```
:<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 \
```

1.8.5.7 Special Motif Bindings for HDSterm Keys

Some HDSterm keys have special definitions in the Motif widget set, such as F1, F4, and F10 (see the table in Section 2.2.9.1 for a complete list). If you try to modify these keys you must refer to them by that name in the translation table, as shown in the example above. You should also consider the consequences of these changes on the behavior of Motif window manager.

In the special case of F10, which is an accelerator key, you must also explicitly disable the key before you redefine it in order to disable the accelerator function, which normally overrides the key translation. In the example is given above, the osfMenuBar key is disabled as the first line of the file, then redefined in the body of the translation table.

redefined as part of the set. Keys that do not belong to a special set can be redefined with normal key override translations, the “*screen.translations” resource. Full key listings are shown in Section 2.2.9.1 of the *HDS ViewStation System Administrator’s Guide*.

Ordinary Keys - For ordinary keys, you 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 the following (but note the use of the osfBackSpace name for the keysym):

```
HDSTerm*screen.translations:  #override \
<Key>osfBackSpace:  key-input("\177")
```

Keyset Keys - If the key you want to redefine is in one of these keyset groups, you must use the entire keyset in your translation. This is necessary because the new keyset resource **completely replaces** the built-in default for that resource. Therefore, you may not use an "#override" directive to list only those keys you want changed. You must include all keys which have translations defined in the keyset. For ordinary keys, you may use the "#override" directive on the *screen.translations resource, as these are applied at widget initialization, as in the example above.

1.8.5.6 Customizing 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: ""
! this redefines the ‘fkey’ group for HDSTerm keys
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 \
```

```
more HDSTerm
```

This is the HDSterm *app-defaults* file. It has the HDSTerm resources (note the uppercase T in the resource name) and then the keymap definitions. You can put these modified resources in your *.Xdefaults* file in your home directory, or any other location you read resources.

1.8.5.4 HDSterm Keymap Groups

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-auxTrans-<set name>

The following sets of translations are used (see full listings in Section 2.2.9.1 or in the file itself):

Key-set Name	Translation Application
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 way to understand these translations is to follow 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.

1.8.5.5 Customizing HDSterm Keymaps

To customize the keyboard translations, you must first find the keyset containing the keys you wish to change. Keys that belong to one of the special sets can be key such as F10 (an event), generates a specific, defined response, which would be the "translation". These event translations are defined in a table, which is used as a resource for each individual client.

A translation table is handled as if it were a single resource; the "override" indicates that only the following definitions override the default definitions. A sample translation table might be:

```
XTerm*VT100.Translations:  #override\
<Key>F1:string ("lpr -Pps") string (0x0d)\n\
<Key>F2:string ("xwd | xpr -device ps | lpr") string
(0x0d)\n\
<Key>F3:string ("ls -C") string (0x0d)\n\
<Key>F4:string ("Mary Smith, Comptroller, XYZ Corporation")
```

Note these syntax requirements:

- The resource needs the "#override\" entry to show they are new translations for these keys.
- Strings are enclosed in parentheses; strings with spaces or special characters also need double quotation marks as delimiters.
- Commands that require a carriage return for execution have the hex value for CR entered as a string at the end of the entry.
- The table must be a single entry, so the `\n\` must appear at the end of each line except the last.
- This table is a single resource in your .Xresources file.

This example shows a number of applications for event translations. Note that the resource specification is for "Xterm*VT100" so it applies to all xterm clients that are started. It would not apply to other applications.

It shows some useful definitions, such as F1 doing a printer command, and F3 doing a list command. F4 prints a simple string for a frequently used phrase. F2 shows a more complicated command for a screen dump to a postscript printer.

Using resources and event translations is discussed in a number of X reference books. Refer also to the man pages for xrdb (and your clients), and also to Section 3.4 of this manual.

1.8.5.3 HDSterm Event Translations

HDSterm keymappings are described in Sections 2.2.9ff of the *HDS ViewStation System Administrator's Guide*. Refer to those sections for details and examples.

There are two special things to consider in modifying HDSterm keymappings:

- HDSterm uses sets of keys with separate translations for each set
- HDSterm has some keys defined as Motif special keys

The default mapping is a text file which you can see (or cut and paste) from the ViewStation's Console window with these commands:

```
cd HDS$$ROM
```

labels associated with each key and function. There are commands in xmodmap to print the current keyboard mapping and also help commands. Refer to other X manuals for more details and examples.

Note also that these keyboard settings stay in place until the ViewStation is reset. This can be a problem if the ViewStation is used by different users with different customized xmodmap files. If user A uses xmodmap to change the Caps Lock and the CTRL keys and user B uses it to change only the Delete key, user B will find user A's settings still in place when he starts working on the keyboard. By selecting the "Reset server when last client exits" field in the User Preferences menu of Setup Mode, the xmodmap settings of one user are cleared when his session ends.

Multinational Keyboards and Mappings

Multinational keyboards present some additional functions and considerations. The ViewStation's Setup Mode, User Preferences menu, offers a number of multinational keyboard types that the ViewStation supports. These keyboard selections select particular internal keymaps for the ViewStation.

Since these keymaps need to be in place very early in the ViewStation's startup cycle (such as entering the user's name in an Xdm login banner), the keyboard type should be set in the configuration file. Additionally, since xmodmap is a ViewStation local client, it can be run from a server-reset file to provide additional keyboard customization.

Some multinational keyboards use four key legends per keytop. The upper and lower are shifted/unshifted values; the right and left pairs are selected with the "mode-switch" key (typically Alt_R). Note that this mode-switch function must be defined and then the key defined as a modifier key. The mode-switch key is assigned to keycode 151 (Alt_R). You must then make this a modifier function:

```
mod1 Alt_R(0xb2) mode-switch(0xnn)
```

This is all done automatically for the ViewStation, but you can make additional customizations in this way.

1.8.5.2 Keymap Changes with Event Translations

The Event Translation changes are localized, rather than global, changes. They apply to one client or one window only. For instance, an email window might have the F1 key function as a "Send" command, while your word processor might have F1 function as "Save". These definitions are all defined as resources of the client itself. They are entered in the client's resource file *./app-defaults*, or in your resource file *.Xresources* or *.Xdefaults*. It doesn't matter where the resource is defined (except for its global or local applicability), but you must be sure the resource is always read when the client starts.

There is a special kind of resource is called an Event Translation. This is quite a simple concept, despite its name. It simply means that one event (such as a key press or a button click) is translated into another. The most obvious example is that of a definition programmed onto a function key. In the X environment this process is handled as an "event translation", which means that pressing a function

Control functions are removed, then keysyms assigned, then the Lock and Control functions are added.

```
! Xmodmap data file which swaps the CapsLock and Left Ctrl keys.
! Usage: xmodmap    .xmodmap-swap-keys
!
remove Lock = Caps_Lock
remove Control = Control_L
keysym Control_L = Caps_Lock
keysym Caps_Lock = Control_L
add Lock = Caps_Lock
add Control = Control_L
```

Keysym Changes

Other changes often used are assigning the Backspace function to the Delete key, which is done as follows:

```
xmodmap -e "keysym BackSpace = Delete"
```

Note that you could also make this change in your xterm resource file, in the Event Translation table, or you could make it with a Unix stty command in your .login file.

Note that this entry changes one keysym to another. This is the recommended procedure. Users sometimes modify the modmap table that links a keycode to a keysym. This procedure is NOT recommended because it links the new keysym to a particular keycode (hardware type). If you use that keymap with another keyboard, it may not work. The keysym/keysym translation will always work.

These commands can be used on an individual xmodmap command line or grouped in a file which is then read by xmodmap. Xmodmap is typically read from an .xsession file or some other startup file. Note that the xmodmap client is available as a ViewStation local client. You could use the local xmodmap in a server-reset file (described in Section 1.2.8.1) to make your keyboard modifications even before you log in. This is an advantage to users with special keyboard requirements because even the username and password will use the modified keymapping (this is not true of host-based xmodmap).

Be careful with your changes so you are certain the specified key function behaves the way you want in all environments. These keyboard settings need to be considered carefully and depend on the client and client window and their definitions. Another consideration is the timing of the xmodmap execution. Since it is a global change, you probably want it to cover all clients at all times. This makes an xdm .xsession file or a .login file or a ViewStation server-reset file a good place to run xmodmap.

The xmodmap utility is very powerful. You can construct an xmodmap file for each user, or for the whole system as needed. Be careful with the syntax and the or other modifier keys, etc.) on the X terminal. This keyboard configuration is handled by a server file called xmodmap.

At the same time, the "F24" keysym may be interpreted by clients in any way they wish (such as "Home the cursor" in one application, "Insert marker text" in another, and "<string = your email address>" in another). This keysym interpretation is handled by a client resource called an event translation. This design lets any user produce an "F24" keysym, with whatever keyboard configuration, and have F24 produce the expected behavior on any application.

For the user wanting to modify the keymapping, there are two avenues of approach:

- Change xmodmap (the keyboard configuration file to produce the keysym on the X terminal), or
- Change the event translation file (to interpret the keysym in the client application)

There are some considerations that will influence your choice.

Changing xmodmap (the Keyboard Mapping on the ViewStation)

You can change the ViewStation's keymapping (both keyboard and mouse), but these changes will apply to everything you do. This may be desired in the case of switching mouse buttons for a left-handed user, or using a multinational keyboard. However, for most cases, this change is too broad and may introduce unexpected behavior and problems.

Changing the event translation (the Keysym Interpretation on the Client)

In most cases, this is the desired action since it applies to a specific application and key behavior. This change requires modifying the event translation resource for the client in your .Xdefaults file.

Both of these methods are described below, with examples.

1.8.5.1 Keymap Changes with Xmodmap

The xmodmap utility allows you to specify definitions and functions for each key on the keyboard. These are global changes.

Mouse Button Changes

For example, you can change the function of the mouse keys to make it easier for left-handed users to operate the mouse. Normally the most frequently used mouse key is the left button, which corresponds to the index finger of a right-handed user. To reverse the button order (to use the index finger of a left-handed person) use this entry in the xmodmap file:

```
xmodmap -e "pointer = 3 2 1"
```

Modifier and Lock Key Changes

Another commonly requested change is to swap the CapsLock key with the Left Ctrl key (note that some keyboards have a DIP switch to make this change). This example is taken from the manual pages for xmodmap. Note that the Lock and

1.8.4 Xev Key Monitoring

You can use the Xev (X Event) utility program (this is a standard X client) to see what keycodes and keysyms are actually being transmitted. When you start Xev, place the cursor in the event box, and then press a key, or move the mouse, etc. You will see the event logged on your screen with a lot of information about the event. This example shows the keycode, the modifiers’ state, and the keysym. This helps you analyze what is being sent and how the assignments are made.

This is a sample Xev event that reports the sequence <Press Shift_L>, <Press a>, <Release a>, <Release Shift_L>, which produces “A”:

```
KeyPress event, serial 15, synthetic NO, window 0x3800001,
 root 0x29, subw 0x3800002, time 95819540, (47,49),
root:(112,135), state 0x0, keycode 174 (keysym 0xffe1,
Shift_L), same_screen YES, XLookupString gives 0 characters:
“”
KeyPress event, serial 15, synthetic NO, window 0x3800001,
 root 0x29, subw 0x3800002, time 95821980, (47,49),
root:(112,135), state 0x1, keycode 194 (keysym 0x41, A),
same_screen YES, XLookupString gives 1 characters: “A”
KeyRelease event, serial 15, synthetic NO, window 0x3800001,
 root 0x29, subw 0x3800002, time 95822180, (47,49),
root:(112,135), state 0x1, keycode 194 (keysym 0x41, A),
same_screen YES, XLookupString gives 1 characters: “A”
KeyRelease event, serial 15, synthetic NO, window 0x3800001,
 root 0x29, subw 0x3800002, time 95823630, (47,49),
root:(112,135), state 0x1, keycode 174 (keysym 0xffe1,
Shift_L), same_screen YES, XLookupString gives 0 characters:
“”
```

Note the keycode and keysym information. The “state” refers to the state of the modifier keys (Shift = 1, Ctrl = 2, Alt = 4, and so on). The XLookupString interprets the keysym for you.

1.8.5 Key Handling in a Client/Server Environment

One objective of Client/Server architecture is to handle configurations such as keyboard layouts or keypress assignments separately, either on the server or on the client, so communication between the two is independent of the actual configuration of either. What this means for keyboards and keymaps is that the X terminal server handles everything about the keyboard configuration (hardware connections, keyboard layout, etc.) and produces a resulting key function called a “keysym”. This keysym is a neutral function, such as “A” or “Backspace” or “F4”. The client application receives this keysym and interprets it according to its own configuration (F4 may mean “Save and Exit” or “Next Page” or something else).

In this way a keysym, such as “F24”, may be generated by any keyboard configuration (such as different keyboard layouts, multinational languages, use of Shift

```
212 0x0020 (space)
213
214 0x0035 (5) 0x0025 (percent)
215 0x0074 (t) 0x0054 (T)
216 0x0067 (g) 0x0047 (G)
217 0x0062 (b) 0x0042 (B)
218
219 0x0036 (6) 0x005e (asciicircum)
220 0x0079 (y) 0x0059 (Y)
221 0x0068 (h) 0x0048 (H)
222 0x006e (n) 0x004e (N)
```

From this excerpt you can see that keycode 222 refers to keysyms "n" unshifted and "N" shifted. The columns indicate the keysym with the shift key. Other modifiers, if defined, are used the same way.

1.8.3.3 Modifier Keymappings

This is the modifier mapping for the ViewStation.

```
xmodmap: up to 2 keys per modifier, (keycodes in parenthe-
ses):
shift Shift_R (0xab), Shift_L (0xae)
lock Caps_Lock (0xb0)
control Control_L (0xaf), Control_R (0xb3)
mod1 Alt_L (0xac), Alt_R (0xb2)
mod2
mod3
mod4
mod5 Num_Lock (0x84)
```

This modifier map reports the current modifier settings.

1.8.3.4 ViewStation Pointer Mappings

This is the ViewStation's pointer mapping.

```
There are 3 pointer buttons defined.
 Physical    Button
 Button      Code
 1           1
 2           2
 3           3
```

This keycode (sent from the keyboard to the ViewStation) is matched to a key symbol (a character or function) in a table called the xmodmap table. This table is visible to the user by the xmodmap command:

```
xmodmap -pk (this prints the keymap table)
xmodmap -pm (this prints the modifier table)
xmodmap -pp (this prints the pointer table)
```

Refer to Section 2.6.13 for more details on xmodmap.

1.8.3.1 Keycode Modification Tables

This xmodmap table of keycodes can be modified by the user if needed, but this is usually not recommended. Modifications to the keymap can be made in the keycode translation file in *./hds-fx/config/<keycode file>*.

This keyboard translation mapping is for sites using keycodes in their xmodmap files (this is not recommended - keysym to keysym remappings are preferred and do not require keycode remapping). In order to modify the keycodes for any keyboard, set this resource in the ViewStation's configuration file:

```
HDSsetup.KeycodeTranslation.path: /FXUtilities/hds-fx/config
```

This specifies the directory for the KeycodeTranslation file, whose name is the hexadecimal keyboard identifier.

For example:

IBM-101+ keyboard identifier is AB83

Sun4-style keyboard identifier is AB04

The keyboard identifier is a property of the root window, so if you don't know the identifier, you can look at the properties of the root window for its value (use the xprop command and click on the root window).

1.8.3.2 Key Mappings

Xmodmap is an X client on your host computer (it is also a ViewStation local client) that reports your keyboard mappings. Refer to the xmodmap manual pages for more examples and command syntax.

This is a sample of an xmodmap -pk print. It is a section of the file indicating the kind of thing you'll see in the file.

```
203 0x0033 (3) 0x0023 (numbersign)
204 0x0065 (e) 0x0045 (E)
205 0x0064 (d) 0x0044 (D)
206 0x0063 (c) 0x0043 (C)
207
208 0x0034 (4) 0x0024 (dollar)
209 0x0072 (r) 0x0052 (R)
210 0x0066 (f) 0x0046 (F)
211 0x0076 (v) 0x0056 (V)
```

The default assignments for the hotkeys are as follows:

Function	IBM kybd	DEC kybd	Sun4 kybd	Sun5 kybd
Setup Mode Hotkey	Alt-SysRq	Alt-Help	Alt-PrSc	Alt-PrintScreen

1.8.2.3 Monitor Mode Keystrokes

When you first turn on the ViewStation, the white screen you see is called the Boot Monitor. If you want to go directly to Monitor Mode to make your settings, you can press the “m” key while the “Initializing” message is displayed (3 seconds); otherwise you have to wait until the boot process finishes in order to get into Monitor Mode.

There are two other keystrokes that can be useful, a “Pause” function to pause (and resume) the boot process, and a “Debug message” function to display more detailed messages (two levels of debug messages, activated with one or two keypresses).

Function	IBM kybd	DEC kybd	Sun4 kybd	Sun5 kybd
Pause (Resume)	Pause	F1 (Hold)	Pause	Pause
Debug messages (1x or 2x)	PrntScrn	Help	PrSc	PrintScreen

1.8.2.4 HDSterm Environment Variable Keystrokes

You have to set your environment variable when you login into a host computer so the computer can send the application client windows to the proper display. The ViewStation’s Telnet clients (HDSterm and tty Telnet) have special key definitions to make this setting easy for you. These definitions automatically insert the IP address of your ViewStation.

There are settings for several different environments.

Function	IBM kybd	DEC kybd	Sun4 kybd	Sun5 kybd
Env. variable (C shell)	Alt-F12	Alt-F12	Alt-Again	Alt-Again
produces: “setenv DISPLAY <*IPaddress*>:0”				
Env. variable (Bourne shell)	Alt-F11	Alt-F11	Alt-Stop	Alt-Stop
produces: “DISPLAY=<*IPaddress*>:0;export DISPLAY”				
Env. variable (VMS hosts)	Alt-F10	Alt-F10	Alt-F10	Alt-F10
produces: “set display/create/server=0/trans=tcpip/node=<*IPaddress*>”				

1.8.3 Keyboard Operations

The keyboard is sensed automatically when it is connected. If it is unplugged and then reattached, the ViewStation automatically re-initializes the keyboard. You can also use the “keyboard” command from the Console window to reset the keyboard.

As part of its initialization, the ViewStation queries the keyboard for its type and the keyboard replies with an identifier code. The ViewStation matches this code to a table of keycodes appropriate to that keyboard.

1.8.1.1 Keyboard Connections

The standard keyboard connector is a five-pin DIN male connector with an IBM AT interface, see Section 1.1.10.2.

Sun keyboards use a separate connector on the back panel.

1.8.2 Keystroke Functions on the ViewStation

As a convenience to users, the HDS ViewStation uses a number of special keystrokes for various functions. These are listed below, along with their variations on optional keyboards. These key definitions apply only in the clients indicated and only under the correct circumstances.

1.8.2.1 Keyboard Cursor Mode

The ViewStation can use its keyboard cursor keys to simulate the mouse movement. This is called "keyboard cursor mode" and is used in environments where a mouse is not practical. Keyboard cursor mode is activated by an entry in the configuration file, config.fx:

```
HDSsetup.keyboard_cursor_mode: [ ENABLED | DISABLED ]
```

When keyboard cursor mode is enabled, the cursor keypad functions as a mouse. The arrow keys move the cursor (note the increasing acceleration of the mouse cursor when the key is held down), and the lower three keys of the editing keypad function as mouse buttons.

When keyboard cursor mode has been enabled in the configuration file, you can toggle the action by using the Alt-Ctrl_R keys. These keys toggle keyboard cursor mode On and Off; the Scroll Lock LED lights when it is enabled.

1.8.2.2 Setup Mode Keystrokes

Setup Mode Editing Keys

The editing keypads on the keyboards are active in Setup Mode on most keyboards.

Function	IBM kybd	DEC kybd	Sun4 kybd	Sun5 kybd
Cursor keys	yes	yes	yes	yes
Insert function	Insert	InsertHere	Ins	Insert
Delete function	Delete	none	Del	Del
Home/PageUp	Home/PgUp	Prev	none	Home/PageUp
End/PageDown	End/PageDn	Next	none	End/PageDown

Setup Mode Hidden Icon Hotkey

If you have the Setup Mode resource set to "Hide Icon", the Setup Mode icon doesn't appear on the screen unless you hit the Setup Mode hotkey. This is a definable key sequence which you can set with the hotkey resource in the Setup client's defaultArgs table in the *./hds-fx/config/clients.config* file. In this example the new hotkey would be Shift-Alt-F1:

```
hds.client.setup.defaultArgs: -xrm ‘*hotkey: Shift Alt<Key>F1’
```

HDS ViewStation: Keyboards and their Operations

This chapter discusses ViewStation keyboards and their operation. Some customization is also discussed.

Section 1.0 - HDS ViewStation

1.8 Keyboards and their Operations

Keyboards are the primary user interface to the desktop device. Users have many requirements and preferences, so keyboards and their operation and configuration are important considerations.

1.8.1 HDS ViewStation Keyboards

The HDS ViewStation supports a number of keyboards, shown below.

- IBM 101+ style keyboard (AT type) - this is the standard keyboard
- DEC VT420 type
- IBM 102 key multinational keyboard
- Lexmark 122 key IBM 3270 keyboard
- Sun Type 4 style or Type 5 style (using either Serial Port 1 with special modifications for older ViewStations) or with a Sun keyboard connector on FXD and Ultra ViewStations
- Multinational keyboards of various configurations are also supported.

All keyboards use a five-pin DIN male connector with an IBM interface. Complete key remappings and configurations are selectable from software.

Other keyboards that support the IBM AT keyboard interface will probably work with the ViewStation, but are not supported without HDS's testing and approval. They may also require special user-defined keymap tables.