor Positioning

CSI *Pn* A	- move cursor up Pn lines

```
HDSTerm*hostl.labelString: Connected To:
```

2.2.5.2 HDSTerm Resources in $HOME/.Xdefaults

You can use your own resource specifications for HDSterm by creating a resource file of your own. These resources will be read from the home directory, or some other location, and will replace some or all of the default resources in HDS$$ROM as you specify.

For example, if you want to use different settings or different fonts or colors, you could change the resource specifications as follows:

```
HDSTerm*largeFont: 9x15

HDSTerm*screen.BlinkInterval: 200

HDSTerm*screen.background: darksteelblue
```

2.2.6 HDSterm Command Sequences

The HDSterm client is a VT320 terminal emulator. It responds to all the standard VT320 command sequences. These are listed below:

HDSterm uses 7-bit control characters. Therefore, 8-bit control codes (values 0x80 through 0x9f) should be expressed as two byte sequences:

For example, the CSI character (decimal 155, or hex 9B) :

CSI (ASCII 0x09B) should be expressed as "ESC ["
DCS (ASCII 0x090) should be expressed as "ESC P"
and so on.

2.2.6.1 HDSterm Supported Command Set:

- Designating "Hard" Character Sets

ESC (B	- ASCII -> G0
ESC) B	- ASCII -> G1
ESC * B	- ASCII -> G2
ESC + B	- ASCII -> G3
ESC (<	- DEC Suppl. -> G0
ESC) <	- DEC Suppl. -> G1
ESC * <	- DEC Suppl. -> G2
ESC + <	- DEC Suppl. -> G3
ESC (A	- U.K. Ntl. -> G0
ESC) A	- U.K. Ntl. -> G1
ESC (0	- DEC spec. graphics -> G0
ESC) 0	- DEC spec. graphics -> G1
ESC * 0	- DEC spec. graphics -> G2

```
HDSTerm*mediumFont: 8x13
HDSTerm*mediumBoldFont: 8x13bold
HDSTerm*largeFont: 10x20
HDSTerm*term.marginWidth: 10
HDSTerm*term.marginHeight: 10
HDSTerm*term.shadowThickness: 4
HDSTerm*screen.WrapAround: True
HDSTerm*screen.CursorBlinking: True
HDSTerm*screen.BlinkWOFocus: False
HDSTerm*screen.BlinkInterval: 400
HDSTerm*screen.CursorHeight: 3
HDSTerm*screen.CursorWidth: 0
HDSTerm*screen.font: 9x15
HDSTerm*screen.background: lightsteelblue
HDSTerm*screen.printDevice:  /dev/pp0
HDSTerm*top.scrollBarPlacement: bottom_left
HDSTerm*XmText.fontList: 9x15
HDSTerm*XmTextField.fontList: 9x15
HDSTerm*XmPushButton.fontList: menu12
HDSTerm*XmPushButtonGadget.fontList: menu12
HDSTerm*XmLabel.fontList: menu12
HDSTerm*XmLabelGadget.fontList: menu12
HDSTerm*XmCascadeButton.fontList: menu12
HDSTerm*XmCascadeButton.marginWidth: 10
HDSTerm*XmToggleButtonGadget.fontList: menu12
HDSTerm*XmList.fontList: 9x15
HDSTerm*hosts.selectionLabelString: Selected Host:
HDSTerm*hosts.listLabelString: Host List:
HDSTerm*hosts.listVisibleItemCount: 6
HDSTerm*serial_port.messageString: Please Select Serial
Port:
HDSTerm*sessionl.labelString: Session:
```

```
    Shift <Key> osfPageDown:        call-canvas(46, “”, a, -200, b, 1) \n\
    Shift <Key> osfUp:              call-canvas(46, “”, a, 1, b, 1) \n\
    Shift <Key> osfDown:            call-canvas(46, “”, a, -1, b, 1) \n\
    <Btn1Down>:                     select-start() \n\
    <Btn1Motion>:                   select-extend() \n\
    <Btn2Up>:                       insert-selection(PRIMARY,
CUT_BUFFER0)\n\
    <Btn3Down>:                     select-extend() \n\
    <Btn3Motion>:                   select-extend() \n\
    <Btn1Up>:                       select-end(PRIMARY, CUT_BUFFER0) \n\
    <Btn3Up>:                       select-end(PRIMARY, CUT_BUFFER0) \n\
    <Key>:                          key-input()
```

ENVIRONMENT

HDSterm sets the environment variable “TERM” to vt320 when the command resource is set to telnet, or when starting HDSterm from **Setup**’s *Telnet Terminal Window* menu button.

SEE ALSO

Xterm(1), X(1)

BUGS

Please report all bugs to Human Designed Systems, Inc. at *support@hds.com.*

COPYRIGHT

Copyright 1992, Human Designed Systems, Inc. Portions adapted from emu, copyright 1990, PCS Computer Systeme GmbH.

AUTHORS

Edward Parks, Michael Hartman, Bill Bredbenner (Human Designed Systems). Thanks to Jordan Hubbard, Michael Elbel, Terry Jones, and Thomas Bagli (PCS Computer Systeme GmbH) for donating the emu source to the MIT Consortium, and to all those *far too many people* who developed Xterm.

2.2.5 HDSTerm Resource Files

Resource files for configuring the HDSterm appearance and operation are contained in resource files located in the ViewStation server code, specifically in the path /HDS$$ROM/app-defaults. These default resources will always be found when the ViewStation is running.

Note that HDSterm resources use uppercase “T” in the client name.

2.2.5.1 HDSTerm Resources in HDS$$ROM

A listing of the default resources follows:

```
HDSTerm*smallFont: 6x10
```

unpredictable behavior. The following actions are provided for using within the *screen* **translations** resource:

key-input()

Without arguments, this action inserts the 8-bit ISO Latin-1 character or string associated with the keysym that was pressed. If the Mod1 modifier is pressed with the key event, the high-order bit will be set. If a single parameter is supplied, this string will be inserted instead.

insert-selection *(sourcename [, ...])*

This action inserts the string found in the selection or cutbuffer indicated by sourcename. Sources are checked in the order given (case is significant) until one is found. Commonly-used selections include: *PRIMARY, SECONDARY, and CLIPBOARD*. Cut buffers are typically named *CUT_BUFFER0* through *CUT_BUFFER7*.

select-start()

This action begins text selection at the current pointer location. See the section on **POINTER USAGE** for information on making selections.

select-extend()

This action tracks the pointer and extends the selection.

select-end*(destname [, ...])*

This action puts the currently selected text into all of the selections or cutbuffers specified by destname.

focus-in()

This action modifies the text cursor to be a solid block or solid underline, showing that the window has keyboard focus. If blinking cursor is enabled, the cursor will begin blinking.

focus-out()

This action modifies the text cursor to be an outline block or underline, showing that the window does not have keyboard focus. If blinking cursor is enabled, but blinkWOFocus is False, the cursor stops blinking and becomes solid.

call-parser*(opcode, buffer, [register, int_val], ...)*

This action calls the reverse parser directly. Currently this function is for internal use only.

call-canvas*(opcode, buffer, [register, int_val], ...)*

This action calls the canvas widget function dispatcher directly. Currently this function is for internal use only.

The default bindings in the VT320 window are:

```
<FocusIn>:                  focus-in() \n\
<FocusOut>:                 focus-out() \n\
Shift <Key> osfPageUp:      call-canvas(46, “”, a, 200, b, 1) \n\
```

HDSterm will leave the selected text highlighted for as long as it is the selection owner.

Pointer button two (usually middle) 'types' (pastes) the text from the PRIMARY selection, if any, otherwise from the cut buffer, inserting it as keyboard input.

Pointer button three (usually right) extends the current selection. The original selection starting point is retained, and the endpoint is moved to correspond to the new pointer selection. When button three is released, the newly extended selection is saved.

By cutting and pasting pieces of text without trailing new lines, you can take text from several places in different windows and form a command to the shell, for example, or take output from a program and insert it into your favorite editor. Since the cut buffer is globally shared among different applications, you should regard it as a 'file' whose contents you know. The terminal emulator and other text programs should be treating it as if it were a text file, i.e. the text is delimited by new lines.

The scroll bar position and length corresponds to the position and amount of text currently showing in the window, relative to the amount of text actually saved. As more text is saved (up to the maximum), the length of the scrollbar thumb decreases.

CUSTOMIZING HDSTERM

Once you have created an HDSterm window, you can choose the operating features you want the terminal to have by using the HDSterm Customize menu.

The **Window...** menu item pops up a dialog box which allows you to modify the size and appearance of the terminal window. The current settings are indicated by shaded buttons.

Use the Apply button to view your changes without closing the dialog box. The OK button will apply your changes and close the dialog.

The Window dialog will allow you to enable or disable the vertical scroll bar, change fonts, and change the number of rows and/or columns in the displayed window.

The **Display...** menu item pops up a dialog box which allows you to modify various VT320 emulation features which affect the way text is displayed, including Auto Wrap, Cursor Blink, Cursor Type, and Normal/Reverse video.

The **General...** menu item pops up a dialog box which allows you to modify internal settings of the VT320 emulation, including Linefeed/Newline mode, keyboard autorepeat, keyboard emulation, and the ascii codes generated by numeric keypad keys and cursor keys.

ACTIONS

It is possible to rebind keys (or sequences of keys) to arbitrary strings for input, by changing the translations for the *screen* widget. Changing the translations for events other than key and button events is not expected, and will cause

When set True, specifies that LF, VT, and FF control characters received should cause the cursor to move to the leftmost column of the next line, and scroll if necessary. Also, pressing the "Enter" key will transmit both CR and LF to the program running in the HDSterm window. The default is False, which specifies that LF, VT, and FF cause the cursor to move to the current column of the next line, and pressing "Enter" will transmit only a CR character.

printDevice (class PrintDevice)

Specifies the device driver for local printing. The default is "/dev/pp0" (the parallel port). Serial ports 1 and 2 use drivers "/dev/tty1" and "/dev/tty2" respectively.

ansiFg_n (class AnsiFg_n)

Specifies the color to be used as the text foreground when the ANSI escape sequence to change foreground color is received. The digit "*n*", which must be in the range 0-7, corresponds to the ANSI SGR sequence "ESC [3*n* m". The ANSI sequence to clear all attributes, "ESC [0 m", will reset both the text foreground and background colors to their original values specified by the screen widget's foreground and background resources.

ansiBg_n (class AnsiBg_n)

Specifies the color to be used as the text background when the ANSI escape sequence to change background color is received. The digit "*n*", which must be in the range 0-7, corresponds to the ANSI SGR sequence "ESC [4*n* m". The ANSI sequence to clear all attributes, "ESC [0 m", will reset both the text foreground and background colors to their original values specified by the screen widget's foreground and background resources.

EMULATIONS

The VT320 emulation is fairly complete, but does not support the host-writable status line. *Termcap(5)* entries that work with *HDSterm* include "vt320", "xterm", "vt220", "vt102", "vt100" and "ansi".

POINTER USAGE

Once the HDSterm window is created, *HDSterm* allows you to select text and copy it within the same or other windows.

The selection functions are invoked when the pointer buttons are used with no modifiers. The assignment of the functions described below to keys and buttons may be changed through the resource database; see **ACTIONS** below.

Pointer button one (usually left) is used to save text into the cut buffer. Move the cursor to the beginning of the text, and then hold the button down while moving the cursor to the end of the region and releasing the button. The selected text is highlighted and is saved in the global cut buffer and made the PRIMARY selection when the button is released. Double-clicking selects by words. Triple-clicking selects by lines. Multiple-click is determined by the time from button up to button down, so you can change the selection unit in the middle of a selection. If the key/button bindings specify that an X selection is to be made,

columns (class Columns)

Specifies the number of text columns in the displayed window.

cursorHeight (class CursorHeight)

Specifies the height of the text cursor in pixels.

cursorWidth (class CursorWidth)

Specifies the width of the text cursor in pixels.

cursorBlinking (class CursorBlinking)

Specifies that the text cursor should blink. The default is True.

blinkInterval (class BlinkInterval)

Specifies the interval in milliseconds for the blinking text cursor.

blinkWOFocus (class BlinkWOFocus)

Specifies that the text cursor should continue to blink when the window loses input focus. The default is False.

textBlinkInterval (class TextBlinkInterval)

Specifies the interval in milliseconds for blinking text.

wrapAround (class WrapAround)

Specifies that text should wrap to the next line after reaching the last column in the previous line. The default is True.

insertMode (class InsertMode)

Specifies that text should be inserted, causing characters to the right of the insert point to be shifted right. The default is False.

bellVolume (class BellVolume)

Specifies the bell volume as a percentage from 0 to 100.

defTabWidth (class DefTabWidth)

Specifies the default spacing of initial tabstops.

multiClickTime (class MultiClickTime)

Specifies the maximum time in milliseconds between multi-click select events. The default is 300 milliseconds.

pointerColor (class PointerColor)

Specifies the foreground color of the pointer.

pointerShape (class PointerShape)

Specifies the name of the shape of the pointer. The default is ‘‘xterm.’’

saveLines (class SaveLines)

Specifies the number of lines to save beyond the top of the screen. The default is the number of text lines or 64, whichever is greater.

lineMode (class LineMode)

MainWindow, ScrollBar, TextField, Label, Frame, PushButton, CascadeButton, and other MOTIF widgets used may be modified by the user. For example, setting the *top.scrollBarPlacement resource to bottom_right or top_right will move the vertical scroll bar to the right side of the window.

The following resources are specified as part of the term widget (class Term):

command (class Command)

Specifies the command to be run in the HDSTerm window. The default is the local shell (/bin/sh).

termType (class TermType)

Specifies the terminal emulation to be used. Currently only vt320 is supported.

marginWidth (class MarginWidth)

This option specifies the number of pixels between the characters and the left and right window borders.

marginHeight (class MarginHeight)

This option specifies the number of pixels between the characters and the top and bottom window borders.

The following resources are specified as part of the screen widget (class TermCanvas):

background (class Background)

Specifies the color to use for the background of the window. The default is “lightsteelblue.”

foreground (class Foreground)

Specifies the color to use for displaying text in the window. The default is “black.”

cursorFg (class CursorFg)

Specifies the color to use for the text cursor. The default is “black.”

cursorBg (class CursorBg)

Specifies the color to use for the text cursor background. The default is “white.”

reverseVideo (class ReverseVideo)

Specifies whether or not reverse video should be simulated. The default is “false.”

underlineWidth (class UnderlineWidth)

Specifies the width in pixels of the underline for characters written with the underline attribute.

lines (class Lines)

Specifies the number of lines of text in the displayed window. This resource also sets the initial scrolling region to include all visible lines.

vertScrollBar (class VertScrollBar)

Setting this option True indicates that a scrollbar should be displayed so that the lines which are saved after scrolling off the top can be viewed. This option may be turned on and off from the "Customize Window" menu.

statusLine (class StatusLine)

Setting this option True indicates that the info/status area should be displayed at the bottom of the main window. This option may be turned on and off from the "Customize Window" menu.

smallFont (class SmallFont)

This option specifies a font to be designated as the *small* font, which is selected via the "Customize Window" menu.

mediumFont (class MediumFont)

This option specifies a font to be designated as the *medium* font, which is selected via the "Customize Window" menu.

largeFont (class LargeFont)

This option specifies a font to be designated as the *large* font, which is selected via the "Customize Window" menu.

smallBoldFont (class SmallBoldFont)

This option specifies a font to be designated as the *small bold* font, which is used for bold text when the small font is selected via the "Customize Window" menu. This font must be the same height and width as the normal small font.

mediumBoldFont (class MediumBoldFont)

This option specifies a font to be designated as the *medium bold* font, which is used for bold text when the medium font is selected via the "Customize Window" menu. This font must be the same height and width as the normal medium font.

largeBoldFont (class LargeBoldFont)

This option specifies a font to be designated as the *large bold* font, which is used for bold text when the large font is selected via the "Customize Window" menu. This font must be the same height and width as the normal large font.

vtEmulation (class VtEmulation)

Setting this option True turns on IBM->VT keyboard emulation mode. See the "-VT" command line option above for a description.

HDSterm creates a MOTIF MainWindow widget ("top"), consisting of a MenuBar widget ("MenuBar"), a vertical ScrollBar widget ("vsb"), a message area consisting of a TextField status widget ("status"), and some informational Label widgets. The WorkArea of the MainWindow widget is a custom widget ("term") which is a subclass of the MOTIF Frame widget. The term widget in turn manages a single custom widget ("screen"), which is a subclass of the Xt Core widget. Most of the MOTIF resources applying to

This option specifies the color to use for the border of the window. The default is "black."

-fg ***color***

This option specifies the color to use for displaying text. The default is "black".

-fn ***font***

This option specifies the font to be used for displaying normal text. The default is 9x15.

-name ***name***

This option specifies the application name under which resources are to be obtained, rather than the default **HDSTerm**. *Name* should not contain "." or "*" characters.

-title ***string***

This option specifies the window title string, which may be displayed by window managers if the user so chooses. The default title is hdsterm.

-rv

This option indicates that reverse video should be simulated by swapping the foreground and background colors.

-geometry ***geometry***

This option specifies the preferred size and position of the window.

-display ***display***

This option specifies the X server to contact; see X(1).

-xrm ***resourcestring***

This option specifies a resource string to be used. This is especially useful for setting resources that do not have separate command line options.

-iconic

This option indicates that *HDSterm* should ask the window manager to start it as an icon rather than as the normal window.

RESOURCES

The program understands all of the core X Toolkit resource names and classes. Note that the client name is HDSTerm (note uppercase "T"). In addition, the following Application Resources are defined:

title (class Title)

Specifies a string that may be used by the window manager when displaying this application.

iconName (class IconName)

This option specifies the icon name for HDSTerm's window.

-bfocus

This option indicates that the text cursor should stop blinking when the *HDSterm* window loses input focus. This is the default.

+bfocus

This option indicates that the text cursor should continue to blink when the *HDSterm* window loses input focus.

-wrap

This option indicates that text should wrap to column one of the next line after reaching the last column of the previous line. This is the default.

+wrap

This option disables wrapping of text after reaching the last column of the previous line. Additional characters overwrite the character in the last column.

-insert

This option turns off insert mode, and enables replace mode. This is the default.

+insert

This option turns on insert mode, and disables replace mode.

-bell *number*

This option specifies the bell volume, as a percentage from 0 to 100.

-tab *number*

This option specifies that tabstops should be evenly placed at every *number* character positions. The default is 8.

-VT

This option turns on IBM->VT keyboard emulation mode. This is useful when using an IBM style keyboard with applications which use PFx function keys and/or "editing" keys. This option causes function keys F1-F4 to generate the strings normally generated by the PF1-PF4 keys on a VT320 terminal keyboard, and remaps the editing keys to their DEC keyboard equivalents based on position rather than the legends printed on the key tops.

The following standard X Toolkit command line arguments are commonly used with *HDSterm*:

-bg *color*

This option specifies the color to use for the background of the window. The default is "lightsteelblue."

-bd *color*

+vsbar

This option indicates that a vertical scrollbar should not be displayed.

-nostatus

This option indicates that the info/status area at the bottom of the main window should not be displayed.

-sl *number*

This option specifies the number of lines to save that have been scrolled off the top of the screen. The default is 64.

-T *string*

This option specifies the title for *HDSterm*'s window. It is equivalent to -title.

-n *string*

This option specifies the icon name for *HDSterm*'s window. It is shorthand for specifying the "*icon-Name" resource. Note that this is not the same as the toolkit option -name (see below). The default icon name is hdsterm.

-mw *number*

This option specifies the number of pixels between the characters and the left and right window borders.

-mh *number*

This option specifies the number of pixels between the characters and the top and bottom window borders.

-ul *number*

This option specifies the width in pixels of the underline for characters written with the underline attribute.

-ch *number*

This option specifies the height of the text cursor in pixels.

-cw *number*

This option specifies the width of the text cursor in pixels.

-cbl

This option indicates that the text cursor should not blink.

+cbl

This option indicates that the text cursor should blink.

-cbival *number*

This option specifies the text cursor blink interval, in milliseconds. The default is 400 ms.

-tbival *number*

This option specifies the blink interval for text written with the blinking attribute. The default is 500 ms.

This option specifies the program (and its command line arguments) to be run in the HDSterm window. **This must be the last option on the command line.**

-fb *font*

This option specifies a font to be used when displaying bold text. This font must be the same height and width as the normal font. If only one of the normal or bold fonts is specified, it will be used as the normal font and the bold font.

-fs *font*

This option specifies a font to be designated as the small font, which is selected via the "Customize Window" menu.

-fm *font*

This option specifies a font to be designated as the medium font, which is selected via the "Customize Window" menu.

-fl *font*

This option specifies a font to be designated as the large font, which is selected via the "Customize Window" menu.

-fsb *font*

This option specifies a font to be designated as the small bold font, which is used for bold text when the small font is selected via the "Customize Window" menu. This font must be the same height and width as the normal small font.

-fmb *font*

This option specifies a font to be designated as the medium bold font, which is used for bold text when the medium font is selected via the "Customize Window" menu. This font must be the same height and width as the normal medium font.

-flb *font*

This option specifies a font to be designated as the large bold font, which is used for bold text when the large font is selected via the "Customize Window" menu. This font must be the same height and width as the normal large font.

-mclick *milliseconds*

This option specifies the maximum time between multi-click selections.

-pcol *color*

This option specifies the color to be used for the pointer cursor. The default is to use the foreground color.

-pcur *cursor*

This option specifies the cursor shape to be used for the pointer cursor. The default is the "xterm'"text cursor.

-vsbar

This option indicates that a scrollbar should be displayed so that the lines which are saved after scrolling off the top can be viewed. This option may be turned on and off from the "Customize Window" menu.

which then sends the answerback string (your command line) to be executed in the new shell.

If you will be logging in from something other than HDSterm, Xterm, or a VTxxx terminal, you may wish to execute the echo command conditionally based on the appropriate TERM environment variable.

2.2.4 HDSterm Manual Page

This note contains a printed copy of the HDSterm manual page. This page is available on the HDSware tape as an nroff file, labeled hdsterm.1 in the *./hds-fx/man* directory. This means that you can add this nroff file to your other man pages and have help for the HDSterm client available with your normal man utility. If you want to see the file as text output, you can use the command:

```
nroff -man hdsterm.1
```

The HDSterm man page contains all the command line arguments and resources for HDSterm.

= =

HDSTERM(1) USER COMMANDS HDSTERM(1)

NAME

HDSterm - MOTIF based VT320 compatible terminal emulator for X

SYNOPSIS

hdsterm [-*toolkitoption ...*] [-option ...]

DESCRIPTION

The *HDSterm* program is a MOTIF based terminal emulator for the X Window System. It provides DEC VT320 compatible terminals for programs that can't use the window system directly.

OPTIONS

The *HDSterm* terminal emulator accepts all of the standard Goatlike command line options as well as the following (if the option begins with a '+' instead of a '-', the option is restored to its default value):

-help

This causes *HDSterm* to print out a verbose message describing its options.

-cfg ***color***

This option specifies the foreground color to use for text cursor. The default is to use the same foreground color that is used for text.

-cbg ***color***

This option specifies the background color to use for text cursor. The default is to use the same foreground color that is used for text.

-e ***program [arguments ...]***

```
8 <\\E[?62;1;2;6;7;8;9c> \

and so on ....
```

Note that you must use the entire “vt220-rops:” entry - these are just file fragments.

The syntax for the answerback line is a “6”, followed by a space, followed by “<%”, followed by the desired string in double quotes, followed by “%s>”, a space, and a backslash. There may not be any spaces or other characters following the backslash. In order for the user’s shell to execute the answerback string as a command, it must end with a newline. This is specified by adding \\n just before the terminating double quote (two backslashes before the “n”).

3. Put these resources in your .Xdefaults file in your home directory. Preload the .Xdefaults resource file shown above into the ViewStation’s RESOURCE_MANAGER property by running **xrdb**. This may be done as part of an Xdm .xsession startup script. Loading these resources into the X Server’s RESOURCE_MANAGER property allows them to override HDSterm’s app-defaults, without requiring HDSterm to load a remote resource file every time it starts.

Step 4. Add the commands to open the HDSterm window to your client launcher.

For example, you could auto-start Uniplex by adding the following line to your window manager menu (such as .mwmrc) , or to your .xsession file (if you’re using Xdm), or some other client launcher file:

For an .mwmrc menu entry:

```
“Uniplex Window” f.exec “hdsterm -name Uniplex -e rlogin
myhost”

“Mail Window” f.exec “hdsterm -name Mail -e rlogin myhost”
```

For an .xsession file:

```
 rsh <ViewStation name> “hdsterm -name Uniplex -e rlogin
<hostname> &”
```

These command lines do a remote login to the host specified and execute the user’s startup shell script (which contains the “echo “^E” line at the end).

The “name” argument finds the appropriately renamed *term resource and uses the answerback message defined in the resource entry.

Note that this requires that HDSterm was started remotely, or from a remotely started window manager, and that the */etc/hosts.equiv* and/or *$HOME/.rhosts* files are set up properly. These are prerequisites to correct operation of the remote shell command. It also requires that the user be logged in (username and password) before the command is sent.

If this is correctly configured and run, the rlogin request to the <remote host> will pass the user’s name and password, and will then immediately execute the user’s startup shell script. The shell script in turn sends the ENQ code to HDSterm,

```
“miscFkeys” pa 32 C 0 pX ; \
34 C gy 1 - pa 0 pb 21 C \
53 C 0 px 0 py 1 C \
0 pa 60 C \
0 pb 61 C \
}>
```

The line which specifies the answerback message is the fifth line:

```
6 <%”Welcome to HDSterm”%s> \
```

The procedure to customize the message for specific instances of HDSterm is as follows:

1. Choose a unique name for each instance of HDSterm, one for each different application which you want to autostart.

For example, you could use “Uniplex” for an HDSterm which runs Uniplex, and “Mail” for an HDSterm which runs */usr/ucb/mail*, your mail application.

2. Create a separate resource file entry which contains a ”vt220-rops” resource for each different instance, using the chosen names as the “application name”. You can do this either by copying this part of the app-defaults file, or by using cut-and-paste from the Console window into your .Xdefaults file. You must use this whole segment of the file.

Customize the answerback string in each resource with the necessary command line to launch the desired application. Follow these examples:

```
Uniplex*term.vt220-rops: \
1 <%gA%c> \
2 <\\E[%gy%d;%gx%dS> \
3 <\\E[%{gy,1,+,d};%{gx,1,+,d}R> \
6 <%”setenv TERM vt100; /usr/bin/uniplex; exit\\n”%s> \
7 <\\E[0n> \
8 <\\E[?62;1;2;6;7;8;9c> \
and so on ....
Mail*term.vt220-rops: \
1 <%gA%c> \
2 <\\E[%gy%d;%gx%dS> \
3 <\\E[%{gy,1,+,d};%{gx,1,+,d}R> \
6 <%”/usr/ucb/mail; exit\\n”%s> \
7 <\\E[0n> \
```

- It starts Uniplex
- It sets the “exit” command, which causes the window to close when you leave Uniplex.

Test the command line for correct operation before you use it.

Step 3. Modify the HDSterm resource to set the new answerback message:

HDSterm’s resource which sets the answerback message also includes a number of other items which must remain exactly as in the HDSTerm app-defaults file. You can find this file by opening a Console window and entering these commands:

```
HDS> cd HDS$$ROM

HDS> more HDSTerm
```

This is part of the HDSTerm app-defaults file (the relevant section is at the end of the file).

The following entry is the resource as given in the HDSTerm app-defaults file:

```
*term.vt220-rops: \

1 <%gA%c> \

2 <\\E[%gy%d;%gx%dS> \

3 <\\E[%{gy,1,+,d};%{gx,1,+,d}R> \

6 <%”Welcome to HDSterm”%s> \

7 <\\E[0n> \

8 <\\E[?62;1;2;6;7;8;9c> \

9 <\\E[%{gx,2,+,d};1;1;112;112;1;0;x> \

100 <%{ \

 77 C 75 C 0 pP \

 “ascii” p0 “dec” p1 “graphics” p2 “uk” p3 \

 0 p5, 1 p6 \

 “ascii” pa 32 pb 33 C \

 “dec” pa 160 pb 33 C \

 0 pF \

 0 gT = ? “composeKeys” pa 32 C \

 1 pK 1 pC 1 pX 1 pT ; \

 1 gK = ? “numKeypad” pa 32 C 0 pK ; \

 1 gC = ? “normCKeys” pa 32 C 0 pC ; \

 1 gX = ? “fkeys” pa 32 C \
```

cation with the Uniplex application. Most character-based applications that run in a terminal emulation window, including Uniplex, vi, etc., do not operate properly if their input and output is to a network socket rather than to a tty connection.

Unfortunately, neither the rlogin daemon nor the telnet daemon have provision in their operation protocol to launch another task once they set up the pseudo-tty communication. Both of these daemons execute the "/bin/login" process, which records the login system files and then starts up the user's default shell.

An HDSterm solution

A simple solution exists which makes use of HDSterm's customizable "answerback" message. The general idea is to request the answerback message from HDSterm as part of your startup shell script, and then to enter your application program command line as the content of the answerback message. In this way, the application program is started by your startup shell script.

Additional customization, described below, allows you to use this facility from your window manager's menu, and lets you have several applications available to you with this feature.

Follow these steps:

Step 1. Add the following entry at end of your default shell startup script (.cshrc or .profile in your home directory):

```
echo "^E"
```

This "^E" (Control-E) is the standard ENQ or "Enquiry" ASCII control code. Most VTxxx terminals, including HDSterm, respond to this control code by sending a user-specified answerback message (note that currently the standard MIT Xterm ignores the ENQ code). The answerback message is used infrequently and the default setting for most VTxxx terminals is an empty answerback string.

By putting the echo command at the end of the user's startup script, the echo is executed just as the login shell is ready to accept user input. If the answerback message consists of one or more shell commands followed by a newline, the answerback message will provide the commands to be executed by the shell.

Step 2. Make the command line for the answerback message:

If you wanted to start the Uniplex application from HDSterm, use this command line:

```
setenv TERM vt100; uniplex; exit\n
```

Since HDSterm's answerback message is specified by an X resource, it is possible to customize the message for any given instance of the HDSterm client.

For example, the answerback message shown above does three things.

- It sets the TERM environment variable for the window. Note that the rlogin daemon automatically sets the TERM environment variable to the "vt220", since HDSterm specifies this to rlogind. For various reasons, you may wish to override the TERM variable with a different terminal type for the process group running this specific application.

with no host name specified, which brings up the host chooser menu so you can pick the host you want. This menu has an added "rlogin" button in the bottom section between the "Telnet" and "Cancel" buttons.

2.2.3.3 Auto-Starting an Application in HDSterm

Many ViewStation users would like to customize their environment and make their operations more productive by having an automatic start for their applications in a terminal window. This is possible with the MIT Xterm client, and many ViewStation users would like the same capability in their HDSterm windows.

Xterm Operation

Automatic starting of clients is possible using Xterm and is typically used with a window manager "root menu" to enable the user to kick-off customized Xterms running specific applications.

For instance, a user might want to start the Uniplex application in an Xterm window from the window manager menu. With Xterm, a user would add a line to their .mwmrc RootMenu such as:

```
"Uniplex" f.exec "xterm -e uniplex"
```

When the user clicks on this menu item, a new Xterm window opens and automatically starts a Uniplex process in the window. When the user quits Uniplex, the Xterm automatically quits too.

HDSterm Operation

The principal difference between Xterm and HDSterm is that one is run on the host computer (Xterm) and the other is run on the local display (HDSterm on the ViewStation). The advantages of this local operation are significant, but it does present a limitation in using an autostart operation.

While HDSterm supports the "execute" (-e) command argument to autostart another program to run in the HDSterm's window, it has the same limitation that Xterm has: The autostart program must be a process running on the "local" machine, that is, the machine running the HDSterm or Xterm client. In the case of Xterm, this can be any process running on the host computer. In the case of HDSterm, it can only be a process running on the ViewStation.

This limitation of running clients only on the local machine applies both to Xterm and HDSterm. For instance, if you wanted to run Xterm on host 1 and execute a client on host 2, the autostart command would fail. You might think that you could get around this limitation of Xterm by typing:

```
xterm -e rsh <anotherhost> uniplex
```

but this process would fail too.

Daemon Operations

The problem with this attempt is that the remote shell process (rsh) connects to a remote shell daemon (rshd) on the remote host, which in turn launches the Uniplex program. But the rsh daemon, unlike the rlogin daemon or the telnet daemon (other processes to do this operation), does not create a pseudo-tty for communi-

2.2.3.1 HDSterm Manual Starting

You can also start it directly as a client program. From an HDS shell, you can use the command line:

```
hdsterm -e telnet -T "Telnet"
```

which starts the HDSterm client as a Telnet window. (The -e command executes a command and the -T command sets the name in the title bar.) This is the command you execute when you click on the Telnet button in the main menu of Setup Mode.

If you want to select a particular host for the Telnet session, use the following command line. If you need to specify a specific telnet port number (default is port 13), you can do it on the command line as the third argument, which is shown as the second example.

```
hdsterm -e telnet <hostname> -T "Telnet"

hdsterm -e telnet <hostname> <port #> -T "Telnet"
```

In the same way, you can start other HDSterm clients for other interfaces:

```
hdsterm -e opendev -T "Terminal"

hdsterm -e /bin/sh -CL -T "Console"

hdsterm -e opendev /dev/tty2 -T "Terminal"

hdsterm -e /bin/sh
```

These are the commands for a Serial Terminal window, a Console window, a Serial Terminal window on serial port 2, and an HDSterm window, respectively. Refer to the HDSterm manual pages following for more command line options and details.

2.2.3.2 HDSterm rlogin Capability

HDSterm supports an rlogin capability. In the simplest form, this allows you to start an HDSterm window with an automatic login to a remote host. Note that the correct permissions to do the rlogin must be in place. This is accomplished by the rsh command, which initially establishes the correct environment and permissions for the rlogin capability. For example:

```
hdsterm -e rlogin <hostname>
```

starts an HDSterm window and automatically logs you into the host. You would typically use this command line in your local window manager’s menu (the window manager must have been started by a rsh command).

You can also do this from a remote host with the command:

```
rsh <fxname> hdsterm -e rlogin <hostname>
```

You could also use the command:

```
rsh <fxname> hdsterm -e rlogin
```

specific DEC keyboard functions. Refer to Section 2.2.8 at the end of this chapter for details about the different mappings.

Numeric Keypad/Application Keypad

This selects whether the Numeric keypad transmits numbers or whether it transmits the special DEC command sequences required in Application mode. Some application programs want the special sequences for correct operation. This can also be set with a DEC escape sequence.

Normal cursor keys/Application cursor keys

This selects whether the cursor keypad transmits cursor control commands or special DEC command sequences required in Application mode. This setting is also available by DEC escape sequence.

2.2.2.3 Serial Options Pull-down Menu

The "Serial Options" pull-down menu is shown below:

The "Serial Options" pull-down menu applies to the Serial Port you selected when you opened the Serial Terminal window, and lets you select the type of connection you want to use, such as SLIP, CSLIP, or PPP. It also offers you a quick way to send a Short Break or a Hangup signal.

For more details and examples of Serial Terminal operation, see Section 1.4.4.

2.2.2.4 Help Pull-down Menu

The "Help" pull-down menu contains one entry:

About HDSterm

This menu brings up a brief message about HDSterm containing references to the location of manual pages and other reference materials that apply to HDSterm.

2.2.3 HDSterm Operation

The HDSterm local client is usually run from menus and button selections, either from Setup Mode to start it in one of its interfaces, such as Telnet or Serial Terminal or Console windows, or from its own menus.

This selects whether the cursor is visible or not.

Cursor blink

This selects whether the cursor, of whatever type, blinks or not.

Block cursor/Underline cursor

This selects whether the window’s cursor is a block or an underline cursor.

Normal video/Reverse video

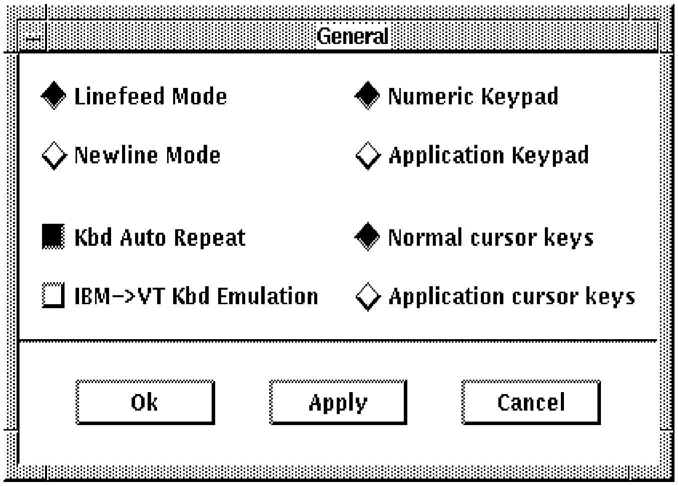
This selects whether the window foreground and background colors are normal or reversed.

Host status line/No status line

This selects whether the window displays a host status line or not. Typically this status line reports error messages or other information coming from the host computer.

■ The "General" Dialog Box

This dialog box selects options for keyboard and text operation for the HDSterm window. These settings are mode switches, either one setting or the other.



◆ Linefeed Mode
◇ Newline Mode
◆ Numeric Keypad
◇ Application Keypad
■ Kbd Auto Repeat
☐ IBM->VT Kbd Emulation
◆ Normal cursor keys
◇ Application cursor keys

Linefeed Mode/Newline Mode

This setting selects whether the Return key sends a linefeed character or a "newline", which is a linefeed and a carriage return. This setting is useful for text editors which may require one or the other for correct operation.

Auto repeat/No Auto repeat

This setting selects whether keys have Auto repeat On or Off.

IBM -> VT Kbd Emulation

This selects whether HDSterm substitutes a VT320 key mapping for users who have an IBM keyboard. This is sometimes useful for editing programs that have

Info/Status Area

This selection lets you remove the Information/Status Line from the HDSterm window. Clicking on the button removes this line. The default setting is to display the line. Default settings for the values are shown in bold in the table below.

The Information/Status line displays several of the current HDSterm settings, as follows:

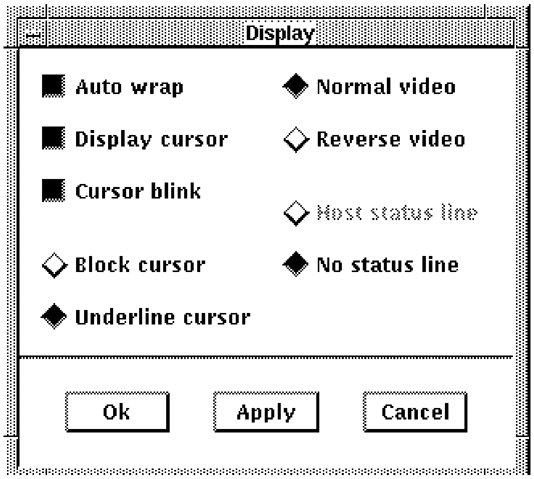
Settings:	**Permitted Values:**
Insert Mode	Insert | **Replace**
Autowrap Mode	**Wrap** | No-wrap
Newline Mode	**LineFeed** | NewLine
Print Device	/dev/*xxx* [**pp0** | tty1 | tty2]
Printer Status	Open | **Closed**
Autoprint Mode	[Logging | ***nothing***]
Printer Controller Mode	[PassThru | ***nothing***]

Small font/Medium font/Large font/Default font

These selections choose the size of the font for the HDSterm window. The actual fonts used for these font selections are set in the HDSterm resource files, which are discussed later.

■ The "Display" Dialog Box

This dialog box selects options for the display within the HDSterm window.



☑ Auto wrap

☑ Display cursor

☑ Cursor blink

○ Block cursor

● Underline cursor

● Normal video

○ Reverse video

○ Host status line

● No status line

The selections are:

Auto wrap

This setting selects whether Auto wrap is On or Off, that is, whether characters automatically wrap to the next line when they reach the end of a line.

Display cursor

2.2.2.2 Customize Pull-down Menu

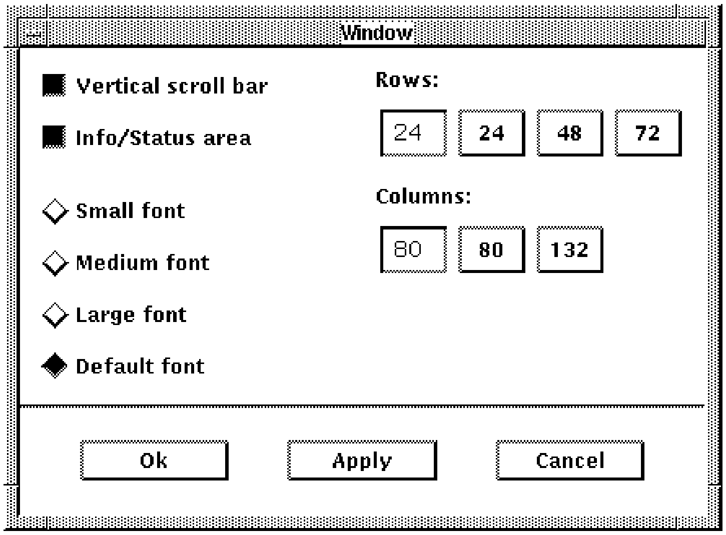
The "Customize" pull-down menu is shown below:

Window ...
Display ...
General ...

The Customize pull-down menu has three dialog boxes, each with a number of selections. The dialog boxes have three buttons at the bottom to control the settings you have made. The "Ok" button selects your settings and closes the dialog box; the "Apply" button applies your settings immediately so you can see their effect; and the "Cancel" button closes the dialog box.

- The "Window" Dialog Box

This dialog box selects features about the HDSterm window itself, such as window size and font.



☑ Vertical scroll bar
☑ Info/Status area
○ Small font
○ Medium font
○ Large font
● Default font

Rows: 24 | 24 | 48 | 72

Columns: 80 | 80 | 132

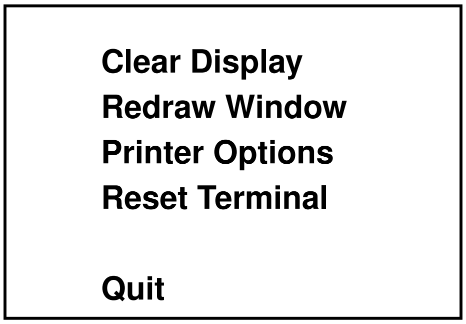
The selections are:

Vertical scroll bar

This selection creates or removes a vertical scroll bar in the HDSTerm window. You can use the scroll bar to move up and down in the window. Utilizing DEC VT320 commands, you can also use the cursor keys for line by line movement or the Shift-cursor keys for screen by screen movement.

2.2.2.1 Commands Pull-down Menu

The "Commands" pull-down menu is shown below:



- Clear Display

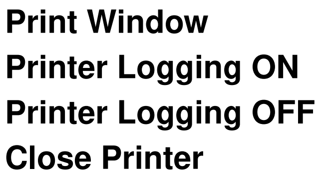
This selection clears the HDSterm window.

- Redraw Window

This selection redraws the HDSterm window.

- Printer Options

This selection brings up a sub-menu which has some options for printing. These options are discussed in greater detail in Section 2.2.7.



Clicking on the "Print Window" line sends the current contents of the HDSterm window to the printer. This corresponds to the VT320 command CSI i.

The "Printer Logging" lines refer to the VT320 Autoprint function, corresponding to the CSI ? 4/5 i commands.

The "Close Printer" line empties the printer buffer and closes the connection to the printer.

- Reset Terminal

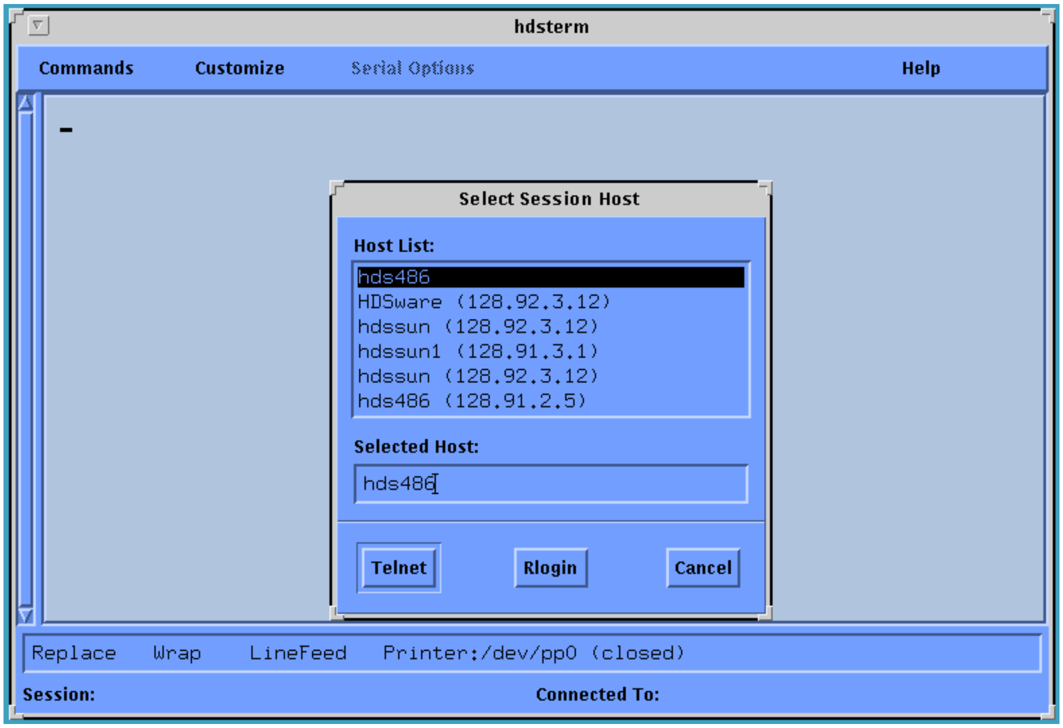
This selection resets the HDSterm window.

- Quit

This selection closes the HDSterm window.

2.2.1.1 HDSterm Window

The HDSterm window with the Telnet selection menu is shown below.



Host List:

```
hds486
HDSware (128.92.3.12)
hdssun (128.92.3.12)
hdssun1 (128.91.3.1)
hdssun (128.92.3.12)
hds486 (128.91.2.5)
```

Selected Host: hds486

```
Replace   Wrap   LineFeed   Printer:/dev/pp0 (closed)
```

Session:

Connected To:

The HDSterm pull-down menus contain options for operating and configuring the window. HDSterm is loaded automatically if you start the ViewStation with server code Vnn-D.hds.

2.2.1.2 Using the tty Telnet Client

If you have selected a server code version that doesn't load HDSterm, your Telnet selection will use the ViewStation's tty client with a limited ANSI emulation.

The tty Telnet client is loaded automatically if you start the ViewStation with server code Vnn.hds.

2.2.2 HDSterm Functions

The HDSterm window has a number of functions and settings which are available from pull-down menus.

HDSterm Terminal Emulation Client

This section describes HDSterm, the ViewStation's local terminal emulation client, including its properties and customizing options.

Section 2.0 - HDS ViewStation Local Clients

2.2 HDSterm Terminal Emulation Client

HDSterm is the ViewStation's terminal emulation client. It provides DEC VT320 terminal emulation in a graphical user interface window. HDSterm offers substantial enhancements of the VT320 terminal emulation, including local printer support and DEC color command support.

There are also other ViewStation terminal emulations discussed in Section 2.3.

2.2.1 Using the HDSterm Client

The HDS ViewStation client HDSterm serves as a Telnet window, a Serial Terminal window, and a Console window interface.

This terminal emulation gives you additional capability in the window, including VT320 capabilities such as Numeric and Application keypad functions, cursor keys for single line scrolling (and Shift-cursor keys for scrolling screen-by-screen), cursor selection, and so on. A complete list of these features appears later in this chapter. Many of the features can be selected by the pull-down menus, or are selectable with DEC VT320 command sequences, also listed later in this section.

The HDSterm window appears when you select either the Telnet Terminal Window or the Serial Terminal Window or the Console Window from the Setup Mode Main Menu. For Telnet, you have a dialog window offering you a choice of connecting with any of the hosts in your Remote hosts list, with the Default Telnet host highlighted for easy selection. For the Serial Terminal window, you are prompted for the serial port you want to use. The Console window appears with current output of diagnostic messages.