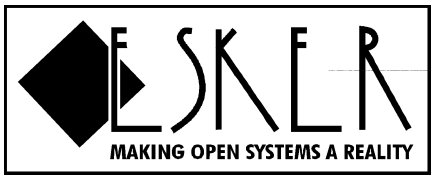
USER MANUAL

Tun EMUL

Terminal Emulation for DOS

Version 8.00



DEMULUS8.DOC

In continuing efforts to improve our products, keep up with current technology, and incorporate user's suggestions, ESKER may change the look and the characteristics of the product described in this manual at any time.

ESKER makes no warranty of any kind, expressed or implied, with regard to the contents of this manual. This is a purely technical document and does not represent a contractual obligation of any kind. ESKER shall not be held liable for incidental or consequential damages in connection with the use of the programs described herein.

PREFACE

Tun EMUL for MS-DOS is an advanced terminal emulator for the MS-DOS environment. Unlike the Windows version only the asynchronous emulations are supported.

Tun EMUL for MS-DOS is one of a range of complementary packages including Tun KERNEL and Tun TCP. (see table below).

	WINDOWS	**MS-DOS**
Tun KERNEL	TCP/IP protocol stacks for Windows	TCP/IP protocol stacks for MS-DOS (TSR)
Tun TCP	TCP/IP applications for Windows (NFS, Printer sharing, FTP, TELNET, TAR ...)	TCP/IP applications for MS-DOS (NFS, Printer sharing, FTP, TELNET, TAR ...)
Tun EMUL	Comprehensive terminal emulator for Windows (asynchronous emulation, IBM3270, IBM5250)	Comprehensive terminal emulator for MS-DOS (asynchronous emulation)
Tun MAIL	Comprehensive TCP/IP E-Mail	N/A
Tun SQL	ODBC drivers for the Client-Server mode over TCP/IP (RDBMS Oracle, Informix, Sybase)	N/A

Tun EMUL for MS-DOS is delivered as standard with the package Tun PLUS which incorporates all the above software.

Tun EMUL for MS-DOS can be installed independently from Tun PLUS. However the Tun PLUS installation procedure will handle this automatically.

When purchased by itself, Tun EMUL for MS-DOS is delivered with ESKER's TCP/IP protocol stack, Tun KERNEL. This complementary package provides complete TCP/IP transport services for MS-DOS.

TABLE OF CONTENTS

CHAPTER 1 - INSTALLATION

PACKAGE CONTENTS

Please make sure that your Tun EMUL package contains the following:

- Tun EMUL User Manual
- Tun EMUL program floppy disks in 1.2 Mb 5.25" and 1.44 Mb 3.5" formats
- User License
- A sealed envelope containing a serial number, activation key, and a key disk (if applicable)
- Miscellaneous technical bulletins (if applicable)

NOTE: Opening the sealed envelope indicates that you have accepted the terms and conditions of the User License (indicated on the sealed envelope) for using the Tun EMUL product.

HARDWARE CONFIGURATION

To use Tun EMUL, you will need the following equipment:

- a 100% PC XT or AT compatible computer, with an 84, 101, or 102-key keyboard
- MS-DOS 3.30 and higher (or Microsoft Windows 3.x)
- 640 Kb of RAM (4 Mb recommended on Windows PCs)
- a monochrome, EGA/CGA, or VGA video adapter
- a hard disk (with approximately 2-3 Mb free space)

Depending on the type of connection:

- an Ethernet or Token Ring network interface card for TCP/IP connections
- up to four serial ports (COM1, COM2, COM3, COM4) for RS232 asynchronous connections.
- An internal or external modem for establishing remote connections.

INSTALLATION

If you have purchased Tun EMUL as part of the Tun PLUS package it is not necessary to follow the instructions below as the Tun PLUS installation procedure will handle the necessary installation.

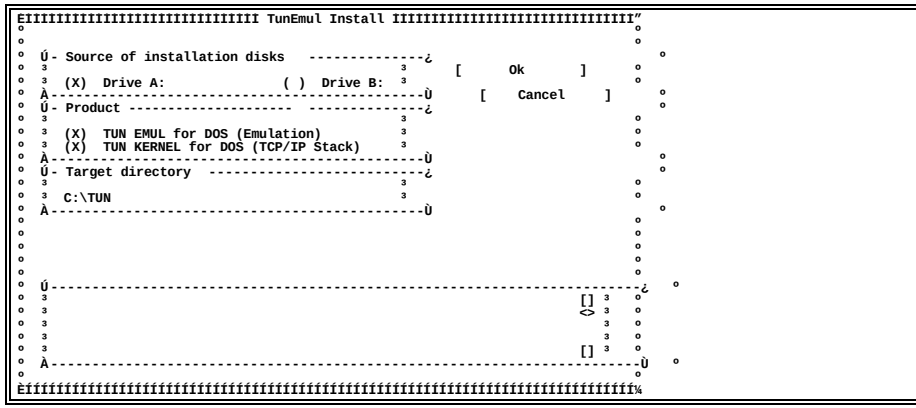
For configuration in a TCP/IP network, please install Tun KERNEL for MS-DOS before proceeding with the installation of Tun EMUL.

To install Tun EMUL on a PC's hard disk, follow this procedure:

1. Load the mouse driver if you would like to be able to navigate within the installation menus with a mouse.
2. To install the Tun EMUL programs and files, insert the floppy disk **Terminal Emulation for DOS 1/2** into a drive and type:

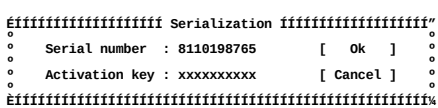
   ```
   A:\> INSTALL [language]
   ```

   [language] may be used to set the default national keyboard.
3. Select the appropriate parameters using the mouse, arrow keys, or the <Tab> key. In each section, use the space bar to select an option.



- Choose the program floppy drive if necessary,
- Change the destination directory if necessary,
- If you do not want to install the TCP/IP Stack (Tun KERNEL for MS-DOS) or if it is already installed, unmark the corresponding check box.
- Press the <Enter> key or click on <OK>.

4. Next, enter the product's serial number and activation key:



Enter the serial number and activation key found in the sealed envelope of your Tun EMUL package, then click on <OK> or press <Enter>.

5. The installation program will prompt you to insert the program floppy disks. Click on <OK> or press <Enter> after inserting the requested disk. During installation, a window will display the names and directories of the files as they are extracted and installed.

6. The installation is now complete.

DIRECTORY STRUCTURE AND INSTALLED FILES

The following files are installed into the default directory:

Asynchronous emulator:

File	Description
TUNEMUL.EXE	Configuration module for MS-DOS
DTALOGO.DAT	Tun EMUL screen logo
EMUL.EXE	Asynchronous emulation module for MS-DOS
*.DF	Character font for MS-DOS emulator
*.CFG	Session configuration files
*.TER	Terminal definition files
*.NAT	National keyboard definitions
*.KEY	Emulation keyboard definitions
*.FUN	Function-key definitions
*.SEQ	Escape sequence definitions
*.COD	Control code definitions
*.SND	Special character conversions
*.TAB	Character table definitions
*.CTX	Display settings files
ACTION.*	Descriptions of the actions supported by the emulators
*.PAN	Function-key panel definitions
*.WLG, *.LG	Message files
*.MAC	Sample macros
TABLE.EXE	Utility for viewing character tables
RTUNPLUS and RTUNPLUS.C	UNIX executable (SCO, IX386) and C source-code for file transfer module
NVT.EXE	IPX interface
LM2TUN.EXE	Microsoft LAN-Manager TCP/IP interface
LANW2TUN.EXE	Novell LAN Workgroups/Workplace interface
TRANSFILE.SH	UNIX file transfer shell script
MOUSE.C	Example of mouse programming (in C)

INSTALLING FILE TRANSFER MODULE ON UNIX

In addition to X, Y, and Z modem, file transfer between a PC in emulation and a UNIX server is made possible by **rtunplus**, an executable program installed on the UNIX machine. Tun EMUL supplies two versions of this program:

- a binary executable, directly usable on SCO XENIX, SCO UNIX, or AT&T 386 compatible.
- a file in source code that must be compiled to run on other UNIX systems.

The RTUNPLUS binary executable

This section describes the procedure for three UNIX systems in particular: **SCO-XENIX 386**, **SCO-UNIX** and **IX/386**. Other UNIX systems that run on 386 microcomputers may also use this procedure.

The executable file **rtunplus** must be installed in a public directory on the host machine.

1. Place the Tun EMUL **Prog 1** in the floppy drive of the UNIX system
2. Login as "root"
3. Execute the following commands:

On SCO-XENIX 386 and SCO-UNIX:

```
# doscp -r a:/rtunplus /usr/bin/rtunplus
# chmod 4755 /usr/bin/rtunplus
```

On IX/386:

```
# dosget -b a:/rtunplus /usr/bin/rtunplus
# chmod 4755 /usr/bin/rtunplus
```

After this operation, **rtunplus** is operational.

Using the source code RTUNPLUS.C

> **NOTE**: Emulation must be operational in order to perform the following installation (see chapter Using Tun EMUL).

For installation on UNIX hosts other than the three mentioned above, you must compile the source code. This module was conceived to be compiled on the vast majority of UNIX systems.

First, the source code needs to be transferred to the target machine. For this, you may use the **Send File** feature in emulation as described in the following procedure:

1. Use Tun EMUL to log in to the host system as the user "root" (see Chapter Using Tun EMUL in MS-DOS for instructions on configuring the emulator).

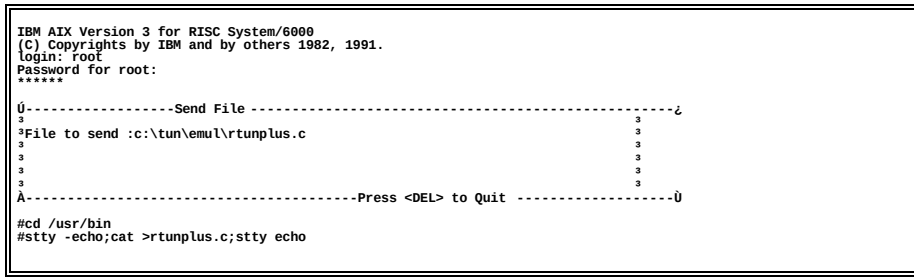
2. Execute the following commands:

```
# cd /usr/bin
# stty -echo;cat >rtunplus.c;stty echo
```

   No response will be shown on the screen after these commands.

3. Simultaneously press the <Alt> and <F7> keys (to send a file). A window will appear asking for the name of the file you wish to send. In this case, type**:**

   `c:\tun\emul\rtunplus.c`



4. After the file has been sent, press any key to continue. The window will disappear immediately.

5. To close the capture of the source file on the UNIX side, press <Ctrl><D> on the keyboard. The # should reappear.

Compiling rtunplus.c

After the source file has been transferred, all you have to do is compile it. If your UNIX system is **UNIX SYSTEM V non XOPEN, SCO XENIX, UNIX SYSTEM III**, set the environment variable CFLAGS as follows:

SYSTEM V non XOPEN:

```
# CFLAGS=-DSYSV ; export CFLAGS
```

SCO-XENIX:

```
# CFLAGS=-DXENIX; LDFLAGS=-lx; export CFLAGS LDFLAGS
```

SYSTEM III:

```
# CFLAGS="-DSYSIII" ; export CFLAGS
```

For other UNIX systems, do not set CFLAGS.

Then, for all systems, execute the following commands:

```
# make rtunplus
# chmod u+s rtunplus
# rm rtunplus.c
```

When this is done, the executable **rtunplus** file will be installed and usable in the public directory **/usr/bin**. You may repeat this installation procedure on as many servers as you like.

CHAPTER 2 - INTRODUCTION

Tun EMUL is a communications package that allows you to integrate microcomputers into multi-user computing environments.

Tun EMUL may be configured to run over any of the following types of connections:

- RS232, asynchronous (serial)
- Local area networks running TCP/IP
- Dial-up networks using modems
- Interrupt 14h
- Interrupt 6Bh
- NVT over IPX (Novell Netware)

Some of Tun EMUL's main features include:

- Entirely configurable terminal emulation
- Complete Microsoft Windows integration
- Versatile and easy-to-use file transfer
- Powerful Macro language for process automation

THE CONCEPT OF EMULATION

Terminal Emulation refers to the process of making a PC behave as a terminal.

Used for terminal emulation, Tun EMUL controls the screen display of characters sent by the server, and the correct emission of characters typed by the user. Of course, this is a simple definition of the concept of emulation.

In addition to these basic functions, the Tun EMUL emulator treats escape sequences and other character strings sent by a server to perform actions such as switching to reverse video, moving the cursor, or changing the screen colors. Characters and character strings are sent to the server by pressing keys on the PC in emulation.

One of the hardest problems in the domain of terminal emulation is that there are over fifty different types of "standard" terminal available on the market (ANSI, IBM 3151, vt220 & 320, WYSE 60, etc.). The choice of a terminal emulator must therefore be made carefully in order to guarantee compatibility with the software that will be used.

Tun EMUL EMULATIONS ARE COMPLETELY CUSTOMIZABLE

PC microcomputers are far more advanced than ordinary terminals because they are programmable. One of the strongest points about Tun EMUL is its ability to use the PC's processing power to customize emulations.

Because terminal emulation is a complicated process to define, Tun EMUL delivers many of the most common terminals on the market, while providing users the possibility of using menus to customize or define their own.

Tun EMUL SUPPORTS MANY COMMUNICATIONS TYPES

The Tun EMUL emulation and file transfer package can run using the following types of connections:

- RS-232, asynchronous (serial)
- Local Area Networks running TCP/IP (native or WinSock DLL)
- Dial-up networks using modems
- Interrupt 14h
- Interrupt 6Bh
- NVT over IPX (Novell Netware)

For asynchronous connections under MS-DOS, Tun EMUL can use COM1, COM2, COM3, and COM4 simultaneously. In order to improve performance, these ports are accessed directly, without passing through the BIOS.

Under MS-DOS, Tun EMUL interfaces with ESKER TCP/IP, Novell's LAN Workplace, FTP Software's PC/TCP and Microsoft MS TCP/IP.

Under Microsoft Windows, Tun EMUL can use any TCP/IP kernel that provides a compatible (standard) WinSock.DLL.

Tun EMUL manages Hayes-compatible modems for telephone connections. By default, Tun EMUL manages HAYES-compatible modems. If your modem needs more than standard HAYES commands, you may enter your own modem control codes.

Interrupt 14h may be used in order to run Tun EMUL over numerous communications supports (i.e. X.25 PADs, ISDN, etc.)

BIOS Interrupt 6Bh provides a similar service to Interrupt 14h, but runs in packet mode rather than character mode.

For use with IPX/SPX from Novell, Tun EMUL interfaces with the NVT protocol to access UNIX hosts running IPX/SPX.

Tun EMUL BEHAVES AS A MULTI-SESSION TERMINAL

Tun EMUL is capable of using any combination of the previous communication supports in order to achieve up to four simultaneous sessions, even using different terminal types.

TCP/IP and IPX allow several connections on a single server or on different servers. Using these protocols, Tun EMUL behaves as (up to) four different terminals. A mouse-click or <Alt>-key combination may be used to switch to another open session.

Tun EMUL IS NATIVE TO BOTH MS-DOS AND WINDOWS

The emulation and file transfer modules run under MS-DOS (EMUL.EXE) as well as under Microsoft Windows (EMULWIN.EXE). Both modules use the same parameters and configuration files. The Windows emulator also provides *cut & paste*, graphical print-screen, and automatic sizing of the emulation window.

In addition, the emulation module under Windows is compatible with the WinSock standard defined by Microsoft in order to use any TCP/IP kernel that provides a WinSock DLL.

Tun EMUL ALLOWS ACCESS TO MS-DOS DURING EMULATION

A microcomputer running terminal emulation does not lose its ability to act as a microcomputer. Tun EMUL provides a hot-key combination for switching between active MS-DOS programs and programs running on remote hosts.

In addition, the emulation modules are able to capture the data sent from remote connections into MS-DOS files, and to send the contents of MS-DOS files towards the host computer.

Tun EMUL PROVIDES FILE TRANSFER

Functioning between a PC running Tun EMUL and a UNIX host, the file transfer module offers the possibility of exchanging files between MS-DOS PCs and UNIX hosts. This feature is available to a user with an active emulation session.

There are two sides to the file transfer module:

- within Tun EMUL on the PC side
- the other installed on the host machine

The module may either be activated during an emulation session by simple keystrokes or mouse clicks, or by macros and MS-DOS batch files that can be called automatically, even from within an application.

File transfer may be initiated in a pop-up window in the emulation screen. Through the use of the RCOPY (remote copy) command, UNIX file systems are seen and treated as MS-DOS drives (W:, X:, Y:, or Z:). RCOPY can transfer files in either text or binary format, and can use wildcard characters (* and ?) to transfer entire directories. X, Y and Z modem protocols are also included.

In addition, UNIX shell scripts may be used to send escape codes that initiate file transfer, as well as many other actions.

Tun EMUL PROVIDES EMULATION FOR IBM 5250 AND 3270 TERMINALS

In addition to traditional asynchronous emulations, Tun EMUL also offers a separate 5250 emulation module for TCP/IP under Windows. 5250 is a synchronous emulation typically used in IBM AS/400 environments.

Tun EMUL PROVIDES REVAMPING

Under Windows, Tun EMUL integrates new three-dimensional graphics attributes associated with standard attributes (inverse video, underline, etc.) giving traditional UNIX applications an improved graphical aspect.

CHAPTER 3 - USING Tun EMUL

This chapter describes the use of Tun EMUL in a MS-DOS environment. In order to cover the basic functions used in terminal emulation, the following topics are discussed:

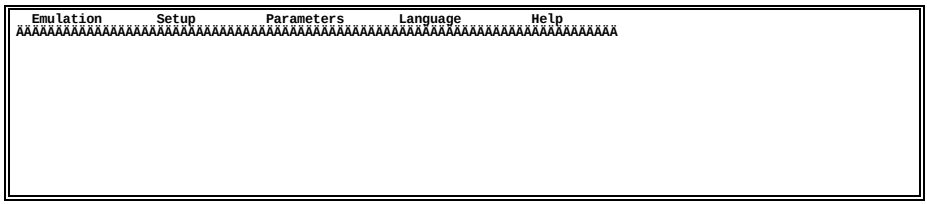
- Creating **configurations** for the various types of supported communications (RS-232, TCP/IP) using the Tun EMUL menus
- Running the emulator using a defined configuration
- Transferring files

USING THE CONFIGURATION PROGRAM

To enter the Tun EMUL Supervisor program, execute the following command:

C:\TUN\EMUL>

The figure below shows the Tun EMUL main menu:



The main menu options are as follows:

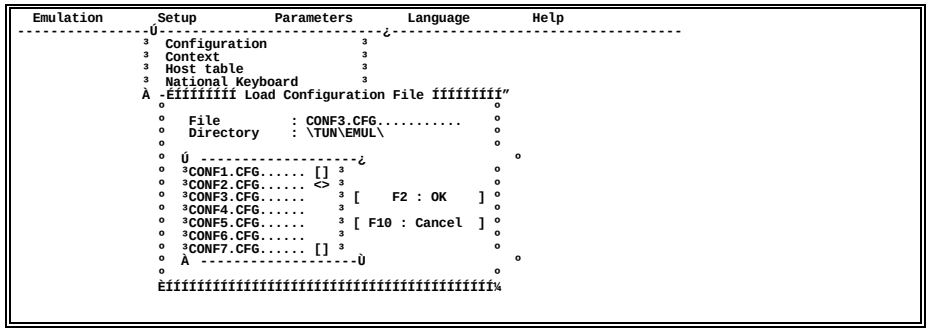
Emulation	Contains options to begin a terminal emulation session or exit Tun EMUL.
Setup	Definition of emulation configuration, the TCP/IP host table, and the national keyboard used in emulation.
Parameters	Customization of an existing emulation, or creation of a new one.
Language	Selection of the language used in Tun EMUL menus.
Help	Lists information regarding product version and serial numbers.

> **NOTE**: You may navigate within Tun EMUL menus with either the directional arrows and <Enter> key or with a mouse, if the mouse driver has been loaded.

CREATING CONFIGURATIONS

Defining **configuration** files is the first procedure after installing Tun EMUL. A configuration file tells the emulator what type of terminal (Ansi, vt220, Wyse60, etc...), as well as the type of connection (RS-232, TCP/IP, etc...) that you wish to use.

To define a configuration for an emulation session, use the mouse or the arrow keys to select **Setup ⇨ Configuration**:



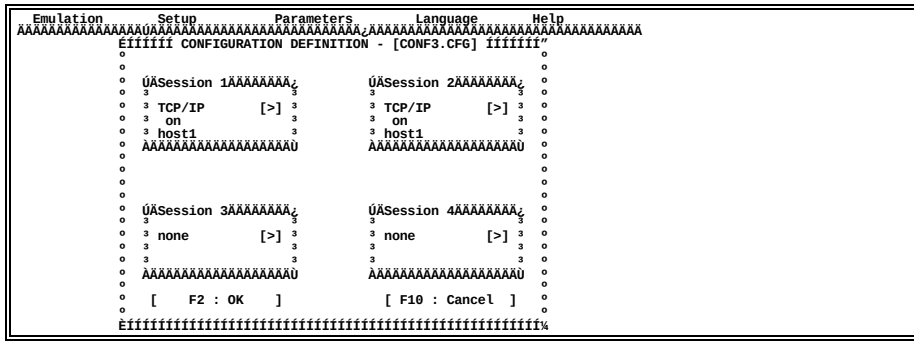
This window shows the sample configuration files already set up in Tun EMUL for your reference. You may select a configuration from this list and modify it to meet your needs, or create a new one by typing the new name in the **File** field.

The table below summarizes the contents of the pre-defined configurations:

Name	Connection type	Sessions
CONF1.CFG	Asynchronous	1
CONF2.CFG	Asynchronous	2
CONF3.CFG	TCP/IP	2
CONF4.CFG	TCP/IP	4
CONF5.CFG	Asynchronous + TCP/IP	1 rs-232, 2 TCP/IP
CONF6.CFG	Telephone	1
CONF7.CFG	Interrupt 14h	1

Media and session settings

After loading a configuration file, a summary of the current media settings and hosts is displayed, as shown in the sample CONF3.CFG shown below:



The preceding window shows the status of each terminal session defined in a configuration.

You may use the arrow keys or the mouse to select the session you wish to modify.

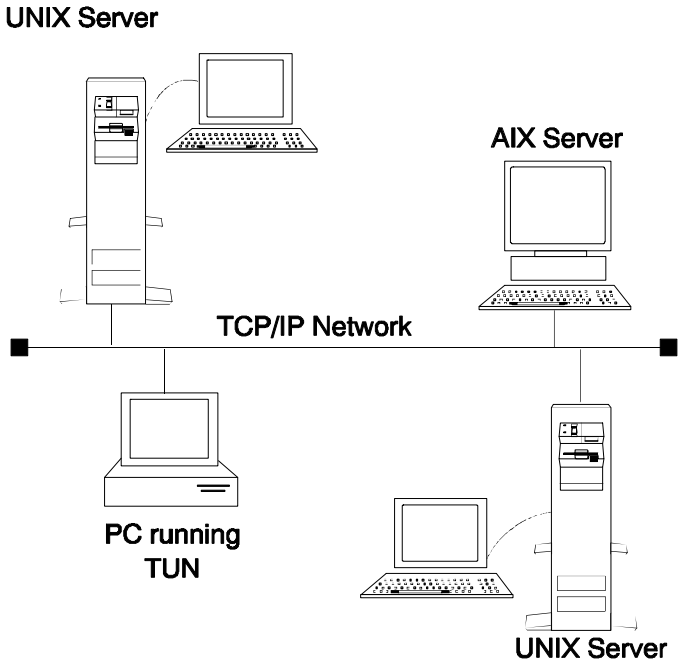
Press <Enter> to obtain the list of supported communications media. Depending on your choice, the screens that follow will present the options necessary to configure the session.

The next sections in this manual describe the various possibilities for defining emulation sessions using various type of communications.

> **Note**: It is possible to have a configuration file that defines simultaneous sessions that use different interfaces.

CONFIGURATION USING TCP/IP

On a TCP/IP network, you may access one or more remote servers and establish up to four simultaneous sessions. This allows you to fully take advantage of Tun EMUL's multi-session capabilities. A simple keystroke is all that is needed to switch from one session to another.



CONF3.CFG and CONF4.CFG are pre-defined TCP/IP configurations, and may be used as models for your own configurations.

With TCP/IP, you may have up to four (4) simultaneous sessions; for each session, you must fill in three fields (the **Display context** field is optional)

1. Host (name or IP address)
2. Terminal type
3. Display context
4. Time-out

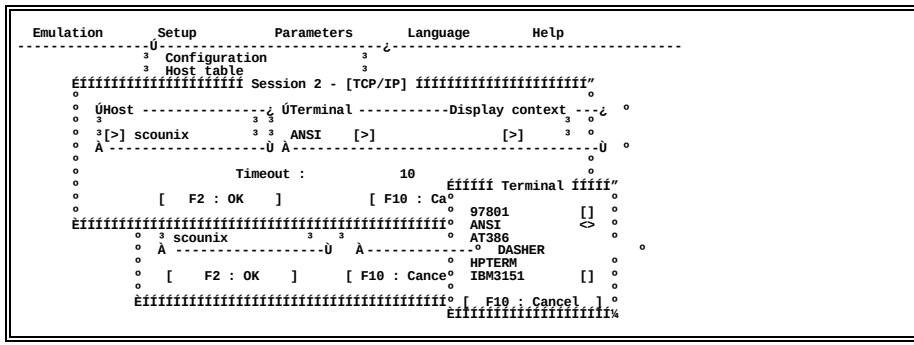
Host

Indicate the IP address or server name of an existing server on your network. If your network does not have a "name server", fill in the name of a server defined in the **Host table** on your PC (using options **Setup** ⇨ **Host Table** in Tun EMUL)

If you type a "?" in place of the IP address or server name, the emulator will display a prompt in which to enter them when you start the emulator. To select from the list of defined servers, click on the help button (**[>]**) with the mouse, or press <Enter> when it is highlighted.

Terminal

As the cursor enters the **Terminal** field, a help window listing the available terminals may be opened by clicking with the mouse, or pressing <Enter>:



Choose from one of the available terminals:

TERMINAL	DESCRIPTION
ANSI	SCO UNIX, SCO XENIX console
AT386	Interactive UNIX (386) console, UNIVEL
ATO300	ALCATEL APX
DASHER	Data General Dasher 412
FT	Fortune
IBM3151, HFT	IBM RS6000
HPTERM	Hewlett Packard console (2382A)
IMP	Printer (soft copy)
MINITEL	Minitel
TM266	Phillips P90x0
TO300	UNISYS U6000
TWS	BULL DKU 7102
VT52, VT100, VT220, VT320	DEC
WYSE50, WYSE60	WYSE
97801	Siemens/Nixdorf

Choose the terminal type appropriate for the host machine or the application you intend to use.

If none of the available terminals suit your needs, Tun EMUL offers you the possibility of defining a new terminal emulation.

For more details, see the **Parameters** option described in the chapter **Customizing Terminal Parameters** in this manual.

Timeout

Expressed in seconds, the value in this field is used mostly by the **File transfer module**. It determines how long the PC will wait before non-response from the host computer is considered as a transmission error. In general, the faster the host computer, the shorter this value may be.

Display context

A display settings file (.CTX) may be used in order to personalize emulation sessions. For example, screen settings and colors can be defined in a .CTX file, and assigned to an emulation session by placing its name in this field.

> **Note**: When telnet connection is established, the emulator supplies the name of the terminal it is emulating. By default, the name shown in the **Terminal** field (without the suffix) is given. If the server does not recognize this name (i.e. not described in its **terminfo** and **termcap** files), then you may change the name of the terminal within Tun EMUL.

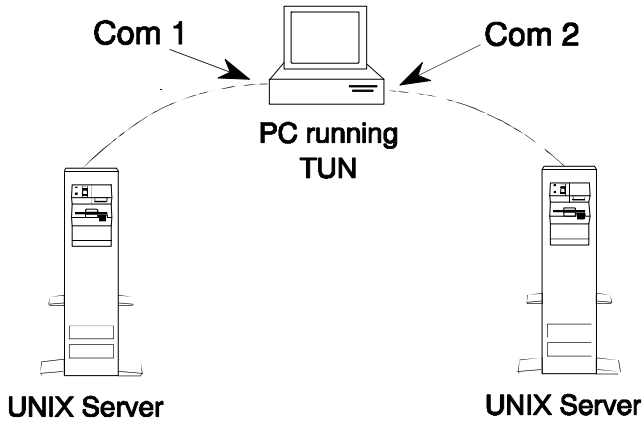
Saving your changes

After confirming your changes, click on <OK> or press <F2> to confirm and return to the main menu. You may click on <Cancel> or press <F10> to cancel any changes you may have made.

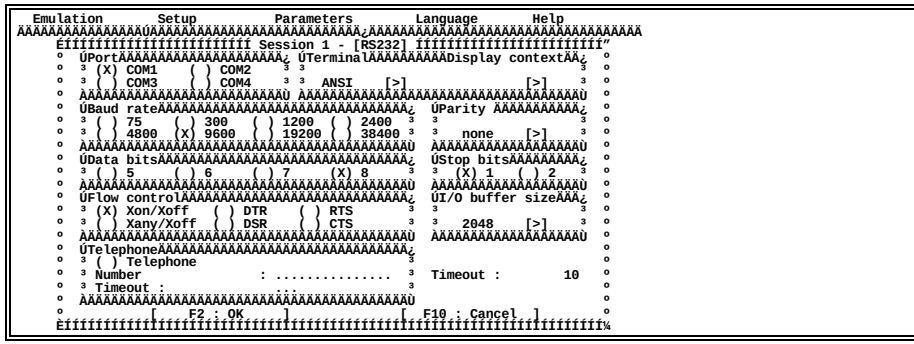
ASYNCHRONOUS (RS232) CONNECTION

Over asynchronous (RS-232) lines, the emulator can use COM1 - COM4 to establish multiple emulation sessions.

Except in special cases (such as **mscreen**), it is only possible to have a single session per physical connection. To obtain two sessions (even on different servers), there would have to be two physical connections, as shown in this example:



CONF1.CFG and CONF2.CFG contain sample configurations for serial connections.
Various parameters need to be specified for each session, as shown in the window below:



To navigate within the configuration window, use a **mouse**, the <Up> and <Down> arrows, or the <Tab> key.

When the cursor enters a field that has options in addition to those displayed, a **help window** may be displayed by pressing <Enter>. Available choices for fields such as **Terminal**, **Display context**, and **Parity** are given in pop-up windows. Press <Enter> to confirm a selection.

You may return to a previous field by pressing the <Up> arrow, or cancel the current operation and return to the previous screen by pressing the <F10>. When all options are correct, press <F2> to save any changes and return to the previous screen.

This next section discusses the fields contained in the configuration screen.

Port

This field specifies the communications port to be used: COM1 through COM4. Of course, ports in different sessions can be configured differently from one another (speed, data bits, stop bits, etc...) and emulate different terminal types (for example VT100.TER on COM1 and ANSI.TER on COM2).

To use COM3 or COM4 under MS-DOS, various parameters need to be defined using the option **Setup ⇨ Hardware parameters**.

Terminal

To display the list of available terminals, press <Enter> when the cursor is in the **Terminal** field. Choose one of the available terminals:

TERMINAL	DESCRIPTION
ANSI	SCO UNIX, SCO XENIX console
AT386	Interactive UNIX (386) console, UNIVEL
ATO300	ALCATEL APX
DASHER	Data General 412
FT	Fortune
IBM3151, HFT	IBM RS6000
HPTERM	Hewlett Packard console (2382A)
IMP	Printer (soft copy)
MINITEL	Minitel
TM266	Phillips P90x0
TO300	UNISYS U6000
TWS	BULL DKU 7102
VT52, VT100, VT220, VT320	DEC
WYSE50, WYSE60	WYSE
97801	Siemens/Nixdorf

Choose the terminal type appropriate for the host machine or the application you are going to use.

If none of the available terminals meets your exact needs, Tun EMUL offers the capability of defining a new terminal emulation. See the **Parameters** option described in the chapter **Customizing Terminal Parameters** in this manual.

Display context

This option is used to assign a **display context** file (.CTX) to the configuration. Display files may be used to assign color options to a monochrome emulation. You may also choose to not fill in this field.

Serial Line Settings

> **Note:** Communication parameters on a serial connection will vary with the UNIX host. Information regarding the transmission protocols and line speed may be found in the **/etc/ttytype**, **/etc/gettydefs**, **/etc/inittab** files on the host machine.
>
> Please make sure that the line settings in Tun EMUL and on the host machine are identical in every way.

Baud Rate

The transmission speed on an asynchronous line is measured in **baud** (bits per second). The baud rate may be set between 75 and 38400. This value should be set in "harmony" with the speed set on the server (in **/etc/gettydefs**).

Parity

The **parity bit** provides a way to protect against transmission errors. The parity bit may be either **even** or **odd**; **None** means that no parity bit is sent after a byte.

In 8-bit transmission, **space** or **mark** may used to set the last bit in a byte to 0 or 1.

Data bits

This field describes the number of significant bits used to make a byte. This number is almost always either 7 or 8. Check the setting on your host machine to be sure.

Stop bits

Either 1 or 2 bits mark the end of a byte.

I/O buffer size

This field defines (in bytes) the size of the Input/Output buffers. The default value of 2048 may usually be used.

Flow control

Flow control keeps the Input/Output buffers from overflowing and losing data. It is very important for the host machine to be set the same as Tun EMUL.

With Xon/Xoff, when the buffers in the PC under emulation become 75% full, it sends a DC1 (^S) character to the host asking it to suspend sending data. When the buffers become 75% empty, the PC sends a DC3 (^Q) character to request that the host resume sending data.

When using Xany/Xoff, the PC under emulation still sends a DC1 to suspend transmission, but can send **any character** to resume.

Hardware handshaking

Some UNIX servers handle flow control directly through cabling. Instead of using special characters (DC1 & DC3), electronic signals are sent when the PC's buffers are full. In general, two types of hardware handshaking are used:

- DTR and DSR signals
- RTS and CTS signals

Use the space bar or your mouse to select the values you need.

Timeout

Expressed in seconds, the value in this field is used mostly by the **File transfer module**. It determines how long the PC will wait before non-response from the host computer is considered as a transmission error. In general, the faster the host computer, the shorter this value may be.

Telephone

This option is used to indicate the presence of a telephone line, the associated number, and dial type. Please see the following section on **Modem Configuration** for more details.

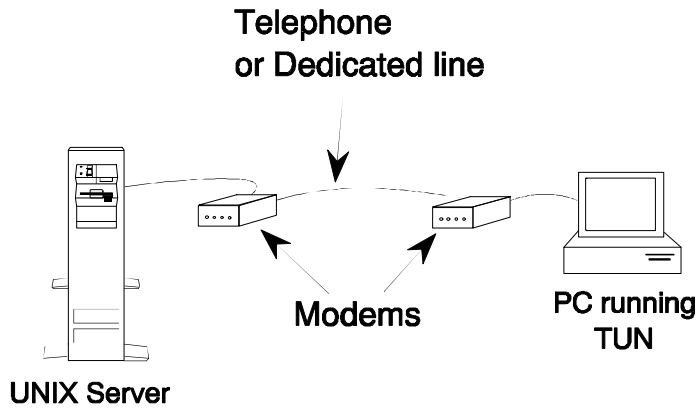
Saving Changes

To save changes to a configuration, press the <F2> key at any time. Press <F10> to discard any changes.

TELEPHONE LINE CONFIGURATION

A configuration over telephone line using a a modem is identical to that for a local serial connection. To perform emulation over this type of connection, the emulator needs to be given the necessary information for dialing and managing the modem.

Here is a sample diagram of a modem connection:



In this configuration, the PC acts just like a local terminal. Any host machine correctly equipped with a modem is accessible by PCs running Tun EMUL.

You may open and modify the default RS232 configuration file (CONF1.CFG), or make your own to dial the UNIX host of your choice.

Telephone

Enter a telephone number in the sub-window **Telephone** in the configuration screen for serial connections:

```
ÚTelephoneÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ¿
³ (X) Telephone                                    ³
³ Number               : 9,14153880752..           ³
³ Timeout :              30.                       ³
ÀÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÙ
```

If you place an "X" in the **Telephone** field, the telephone number contained in the **Number** field will be dialed when emulation is started. Tun EMUL will then wait for connection to be established in order to enter emulation. If you do not enter a number in the **telephone number** field, you must enter one manually after you start emulation.

You may use commas (**,**) to have Tun EMUL pause for two (2) seconds before continuing to dial. In many offices this is necessary in order to obtain an outside line or to call outside the country.

Tun EMUL uses the HAYES commands given in **Setup** ⇨ **Hardware parameters** for modem control. With this menu option, you may customize the codes needed by your particular modem or connection (MINITEL, 14400 baud...)

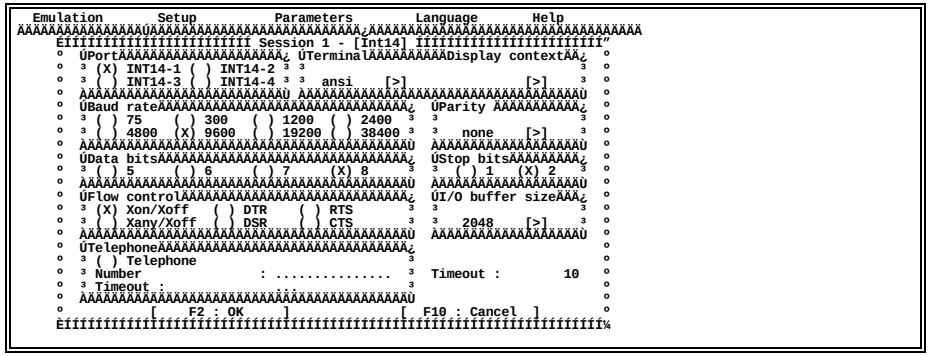
Saving changes

Click on **OK** or press <F2> to keep any changes; click on **Cancel** or press <F10> to return to the previous screen.

USING INTERRUPT 14H

Some communications cards and software (X.25, ISDN, NVT/IPX, TCP/IP) redirect BIOS Interrupt 14 in order for connections to take place as if over standard COMx ports. This type of interface "masks" the specifics of the particular protocol and provides a sort of "universal interface".

The drawback to this system is that connections actually take place "outside" the emulator, and that data is exchanged in character mode (character by character).



To configure a connection on Int 14h, follow the same instructions as for a serial connection. Instead of choosing COM1 or COM2, choose INT14-1, INT14-2, INT14-3, or INT14-4 depending on your software's Int 14h driver.

Saving changes

Click **<OK>** or press **<F2>** in order to save your changes; click on <Cancel> or press <F10> to return to the previous screen.

USING INTERRUPT 6BH

The "universal interface" concept with Interrupt 6Bh is very similar to Interrupt 14h, except that data is exchanged in **packet mode**.

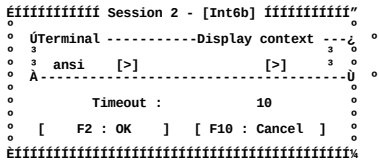
For example, NVT/IPX from Novell provides an Interrupt 6Bh interface with its resident program NVT.EXE.

The diagram below shows how the Tun EMUL emulator interfaces with Int 6Bh.

Tun EMUL
Int 6Bh (or Int 14h)
NVT.EXE (by Novell)
IPX, IPXODI
Network Card

When NVT.EXE is loaded, the user must press ^T to view the list of available servers currently running the NVT protocol. Selecting a server then establishes the connection that will be used by Tun EMUL on Interrupt 6Bh.

To configure a connection on Int 6Bh, select Int6Bh from the media selection window. The figure below shows the configuration screen for this type of connection.



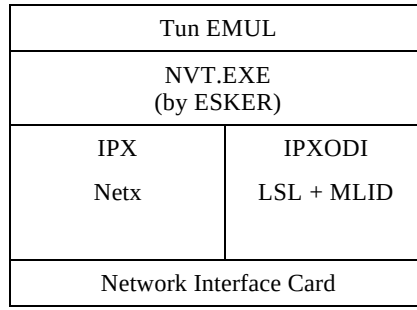
The version of NVT.EXE supplied with the IPX package for the UNIX server requires the user to select a server for connection. Therefore there is no need to select a host name within Tun EMUL. The only field necessary to fill in for this type of connection is **Terminal** (type); **Display context** is optional.

Click on <OK> or press <F2> in order to save your changes and return to the previous screen. Click on <Cancel> or press <F10> to quit without saving changes.

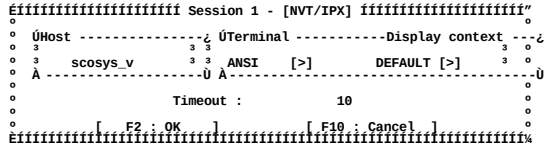
CONFIGURATION FOR NVT/IPX

The main drawback to using the Interrupt 6Bh interface provided by Novell is that server connections must be established before starting the emulator. In order to avoid this problem, Tun EMUL provides a version of NVT.EXE reads and writes to NVT/IPX connections, in addition to opening and closing them.

The NVT.EXE developed by ESKER is easier to use and more specifically designed for the Tun EMUL emulator. There is no need to leave the emulator to open or close a session.



After loading NVT.EXE (on top of IPX, IPXODI, or PDIPX), you may configure an emulation session using the following screen:

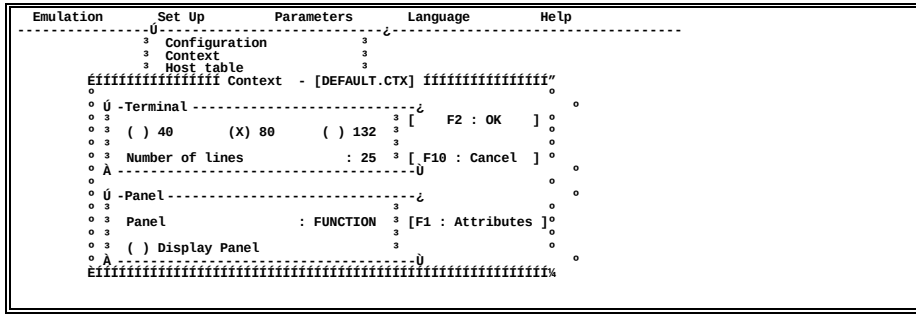


The **host** name is the name that was assigned to the UNIX server during installation of IPX. You may configure up to four (4) NVT/IPX sessions on different servers simultaneously.

DEFINING DISPLAY SETTINGS

As explained earlier, **context files** may be associated with emulation sessions.

Context files may be used to set the number of lines and columns used in an emulation, as well as to assign colors to (otherwise) monochrome video attributes such as underline, reverse video, etc. Mouse-driven key panels may also be associated with an emulation session.
To define a context, select **Setup ⇨ Context**. You may choose an existing .CTX file or create a new one by typing the name:



Terminal

The **Terminal** zone allows you to define the number of lines and columns that will be displayed when emulation is started. To select the number of columns, highlight the appropriate option and press the **space bar**.

You may use either 24 or 25 lines in emulation under MS-DOS.

> **Note**: PCs must be equipped with SVGA cards in order to perform emulation with 132 columns under MS-DOS (because the **bios mode** field needs to be filled in under **Options ⇨ Hardware parameters**). See section **Emulation with 132 columns** later in this manual.

Panel

The **Panel** option is used to associate a visual key-panel with an emulation session.

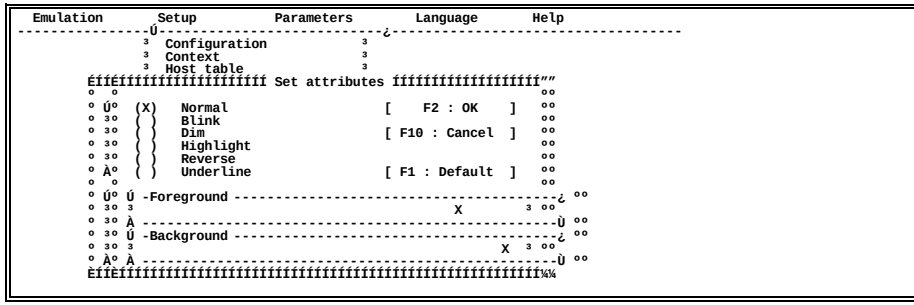
A key-panel is a window with buttons in an emulation screen. The user may click with the mouse on a particular button in order to start pre-defined functions or sequences. Defined in Tun EMUL under Windows, key-panels may be used to navigate within UNIX applications, to start Tun EMUL macros, or trigger other customized sequences.

In this menu, the **Panel** field contains the name of the desired **.PAN** file that defines the key-panel. If the **Display panel** box is marked with an "x", then the key-panel will be displayed on the emulation screen when the session is opened.

> **Note**: In order to be able to use key-panels, you must first load the mouse driver under MS-DOS.

Attributes

In order to modify video color attributes, click on **Attributes** or press <F1>:



With this screen you may change the foreground and background colors used in emulation for various video attributes. To do this, use the mouse click on the attribute you would like to change, then choose the desired color.

Saving your changes

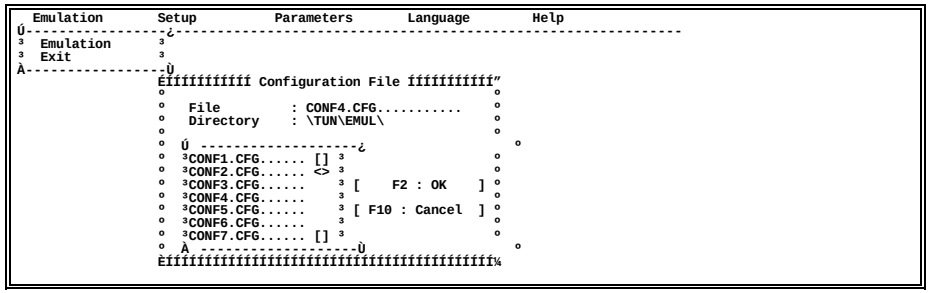
Press <F2> or click on <OK> to save your changes.

To save all the changes you have made to the current context file, press <F2> or click on <OK> again. If the context file is new, it will be saved at this point. To discard your changes, press <F10> or click on <Cancel>.

STARTING THE EMULATOR

Starting emulation using the menus

Select **Emulation** ⇨ **Emulation** from the Tun EMUL menu options in order to start the emulator using a pre-defined configuration. Select a configuration file from the following screen:



Emulation begins by displaying the numbers 1 to 5, (or 1 to 10 for two sessions, or 1 to 15 for three sessions, etc.).

After this initialization, the screen will be erased, and the PC will behave as a terminal waiting to receive a **LOGIN** from the host machine.

Starting emulation from the MS-DOS prompt

The emulator may also be started from the MS-DOS prompt by issuing the following command:

```
C:\TUN\EMUL>EMUL config
```

where:

config the name of an existing configuration file (.CFG).

Connection problems

When starting a serial configuration, pressing <Enter> several times usually establishes connection with the host. If no communication takes place, and no **login** is presented on the screen, it may be due to one of the following reasons:

1. Incorrect cabling (See **Cabling for serial connections** in the Appendices).

2. The communications protocol configured in Tun EMUL is not the same as on the host machine. Notably, for serial connections verify that the communication parameters such as speed, data bits, stops bits, etc. in the **Setup**, **Emulation** menus.

3. In a TCP/IP network, first make sure that the TCP/IP kernel is active on the PC, and then make sure the PING utility returns the message "Host responding" on the desired server.

Quitting emulation

The key-combination **<Alt><F10>** can be used to end an emulation session, whether you started the emulator from within TUNEMUL or from the MS-DOS prompt.

USING THE EMULATOR

Pre-defined function-keys

A series of special actions are available during terminal emulation sessions by simultaneously pressing the <Alt> key and a particular function key <Fn>. These actions are described in table below:

Key combinations	Action
<Alt><F1>	Switch to terminal session 1
<Alt><F2>	Switch to terminal session 2
<Alt><F3>	Switch to terminal session 3
<Alt><F4>	Switch to terminal session 4
<Alt><F5>	Switch to a MS-DOS session
Alt><F6>	Open command execution window
<Alt><F7>	Send files to the host machine
<Alt><F8>	Begin receiving a file (see description further in this chapter)
<Alt><F9>	End of file reception (see description later in this chapter)
<Alt><F10>	End of emulation
<Alt><F12>	Switch to a MS-DOS session by swapping the emulator out of RAM

Switching sessions (<Alt><F1>- <Alt><F4>)

These key combinations are used to switch between active emulation sessions. You may have up to four (4) simultaneous sessions in a (TCP/IP) configuration.

Quitting the emulator (<Alt><F10>)

In order to close all open emulation sessions and return to MS-DOS, press <Alt><F10>.

Switching to MS-DOS (<Alt><F5>)

You may switch between a MS-DOS session and a terminal emulation session by pressing the <Alt><F5> keys.

> **NOTE:** You must close an open MS-DOS session by using "exit" before Tun EMUL will let you quit the emulator.

The emulation program remains resident when <Alt><F5> is used, and the memory used by EMUL.EXE is not freed. <Alt><F5> reacts as a true "hot-key".

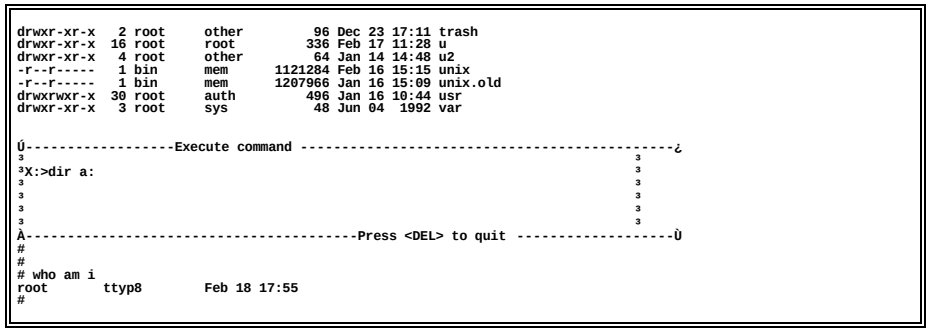
Switching to MS-DOS (<Alt><F12>)

You may also open a MS-DOS session from within an emulation session by pressing <Alt><F12>. In this case, the memory occupied by the emulator is swapped, and there should be enough memory left under MS-DOS to execute most programs.

Unlike <Alt><F5>, this action does not perform as a true "hot-key". In order to return to the emulation session you must close the MS-DOS session with an **EXIT** command.

Command execution (<Alt><F6>)

Pressing <Alt><F6> opens the window shown below in the emulation screen:



In a window that resembles a MS-DOS environment, UNIX volumes in open emulation sessions are represented by "W:", "X:", "Y:", and "Z:".

You may change drives by using the usual MS-DOS commands:

```
X:>C:
```

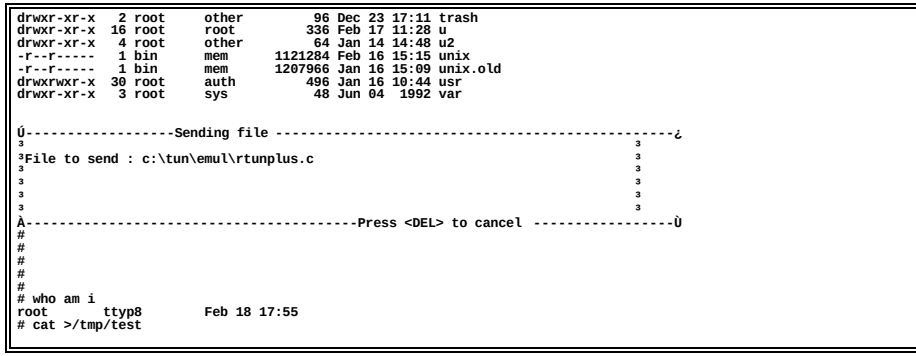
or

```
X:>A:
```

From this window, you may execute MS-DOS programs on drives A: through E:, or commands on the host systems W: through Z:. Commands executed on host systems appear on the emulation screen.

Sending files (<Alt><F7>)

This feature allows you to send a MS-DOS file to the host. When you press <Alt> and <F7> simultaneously, the following window appears:



This is very useful for sending the source code of the **File transfer module** (rtunplus.c) to the host machine during installation. After that, it is best to use <Alt><F6> in order to transfer files using the **rcopy** command.

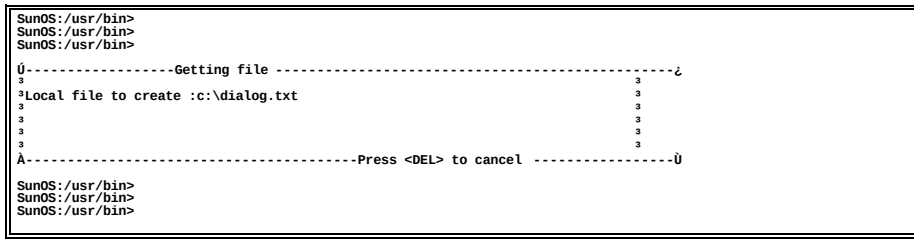
Pressing <Del> or <Esc> will bring you back to emulation screen.

Receiving a file (<Alt><F8> and <Alt><F9>)

With this feature, you may capture all data transmitted across a connection into a MS-DOS file.

Since host machines can most often "echo" on the screen whatever is typed at the keyboard, you can use this feature to capture **escape sequences** and other codes sent by the server in order to analyze them; the entire man-machine dialogue may be observed in this way to isolate problems.

Pressing <Alt> and <F8> simultaneously displays the following screen:



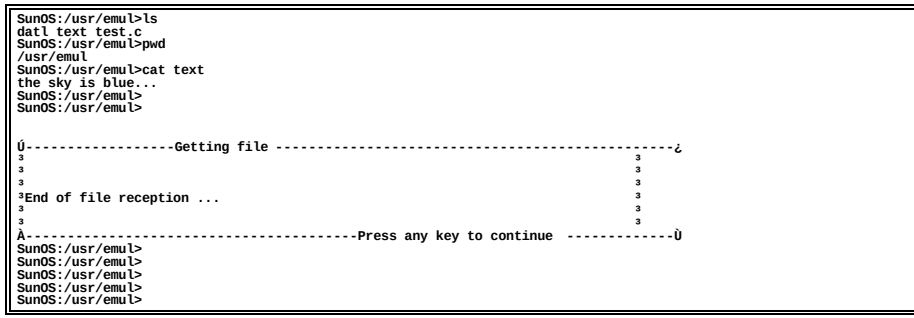
Pressing <Del> will erase this window and return you to the emulation session.

In order to illustrate this function, type "c:\dialog.txt" at the prompt in the window, and press <Enter> to confirm. The window will disappear, and you will be brought back to the emulation screen. From then on, everything that appears on the screen will also be sent to a file called "c:\dialog.txt".

Type a few commands at the UNIX prompt (# or $). For example:

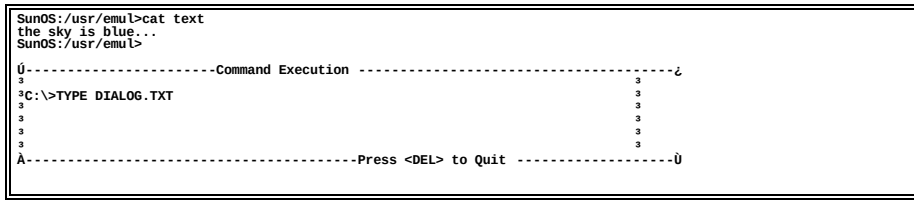
```
SunOS:/usr/emul> ls
datl text test.c
SunOS:/usr/emul> pwd
/usr/emul
SunOS:/usr/emul> cat text
the sky is blue...
SunOS:/usr/emul>
```

After executing these commands, press <Alt> and <F9> at the same time. The following message is displayed:



The file "dialog.txt" is then created on the C: drive, and redirection is closed. Press any key to return to emulation.

To view the contents of the captured file, press <Alt><F6>, then type:



The preceding dialogue is then displayed:

```
SunOS:/usr/emul>ls
datl text test.c
SunOS:/usr/emul>pwd
/usr/emul
SunOS:/usr/emul>cat text
the sky is blue...
SunOS:/usr/emul>
```

Press <Enter> to return to the **command execution** window, then <Del> to return to the emulation session.

> **Note**: This feature is most useful for capturing and analyzing codes and escape sequences sent by host applications, and for analyzing keystrokes that do not seem to produce to the desired reaction.

Please consult the Customizing Terminal Parameters chapter for details on how to modify keyboard and function keys.

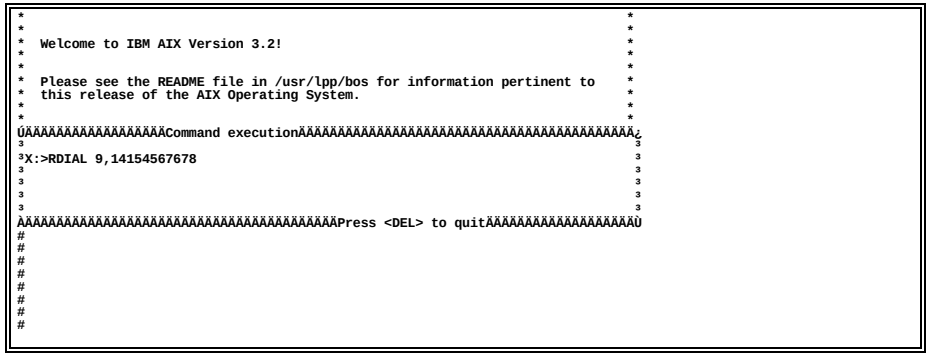
The **Send File** and **Receive File** functions are limited by the fact that only ASCII files may be transferred. For regular exchange of files between host machines and PCs, please use the **File Transfer Module** (rcopy).

EMULATION OVER TELEPHONE LINE

Manual dialing

If you specify a telephone number in a configuration file, the emulator will automatically dial it when started. By default Hayes-compatible commands are used for modem control, but if your modem differs slightly from this standard, you may enter other codes using **Setup ⇨ Hardware parameters** in TUNEMUL.EXE.

You may also choose to manually enter a phone number after starting the emulator. To do this, press <Alt><F6> and enter the RDIAL command followed by a phone number, as shown below:



RDIAL is the command to open the telephone line, dial a number, and wait for the host to respond.

To end a telephone connection, simply quit the emulator by pressing <Alt><F10>.

If you wish to close the connection but remain in the emulator, use the command **RHANGUP** in the command execution window <Alt><F6>.

Minitel Emulation

CONF6.CFG has been configured to use a modem connection in order to emulate a **MINITEL** terminal. In France, MINITEL is an on-line information service with access to telephone books, transportation reservations, financial services, etc.. The local telephone number for France Telecom's MINITEL service is **11**.

For MINITEL emulation, the modem must be capable of standard v23 communication. This standard calls for a transmission speed of 75 baud, and a reception speed of 1200 baud. See the documentation for your modem to find the code to initialize the modem in v23 mode (such as Hayes command **ATB2**). You may then enter this code using the option **Setup ⇨ Hardware parameters**.

The following diagram shows the function-key mapping used in the MINITEL emulation:

Return	Next	F1	F2
Void	Correction	F3	F4
Guide	Summary	F5	F6
Repeat	End Conn.	F7	F8
Escape	Graph/text	F9	F10

INTERACTIVE FILE TRANSFER

Performing transfers

The **file transfer** module allows you to easily exchange files between MS-DOS PCs and UNIX hosts. The only protocol available when using Tun EMUL's MS-DOS emulator is ESKER's proprietary RTUNPLUS (X, Y and Z modem may be used in the Windows emulator).

RTUNPLUS must first be installed on the host machine in order for you to be able to copy files as described in this section. See **Installation of rtunplus** in Chapter 1 for more details.

> Note: The **Timeout** value in the configuration setup plays an important role in file transfer. If the PC has not received a response from the host after the specified lapse of time, Tun EMUL considers that a file transfer error has occurred. Usually, the default values may be used, but if response time from your server is slow, you may need to increase the Timeout value.

To perform file transfer, run Tun EMUL and enter the emulation of your choice. Once logged in to UNIX host (with a # or $ prompt), activate the **Command Execution** window by simultaneously pressing <Alt><F6>:

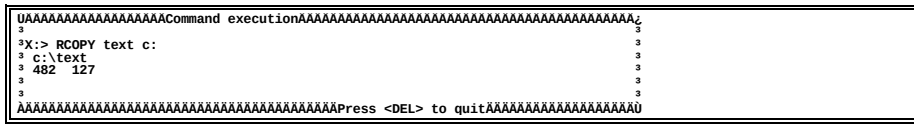
```
ÚÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄCommand executionÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ¿
³                                                                                ³
³X:>                                                                             ³
³                                                                                ³
³                                                                                ³
³                                                                                ³
³                                                                                ³
ÀÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄPress <DEL> to quitÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÙ
```

Using RCOPY

The command used to initiate file transfer from this window is **rcopy**.

Here is an example of how to copy a file from the UNIX host to the MS-DOS PC; in this case the file "text" is copied from the current UNIX directory to the current MS-DOS directory on the C: drive.

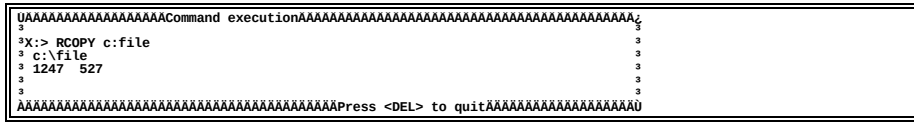
After you press enter, the status of the transfer is displayed:



The figure on the left (482) indicates the size of the file to transfer, and the figure on the right (127) counts the characters as they are sent.

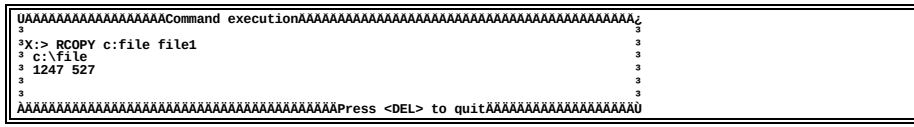
The **rtunplus protocol** contains error checking and automatic retry in case of errors. For copying files across telephone connections, you may need to reduce transmission speed depending on the quality of the line.

Here is an example of a file copy from the current MS-DOS directory to the current UNIX directory:

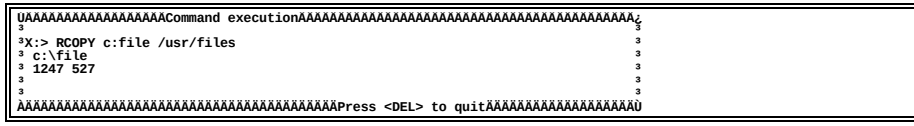


X: is used to present a MS-DOS-type environment for the UNIX volume in the first emulation session. In this way, you may copy from C: or A: to X: just as you would in MS-DOS. X: represents the current drive and directory on the UNIX host.

Just as in MS-DOS, you may change the name of the destination file:



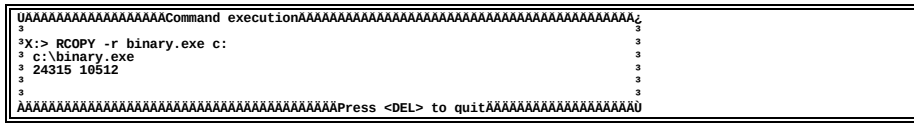
or directory:



CR/LF Conversion

When used without options, **rcopy** performs CR/LF conversion (Carriage Return/Line Feed) on the CR (0x0d) and LF (0x0a) characters in order to maintain file consistency between the two systems: LF (UNIX) ⇔ CR/LF (MS-DOS)

In order to copy **binary** files, use the option **-r** in order to avoid conversion:



When transferred with **-r**, the file will be identical, byte-for-byte, on both UNIX and MS-DOS.

Using wildcard characters

You may also use wildcard characters to transfer several files at once.

For UNIX files:

?	replaces any single character
*****	replaces any string of characters

UNIX files may have names like:

```
hosttab
client.info
help.network
```

Therefore, using wildcard characters:

f*	designates **all** files beginning with the letter "f"
???.*	designates all files with a suffix of any kind, with a prefix of three characters.

Since file names in MS-DOS are composed slightly differently, the meaning of wildcard characters is also slightly different. Files in MS-DOS can contain two parts:

1. a prefix (8 characters maximum)
2. a suffix (3 characters maximum)

The two parts are separated by a period, for example:

```
COMMAND.COM
AUTOEXEC.BAT
```

For MS-DOS files:

?	replaces any single character
*****	designates any prefix or suffix

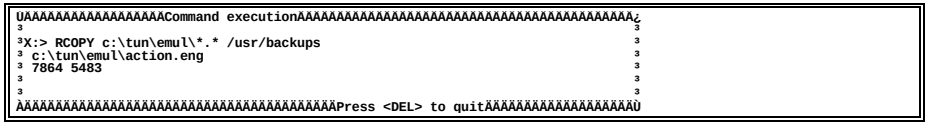
With wildcards:

f*	designates all files starting with f, but without a suffix.
f*.*	designates all files starting with f.

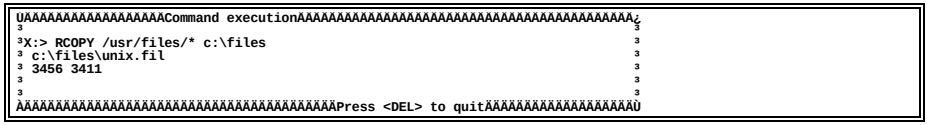
UNIX files with names longer than that allowed by MS-DOS will be truncated.

Copy example: MS-DOS to UNIX

A **directory name** may also be given as the name for the source file; in this case, all files in the directory and all subdirectories will be copied.



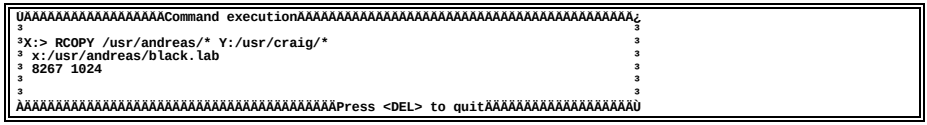
Copy example: UNIX to MS-DOS



The **File transfer module** may also be used to copy files between UNIX hosts. When emulation sessions are open on several UNIX servers, **X:** represents the UNIX drive in the first session, and **Y:**, **Z:**, and **W:** represent those that follow.

To copy between UNIX hosts, simply use the mnemonics for the UNIX drives as you would for copying between MS-DOS drives.

Copy example: UNIX to UNIX:



The **rcopy** command can transfer files between drives A:, B:, C:, D:, E:, X:, Y:, W: and Z: (but not from X: to X:, Y: to Y:, Z: to Z:, and W: to W:).

Transfer problems: SCO mapchan

File transfer does not work correctly if **mapchan** is active on an SCO UNIX system because the "checksums" used in the **rcopy** protocol do not come out correctly.

To avoid any problems, mapchan must be deactivated during the transfer:

1. Rename /usr/bin/**rtunplus** to /usr/bin/**rtunplusmap**,

2. Create a shell script /usr/bin/**rtunplus** as follows:

```
/usr/bin/mapchan -n
/usr/bin/rtunplusmap $*
/usr/bin/mapchan -s
```

3. Make the shell script **executable**.

```
chmod a+x /usr/bin/rtunplus
```

Your session must be in "shell"; file transfer will not work if a UNIX application is running at the same time.

Note:	To perform more complicated file transfers, please refer to **Automation of Connections and Transfers** in this manual.

CHAPTER 4 - EMULATOR AUTOMATION

This chapter is intended for service providers, system integrators, and information services staff who wish to provide their users with transparent access to UNIX resources.

The techniques explained in this section make it possible to completely automate connection, disconnection, file transfer, UNIX program startup, and other actions.

MACRO PRINCIPLES

Tun EMUL includes a high-level macro-language which serves to create programs that may replace the keyboard and totally or temporarily control an emulation session. Such a program may be called when the emulator is started, and/or when the user exits emulation.

For example, the instructions of the Tun EMUL macro-language can be used for the following:

- sending a character string over the connection
- waiting for a particular character string from UNIX within a specified time
- waiting for a period of time expressed in seconds.
- retries
- testing return codes of certain instructions
- testing characters received
- ending an emulation session
- returning to an emulation session
- to show or hide characters sent by the host
- prompting the user to enter information
- handling variables

SYNTAX

All automation macros must be stored with the suffix ".MAC" (i.e. connect.mac) in the directory containing Tun EMUL. Macros may be created using an ordinary text editor, and do not have to be compiled since they are directly interpreted by Tun EMUL.

One of the most common ways to use macros is to give them as parameters on the command line when starting the emulator.

In character mode:

```
emul config.cfg [-M in_macro] [-Q out_macro]
```

-M indicates that the next parameter is the name of a program file (.MAC) to be run when emulation is started.

-Q indicates that the next parameter is the name of a program file (.MAC) to be run when the emulator is stopped (<Alt><F10> or **File ⇨ Quit**).

It is not necessary to indicate the suffix (.MAC) in either case.

> **Note:** **"-Q"** is often used for disconnection operations such as "**Send Ctrl-D"** to make sure that UNIX sessions are closed correctly.

SAMPLE MACRO

The macro given below as an example automates access to an SCO UNIX application (**sysadmsh**) by prompting the user for his account name and password, then executing the program:

```
# Characters sent by Host computer not displayed
Hide

# Start
label BEGIN
# Read login and passwd
ReadVar "Enter your user name : " USER
ReadPasswd "Enter your password : " PASSWD
# Make connection
Repeat 3
# Send carriage-return character
        Repeat 5
                SendAndReceive 1 "\n" "login"
                IfNoError break
        end
        IfError NOCONNECTION

# Send login
        SendAndReceive 15 "%USER\n" "assword" "# "
        IfError continue
        IfEqual "# " break
        SendAndReceive 15 "%PASSWD\n" "# " "ogin" "TERM"
        IfError continue
        IfEqual "# " break

# Return to start of program if login incorrect
        IfEqual "login" BEGIN
# Set the TERM variable if necessary
        SendAndReceive 15 "\n" "# " "ogin"
        IfError continue
        IfEqual "# " break
        IfEqual "ogin" BEGIN
end
# Start application
Send "sysadmsh\n"
# Display received characters
Display
# Return to the emulator
Return

# No login
label NOCONNECTION
Echo "communication failed"
ReadVar "Press Return to quit" RESPONSE
# Exit the emulator
exit
```

The same type of program may be designed to establish more complex connections, such as sending modem commands, connecting through an x.25 PAD, navigating within a Unix application, etc.

The preceding model may also be used to run UNIX applications other than "sysadmsh". For example, you may replace the line **send "sysadmsh\n"** by a "send" of another UNIX command or shell script.

LANGUAGE SYNTAX

An instruction is a series of words separated by spaces, with only one instruction per line. The first word in a line is the name of the instruction.

In this manual, instruction names are combinations of upper and lower case letters, merely to simplify reading. This is of no importance with respect to programming, and either upper or lower case letters may be used to write instructions (i.e. "*SendAndReceive"* may be written "*SENDANDRECEIVE*" or "*sendandreceive*").

If the first character of a line is "*#*", the entire line will be considered to be a comment.

The macro-language only manipulates character strings. A character string is a series of symbols from 0 to 255, enclosed within double quotes (i.e. "Tun is communications software").

The following instructions have a return code that can be tested by the "*IfError*" or "*IfEqual*" commands:

```
ChangeTerminal
LoadPanel
RcopyGet / RcopyPut
Rcoupe
Receive
Rdial / Rhangup
Rnumero
SendAndReceive
```

To represent non-printable characters, the programmer may use the following notation :

\n	"line-feed" character (0x0a)
\r	"return" character (0x0d)
\t	"tab" character (0x09)
\E	"escape" character (0x1b)
\%	(%) character
\\	(\) character
^A to ^Z	control characters
\xnn	any hexadecimal character
\nnn	any octal character
\fnn	to use the value of one of the function-keys defined in Tun EMUL

Variables may be defined to store character strings in memory, and the number of variables is unlimited. Variables may be used to replace one or more instruction parameters.

When a variable is used, its name must be preceded by the "%" character (i.e. : SendAndReceive 3 "%toto" "%titi").

Conversely, when a variable is defined and assigned, its name must not be preceded by the "%" character (i.e. : Set toto "abcde").

If an instruction uses a variable which was not first defined in the macro, the interpreter interrogates the MS-DOS environment to check that it is present. If it is not, a null string is substituted.

LIST OF INSTRUCTIONS

Index of Macro Language instructions:

Break	Unconditional exit from a loop
ChangeTerminal	Instant change of terminal type
ClearMessage	Erase messages in a dialogue box
ClearScreen	Clears the screen
ClosePanel	Erases and unloads the current key panel from memory
Continue	Unconditional return to a loop
Display	Displays characters received by the emulator
Dos	Executes a MS-DOS command
Echo	Displays character strings in the dialog box
End	Marks the end of a repetitive block
Exit	Unconditional exit from the macro as well as the emulator
ExitIfDisconnect	Causes exit from emulation in case disconnection is detected (TCP/IP)
GetVar	Assignment to the last string received (by a Receive instruction) of a variable
Goto	Jump to a label
Hide	Characters received by the emulator are not displayed
HideMessage	Dialog field is not displayed
HidePanel	Erases current key panel
IfConnected	Tests to see of a connection is still active (TCP/IP, modem)
IfEqual	Tests to see if the last string received is equal to a certain value
IfError	Tests the status of the last Receive command

IfNoEqual	Compares the last string received to a value for inequality
IfNoError	Tests the status of the last Receive command
Label	Defines a label
LoadPanel	Loads a function-key panel
NoExitIfDisconnect	Inhibits ExitIfDisconnect (default setting)
Pause	Waits a specified period of time
RcopyGet	File transfer from UNIX to MS-DOS
RcopyPut	File transfer from MS-DOS to UNIX
Rcoupe	Disconnects modem communication
ReadVar	Assigns a character string typed on the keyboard to a variable
ReadPasswd	Assigns a character string typed on the keyboard to a variable without displaying it on the screen
Repeat	Starts a block of reiterative instructions
Receive	Waits to receive one or more strings
Return	Unconditional exit from macro and return to the emulator
Rdial	Dialing using a Hayes-compatible modem
Rhangup	Ends modem communication
Rnumero	Dialing using a Hayes-compatible modem
Send	Sends a character string over the communication channel
SendAndReceive	Sends a character string over the communication channel and waits to receive one or more character strings.
SessionTitle	Assigns a name to the emulation session
Set	Defines and assigns a variable
SetHelpFile	Defines a help file

SetVar	Assigns the last string received by a "Receive" type command to a variable.
ShowMessage	Displays the dialog field
ShowPanel	Displays a function-key panel
Sleep	Waits a specified period of time
Title	Assigns a title to a dialog box

For detailed information on the syntax and use of these commands, please see chapter **Macro Language Syntax**.

CHAPTER 5 - SPECIAL ACTIONS

Tun EMUL offers a wide variety of basic **actions** that have been pre-defined in the configuration files for many terminals. These actions include cursor movement, clear screen, switch to reverse video, etc.

In addition to providing these types of basic actions necessary for proper emulation, Tun EMUL was designed to take advantage of the computing power and intelligence of PCs, and offers many additional actions that are not usually found in terminals.

Because these actions are triggered by **escape sequences**, it is not possible to deliver a standard, pre-configured product that meets all the possibilities that users may desire.

To help you enrich your applications and use Tun EMUL to its maximum, this chapter presents some of the most-requested modifications.

Note: Please also refer to the chapter **TERMINAL CUSTOMIZATION** in this manual for a detailed explanation of the configuration files described in the following examples.

The actions described in this chapter are also documented in the files **ACTION.ENG** and **ACTION.DOC** in the Tun EMUL default directory.

QUITTING EMULATION UPON SERVER REQUEST

Parametered action no. **299** may be used to quit the emulator.

The parameter associated with this action is the **return code** that the emulator sends to the program that sent the escape sequence. For example, **299(1)** is equivalent to **exit(1)** in a C program.

In ANSI emulation, this action is associated by default to the escape sequence **\E\EQ** (which may also be written **1B 1B 51** in hexadecimal). This is defined with the following line in the file ANSI.SEQ:

```
\033Q s 299(0)
```

\033 is the "escape" character in octal notation (1B in hexadecimal). It only needs to be present one time on the line above because the ANSI.SEQ file uses **\033** as a header for all sequences (the header is contained on the second line of ANSI.SEQ).

If you would like to add this action to another emulation, choose an appropriate escape sequence and assign it to action 299.

Sending the configured escape sequence (in a shell script or simple **echo** command) from the UNIX host would then close the emulator:

```
echo "\033\033Q"
```

The best way to understand this procedure is to see it work, so feel free to try!

FILE TRANSFER REQUESTED BY SERVER

This is an example that may be used in the **MS-DOS** emulator, but you may easily adapt it for use under Windows.

Parametered action no. **268** initiates file transfer between the PC and the server without user intervention. The parameters associated with this action are those that are given to the **RCOPY** command under emulation (see chapter **File Transfer**).

For example, the string **C:\AUTOEXEC.BAT X:/tmp** would indicate that the MS-DOS file AUTOEXEC.BAT should be copied to the /tmp directory on the sever in emulation session number one (X:).

In the default ANSI.SEQ file, the following escape sequence has been added in order to perform this type of file transfer:

```
R%p0%s p 268
```

The "escape" character is not needed in the above sequence because it is given as a **header** for all sequences in the ANSI.SEQ file. To add this function to another emulation, simply copy **R%0%s p 268** to the relevant .SEQ file. To trigger this action from the UNIX host, you could use a shell script such as:

```
echo "\033R\"C:\\\\\\AUTOEXEC.BAT X:/tmp\""
read prog
$prog
```

> **Note**: Several "\" characters are necessary because of the succession of command interpreters that "strip" them. File transfer may also be performed using Tun EMUL's macro language (**RcopyPut** and **RcopyGet** commands)

Here is an example of an echo command that would transfer the UNIX file /etc/termcap to \TUN\EMUL on the PC:

```
echo "\033R\"X:/tmp/termcap C:\\\\\\TUN\\\\\\EMUL\""
read A
$A
```

PC PROGRAMS STARTED BY SERVER

Parametered actions nos. **295** and **296** may be used to start MS-DOS applications without user intervention. The difference between these two actions is only noticeable under MS-DOS, where action no. 295 **clears** the emulation screen, and action no. 296 **saves** the emulation screen.

The program to execute is given as a parameter to these actions. For example, to start **Edit.exe** under MS-DOS you could give **C:\DOS\EDIT** as a parameter.

In order to use these actions, add a line similar to the following in the relevant .SEQ file:

```
\033X%p0%s p 296
```

You can use one "escape" character on this line if the header in the .SEQ file already provides one (on the second line). To use this sequence, you could run a shell script as shown below:

```
echo "\033\033X\"C:\\\\\\DOS\\\\\\EDIT\""
```

> **Note**: The use of several "\" characters is necessary because of the succession of command interpreters that "strip" them.

This type of operation may also be performed using Tun EMUL's macro language (**Dos** command)

MACRO EXECUTION REQUESTED BY SERVER

Parametered action no. **264** may be used to start a Tun EMUL macro at the request of the server, without user intervention.

The parameter for this action is the name of the macro file to execute. For example, C:\TUN\EMUL\SYSADMSH.MAC would start the **sysadmsh** macro delivered (as a sample) with Tun EMUL.

In the ANSI.SEQ file, the following line has been added to carry out this action:

```
\033M%p0%s p 264
```

From the UNIX host, you could then execute a macro using an **echo** command or a shell script such as this:

```
echo "\033\033M\"sysadmsh.mac\""
```

TRANSPARENT PRINTING

Printers attached to a PC in emulation may be used to print UNIX files and from UNIX applications. To accomplish this, Tun EMUL has defined several actions that redirect output towards the printer instead of the screen (or both at the same time). This procedure is called Transparent printing.

Here is a description of these actions:

259	reactivate character display on the screen
260	do not display characters on the screen
261	end of printer redirection
267	redirection of characters towards an LPT port

The escape sequences pre-defined to trigger the above actions are as follows (in ANSI, TM266, VT100 et VT220 emulations):

\E[5i	redirection to the printer
\E[4i	end of printer redirection

If you need to define these sequences for other emulations, add the following lines to the appropriate **.SEQ** file:

```
\E[5i s 260 267
\E[4i s 261 259
```

Custom Print Command

You may also write a shell script, such as the one below called **prndos**, to print files using a single command:

```
for file
do
      echo "\033[5i"
      cat $file
      echo "\033[4i"
done
```

Once this shell script has been made executable, it may be used to print UNIX files:

```
prndos /etc/inittab
```

Using other printer ports

To use a second LPT port (LPT2), add the following lines to the appropriate ".SEQ" file:

```
[7i s 260 262("LPT2")
[6i s 259 261
```

LPT3 may also be used by replacing [7i and [6i with [9i and [8i respectively.

Printer redirection for LPT2 and LPT3 will work the same as for LPT1.

DYNAMICALLY CHANGING TERMINAL TYPE

Applications on UNIX hosts do not always use the same terminal type, even on the same server. Using one application after another in an emulation can therefore present some problems.

Action no. **270** was designed to correct this situation, allowing the terminal type to be changed without having to close the current session.

The following line has been added to the default ANSI.SEQ file:

```
\033T%p0%s p 270
```

The UNIX server could then initiate a terminal change by sending the following sequence:

```
echo "\033\033T\"VT220\""
```

AUTOMATICALLY CHANGING SESSIONS

Because Tun EMUL is completely multi-session, an action is provided to allow the UNIX host to change the active terminal session without user intervention.

Action no. **294** is used to switch sessions automatically, with the desired session number given a a parameter (from 0 to 3).

The following line is contained in the default ANSI.SEQ file:

```
\033S%p0%s p 294
```

Here are two possible UNIX commands to change sessions:

To active session 1:

```
echo "\033\033S1"
```

To activate session 2:

```
echo "\033\033S1"
```

MOUSE SUPPORT IN UNIX APPLICATIONS

In order to provide a way to use a standard mouse in UNIX applications, Tun EMUL includes a series of actions for mouse management.

Principle

Tun EMUL is capable of sending definable sequences each time a mouse event occurs, just as if a function key were pressed on the keyboard:

- mouse movement
- press on right button
- release right button
- press left button
- release left button
- press middle button
- release middle button
- double click

Sequences that are sent always include the current mouse position in **screen coordinates** or **virtual coordinates**. In order to limit the data exchange across connections, UNIX applications can request that Tun EMUL send only certain events.

In addition, the application can control the mouse in the following ways:

- display mouse
- hide mouse
- move mouse
- return mouse status and position in a specified format
- define the time interval in a "double-click"
- define the time interval for sending mouse movement
- return mouse status and position

Provided actions

Here is a description of the actions used for mouse management in Tun EMUL:

Action 277: Mouse initialization

This action has three parameters:

1. **p0** (in format %d) represents the events mask expected by the program

```
MOUSE_MOVE      0x01
LEFT_PRESS      0x02
LEFT_RELEASED   0x04
RIGHT_PRESS     0x08
```

```
RIGHT_RELEASED   0x10
CENTER_PRESS     0x20
CENTER_RELEASE   0x40
DOUBLE_CLICK     0x80
```

p1 (format %d) represents the time interval between mouse "reporting", expressed by the number of clock ticks (1 second = 18.2 ticks)

p2 (format %d) represents the length of time used by a double-click, expressed by the number of clock ticks (5 tends to be a good value)

Action 278 : Mouse close

Used by an application when it no longer needs the mouse, this action does not have parameters

Action 279 : Display mouse cursor

Used by an application to indicate that the mouse cursor should be displayed on the screen, this action does not have parameters.

Action 280 : Erase mouse cursor

Used by an application to indicate that the mouse cursor should no longer be displayed on the screen, this action does not have parameters.

Action 281 : Move mouse cursor

This action has two parameters:

p0 (format %d) represents the new "X" position of the cursor (in columns)

p1 (format %d) represents the new "Y" position of the cursor (in lines)

Action 282

Without parameters, this action returns the mouse status and position in the format specified by Action **284**.

Action 283

Following the action for mouse initialization, this action has one parameter:

p0 (format %d) represents what the application expects to receive:

```
EVENT_MOVE          0x01
EVENT_ALL           0x02
XY_PHYSICAL         0x04
XY_RELATIVE         0x08
LEFT_PANEL          0x10
RIGHT_PANEL         0x20
CENTER_PANEL        0x40
```

EVENT_MOVE Activates all the events defined by initialization

EVENT_ALL If (!EVENT_ALL) all events except mouse movement will be returned if one of the buttons is not pressed

XY_PHYSICAL The current cursor position will be returned in screen coordinates (virtual coordinates are the default)

XY_RELATIVE Returns the position of the mouse cursor in relative coordinates to its previous position

LEFT_PANEL To indicate that the left mouse button is reserved for the function-key panel

RIGHT_PANEL To indicate that the right mouse button is reserved for the function-key panel

CENTER_PANEL To indicate that the center mouse button is reserved for the function-key panel

Action 284 : Format definition of events expected by the application

Following the action for mouse initialization, this action has one parameter:

p0 is a character string in C format that indicates to Tun EMUL how to encode transmitted events. For example:

```
\033[ME%d;%d;%d
```

The first parameter in this string always represents the mouse status (see initialization constants). The following two parameters show the X and Y positions of the cursor.

By default, string format is:

```
%02x%03x%03x
```

Configuring mouse support

Mouse support has already been configured in ANSI emulation with the following lines added to ANSI.SEQ

```
\033Mi%p0%2d;%p1%2d;%p2%2dX p 277
\033Mc s 278
\033Md s 279
\033Mh s 280
\033Mm%p0%d;%p1%dX p 281
\033Mq s 282
\033Me%p0%dX p 283
\033Mf%p0%s p 284
```

Using a mouse is too complicated to simulate with a simple UNIX command or shell script. For this reason, C source code (MOUSE.C) is provided in the Tun EMUL directory. This file contains the programming that you will need to implement a mouse interface for UNIX. Try compiling it and using it under emulation.

CHAPTER 6 - MISCELLANEOUS SOLUTIONS

COLOR ATTRIBUTES IN EMULATION

To make emulation more "attractive", Tun EMUL offers the possibility of changing the colors of classic screen attributes such as highlighting, underlining, etc. In order to be able to use different colors, the **Initialization** line (the first line) of the appropriate **.SEQ** file needs to be modified.

Note	Another simpler solution would be to redefine the display settings (cf. "Defining Display Settings" section in Chapter 3)

The colors that can be used and their associated codes are:

```
0   :   black
1   :   blue
2   :   green
3   :   cyan
4   :   red
5   :   magenta
6   :   brown
7   :   light gray
8   :   dark gray
9   :   light blue
10  :   light green
11  :   light cyan
12  :   light red
13  :   light magenta
14  :   light brown (yellow)
15  :   white
```

Six different actions are used to control color selection:

action 30	color for normal video
action 31	color for reverse video
action 66	blinking color
action 67	underlining color (since VGA displays are unable to provide underlining in text mode)
action 68	highlight color
action 69	dim color

Action **30** is used with two parameters: the first determines character color; the second determines background color. For example, white characters on a blue background can be obtained using the parameters (7,1).

For color changes to be taken into account, enter the action number followed by the parameters between parentheses on the **Initialization** line of the **.SEQ** file, for example:

```
30 (7,1)
```

The same logic applies to action 31 (selection of reverse video color).

Action 66 is defined with only one hexadecimal parameter. If you would like a white blinking character with a blue background, you can include the sequence : **66(0x71)** on the **Initialization** line. The same applies to actions 67, 68, and 69.

These parameters (66, 67,68,69) are coded on one byte and are defined in hexadecimal notation. For example, if you would like a light green blinking character on a light magenta background, use the corresponding hexadecimal codes:

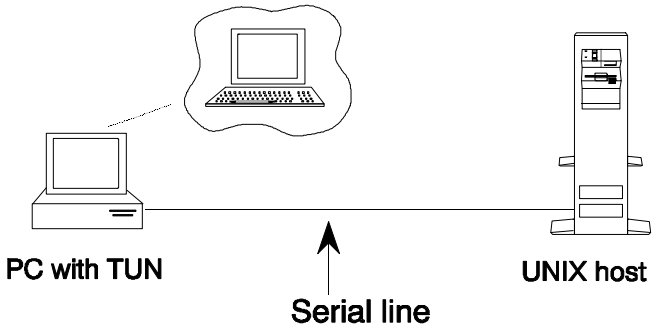
```
0   :   black
1   :   blue
2   :   green
3   :   cyan
4   :   red
5   :   magenta
6   :   brown
7   :   light gray
8   :   dark gray
9   :   light blue
A   :   light green
B   :   light cyan
C   :   light red
D   :   light magenta
E   :   light brown (yellow)
F   :   white
```

Sample sequence using this notation: 66(0xAD).

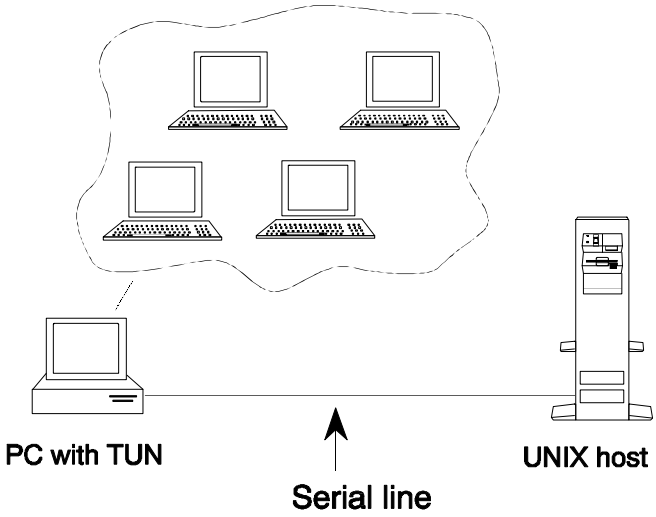
MULTI-SESSION ON SERIAL LINE (MSCREEN)

Usually Tun EMUL provides one session per physical connection when using RS232 (serial) communications. However, some UNIX systems offer the possibility of **multiplexing** sessions on a single line. On SCO UNIX and SCO XENIX systems the utility **mscreen** is used for this purpose.

The configuration:



is seen as:



To illustrate **mscreen** configuration, we will show the modifications necessary for 4 sessions on an ANSI terminal.

MSCREEN Setup

To be able to use Tun EMUL and mscreen together, the file **/etc/mscreencap** needs to be edited. The following lines need to be added to the end of the file:

```
ansi|ansimono:\
        :who,Ctrl-F7,\E[q,:\
        :help,Ctrl-F8,\E[r,:\
        :stop,Ctrl-F9,\E[s,:\
        :quit,Ctrl-F10,\E[t,:\
        :ttyp0,Ctrl-F1,\Ea,\EJ:\
        :ttyp1,Ctrl-F2,\Eb,\EK:\
        :ttyp2,Crtl-F3,\Ec,\EL:\
        :ttyp3,Ctrl-F4,\Ed,\EM:
```

These actions have the following meaning:

who	indicates the current session
help	displays a menu of available actions
stop and **quit**	stop multisession.

The last 4 lines contain the sequences used to manage multi-screen. For example **":ttyp0,Ctrl-F1,\Ea,\EJ:\"** :

ttyp0 refers to the first session (pseudo terminal)

Ctrl-F1 the key combination that will cause the emulator to switch to the first session

\Ea the sequence associated with Ctrl-F1 that is sent to the host machine to request the first session

\EJ the sequence sent by the host machine to the terminal to declare that the characters that follow belong to the first session

Modifications for SCO-XENIX

Files in Tun EMUL and on the XENIX host need to be edited for mscreen to work.

1. In our example, 4 lines need to be added to the end of the file/etc/ttytype:

```
ansi      ttyp0
ansi      ttyp1
ansi      ttyp2
ansi      ttyp3
```

ttyp0 - **ttyp3** correspond to each of the different pseudo terminals on which you wish to work. The line "ansi ttyp0" associates an ansi terminal to the session ttyp0.

2. For Tun EMUL to work with **mscreen**, certain modifications need to be made to the appropriate keyboard, function key, and escape sequence files.

Select **Parameters**, then **Keyboard Parameters**; and selecting the "ANSI.SEQ" keyboard, change the codes for function keys F1, F2, F3, and F4 in the CNTRL column as follows:

```
BASE ..............CNTRL
 fkey1                altpg1
 fkey2                altpg2
 fkey3                altpg3
 fkey4                altpg4
```

Select **Function key parameters**; near the end of the table describing the function key mnemonics, enter the following parameters:

```
\033a          for     altpg1
\033b          for     altpg2
\033c          for     altpg3
\033d          for     altpg4
```

Finally, the escape sequence file (.seq) file needs to be edited in order for Tun EMUL to be able to interpret the codes sent by the host machine. Select **Parameters**, then **Escape sequences**. Do not change the initialization or Sequence header lines, but add the following sequences and actions (for our example):

```
J p 8(0)

K p 8(1)

L p 8(2)

M p 8(3)
```

Action 8 is used to switch between virtual screens; the number in parentheses contains the number of the virtual screen.

Once all the changes have been made, you may need to issue the following commands on the host machine to activate the sessions:

```
enable ttyp0
enable ttyp1
enable ttyp2
enable ttyp3
```

Updating the SCO UNIX inittab

After changing /etc/mscreencap, the file /etc/inittab needs to be updated. Normally, the lines p0, p1, p2, and p3 will be set to "off":

```
p0:2:off:/etc/getty ttyp0 m
p1:2:off:/etc/getty ttyp1 m
p2:2:off:/etc/getty ttyp2 m
p3:2:off:/etc/getty ttyp3 m
```

They need to be changed from "off" to "respawn":

```
p0:2:respawn:/etc/getty ttyp0 m
p1:2:respawn:/etc/getty ttyp1 m
p2:2:respawn:/etc/getty ttyp2 m
p3:2:respawn:/etc/getty ttyp3 m
```

Once you have modified the inittab file, issue a **telinit q** for the changes to be taken into account by the host machine.

> **Note**: The same changes need to be made in Tun EMUL as were described for installation on XENIX.

Using mscreen

After entering emulation, execute the command **mscreen** to obtain multiple sessions. The following lines will be displayed:

```
who     is Ctrl-F7
help    is Ctrl-F8
stop    is Ctrl-F9
quit    is Ctrl-F10
ttyp0   is Ctrl-F1
ttyp1   is Ctrl-F2
ttyp2   is Ctrl-F3
ttyp3   is Ctrl-F4
```

Once this message is displayed, press <Ctrl><F1> to <Ctrl><F4> to switch between terminal sessions.

132 COLUMN EMULATION

132 Columns under MS-DOS

Most VGA and Super VGA cards support the use of 132 columns under MS-DOS. However, there is no standard and video card manufacturers have each defined their own BIOS mode for switching their cards to 132 x 25 mode.

If you would like to have 132 column emulation under MS-DOS, you must give Tun EMUL the hardware parameters for your card by filling in the BIOS mode field using **Setup ⇨ Hardware parameters**. Also update the display settings file (.CTX) to indicate that your configuration should use 132 columns.

BIOS MODE (hexadecimal)	**SVGA Card**
57	Paradise VGA with mono. monitor
55	Paradise VGA with color monitor
23	ATI VGA Wonder with color monitor
53	Trident 8900-C with color monitor
55	Orchid Fahrenheit 1280 with color monitor
23	ET4000 Tseng Labs
60	EVGA-8 451
57	NT-200B
57	SMART VGA
57	TVGA 9000 or 8900
57	PTI-223D
50	OAK

Setting number of columns in a .CTX file

Display settings files may be used to set the number of columns in an emulation session. You may use the following option:

Setup ⇨Display settings

Assigning 132 columns in the .SEQ file

For an emulation to always run with 132 columns, you may also change the current SEQ file. Modify the value of the parameter given to action 1 as follows:

1(4) 132 columns on a monochrome VGA screen

1(5) 132 columns on a VGA color screen.

Action "1" determines the video display at startup; usually the parameter would be:

1(3) 80 columns on a VGA color screen.

EMULATION WITH 25 LINES

Most emulations are defined by default to display 24 lines on the screen. The following action may be included in the initialization string in the ".SEQ" file to specify the number of lines used:

```
5(0,23)
```

Action 5 defines the screen margins: the first parameter (0) designates the upper margin; the second parameter (23) designates the lower margin.

This action may be found in the initialization string as well as elsewhere in the .SEQ file.

To obtain an emulation with 25 lines, add 1 to the second parameter every time action 5 is defined. If it is not present in the .SEQ file, insert it only in the initialization string. The following parameters will cause emulation to use 25 lines:

```
5(0,24)
```

For example, the file WYSE60.SEQ contains the following lines:

```
5(0,23) 62 72       (First line, initialization string)
\033
...
...
e( s 5(0,23)
e) s 5(0,24)
...
...
```

These need to be changed to:

```
5(0,24) 62 72       (First line, initialization string)
\033
...
...
e( s 5(0,24)
e) s 5(0,25)
...
...
```

FONT EDITOR

Tun EMUL is able to display 512 characters simultaneously under MS-DOS .

The PC's standard characters are used for the first 256 characters. For the second set of 256 characters, it is necessary to use an alternate font (described in Customizing Terminal Parameters).

For use under MS-DOS, fonts must be created using the font editor (FE.EXE) from ESKER. This editor is not delivered standard with Tun EMUL, but will be provided free of charge upon request.

> **Note**: FE.EXE can be used to create fonts for both MS-DOS and Windows, but to use them under Windows you will need to compile them with SDK.

SCANCODE EMULATION

Some word processing programs under UNIX (Word, WordPerfect) need to use more keys than are usually provided by ordinary terminals. These programs also need the <Alt> keys to send values.

To handle this problem, these programs recommend the use of scancode emulations in which all the keys on a keyboard simply send their scancode, and not several different values.

Tun EMUL supports scancode emulation with actions 152 and 153. In the standard ANSI emulation, these actions have been associated with the following escape sequences:

```
\033~5
\033~4
```

Using scancode mode

Follow these steps to use the emulator in scancode mode:

1. Switch the emulator to scancode mode by sending the sequence **\033~5**

2. Change the tty on the UNIX host using this command:

```
stty isscancode xscancode
```

To switch back to Tun EMUL's native mode:

1. Send the sequence \033~4
2. Change the tty on the UNIX machine back to "normal" with the command:

```
stty -isscancode -xscancode
```

USING COM3 AND COM4

Only COM1 and COM2 are completely standard on PCs. It is possible to add two additional COM ports (COM3 and COM4) with proper definition of the IRQ and I/O addresses.

COM3 and COM4 are defined using the options **Setup ⇨ Hardware parameters** in TUNEMUL.EXE.

Usually, ports COM3 and COM4 use the same IRQ as COM1 and COM 2, but have different I/O addresses (COM3=3E8 and COM4=2E8).

DEFINING MODEM COMMANDS

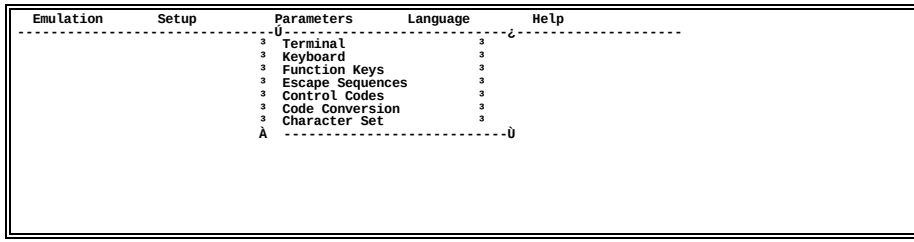
Tun EMUL uses the most standard Hayes commands for dialing modems. If these command are not sufficient for properly using your modem, use **Setup ⇨ Hardware** parameters to define the codes that are required.

CHAPTER 7 - TERMINAL CUSTOMIZATION

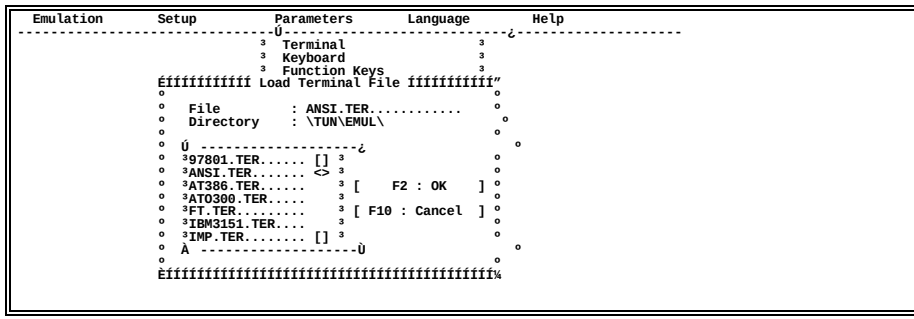
THE CONCEPT OF EMULATION

Given the wide variety of requirements for terminal emulators, Tun EMUL was designed so that users can define and customize every aspect of an emulation, including keyboards, escape sequences, and character tables.

Terminal customization is performed using TUNEMUL.EXE. You can view and modify the parameters of an emulation by selecting the Parameters option from the main menu.



After you select **Parameters** ⇨ **Terminal**, another window presenting the list of available terminals is displayed:

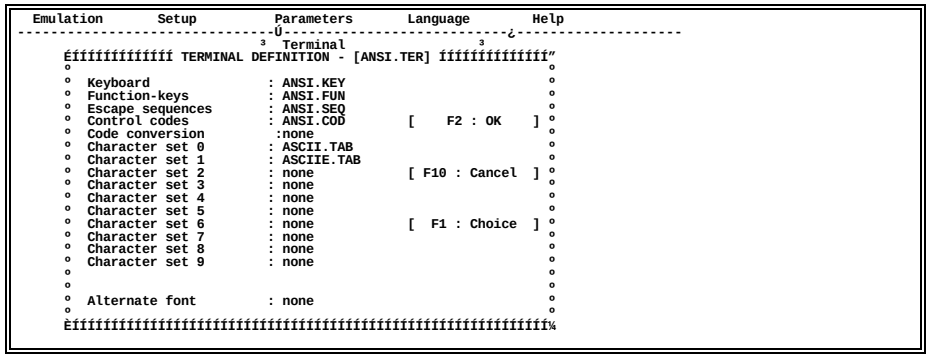


You may scroll through the list by using the mouse to activate the scrollbars, or by using the <Up> and <Down> arrows. When the desired terminal is highlighted, click on <Ok> or press <F2> to accept.

At this point you may also type a new name in the File field to create a new terminal.

COMPOSITION OF A TERMINAL

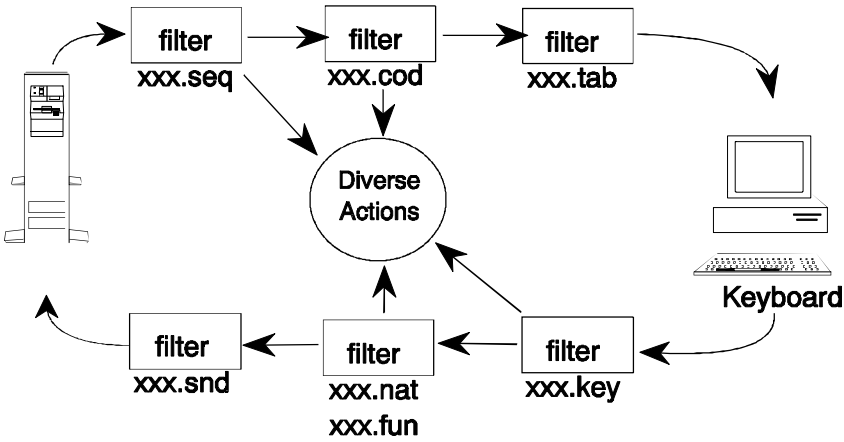
After you select a particular terminal (xxx.ter), the following screen appears:



This is the terminal definition screen, showing the different elements that make up an emulation; each file corresponds to a different step in controlling the exchange of data between a PC and a host machine.

Tun EMUL uses files with predefined suffixes (.key, .nat, .snd, .fun, .seq, .cod, .tab) to define the way in which the terminal emulator reacts in order to perform these functions.

Here is a diagram showing how Tun EMUL handles emulation:



Keyboard

The **.key**, **.nat** and **.fun** files handle all aspects of a PC's keyboard. They determine the character or action that is sent by a particular keystroke. These files are easy to create and modify using editors furnished by Tun EMUL.

Character conversion

Some emulations have certain special characters (extended ASCII) that need to be converted before they can be sent correctly to the host machine. The **.snd** files are provided by Tun EMUL to accomplish this task.

Escape sequences

The **.seq** files correlate character sequences sent by the host to numbered actions pre-defined by Tun EMUL. These actions (such as clearing the screen, positioning the cursor) are described in a file called **ACTION.ENG**.

Control codes

The **.cod** files define the control characters (whose decimal value lies between 0 and 31, or 128 and 159) that perform certain actions (skip a line, carriage return).

Character tables

The **.tab** files are the tables that determine which character Tun EMUL displays on the screen for a given byte. Each emulation has at least two transcoding tables of this type (the first one for characters from 0 to 127, the second one for characters from 128 to 255).

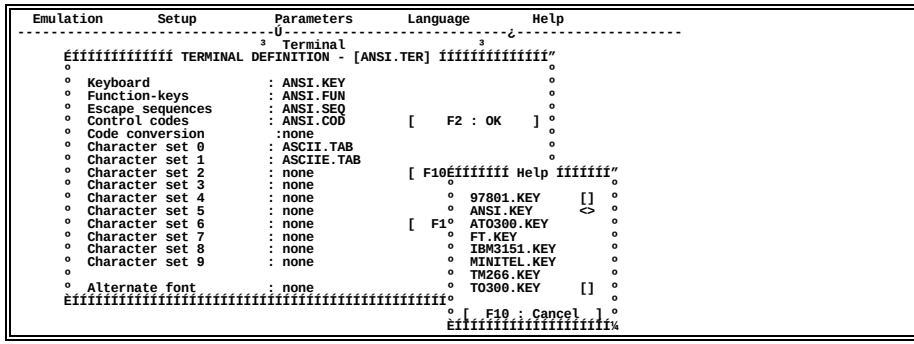
Alternative font

The emulation modules in Tun EMUL support the simultaneous display of 2 x 256 characters. The first 256 characters are read from the traditional PC character set (ASCII or extended ASCII CP437). The other 256 characters are contained in an alternative font that may be edited using a font editor that is available upon request from ESKER.

The .key and .fun files tend to be rather easy to understand and customize, whereas the .seq, .cod, and .tab files are slightly more complicated. The .nat and .snd usually do not need to be changed.

Selecting terminal components

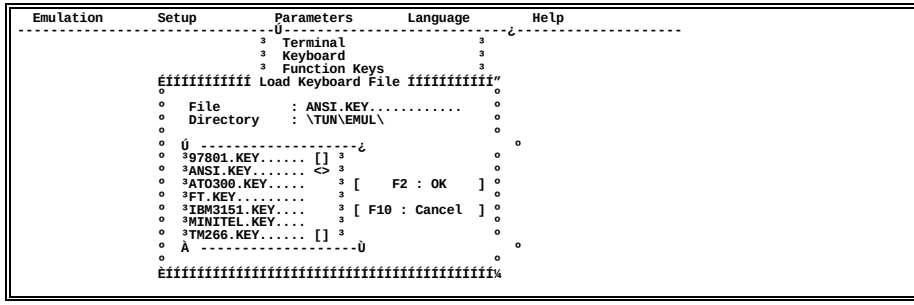
In the terminal definition screen, the available files for every field may be obtained in a help window by pressing <F1> or clicking on <Help>. For example:



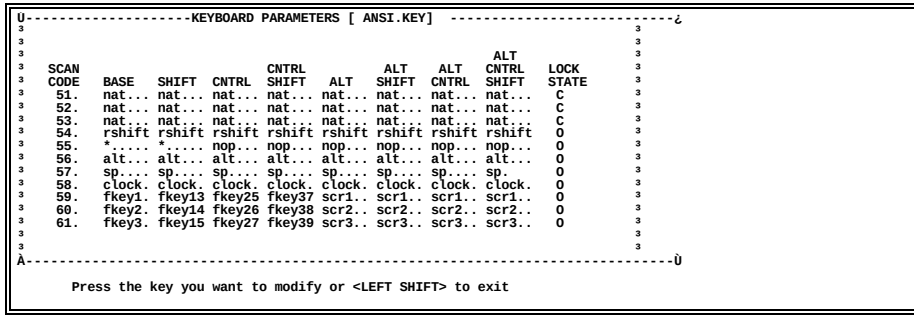
> **Note**: Usually the default terminal composition does not need to be changed. However, if you customize any of the configuration files, this screen should be used to update the terminal definition.

DEFINING KEYBOARD FILES UNDER MS-DOS (.KEY)

Keyboard definition is managed with the options **Parameters** ⇨ **Keyboard**. The available keyboards are then displayed in the following window:



You may scroll through the list of available keyboard files (.KEY) in the list using the mouse or the <Up> and <Down> arrows. After selecting a particular keyboard, such as ANSI.KEY, the following screen appears:



All the physical keys on a keyboard are identified by a unique scancode value. For example, scancode 54 is the <Right Shift> key; scancode 57 is the <Spacebar>, etc.

For any given scancode, there are 8 possible key combinations:

```
BASE
SHIFT
CNTRL
CNTRL SHIFT
ALT
ALT SHIFT
ALT CNTRL
ALT CNTRL SHIFT
```

The last column in the table defines the Lock State (the key's reaction to <Caps Lock> and <Num Lock>.

Any position in the table may contain the following:

- a reference to a .NAT file (described further)
- a character to send
- a chain of characters to send
- a function-key mnemonic (character string)
- an action to carry out
- a dead key (non-operational)

Key definitions can be easily modified using this editor. By pressing a key, the cursor automatically places itself on the line corresponding to its scancode. You can then define the nine fields or keep the current value by pressing <Return>.

Press <F2> to record any changes. When the cursor is in the table, you can return to the preceding field by pressing the up arrow. If you wish to quit without saving your changes, press left <Shift>, then <F10>.

When the cursor is at the end of a line, the program waits for you to choose another key, or to press the left <Shift> key to exit.

With the keyboard editor, you may redefine all of the keys on the keyboard as necessary.

Note:	The entry nat indicates that the codes sent by the key are contained in the active .NAT file.

Sending a single character

The following coding options are available for defining a key to emit a single character:

1. enter the character directly (if it is printable, as long as the character is not a number or a space):

   example:

   ```
   a          B          &
   ```

2. place it between quotes:

   example:

   ```
   'a'        '1'        '?'
   ```

   exceptions:

   ' is coded '\"
   \ is coded '\\' or \\

3. give the character's decimal value between 0 and 255

   example:

   ```
   203        48         53
   ```

4. give the hexadecimal value (0 to FF, preceded by 0x:

   example:

   ```
   0x0a       0x30       0x4f
   ```

5. give the octal value (0 to 0377) preceded by 0:

   example:

   ```
   013        047
   ```

The following mnemonics may also be entered for key definition:

Mnemonic	decimal	octal	hexadecimal	control
nul	0	00	0x00	^@
soh	1	01	0x01	^A
stx	2	02	0x02	^B
etx	3	03	0x03	^C
eot	4	04	0x04	^D
enq	5	05	0x05	^E
ack	6	06	0x06	^F
bel	7	07	0x07	^G
bs	8	010	0x08	^H
ht	9	011	0x09	^I
lf	10	12	0x0a	^J
nl	10	012	0x0a	^J
vt	11	013	0x0b	^K
ff	12	014	0x0c	^L
np	12	014	0x0c	^L
cr	13	015	0x0d	^M
so	14	016	0x0e	^N
si	15	017	0x0f	^O
dle	16	020	0x10	^P
dc1	17	021	0x11	^Q
dc2	18	022	0x12	^R
dc3	19	023	0x13	^S
dc4	20	024	0x14	^T
nak	21	025	0x15	^U
syn	22	026	0x16	^V
etb	23	027	0x17	^W
can	24	030	0x18	^X
em	25	031	0x19	^Y
sub	26	032	0x1a	^Z
esc	27	033	0x1b	^[
fs	28	034	0x1c	^\
gs	29	035	0x1d	^]
rs	30	036	0x1e	^^
us	31	037	0x1f	^_

Mnemonic	decimal	octal	hexadecimal
sp	32	040	0x20
del	127	0177	0x7f
ind	132	0204	0x84
nel	133	0205	0x85
ssa	134	0206	0x86
esa	135	0207	0x87
hts	136	0210	0x88
htj	137	0211	0x89
vts	138	0212	0x8a
pld	139	0213	0x8b
plu	140	0214	0x8c
ri	141	0215	0x8d
ss2	142	0216	0x8e
ss3	143	0217	0x8f
pu1	145	0221	0x91
pu2	146	0222	0x92
sts	147	0223	0x93
cch	148	0224	0x94
mw	149	0225	0x95
spa	150	0226	0x96
epa	151	0227	0x97
csi	155	0233	0x9b
st	156	0234	0x9c
osc	157	0235	0x9d
pm	158	0236	0x9e
apc	159	0237	0x9f

Sending a character string

If you want a key to emit a character string, enter them (in the correct order) on the desired key using the keyboard editor. If a character is not printable, use decimal, hexadecimal, or octal notation, preceded by the character \.

Examples of character strings:

```
aef   a\033be        a\0x08b        \32i\10
```

Other mnemonics specific to Tun EMUL are available for defining a key to directly prompt an action:

nop	no action
lshift	activation of left shift key
rshift	activation of right shift key
ctrl	activation of control (Ctrl) key
alt	activation of Alternative (Alt) key
clock	activation of Caps Lock key
nlock	activation of Num Lock key
slock	activation of Scroll Lock key
cal0.....cal9	successive pressing of numerical keys to obtain the corresponding decimal code (such as <Alt> 1-2-3 in MS-DOS)
hdcopy	Hard Copy of the screen
scr1	switch to session 1
scr2	switch to session 2
scr3	switch to session 3
scr4	switch to session 4
altpg1...8	switch to the specified page (on multi-page terminals)
nscr	switch to the next session
rdos	switch to a MS-DOS session (Alt-F5)
cemul	command execution window (Alt-F6)
emis	send MS-DOS file (Alt-F7)
recu	begin reception of a MS-DOS file (Alt-F8)
frecu	end reception of a MS-DOS file (Alt-F9)
brk	send a break signal to the host
femul	end of emulation (Alt-F10)
dossh	switch to MS-DOS (freeing memory (Alt-F12)
dos, atdos	returns keyboard control to BIOS (<Alt><Tab> Dosshell)

Function-key mnemonics

When codes or character strings are too long to be assigned directly on the keyboard, it is best to define a key using a "function-key mnemonic". Instead of entering the code, a mnemonic referring to a **function key file** is used.

Tun EMUL provides a set of sixty function-keys, from "fkey1" to "fkey60", defined in the fun files.

To assign a function-key mnemonic to a particular key, enter the name (fkey1, fkey2, etc.) in the keyboard file, and then assign the proper codes in the **.fun** file. See **Function Key Mnemonics** in the next section for more details.

If you wish to define a "dead key" (for accented characters entered with two different keystrokes : û, ë, ä, etc), use one of the following mnemonics :

Mnemonic	**Example**
aigu	é
grave	è
cflexe	â
tilde	ñ
trema	ï
rond	Å
cedilla	ç
barr	¢

Lock state

The last field of each line is LOCK STATE, indicating a key's reaction when the **clock** key (Caps Lock) or the **nlock** key (Num Lock) is activated. There are three possibilities:

O	no influence
C	considered as **shifted** if **clock** is on
N	considered as **shifted** if **nlock** is on

In general, keys corresponding to single characters have their LOCK STATE set at **C**, the numerical keys at **N**, and the rest of the keys at **O**.

NATIONAL KEYBOARDS (.NAT)

In Tun EMUL, .NAT files contain keyboard characteristics specific to each language supported by Tun EMUL. The key-word "**nat**" in a .KEY file refers to the scancode of a particular key in the current .NAT file. For example, here is part of the national keyboard for France (FR.NAT):

```
  2      &     '1'    nop    nop    nop    nop    nop    nop
  3      é     '2'    nop    nop     ~      ~      ~     nop
  4      "     '3'    nop    nop     #      #      #     nop
  5    '\''    '4'    nop    nop     {      {      {     nop
  6      (     '5'    nop    nop     [      [      [     nop
  7      -     '6'    nop    nop     |      |      |     nop
  8      è     '7'    nop    nop     `      `      `     nop
  9      _     '8'    nop    nop   '\\'   '\\'   '\\'    nop
 10      ç     '9'    nop    nop     ^      ^      ^     nop
 11      à     '0'    nop    nop     @      @      @     nop
 26 cflexe  trema    esc    esc    nop    nop    nop    esc
 27      $       £     gs     gs    nop    nop    nop     gs
  .
  .

 47      v       V    syn    syn    nop    nop    nop    nop
 48      b       B    stx    stx    nop    nop    nop    nop
 49      n       N     so     so    nop    nop    nop    nop
 50      ,       ?    del    del    nop    nop    nop    nop
 51      ;       .    nop    nop    nop    nop    nop    nop
 52      :       /    nop    nop    nop    nop    nop    nop
 53      !     nop    nop    nop    nop    nop    nop    nop
 86      <       >    nop    nop    nop    nop    nop    nop
```

The .NAT files provide characters that are specific to each language, whereas the .KEY files contain the codes and characters necessary to the emulation itself. For example, the function-keys on a VT220 are the same in any language, but the accents and symbols may vary greatly.

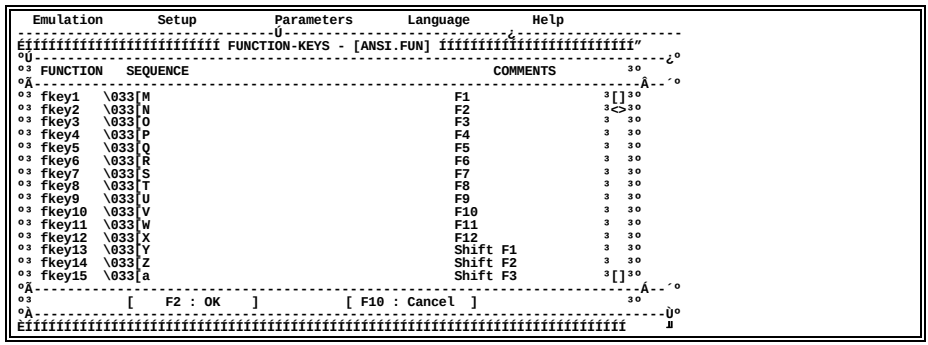
In the above table, the line corresponding to scancode 52 is referred to in the **.KEY** file by the key-word **nat**. Looking in the **.NAT** file to see what characters are actually sent, we see the character "**:**" in Base; the character "/" in Shift; the keyword **nop** (non-operational) for the other key combinations indicates that nothing is sent.

With this system, only the .NAT file may need to be modified for languages with different accented characters.

FUNCTION KEY MNEMONICS (.FUN)

The function-key mnemonics used in the **.key** files may be edited with **Parameters** ⇨ **Function keys**.

The **.fun** files contain the sixty different function-key mnemonics (fkey1 through fkey60). After you press <Enter>, a function-key table is displayed, such as the one shown in the figure below:



There are three columns in **.fun** files:

FUNCTION Function key mnemonic referenced in the **.key** file

SEQUENCE Character string to send

COMMENTS Optional field used to describe the function.

Only the SEQUENCE and COMMENTS fields may be edited.

You may use the mouse to place the cursor in the field you wish to edit; pressing <Enter> advances the cursor one field at a time; the <Up> and <Down> arrows and <PgUp> and <PgDn> keys may be used to scroll line-by-line through **fkey mnemonics**.

The rules for encoding sequences on function-key mnemonics are the same as for the **.key** files.

ESCAPE SEQUENCES (.SEQ)

Tun EMUL uses SEQ files to interpret the flow of data from the host machine. The .SEQ files associate one or more **actions** (cursor movement, clear screen...) with the reception of character strings (typically called **escape sequences**).

Escape sequence files may need editing:

1. if the current initialization strings are not appropriate,
2. to change the action associated with a particular escape sequence.

Documentation on Tun EMUL Actions

The actions supported by the emulator are documented in several text files:

***.ACT** brief descriptions of Tun EMUL actions in several different languages (DEUTSCH.ACT, ENGLISH.ACT, ESPAGNOL.ACT, FRANCAIS.ACT, ITALIANO.ACT); available in help screens while defining **escape sequence** files

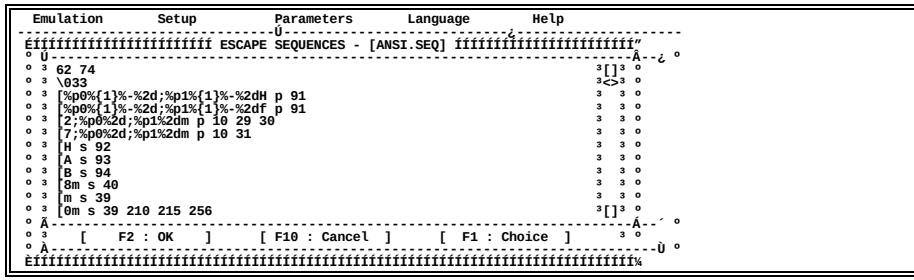
ACTION.ENG a detailed list of Tun EMUL actions in English

ACTION.DOC a detailed list of Tun EMUL actions in French.

These files will help you tremendously when programming an emulation.

> **Note**: If an emulation has been customized or newly defined, the **File Reception** feature (with <Alt><F8> and <Alt><F9>) will help you to capture and analyze escape sequences and display characters sent by the host (using a Debug utility).

Load a .SEQ file by selecting **Parameters** ⇨ **Escape sequences** in the Tun EMUL menus:



Escape sequence files are composed of three parts:

1. An initialization sequence (the first line)
2. A sequence header (the second line)
3. Defined escape sequences (all remaining lines)

The screen shown above serves as a text editor specially-designed for editing .SEQ files.

Terminal Initialization

The first line of an escape sequence file contains the list of actions necessary for the terminal to function properly. You may add or replace actions according to your needs (see following paragraphs for information on obtaining on-line help with escape sequences).

The **Initialization** line contains different actions separated by spaces. Parametered actions must be given with the appropriate parameters enclosed in parentheses and separated by a comma.
If there are a lot of actions it is possible to break up the initialisation sequence into several lines, ending each line except the last with the backslash character '\' (An example of this can been seen in the second line of WYSE60.SEQ);

For example, for initialization in VT220.SEQ:

```
195(0) 196(2) 197(2) 216
```

These actions are defined as follows:

ACTION	DESCRIPTION
195(0)	Assignment of character table 0 as G1
196(2)	Assignment of character table 2 as G2
197(2)	Assignment of character table 2 as G3
216	Lock G2 in GR

Sequence Headers

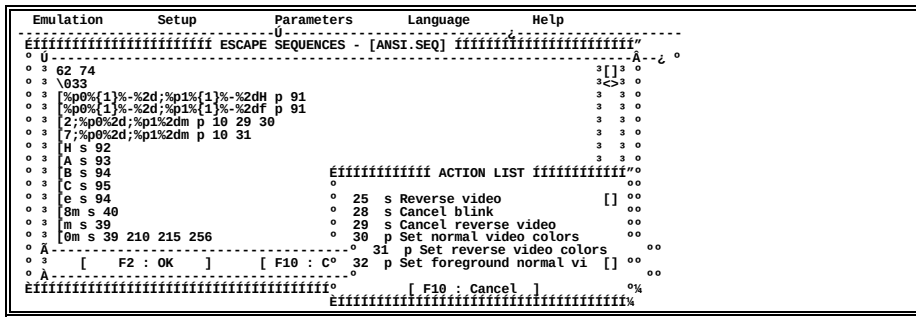
If all the sequences in an emulation begin with the same characters, it is best to enter them on the second line of the .SEQ file. This line serves as a header for every line that follows, and allows the emulator to treat sequences sent by the server more rapidly. The **escape character** (\033) is very often used as a header.

If you do not use a **Sequence Header**, you must leave the second line blank.

DEFINING ESCAPE SEQUENCES

The remaining fields define the actions that are related to a particular sequence. You may add sequences using TUNEMUL.EXE or any other standard text editor.

At any time you may open a help window describing the numbered actions by pressing the <F1> key:



The help feature is useful to see what actions are triggered by a particular escape sequence sent by the host. The help screen automatically opens the action contained in the current line.

For example, if you know that the host sends **\E[D** (to accomplish something, but you do not know what), you can place the cursor on the line containing **[D** in the .SEQ file, then press <F1>. The **Action List** is opened to action number 96, which is used to move the cursor to the left.

> **Note**: You can use the <Alt><F8> and <Alt><F9> functions (**Receive a file**) to capture the sequences sent on the line by the host computer, in order to analyze them.

There are two types of sequences:

1. simple sequences that do not change
2. parametered sequences that may vary

A **simple** sequence is a character string that does not contain a variable zone, and may be directly associated with one or more actions. For example, here is a string of three characters that cause the cursor to move one position to the left:

```
\E[D s 96
```

A **parametered** sequence is composed of a succession of strings beginning with the character **%**, which serves to identify the presence of a variable. A sequence may contain several parameters, defined in three parts:

1. definition of the parameters themselves
2. calculations and controls to be carried out on the parameter
3. parameter format

> **Note**: In the examples that follow, **[]** indicates an optional interval.

Defining parameters

Parameter definition uses the following syntax:

%[?value by default]p[0-9] allocation of a parameter

example %?1p2 third parameter with a default value = 1

%[?value by default]pi allocation of several parameters

example %?1pi

%g[a-z] allocation of a variable

example %gh allocation of the variable h

Calculations and controls

These sequences are arranged in Reverse Polish Notation (RPN).

%[min,max] control of the contents in an interval

example %[0x40,0x7f] the variable must be between 0x40 and 0x7f

%'c' to stack a constant

example %'b'

%"string" to stack a character string

example %"green"

%{nn} to stack a decimal constant

example %{64}

%g[a-z] to unstack a variable

example %gh

%P[a-z] to stack a variable

example %Ph

%V to stack the vertical position of the cursor

%H to stack the horizontal position of the cursor

%+ addition

%- subtraction

%* multiplication

%/ division

%m modulo

%& And "bit by bit"

%| Or "bit by bit"

%^ Xor "bit by bit"

%= Identity

%> Greater than

%< Less than

%A Logical And

%O Logical Or

%! Logical Not

%~ Not "bit by bit"

%I Inversion of bits : (01100010 becomes 01000110)

Parameter format

%c	single character
%s	character string delimited by " or '
%S(string)	character string ending by **string**. **string** is not stacked, and must be less than 10 characters. The decimal, hexadecimal and octal notations must begin with the character \. The character **)** must not be used within **string**, and must be coded \0x29.

> **Note:** **%S()** represents a character string ended by the first received character.

% [[:]flag] [dim[.precision]][type]

flag can have the values - + or # :

- the result is centered to the left
+ the result always includes a sign + or -

Blank if the first character of a conversion with a sign does not have a sign, a space precedes the result. This implies that if the flags **blank** and **+** are listed, the blank flag is not taken into account.

this flag means that the value has to be converted to a format depending upon the type of the corresponding argument. This flag has no effect on the type d. In the case of a conversion of type o, it raises the precision in such a way as to force the first digit of the result to 0. In the case of a conversion of type x or X, a result other than zero is prefixed as 0x or 0X.

dim gives the minimum number of characters occurring in the parameter. If this dimension begins with '0', the number is padded on the left by zeros and not blanks.

precision indicates the required number of digits (and not characters) corresponding to the parameter.

type can have the following values :

d a signed decimal is converted into a integer

o an octal notation is converted into an integer

x a non-signed hexadecimal is converted into an integer (use the lower-case letters a, b, c, d, e and f).

X a non-signed hexadecimal is converted into an integer (use the upper-case letters A, B, C, D, E and F).

Examples

The following examples illustrate this notation:

1. Encoding reverse video colors in "ANSI.SEQ":

```
\033[7;%p0%2d;%p1%2dm
```

Two parameters are present in this sequence:

```
%p0%2d and %p1%2d
```

In these two parameters, no calculations are to be carried out. The parameters are a succession of digits indicating an integer. The number of minimum characters per parameter is two. The first sequence is affected in parameter 0, the second in parameter 1.

2. Encoding the cursor address in **MINITEL.SEQ**:

```
\0x1f%p0%[64,127]%{64}%-%c%p1%[64,127]%{64}%-%c
```

Two parameters are present in this sequence **:**

```
%p0%[64,127]%{64}%-%c and %p1%[64,127]%{64}%-%c
```

The first sequence is in parameter 0 (p0), the second in parameter 1 (p1). These two sequences are isolated characters (%c). The sequences are treated as follows:

%[64,127]	control that the character is included between the decimal values 64 and 127
%{64}	stack the value 64
%-	subtraction in reverse Polish notation : char 64 - is equivalent to char - 64

After noting a sequence from the help window, or entering your own sequence, you must indicate whether the sequence is **simple** or **parametered**.

If the selected actions have parameters, they are used in order p0, p1, p2, etc. Note that a simple sequence can generate parametered actions as long as the given parameters are constant: example 1(3).

Some operations can be performed on the parameter by the action itself. Example : **31(-30)[30,37]** :

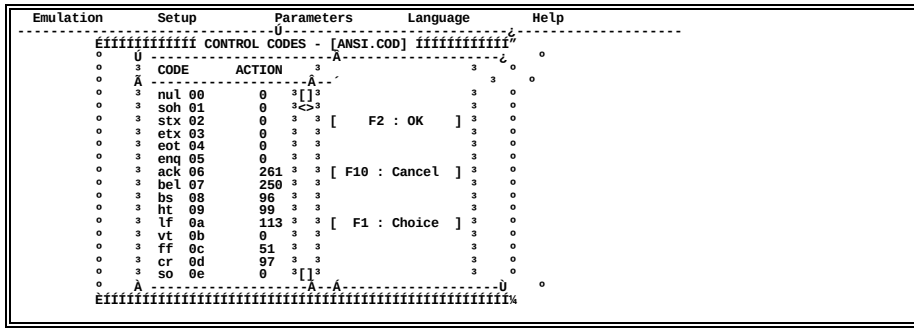
1. Checks if the parameter belongs to the interval [30,37]. If it does not, the action is not carried out.

2. Subtracts 30 from the parameter value before it is used by the action 31.

CONTROL CODES (.COD)

Characters whose decimal values range from 0 to 31, and from 128 to 159 are called control codes. Control codes often directly trigger particular actions. In Tun, control codes are configured in files with a **.COD** extension.

To modify a control code files, select **Parameters** ⇨ **Control Codes**. Here is an example showing ANSI.COD:

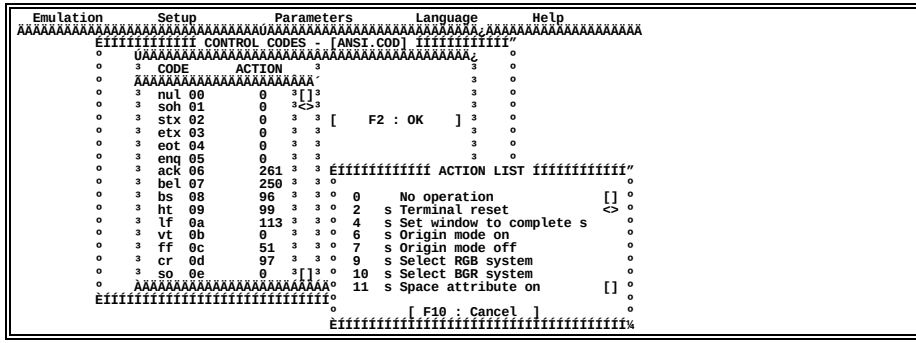


There are three possibilities for each ACTION field :

1. Leave the field blank (In this case, the character is displayed on the terminal).

2. Respond with 0 (In this case, no action is carried out, and the character is not listed).

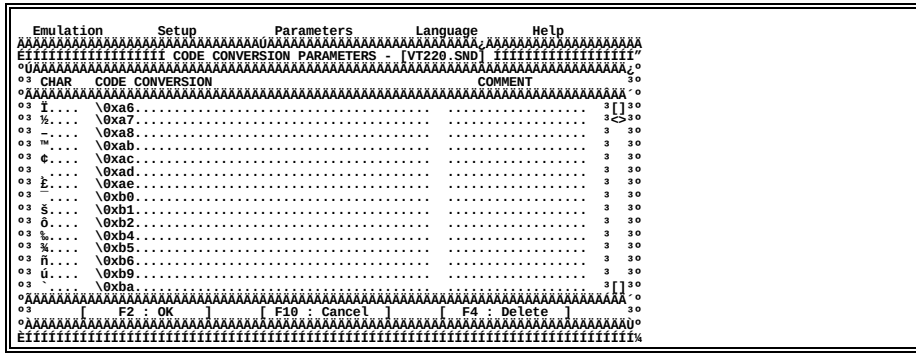
3. Select an action (Actions can be obtained be clicking on <Choice> or pressing <F1>.



A control code can only execute a simple action. Pressing <PgDn> passes to the next window, where the codes range from 128 to 159 (these codes are only for 8-bit protocols).

CODE CONVERSION (.SND)

In some emulations, ASCII characters need to be converted before they can be sent as correctly. This conversion takes place in the .SND files. In the following sample of VT220.SND, the column on the left contains the ASCII character, and the column on the right shows the code that will really be sent to the host machine:



CHARACTER TABLES (.TAB)

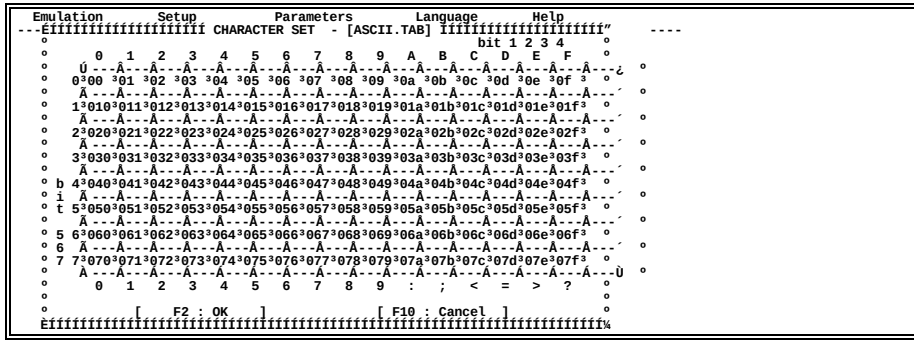
The character tables act as filters for displaying characters on the screen; an 8-bit character has 256 possible values. IBM compatible micro-computers have their own representation of these 256 characters. Certain characters are standard : 65 is represented as 'A', 66 as 'B', 48 as 'O', etc... Other characters, such as control characters, have a particular representation for IBM. Therefore, many different terminals possess different character sets.

The object is to define the tables for character representation. A representation is always defined for 7 bits, that is, from 0 to 127.

The tables **ASCII.TAB** and **ASCIIE.TAB** (ASCII and the extended ASCII character set) correspond to the representation of the micro-computer codes 0 to 127 for ASCII.TAB and 128 to 255 for ASCIIE.TAB. Other tables are :

UK.TAB	Britain
DECSU.TAB	DEC supplementary
DECSP.TAB	DEC special graphics

The **Character table editor** allows you to define other tables. To do this, select **Character Tables** from the **Parameters** menu, and then a ".TAB" file (in our example ASCII.TAB). The following screen appears :



This table consists of 128 fields corresponding to the 128 possible arrangements of 7 bits. The Horizontal axis shows the first four bits and the vertical axis shows the last three bits. Hexadecimal codes are entered in this table.

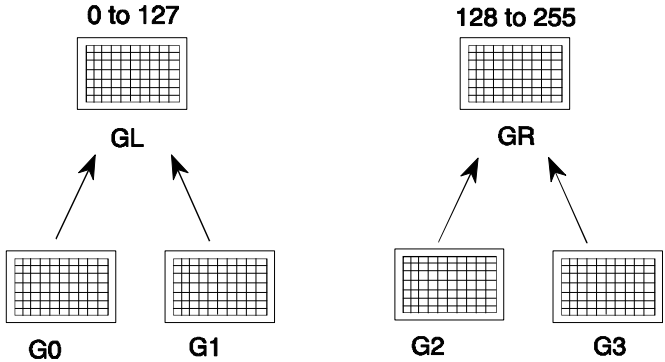
You may use the following command to display the ASCII and extended ASCII sets:

```
C:\TUNEMUL> TABLE
```

If you want to change a particular character, locate it first in the ASCII table, read its hexadecimal code, then copy it into the ".tab" file.

Internal character table management

To manage these tables during an emulation session, Tun EMUL uses a model based on VT100 emulation. Several tables are present in the memory in a VT100, but only 4 tables are available at any given moment:



By default, one of the tables G0, G1, G2, G3 is loaded into GL and GR. GL represents all the characters from 0 to 127, and GR, the characters from 128 to 255.

There are four Tun EMUL actions relating to this procedure:

```
194   p      assigns a character table to G0
195   p      assigns a character table to G1
196   p      assigns a character table to G2
197   p      assigns a character table to G3
```

These actions are defined by a table number corresponding to those indicated in the terminal definition file (".ter").

Eight Tun EMUL actions allow you to fill GL and GR:

```
210   s load G0 into GL
211   s load G1 into GL
212   s load G2 into GL
213   s load G3 into GL
214   s load G0 into GR
215   s load G1 into GR
216   s load G2 into GR
217   s load G3 into GR
```

Finally, four other actions permit you to access the next character in a table G0 G1 G2 or G3 without using GL or GR:

```
218   s      selective use of G0
219   s      selective use of G1
220   s      selective use of G2
221   s      selective use of G3
```

This organization of 4 active tables (two of which are available by default) is complex. Most emulations possess two permanent tables (GL and GR). The configuration file allows you to use 10 alternative tables. Loading one of these tables into GR or GL is achieved in the following way:

Example : 194(4) 214

Loads the 5th table into G0, then locks G0 into GR.

This organization should allow you to set the parameters to match virtually all existing terminal emulations.

Alternate character font

By default, PCs are only able to display 256 characters simultaneously. This limit poses some problems when trying to create emulations for more complex terminals that offer four or five different fonts.

Under MS-DOS with a VGA or SVGA card, or under Windows, Tun EMUL is capable of displaying 2 x 256 characters simultaneously by using an **alternate font**.

Use the font editor (FE.EXE) delivered with Tun EMUL to create additional fonts if you need to. For now, the only alternate font shipped standard is MINI14.DXF.

For a .TAB file to be able to use an alternate font, simply place the desired hexadecimal value by the number **1**.

For example, the value 182 refers to the 130th position (82nd in hexadecimal) of the alternate font.

CHAPTER 8 - PROGRAM SYNTAX

LIST OF Tun EMUL COMMANDS

EMUL	Asynchronous emulation for MS-DOS
TABLE	Character table viewer
TUNEMUL	Configuration module for MS-DOS

EMUL.EXE EMUL.EXE

Emul.exe starts emulation under MS-DOS.

Syntax

```
EMUL  [-S][-m][-noled] [config.cfg]
      [-a keyb_del][-p page_code]
```

or

```
EMUL  [-S] [-m] [-noled] [config.cfg]
      [-a keyb_del]
      [-p page_code]
      [-l language]
      [-C dos_command]
      [-D [dos_command]]
      [-H[host_command]]
      [-B [-s] batch_file]        [-M in_macro]
      [-Q out_macro]
      [-cga]
```

Description

When executed without the arguments **"config.cfg"** and **"language"**, the emulator reads the current information in the ACTIF.TUN EMUL file.

-S Does not display 1 2 3 4 5 ....(silent...)

-m To start Tun EMUL on a monochrome VGA monitor.

-a keyb_del Inserts a wait-state between keystrokes, so as not to overflow the TELNET buffer. For example, **EMUL CONF1 -a 30** sets the number of key repetitions-per-second to 30 (while continually pressing a key). The accepted values range between 20 and 300.

-cga Allows the use of the machine's BIOS cga to avoid any snow effect which might arise with some video cards.

-p page_code By default the installation procedure calculates the values of **dead keys** using page 437. You must use this option if you use another page code (850, 852, 863...).

-noled If you use an IBM 55 SX, IBM 65 SX, VICTOR 386 DSX, or DELL 386D computer, use this option so that the keyboard is not blocked by the use of the CapsLock and NumLock keys.

-l Indicates the language used during a terminal emulation session (English, German, French, etc.). This option must be followed by the name of the ".LG" file (ENGLISH.LG, ESPANOL.LG, FRANCAIS.LG, DEUTSCH.LG, etc.).

-C Executes the MS-DOS command following this option after the starting and initializing the emulation. "-C" must be followed by a MS-DOS program name, with its own options and parameters.

You can switch between the MS-DOS program and the terminal emulation by pressing <ALT-F5>. When you exit the MS-DOS program, the emulation is closed. This option lets the user associate an MS-DOS application with the Tun EMUL emulator.

-D Is the same option as "-C", but the end of the MS-DOS program does not terminate the terminal emulation.

If the MS-DOS program has no "-D" option, the command interpreter (COMMAND.COM) is automatically executed.

-H By starting the emulator, the command corresponding to this option is automatically sent to the host computer. Without this command, a <RETURN> is sent in order to display the prompt.

Note: Options "-D", "-C" and "-H" are mutually exclusive. If you supply a second file name in the command line, without the "-D", "-C", or "-H" option, the emulator is used in batch mode. This second file must contain commands for the file transfer (RLOGIN, RCOPY, etc...).

-s When used followed by a batch file, this option prevents the running operations from being displayed.

-M Followed by the macro name. This macro is executed when emulation is run.

-Q This option has to be followed by a macro name that will be executed when quitting emulation.

Example

To starts the emulator and the MS-DOS program "MP" (Multiplan), and provide access to the host machine with <Alt><F5>, you could issue this command:

```
EMUL CONF1 -C MP FILE-MP
```

See also

TUNEMUL.EXE

TABLE.EXE TABLE.EXE

TABLE is a utility delivered with Tun EMUL to makes it easier to define character tables (with ".tab" extensions).

Syntax

```
TABLE [-x] [-o] [-d] [".tab file"]
```

Description

"**-x**", "**-o**" and "**-d**" are mutually exclusive. Started without parameters, TABLE displays characters 0 to 127 and 128 to 255.

-x	displays characters in hexadecimal
-o	displays characters in octal
-d	displays characters in decimal

tab-file	Optional. If used, the suffix ".tab" does not need to be specified.

TUNEMUL.EXE TUNEMUL.EXE

TUNEMUL.EXE is the configuration program for Tun EMUL under MS-DOS.

Syntax

```
TUNEMUL [-m] [-noled] [config.cfg [-b batch-file]]
```

Description

-m To start Tun EMUL on a monochrome VGA monitor.

-noled If you use a IBM 55 SX, IBM 65 SX, VICTOR 386 DSX, or DELL 386D computer, use this option so that the keyboard is not blocked by the use of the CAPSLOCK and NumLock keys.

config.cfg is a configuration file (".cfg" file). The suffix is optional. If you use this argument Tun EMUL automatically starts the emulator using the specified emulation configuration.

-b must be followed by "batch file". This option is used with a file "file-config", which contains commands for file transfer (RLOGIN, RCOPY, etc...)

See also

EMUL.EXE

CHAPTER 9 - INTERNAL COMMANDS

The following commands may be used in batch files as EMULWIN or EMUL parameters. They may also be used during terminal emulation by pressing <Alt><F6> for Command Execution.

MACRO	Macro execution
RCOPY	File transfer within emulation
RDIAL	Telephone dialing
REXEC	Remote command execution
RHANGUP	Disconnects a modem connection
RLOGIN	Automatic login

MACRO MACRO

Executes a macro.

Syntax

```
MACRO macro_file
```

Description

This command may be used to execute a macro while in an emulation session.

macro_file the name of the .MAC file containing the macro instructions.

RCOPY RCOPY

This command is used to copy files between different volumes, either UNIX or MS-DOS.

Syntax

```
RCOPY [-r] [device:]source [device:][dest]
```

Description

-r is used for the transfer of binary files. No conversion is performed, and the files are transmitted byte for byte. If this option is not used, the UNIX "Line feed" (10 or 0x0a) character is replaced by the MS-DOS "Line feed" and "Carriage return" (13 or 0x0d) characters, and vice versa.

device: represents the source and destination drives. Depending on your configuration, the following drives are available:

A: first floppy drive on the MS-DOS PC
B: second floppy drive on the MS-DOS PC
C: first hard disk drive on the MS-DOS PC
D: second hard disk on MS-DOS PC
E: third MS-DOS hard disk
X: UNIX file system corresponding to the first emulation session
Y: UNIX file system corresponding to the second emulation session
Z: UNIX file system corresponding to the third emulation session
W: UNIX file system corresponding to the fourth emulation session.

If the "device:" argument is not specified, the default volume will be the current drive and directory in the emulation session. It is possible to copy files between all existing drives, except from X: to X:, Y: to Y:, etc.

source This must be specified, as it indicates the files to copy. Possibilities for source files are:

- file names
- wildcard characters * or ? to copy several files at once
- directories (subdirectories will also be copied)

dest indicates the destination; if absent, the destination will be the current directory on the destination volume. If you are copying only one file, you may enter a file name as the destination.

RDIAL RDIAL

This command dials a telephone number for an emulation session.

Syntax

```
RDIAL [Telephone_number]
```

Description

You must enter the telephone number the first time you start a session, otherwise the last number given is used. Commas (,) contained in the number generate a two-second pause during dialing. This command works on Hayes-compatible modems.

See also

RHANGUP

REXEC REXEC

Executes a remote command.

Syntax

```
REXEC UNIX-command
```

Description

This instruction executes a UNIX command, and then waits for completion.

RHANGUP RHANGUP

Disconnects a telephone connection.

Syntax

```
RHANGUP
```

Description

Used in emulation over modem, this command immediately cuts off telephone communication

See also

RDIAL

RLOGIN RLOGIN

Automatically begins a connection.

Syntax

```
RLOGIN [Y:]user_name [password]
```

Description

This command allows you to log in to another UNIX host.

Y: represents a session on COM2. When used without this argument, RLOGIN assumes a connection on COM1 (except if the current volume is Y:)

user_name is the account name to use for the connection

password is only necessary if a password has been defined for the user

CHAPTER 10 - MACRO LANGUAGE SYNTAX

MACRO LANGUAGE REFERENCE

Index of Macro Language instructions:

GetVar Assignment to the last string received (by a Receive instruction) of a variable

Goto Jump to a label

Hide Characters received by the emulator are not displayed

HideMessage Dialog field is not displayed

HidePanel Erases current key panel

IfConnected Tests to see of a connection is still active (TCP/IP, modem)

IfEqual Tests to see if the last string received is equal to a certain value

IfError Tests the status of the last Receive command

IfNoEqual Compares the last string received to a value for inequality

IfNoError Tests the status of the last Receive command

Label Defines a label

LoadPanel Loads a function-key panel

NoExitIfDisconnect
Inhibits ExitIfDisconnect (default setting)

Pause Waits a specified period of time

RcopyGet File transfer from UNIX to MS-DOS

RcopyPut File transfer from MS-DOS to UNIX

Rcoupe Disconnects modem communication

ReadVar Assigns a character string typed on the keyboard to a variable

ReadPasswd	Assigns a character string typed on the keyboard to a variable without displaying it on the screen
Repeat	Starts a block of reiterative instructions
Receive	Waits to receive one or more strings
Return	Unconditional exit from macro and return to the emulator
Rdial	Dialing using a Hayes-compatible modem
Rhangup	Ends modem communication
Rnumero	Dialing using a Hayes-compatible modem
Send	Sends a character string over the communication channel
SendAndReceive	Sends a character string over the communication channel and waits to receive one or more character strings.
SessionTitle	Assigns a name to the emulation session
Set	Defines and assigns a variable
SetHelpFile	Defines a help file
SetVar	Assigns the last string received by a "Receive" type command to a variable.
ShowMessage	Displays the dialog field
ShowPanel	Displays a function-key panel
Sleep	Waits a specified period of time
Title	Assigns a title to a dialog box

BREAK BREAK

Unconditional exit from a loop.

Syntax

```
Break
```

Description

This instruction forces the interpreter to jump to the instruction that immediately follows the current loop. This instruction functions identically to a loop in C language.

A loop is bordered by the *Repeat* and *End* instructions. A break is generally triggered by a test.

Example

```
Repeat 5
      Receive 1 "login:"
      IfNoError break
      Send "\n"
End
```

See also

End, Continue, Repeat, Goto, Label

CHANGETERMINAL CHANGETERMINAL

Instant change of terminal type.

Syntax

```
ChangeTerminal string
```

Description

This instruction allows you to change the terminal type during an emulation session. ChangeTerminal takes the information from the "*string*.TER" file to load the new terminal; the .TER file must be located in the directory where Tun EMUL was installed.

Example

```
Label BEGIN
      ReadVar "Terminal type : " TERM
      ChangeTerminal "%TERM"
      IfNoError OK
      Echo -c "Unknown Terminal type ! "
      Sleep 2
      Goto BEGIN
Label OK
```

CLEARMESSAGE CLEARMESSAGE

Erases messages contained in a dialog box. (Only in Tun EMUL for Windows)

Syntax

```
ClearMessage
```

Description

This instruction erases the contents of a dialog box.

See also

HideMessage, ShowMessage

CLEARSCREEN CLEARSCREEN

Clears the screen. (Only in Tun EMUL for Windows)

Syntax

```
ClearScreen
```

Description

This instruction clears the emulation screen.

CLOSEPANEL CLOSEPANEL

Erases and unloads the current key panel from memory.

Syntax

```
ClosePanel
```

Description

This instruction erases the current function-key panel from the screen, and removes it from memory.

See also

LoadPanel, HidePanel, ShowPanel

DISPLAY DISPLAY

Displays characters received by the emulator

Syntax

```
Display
```

Description

This instruction causes characters received on the communication channel to be displayed on the screen, even if they are filtered by a *Receive* command. This instruction is useful for viewing the progression of a connection when writing a macro.

See also

Hide

DOS DOS

Executes a MS-DOS command.

Syntax

```
Dos DOS_command
```

Description

This instruction executes the MS-DOS command *DOS_command.*

Example

```
Dos "COPY C:\\TUN\\EMUL\\TRANSFER.TXT A:"
```

ECHO ECHO

Displays character strings in the dialog box.

Syntax

```
Echo [-c|-r] string
```

Description

This instruction displays the string *"string*" in the dialog field. It may be used to keep the user informed of the status of the connection or transfer.

When used without options, the string is aligned on the left side; the option **-c** centers the string; the option **-r** justifies the string on the right.

See also

ReadVar, ShowMessage, HideMessage

END END

Marks the end of a reiterative block

Syntax

```
End
```

Description

This instruction ends a group of repeating instructions beginning with the *Repeat* instruction.

Example

```
Repeat 6
     instruction
     instruction
     instruction
     ...
End
```

See also

Repeat, Break, Continue

EXIT EXIT

Unconditional exit from the macro as well as the emulator.

Syntax

```
Exit
```

Description

This instruction triggers an exit from both the macro interpreter and the emulator. On TCP/IP, the current session will be closed when this function is completed.

See also

Return

EXITIFDISCONNECT EXITIFDISCONNECT

Triggers an exit from emulation in case disconnection is detected.

Syntax

```
ExitIfDisconnect
```

Description

This instruction initiates an exit from the emulator running on TCP/IP if disconnection is detected. It is generally used when an emulation session passes through several phases of connection (RTC, PAD, login, application,..).

See also

NoExitIfDisconnect

GETVAR GETVAR

Assignment of a variable to the last string received (by a Receive instruction).

Syntax

```
GetVar variable
```

Description

This instruction reads the contents of the "*variable*" variable and assigns it to the **last string received** by a *Receive* or *SendAndReceive* command. In general, the designated variable was assigned previously by the *SetVar* instruction.

This instruction can be used to store the "*Receive*" function results in memory for later testing.

Example

```
Receive 2 "login" "password" "#"
SetVar TOTO
Receive 2 "xxxx" "yyyy" "zzzz"
Receive 2 "abcde" "fghij" "klmno"
GetVar TOTO
IfEqual "login" exit
IfEqual "password" return
```

See also

SetVar, Set, ReadVar

GOTO GOTO

Unconditional jump to a label.

Syntax

```
Goto label
```

Description

This instruction causes the interpreter to jump to the "*Label*" instruction corresponding to the desired label.

GoTo and **Label** may be used to create loops.

If the label specified is not found in the program, an error message is displayed and the emulator will be exited.

Example

```
Label BEGIN
Receive 2 "xxxx" "yyyy" "zzzz"
Receive 2 "abcde" "fghij" "klmno"
IfEqual "abcde" exit
IfEqual "fghij" return
Goto BEGIN
```

See also

Label, Repeat, Break, End, Continue

HIDE HIDE

Causes characters received by the emulator to not be displayed

Syntax

```
Hide
```

Description

This instruction serves to prevent display of the characters received over the communication channel. It also prevents the user from modifying the session.

This instruction can be used to prevent the end-user from seeing the connection details (especially if connection is particularly complex). Using this instruction, users may dialog with their usual **UNIX** applications transparently.

See also

Display

HIDEMESSAGE HIDEMESSAGE

Prevents dialog from being displayed. (Only in Tun EMUL for Windows)

Syntax

```
HideMessage
```

Description

This instruction serves to prevent the macro dialog window from being displayed, and possibly concealing the emulation screen.

See also

ClearMessage, ShowMessage

HIDEPANEL HIDEPANEL

Erases the current function-key panel.

Syntax

```
HidePanel
```

Description

This command erases the function-key panel currently displayed on the screen.

See also

ClosePanel, LoadPanel, ShowPanel

IFCONNECTED IFCONNECTED

Tests to see if a connection is still active (TCP/IP and modem).

Syntax

```
IfConnected label|break|continue|exit|return
```

Description

This command tests for an active connection.

The parameter for this instruction specifies the action to be taken if connection is confirmed; below is a list of the possible parameters:

label	go to the specified label
Break	exit the current loop
Continue	repeat the current loop
Exit	quit the emulator
Return	quit the macro and return to the emulator

Example

```
Repeat 3
Rhangup
      Repeat 4
      Sleep 1
      IfConnected Continue
      Goto Next
      End
End
Echo "Connection error !"
Label Next
Return
```

See also

Rhangup, Rcoupe

IFERROR IFERROR

Tests the status of certain commands.

Syntax

```
IfError label|break|continue|exit|return
IfNoError label|break|continue|exit|return
```

Description

The "*IfError*" and "*IfNoError*" instructions test the return code of the last instruction executed. If the condition is verified, the action specified by the parameter is executed.

If none of the strings expected by these instructions was read within the specified time, an error will result.

An error occurs if "*Rdial*" or "*Rhangup*" did not function correctly (in text mode only).

An error also occurs if "*ChangeTerminal*", "*LoadPanel*", "*RcopyPut*", or *RcopyGet*" did not function correctly.

The parameter for this instruction specifies the action to be taken if no error is reported; below is a list of the possible parameters:

label	go to the specified label
Break	exit the current loop
Continue	repeat the current loop
Exit	quit the emulator
Return	quit the macro and return to the emulator

See also

IfEqual, Receive, SendAndReceive

IFEQUAL IFEQUAL

Tests to see if the last string received is equal to a certain value.

Syntax

```
IfEqual string label|break|continue|exit|return
IfNoEqual string label|break|continue|exit|return
```

Description

The "*IfEqual*" and "*IfNoEqual*" instructions test the equality or non-equality of the current string with the "*string*" character string. If the condition is verified, the action corresponding to the parameter is executed.

The current string is the last one received by the "*Receive*" and "*SendAndReceive*" commands. Its value may be assigned by the "*GetVar*" instruction.

The parameter (to be executed if successful) may take the following values :

label	go to the specified label

Break	exit the current loop
Continue	repeat the current loop
Exit	quit the emulator
Return	quit the macro and return to the emulator

See also

IfError, Receive, SendAndReceive, GetVar

LABEL LABEL

Defines a label.

Syntax

```
Label name
```

Description

This instruction defines a label within the macro. The label may be tagged by the "*Goto*" or "*IfError*" function.

See also

Goto

LOADPANEL LOADPANEL

Loads a function-key panel into memory.

Syntax

```
LoadPanel [-s] string
```

Description

This instruction loads the function-key panel specified in the character string "*string*". Using the **-s** option will load and display the function-key panel.

Example

```
ReadVar "Display function-key panel ?" REP
LoadPanel "c:\\tun\\emul\\function.pan"
GetVar REP
IfnoEqual "Y" NEXT
Showpanel
Label NEXT
```

See also

ClosePanel, HidePanel, ShowPanel

NOEXITIFDISCONNECT NOEXITIFDISCONNECT

Inhibits *ExitIfDisconnect*.

Syntax

```
NoExitIfDisconnect
```

Description

This command negates the effect of an *ExitIfDisconnect* command.

See also

ExitIfDisconnect

PAUSE PAUSE

Waits a specified period of time.

Syntax

```
Pause nbsec
```

Description

This command generates a pause for "*nbsec*" seconds.

RCOPYGET RCOPYGET

File transfer from UNIX to MS-DOS.

Syntax

```
RcopyGet [-b] [-t] [-size] src_file [dst_file]
```

Description

This command executes file transfer from UNIX to MS-DOS. The parameters "*src_file*" and "*dst_file*" are character strings. The option **-b** is used to transfer files in binary mode (no CR/LF conversion of LF to CR+LF). The option **-t** is uses only ASCII characters (32 to 127) to transfer files and to resolve problems posed when characters are filtered by the communication media. The *-size* option is used to specify the size of the packets used for the transfer.

Example

The command:

```
RcopyGet "/etc/inittab"
```

transfers the UNIX file "/etc/inittab" into the current directory on the MS-DOS PC. Conversion of LF to CR/LF is performed (because "-r" was not specified).

The command:

```
RcopyGet -b -t -126 "/usr/infor/data" "c:\\tmp\\dat"
```

will copy the UNIX file "/usr/infor/data" into the \TMP directory on the MS-DOS machine. No line-feed conversion is performed; characters are transcoded into ASCII format, and the packets contain 126 characters.

Note

File transfer is only possible if the user is properly connected to the UNIX machine, and currently has a UNIX prompt on the screen.

Wildcard characters ? and * may be used to transfer several files at once.

See also

RcopyPut

RCOPYPUT RCOPYPUT

Transfers files from MS-DOS to UNIX.

Syntax

```
RcopyPut [-b] [-t] [-size] src_file [dst_file]
```

Description

This command executes file transfer from UNIX to MS-DOS. The parameters "*src_file*" and "*dst_file*" are character strings. The option **-b** is used to transfer files in binary mode (no CR/LF conversion of LF to CR+LF). The option **-t** is uses only ASCII characters (32 to 127) to transfer files and resolve problems posed when characters are filtered by the communication media. The *-size* option is used to specify the size of the packets used for the transfer.

Example

The command:

```
RcopyPut "C:\\AUTOEXEC.BAT"
```

will copy the MS-DOS file "C:\AUTOEXEC.BAT" into the current directory on the UNIX machine. LF+CR to LF conversion is performed.

The command:

```
RcopyPut -b -t -126 "c:\\tun\\emul\\french.wlg" "/usr/data"
```

copies the MS-DOS file "\tun\french.wlg" into the /usr/data directory on the UNIX machine. No conversion is performed, characters are transferred in ASCII format, and the size of the data packets is 126 characters.

Note

File transfer is only possible if the user is properly connected to the UNIX machine, and currently has a UNIX prompt on the screen.

Wildcard characters ? and * may be used to transfer several files at once.

See also

RcopyGet

RCOUPE RCOUPE

Disconnects modem communication (Hayes-compatible).

Syntax

```
Rcoupe
```

Description

This command is used to disconnect a communication session over a modem.

> **Note**: In MS-Windows, the Error flag is not affected; proper disconnection should be managed using the *Ifconnected* command.

Example

```
Repeat 3
Rcoupe
      Repeat 4
      Sleep 1
      IfConnected Continue
      Goto Next
      End
End
Echo "Connection error !"
Label Next
Return
```

See also

Rdial, Rnumero, IfConnected, Rhangup

RDIAL RDIAL

For dialing using a Hayes-compatible modem.

Syntax

```
Rdial telephone_number
```

Description

This command will dial a telephone number on a Hayes-compatible modem.

A number may contain commas (,) in order to insert a two-second pause when dialing.

Example

The command:

```
Rdial "0,,11"
```

will connect the emulator to Minitel, the electronic phonebook in France.

> **Note:** For this feature to work, the modem field in the configuration file (*.CFG) must be set to indicate the presence of a modem.

See also

Rhangup

READVAR READVAR

Assigns a character string typed on the keyboard to a variable.

Syntax

```
ReadVar "message" variable
```

Description

This instruction displays the "*message*" message in the dialog field, then assigns the user's response to the "*variable*" variable.

This instruction can be used to request input from the user (i.e. login, etc.).

Example

```
ReadVar "Enter your login" LOGIN
Send "%LOGIN"
Receive 2 "# " "Password"
```

See also

Echo

READPASSWD READPASSWD

Assigns a string of characters entered on the keyboard to a variable, without displaying the characters typed.

Syntax

```
ReadPasswd message variable
```

Description

This instruction displays the "*message*" message in the dialog field, then assigns the user's response to the "*variable*" variable, without displaying the characters that were typed.

This instruction can be used to request input from the user (i.e. password, login, etc.).

Example

```
Send "root"
sleep 1
ReadPasswd "Enter your password" PASSWD
Send "%PASSWD"
Receive 2 "# " "login"
...
```

See also

Echo

REPEAT REPEAT

Starts of block of repetitive instructions.

Syntax

Repeat nbcount

Description

This instruction marks the beginning of a block of repetitive instructions. The whole number "*nbcount*" parameter specifies the number of times the block must be executed.

The end of the block must be marked by the "*End*" command.

As many "*Repeats*" may be embedded as needed.

Example

```
Repeat 4
      Send "toto"
      Send "titi"
      Receive 4 "root"
      IfNoError break
End
```

See also

End, Break, Continue

RECEIVE RECEIVE

Waits to receive one or more strings.

Syntax

```
Receive timeout istring [istring1][istring2] [...]
```

Description

This instruction sets the macro interpreter to *wait state*, until one of the "*istring*" strings is received over the communication channel.

If reception does not occur within the time specified by the "*timeout*" parameter, the function returns control to the interpreter and sets its return code to error.

If a string was received before the timeout elapsed, the function transfers control to the interpreter and sets its return code to "correct". Simultaneously, the string received is stored in the current string.

This instruction's return code may be tested using the "*IfError*" or "*IfNoError*" functions.

The current string may be tested using the "*IfEqual*" and "*IfNoEqual*" functions

If the "*timeout*" parameter is set to 0, the wait time will be infinite until one or more of the strings requested is received. If the emulator is in "*Display*" mode at this time, and if the dialog field is in "*HideMessage*" mode, the user may take over control via the emulator.

Example

```
Label BEGIN
Receive 2 "abcde" "fghij" "klmno"
IfEqual "abcde" exit
IfEqual "fghij" return
IfEqual "klmno" BEGIN
```

See also

Send, SendAndReceive

RETURN RETURN

Unconditional exit from macro and return to the emulator.

Syntax

```
Return
```

Description

This instruction stops macro interpretation and returns to the emulator (and clears the dialog field).

See also

Exit

RHANGUP RHANGUP

Ends modem communication

Syntax

```
Rhangup
```

Description

This command is used to disconnect a communication session over a modem (identical to Rcoupe).

> **Note**: In MS-Windows, the Error flag is not affected; proper disconnection should be managed using the *Ifconnected* command.

Example

```
Repeat 3
Rhangup
      Repeat 4
      Sleep 1
      IfConnected Continue
      Goto Next
      End
End
Echo "Error during disconnection !"
Label Next
Return
```

See also

Rdial, Rnumero, IfConnected, RCoupe

RNUMERO RNUMERO

Dialing using a Hayes-compatible modem (identical to Rdial).

Syntax

```
Rnumero telephone_number
```

Description

This command will dial a telephone number on a Hayes-compatible modem.

A number may contain commas (,) in order to insert a two-second pause when dialing.

Example

The command:

```
Rnumero "0,,11"
```

will connect the emulator to Minitel, the electronic phonebook in France.

> **Note**: For this feature to work, the modem field in the configuration file (*.CFG) must be set to indicate the presence of a modem.

See Also

Rcoupe, RDial

SEND SEND

Sends a character string over the communication channel.

Syntax

```
Send ostring
```

Description

This command sends the characters string "*ostring*" over the communication channel

See also

Receive, SendAndReceive

SENDANDRECEIVE SENDANDRECEIVE

Sends a character string over the communication channel and waits to receive one or more return character strings.

Syntax

```
SendAndReceive      [-nb] timeout ostring istring
                    [istring1] [..]
```

Description

This instruction is a combination of the "*Send*" and "*Receive*" instructions.

It sends the "*ostring*" string over the communication channel, then waits until an "*istring*" string is received.

If reception does not occur within the time specified by the "*timeout*" parameter, the function transfers control to the interpreter and sets its return code to "error".

If a string was received before the timeout elapsed, the function transfers control to the interpreter and sets its return code to "correct". Simultaneously, the string received is stored in the current string.

This instruction's return code may be tested using the "*IfError*" or "*IfNoError*" functions.

The current string may be tested using the "*IfEqual*" and "*IfNoEqual*" functions

If the "*timeout*" parameter is equal to 0, the wait time will be infinite until one or more of the strings requested is received. If the emulator is in "*Display*" mode at this time, and if the dialog field is in "*HideMessage*" mode, the user may take over control via the emulator.

See also

Receive, Send

SESSIONTITLE SESSIONTITLE

Assigns a name to the emulation session. (Only in Tun EMUL for Windows)

Syntax

```
SessionTitle string
```

Description

This command gives the name that will be used in the emulation menus for a particular session. The name is given by the character string "*string*".

SET SET

Defines and assigns a variable.

Syntax

```
Set variable string
```

Description

This command defines the variable "*variable*" and assigns to it the contents of the character string "*string*".

See also

GetVar, SetVar, ReadVar

SETHELPFILE SETHELPFILE

Designates a help file. (Only in Tun EMUL for Windows)

Syntax

```
SetHelpFile string
```

Description

This command is used to specify the help file to be used; the character string "*string*" contains the name of the help file.

Example

```
SetHelpFile "C:\\TUN\\EMUL\\minitel.hlp"
```

SETVAR SETVAR

Assigns the last string received by a Receive-type command to a variable.

Syntax

```
SetVar variable
```

Description

This instruction stores the current string in the "*variable*" variable. The current string is the last received by the *Receive* or *SendAndReceive* command. The results of the *Receive* and *SendAndReceive* functions cannot be assigned to a variable.

In general, the variable thus stored in memory may be reused using the "*GetVar*" function.

This instruction can be used to store the "*Receive*" function result for later testing.

Example

```
Receive 2 "login" "password" "#"
SetVar TOTO
Receive 2 "xxxx" "yyyy" "zzzz"
Receive 2 "abcde" "fghij" "klmno"
GetVar TOTO
IfEqual "login" exit
IfEqual "password" return
```

See also

Set, GetVar, ReadVar

SHOWMESSAGE SHOWMESSAGE

Displays the dialog field. (Only in Tun EMUL for Windows)

Syntax

```
ShowMessage
```

Description

This instruction forces the interpreter to display the macro's dialog field. It can also be used to return control to the user during a particular phase of the session (i.e. requesting password, a specific command, etc...). This instruction cancels the HideMessage instruction.

See also

ClearMessage, HideMessage

SHOWPANEL SHOWPANEL

Displays a function-key panel.

Syntax

```
ShowPanel
```

Description

This command displays a function-key panel previously loaded by *LoadPanel*; it cancels a *Hidepanel.*

See also

LoadPanel, HidePanel, ClosePanel

SLEEP SLEEP

Wait a specified period of time.

Syntax

```
Sleep nbsec
```

Description

This command generates a pause of "*nbsec*" seconds.

TITLE TITLE

Assigns a title to a dialog box.

Syntax

```
Title string
```

Description

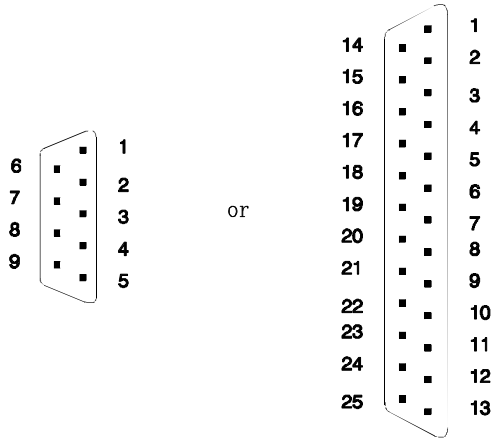
This command places the contents of the character string "*string*" as the title to a dialog box in Windows.

APPENDIX A - SERIAL CABLING

Many problems with serial connections are due to cabling errors. This section will help you construct the proper serial cable for your situation.

Overview of serial ports and cables

Serial ports on both PCs and servers generally consist of a 9 or 25-pin male connector, with the pins numbered as shown in the figure below:



Depending on the pin assignment of a serial (RS232) connector, computers are classified into two categories:

- terminals
- host machines

DTE/DCE

Serial ports may be configured as either **DTE** (Data Terminal Equipment) or **DCE** (Data Communication Equipment). The standard adopted by IBM identifies a port by the type of connector:

male	DTE configuration
female	DCE configuration

In general, PCs are configured as DTE, but they may be used either as terminals (running emulation software) or as host machines (running a multi-user operating system.

Pin assignment varies depending on whether the serial port is configured as DTE or DCE:

SIGNAL	**DTE 25 PINS**	**DCE 25 PINS**
Transmit Data (TD)	2	3
Receive Data (RD)	3	2
Request to send (RTS)	4	5
Clear to send (CTS)	5	4
Data station ready (DSR)	6	20
Ground (GRD)	7	7
Data Carrier Detection	8	8
(DCD)	20	6
Data terminal ready	22	22
(DTR)		
Recall indicator (RI)		

Minimum Cabling

In a direct connection between a PC running Tun EMUL and a host machine, at least three wires must be used:

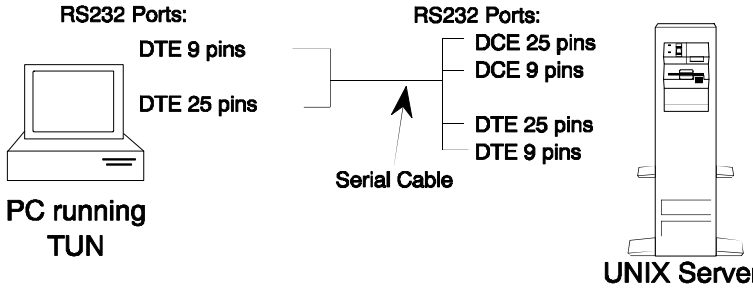
PC 25-pin male DTE		Host Machine 25-pin female
TD 2	⟷	2 RD
RD 3	⟷	3 TD
GRD 7	⟷	7 GRD

9-to-25 pin conversion

To correctly convert a 9-pin connector to a 25-pin connector, follow the diagram below:

9-pin female connector		25-pin male connector
1	⟷	8
2	⟷	3
3	⟷	2
4	⟷	20
5	⟷	7
6	⟷	6
7	⟷	4
8	⟷	5
9	⟷	22

The possible combinations of serial connections are summarized in the following figure:

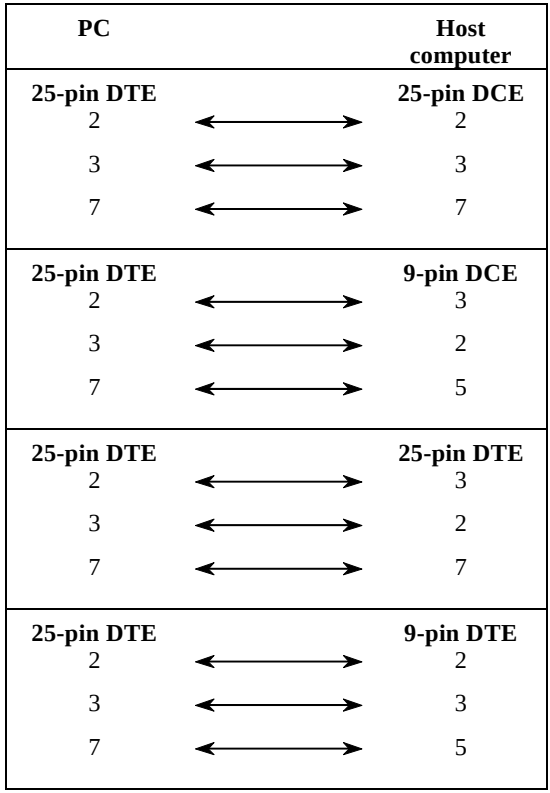


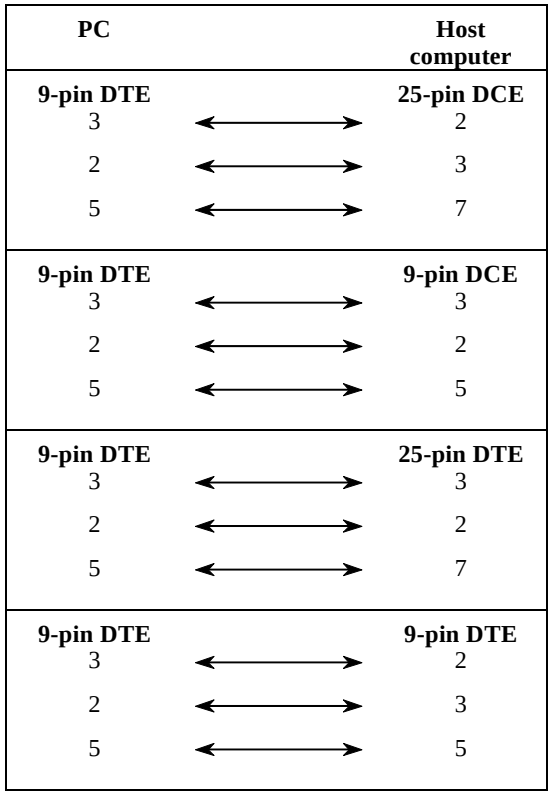
Note: In some cases, the serial ports on the host machine may be configured as DTE. This is the case for UNIX and XENIX operating systems that run on micro-computers.

Pins 2 and 3 need to be crossed when connecting the PC running Tun EMUL to this type of host machine (from DTE to DTE).

Cable possibilities

The different possible (minimum) cabling configurations are shown in the following diagrams:




PC
Host computer
9-pin DTE
25-pin DCE
3
2
2
3
5
7
9-pin DTE
9-pin DCE
3
3
2
2
5
5
9-pin DTE
25-pin DTE
3
3
2
2
5
7
9-pin DTE
9-pin DTE
3
2
2
3
5
5

APPENDIX B - MACRO EXAMPLES

The following macro automates connection, login, and automatic transfer of selected files (VT220.* and ANSI.SEQ) from the MS-DOS PC to the /tmp directory of the UNIX host.

```
# Characters sent by Host computer not displayed
Title "File Transfer"

# Start
Label BEGIN
# Read password
ShowMessage
ReadPasswd "Enter your password : " PASSWD
HideMessage

# Make connection
Repeat 3
# Send carriage-return character
      Repeat 5
             SendAndReceive 5 "\n" "ogin"
             IfNoError break
      end
      IfError NOCONNECTION

# Send login
      SendAndReceive 9 "root\n" "assword" "# "
      IfError continue
      IfEqual "# " break
      SendAndReceive 9 "%PASSWD\n" "# " "ogin:" "ERM"
      IfError continue
      IfEqual "# " break

# Return to start of program if login incorrect
      IfEqual "ogin:" BEGIN
# Set the TERM variable if necessary
      SendAndReceive 9 "\n" "# "
      IfError continue
      IfEqual "# " break
end
# Start application
Set FILE "VT220.*"

RcopyPut c:\\TUN\\EMUL\\VT220.* /tmp
IfError TRANSFERERROR
Set FILE "ANSI.SEQ"
RcopyPut c:\\TUN\\EMUL\\ANSI.SEQ /tmp
IfError TRANSFERERROR
```

```
# Transfer done
ShowMessage
ClearMessage
Echo -c "Transfer Done"
Sleep 3
# Exit from the emulator
Exit

#Transfer error
Label TRANSFERERROR
ShowMessage
Echo "Error Transferring %FILE"
ReadVar "Press Return to quit" ANSWER
# Exit the emulator
exit


# No login
Label NOCONNECTION
ShowMessage
Echo "Communication failed"
ReadVar "Press Return to quit" ANSWER
# Exit the emulator
exit
```

This next macro automates connection, login, and then starts the system maintenance utility "**sysadmsh**" on an SCO UNIX host:

```
# Characters sent by Host computer not displayed
Hide

# Start
Label BEGIN
# Read login and password
ReadVar "Enter your user name : " USER
ReadPasswd "Enter your password : " PASSWD

# Make connection
Repeat 3
# Send carriage-return character
      Repeat 5
             SendAndReceive 5 "\n" "ogin"
             IfNoError break
      end
      IfError NOCONNECTION
# Send login
      SendAndReceive 9 "%USER\n" "assword" "# "
      IfError continue
      IfEqual "# " break
      SendAndReceive 9 "%PASSWD\n" "# " "ogin:" "ERM"
      IfError continue
```

```
      IfEqual "# " break
# Return to start of program if login incorrect
      IfEqual "ogin:" BEGIN
# Set the TERM variable if necessary
      SendAndReceive 9 "\n" "# "
      IfError continue
      IfEqual "# " break
end
# Start application
#Send " sysadmsh \n"
Send "# Starting the program \033[35h\n"
# Display received characters
Display
# Return to the emulator
Return
# No login
Label NOCONNECTION
Echo "Communication failed"
ReadVar "Press Return to quit" ANSWER
# Exit the emulator
exit
```

Note: You may substitute the name of another UNIX application or shell script in place of "sysadmsh".

INDEX

H

I

K

L

M

N

P

R

S

T

U

W

X