AT&T

AT&T 3270 Emulator+
System Administrator's Guide

NOTICE

The information in this document is subject to change without notice. AT&T assumes no responsibility for any errors that may appear in this document.

Table of Contents

5   Terminal Customization

A   Documentation

B   Variables

C   Sample Keyboard Source File

D   Sample Screen Control Source File

E   Key Sequences

F   Sample kyinit Report File

G   Non-supportable Terminal Messages

H   The te3279 Command

Index

List of Figures

About This Guide

This book describes the procedures required to customize, administer, and run the AT&T 3270 Emulator+ software.

The AT&T 3270 Emulator+ can operate using either Binary Synchronous Communications (BSC) protocols or Systems Network Architecture (SNA) protocols (SDLC). Depending on the mode of operation, the environment and the controller configuration file must be set up for either SNA or BSC operation before running the emulator. Set-up procedures for each mode of operation are similar, but certain parameters, filenames, and environment variables that are used differ.

This guide is divided into five chapters:

1. "SNA Operation" provides instructions for configuring and running the software in an SNA environment. The AT&T SNA/3270 Emulator+ software must be installed on your 3B Computer before you use these procedures.

2. "BSC Operation" provides instructions for configuring and running the software in a BSC environment. The AT&T BSC/3270 Emulator+ software must be installed on your 3B Computer before you use these procedures.

3. "Running the 3287 Printer Emulator" explains how to invoke the **pe3287** process.

4. "Setting Up the User's Environment" describes how to set up the user's environment so that the user need not enter most **te3279** command options when starting the 3278/9 Terminal Emulator. It also explains what information the System Administrator must give to each terminal user.

5. "Terminal Customization" provides instructions for customizing the keyboard and screen files provided with the AT&T 3270 Emulator+ software and creating new files for terminals not currently supported.

    Before running the SNA Controller, the System Administrator must configure the environment and create a controller configuration file according to local and host requirements.

Configuring the Environment

The AT&T 3270 Emulator+ uses environment variables as pre-defined defaults for command options. This means that when an option to a 3270 Emulator+ command is omitted, the list of environment variables is searched for a specific variable associated with that option. If that variable is set, its value is used as the option to the command. Therefore, if the appropriate environment variables are set, the user need not specify most options when issuing a command, unless the user chooses to override the defaults.

Configuring the environment involves setting the environment variables used by the 3270 Emulator+. For convenience, two shell scripts, **snaenvset** and **snaenvcust**, are provided to set these variables to default values.

Creating a Controller Configuration File

The controller configuration file contains the parameters the controller processes need to communicate properly with a particular host. When the SNA Controller is started, the configuration file is used as input to customize the controller processes with the options specified in the file.

Alternate configuration files can be created for different remote host and local configurations.

1 SNA Operation

Configuring the Environment for SNA

The AT&T SNA/3270 Emulator+ processes refer to various environment variables during execution. These variables are used to provide information to the processes, such as default values for command options and specific system start-up information.

You must configure the environment for SNA/3270:

■ Each time you log in, before you begin to use the SNA/3270 software

■ Before starting or stopping an SNA Controller

Two shell scripts, **snaenvset** and **snaenvcust**, both in the /usr/snaadm/runtime directory, may be used to configure the environment for SNA operation. (Note that **snaenvset** calls the **snaenvcust** script.) Appendix B lists and describes the environment variables set by these scripts. It also provides the initial default values that the scripts assign to the variables.

Invoking snaenvset

Before configuring the environment, you should check that the default values set by the scripts are appropriate. If you must change any of these values, refer to the section Changing Environment Variable Defaults.

If the default values set by the scripts are appropriate, you can configure the environment for SNA operation by executing **snaenvset** as follows:

Step 1.   Log in as root.

Step 2.   Change directory to /usr/snaadm/runtime.

Step 3.   Enter

    . ./snaenvset

Step 4.   Use the UNIX system commands **env**(1) and **set**(see **sh**(1)) to check your environment. (See the *UNIX System V User's Reference Manual* for information about these commands.)

The **snaenvset** script has to be executed only once per log in. To have **snaenvset** execute automatically when you log in, add the following entry to your **.profile** file:

. ./usr/snaadm/runtime/snaenvset

Changing Environment Variable Defaults

If the local configuration makes it necessary for you to change the values of any of the environment variables, you should be aware of the following:

- **PCD** specifies the communications port number that the SNA Controller is to use. You must change this variable if you want the controller to use a port other than that provided by the **snaenvset** script.

  □ On a 3B2 Computer, the port denoted by **PCD** is the slot number that the ISC card is in. For example, if the ISC card you want to use is in slot 4, set **PCD=4**.

  □ On a 3B5 or 3B15 Computer, **PCD** denotes the path to the SDLI port. For example, to use the third SDLI port on the second IOA, set **PCD=203**.

  □ On a 3B4000 Computer, **PCD** is a combination of the **pe** and ISC slot number (e.g., 120.6 for **pe** 120, slot 6).

    The default value for **PCD**, as set by **snaenvset**, is stored in a file called **PCD.file** in the **snaadm** directory.

- **SNAHOST** is used to change the **CONFIG** variable, which specifies the full pathname of the controller configuration object file to be used when the SNA Controller is started. Configuration object files are named with the prefix **CSNA.**, for example, **CSNA.hostx**. **SNAHOST** indicates which "CSNA." file is to be used. For example, if **CSNA.hostx** (located in **/usr/snaadm/runtime**) is to be used, **SNAHOST** should be set equal to **hostx**. Then, when the **snaenvcust** script is executed, **CONFIG** will be set to **/usr/snaadm/runtime/CSNA.hostx**.

- **P3274** specifies the named pipe the terminal and printer devices use to communicate with the controller. This pipe must have a unique name for each controller running on the system.

Temporary Changes

To change the value of an environment variable temporarily:

Step 1.   Execute **snaenvset**, as described in the section Invoking
            snaenvset. (This has to be done only once per log in.)

Step 2.   Reset the environment variable(s) that must be changed. For
            example, to specify that the controller should use the
            configuration object file named **CSNA.hostb** (instead of the
            default **CSNA.teke**), you would enter:

            SNAHOST=hostb

Step 3.   Execute **snaenvcust** by entering:

            . ./snaenvcust

            The **snaenvcust** shell script will re-customize the environment
            using the new values.

Permanent Changes

To change the value of an environment variable permanently:

Step 1.   Use a text editor such as **vi** to edit the **snaenvset** shell script,
            changing the setting of the environment variables as necessary.

Step 2.   Execute **snaenvset** as described in the section Invoking
            snaenvset.

Creating an SNA Controller Configuration File

You must create a configuration file for each different host and local configuration an SNA Controller may use.

It takes three steps to create an SNA Controller configuration file. You must:

Step 1.   Determine the configuration options needed to customize the SNA Controller for your particular site and host.

Step 2.   In the **/usr/snaadm/cust/snaoptions** directory, create an SNA configuration source file in which each option is defined as required.

Step 3.   Create an SNA configuration object file. This involves processing the configuration source file through the **snopts** utility. When the **snopts** utility is executed from the **/usr/snaadm/runtime** directory, it creates an object file from your source file and displays the completed configuration, including the values for any defaulted options.

You can also invoke **snopts** interactively, in which case you do not use a source file as input. Instead, you enter the configuration options directly into the **snopts** program. However, using a source file allows you to retain the exact configuration for later use.

SNA Configuration Options

To determine the options required for the SNA Controller configuration file, consult the Network Administrator at your host site for precise information regarding your configuration. The options you define in the configuration source file should coincide with your host configuration.

Configuration options for the SNA Controller are categorized as follows:

Line Options: These options are used to ensure compatibility between the physical communication media on both sides of the link. They define communication line characteristics such as full-duplex or half-duplex operation and NRZ or NRZI operation.

Controller Options: These options define the controller (PU) type and its associated devices (LUs). There must be agreement between the host and the controller concerning the identity and nature of all the devices (terminals and printers) associated with the controller.

Dual-identity SDLC: The AT&T SNA/3270 Emulator+ software is designed to run concurrently with the AT&T 3770 Emulator+ (Release 2.0.0 and later releases). Assuming the AT&T 3770 Emulator+ has been installed on your 3B Computer, the dual-identity SDLC feature of the SNA Controller (which is a component of both software packages) allows a 3770 SNA/RJE workstation and a 3270 workstation to operate simultaneously using one SDLC link to the host.

To implement dual-identity SDLC, the SNA Controller must be configured with one 3770 type PU, and one 3270 type PU. Two separate sets of controller options must be defined in the configuration file, one set for each controller (PU) type. It is necessary for the 3770 PU definitions to occur first in the file. However, only one set of line options is needed— both PUs use the same SDLC link. In addition, the line must be defined at the host as a multi-drop line with a station address for each PU.

For information on SNA/RJE operation and on configuring the controller options for a 3770 controller (PU) type, refer to the *AT&T SNA/RJE Emulator+ System Administrator's Guide*.

| NOTE | The AT&T SNA/3270 Emulator+ and the AT&T SNA/RJE software can also be run independently, using separate SDLC links. |

Figure 1-1 lists the valid SNA configuration source file options and their corresponding defaults.

Option	Function	Default
Line-Options		
fdx	Sets full-duplex mode.	hdx
multipoint	Sets multi-point operation.	pt-to-pt
nrzi	Sets NRZI mode of transmission.	nrz
Controller-Options		
controller *aaaa*	Sets controller (PU) type to *aaaa*, where *aaaa* is 3270.	(See option description.)
luxx *t* [*termself*]	Assigns the LU numbered xx to device type *t*	LU is not defined.
	valid values for xx are 0 - 31	
	valid values for *t* are	
	1 — Printer (SCS or DSC) 2 — Display Station	
	(SNA distinguished between SCS and DSC printers dynamically, by means of the session parameters in the BIND command.)	
	termself is an optional parameter. Specifies that a TERM_SELF be sent to the host whenever the LU session is terminated. (Provides high security for the LU session.)	LUSTAT is sent. (low-security LU)

Option	Function	Default
xid *xxxxx*	Sets last 5 digits of the XID to *xxxxx* (hex).	00000
address *xx*	Sets SDLC station address to *xx* (hex).	01
seg *n*	Sets SNA segment size to *n* (decimal).	265
broadcast	In a multi-station configuration, designates this station as the one to respond to broadcast messages from the host.	Not a broadcast station.

Figure 1-1: SNA Configuration Source File Options

Support for an 8100 Controller

In order to emulate an 8100 controller, the dual identity feature of the SDLC must be used along with the following configuration information:

Line options:	fdx pt-to-pt nrz
Controller options:	controller 8100 xid 020000600000 seg 265 address C3 lu2 3 lu3 3 lu4 3 lu5 3

lu6 3
lu7 1
lu8 1
lu9 2
lu10 2
lu11 2
lu12 2
lu13 2
lu14 2
lu15 2
lu16 2
lu17 2

Source File Options

The following is a functional description of each of the options listed in Figure 1-1.

Line Options

fdx/hdx
Full-duplex (four-wire) operation allows two stations to transmit and receive at the same time. In half-duplex (two-wire) operation, the stations take turns transmitting, using pin-level (RTS, CTS) controls. (Dial-up lines are half-duplex.) SDLC is strictly a half-duplex protocol, so full-duplex traffic cannot occur even with four wires. However, when full-duplex is specified in a point-to-point configuration, even though true four-wire traffic is not possible, modem line turnaround delays are eliminated because RTS is always kept high. The default is **hdx** (half-duplex).

multipoint/pt-to-pt
A multipoint configuration consists of two or more secondary stations communicating with a single primary station (the host). The SDLC station address is checked to determine if the message is for an enabled station. A point-to-point configuration consists of one secondary station communicating with one primary station. This line option must be defined as **multipoint** if the dual-identity SDLC

feature is to be implemented. (Dual-identity SDLC allows two controller types to operate concurrently using one SDLC link. See the **controller** option below.) The default value is **pt-to-pt** (point-to-point operation).

nrzi/nrz

Non-Return To Zero Inverted (NRZI) transmission is used to reduce the probability of losing bit synchronization. Because bit synchronization is maintained by the pace of the bit stream itself, periodic signal polarity transitions are necessary. In NRZI transmission a polarity change occurs whenever a binary zero appears in the message. Since "bit stuffing" ensures that there will never be more than six contiguous bits with values of one, there must be a zero, and hence a polarity change, at least every six bits. Because it may be prohibited by specific hardware limitations on some data communications equipment, NRZI transmission is optional. The default value is **nrz** (NRZ transmission).

| NOTE | Only one set of line options should be defined in each source file. With dual-identity SDLC, the SNA Controller is configured with two controller types, but only one set of line options is used, since both controller types use the same SDLC link.

Controller Options

controller *aaaa*

The controller type (specified by *aaaa*) denotes the physical unit (PU) type of the SNA product being emulated. Certain aspects of the controller's operation and the types of devices that can be attached to the controller are determined by the controller type. Up to two controller types (PUs) can be defined in the source file.

For 3270 emulation, the controller type 3270 should be specified.

If a controller type is not specified, **snopts** will check the **lu** device type definitions (see the **lu** option below). If an **lu** device type 1 or 2 is defined, 3270 is assumed as the controller type. An error message is given if there are any errors in the **lu** definitions, such as an illegal device type.

When implementing dual-identity SDLC, each controller type defined requires its own **lu, xid, address,** and **seg** definitions. Two sets of controller options, defining two PUs—one 3270 workstation and one 3770 SNA/RJE workstation—can be defined in a source file. The PU definitions for 3770 must occur prior to the 3270 PU definitons. Refer to your *AT&T SNA/RJE Emulator+ System Administrator's Guide* for information on defining 3770 controller options.

lu*xx t [termself]*   A maximum of 32 logical units (LUs) can be assigned to a 3270 controller type.

xx denotes the LU number being assigned. Valid values for this parameter are 0 - 31.

t denotes the device type assigned to this LU. For controller type 3270, only two device types are valid:

   1 - Specifies the device is a printer. (SCS or DSC)

   2 - Specifies the device is a terminal.

termself is an optional LU parameter. If *termself* is specified, when the LU session ends the terminal is powered off, or an **xlu2clos** for this LU is issued from an API program (see the *AT&T 3270 Emulator+ HLLAPI Programmer's Guide*), and a TERM_SELF is sent to the host. Another session cannot be opened on this LU until an UNBIND has been received from the host for this LU. When *termself* is specified, the LU is considered to be a "high security" LU.

When *termself* is not specified for the LU, when the session ends, an LUSTAT is sent to the host. The LU

is considered to be a "low security" LU.

xid *xxxxx*   The SDLC XID (eXchange station IDentification) is used mainly in dial-up configurations to identify the calling station to the host. If the controller receives the XID command, it must identify itself to the primary SDLC station using a 6-byte (12 hex digits) sequence number. The first 7 digits are coded automatically, and are dependent on the controller type specified.

For controller type 3270 the first 7 digits will automatically default to 0200017. The last 5 digits are specified by the user using the **xid** option. This 5-digit number is an identification number that must be obtained from the Network Administrator at the host site.

The default value is 00000.

address *xx*   The SDLC station address. Each SDLC secondary station is distinguished by a 1-byte (2 hex digits) station address. This is the first byte after the flag in an SDLC frame.

The default value is 01.

seg *n*   Segment size refers to the maximum (decimal) size of the Path Information Unit (TH + RH + RU) sent from or to the controller.

The default value is 265.

broadcast   In the SDLC protocol all stations will accept a frame with the broadcast address. In a multi-station configuration, or with dual-identity SDLC, the user can designate which station will reply to the broadcast message by specifying the broadcast option.

If the option is not specified, the station will not be a broadcast station.

Creating a Configuration Source File

The configuration source file is a standard text file that can be created in the **/usr/snaadm/cust/snaoptions** directory using a text editor such as **vi**. It has the following characteristics:

- Configuration options may be specified in any order, one per line.

- The option must be the first item on the line.

- Extra text following each option specification is ignored, allowing the user to insert comments after each option.

- If the first character on a line is the # sign, the entire line is treated as a comment and ignored.

- If an option is not specified in the source file, its default value will be used.

- Invalid options may be ignored. (Error messages are not always given.) For example, if an option is misspelled or is not the first item on the line, it is ignored. (You should check the completed configuration displayed by **snopts** to verify that each option has been properly defined.)

Figure 1-2 is an example of an SNA configuration source file, illustrating the characteristics noted above. (A sample configuration source file is provided in the **/usr/snaadm/cust/snaoptions** directory.)

```
#This is a sample configuration source file defining one 3270 physical
#unit. Note the free format of this file.


controller 3270          Specifies the controller type.


     xid     00018        Sets XID to 00018.
seg 265                  Maximum segment size for this type of
#   SNA Controller.

broadcast                Will respond to broadcast messages.

#The following LU's are configured:

lu1  2                   For controller type 3270, only device types
#                        1 (printer) and 2 (terminal) are valid.

lu03  2 termself         This is a "high-security" terminal type lu.

lu4  2
lu31  1

        fdx              This sets SDLC to full-duplex mode.


#The next three options are invalid, and therefore will be ignored by
#snopts.  Note that if a default is available for the option, it will
#be taken.

address ty               ty is not a valid hex number.

multipnt                 It must be spelled "multipoint" to be
#                        valid, therefore this option will default
#                        to point-to-point operation.

-nrzi                    The option must be the first item on the
#                        line.
```

Figure 1-2: Sample SNA Configuration Source File

Creating a Configuration Object File

To create an SNA configuration object file, invoke **snopts** from the
/usr/snaadm/runtime directory using the command format:

       snopts [−**w**] [*object_file*] [< *source_file*]

Figure 1-3 lists the arguments to **snopts**.

Option	Description	Default Environment Variable	Default Value
object_file	Names the output object file. Must be prefixed by "CSNA."	**CONFIG**	**config**
source_file	Names the configuration source file. Must be specified by the user.	—	interactive mode

Figure 1-3: Arguments to **snopts**

If the *object_file* is specified on the command line, the file name must
have the prefix **CSNA.**, for example, **CSNA.hostx**.

If *object_file* is not specified, the environment list will be searched for
the variable **CONFIG**. If **CONFIG** has been set, its value will be used as
the name of the output object file. If **CONFIG** has not been set, then
snopts will use the default name **config** for the object file.

If < *source_file* is not specified on the command line, **snopts** executes
interactively. (See the section "Using **snopts** Interactively.")

The **snopts** utility creates the configuration object file from the
configuration source file and displays a screen showing the configuration as
snopts processed it.
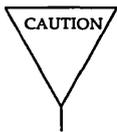
CAUTION: You should check that all the options specified in the source file match the displayed configuration.

If an option is not displayed or is incorrectly set, check your configuration source file for invalid options.

Example:

snopts −w < /usr/snaadm/cust/snaoptions/config1

snopts will use the configuration source file **config1** to create a configuration object file. The name of the object file will default to the value of the **CONFIG** environment variable, if it has been set. If **CONFIG** has not been set, the object file will be named **config** and placed in the **run-time** directory.

Using snopts Interactively

The **snopts** utility can also be used interactively to create a configuration object file. When you do this you do not have to create a configuration source file; **snopts** accepts input from your terminal as the source to create the object file. To use **snopts** interactively enter:

snopts −w *object_file*

The **snopts** utility will accept whatever you enter from your terminal as the configuration source options to be processed. You simply enter configuration options as required. When you have entered all the options you need, enter Ctrl-d to end the session. **snopts** will create the SNA configuration object file and display the completed configuration. You should check this display to make sure all the required options have been properly defined.

Displaying an Existing Object File

To display an existing configuration object file, execute **snopts** without the −w flag.

snopts [*object_file*]

As in the example above, if *object_file* is not specified on the command line, **snopts** will use the object file named by the **CONFIG** variable if that variable is set. If **CONFIG** is not set, **snopts** will use a file in the **runtime** directory with the default name **config**.

Figure 1-4 shows a sample **snopts** configuration screen.

```
SNA/SDLC CONFIGURATION

  full duplex    nrz    point-to-point

Controller 3270

  lu1  2
  lu3  2
  lu4  2
  lu5  1

  station address = 01
  xid = 020001700018
  segment size = 256
  responds to broadcast message
```

Figure 1-4: Sample **snopts** Configuration Screen

Running the SNA Controller

After you have configured the controller and the environment for your site and for the host, you can start running the AT&T SNA/3270 Emulator+.

The System Administrator may designate one terminal as the master terminal for the AT&T SNA/3270 Emulator+. The shell that runs on this terminal must have root privileges to start the SNA Controller processes. All status and error messages from the SNA Controller will be displayed on this terminal.

Starting the SNA Controller

The SNA Controller is started by the shell script **startsna** in **/usr/snaadm/runtime**. The **startsna** script checks to see that no other controller processes are running on the communications port to be used by this SNA Controller. (The communications port number is obtained from the **PCD** environment variable.) **startsna** then executes the commands to start the **SNA** process and the **sdlc** program.

At the start of execution, the **SNA** process reads the object file containing the configuration options for the controller. (The name of the object file is obtained from the **CONFIG** environment variable.) Configuration information is passed to the **sdlc** program, which is downloaded and started on the communications board. (Note that you will not see **sdlc** running on the UNIX operating system if you execute the **ps** command, because **sdlc** is running on a separate processor.) Then the **SNA** process enables and establishes communication paths to and from the **sdlc** program. As soon as **sdlc** and **SNA** are running, the SNA Controller is ready to communicate with the host. (Communication with the host is possible even if no terminal emulator is active.)

To start the SNA Controller, you must:

Step 1. Log in as root.

Step 2. Change directory to **/usr/snaadm/runtime**.

Step 3. Check that the environment is properly set. Be sure that **PCD** is set to the communications port the SNA Controller is using, and **P3274** is set to the named pipe for this controller. See the section Configuring the Environment for SNA if these variables must be changed. Setting variables properly is especially

important if more than one SNA Controller is running.

Step 4.   Execute the shell script **startsna**.

Step 5.   When the SNA Controller has been started successfully, a copy-
            right message and controller ready message are displayed. If you
            are running on a 3B2 Computer, the following message is also
            displayed:

            ISC NUMBER n IS USED FOR THE SNA CONTROLLER

            If you are on a 3B5 or 3B15 Computer, no such message is
            displayed.

┌──────┐   If connection to the host is via a dial-up line, you should
│ NOTE │   in at this point.
└──────┘
    │

Step 6.   The printer and terminal emulators can now be started.

Error Messages

Figure 1-5 contains a list of error messages displayed if the **SNA** process
fails during initialization. Failures may result from incorrect or incomplete
setup procedures. When these error messages are displayed, the **SNA** pro-
cess is aborted and must be restarted after you correct the problem. The
error messages are listed in the order that the **SNA** process generates them.
Some error messages are followed by a number. This number is the **errno**
value returned by the failed system call and may provide more specific
information about the cause of the error. For information about **errno**, con-
sult **intro**(2) in the *UNIX System V Programmer's Reference Manual*.

Error Message	Explanation/Possible Cause
sdlc open error	**sdlc** has not been successfully started or the device driver pathname for **sdlc** is incorrect.

name too long	Pipe prefix specification is too long (must be less than 100 characters).
path from devices open error	Inbound SNA-to-devices pipe does not exist.
configuration file not found	Named configuration object file does not exist.
configuration file open error	Permission bits incorrect for configuration file (should be readable).
configuration file read error	Permission bits incorrect for configuration file (should be readable).
error in configuration file	Configuration file is not the result of a successful execution of **snopts**.
too many lu's configured	More than 32 LU's are defined.

Figure 1-5: SNA Process Initiation Error Messages

SNA System Shutdown

You should stop the SNA Controller only after all device emulators have been powered down. Use the **stopsna** shell script to stop the **SNA** process and the **sdlc** program.

The following steps will stop the SNA Controller:

Step 1.   Log in as root.

Step 2.   Change directory to **/usr/snaadm/runtime**.

Step 3.   Check that the environment is properly set. Be sure that **PCD** is set to the communications port the SNA Controller is using, and **P3274** is set to the named pipe for this controller. See the section Configuring the Environment for SNA if these variables must be changed. Setting variables properly is especially

important if more than one SNA Controller is running.

Step 4.   Enter the command

stopsna

The SNA Controller that is running using the pipe specified by the **P3274** environment variable will be stopped.

2   BSC Operation

Introduction

Before running the BSC Controller, the System Administrator must configure the environment and create a controller configuration file according to local and host requirements.

Configuring the Environment

The AT&T 3270 Emulator+ uses environment variables as pre-defined defaults for command options. This means that when an option to a 3270 Emulator+ command is omitted, the list of environment variables is searched for a specific variable associated with that option. If that variable is set, its value is used as the option to the command. Therefore, if the appropriate environment variables are set, the user need not specify most options when issuing a command, unless the user chooses to override the defaults.

Configuring the environment involves setting the environment variables used by the 3270 Emulator+. For convenience, two shell scripts, **bscenvset** and **bscenvcust,** are provided to set these variables to default values.

Creating a Controller Configuration File

The controller configuration file contains the parameters the controller processes need to communicate properly with a particular host. When the BSC Controller is started, the configuration file is used as input to customize the controller processes with the options specified in the file.

Alternate configuration files can be created for different remote host and local configurations.

Configuring the Environment for BSC

The AT&T BSC/3270 Emulator+ processes refer to various environment variables during execution. These variables are used to provide information to the processes, such as default values for command options and specific system start-up information.

You must configure the environment for BSC/3270:

- Each time you log in, before you begin to use the BSC/3270 software

- Before starting or stopping a BSC Controller

Two shell scripts, **bscenvset** and **bscenvcust**, both in the /usr/bscadm/runtime directory, can be used to configure the environment for BSC operation. (Note that **bscenvset** calls **bscenvcust**.) Appendix B lists and describes the environment variables set by these scripts. It also provides the initial default values that the scripts assign to the variables.

Invoking bscenvset

Before configuring the environment, you should check that the default values set by the scripts are appropriate. If you must change any of these values, refer to the section Changing Environment Variable Defaults below.

If the default values set by the scripts are appropriate, you can configure the environment for BSC operation by executing **bscenvset** as follows:

Step 1.   Log in as root.

Step 2.   Change directory to **/usr/bscadm/runtime**.

Step 3.   Enter

   . ./bscenvset

Step 4.   Use the UNIX system commands **env(1)** and **set**(see **sh(1)**) to check your environment. (See the *UNIX System V User's Reference Manual* for information about these commands.)

The **bscenvset** script has to be executed only once per log in. To have **bscenvset** execute automatically when you log in, add the following entry to your .profile file: (**/usr/bscadm/.profile**):

   . ./usr/bscadm/bscenvset

Changing Environment Variable Defaults

If the local configuration makes it necessary for you to change the values of any of the environment variables, you should be aware of the following:

- **PCD** specifies the communications port that the BSC Controller is to use. You must change this variable if you want the controller to use a port other than that provided by the **bscenvset** script.

  - On a 3B2 Computer, the port denoted by **PCD** is the slot number that the ISC card is in. For example, if the ISC card you want to use is in slot 4, set **PCD=4**.

  - On a 3B5 or 3B15 Computer, **PCD** denotes the IOA device to be used. For example, to use the third SDLI port on the second IOA, set **PCD=203**.

  - On a 3B4000 Computer, **PCD** is a combination of the **pe** and ISC slot number (e.g., 120.6 for **pe** 120, slot 6).

    The default value for **PCD** as set by **bscenvset** is stored in a file called **PCD.file** in the **bscadm** directory.

- **BSCHOST** is used to change the **BS3274** variable, which specifies the full pathname of the controller configuration object file to be used when the BSC Controller is started. Configuration object files are named with the prefix **BS3274.**, for example, **BS3274.hostx**. **BSCHOST** indicates which **BS3274.***file* is to be used. For example, if **BS3274.hostx** (located in **/usr/bscadm/runtime**) is to be used, **BSCHOST** should be set equal to **hostx**. Then, when the **bscenvcust** script is executed, **BS3274** will be set to **/usr/bscadm/runtime/BS3274.hostx**.

- **P3274** specifies the named pipe the terminal and printer devices use to communicate with the controller. This pipe must have a unique name for each controller running on the system.

Temporary Changes

To change the value of an environment variable temporarily:

Step 1.   Execute **bscenvset**, as described in the section Invoking
bscenvset. (This has to be done only once per log in.)

Step 2.   Reset the environment variable(s) that must be changed. For
example, to specify that the controller should use the
configuration object file named **BS3274.hostb**, you would enter:

> **BSCHOST=hostb**

Step 3.   Execute **bscenvcust** by entering:

> **. ./bscenvcust**

The **bscenvcust** shell script will re-customize the environment
using the new values.

Permanent Changes

To change the value of an environment variable permanently:

Step 1.   Use a text editor such as **vi** to edit the **bscenvset** shell script,
changing the setting of the environment variables as necessary.

Step 2.   Execute **bscenvset** as described in the section Invoking
bscenvset.

Creating a BSC Controller Configuration File

You must create a configuration file for each different host and local configuration a BSC Controller may use.

It takes three steps to create a BSC Controller configuration file. You must:

Step 1.   Determine the configuration options needed to customize the BSC Controller for your particular site and host.

Step 2.   In the **/usr/bscadm/cust/bscoptions** directory, create an BSC configuration source file in which each option is defined as required.

Step 3.   Create an BSC configuration object file. This involves processing the configuration source file through the **bsopts** utility. When the **bsopts** utility is executed from the **/usr/bscadm/runtime** directory, it creates an object file from your source file and displays the completed configuration, including values for any defaulted options.

You can also invoke **bsopts** interactively, in which case you do not use a source file as input. Instead, you enter the configuration options directly into the **bsopts** program. However, using a source file allows you to retain the exact configuration for later use.

BSC Configuration Options

To determine the options required for the BSC Controller configuration file, consult the Network Administrator at your host site for precise information regarding your configuration. The options you define in the configuration source file should coincide with your host configuration.

Figure 2-1 lists the valid BSC configuration source file options and their corresponding defaults.

Option	Function	Default
Line Options:		
xx yy	Poll and select addresses of the controller where: xx = poll address (hex) yy = select address (hex). No keyword is specified for this option. This must be the first option specified.	None: poll and select addresses must be explicitly defined.
syns *n*	The number of leading SYN (synchronization) characters preceding each transmission.	2
hdx	Indicates half-duplex (2-wire) mode.	**fdx** (4-wire)
port *n*	Selects communications port number *n*.	1
max_blk *n*	Maximum inbound block size. The valid values for *n* are 0, 1, 2, and 3, corresponding to block sizes of 256, 512, 768, and 1024 bytes, respectively.	256
no_status	Suppresses the reporting of status error messages.	**status**
Error Retry Counts:		
bcc *n*	*n* is the maximum number of consecutive retries permitted following BCC errors.	12

Option	Function	Default
naks *n*	*n* is the maximum consecutive NAK errors.	12
acks *n*	*n* is the maximum consecutive wrong ACK.	12
enqs *n*	*n* is the maximum consecutive ENQ.	12
buffers *n*	*n* is the maximum number of consecutive buffer overruns allowed before an error occurs.	3

Time-Out Values:

idle *s*	*s* is the time-out in seconds waiting for synchronization while the line is idle.	20 seconds
cntr *s*	*s* is the time-out in seconds waiting for a control sequence.	3 seconds
data s	*s* is the time-out in seconds waiting for a data block.	45 seconds
sync *s*	*s* is the time-out in seconds waiting for synchronization within a data block.	2 seconds

Option	Function	Default
ASCII/EBCDIC Line **Control Options:**		
ascii	ASCII character set used on the communication line.	**ebcdic**
even	Sets even parity for ASCII.	Since the default is
odd	Sets odd parity for ASCII.	EBCDIC, these ASCII
space	Sets 0 parity for ASCII.	options do not default.

Figure 2-1: BSC Configuration Source File Options

Source File Options

The following is a functional description of the options listed in Figure 2-1.

Line Options

xx yy Each BSC 3274 controller has two distinct addresses, a poll address and a select address. The host uses the poll address to transmit a general poll or specific poll sequence; the controller uses it in the header of the first (or only) block of any device-specific inbound data message. The host uses the select address when transmitting a device selection sequence. The poll and select addresses must be the first option specified in the controller configuration. (There is no keyword for this option.)

syns *n* The number of leading synchronization characters preceding each transmission.

fdx/hdx	Full-duplex (four-wire) operation allows two stations to transmit and receive at the same time. In half-duplex (two-wire) operation, the stations take turns transmitting, using pin-level (RTS, CTS) controls. (Dial-up lines are half-duplex.) BiSync is strictly a half-duplex protocol, so full-duplex traffic cannot occur even with four wires. However, when full-duplex is specified in a point-to-point configuration, even though true four-wire traffic is not possible modem line turnaround delays are eliminated because RTS is always kept high. The default is **fdx** (full-duplex).
port n	This option specifies the communications board port number to be used by the controller. On a 3B2, port 1 must always be specified. On a 3B5 or 3B15, the exact port number to be used on the SDLI board should be specified.
max_blk n	This option specifies the maximum size of inbound blocks transmitted to the host. Valid values for n are 0, 1, 2, and 3, corresponding to block sizes of 256, 512, 768, or 1024 bytes, respectively. True IBM-configured hosts expect blocks of 256 bytes.
no_status/status	If desired, the BSC Controller can be configured to suppress its status messages by specifying the no_status option. True IBM-configured hosts expect status messages.

Error Retry Counts

A number of different conditions may occur that cause the BSC Controller to retry an operation in an attempt to recover. The **bcc, acks, naks, enqs,** and **buffers** parameters specify the maximum number of retries allowed for each operation. If the number of consecutive errors exceeds the specified maximum, the controller will transmit EOT to terminate the operation.

Time-Out Values

The BSC Controller starts timers appropriate to each state of the BiSync protocol to determine whether the maximum time has elapsed without a response from the host system. The time allowed for the host to respond depends on the state of the protocol and the device. The **idle, cntl, data,** and **sync** parameters allow the user to specify the number of seconds that must elapse before a time-out is declared.

ASCII/EBCDIC Line Control Options

The **ascii** option is used to specify ASCII line control. (The default is **ebcdic.**) A parity option (**even, odd,** or **space**) can be specified for **ascii** line control.

Creating a Configuration Source File

The configuration source file is a standard text file that can be created in the **/usr/bscadm/cust/bscoptions** directory using a text editor such as **vi.** It has the following characteristics:

- Configuration options may be specified in any order, one per line.

- The option must be the first item on the line.

- Extra text following each option specification is ignored, allowing the user to insert comments after each option.

- If the first character on a line is the # sign, the entire line is treated as a comment and ignored.

- If an option is not specified in the source file, its default value will be used.

- Invalid options may be ignored. (Error messages are not always given.) For example, if an option is misspelled or is not the first item on the line, it is ignored. (You should check the completed configuration displayed by **bsopts** to verify that each option has been properly defined.)

Figure 2-2 is an example of a BSC configuration source file, illustrating the characteristics noted above. A sample configuration source file is provided in the **/usr/bscadm/cust/bscoptions** directory.

```
#This is a sample source file for BSC customization.
#Comments may be inserted using the # as the first
#character on the line. Valid options
#must be the first non-white characters on their line.

40  60    define controller host identity
#         poll address = 40 hex
#         select address = 60 hex

port 1    select communications port 1

hdx       half-duplex (2-wire)

data 20 wait 20 sec. max for data block

bcc 10    retry max 10 times on bcc errs.

#The following specifications are invalid and are there-
#for ignored:

acks p    ("p" is not a number)
bufrs 17  (it must be spelled "buffers")
- hdx     (spec must be the first item on a line)

#Options not specified in the file, such as the number of
#leading SYNs, take default values.
```

Figure 2-2: Sample BSC Configuration Source File

Creating a Configuration Object File

To create a BSC configuration object file, invoke **bsopts** from the /usr/bscadm/runtime directory using the command format:

bsopts [−w] [*object_file*] [< *source_file*]

Figure 2-3 lists the arguments to **bsopts**.

Option	Description	Default Environment Variable	Default Value
-w	Creates an object file		
object_file	Names the output object file. Must be prefixed by "BS3274."	BS3274	BS3274.teke
source_file	Names the configuration source file. Must be specified by the user.	—	interactive mode

Figure 2-3: Arguments to **bsopts**

If the *object_file* is specified on the command line, the file name must have the prefix **BS3274.**, for example, **BS3274.hostx.**

If *object_file* is not specified, the environment list will be searched for the variable **BS3274.** If **BS3274** has been set, its value will be used as the name of the output object file. If **BS3274** has not been set, then **bsopts** will use the default name **BS3274.teke** for the object file.

If < *source_file* is not specified on the command line, **bsopts** executes interactively. (See the section Using **bsopts** Interactively below.)

The **bsopts** utility creates the configuration object file from the configuration source file and displays a screen showing the configuration as **bsopts** processed it.
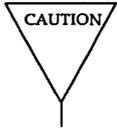
⚠ CAUTION You should check that all the options specified in the source file match the displayed configuration.

If an option is not displayed or is incorrectly set, check your configuration source file for invalid options.

Example:

bsopts −w < /usr/bscadm/cust/bscoptions/config1

bsopts will use the configuration source file **config1** to create a configuration object file. The name of the object file will default to the value of the **BS3274** environment variable, if it has been set. If **BS3274** has not been set, the object file will be named **BS3274.teke** and placed in the **runtime** directory.

Using bsopts Interactively

The **bsopts** utility can also be used interactively to create a configuration object file. When you do this, you do not have to create a configuration source file; **bsopts** accepts input from your terminal as the source to create the object file. To use **bsopts** interactively enter:

bsopts −w *object_file*

The **bsopts** utility will accept whatever you enter from your terminal as the configuration source options to be processed. You simply enter the configuration options as required. When you have entered all the options you need, enter Ctrl-d to end the session. **bsopts** will create the BSC configuration object file and display the completed configuration. You should check this display to make sure all the required options have been properly defined.

Displaying an Existing Object File

To display an existing configuration object file, execute **bsopts** without the −w flag.

bsopts [*object_file*]

As in the example above, if *object_file* is not specified on the command line **bsopts** will use the object file named by the **BS3274** variable if that variable is set. If **BS3274** is not set, **bsopts** will use a file in the **runtime** directory with the default name **BS3274.teke**.

Figure 2-4 shows a sample **bsopts** configuration screen.

```
cBSC/3270 CONTROLLER CONFIGURATION
1 virtual controller(s)
poll address   <40>    selection address    <60>

ebcdic character set
port number = 1
full duplex
buffer overrun linit = 3
nak limit = 12
ack limit = 12
enquiry limit = 12
bcc limit = 12
idle timer = 20
control message timer = 3
data message timer = 45
sync interval timer = 2
number of leading SYNs = 2
status suppression is set OFF
inbound block size = 256
call control string = ""
```

Figure 2-4: Sample **bsopts** Configuration Screen

Running the BSC Controller

After you have configured the controller and the environment for your site and for the host, you can start running the AT&T BSC/3270 Emulator+.

The System Administrator may designate one terminal as the master terminal for the AT&T BSC/3270 Emulator+. The shell that runs on this terminal must have root privileges to start the BSC Controller processes. All status and error messages from the BSC Controller will be displayed on this terminal.

Starting the BSC Controller

The BSC Controller is started by the shell script **startbsc** in **/usr/bscadm/runtime**. The **startbsc** script checks to see that no other controller processes are running on the communications port to be used by this BSC Controller. (The communications port number is obtained from the **PCD** environment variable.) **startbsc** then executes the commands to start the **TM3274** process and the **b3274** program.

At the start of execution, the **TM3274** process reads the object file containing the configuration options for the controller. (The name of the object file is obtained from the **BS3274** environment variable.) Configuration information is passed to the **b3274** program, which is downloaded and started on the communications board. (Note that you will not see **b3274** running on the UNIX operating system if you execute the **ps** command, because **b3274** is running on a separate processor.) Then the **TM3274** process enables and establishes communication paths to and from **b3274** and the devices. As soon as **b3274** and **TM3274** are running, the BSC Controller is ready to communicate with the host. (Communication with the host is possible even if no terminal emulator is active.)

To start the BSC Controller you must:

Step 1.   Log in as root.

Step 2.   Change directory to **/usr/bscsadm/runtime**.

Step 3.   Check that the environment is properly set. Be sure that **PCD** is set to the communications port the BSC Controller is using, and **P3274** is set to the named pipe for this controller. See the section Configuring the Environment for BSC if these variables must be changed. Setting variables properly is especially important if more than one BSC Controller is running.

Step 4.   Execute the shell script **startbsc**.

Step 5.   When the BSC Controller has been started successfully, a copyright/release number message and controller ready message are displayed. If you are on a 3B2 Computer, the following message is also displayed:

ISC NUMBER n IS USED FOR THE BSC CONTROLLER

If you are on a 3B5 or 3B15 Computer, no such message is displayed.

NOTE	If connection to the host is via a dial-up line, you should dial in at this point.

Step 6.   The printer and terminal emulators can now be started.

Error Messages

Figure 2-5 contains a list of error messages displayed if the **TM3274** process fails during initialization. Failures may result from incorrect or incomplete setup procedures. When these error messages are displayed, the **TM3274** process is aborted and must be restarted after you correct the problem. The error messages are listed in the order that the **TM3274** process generates them. some error messages are followed by a number. This number is the **errno** value returned by the failed system call, and may provide more specific information about the cause of the error. For information about **errno**, consult **intro(2)** in the *UNIX System V Programmer's Reference Manual*.

Error Message	Explanation/Possible Cause
bsc open error	AT&T BSC/3270 controller has not been started or device driver pathname for controller is incorrect.
name too long	Pipe prefix specification is too long (must be less than 100 characters).
path from devices	Inbound TM3274-to-devices pipe does not exist. open error
initialization file	Named configuration object file does not exist. not found
initialization file	Permission bits incorrect for configuration file open error (should be readable).
initialization file	Permission bits incorrect for configuration file. read error (should be readable).
error in initialization file	Initialization file is not the result of a successful execution of **bsopts**.

Figure 2-5: BSC Process Initiation Error Messages

BSC System Shutdown

You should stop the BSC Controller only after all device emulators have been powered down. Use the **stopbsc** shell script to stop the **TM3274** process and the **b3274** program.

The following steps will stop the BSC Controller:

Step 1. Log in as root.

Step 2. Change directory to **/usr/bscadm/runtime**.

Step 3. Check that the environment is properly set. Be sure that **PCD** is set to the communications port the BSC Controller is using. and **P3274** is set to the named pipe for this controller. See the section Configuring the Environment for BSC if these variables must be changed. Setting variables properly is especially important if more than one BSC Controller is running.

Step 4. Enter the command

      stopbsc

The BSC Controller that is using the pipe specified by the **P3274** environment variable will be stopped.

3 Running the 3287 Printer Emulator

Introduction

The 3287 Printer Emulator process, **PE3287,** does not require customization. You may use the regular UNIX operating system utilities such as the line printer spooler to print files. 3B printers are generally accessed through spoolers and filter programs. The filter programs are written for a specific printer (or class of printers) and perform all necessary configurations.

The procedures in this section are based on a printer with the following characteristics:

- a Carriage Return to change the current print position to column 1 of the current line

- a Line Feed (LF) to change the current print position to the current column of the next line

- backspace capability

- formfeed capability

Backspace and RETURN are often used by a filter program to move the cursor to characters to be underscored.

Starting the Printer Emulator

To invoke the Printer Emulator, use the command format:

> **pe3287** [−t *termno*] [−f *printer_path*] [−x *process_name*]
> [−u] [−p *controller_path*] [−F] [−M *msg_file*]

You should run this command as a background task, i.e., use an **&** at the end of the command line so your terminal remains available for use. For example:

> **pe8387 -t 7 &**

The following list describes the options in detail:

−t *termno*
  termno specifies acceptable printer device numbers. The controller selects a device number for this printer from those specified. The valid range of device numbers is 0-31. You may specify a range of device numbers such as 10-15, or a mix of specific numbers and ranges such as 1,5,7,10-15,18,20-24. The device numbers must be defined at the host as printers. If this option is omitted, the environment variable **D3274** is sought. If **D3274** is not defined, a default range of 0-31 is used.

−f *printer_path*
  printer_path specifies a file or named pipe for print output. If this option is omitted, the environment variable **PF3274** is sought. If the printer output path does not exist when the printer process is started, the path created is a file. Use the −f option only if the print path is a file; otherwise, use the −x option.

−x *process_name*
  process_name is a process for print output. If this option is chosen, print output is routed to another process. This option should not be selected if the −f option (routing print output to a file or named pipe) has been selected.

−u

Indicates that the data sent to the printer will not be formatted for printing. This option is useful if the physical printer is not a standard ASCII printer and special formatting is required.

−p *controller_path*

controller_path is a path (named pipe) to the controller process. If this option is omitted, the environment variable **P3274** is used to obtain the path to the protocol emulation process.

−F

According to BSC protocol, upon receipt of a start print command from the host system to an enabled printer, the controller will always respond with a WACK as a positive acknowledgement. (WACK implies a temporary device busy condition; the host must wait for an ACK from the controller before it can transmit any more data to that printer. In response to a WACK, the host normally sends an ENQ (enquiry) and will continue to send ENQs until either the controller responds with an ACK or until the host sends its maximum number of ENQ's. When the −F option is specified, instead of sending a WACK as a positive acknowledgement, the controller sends an ACK. This decreases response times by eliminating the WACK-ENQ-ACK sequence that normally takes place. (Note that this option bypasses normal BSC protocol procedures.)

−M *msg_file*

msg_file is the file containing the error messages that will be used by the printer process. The default is **pe3287.msg**.

Stopping the Printer Emulator

To stop the Printer Emulator, use the *kill* command. See kill(1), ps(1) and sh(1) in the *UNIX System V User Reference Manual* and kill(2), signal(2) in the *UNIX System V Programmer Reference Manual*.

Error Messages

The following are some of the error messages that may appear when starting the Printer Emulator:

Error Message	Explanation
can't specify both the −x and −f options	Specify either −x or −f, but not both.
incorrect terminal number	An invalid device number was used to specify a printer number. Check the **D3274** environment variable.
start > end of termno range	The device number specified at the beginning of the range was greater than the number specified at the end of the range. For example, 10-7 is an invalid range specification. Check the **D3274** environment variable.
open controller pipe fails	The controller pipe cannot be opened. Check to see that the specified pipe name is current. Check the **P3274** environment variable.
INSUFFICIENT RESOURCES	The controller cannot handle another terminal or printer process.
EMULATOR TERMINATED	A serious error has occurred or the printer process has stopped.

Stopping the printer process generates a power off notification to the controller.

4 Setting Up the User's Environment

Introduction

Each time a user starts a 3278/9 Terminal Emulator process, there are certain options for which **te3279** requires values. (See Appendix H for a complete list of the options to the **te3279** command.) If these options are not specified on the command line, **te3279** looks for a specific environment variable associated with each missing option. If the environment variable is set, **te3279** uses its value as the option default. Therefore, if the user's environment is set up properly, the user does not have to specify any options when starting the emulator. (The −e option is an exception.)

To ensure that this will be the case, the System Administrator should configure the user's environment with appropriate values for each of the environment variable defaults. This must be done before the user invokes **te3279**. One way to do it is to set and export the required environment variables in each user's login **.profile**. Another is to add **snaenvset** (or **bscenvset**) to each user's **.profile**. In the latter case you must specify the full pathname of the script and precede the script name with a period and space, for example, . ./**usr/snaadm/runtime/snaenvset** (see the *Additional Considerations* section, later in this chapter).

te3279 uses the following environment variables:

D3274
    Corresponds to the −t *termno* option. This provides **te3279** with the device number(s) it is allowed to use. It allows the System Administrator to control the use of the available devices by only allowing certain users access to particular devices.

    Example: **D3274=0,1,3,5,10-14**

    This would allow the user to access only the devices numbered 0, 1, 3, 5, and 10 through 14. If one of these devices is already connected to a host, the controller will select the next available device from the list.

SC3279
    Corresponds to the −s *screen_file* option. **te3279** uses the screen control file named by this option or variable to emulate the 3278/9 display capabilities on the user's terminal.

    Example: **SC3279=/usr/snaadm/runtime/SC.5425**

KY3279   Corresponds to the −k *key_file* option. **te3279** uses the
                  keyboard mapping file named by this option or variable
                  to emulate the 3278/9 key functions on the user's termi-
                  nal.

                  Example: **KY3279=/usr/snaadm/runtime/KY.5425**

P3274   Corresponds to the −p *pipe* option. This provides **te3279**
                  with the exact name of the pipe for the controller it will
                  be using. (**te3279** and the controller will communicate
                  via this pipe.)

                  Example: **P3274=/usr/snaadm/runtime/P3274.1**

PF3274   Corresponds to the −f *printer_path* option. Specifies the
                  printer pathname to be used for local print functions.

                  Example: **PF3274=/usr/snaadm/runtime/filename**

TM3279   Corresponds to the −m *model* option. If a model 5 (132
                  column mode) is to be invoked, **TM3279** should be set
                  equal to 5.

                  Example: **TM3279=5**

PATH   Each user must add the **runtime** directory to their
                  PATH.

                  Example: **PATH=/usr/snaadm/runtime:$PATH**

The System Administrator should provide each user with a list containing
the appropriate setting of these environment variables for that user.

Additional Considerations

The following environment variables are used by the **snaenvset** and **bscenvset** scripts.

TERM   The scripts use the value of the user's **TERM** environment variable to set the **SC3279** variable. Thus if the user enters a value for **TERM** that does not correspond to the name of the screen file in the **run-time** directory for that terminal type, **te3279** will not execute. For example, if a user logs in and sets **TERM=hp2621**, the script will set **SC3279=SC.hp2621**. **te3279** will not find the right screen file because the file for the hp2621 that is provided with the software is named **SC.2621**. There are three possible solutions to this problem:

1. Have the user log in using the names provided with the software.

2. Link the existing keyboard and screen files to files named after the user's **TERM** setting. For example, in the above case, you could link **SC.2621** to **SC.hp2621** using the UNIX system **ln** command.

3. Have the user use the **−k** and **−s** options to specify the full pathname of each file when starting **te3279**.

KYTYPE   The scripts use the environment variable **KYTYPE** to set the **KY3279** variable. **KYTYPE** should be originally set to correspond to **TERM**. When **snaenvcust** or **bscenvcust** is run, KY3279 is set to **/usr/snaadm/runtime/KY.<$TERM>** or **/usr/sanadm/runtime/KY.std** if **KY.<$TERM>** does not exist.

   For example, to change **KY3279** to use a 5425-specific file, set **KYTYPE=5425**; then run **bscenvcust** or **snaenvcust**, as appropriate.

Programming Considerations

Environment Variables

The AT&T 3270 Emulator+ HLLAPI uses the environment variables that are set and exported by the **snaenvset** or **bscenvset** shell scripts. In addition, the AT&T HLLAPI uses the following UNIX System environment variables and files:

KY3278: name of the keyboard customization file (object file)

LUPORT: contains the logical channel to be used for a particular session. If this environment variable is NULL, the LUPORT uses the next available logical channel.

LUTABLE: file name for LUTABLE

P3274: named pipe to the controller process (e.g., SNA or tm3274)

RTMSG: pathname for the location of the run-time message file

SC3278: name of the screen customization file, described in the *Additional Considerations* section in this chapter.

STSIZE: maximum storage size that you can allocate, using the **Storage Manager** function. A practical maximum value is 64K, or the amount left over in the address space by the **hllapi** process, including your application program (your application program is linked with the **hllapi** library)

TE3279.MSG: pathname of the file to which 3279 terminal message file, described in the *Additional Considerations* section in this chapter.

TM3279: terminal model, either 2 or 5, described in the *Additional Considerations* section in this chapter.
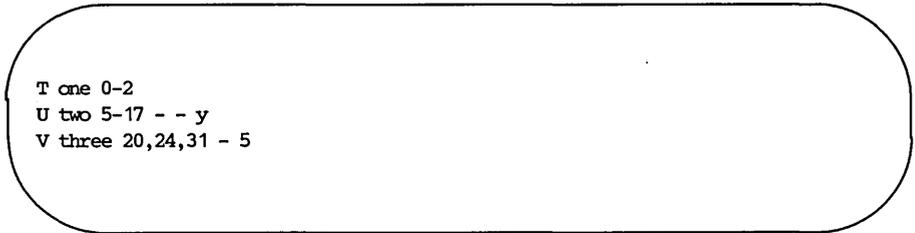
LUTABLE File

The **LUTABLE** file describes the logical units (LUs) that will be assigned to the controller type. The **hllapi** process will first look in the user's home directory to see if there is an **LUTABLE** file. If not, the file defined by the **LUTABLE** environment variable (above) will be used. The user or the Systems Administrator can restrict access to the **LUTABLE** file in the user's home directory (see chmod(1) in the *UNIX System V User Reference Manual* and chmod(2) in the *UNIX System V Programmer Reference Manual*). This feature provides additional security, since it in turn restricts **hllapi** use.

A HLLAPI application program can interact with up to 4 presentation spaces, although at any given time, it can only be connected to one; therefore, the **LUTABLE** contains a maximum of 4 entries, with a maximum of 6 fields in each entry. Each entry in the **LUTABLE** file contains the following fields, separated by blanks:

- The presentation space short name. You can use any capital letter from A through Z in the presentation space short name, but no numbers.

- The presentation space long name. You can use any combination of 8 characters, excluding white space characters (space and tab).

- LU number(s) or number range: the AT&T 3270 Emulator+ allows up to 32 LUs to be connected at any given time, numbered from 0-31. LU numbers can be separated by commas (,) (e.g., 1,3,5), or you can use a dash (-) to indicate a range (e.g., 2-6).

- The name of the controller pipe.

- The terminal model number, either 2 or 5.

- An extended attribute bytes indicator, either a 'y' or a blank.

Figure 2-1 shows a sample LUTABLE. This LUTABLE contains three (3) entries, which specifies 3 presentation spaces associated with the user application program. The first entry shows the presentation space short name "T," the presentation space long name "one," and the LU numbers 0 through 2. The name of the controller pipe, and the terminal model are not specified; therefore, the default values will be used. The default values are those that were assigned to the environment variables.

The second entry in the sample LUTABLE shown in Figure 2-1 shows the presentation space short name "U," the presentation space long name "two", and the LU numbers 5 through 15. The two fields that follow contain a dash (-); this means that the default values will be used for the name of the controller pipe and the terminal model. The last field contains a 'y', which specifies that extended attributes will be used. The last entry in Figure 2-1 is similar to the second entry, except that the terminal model is specified, and the extended attributes field is left blank.

```
T one 0-2
U two 5-17 - - y
V three 20,24,31 - 5
```

Figure 4-1: Sample LUTABLE

5   Terminal Customization

Overview

The AT&T 3270 Emulator+ software is designed to work with many different types of ASCII terminals. Because the screen and keyboard layouts of these terminals differ from those of an actual IBM 3278 or 3279 Display Station, the 3278/9 Terminal Emulator process (TE3279), which is invoked by **te3279**, must translate the logical 3278/9 functions to the target ASCII terminal's physical features. To accomplish this, there are two binary files (and two related source files) associated with each supported terminal type: a keyboard mapping file, and screen control file. These two binary files are jointly used by the **TE3279** process to define how the 3278/9 functions are to be emulated on the target terminal.

The AT&T 3270 Emulator+ provides customized source and binary (object) files for the following terminals:

- AT&T 4410, 4415, 4418, 4425, 605, 610, 615, 620, 630

- Teletype 5410, 5418, 5420, 5425

- Hewlett-Packard 2621

- Lear-Siegler ADM-3a

- Televideo 910, 924

- DEC VT-100

- Tektronix 4105A

Appendixes C and D contain sample keyboard and screen source files.

Because the 3278/9 terminal mapping information is contained in files, it can be customized to suit the user's specific needs. Additionally, customized files can be created for terminals other than those listed above.

The AT&T 3270 Emulator+ provides two stand-alone, off-line utility programs, **kyinit** and **scinit** (both located in the **runtime** directory), that generate customized keyboard and screen object files to be used by the 3278/9 Terminal Emulator.

kyinit   The **kyinit** utility program generates a keyboard object file using a keyboard mapping source file. The source file is a user-defined text file containing the physical key assignments (mappings) for the logical 3278/9 key capabilities to be emulated on the target terminal.

Optionally, the user can input the keyboard mapping directly to **kyinit** by invoking **kyinit**'s interactive feature and then typing in the physical key sequences at the **kyinit** prompts for each capability. Additionally, the **kyinit** utility provides its own default definitions for key capabilities not defined by the user. **kyinit** also has an optional report generation feature that can be used to list all the physical key assignments made in the keyboard file.

scinit

The **scinit** utility generates a screen control object file using a screen control source file created by the user. Unlike **kyinit**, **scinit** cannot be used interactively, nor does it have a report generation feature. However, **scinit** does have the intelligence to determine if 3278/9 display emulation is possible given the screen control definitions provided in the source file.

If the screen control capabilities have not been properly defined in the source file, **scinit** will issue an error message. The **scinit** utility does not provide its own program defaults, but it can use the source **terminfo** description of the target terminal to obtain definitions for certain capabilities not defined in the source file. (**terminfo** is the standard UNIX system database of terminal capabilities.)

Although the **kyinit** and **scinit** utilities actually create the customized object files to be used by the 3278/9 Terminal Emulator, most of the work in customizing a terminal involves defining the 3278/9 key and screen capabilities in the source files used by these utilities.

This section describes the procedures to create customized keyboard mapping and screen control files for the 3278/9 Terminal Emulator. Note that although the keyboard and screen files are jointly used by the terminal emulator, they are customized independently. This allows you to customize either the keyboard functions or the screen control functions, or both. However, for many 3278/9 capabilities to be implemented, they must be defined in both files.

Before you begin, you should be familiar with:

- the IBM 3279 Information Display Station

- the UNIX system terminal capability database, **terminfo**

For a complete description of **terminfo**, refer to the **terminfo**(4) entry in the *UNIX System V Programmer's Reference Manual*.

Keyboard Customization

Three steps are involved in customizing the keyboard for a target terminal:

Step 1.   Define (map) the keys or key sequences to be used to emulate the 3278/9 key capabilities on the target terminal.

Step 2.   Identify any default key definitions provided by the **kyinit** utility that are not appropriate for the target terminal.

Step 3.   Input the key definitions to the **kyinit** utility to produce the customized keyboard object file to be used by the 3278/9 Terminal Emulator process, **TE3279**.

When a Terminal Emulator is started on the target terminal, it will use the mapping defined in this object file to emulate the 3278/9 key functions on that terminal's keyboard.

The following describes steps 1 and 3 in detail.

Mapping the 3278/9 Key Capabilities

Before actually mapping the keyboard, you must select the 3278/9 keyboard features to be implemented. In general, all 3278/9 functions may be implemented unless there are physical limitations on the target terminal, or special applications to be considered. For example, not all terminals support two cursor types.

The target terminal's key functions, both standard and 3278/9-specific, are defined as a set of string value capabilities. (Note that **kyinit** does *not* use **terminfo** descriptions.) See Figure 5-1 for a complete list of these string value capabilities.

The key capabilities for the target terminal can be mapped in a source file, and/or entered directly to the **kyinit** program using its interactive feature:

- When entered interactively to **kyinit**, the capabilities are defined by entering the exact *keystrokes* to be used.

- When mapped in a source file, the capabilities are defined by specifying the *codes* generated by the keys, rather than the keys themselves. For example, \E represents the Escape key, and ^ represents the Ctrl key.

Creating a Keyboard Mapping Source File

The easiest way to create a keyboard mapping source file is to copy an existing source file from the **keyboard** directory (**/usr/snaadm/cust/keyboard** or **/usr/bscadm/cust/keyboard**) and make appropriate changes to the file using **vi** or any other available text editor.

Each definition must begin with a tab followed by a colon (:) and end with a colon and a backslash (:\). For example,

    :KY_ENTER=ˆm:\

A key capability may also be left undefined, for example

    :KY_RSQ=:\

A generic keyboard source file named **KY.std** is provided with the software package, as well as default keyboard source files for AT&T 3270 Emulator+ supported terminals. The key definitions in **KY.std** and the **kyinit** default definitions are the same. This keyboard mapping is a general-purpose mapping. However, it will probably not be the optimal mapping strategy for a particular target terminal. You can use it as a baseline, to be adjusted as necessary, particularly with respect to the cursor movement keys.

To define the key capabilities in a source file, you must identify the codes sent by the function and cursor movement keys on the target terminal. To find these codes see the user's guide for the target terminal.

Keyboard Mapping Considerations

As you define each 3278/9 key capability, try to create a close mnemonic association between the logical 3278/9 key and the physical key sequence you assign it. For example, defining the logical 3278 RESET key capability KY_RESET as ˆr (the Ctrl and r keys) is a simple, mnemonically appropriate choice. Also, it is best to assign the simplest keystroke sequences to those features that are likely to be used most often.

Note that most key sequences begin with either the Escape key (\E) or the Ctrl key (^).

Map key sequences of the target terminal to similar 3278/9 keys. Select the physical key sequences for the PF (Programmable Function) keys as logically as possible, taking care to consider the implementation of any of the target terminal's own function keys. For example, if PF1 is defined as \E1^m, PF2 should be defined as \E2^m and so on for the remaining function keys.

Do not assign the control characters ^S and ^Q to other functions: they often have a predefined significance to the communications medium between the target terminal and computer.

If the target terminal has ENTER and RETURN keys that generate different codes, assign the target terminal's ENTER key to either KY_ENTER or KY_ENTER1.

You may find that the KY_CENT, KY_SOLID, and KY_NOT capabilities need alteration. The defaults for these capabilities map the three ASCII characters that are non-EBCDIC to the three EBCDIC characters that are non-ASCII.

No physical key sequence may generate more than 20 characters, or be a substring of any other physical key sequence. For example, \Es and \Esi may not both be used, but Es and ES could be.

To get a "^" in a definition (actually hitting the carat key), enter "\^", since "^" by itself indicates a control sequence. Thus "\^s" would indicate a carat followed by "s", while "^s" would indicate "Ctrl s".

When you finish mapping the key functions, recheck your key capability definitions for possible duplication. (The **kyinit** program will warn you.) For example, if the cursor-down function code on a target terminal is "^j" (the Ctrl and "j" keys) and the new-line capability is also assigned "^j", you must reassign one of these functions.

The table in Figure 5-1 lists all the key capabilities that can be mapped for a target terminal. The **kyinit** program defaults (enclosed in quotes) are also included for each key capability.

Capability	Description	kyinit Default
KY_SPACE	space/blank	" "
KY_CENT	cent sign	"^"
KY_DOT	period	"."
KY_LT	less than sign	"<"
KY_LPAR	left parenthesis	"("
KY_PLUS	plus sign	"+"
KY_SOLID	solid vertical bar	undefined
KY_AMPERS	ampersand	"&"
KY_BANG	exclamation point	"!"
KY_DOLLAR	dollar sign	"$"
KY_STAR	asterisk	"*"
KY_RPAR	right parenthesis	")"
KY_SEMI	semi-colon	";"
KY_NOT	NOT sign	"]"
KY_DASH	minus sign	"-"
KY_SLASH	slash	"/"
KY_SPLIT	OR sign	"¦"
KY_COMMA	comma	","
KY_PRCNT	percent sign	"%"
KY_UNDER	underscore	"_"
KY_GT	greater than sign	">"
KY_QUEST	question mark	"?"
KY_BOPHE	backwards apostrophe	"\`"
KY_COLON	colon	":"
KY_OCTO	pound sign	"#"
KY_AT	at sign	"@"
KY_STROPHE	apostrophe	"'"
KY_EQUAL	equal sign	"="
KY_QUOTE	double quote	"""
KY_TILDE	tilde	"~"
KY_LCURL	left curly bracket	"{"
KY_RCURL	right curly bracket	"}"
KY_BSLASH	backslash	"\\"
KY_FM	3278/9 FIELD MARK	"^k"

Capability	Description	kyinit Default
KY_CARAT	carat/up-arrow	undefined
KY_LSQ	left square bracket	undefined
KY_RSQ	right square bracket	undefined
KY_DUP	3278/9DUP	"^d"
KY_UP_A	cursor-up	"^t"
KY_DOWN_A	cursor-down	"^v"
KY_RIGHT_A	cursor-right	"^g"
KY_LEFT_A	cursor-left	"^f"
KY_RDUB	fast cursor-right	"^y"
KY_LDUB	fast cursor-left	"^r"
KY_BS	backspace (may be same as cursor-left)	"^h"
KY_TAB	tab	"^i"
KY_BAKTAB	backwards tab	"^e"
KY_NEWL	3278/9/newline	"^j"
KY_HOME	3278/9 home	"^o"
KY_E_EOF	3278/9 ERASE EOF	"\Eef"
KY_E_INPUT	3278/9 ERASE INPUT	"\Eei"
KY_DEL	3278/9 delete	"\177" (octal)
KY_INS	3278/9 insert-mode	"^u"
KY_RESET	3278/9 RESET	"^a"
KY_ENTER	3278/9 ENTER	"^m"
KY_ENTER1	alternate ENTER key	"^m"
KY_CLEAR	3278/9 CLEAR	"\Ez"
KY_PF1 to KY_PF24	3278/9 Program Function Keys 1 to 24	"\E1^m" thru "\E24^m"
KY_PA1 to KY_PA3	3278/9 Program Access Keys 1 to 3	"\Ea1^m" thru "\Ea3^m"
KY_SYS_REQ	3278/9 SYS REQ	"\Eq"
KY_DEV_CNCL	3278/9 DEV CNCL	"\Ed"
KY_ATTN	3278/9 ATTN	"\Ea^m"
KY_CURSR_SEL	3278/9 CURSOR SEL	"\Ec^m"

Capability	Description	kyinit Default
KY_PRINT	3278/9 print	"\Ep"
KY_IDENT	3278/9 IDENT	"\Ei"
KY_CLICK	keyclick toggle	"\Ek"
KY_NUM_OV	numeric over-ride	"^n"
KY_REDRAW	locally redraw the screen	"\Er"
KY_SHELL	escape to shell	"\Es"
KY_EXIT	exit emulation	"\Exx"
KY_STAT	toggle the type of status area display	"\El"
KY_BLINK	toggle blinking cursor	"\Ebl"
KY_ALT_CR	toggle cursor shape	"\En"
KY_CAN	cancel partially sequence	"^x"
KY_STMV	move status line to current cursor line	"\Eh"
KY_ALLCAP	toggle for displaying all alphabetic characters in upper case	"\Em"
KY_NULLEND	erase trailing blanks in field	"^b"
KY_a to KY_z	lower-case alphabetic characters	"a" to "z"
KY_A to KY_Z	upper-case alphabetic characters	"A" to "Z"
KY_0 to KY_9	numeric characters	"0" to "9"
KY_COLR	monochrome/color toggle	"\Eclr" (color)
KY_BOT	display bottom of the screen buffer	"\Ebot" (model 5)

Capability	Description	kyinit Default
KY_TOP	display top of the screen buffer	"\Etop" (model 5)
KY_CTRL	display menu	"\Efd"
KY_PREVS	previous session	"\E-"
KY_NEXTS	next session	"\E+"

Figure 5-1: 3278/9 Key Capabilities

Using the kyinit Utility

The **kyinit** utility accepts logical 3278/9 key definitions from three sources. Definitions from a high priority source override those from a low priority source. In order of priority the sources are:

- The **kyinit** user during an interactive session.

- A keyboard mapping source file.

- The **kyinit** program defaults (see the "**kyinit** Defaults" column in Figure 5-1).

kyinit Output

Depending upon the command line parameters, the **kyinit** utility may generate the following three forms of output:

- A customized keyboard mapping object file for **TE3279**.

- A report file that lists the keyboard mapping created during the **kyinit** session (see the listing in Appendix F).

- A list of error messages that were generated during the session for invalid **kyinit** conditions.

Invoking kyinit

The **kyinit** utility creates a binary file that tells the **TE3279** process what mapping to use to emulate the 3278/9 keyboard on the target terminal. To invoke **kyinit,** use the command format:

kyinit [−0] [−nx] [−i] [−r *report_file*] [−k *source_file*] [−o *object_file*]

The following is a list of all the command options:

−0	Cancels default values for all options that have defaults.
−nx	Cancels the default value for the option name x that follows. More than one option name may be specified in string format. For example, to cancel the default −k and −o parameters, specify −**nko.**
−i	Indicates an interactive **kyinit** session, in which you may assign key sequences to logical 3278/9 key capabilities. Interactive assignments override source file definitions (if a source file is also specified) and **kyinit** defaults.
−r *report_file*	*report_file* is a file name in which the report file for the session's key assignments is compiled. If the named file does not already exist, it is created dynamically.
−k *source_file*	*source_file* is the name of the keyboard source file. If the source file name is absent, **kyinit** searches for the value of the environment variable **KYI3279.** If the variable is not defined in the environment, the file **KYI3279** is sought. The file definitions of logical 3278/9 functions override the program default definitions.
−o *object_file*	*object_file* is the binary file used by **TE3279** for keyboard emulation. If the named file does not exist, it is created dynamically. If this option is missing, **kyinit** searches for

the environment variable **KY3279**. If it is
not defined in the environment, **KY3279** is
used as the output file pathname.

Using kyinit Interactively

To use **kyinit** in an interactive session:

■ Enter the **kyinit** command, including the −i option, to start the
session.

■ Designate your NULL and CANCEL keys.

   □ The NULL key is used to end each definition.

      □ To change a default (current) value, hit the keys for the new
key sequence, and press NULL to terminate the entry.

      □ Type NULL by itself to accept the default (current) value.
Some keys may be defined as "undefined," such as KY_RSQ.
Pressing just NULL leaves the key undefined.

   □ If you make a mistake while entering a new key definition, press
CANCEL. You will be re-prompted for that key definition. Typ-
ing CANCEL alone on the key definition line allows you to go
back to the previous key definition. Two consecutive CANCELs
end the **kyinit** session.

■ At the **kyinit** prompts, choose to accept or not to accept the defaults
for the following sets of keys:

   □ upper and lower case characters,

   □ numeric characters, and

   □ punctuation keys.

If you choose to accept the defaults for all three sets of keys, you sim-
ply go on to the next step. If not, for each set that you did not
choose to take the defaults, you will be prompted to enter a
definition (or take the default) for each individual key.

■ Define the remaining key capabilities in response to the **kyinit** prompts.

The following takes you through the **kyinit** session step-by-step.

Step 1. To start a **kyinit** session, change directory to your **runtime** directory, and enter:

kyinit −i [−k *source_file*] [−o *object_file*] [−r *reportfile*]

where *source_file* is the full pathname of a keyboard mapping source file, and *object_file* is the name of the keyboard output file. (A keyboard source file is not needed, but can be useful for leaving keys undefined or changing **kyinit** defaults. If you use a source file that is close to what you need, then you will only have to change the few keys that are not appropriately defined for your needs.) The system returns the message:

```
welcome to interactive 3278/9 KEYBOARD MAPPING
INITIALIZATION
```

Step 2. At the system prompt

```
DO YOU WANT TO CONTINUE?
```

type **y** and press RETURN to continue with the interactive session or type **n** and press RETURN to exit the session.

Step 3. The system then prompts you to enter the keys you will be using as NULL and CANCEL.

Step 4. As the system prompts for each of the following standard character sets type **y** or NULL to accept the default values or press RETURN if you want to create your own definitions. If you accept the defaults, the message DEFAULTS TAKEN is displayed. If you don't accept the defaults no message is displayed.

```
TAKE DEFAULTS FOR UPPER CASE A THRU Z??

Take defaults for lower case a thru z??

TAKE DEFAULTS FOR NUMBERS 0 THRU 9??

TAKE DEFAULTS FOR THE FOLLOWING PUNCTUATION MARKS:
<SPACE>!@#$%^&*{}_-+='\¦[];:'''',.<>?/{}??
```

Step 5.   The system will then prompt you to specify definitions for any of
the above key capabilities for which you did not accept defaults
and all the remaining key capabilities. For each capability, type
NULL to accept the default (current) value, or type the value you
want to be mapped to that key, followed by NULL. Note that
NULL terminates the entry, not RETURN. When all the key
capabilities have been defined, the system displays any incon-
sistencies in or possible problems with your keyboard definitions
and a message giving the length of the file.

LENGTH OF OUTPUT FILE keyout IS 606 BYTES.

Step 6.   When you have ended the **kyinit** session, if you specified the −r
reportfile option, you may wish to print the **kyinit** report file to
verify the current mapping of key sequences.

The **kyinit** report file lists the assigned key sequences as they are
currently defined in the **kyinit** output file. To print the **kyinit**
report file, use any of the standard UNIX operating system print
commands, such as **lp** or **pr**.

Appendix F contains a sample **kyinit** report file.

Screen Customization

Two steps are involved in customizing the screen control functions for a target terminal:

Step 1.   Create a customized screen control source file defining the screen capabilities to be implemented on the target terminal.

Step 2.   Process the source file through the **scinit** utility to create a screen control object file.

Creating a Customized Screen Control Source File

The screen control source file defines the logical 3278/9 display control functions to be emulated on the target terminal. The source file consists of a small subset of standard **terminfo** capabilities and additional 3278/9-specific capabilities. For a complete list of the screen capabilities used to define the screen control functions for a target terminal, refer to the "Screen Capability" column in Figure 5-2.

To create the screen control source file, you should be thoroughly familiar with the screen control features on the target terminal. Check the user's guide for the target terminal to obtain this information.

An easy way to create a source file is to copy an existing source file in the screen directory (**/usr/snaadm/cust/screen** or **/usr/bscadm/cust/screen**) and make the appropriate changes to the file using **vi** or any other available text editor.

A generic screen control mapping is not provided because it is impossible to generalize features such as cursor motion and standout mode. (Since the terminal emulator responds to all keyboard input, it is possible to provide a generic keyboard mapping for all 3278/9 keys functions).

Screen definitions are categorized and formatted in the same way **terminfo** definitions are; they are Boolean, numeric, or string-type. Each definition must begin with a tab followed by a colon (:) and end with a colon and a backslash (:\), for example

:red=\E[1m:\

To build the source file, you must:

Step 1.   Identify the 3278/9 display features to be implemented. Figure 5-2 contains a complete list of the screen capabilities that may be defined to emulate 3278/9 display functions.

Step 2.   Identify any **scinit** program defaults (obtained from the **terminfo** entry for the terminal) which are not appropriate for the target ASCII terminal, for example **is** and **es**. See Figure 5-2 for a list of the **scinit** defaults.

Step 3.   Designate echo strings for the five non-defaulted symbols: CENT, NOT, SOLID, DUP, and FM.

Step 4.   Define the **cl, cm, co, li** standard **terminfo** capabilities for the screen clear character (clear).

Step 5.   Check all definitions to ensure that there are no undesirable conflicts with the default definitions.

terminfo Database

The **terminfo** database consists of a series of capability definitions for each specified terminal. Three types of terminal capabilities may be specified:

■ **Boolean** capabilities are simply on/off switches. If a boolean capability is specified for a terminal, the terminal possesses that capability. If a boolean capability is not specified, the terminal lacks that capability. For example, "ns" signifies a terminal whose screen does not scroll.

■ **Numeric** capabilities support specification of dimensions, such as the number of lines on a screen or page, or the number of blank characters left on the terminal display before and after a standout field (xmc). They are defined with a pound sign (#) followed by the value.

■ **String-type** capabilities are used to assign terminal-specific character sequences to standard terminal control features. String-type capabilities are specified with an equal sign (=) followed by a special code. For more information on string-type capabilities, refer to the **terminfo(4)** manual page in the *UNIX System V Programmer's Reference Manual*.

terminfo descriptions are processed (compiled) into object files by the **terminfo** Compiler (see **tic(1M)**).

scinit uses the **terminfo** source files to obtain definitions for certain screen control capabilities of the target terminal. You do not need to change the **terminfo** description that is provided with your 3B Computer for use with **scinit**. Think of the **terminfo** description as simply another source of input to **scinit**, which may be overriden by values specified in the **scinit** source file. For a complete description of **terminfo**, refer to the *UNIX*

System V Programmer's Reference Manual. A sample **terminfo** entry for the Teletype 4410 terminal is shown below:

```
att4410|4410|tty5410|5410|AT&T 4410/5410 terminal in 80 column mode,
    am, mir, msgr, xon,
    cols#80, it#8, lines#24,
    bel= ^G, blink = \E[5m, bold = \[2;7m, clear = \E[H\E[J,
    cr = \r, cub1 = \B, cud1 = \n,
    cuf1 = \E[C, cup = \E[%i%p1%d;%p2%dH, cuu1 = \E[A,
    dch1 = eE[P, dim = \E[2m, dl1 = \E[M, ed = \E[J, el = \E[K,
    home = \E[H, ht = \t, ich1 = \E[@, il1 = \E[L,
    ind = \n, invis = \E[8m,
    kbs = \b, kcir = \E[2J, kcub1 = \E[D, kcud1 = \E[B,
    kcuf1 = \E[C, kcuu1 = \E[A,
    khome = \E[H, kl1 = \E[24;1H, ll = \E[24H, rev = \E[7m, ri = \EM,
    rmacs = ^O, rmso = \E[m, rmul = \E[m,
    sgr = \E[0%?%p1%p5%|%t;2%;%?%p2%6%|%t;4%;%?%p4%t;5%;
%?%p3%p1%|%p6%|%t;7%$"" tilde$;%?%p7%t;8%;m%?%p9%t^N%e^O%;,
    sgr0= \E[m^O, smacs = ^N, smso = \E[7m, smul = \E[4m,
```

Screen Control Considerations

Implementing the Status Line

The 3278/9 Operator Information Area is emulated as a status line. You may designate the twenty-fifth line of a display screen (if available) as a permanent status line or set the status line on a line of your choice.

If the target terminal has a special status line that is not accessible with normal cursor movement sequences, your must define the **tsl**, **fsl**, **esl**, **dsl**, and **slF** characteristics.

- The **tsl** and **fsl** capabilities define the start and end sequences, respectively, for status line data. The **tsl** capability also clears the status line of its previous contents. The **fsl** capability also returns the cursor to where it was before the status line load operation was executed.

- The **esl** capability turns on the status line display.

- The **dsl** capability turns off (disables) the status line display.

- **slF** (status line format) defines the type of status line that the terminal has. It is only needed for terminals with unusual status lines. For nearly all terminals, where the status line data is sent directly to the terminal, the default value, 0, is used. For a Televideo 970, where the status line data must be encoded, the value of **slF** should be 1. The **slF** parameter is extremely machine-specific, and in general need not be used unless the target terminal is a Televideo 970.

- Use the **wsl** capability to define the width of the status line. If you do not specify a value, the width of the status line defaults to the terminal's standard line width.

If the clear-to-end-of-line capability has no effect on the addressable status line at li+1 (n25), the **xcl25** capability should be specified.

Standout Mode

Standout mode supports highlighting and brightening of certain words on the terminal screen.

visitype and **sg** together define the control of standout mode on the target terminal. **visitype** can define five types of standout mode implementation:

0   No standout mode for this terminal

1   This terminal type requires one null screen position
        in order to begin or end a defined standout field.
        Moving the cursor from field to field does not change
        the attribute of any field.

2   This is similar to type 1, but standout mode ends at
        the end of the line.

3   Standout mode may begin and end without leaving
        any blank screen positions. There is no defined stan-
        dout field as in types 1 and 2. Once one mode is
        entered, that mode affects all characters sent to the
        screen until another mode is entered.

4   This type is similar to type 3. However, clear opera-
        tions such as erase-to-end-of-line result in blanks in
        the current visual attribute, whereas in type 3 termi-
        nals the resulting blanks are in the default color.

5   Implies hp2621 standout mode implementation.

Examples of terminals differentiated by **visitype** are:

0   -   ADM 3a
1   -   TVI 950 (with \EGx visual attributes), Wyse 100
2   -   TVI 910 or ADM 31a
3   -   ANSI 3.64 (TVI 970, VT-100, VTZ 2/10, etc.)
4   -   TVI 924, TVI 950 with \E (and \E) visual attributes
5   -   HP2621 only

If **visitype** is not defined, it defaults to type 0. If **visitype** is defined to
be type 3 or 4, **sg** must be 0. Otherwise, an error message is displayed, and
customization proceeds with **visitype** forced to type 0.

Other errors that may cause **visitype** to be forced to 0 are:

- **se** is undefined

- **so** is undefined

- **se** and **so** have the same definition

Danger Spot Considerations

If a terminal has automatic margins, printing a character in the lower right corner may cause it to scroll up. The **nodanger** boolean capability may be specified so that the terminal will not scroll anyway, even though **am** (automatic margins) is true.

Character insertion can be used to display the "danger spot" character; in this case **TE3279** positions the cursor one spot behind the danger spot, outputs the "danger spot" character, backspaces, then inserts the character that belongs there.

Alternatively, if the **dset** and **dunset** string capabilities are specified, **TE3279** issues the **dset** sequence, outputs the "danger spot" character, then issues the **dunset** sequence. For example, for the vt100, we have **dset=\E[?71:dunset=\E[?7h:**, which turns automatic margins off and on (normal operations are more efficient with **am** than without it).

For future use, the boolean **bigwrap** will specify that not only is there no danger, but the cursor will jump from the last position of the screen to the "home" position when a character is printed there. (The terminal itself emulates 3270 buffer-wrap).

Initialization

The **is** and **es** capabilities define character sequences to be sent to the terminal on entry to and exit from the 3278/9 Terminal Emulator. When the 3278/9 Terminal Emulator is running, the **es** sequence is also sent when you escape to the shell or CTRL to the Terminal Function Selection Menu, and **is** is sent on return to a session.

The **terminfo is** and **es** definitions are NOT used by **scinit**. Instead, these capabilities may be defined in the source file in order to perform terminal initialization in a manner specific to 3278/9 emulation.

Other Screen Control Considerations

The **clon** and **cloff** attributes usually require large amounts of padding, particularly when keyclick is controlled by an 8048 micro-processor residing in a detached keyboard (a common case). If the target terminal's user's guide does not specify the amount of padding required for this or any other control function, the user initially should specify either no padding or a large amount of padding, and then, on successive attempts, increment or decrement the count until the correct amount is determined.

In defining the insert operation, either insert character (**ic**) or insert mode (**im**) should be specified; one should not specify both. If both are available, **ic** should be chosen as it is faster.

If the target terminal is able to clear the screen with spaces rather than nulls, it is better to use spaces.

Therefore, the clear operations (**cd, ce, cl**) should be defined to clear with spaces.

Figure 5-2 lists all the screen control capabilities that can be defined for a target terminal. The **scinit** defaults are shown in the "Default" column. The "Sample Definition" column provides an example of a possible screen control definition for the capability. This sample definition is for a target terminal with the following characteristics:

- no keyclick attribute
- cursor appearance cannot be changed
- requires no null screen position in entering or leaving standout mode (**sg** default)
- the visual attribute type (**visitype**) is 3

Screen Capability	Type	Description	Default	Sample Definition
clon	str	keyclick on		
cloff	str	disableoff		
esl	str	display status line if 'hs'		
slF	num	status line data format if 'hs'	"0"	
xc125	bool	clear 25th line glitch	false	
n25	bool	addressable stat line exists at 'li'+1	false unless "li#25"	
statline	num	status line # if neither 'hs' nor 'n25'	"25" if "li#25" else "1"	24
visitype	num	visual attribute # behavior type	see above description	3
cubl	str	cursor left		^H
cb	str	clear to beginning of line from cursor position (ANSI 3.64)		\E[1K
dset	str	remove danger of scrolling		\E[?71
dunset	str	return to normal mode		\E[?7h
nodanger	bool	no scrolling danger despite "am"		
alt_to (model 5)	str	bring terminal to 132 column mode		E[?3 h
alt_fr (model 5)	str	bring terminal to 80 column mode		
red (base color)	str	change characters to red		
grn (base color)	str	change characters to green		E[32m
blu (base color)	str	change characters to blue		E[34m

Screen Capability	Type	Description	Default	Sample Definition
wht (base color)	str	change characters to white		E[37m
pnk (ext. color)	str	change characters to pink		E[35m
ylw (ext. color)	str	change characters to yellow		E[33m
tqs (ext. color)	str	change characters to turquoise		E[36m
bnk_to (ext. attr.)	str	blinking on		E[5m
bnk_fr (ext. attr.)	str	blinking off		E[25m
rev_to (ext.attr.)	str	reverse video on		E[7m
rev_fr (ext. attr.)	str	reverse video off		E[27m
undsc_to (ext. attr.)	str	underscore on		E[4m
undsc_fr (ext. attr.)	str	underscore off		E[24m
rcol (ext. attr.)	str	reset to power-on attributes		
BBCR	str	block blinking cursor		
BSCR	str	block static cursor		
UBCR	str	underline blink cursor		
USCR	str	underline static cursor		

Figure 5-2: 3278/9 Screen Capabilities

The following attributes define the echo strings that are used for character display. They must occupy exactly one screen position. They are defined by specifying the capability followed by an equal sign (=) and the definition.

Capability	Type	Description	Default	Sample Definition
SC_0 to		numeric chars	"0" to	
SC_9			"9"	
SC_a to		lower-case alpha-	"a" to	
SC_z		betic characters	"z"	
SC_A to		upper-case alpha-	"A" to	
SC_Z		betic characters	"Z"	
SC_AMPERS		ampersand	"&"	
SC_AT		at sign	"@"	
SC_BANG		exclamation point	"!"	
SC_BOPHE		back apostrophe	"'"	
SC_BSLASH		backslash	"\"	
SC_CARAT		carat/up-arrow	"^"	
SC_CENT		cent sign '¢'	"^"	\E\030
SC_COLON		colon	":"	
SC_COMMA		comma	","	
SC_DASH		minus sign	"-"	
SC_DOLLAR		dollar sign	"$"	
SC_DOT		period	"."	
SC_DUP		3278 DUP '*'	"*"	\022
SC_EQUAL		equal sign	"="	
SC_FM		3278 FIELD MARK	";"	^\E 31
SC_GT		greater than sign	">"	
SC_LCURL		left curly brckt	"{"	
SC_LPAR		left parenthesis	"("	
SC_LSQ		left sqr brckt	"["	
SC_LT		less than sign	"<"	
SC_NOT		NOT sign '7'	"]"	\033\025
SC_OCTO		pound sign	"#"	
SC_OTH		unrecognized date	"-"	

Capability	Type	Description	Default	Sample Definition
SC_PLUS		plus sign	"+"	
SC_PRCNT		percent sign	"%"	
SC_QUEST		question mark	"?"	
SC_QUOTE		double quote	"""	
SC_RCURL		right curly brckt	"}"	
SC_RPAR		right parenthesis	")"	
SC_RSQ		right sqr brckt	"]"	
SC_SEMI		semicolon	";"	
SC_SLASH		slash	"/"	
SC_SOLID		solid vert bar ↑	"["	
SC_SPACE		space/blank	" "	
SC_SPLIT		OR sign	"↑"	
SC_STAR		asterisk	"*"	
SC_STROPHE		apostrophe		"'"
SC_TILDE		tilde	"~"	
SC_UNDER		underscore	"_"	

The following standard **terminfo** capabilities are used in the 3278/9 customization process.

terminfo Capability Name	Type	Description	Default	Sample Definition
am	bool	automatic margins	false	am
bw	bool	backspace wraps from col 0 to last col on previous line	false	bw
ed	str	clear to end of display		\EJ

terminfo Capability Name	Type	Description	Default	Sample Definition
el	str	clear to end of line		\EK
clear	str	clear screen		\EH\EJ
cup	str	move cursor		\E&a%2r%2C
cols	num	# of cols in screen width		80
dch1	str	delete char		\EP
cud1	str	cursor down		\EB
rmir	str	end ins mode		\ER
ich1	str	insert char		
smir	str	enter ins mode		\EQ
is1	str	term init string		\E&jB\r
lines	num	# of lines on screen		24
msgr	bool	safe to move while in standout mode	false	
cuf1	str	cursor right		\EC
pad	str	pad char	null	
smso	str	end standout mode		\E&d@
xmc	num	# of blank chars left by rmso/smso	"0"	
rmso	str	begin standout mode		\E&dA
cuu1	str	cursor up		\EA
flash	str	visible bell (may not move cursor)		^G
xhp	bool	standout not erased by	false	

terminfo Capability Name	Type	Description	Default	Sample Definition
		overwrite*		
ns	bool	screen does not scroll	false	
tsl	str	start of status line load seq if 'hs'		
fsl	str	end of status line load seq if 'hs'		
dsl	str	disable status line if 'hs'		
wsl	num	width of status line if not==co	co	
bel	str	bell with pad		
hs	bool	special status line	false	

* This is a restriction on Hewlett-Packard terminals.

Using the scinit Utility

The **scinit** utility accepts screen control definitions from three sources. In order of decreasing priority these sources are:

- The screen control source file.

- The standard **terminfo** description file for the target terminal.

- The **scinit** program defaults (see the default column in Figure 5-2).

If a definition is not obtained from one of these three sources, the screen control capability will be left undefined.

scinit Output

The **scinit** utility generates two types of output:

- a customized screen control object file, and

- a list of error messages generated by invalid **scinit** conditions (see Appendix G). A message appears when a screen capability definition does not provide adequate support for emulation.

Invoking scinit

The **scinit** utility creates a binary file that defines to the TE3279 process the mapping to be used in emulating the 3278/9 screen on the target terminal. To invoke **scinit**, use the command format:

scinit [−0] [−n*x*] [−s *source_file*] [−o *object_file*] [−t *tty_name*] [−b *tty_baud*]

The following is a complete list of options to **scinit**:

−0	Cancels default values for all **scinit** options for which default values were specified.
−n*x*	Cancels the default values for the option name *x* that follows this option. More than one option name may be specified on a string format. For example, to cancel the default −s and −o options, specify −**nso**.
−s *source_file*	*source_file* is the name of the screen control source file. If the −s option is omitted, the environment variable **SCI3279** is used to get the input file pathname. If the variable is not defined in the environment, a file named **SCI3279** is sought. The source file definitions override **scinit** program default definitions.
−o *object_file*	*object_file* is the object file name. If the named file does not exist, **scinit** creates it dynamically. The output file is the file used by the terminal process to control the 3278 emulation of the target terminal's display screen. If the −o option is omitted, the environment variable **SC3279** is used to get

the output file pathname. If the variable is not defined in the environment, SC3279 is used as the output file pathname.

−**t** *tty_name*

tty_name is the standard **terminfo** name of the target terminal. If the −**t** option is omitted, the environment variable **TERM** is used to obtain the target terminal name. If the terminal currently in use is not the same as the target terminal, this option must be used; otherwise, the customization file may be compiled with incorrect **terminfo** defaults.

−**b** *tty_baud*

tty_baud is the target terminal's baud rate setting. If this option is omitted, **scinit** searches the screen control file for the ospeed capability to determine the baud rate. If ospeed is not defined in the file, the baud rate for **stdin** is used if **stdin** is a terminal device. Otherwise, the baud rate of the current system console terminal is used.

A   Documentation

Documentation

AT&T 3270 Emulator+

- *AT&T 3270 Emulator+ User's and System Administrator's Guides* (select code 308-400)

- *AT&T 3270 Emulator+ HLLAPI Programmer's Guide* (select code 308-332)

- *AT&T 3270 Emulator+ Product Overview* (select code 308-001)

- *AT&T 3270 Emulator+ 3B2 Release Notes* (select code 308-002)

- *AT&T 3270 Emulator+ 3B5/3B15/3B4000 Release Notes* (select code 308-335)

Intelligent Serial Controller

- *AT&T 3B2 Computer Intelligent Serial Controller Manual* (select code 305-531)

3B Computers

- *AT&T 3B2/300 Computer Owner/Operator Manual* (select code 305-301)

- *AT&T 3B5/300 User Guide and Essential Utilities (select code 305-302)*

UNIX System V

Documents describing UNIX System V are available from the Customer Information Center (see "Ordering AT&T Documentation" below).

IBM 3270 and SNA Documentation

The following IBM publications are relevant:

- *An Introduction to the IBM 3270 Information Display System* (GA27-2739).

- *IBM 3274 Control Unit Planning, Setup, and Customizing Guide* (GA27-2827).

- *IBM 3270 Information Display System, 3274 Control Unit Description and Programmer's Guide* (GA23-0061)

- *Systems Network Architecture Concepts and Products* (GC30-3072).

- *Systems Network Architecture Technical Overview* (GC30-3073).

- *Systems Network Architecture Reference Summary* (GA27-3136).

- *IBM Systems Network Architecture Format and Protocol Reference Manual* (SC30-3112).

- *IBM Synchronous Data Link Control General Information* (GA27-3093).

Ordering AT&T Documentation

AT&T documentation may be ordered by calling the AT&T Customer Information Center (CIC) at:

1-800-432-6600 (toll free within the continental United States)

or by writing to:

AT&T Customer Information Center
Customer Service Representative
P.O. Box 19901
Indianapolis, Indiana 46219

B   Variables

Environment Variables

The following is a list of the environment variables used by the AT&T SNA/3270 and BSC/3270 processes. If a variable is set by the **snaenvset** or **bscenvset** script, the default setting is given below; otherwise an example is provided. The variables that are used only with SNA or BSC are listed separately at the end.

cust
This is the base path to the customization directory. It is a composite of the **sna** environment variable and "**cust**" for SNA, or the **bsc** variable and "**cust**" for BSC.

SNA Default: **/usr/snaadm/cust**

BSC Default: **/usr/bscadm/cust**

PCD
On a 3B2 Computer this variable is the number of the 3B2 slot in which the ISC board that is to be used is installed. It is referenced by **snaenvcust** and **bscenvcust** to set the value of **P3274**, and by the controller start and stop scripts. If its value is changed, either **snaenvcust** or **bscenvcust**, whichever is appropriate, should be run again. For both SNA and BSC, **PCD** defaults to the lowest numbered slot in the 3B2 Computer that has an ISC card in it.

On a 3B5 or 3B15 Computer, the variable specifies the IOA device number which is to be used to communicate with the host. If this value is changed, PCD should be exported again. Depending on whether you are using BSC or SNA, the default value is stored in a file named **PCD.file** in the **bscadm** or **snaadm** directory. The value in this file is set up during installation of the product, but may be edited to change the system default.

On a 3B4000 Computer, **PCD** is a

combination of the **pe** and ISC slot number (e.g., 120.6 for **pe** 120, slot 6).

PATH

This is the standard UNIX system environment variable, modified to include the value of the **rt** environment variable as one of its entries. Each user's PATH variable should be set to include the appropriate SNA/3270 or BSC/3270 **runtime** directory.

Example:
PATH=/etc:/bin:/usr/snaadm/runtime:/usr/bin

P3274

This specifies the named pipe used for communications between the 3278/9 Terminal Emulator process, **TE3279** and the SNA Controller process **SNA**, or between **TE3279** and the BSC Controller process **TM3274**. **P3274** should be appropriately set before starting or stopping a controller. **P3274** should be appropriately set in the user's environment before starting the terminal emulator.

This value is closely tied to the value of **PCD**. PCD is set to x and **snaenvcust** or **bscenvcust** run. This will set **P3274** to /tmp/P3274.x.

rt

This is the base path to the runtime directory. It is a composite of the value of the **sna** variable and "**runtime**" when using SNA, or the value of the **bsc** variable and "**runtime**" when using BSC.

SNA Default: **rt=/usr/snaadm/runtime**

BSC Default: **rt=/usr/bscadm/runtime**

KY3279

Specifies the keyboard mapping object file to be used when the 3278/9 Terminal Emulator process (**TE3279**) is invoked by its start-script (**te3279**).

	Default: $rt/KY.<$TERM>, or $rt/KY.std if $TERM does not exist.
SC3279	Specifies the screen control object file to be used when the 3278/9 Terminal Emulator is invoked by its start-script (te3279).
	Default: $rt/SC.$TERM
PF3274	Specifies the default file to which local print output will be directed when using the 3278/9 Terminal Emulator.
	Default: the value of the **prnt** variable
prnt	This variable is used to set the default for the PF3274 variable.
	Default: /tmp/PF3274.$LOGNAME
TM3279	Specifies the IBM 3278 or 3279 model number that will be emulated when the 3278/9 Terminal Emulator is invoked. It has no default, but if the 3278/9 Terminal Emulator is started and TM3279 is not set, model 2 is assumed.
TERM	This is a standard UNIX system environment variable that must be set for your terminal type and exported.
LOGNAME	This is a standard UNIX system environment variable that is set by the system to your login name.

SNA-Specific Variables

CONFIG

This is the full pathname of the configuration object file to be used when starting the SNA Controller; it is referenced by **snopts** and **startsna**. CONFIG is a composite of the value of the **rt** variable, "CSNA." and the value of the **SNAHOST** variable. (This variable is referenced by **snopts** and **startsna**.)

Default: /usr/snaadm/runtime/CSNA.teke

sna

Specifies the base path for the SNA/3270 directories. It is obtained from the home directory of the **snaadm** login as listed in /etc/passwd.

Default: /usr/snaadm

SNAHOST

SNAHOST is used to change the CONFIG variable, which specifies the full pathname of the controller configuration object file to be used when the SNA Controller is started. Configuration object files are named with the prefix **CSNA.**, for example, **CSNA.hostx**. SNAHOST is used to specify which of the "CSNA." files is the one to be used. For example, if **CSNA.hostx** (located in /usr/snaadm/runtime) is to be used, SNAHOST should be set equal to **hostx**. When the **snaenvcust** script is executed, CONFIG will then be appropriately set to /usr/snaadm/runtime/CSNA.hostx.

Default: **teke**

BSC-Specific Variables

BS3274

This is the full pathname of the configuration object file to be used when starting the BSC Controller; it is a composite of the value of the **rt** variable, "BS3274.," and the value of the **BSCHOST** variable. (This variable is referenced by **bsopts** and **startbsc**.)

Default: /usr/bscadm/runtime/BS3274.teke

bsc

Specifies the base path for the BSC/3270 directories. It is obtained from the home directory of the **bscadm** login as listed in /etc/passwd.

Default: /usr/bscadm

BSCHOST

BSCHOST is used to change the **BS3274** variable, which specifies the full pathname of the controller configuration object file to be used when the BSC Controller is started. Configuration object files are named with the prefix **BS3274.**, for example, **BS3274.hostx**. **BSCHOST** is used to specify which of the "BS3274." files is the one to be used. For example, if **BS3274.hostx** (located in /usr/bscadm/runtime) is to be used, **BSCHOST** should be set equal to **hostx**. When the **bscenvcust** script is executed, **BS3274** will then be appropriately set to /usr/bscadm/runtime/BS3274.hostx.

Default: **teke**

C   Sample Keyboard Source File

Sample Keyboard Source File

This appendix contains a sample **kyinit** input file.

```
sample:\
     :KY_ALLCAP = \Em:\
     :KY_ALT_CR = \En:\
     :KY_ATTN = \Ea^m:\
     :KY_BAKTAB = ^e:\
     :KY_BLINK = \Eb:\
     :KY_BS =^h:\
     :KY_CAN = ^x:\
     :KY_CENT = \ ^:\
     :KY_CLEAR = \Ez:\
     :KY_CLICK = \Ek:\
     :KY_CURSR_SEL =\Ec^m:\
     :KY_DEL = \177:\
     :KY_DEV_CNCL = \Ed:\
     :KY_DOWN_A = ^v:\
     :KY_DUP = ^d:\
     :KY_ENTER1 = ^m:\
     :KY_ENTER = ^m:\
     :KY_EXIT = \Exx:\
     :KY_E_EOF = \Eef:\
     :KY_E_INPUT = \Eei:\
     :KY_FM = ^k:\
     :KY_HOME = ^o:\
     :KY_IDENT = \Ei:\
     :KY_INS = ^u:\
     :KY_KY_ERR = \200:/
     :KY_LDUB = ^r:\
     :KY_LEFT_A = ^f:\
     :KY_NEWL = ^j:\
     :KY_NOT = ]:\
     :KY_NULLEND = ^b:\
     :KY_NUM_OV = ^n:\
     :KY_PA1 = \Ea1^m:\
     :KY_PA2 = \Ea2^m:\
     :KY_PA3 = \Ea3^m:\
     :KY_PF10 = \E10^m:\
     :KY_PF11 = \E11^m:\
     :KY_PF12 = \E12^m:\
```

```
:KY_PF13 = \E13^m:\
:KY_PF14 = \E14^m:\
:KY_PF15 = \E15^m:\
:KY_PF16 = \E16^m:\
:KY_PF17 = \E17^m:\
:KY_PF18 = \E18^m:\
:KY_PF19 = \E19^m:\
:KY_PF1 = \E1^m:\
:KY_PF20 = \E20^m:\
:KY_PF21 = \E21^m:\
:KY_PF22 = \E22^m:\
:KY_PF23 = \E23^m:\
:KY_PF24 = \E24^m:\
:KY_PF2 = \E2^m:\
:KY_PF3 = \E3^m:\
:KY_PF4 = \E4^m:\
:KY_PF5 = \E5^m:\
:KY_PF6 = \E6^m:\
:KY_PF7 = \E7^m:\
:KY_PF8 = \E8^m:\
:KY_PF9 = \E9^m:\
:KY_PRINT = \Ep:\
:KY_RDUB = ^y:\
:KY_REDRAW = \Er:\
:KY_RESET = ^a:\
:KY_RIGHT_A = ^g:\
:KY_SHELL = \Es:\
:KY_SOLID = [:\
:KY_STAT = \El:\
:KY_STMV = \Eh:\
:KY_SYS_REQ = \Eq:\
:KY_TAB = ^i:\
:KY_UP_A = ^t:\
```

D Sample Screen Control Source File

Sample Screen Control Source File

This appendix contains a sample **scinit** input file.

```
sample1|smp1|sample terminal 1,
am,bw,
ed = \E[J,
el = \E[K,
clear = \E[H\E[J,
cup = \E[%i%p1%d;%p2%dH,
cols#80,
dch1 = \E[P,
cud1 = \n,
rmir = \E[R,
smir = \E[Q,
lines#24,
culf = \E[C,
rmso = \E[m,
smso = \E[7m,
cuu1 = \E[A,
:is = \E&jB\r:\
:statline#25:\
:cub1 = bs:\
:SC_CENT = ^nc^O:SC_DUP = ^n/^O:\
:SC_FM = ^ni^O:SC_NOT = ^nkO:SC_SOLID = ^nxO:\
```

E Key Sequences

Key Sequences for Standard ASCII Terminals

The AT&T 4415 and 5420, and the Tektronix 4105 terminals use these key sequences. Certain key functions, such as COLR, are ignored if the terminal does not support them. Note that any terminal can use KY.std.

3278/9 Key Function	Standard ASCII Terminal Key Sequence
ALLCAP	\<ESC\> m
ALT_CR	\<ESC\> n
ATTN	\<ESC\> a \<RETURN\>
BAKTAB	\<CTRL\> e
BLINK	\<ESC\> b l
BOT	\<ESC\> bot
BS	\<CTRL\> h
CAN	\<CTRL\> x
CENT	^
CLEAR	\<ESC\> z
CLICK	\<ESC\> k
COLR	\<ESC\> clr
CTRL	\<ESC\> fd
CURSR_SEL	\<ESC\> c \<RETURN\>
DEL	\<DEL\>
DEV_CNCL	\<ESC\> d
DOWN_A	\<CTRL\> v
DUP	\<CTRL\> d
E_EOF	\<ESC\> e f
E_INPUT	\<ESC\> e i
ENTER	\<RETURN\>
ENTER1	\<RETURN\>
EXIT	\<ESC\> x x
FM	\<CTRL\> k
HOME	\<CTRL\> o
IDENT	\<ESC\> i
INS	\<CTRL\> u
LDUB	\<CTRL\> r
LEFT_A	\<CTRL\> f

3278/9 Key Function	Standard ASCII Terminal Key Sequence
NEWL	<CTRL> j
NEXTS	<ESC> +
NOT	]
NULLEND	<CTRL> b
NUM_OV	<CTRL> n
PA1 TO PA3	<ESC> a *key number* <RETURN>
PF1 to PF24	<ESC> *key number* <RETURN>
PREVS	<ESC> +
PRINT	<ESC> p
RDUB	<CTRL> y
REDRAW	<ESC> r
RESET	<CTRL> a
RIGHT_A	<CTRL> g
SHELL	<ESC> s
SOLID	[
STAT	<ESC> l
SYS_REQ (SNA only)	<ESC> q
TAB	<CTRL> i
TEST_REQ (BSC only)	<ESC> q
TOP	<ESC> top
UP_A	<CTRL> t

Figure E-1: AT&T 3278/9 Key Sequences For Standard ASCII Terminals (using KY.std).

Key Sequences for AT&T 4410 and Teletype 5410 Terminals

3278/9 Key Function	AT&T 4410 and Teletype 5410 Key Sequence
ALLCAP	\<ESC> m
ALT_CR	\<ESC> n
ATTN	\<ESC> a \<RETURN>
BAKTAB	\<DOWN DIAGONAL ARROW>
BS	\<CTRL> ^h
CAN	\<CTRL> x
CENT	^
CLEAR	\<ESC> z
CLICK	\<ESC> k
CTRL	\<ESC> f d
CURSR_SEL	\<ESC> c \<RETURN>
DEL	\<DEL>
DEV_CNCL	\<ESC> d
DOWN_A	\<DOWN ARROW>
DUP	\<CTRL> d
E_EOF	\<ESC> e f
E_INPUT	\<ESC> e i
ENTER	\<RETURN>
ENTER1	\<RETURN>
EXIT	\<ESC> x x
FM	\<CTRL> k
HOME	\<UP DIAGONAL ARROW>
IDENT	\<ESC> i
INS	\<CTRL> u
LDUB	\<CTRL> r
LEFT_A	\<LEFT ARROW>
NEWL	\<CTRL> j
NEXTS	\<ESC> >
NOT	]
NULLEND	\<CTRL> b
NUM_OV	\<CTRL> n

3278/9 Key Function	AT&T 4410 and Teletype 5410 Key Sequence
PA1 to PA3	<ESC> a *key number* <RETURN>
PF1 to PF9	<ESC> *key number*
PF10	<ESC> 0
PF11	<ESC> -
PF12	<ESC> =
PF13	<ESC> <SHIFT> 1
PF14	<ESC> <SHIFT> 2
PF15	<ESC> <SHIFT> 3
PF16	<ESC> <SHIFT> 4
PF17	<ESC> <SHIFT> 5
PF18	<ESC> <SHIFT> 6
PF19	<ESC> <SHIFT> 7
PF20	<ESC> <SHIFT> 8
PF21	<ESC> <SHIFT> 9
PF22	<ESC> <SHIFT> 0
PF23	<ESC> <SHIFT> -
PF24	<ESC> <SHIFT> =
PREVS	<ESC> <
PRINT	<ESC> p
RDUB	<CTRL> y
REDRAW	<ESC> r
RESET	<CTRL> a
RIGHT_A	<RIGHT ARROW>
SHELL	<ESC> s
SOLID	[
STAT	<ESC> l
SYS_REQ (SNA only)	<ESC> q
TAB	<CTRL> i
TEST_REQ (BSC only)	
UP_A	<UP ARROW>

Figure E-2: AT&T 3278/9 Key Sequences for AT&T 4410 and Teletype 5410 Terminals

Key Sequences for AT&T 4418 and Teletype 5418 Terminals

3278/9 Key Function	AT&T 4418 and Teletype 5418 Key Sequence
ALLCAP	<ALT CSR>
ATTN	<ATTN>
BAKTAB	<BACK TAB>
BANG	!
BS	<BACK SPACE>
CAN	<CTRL> x
CENT	^
CLEAR	<CLEAR>
CTRL	<ESC> f d
CURSR_SEL	<CSR SEL>
DEL	<DEL>
DEV_CNCL	<DEV CNCL>
DOWN_A	<DOWN ARROW>
DUP	<DUP>
E_EOF	<ERASE EOF>
E_INPUT	<ERASE INPUT>
ENTER	<ENTER> (lower right side)
ENTER1	<ENTER> (upper left side)
EXIT	<ESC> x x
FM	<FIELD MARK>
HOME	<HOME>
IDENT	<IDENT>
INS	<INS>
LDUB	<LEFT DOUBLE ARROW>
LEFT_A	<LEFT ARROW>
NEWL	<NEW LINE>
NEXTS	<ESC> +
NOT	]
NULLEND	<CTRL> b
NUM_OV	<CTRL> n
PA1	<PA1>

3278/9 Key Function	AT&T 4418 and Teletype 5418 Key Sequence
PA2	<PA2>
PA3	<SHIFT> <INS>
PF1 to PF24	<PF1> TO <PF24>
PREVS	<ESC> −
PRINT	<PRINT LCL>
RDUB	<RIGHT DOUBLE ARROW>
REDRAW	<ESC> r
RESET	<RESET> (lower left side)
RIGHT_A	<RIGHT ARROW>
SHELL	<SHIFT> <RESET> (upper left side)
SOLID	[
STAT	<RESET> (upper left side)
SYS_REQ (SNA only)	<SYS REQ>
TAB	<CURSOR TAB>
TEST_REQ (BSC only)	
UP_A	<UP ARROW>

Figure E-3: AT&T 3278/9 Key Sequences for AT&T 4418 and Teletype 5418 Terminals

──

NOTE
On these terminals no key is marked ESC or CTRL. For ESC, hold ALT and press [. For CTRL use the key immediately to the left of the space bar.

Key Sequences for AT&T 4425 and Teletype 5425 Terminals

3278/9 Key Function	AT&T 4425 and Teletype 5425 Key Sequence
ALLCAP	\<ESC\> m
ATTN	\<ESC\> a \<RETURN\>
BAKTAB	\<SHIFT\> \<TAB\>
BOT	\<ESC\> \<btm\>
BS	\<BACK SPACE\>
CAN	\<CTRL\> x
CENT	^
CLEAR	\<CLEAR\>
CTRL	\<ESC\> f d
CURSR_SEL	\<ESC\> c \<RETURN\>
DEL	\<DEL\>
DEV_CNCL	\<ESC\> d
DOWN_A	\<DOWN ARROW\>
DUP	\<CTRL\> d
E_EOF	\<CLEAR LINE\>
E_INPUT	\<DELETE LINE\>
ENTER	\<RETURN\>
ENTER1	\<ENTER\> (on keypad)
EXIT	\<ESC\> x x
FM	\<CTRL\> k
HOME	\<HOME\>
IDENT	\<ESC\> i
INS	\<INSERT CHAR\>
LDUB	\<CTRL\> r
LEFT_A	\<LEFT ARROW\>
NEWL	\<CTRL\> j
NEXTS	\<ESC\> +
NOT	]
NULLEND	\<INSERT LINE\>
NUM_OV	\<CTRL\> ñ
PA1 TO PA3	\<ESC\> a *key number* \<RETURN\>

3278/9 Key Function	AT&T 4425 and Teletype 5425 Key Sequence
PF1 TO PF4	<PF1> to <PF4>
PF5	7 (on keypad only)
PF6	8 (on keypad only)
PF7	9 (on keypad only)
PF8	- (on keypad only)
PF9	4 (on keypad only)
PF10	5 (on keypad only)
PF11	6 (on keypad only)
PF12	, (on keypad only)
PF13 TO PF24	<ESC> *key number* <RETURN>
PREVS	<ESC> −
PRINT	<ESC> p
RDUB	<CTRL> y
REDRAW	<ESC> r
RESET	<CTRL> a
RIGHT_A	<RIGHT ARROW>
SHELL	<ESC> s
SOLID	[
STAT	<ESC> l
SYS_REQ (SNA only)	<ESC> q
TAB	<TAB>
TEST_REQ (BSC only)	<ESC> q
TOP	<ESC> <top>
UP_A	<UP ARROW>

Figure E-4: AT&T 3278/9 Key Sequences for AT&T 4425 and Teletype 5425 Terminals

Key Sequences for AT&T 605 Business Communications Terminal with 102-Key Keyboard

3278/9 Key Function	AT&T 605 Terminal Key Sequence
ALLCAP	<ESC> m
ATTN	<ESC> a <RETURN>
BAKTAB	<SHIFT> <TAB>
BOT	<ESC> bot
BS	<BACK SPACE>
CAN	<CTRL> x
CENT	^
CLEAR	<SHIFT> <CLEAR>
CTRL	<ESC> f d
CURSR_SEL	<ESC> c <RETURN>
DEL	<CTRL> <DEL>
DEV_CNCL	<ESC> d
DOWN_A	<DOWN ARROW>
DUP	<CTRL> d
E_EOF	<ESC> e f
E_INPUT	<SHIFT> <DELETE LINE>
ENTER	<RETURN>
ENTER1	<RETURN>
EXIT	<ESC> x x
FM	<CTRL> k
HOME	<CLEAR> <HOME>
IDENT	<ESC> i
INS	<INSERT LINE>
LDUB	<CTRL> r
LEFT_A	<LEFT ARROW>
NEWL	<CTRL> j
NEXTS	<ESC> +
NOT	]
NULLEND	<SHIFT> <INSERT>

3278/9 Key Function	AT&T 605 Terminal Key Sequence
NUM_OV	<CTRL> n
PA1 TO PA3	<ESC> a *key number* <RETURN>
PF1 TO PF24	<ESC> *key number* <RETURN>
PREVS	<ESC> -
PRINT	<ESC> p
RDUB	<CTRL> y
REDRAW	<ESC> r
RESET	<ESC> c
RIGHT_A	<RIGHT ARROW>
SHELL	<ESC> s
SOLID	[
STAT	<ESC> l
SYS_REQ (SNA only)	<ESC> q
TAB	<TAB>
TEST_REQ (BSC only)	<ESC> q
TOP	<ESC> top
UP_A	<UP ARROW>

Figure E-5: AT&T 3278/9 Key Sequences for AT&T 605 Business Communications Terminal with 102-Key Keyboard

Key Sequences for AT&T 610, 615, 620, and 630 Terminals with 98-Key Keyboard

3278/9 Key Function	AT&T 620, 615, 620 and 630 Terminals Key Sequence
ALLCAP	<ESC> m
ATTN	<ESC> a <RETURN>
BAKTAB	<SHIFT> <TAB>
BOT	<ESC> bot
BS	<BACK SPACE>
CAN	<CTRL> x
CENT	^
CLEAR	<CLEAR>
CTRL	<ESC> f d
CURSR_SEL	<ESC> c <RETURN>
DEL	<DEL>
DEV_CNCL	<ESC> d
DOWN_A	<DOWN ARROW>
DUP	<CTRL> d
E_EOF	<ESC> e f
E_INPUT	<ESC> e i
ENTER	<RETURN>
ENTER1	<RETURN>
EXIT	<ESC> x x
FM	<CTRL> k
HOME	<HOME>
IDENT	<ESC> i
INS	<CTRL> u
LDUB	<CTRL> r
LEFT_A	<LEFT ARROW>
NEWL	<CTRL> j
NEXTS	<ESC> +
NOT	]
NULLEND	<CTRL> b
NUM_OV	<CTRL> n
PA1 TO PA3	<ESC> a *key number* <RETURN>

3278/9 Key Function	AT&T 620, 615, 620 and 630 Terminals Key Sequence
PF1 TO PF24	\<ESC> *key number* \<RETURN>
PREVS	\<ESC> -
PRINT	\<ESC> p
RDUB	\<CTRL> y
REDRAW	\<ESC> r
RESET	\<ESC> c
RIGHT_A	\<RIGHT ARROW>
SHELL	\<ESC> s
SOLID	[
STAT	\<ESC> l
SYS_REQ (SNA only)	\<ESC> q
TAB	\<TAB>
TEST_REQ (BSC only)	\<ESC> q
TOP	\<ESC> top
UP_A	\<UP ARROW>

Figure E-6: AT&T 3278/9 Key Sequences for AT&T 610, 615, 620 and 630
Terminals with 98-Key Keyboard

F Sample kyinit Report File

Sample kyinit Report File

This appendix contains excerpts from the **kyinit** report file.

```
TABLE LENGTH is 606 BYTES

KY_SPACE space (blank)
CURRENT VALUE == <SPACE>

KY_CENT cent sign (not an ASCII character)    .
CURRENT VALUE == <^>

KY_DOT .(period)
CURRENT VALUE == <.>
```

G Non-supportable Terminal Messages

Non-Supportable Terminal Messages G-1

Non-Supportable Terminal Messages

This section lists messages generated by **scinit** for non-supportable terminals, with the cause of the error. Items that appear in an error message in parentheses are the values of the options that were used in the **scinit** session. The error messages are listed in the order in which the errors are detected during execution. If one of these messages is generated during execution of **scinit,** you will have to define or redefine the corresponding capability to proceed with customization.

Message

    can't do 3270 with (sg_value) space on attribute

Type of Error:

    Invalid sg Capability Definition

Explanation:

    More than one blank character is left by starting or ending standout mode. **scinit** does not abort.

Message:

    3278 will not work on this terminal. No clear screen operation.

Type of Error:

    Missing cl Capability Definition

Explanation:

    The clear screen capability is not defined. **scinit** is aborted.

Message

3278 will not work on this terminal. No direct cursor addressing.

Type of Error:

Missing cm Capability Definition

Explanation:

The cursor motion capability is not defined. **scinit** is aborted.

Message:

Inconsistent usage of sg and visitype options.

Type of Error:

Invalid Visual Attribute Behavior Definition

Explanation:

The relationship between sg and visitype does not conform to the description in the Screen Customization section of Chapter 5 of this manual. **scinit** continues with visitype set to type 0.

scinit Error Messages

The following list describes scinit-generated error messages and their causes. Parenthesized items in the error message represent the actual values that were used during a **scinit** session. The order in which the error messages are listed is the order in which they are detected during execution.

Message:

the **scinit** usage message is displayed.

Type of Error:

Invalid **scinit** Command Parameter

Explanation:

scinit is aborted.

Message:

tgetent fails (tgetent_value)

Type of Error:

scinit File Not Found

Explanation:

The display control file is not found. **tgetent_value** is the value returned by the function tgetent which searches for the terminal description. **scinit** is aborted if neither of the two possible input files are found.

Message:

%E1%:(write_value) returned writing (output_file_name) (errno_value) == errno.(file_size) == length.

%E2%: (write_value) returned writing (output_file_name) (errno_value) == errno.(file_size) == length.

Type of Error:

Output Errors

Explanation:

The output file is written in two stages. E1 is displayed if an error occurs while writing the first part, and E2 is displayed if an error occurs while writing the second part. **scinit** is aborted under both conditions.

H   The te3279 Command

The te3279 Command

The command to invoke the 3278/9 Terminal Emulator is:

te3279 [−t *termno*] [−s *screen_file*] [−k *key_file*] [−p *pipe*]
[−f *printer_path*] [−M *msgfile*] [−m *model*] [−e]

The following list describes each of the options in detail:

−t *termno* where *termno* specifies acceptable terminal (LU) numbers. The controller selects a terminal number from those specified. The valid range of terminal numbers is 0-31. The user may specify a list of numbers (such as 1,5,7), a range of numbers (such as 10-15), or a combination (such as 1,5,7,10-15). The default environment variable is **D3274**, and the default value is the range 0-31.

−s *screen_file* where *screen_file* is the name of the screen control object file to be used for this terminal. If this option is omitted, the **SC3279** environment variable is used to obtain the name of the file.

−k *key_file* where *key_file* is the name of the keyboard mapping object file to be used for this terminal. If this option is omitted, the **KY3279** environment variable is used to obtain the name of the file.

−p *pipe* where *pipe* is the pipe name to the controller process that te3279 will be using. This must be the pipe name created when the controller was started. If this option is omitted, the **P3274** environment variable is used to obtain the name of the pipe.

−f *printer_path* where *printer_path* is the printer pathname for local print functions. If this option is omitted, the **PF3274** environment variable is used to obtain the printer pathname. This value may be overriden by the user

on a per session basis once **te3279** has been invoked, by using the IDENT key.

−**M** *msg_file*   where *msg_file* is the message file name. If this option is omitted, the file name **te3279.msg** is used.

−**m** *model*   where *model* is the terminal model type, which may be:

2 - specifying a 24x80 screen buffer dimension

5 - specifying a 27x132 screen buffer dimension

−**e**   enables extended attribute support, namely, extended highlighting (blink, underscore, and reverse video) and extended color support. These capabilities must also be defined in the screen control object file used for your terminal.

Index

3278/9 key 4: 2; 5: 1-2, 4-6, 11, 16
3278/9 Operator Information Area 5: 18
3278/9 screen 5: 29
3278/9 Terminal Emulator 4: 1; 5: 1-2, 21;
    B: 2-3;   H: 1
3278/9 terminal mapping information 5: 1

A

ALT 5: 9;   C: 1;   E: 1, 3, 5-6
API 1: 10; 4: 6
ASCII 2: 8, 10; 3: 3; 5: 1, 6, 16;   E: 1-2;   F:
    1
AT&T 3270 Emulator 1: 10; 2: 1; 4: 5-6; 5:
    1, 5;   A: 1
AT&T 3770 Emulator 1: 5
AT&T BSC/3270 Emulator 2: 2, 15
AT&T SNA/3270 Emulator 1: 1, 5, 17
AT&T SNA/RJE Emulator 1: 5, 10
automatic margins 5: 21

B

binary file 5: 1, 11, 29
broadcast station 1: 11
BS3274 2: 3-4, 12-15;   B: 5
BSC 2: 1-6, 8-12, 14-18; 3: 3; 5: 24;   B: 1-2,
    5;   E: 2, 4, 6, 8, 10, 12
bscenvcust 2: 3-4; 4: 3-4;   B: 1-2, 5
bscenvset 2: 1-4; 4: 1, 3, 5;   B: 1
bsopts 2: 5, 10, 12-14, 17;   B: 5

C

CANCEL 5: 12-13
CENT 5: 6-7, 17, 25;   C: 1;   D: 1;   E: 1, 3,
    5, 7, 9, 11;   F: 1
clear operations 5: 20, 22
clear screen capability   G: 1
cloff 5: 23
clon 5: 22-23
COLR 5: 9;   E: 1
command line 1: 14, 16; 2: 12, 14; 3: 2; 4:
    1; 5: 10
CONFIG 1: 2, 14-17;   B: 4
configuration file 1: 4, 19; 2: 1, 5, 17
configuration object file 1: 2-4, 14-15, 19; 2:
    3-5, 12-13, 17;   B: 4-5
configuration options 1: 4; 2: 5
configuration source file 1: 4-5, 12-15; 2: 5,
    10, 13
controller configuration file 2: 1
controller options 1: 5, 10
controller type 3270 1: 10-11, 13
CTS 1: 8; 2: 9
cursor motion   G: 2
cursor movement keys 5: 5

D

danger spot 5: 21
detached keyboard 5: 22
device emulators 1: 19; 2: 18
display control file   G: 3
dset 5: 21, 23
dsl 5: 18-19, 28
dunset 5: 21, 23
DUP 5: 8, 17, 25;   C: 1;   D: 1;   E: 1, 3, 5, 7,
    9, 11

R

S

T

U

V

X