CHAPTER 4

Configuring Lines and Terminal Settings

This chapter describes how to configure the communication server for line, terminal, and modem connections. See the *Communication Server Command Reference* for a detailed explanation of the commands used in this chapter.

See the *Communication Server and Protocol Translator Connection Guide* for more information about making connections to communication servers.

Cisco's Implementation

Your communication server system software provides a full complement of line-specific commands for connecting asynchronous serial devices such as terminals, printers, and modems, and configuring custom device operation. Capability such as configuring a single line or a range of lines allows configuring for a specific functionality. For instance, you can set one line for a laser printer and then configure a set of lines to switch incoming modem connections to the next available line. Customization features such as defining line-specific transport protocols are also available, as is controlling character and packet transmissions and establishing time limits for user access.

You can place configuration commands in either the network or the host configuration files or in both types of files. Duplicating configuration information in the network and the host configuration files does not present a problem. Furthermore, you can place all configuration commands in the host configuration file, reducing the network configuration file to just the **end** keyword.

However, if conflicting configuration information exists in the network and the host configuration files, the information in the host file takes precedence. This fact allows you to configure all network hosts in general and adjust for specific hosts as necessary.

Comment lines are entered by including an exclamation point (!) followed by a space and some text, or by simply pressing the Return key to leave a blank line and improve readability.

Line Configuration Task List

One of the first things you will want to do for line configuration is to configure the lines for the terminals or other asynchronous serial devices attached to them. The line configuration tasks are listed next. Which tasks you perform and the order in which you perform them are determined entirely by the needs of your network environment.

- Prepare to Configure the Lines
- Define Terminal Operation Characteristics
- Record the Device Location
- Configure Software Flow Control

- Configure Software Flow Control Start and End Characters
- Define a Command String for Automatic Execution
- Create Character and Packet Dispatch Sequences
- Enable the Auxiliary Port
- Enable the Auxiliary Port
- Set a Line as Insecure
- Save Local Settings Between Sessions
- Set Pending Output Notification
- Establish and Control the EXEC Process
- Configure Modem Lines
- High-Speed Modem Support
- Connections to an Individual Line
- Enable Password Checking at Login
- Enable Password Checking at Login
- Display Terminal Banner Messages
- Set a Terminal-Locking Mechanism

See the system configuration chapter for instructions on setting password protection. The following sections describe the line configuration tasks. For examples of these configuration tasks, see the Examples section at the end of this chapter. See the *Communication Server Command Reference* for information about the commands in these tasks.

Prepare to Configure the Lines

You begin line configuration in command collection mode. To access this mode, follow these steps:

- Enter the EXEC command **configure** at the privileged system prompt.
- Once in the command collection mode, configure the line by entering the **line** command. Follow each **line** command with the line configuration commands that set the desired characteristics for the line. The commands are collected and applied to the **line** command until you enter another **line** command, a command that is not a line configuration command, or a **Ctrl-Z** sequence.
- Once a line is configured, you can check its status by entering the EXEC **show users all** command

To configure a line, perform the following tasks:

Task	Command
Enter global configuration mode.	**configure**
Begin line configuration.	**line** [**console** | **tty** | **aux** | **vty** | **printer**] *line-number* [*ending-line-number*]
Check the line status.	**show users all**

Absolute and relative line numbers are kept by software in a table that you can display with the EXEC command **show users all**. A truncated sample display follows.

```
cs> show users all
  Line      User     Host(s)              Idle   Location
   0 con 0                                      chaff console
   1 tty 1                                      Engineering printer
   2 tty 2
   3 tty 3           DREGGS               1:07   Katy x1111
   4 tty 4                                      Console E3-D
   5 tty 5                                      Mkt. demo area
   6 tty 6
   7 tty 7           DREGGS                 14   Marie x1112
  10 tty 10
 .  .  .
 135 tty 135
 136 tty 136
 137 tty 137                                    rp4-printer
 140 tty 140                                    Braille printer
 141 aux 0
 142 vty 0   Denise    idle                     DENISE-MAC.CISCO.COM
 143 vty 1   Michael   idle                   0 DREGGS.CISCO.COM
 144 vty 2
 145 vty 3
 146 vty 4
 147 vty 5
```

The absolute line numbers are listed at the far left, followed by the line type and then the relative line number. Relative line numbers always begin numbering at zero and define the type of line. Addressing the second virtual terminal line as line vty 1, for example, is easier than remembering it as line 143.

The line types are ranked as follows in the line table:

1. Console 0 (con 0)
2. Standard asynchronous line (tty)
3. Auxiliary port (aux)
4. Virtual terminal line (vty)
5. Printer

You can address a single line or a consecutive range of lines with the **line** command. A line number is necessary, though, and you will receive an error message if you forget to include it.

Note Line numbers are in octal form on the modular communication server products (ASM-CS, for example), but are in decimal form on the 500-CS communication server.

Define Terminal Operation Characteristics

The communication server can support a large number of terminal types and configurations. You can configure the following terminal operation characteristics:

- Terminal type
- Terminal screen length and width
- Terminal session limits

- Special character sequences
- International character display
- Communication parameters: baud rate, parity, data bits, and stop bits
- Automatic baud rate detection
- Character padding

The following sections describe the tasks to implement these features.

Specify the Terminal Type

You can specify the type of terminal connected to a line. This feature has two benefits: it provides a record of the type of terminal attached to a line, and it can be used in Telnet and TN3270 terminal negotiations to inform the remote host of the terminal type for display management.

Task	Command
Specify the terminal type.	**terminal-type** *terminal-name*

Set the Terminal Screen Length and Width

By default, the communication server provides a screen display of 24 lines by 80 characters. You can reset these values if they do not meet the needs of your terminal.

Task	Command
Set the screen length.	**length** *screen-length*
Set the screen width.	**width** *characters*

The values set can be learned by some host systems that use this type of information in terminal negotiation. Set a value of zero for the screen length to disable pausing between screens of output.

Establish Terminal Session Limits

You might need to control terminal sessions in high traffic areas to provide resources for all users. You can define these limitations for terminal sessions:

- The maximum number of sessions
- The idle session timeout interval

Task	Command
Set the maximum number of sessions.	**session-limit** *session-number*
Set the idle session timeout interval.	**session-timeout** *minutes* [**output**]

Define Special Character Key Sequences

You can modify the default key sequences to execute functions such as system escape or terminal pause.

Task	Command
Change the system escape sequence.	**escape-character** *ASCII-number*
Define a session activation sequence or character.	**activation-character** *ASCII-number*
Define the session disconnect sequence or character.	**disconnect-character** *ASCII-number*
Define the local hold sequence or character that pauses output to the terminal screen.	**hold-character** *ASCII-number*

Specify the International Character Display

You can enable a full 8-bit international ASCII character set to allow special graphical and international characters for use in banners and prompts, and to add special characters such as software flow control. These settings can be configured globally, by interface, and locally at the user level. Use the following criteria for determining which configuration mode to use to set up this feature:

- If a large number of connected terminals support nondefault ASCII bit settings, use the global configuration commands.
- If only a few of the connected terminals support nondefault ASCII bit settings, use line configuration commands or the EXEC local terminal setting commands.

Note Setting the EXEC character width to eight bits can cause failures. If a user on a terminal that is sending parity enters the command **help**, an "unrecognized command" message appears because the system is reading all eight bits, although the eighth bit is not needed for the **help** command.

To specify the ASCII character set in different configuration modes, perform the following tasks:

Task	Command
Specify the ASCII character set used in EXEC and configuration command characters on a global basis.	**default-value exec-character-bits** {**8** | **7**}
Specify the ASCII character set used in special characters such as software flow control, hold, escape, and disconnect characters on a global basis.	**default-value special-character-bits** {**8** | **7**}
Specify the ASCII character set used in EXEC and configuration command characters on a per-line basis.	**exec-character-bits** {**8** | **7**}
Specify the ASCII character set used in special characters such as software flow control, hold, escape, and disconnect characters on per-line basis.	**special-character-bits** {**8** | **7**}

Task	Command
Locally set the ASCII character set used in EXEC and configuration command characters.	**terminal exec-character-bits** {**8** | **7**}
Locally set the ASCII character set used in special characters such as software flow control, hold, escape, and disconnect characters.	**terminal special-character-bits** {**8** | **7**}
Locally set the ASCII character set sent over network connections to hosts.	**terminal data-character-bits** {**8** | **7**}

Set the Device Communication Parameters

The communication server supplies the following default serial communications parameters for terminal and other serial device operation:

- 9600 bits per second (bps) line speed
- 8 data bits
- 2 stop bits
- No parity bit

You can change these parameters as necessary to meet the requirements of the terminal or host to which you are attached.

Task	Command
Set the line speed. Choose from line speed, transmit speed, or receive speed.	**speed** *bps* **txspeed** *bps* **rxspeed** *bps*
Set the data bits.	**databits** {**5** | **6** | **7** | **8**}
Set the stop bits.	**stopbits** {**1** | **1.5** | **2**}
Set the parity bit.	**parity** {**none** | **even** | **odd** | **space** | **mark**}

Configuring Automatic Baud Rate Detection

You can configure a terminal to automatically detect the baud rate being used over an asynchronous serial line.

Task	Command
Set the terminal to automatically detect the baud rate.	**autobaud** [**fast**]

To start communications using automatic baud detection, type multiple Returns at the terminal. A 600, 1800, or 19200 baud line requires three Returns to detect the baud rate. A line at any other baud rate requires only two Returns. If you type extra Returns after the baud rate is detected, the EXEC simply displays another system prompt.

Set Character Padding

You can change the character padding on a specific output character. Character padding adds a number of null bytes to the end of the string and can be used to make a string an expected length for conformity.

Task	Command
Pad a character with NULL bytes.	**padding** *ASCII-number count*

Record the Device Location

You may record the location of a serial devices. The text provided for the location appears in the output of the EXEC monitoring commands.

Task	Command
Record the location of a serial device.	**location** *text*

Configure Software Flow Control

You can set the following types of flow control between the communication server and devices attached to it:

- Software flow control, both in the incoming and outgoing direction
- Hardware flow control

Task	Command
Set the terminal flow control; select software or hardware flow control.	**flowcontrol** {**none** | **software** [**in** | **out**] | **hardware**}

Configure Software Flow Control Start and End Characters

You can define characters or character sequences that signal the start and end of data transmission when software flow control is in effect. This is useful for providing control of data over the serial line.

Task	Command
Set the flow control start character.	**start-character** *ASCII-number*
Set the flow control stop character.	**stop-character** *ASCII-number*

Define a Command String for Automatic Execution

You can set up a command or string of commands that will automatically execute upon connection to another host. Any appropriate EXEC command and any switch or host name that occurs with the EXEC command are legal.

Task	Command
Define a command or string of commands to be automatically executed.	**autocommand** *command*

Create Character and Packet Dispatch Sequences

The communication server supports configuration of dispatch sequences and TCP state machines that transmit packets of data upon receipt of the defined character or sequence of characters. You can set up dispatch characters that allow packets to be buffered, then transmitted upon receipt of a character. You can set up a state machine that allows packets to be buffered then transmitted upon receipt of a sequence of characters. This allows for packet transmission by pressing a function key, which is typically defined as a sequence of characters ("Esc I C," for example).

State machines allow control of TCP processes based upon a set of inputs. The current state of the device determines what will happen next given an expected input. The state-machine commands configure the server to search for and recognize a particular sequence of characters, then cycle through a set of states. The user defines these states and up to eight states can be defined. (Think of each state as a step the server takes based upon the assigned configuration commands, and the type of information received.)

The software supports user-specified state machines for determining whether data from an asynchronous port should be sent to the network. This is an extension of the concept of the dispatch character and allows, for example, the equivalent of multicharacter dispatch strings.

Up to eight states can be set up for the state machine. Data packets are buffered until the appropriate character or sequence triggers the transmission. Delay and timer metrics allow for more efficient use of system resources. Characters defined in the TCP state machine take precedence over those defined for a dispatch character.

Perform the following tasks, as needed, for your particular system needs.

Task	Command
Specify the transition criteria for the states in a TCP state machine.	**state-machine** *name state firstchar lastchar* [*nextstate* | **transmit**]
Specify the state machine for TCP packet dispatch.	**dispatch-machine** *name*
Define a character that triggers packet transmission.	**dispatch-character** *ASCII-number* [*ASCII-number2 . . . ASCII-number*]
Set the dispatch timer.	**dispatch-timeout** *milliseconds*

Enable the Auxiliary Port

Note This section applies to the ASM-CS only. There is no auxiliary port on the 500-CS.

A backup RS-232 DTE port is available on the ASM-CS. Use this port to attach to an RS-232 port of a CSU/DSU, protocol analyzer, or modem. The ports assert DTR only when a Telnet connection is established. This auxiliary port does not use RTS/CTS handshaking for flow control.

Task	Command
Enable the auxiliary serial RS-232 DTE port available on the ASM-CS.	**line aux** *line-number*

You cannot use the auxiliary port as a second console port, nor can you initiate connections from this port. Its purpose is to *receive* connections from remote systems. You must order a special cable from your technical support personnel for use with this auxiliary port.

Specify the Transport Protocol for a Specific Line

You can selectively specify the protocols allowed on individual lines. You can set the protocol for incoming and for outgoing connections, and change the default protocol for a line. For communication servers that support LAT, the default protocol is LAT. For those that do not support LAT, the default transport protocol is Telnet.

Perform one of the following tasks to specify the transport protocol. Which task you choose depends on whether you are configuring an incoming or outgoing connection.

Task	Command
Specify which protocols to use on an outgoing line.	**transport output** [**telnet** | **lat** | **rlogin** | **pad** | **none**]
Specify which protocols to use on an incoming line.	**transport input** [**telnet** | **lat** | **pad** | **none**]
Specify the protocol to use when one has not been specified.	**transport preferred** [**telnet** | **lat** | **rlogin** | **pad** | **none**]

Set a Line as Insecure

You can set up a line to appear as an insecure, dial-up line. The information is used by the LAT software, which reports such connections as dial-ups to remote systems.

Task	Command
Set the line as a dial-up line.	**insecure**

In the previous versions of Cisco software, any line that used modem control was reported as dial-up through the LAT protocol; this feature allows more direct control.

Save Local Settings Between Sessions

You can configure the communication server to save local parameters set with EXEC **terminal** commands between sessions. This ensures that the parameters the user sets will remain in effect between terminal sessions. This behavior is useful for servers in private offices. By default, user-set parameters are cleared when the session ends.

Task	Command
Save local settings between sessions.	**private**

Set Pending Output Notification

You can set up a line to inform a user who has multiple, concurrent Telnet connections when output is pending on a connection other than the current one.

Task	Command
Set up a line to notify a user of pending output.	**notify**

This task performs the same function on a line as does the local EXEC command **terminal notify**. See the *Communication Server Command Reference* for more information about this command.

Establish and Control the EXEC Process

By default, the communication server starts an EXEC process on all lines; however, you can control EXEC processes, as follows:

- Turn the EXEC on or off.

  A serial printer, for example, should not have an EXEC started.

- Set the idle terminal time-out interval.

  The EXEC command interpreter waits for a specified interval of time until the user starts input. If no input is detected, the EXEC resumes the current connection, or if no connections exist, it returns the terminal to the idle state and disconnects the incoming session.

Task	Command
Turn the EXEC on or off.	**exec**
Set the idle terminal timeout interval.	**exec-timeout** *minutes* [*seconds*]

Configure Modem Lines

Cisco Systems communication servers use six RS-232 signals for each port, an arrangement that allows eight connections to be handled by one 50-pin Telco, RJ-11, or RJ-45 connector. The communication server can support the most popular forms of modem control and hardware flow control as well as high-speed dial-up modems.

The RS-232 output signals are Transmit Data (TXDATA) and Data Terminal Ready (DTR). The input signals are Receive Data (RXDATA), Clear to Send (CTS), and RING. The sixth signal is ground. Depending on the type of modem control, these names may or may not correspond to the standard RS-232 signals, which have similar names.

Dial-up modems that operate over normal dial-up telephone lines at speeds of 9600 bits per second and higher are now available. These modems do not operate at a guaranteed throughput; instead, they operate at a speed dependent on the quality of the line, the effectiveness of data compression algorithms on the data being transmitted, and other variables. These modems use hardware flow control to stop the data from the host by toggling some RS-232 signal when they cannot take any more.

In addition to hardware flow control, dial-up modems require special software handling. For example, they must be configured to create an EXEC when a user dials in and to hang up when the user exits the EXEC. These modems must also be configured to close any existing network connections if the telephone line hangs up in the middle of a session.

Your communication server supports hardware flow control on its CTS input, which is also used by the normal modem handshake.

You can configure the following modem line characteristics and modem features on the communication server:

- Raise the DTR output signal for automatic dialing
- Close connections by raising the CTS signal
- Automatically answer a high-speed modem

- Support a dial-in modem
- Support reverse connections and prevent incoming calls
- Support both dial-in and dial-out modems
- Configure a line timeout interval
- Configure automatic line disconnect

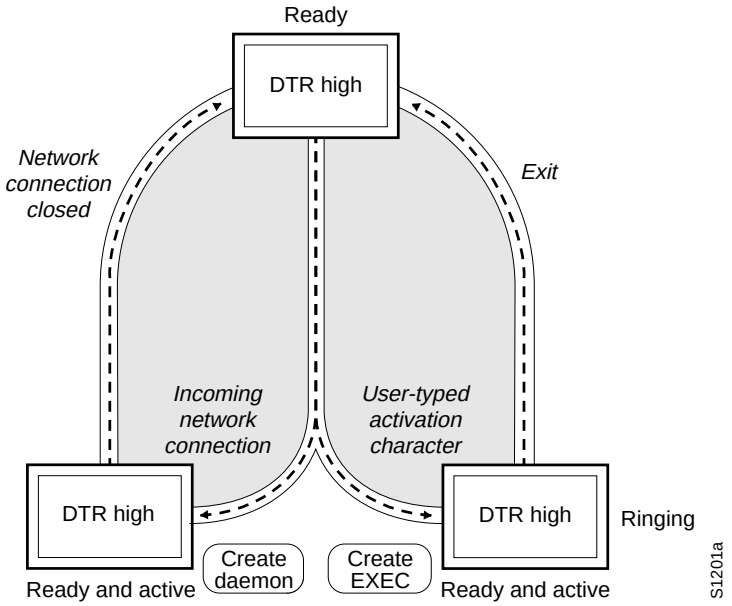
State diagrams accompany some of the tasks in the following sections to illustrate how the modem control works. The diagrams show two processes:

- The “create daemon” process creates a TTY daemon that handles the incoming network connection.
- The “create EXEC” process creates the process that interprets user commands. (Refer to Figures 1-1 through 1-6.)

In the diagrams, the current signal state and the signal the line is watching are listed inside each box. The state of the line (as displayed by the EXEC command **show line**) is listed next to the box. Events that change that state appear in italics along the event path, with actions that the software takes described within the ovals.

Figure 1-1 illustrates line behavior when no modem control is set. The DTR output is always high, and CTS and RING are completely ignored. The communication server creates an EXEC when the user types the activation character. Incoming TCP connections occur instantly if the line is not in use and can be closed only by the remote host.

Figure 1-1 EXEC and Daemon Creation on a Line with No Modem Control



Configure Automatic Dialing

With the dial-up capability, you can set a modem to automatically dial the phone number of a remote communication server. This feature offers cost savings because phone line connections are made as needed. You only pay for using the phone line when there is data to be received or sent. To configure a line for automatic dialing, perform the following task in line configuration mode:

Task	Command
Configure a line to initiate automatic dialing.	**modem dtr-active**

Close Modem Connections

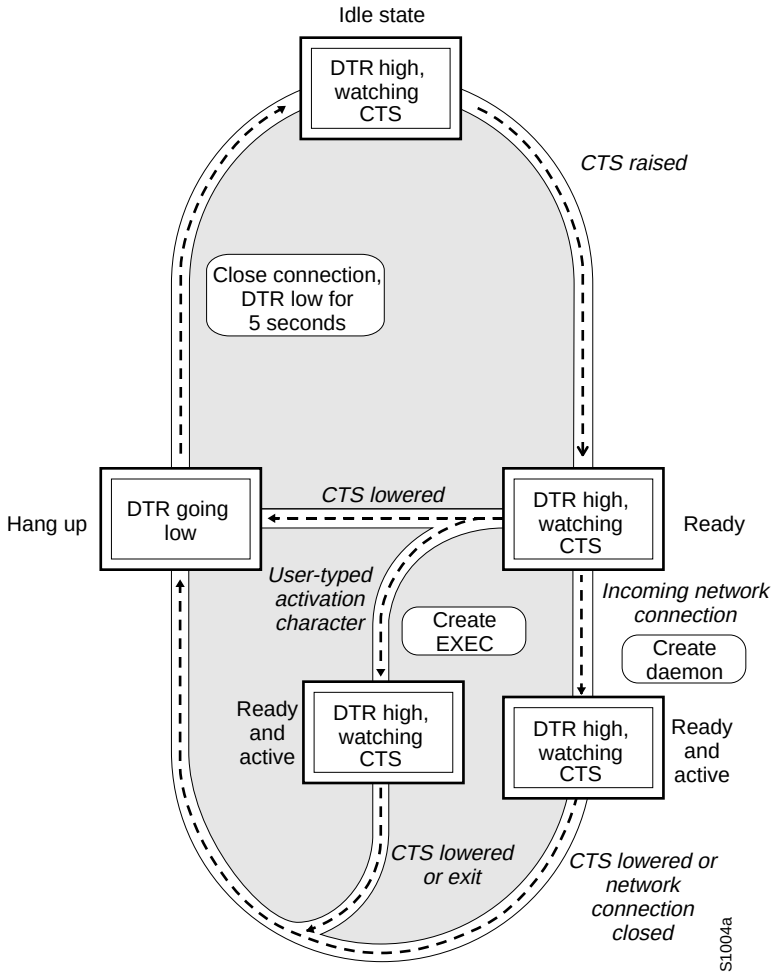
You can configure a line to close connections from a user's terminal when the terminal is turned off, and prevent inbound connections to devices that are out of service. To do so, perform the following task in line configuration mode:

Task	Command
Configure a line to close connections.	**modem cts-required**

Figure 1-2 illustrates the process of closing connections. This form of modem control requires that CTS be high throughout the use of the line. If CTS is not high, the user's typed input is ignored and incoming connections are refused (or step to the next line in a rotary group).

Note In order for a communication server to reliably detect a CTS signal change, the signal must remain in the new state for at least one full second.

Figure 1-2 EXEC and Daemon Creation on a Line Configured for Continuous CTS



Automatically Answer a Modem

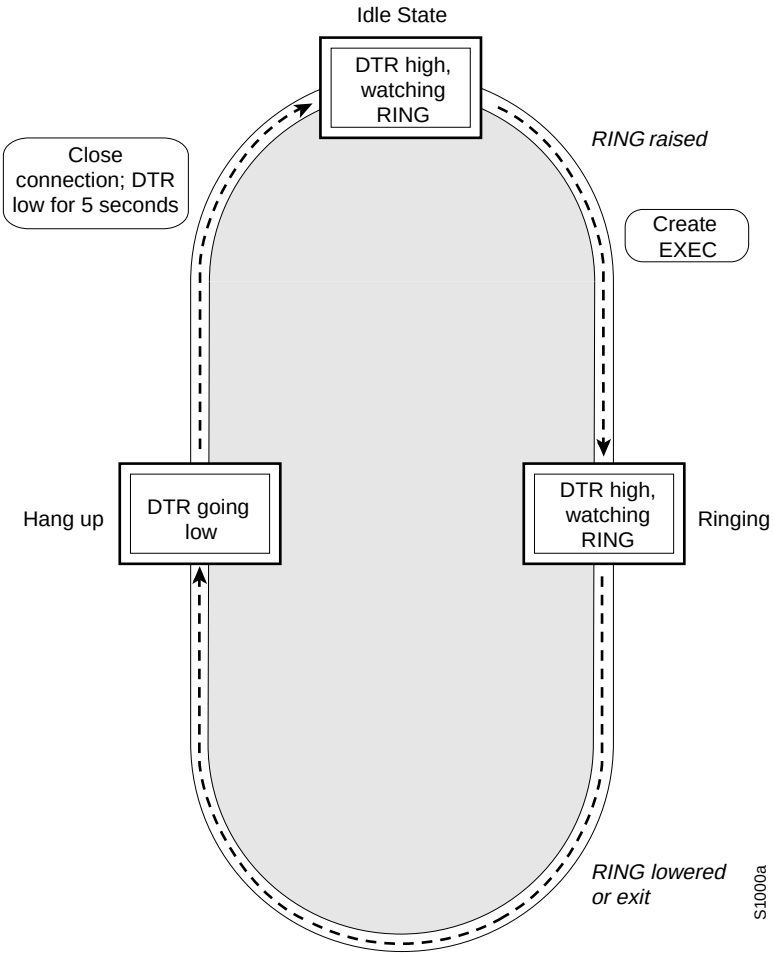
You can configure the modem to answer the telephone on its own as long as DTR is high, drop connections when DTR is low, and use its Carrier Detect (CD) signal to accurately reflect the presence of carrier. Wire the modem's CD signal (generally pin-8) to the communication server's RING input pin-22), and perform the following task in line configuration mode:

Task	Command
Configure a line to automatically answer a modem.	**modem ri-is-cd**

You can turn on the modem's hardware flow control independently to act on the status of the communication server's CTS input. Wire CTS to whatever signal the modem uses for hardware flow control. If the modem expects to control hardware flow in both directions, you might also need to wire the modem's flow control input to some other signal that the communication server always has high (such as DTR).

Figure 1-3 illustrates the process of automatically answering a modem. When the communication server detects a signal on the RING input of an idle line, it starts an EXEC or autobaud process on that line. If the RING signal disappears on an active line, the communication server closes any open network connections and terminates the EXEC. If the user exits the EXEC or the communication server terminates it because of no user input, the line hangs up the modem by lowering the DTR signal for five seconds. After five seconds, the modem is ready to accept another call.

Figure 1-3 EXEC Creation on a Line Configured for a High-Speed Dial-up Modem



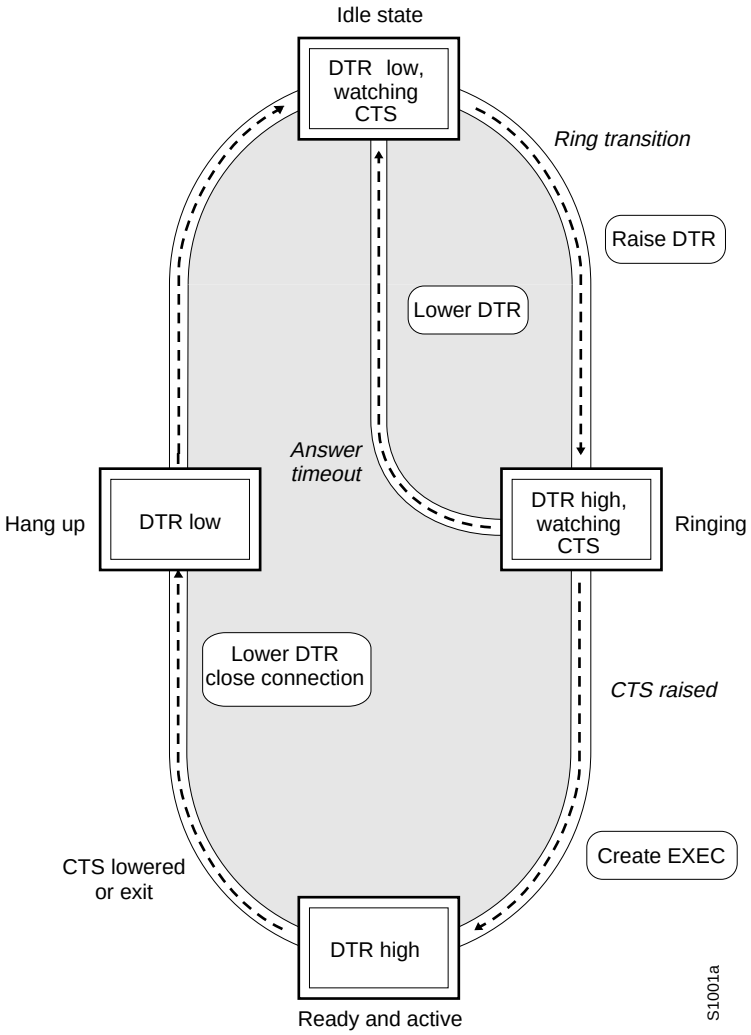
Support a Dial-In Modem

The communication server supports dial-in modems that use DTR to control the off-hook status of the telephone line. Perform the following task in line configuration mode to configure the line to support this feature:

Task	Command
Configure a line for a dial-in modem.	**modem callin**

Figure 1-4 illustrates the dial-in modem process. When a modem dialing line is idle, it has DTR in a low state and waits for a transition to occur on the RING input. This transition causes the line to raise DTR and start watching the CTS signal from the modem. After the modem raises CTS, the communication server creates an EXEC on the line. If the timeout interval (set with the **modem answer-timeout** line configuration command) passes before the modem raises CTS, the line lowers DTR and returns to the idle state.

Figure 1-4 EXEC Creation on a Line Configured for Modem Callin



Note The **modem callin** and **modem cts-required** line configuration commands are useful for SLIP operation. These commands ensure that when the line is hung up or CTS drops, the line reverts from SLIP mode to normal interactive mode. These commands do not work if you use the **async dedicated** interface configuration command to put the line in network mode permanently.

Although you can use the **modem callin** line configuration command with newer modems, the **modem ri-is-cd** line configuration command described earlier in this section is more appropriate. The **modem ri-is-cd** command frees up CTS for hardware flow control. Modern modems do not require the assertion of DTR to take a phone line off-hook.

Support Reverse Modem Connections and Prevent Incoming Calls

In addition to initiating connections, the communication server can receive incoming connections. This capability allows you to attach serial and parallel printers, modems, and other shared peripherals to the communication server and drive them remotely from other systems. The communication server supports reverse TCP, XRemote, and LAT connections.

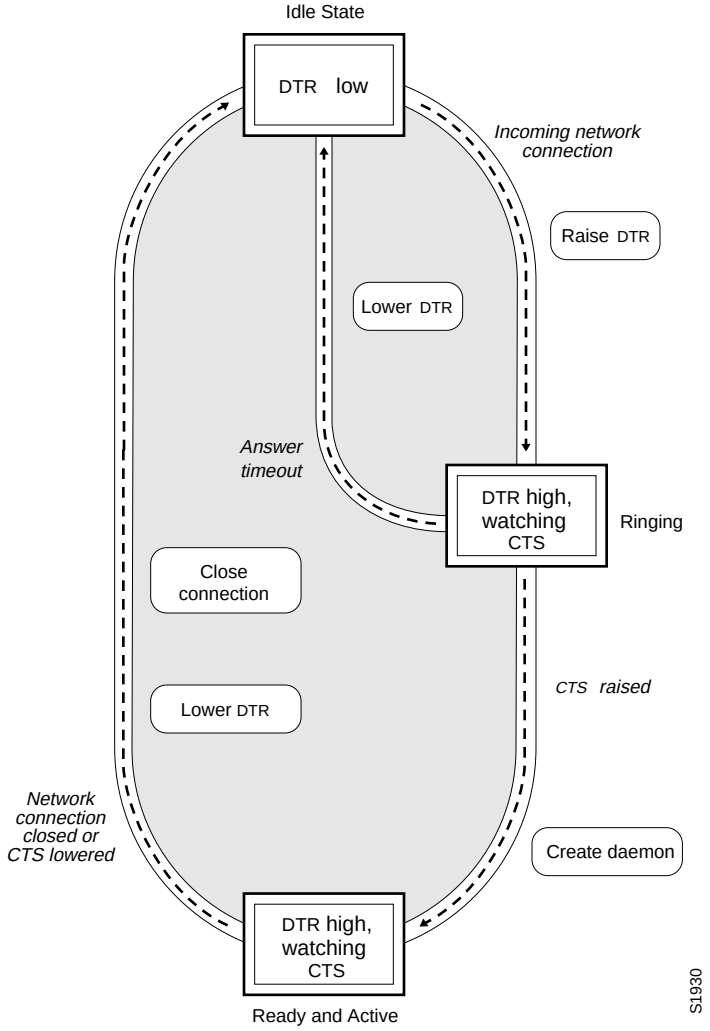
The specific TCP port or socket to which you attach the device determines the type of service the communication server provides on that line. When you attach the serial lines of a computer system or a data terminal switch to the serial lines of the communication server, the communication server acts as a network front-end for a host that does not support the TCP/IP protocols. This arrangement is sometimes called *front-ending*, or *reverse connection mode*.

The communication server supports ports connected to computers that are to be connected to modems. You can configure the communication server to behave somewhat like a modem by performing the following task in line configuration mode. This task also prevents incoming calls.

Task	Command
Configure a line for reverse connections and prevent incoming calls.	**modem callout**

Figure 1-5 illustrates the reverse connection process. When the communication server receives an incoming connection, it raises DTR and waits to see if the CTS becomes high as an indication that the host has noticed its signal. If the host does not respond within the interval set with the **modem answer-timeout** line configuration command, the communication server lowers DTR and drops the connection.

Figure 1-5 Daemon Creation on a Line Configured for Modem Callout



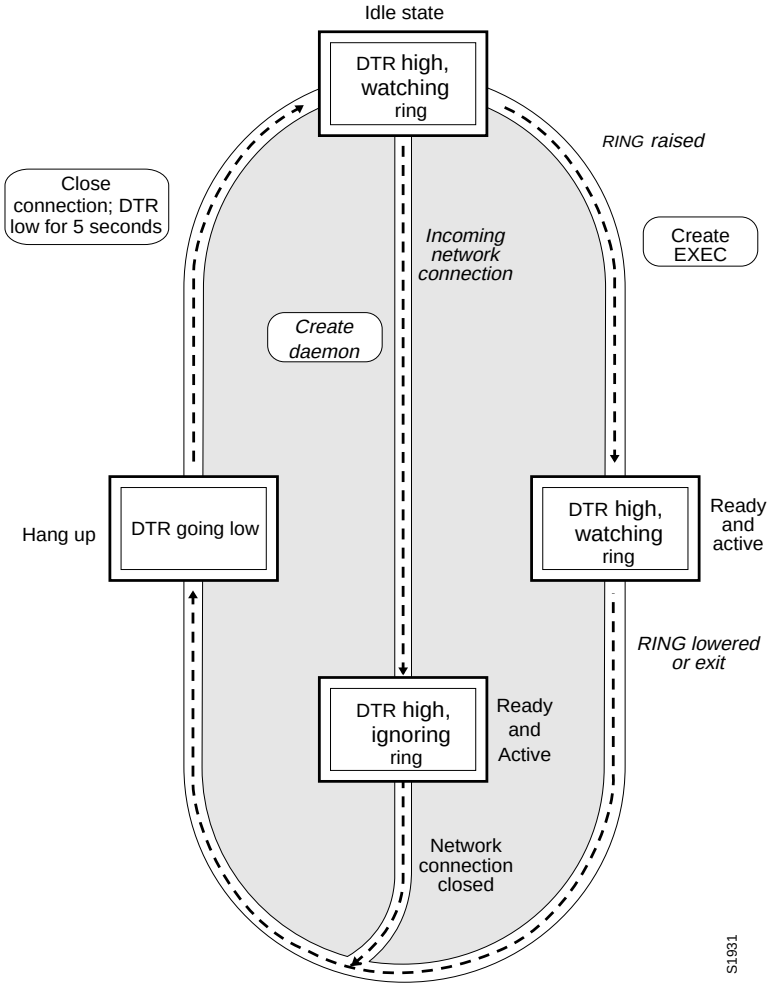
Support Dial-In and Dial-Out Modems

You can configure a line for both incoming and outgoing calls by perform the following task in line configuration mode:

Task	Command
Configure a line for both incoming and outgoing calls.	**modem inout**

Figure 1-6 illustrates the process of incoming and outgoing calls. If the line is activated by raising RING, it behaves exactly as a line configured with the **modem ri-is-cd** line configuration command described earlier. If the line is activated by an incoming TCP connection, the line behaves similarly to a nonmodem line.

Figure 1-6 EXEC and Daemon Creation on a Line Configured for Incoming and Outgoing Calls



Note If your system incorporates dial-out modems, consider using access lists to prevent unauthorized use.

Configure a Line Timeout Interval

You can change the interval that the communication server waits for CTS after raising DTR in response to RING from the default of 15 seconds. To do so, perform the following task in line configuration mode. The timeout applies to the modem callin command only.

Task	Command
Configure modem line timing.	**modem answer-timeout** *seconds*

Configure Automatic Line Disconnect

You can configure automatic line disconnect by performing the following task in line configuration mode. This feature is useful for UNIX UUCP applications that require this behavior, because UUCP scripts cannot issue the command that hangs up the telephone.

Task	Command
Configure automatic line disconnect.	**autohangup**

High-Speed Modem Support

Dial-up modems that operate over normal dial-up telephone lines at speeds of 9600 baud and higher are now available. These modems do not operate at a guaranteed throughput; instead, they operate at a speed dependent on the quality of the line, the effectiveness of data compression algorithms on the data being transmitted, and other variables. These modems use hardware flow control to stop the data from the host by toggling some RS-232 signal when they cannot take any more.

In addition to hardware flow control, dial-up modems require special software handling. For example, they must be configured to create an EXEC when a user dials in and to hang up when the user exits the EXEC. These modems must also be configured to close any existing network connections if the telephone line hangs up in the middle of a session.

Your communication server supports hardware flow control on its CTS input, which is also used by the normal modem handshake.

To support both modem control and hardware flow control on the same line, use the **modem ri-is-cd** subcommand described in this chapter.

The following commands might also be useful as you configure and use a high-speed modem:

- The **flowcontrol hardware** line subcommand enables outgoing hardware flow control based on the CTS input.
- The privileged EXEC command **debug modem** displays informational messages about modem control events, such as signal transitions and autobaud progress, on the console terminal.
- The EXEC command **show line** displays the status of a line. In the detailed command output, a Status line with “Idle” identifies inactive modem ri-is-cd lines and all other modem lines; a Status line with “Ready” identifies lines in use.
- The privileged EXEC command **clear line** closes all the connections on a line and hangs up the modem.

Connections to an Individual Line

Connections to an individual line are most useful when a dial-out modem, parallel printer, or serial printer is attached to that communication server line. To connect to an individual line, the remote host or terminal must specify a particular TCP port on the communication server. If Telnet protocols are required, that port is 2000 (decimal) plus the decimal value of the line number. If reverse XRemote is required, that port is 9000 (decimal) plus the decimal value of the line number.

Note Line numbers are octal on the ASM-CS communication servers and decimal on the 500-CS. On the ASM-CS, for example, line 40 corresponds to decimal 32; the corresponding TCP port is 2032. On the 500-CS, the corresponding TCP port for line 9 would be 2009. There is no octal-to-decimal translation. The **service decimal-tty** global configuration command controls the specification of octal or decimal line numbers; see the "System Management Command," chapter for more information about this command.

If a raw TCP stream is required, the port is 4000 (decimal) plus the decimal line number. The raw TCP stream is usually the required mode for sending data to a printer.

The Telnet protocol requires that carriage return characters be translated into carriage returns and line-feed character pairs. You can turn this translation off by specifying the Telnet binary mode option. To specify this option, connect to port 6000 (decimal) plus the decimal line number.

Assume that a laser printer is attached to line 10 of an ASM-CS communication server. Such a printer usually uses XON/XOFF software flow control. Because the communication server will not receive an incoming connection if the line already has a process, you must ensure that an EXEC is not accidentally started. You must therefore configure it as follows:

```
line 10
flowcontrol software
no exec
```

A host that wants to send data to the printer would connect to the communication server on TCP port 4008, send the data, and then close the connection. (Remember that line number 10 octal equals 8 decimal.)

Configure Rotary Groups

Connections can be made to the next free line in a group of lines, also called a rotary, or hunt, group. A line can be in only one rotary group; a rotary group can consist of a single line or several contiguous lines.

Task	Command
Add a line to the specified rotary group.	**rotary** *group*

Note The console line (line 0) might not be in a rotary group.

Enable Password Checking at Login

You can enable password checking on a particular line so that the user is prompted to enter a password at the system login screen. You must then also specify a password by performing the following tasks in line configuration mode:

Task		Command
Step 1	Enable password checking on a per-line basis using the password specified with the **password** command.	**login**
Step 2	Assign a password to a particular line.	**password** *password*

You can enable password checking on a per-user basis, in which case authentication is based on the username specified with the **username** global configuration command, as described in the chapter, "Managing the System." To enable this type of password checking, perform one of the following tasks in line configuration mode:

Task	Command
Enable password checking on a per-user basis using the username and password specified with the **username** global configuration command.	**login local**
Select the TACACS-style user ID and password-checking mechanism	**login tacacs**

By default, virtual terminals require passwords. If you do not set a password for a virtual terminal, it will respond to attempted connections by displaying an error message and closing the connection. Use the **no login** command to disable this behavior and allow connections without a password.

For examples of password checking, see the configuration examples at the end of this chapter.

For other access control tasks and password restrictions, including the **enable password** global configuration command that restricts access to privileged mode, see the chapter, "Managing the System."

Display Terminal Banner Messages

You can provide the following types of messages to users of terminals connected to the communication server:

- A message-of-the-day banner
- A line activation message
- An incoming message banner
- Idle terminal message
- "Line in use" message
- "Host failed" message

The following sections explain how to display these messages and how to suppress display of message-of-the-day and line activation banners.

Display a Message-of-the-Day (MOTD) Banner

You can display a message of the day to all connected terminals. This message is displayed at login and is useful for sending messages, such as impending system messages, that affect all network users, such as impending system shutdowns. Perform the following task in line configuration mode:

Task	Command
Display a message of the day banner.	**banner motd** *d message d*

Display a Line Activation Message

You can display a line activation message when an EXEC process such as line activation or incoming connection to a virtual terminal is created Perform the following task in line configuration mode:

Task	Command
Display a message on terminals with an interactive EXEC.	**banner exec** *d message d*

Display an Incoming Message Banner

You can display an incoming message on terminals connected to reverse Telnet lines. This message is useful for displaying instructions to users of these types of connections.]

Task	Command
Display messages on terminals connected to reverse Telnet lines.	**banner incoming** *d message d*

Display an Idle Terminal Message

You can display messages on a console or terminal that is not in use. Also called a *vacant message*, this message is different from the banner message displayed when an EXEC process is activated. Perform the following task in line configuration mode:

Task	Command
Display an idle terminal message.	**vacant-message** [*d message d*]

Display a “Line in Use” Message

You can display a “line in use” message when an incoming connection is attempted and all rotary group or other lines are in use. Perform the following task in line configuration mode:

Task	Command
Display a “line-in-use” message.	**refuse-message** *d message d*

If you do not define such a message, the user will receive a system-generated error message when all lines are in use. You can also use this message to provide the user with further instructions.

Display a “Host Failed” Message

You can display a “host failed” message when a Telnet connection with a specific host fails. Perform the following task in line configuration mode:

Task	Command
Display a “host failed” message.	**busy-message** *hostname d message d*

Control Display of Messages

You can control display of the message-of-the-day and line activation banners. Perform the following task in line configuration mode:

Task	Command
Suppress banner display.	**no exec-banner**
Re-instate the display of the EXEC or MOTD banners.	**exec-banner**

Set a Terminal-Locking Mechanism

You can enable a terminal-locking mechanism that allows a terminal to be temporarily locked by performing the following task. Perform the following task in global configuration mode:

Task	Command
Enable a temporary terminal locking mechanism.	**lockable**

When the terminal-locking mechanism is set and you enter the EXEC **lock** command, you are prompted for a password. This password must be supplied before the terminal can be used. This allows a terminal to be left unattended without concern about unauthorized access.

Line Configuration Examples

This section provides examples of line configuration commands to illustrate how you can set up the configuration file for your communication server. Examples include the following:

- Typical line confgiguration
- Auxiliary port configuration
- Printer connection example
- State machine example

Typical Line Configuration

The following example illustrates line configuration commands that can be found in a typical communication server configuration file:

```
!
! Lines 1 through 7 are in a public terminal area. They are restricted
! to making connections to hosts on the local network. They also run at
! 4800 baud and have padding on the Return character.
line 1 7
location Public Terminal Pool
access-class 1 out
speed 4800
pad 13 25
!
! Terminal lines 10 through 16 are dial-in modems that
! can run from 300 to 19,200 baud.
line 10 16
modem callin
autobaud
!
```

```
! Lines 17 through 32 are hooked up in reverse connection mode to
! a large terminal switch. By Telnetting to TCP port 3001 on this
! host, a user gets the next free line in the rotary group.
line 17 32
rotary 1
modem callout
!
! Line 33 has a laser printer attached. A network printer daemon
! connects to TCP port 4027 on chaff to send data to the printer.
! Software flow control is used and the line speed is set to 38,400 baud.
! Use the command no exec for unused terminals; setting the speed to
! a very low number (that is, 110) may also prevent loading under
! certain conditions:
line 33
location Laser Printer/pubs
no exec
flowcontrol software in
flowcontrol software out
stopbits 1
txspeed 38400
rxspeed 38400
!
! Lines 34 through 40 are unused at present. Their line
! speeds are set to zero.
line 34 40
location Unused Terminals
speed 0
!
! There are 5 virtual terminals. The vty keyword makes their exact
! line numbers unnecessary. Only hosts specified by
! access list 1 may connect to the virtual terminals.
line vty 0 4
access-class 1 in
!
! The end keyword terminates a command file.
end
```

Auxiliary Port Configuration Example

The following example configures the auxiliary port with a line speed of 2400 baud and enables the EXEC:

```
line aux 0
exec
speed 2400
```

No modem control signals are supported on this line. If an auto-answer modem is configured on the line, you must dial up, log in, then hang up. The DTR signal will be active whenever an EXEC is configured on the auxiliary port.

Note The EXEC is still present and may be used by the next person that dials into the number. This could cause security problems.

You can monitor that port remotely by connecting to the TCP port whose number is 2000 decimal plus the line number of the auxiliary port. For example, if the auxiliary port was line 1 (obtained from the display of the EXEC command **show users all**), then the TCP port would be 2001.

This example applies the the ASM-CS only. There is no auxiliary port on the 500-CS.

Printer Connection Example

The following example illustrates how to configure a line to support a printer:

```
line 10
flowcontrol software
no exec
```

Assume that a laser printer is attached to line 10 of an ASM-CS communication server. Such a printer usually uses XON/XOFF software flow control. Because the communication server will not receive an incoming connection if the line already has a process, you must ensure that an EXEC is not accidentally started.

A host that wants to send data to the printer would connect to the communication server on TCP port 4008, send the data, and then close the connection. (Remember that line number 10 octal equals 8 decimal.)

State Machine Examples

Following are two examples that show different ways to set up state machines:

The first example illustrates a state machine on the same line as an asynchronously-based packet format. A packet is terminated by a sequence DLE, ETX, CHKSUM, (16, 3, some number) DLE can occur in the packet itself if escaped by another DLE. (In particular, this means that the sequence “DLE, DLE, ETX, x” should not terminate the packet.)

Use a long dispatch time-out period to ensure that large packets can be generated, and to prevent the line from getting permanently stuck in the event of lost data.

```
!
!If we see the first DLE, go to state 1 (from state 0)
!
state-machine packet 0 16 16 1
!
! If we see ETX after the DLE, go to state 2, otherwise (including
! another DLE) return to state 0
!
state-machine packet 1 3 3 2
!
! In state 2, receipt of the checksum causes the packet to be sent
!
state-machine packet 2 0 255 transmit
!
! Add this state machine to the appropriate lines
!
line 1 20
dispatch-machine packet
!
```

This next example ensures that the characters from the function keys on an ANSI terminal are all lumped together into a single packet. Because of this, systems that attempt to distinguish between function keys and the same bytes typed individually do not become confused by variable network delays. An ANSI function key usually generates “Esc [*random* upper-case-alpha”.

```
!
! Recall that the default is to remain in state 0 without
! transmitting anything. We want normal type-in to be transmitted
! immediately
!
state-machine function 0 0 255 transmit
!
! Except for "escape," which starts waiting for the rest of a
! function key.  However, if the user types "escape" we want it
! to be transmitted pretty soon.  This is what the "delay"
! keyword does.
state-machine function 0 27 27 1 delay
!
! Again, "esc foo" should transmit immediately, unless foo is "["
!
state-machine function 1 0 255 transmit
state-machine function 1 91 91 2 delay
!
! Finally, we want to collect perhaps many characters in state 2,
! until we run into an upper case alphabetic character, or the
! line stays idle for a while.
!
state-machine function 2 0 255 2 delay
state-machine function 2 65 90 transmit
!
```