USER
MANUAL

# Tun EMUL™

## Terminal Emulation for Windows

Version 8.50

ESKER
MAKING OPEN SYSTEMS A REALITY

# PREFACE

**Tun EMUL** is a complete terminal emulator package for the Windows environment. Unlike the DOS version, IBM 5250- and IBM 3270-type synchronous emulation is supported.

**Tun EMUL** is a complementary product to the **Tun KERNEL**, **Tun NET** and **Tun SQL** software range (see table below).

| | **WINDOWS** | **MS-DOS** |
|---|---|---|
| **Tun KERNEL** | TCP/IP protocol stacks for Windows 3.x only | TCP/IP protocol stacks for MS-DOS (TSR) |
| **Tun NET** | TCP/IP applications (NIS, NFS Client and Server, PING, Printer redirection and sharing, FTP Client and Server, TELNET, RSH Client and Server, TAR, WALL, TFTP, TIME), and electronic mail and fax application | TCP/IP applications for MS-DOS (NFS, Printer sharing, FTP, TELNET, TAR ...) |
| **Tun EMUL** | Comprehensive terminal emulator (asynchronous emulation, IBM 3270 IBM 5250) | Comprehensive terminal emulator for MS-DOS (asynchronous emulation) |
| **Tun SQL** | ODBC drivers for Client-Server mod under TCP/IP (Oracle, Informix, Sybase DBMS, Progress, DB2) and database revamping tools | N/A |
| **TCP/IP Network Services** | Browser NIS, Printer redirection and sharing (LPR, LPD) | N/A |

**Tun EMUL** is delivered as standard with the **Tun PLUS** package which incorporates all the above software.

**Tun EMUL** may be installed independently of **Tun PLUS**; in any case, the installation procedure of **Tun PLUS** automatically proposes the optional installation of**Tun EMUL**.

# TABLE OF CONTENTS

# PART I
# USER GUIDE

# CHAPTER 1 - INSTALLATION

## PACKAGE CONTENTS

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

- The **Tun EMUL** user manual (Terminal Emulation for Windows).
- The **TCP/IP Network Services** manual (NIS, printer sharing and redirection).
- The **Tun KERNEL** manual (TCP/IP Communication for Windows 3.x).
- A CD-ROM.
- A user license.
- A sealed envelope containing a serial number and an activation key.
- Miscellaneous technical bulletins (if applicable).
- A registration card.

**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 **Tun EMUL**.

## HARDWARE AND SOFTWARE REQUIREMENTS

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

- A 100% PC 486 compatible micro-computer or Pentium.
- 8-16 MB of RAM.
- Windows 3.x or 95.
- TCP/IP stack compatible with the Winsock interface (e.g. **Tun KERNEL** or other Winsock-compatible stacks)

## INSTALLATION

**The following instructions may be ignored if you have acquired Tun EMUL as part of the Tun PLUS package since the installation procedure of the latter product will automatically propose the installation of Tun EMUL.**

The following procedure describes the installation under Windows of the **Tun EMUL** package:

**1.** Insert the CD-ROM into the CD-ROM drive (generally, drive D).
Depending on your Windows version, follow one of these steps:

Under Windows 3.x or Windows NT3.51, choose one of the following methods:

- From the **Program Manager**, select **File➔Run** and type the following command:
  **D:\INSTALL.EXE**

- From the **File Manager**, open the drive containing the CD-ROM (drive D) et double-click on the file INSTALL.EXE.

Under Windows 95 or Windows NT4, choose one of the following methods:

- Click on the **Add/Remove Programs** icon in the **Control Panel** window. Follow the installation prompts that appear, using the file INSTALL.EXE on the CD-ROM.

- From a DOS prompt, type the following command:

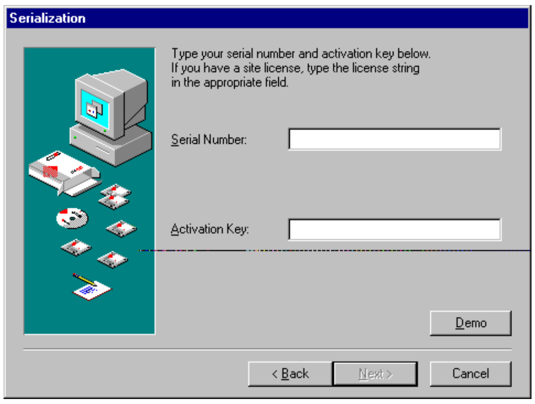
**D:\INSTALL.EXE**

- From the **Start** menu, select **Run** and type the above command.

- From the **Windows Explorer**, open the drive containing the CD-ROM (drive D) and double-click on the file INSTALL.EXE.

| | |
|---|---|
| Note: | If you are using the **Tun EMUL** installation disks instead of the CD-ROM, insert the disk labeled "Tun Setup Disk 1/2" into the disk drive (generally drive A), and follow the same procedure with the following command: **A:\SETUP.EXE** |

**2.** After the Welcome dialog box, you will see the following installation window.


Serialization

Type your serial number and activation key below.
If you have a site license, type the license string
in the appropriate field.

Serial Number:

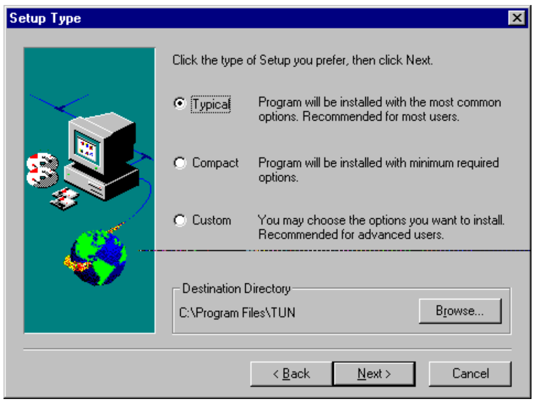
Activation Key:

Enter the serial number and the activation key of the software. This information can be found in the sealed envelope accompanying the software.

Click the button **Demo** if you wish to install the demonstration version of **Tun PLUS**. A serial number and activation key will be proposed for the demo installation.

Then click the button **Next**.

**3.** If the serial number and activation key are correct, the following window will appear:


Setup Type

Click the type of Setup you prefer, then click Next.

[✓ Typical] Program will be installed with the most common options. Recommended for most users.

Compact Program will be installed with minimum required options.

Custom You may choose the options you want to install. Recommended for advanced users.

Destination Directory
C:\Program Files\TUN

This dialog box lets you choose the type of installation you want and the installation directory.

### Type of installation

There are three types of installation:

- **Typical**: installs the necessary components for normal usage of **Tun EMUL**.
- **Compact**: installs the minimum number of components for **Tun EMUL** to function.
- **Custom**: allows you to choose the components you wish to install.

### Installation directory

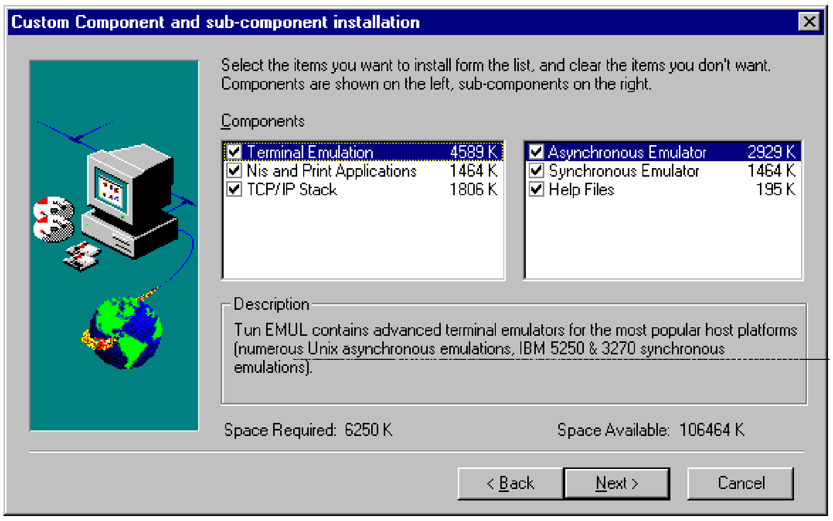
Win 3.X In Windows 3.x, the default directory for the installation of **Tun EMUL** is C:\TUN.

Win 95 In Windows 95, the default directory is C:\Program Files\TUN.

The directory may be changed by clicking the button **Browse...**.

Click the button **Next** when you are ready.

**4.** If you chose the custom installation, the following dialog box will appear:



Custom Component and sub-component installation

Select the items you want to install form the list, and clear the items you don't want. Components are shown on the left, sub-components on the right.

Components

| | |
|---|---|
| ✓ Terminal Emulation | 4589 K |
| ✓ Nis and Print Applications | 1464 K |
| ✓ TCP/IP Stack | 1806 K |

| | |
|---|---|
| ✓ Asynchronous Emulator | 2929 K |
| ✓ Synchronous Emulator | 1464 K |
| ✓ Help Files | 195 K |

Description

Tun EMUL contains advanced terminal emulators for the most popular host platforms (numerous Unix asynchronous emulations, IBM 5250 & 3270 synchronous emulations).

Space Required: 6250 K Space Available: 106464 K

Win 95 In Windows 95, the option **TCP/IP Stack** is not available, since **Tun EMUL** uses Microsoft's TCP/IP stack.

Select the check boxes corresponding to the components or subcomponents you wish to install.

### Terminal Emulation (Tun EMUL)

This component includes the following subcomponents

- Asynchronous Emulator : Top-rank asynchronous emulator (21 possible emulations).
- Synchronous Emulator : 5250 emulator (for the IBM AS/400), 3270 emulator (for IBM mainframes).
- Panel editor.
- Help Files : Terminal Emulation on-line help.

## NIS and Print Applications

This component includes three subcomponents:

- NIS Application: Network Information Service for centralized management of the network's resources.
- Print Applications: LPR printer sharing and LPD printer redirection.
- Help Files: On-line help for NIS, printer sharing and redirection.

For further information on the use of NIS, please refer to the user manual TCP/IP Network Services.

Win 3.X

## TCP/IP Stack (16-bit version only)

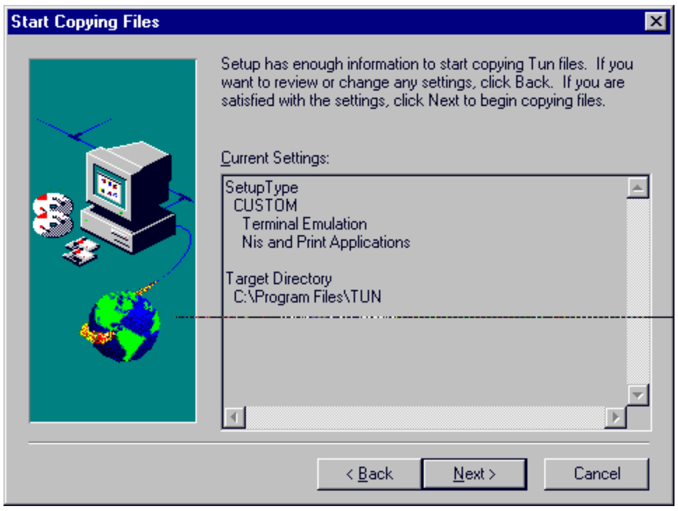
This component includes the following three subcomponents:

- TCP/IP Stack: DLLs et VxD drivers for TCP/IP stack.
- Network Card Driver: Packet Driver, NDIS Driver.
- Help Files: on-line help for the TCP/IP stack.

For more information on this component, please consult the manual **Tun KERNEL**

Then click the button **Next**.

**5.** The following window will appear.



Setup has enough information to start copying Tun files. If you want to review or change any settings, click Back. If you are satisfied with the settings, click Next to begin copying files.

Current Settings:

```
SetupType
  CUSTOM
    Terminal Emulation
    Nis and Print Applications

Target Directory
  C:\Program Files\TUN
```

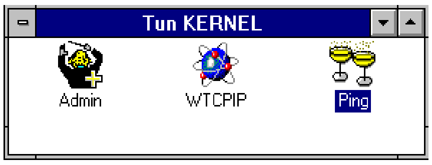
Then click the button **Next** if you are satisfied with the installation options.

Win 3.X

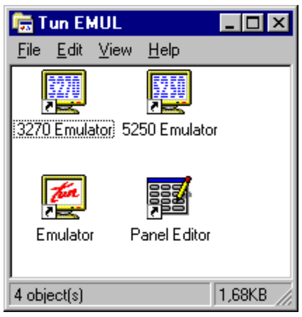
**6.** At the end of the installation process, if you requested the installation of the TCP/IP stacks, a window will appear proposing the immediate configuration of **Tun KERNEL**. If you agree to this option, the installation procedure switches immediately to the **Tun KERNEL** configuration screens as described in the chapter **Using and Configuring Tun KERNEL** in the **Tun KERNEL** manual. If you refuse this option, it will be possible to return to it later by clicking on the **Admin** icon in the **Tun KERNEL** or **Tun NET** groups under the Windows Program Manager.

Win 3.X

**7.** The installation process has now finished. For an installation under Windows 3.x, you should be able to see the new **Tun KERNEL** group with the following icons (if you chose to install it):



and also the new **Tun EMUL** group with these icons:



| | |
|---|---|
| **Note**: | If the **Tun NET** product has already been installed on the PC, the **Tun KERNEL** icons will be included in the g group. |

## DIRECTORY STRUCTURE AND INSTALLED FILES

The following files are installed in the default directory:

### Asynchronous emulator:

| **File** | **Description** |
|---|---|
| EMULWIN.EXE | Emulation and configuration module (Windows 16 bits) |
| EMUL32.EXE | Emulation and configuration module (Windows 32 bits) |
| PANEDIT.EXE | Function-key panel editor (Windows 16 bits) |
| PANED32.EXE | Function-key panel editor (Windows 32 bits) |
| PANEL.DLL | DLL Application (Windows 16 bits) |
| PANEL32.DLL | DLL Application (Windows 32 bits) |
| *.FON | Character font for Windows 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 |
| *.SES | Terminal configurations |
| *.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 |
| *.LG | Message files |
| *.MAC | Sample macros |
| *.HLP | Help files under Windows |
| RTUNPLUS and RTUNPLUS.C | UNIX executable (SCO, IX386) and C source-code for file transfer module |
| NVT.EXE | IPX interface |
| TRANSFILE.SH | UNIX file transfer shell script |
| TUNEMUL.INI | Default configuration file |
| MOUSE.C | Example of mouse programming (in C) |

### IBM 5250 emulation (synchronous)

| **File** | **Description** |
|---|---|
| 5250.EXE | 5250 emulation module (Windows 16 bits) |
| 5250_32.EXE | 5250 emulation module (Windows 32 bits) |
| 5250.INI | Default configuration file |
| *.KBD | International keyboard definitions |
| *.SET | International character set definitions |
| 5250*.LG | Message files |
| 5250*.HLP | Help files |

### IBM 3270 emulation (synchronous)

| File | Description |
|---|---|
| 3270.EXE | 3270 emulation module (Windows 16 bits) |
| 3270_32.EXE | 3270 emulation module (Windows 32 bits) |
| WHLLAPI.DLL | HLLAPI DLL Application (Windows 16 bits) |
| WHLL32.DLL | HLLAPI DLL Application (Windows 32 bits) |
| 3270.INI | Default configuration file |
| *.KBD | International keyboard definitions |
| *.SET | International character set definitions |
| 3270*.LG | Message files |
| 3270*.HLP | Help files |

## INSTALLING THE 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 micro-computers 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. |
|---|---|

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 login to the host system as the user "root".
2. Execute the following commands:

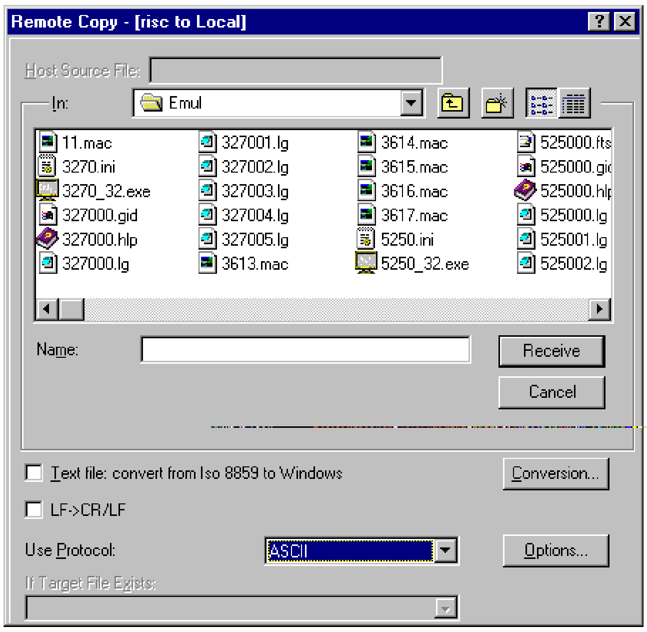
```
# cd /usr/bin
# stty -echo;cat >rtunplus.c;stty echo
```

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 local source code file you wish to send. In this case, type

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

and check that the protocol used is **ASCII**.



Remote Copy - [risc to Local]

Host Source File:

In: Emul

| | | | |
|---|---|---|---|
| 11.mac | 327001.lg | 3614.mac | 525000.fts |
| 3270.ini | 327002.lg | 3615.mac | 525000.gid |
| 3270_32.exe | 327003.lg | 3616.mac | 525000.hlp |
| 327000.gid | 327004.lg | 3617.mac | 525000.lg |
| 327000.hlp | 327005.lg | 5250.ini | 525001.lg |
| 327000.lg | 3613.mac | 5250_32.exe | 525002.lg |

Name:

Receive

Cancel

☐ Text file: convert from Iso 8859 to Windows

Conversion...

☐ LF->CR/LF

Use Protocol: ASCII

Options...

If Target File Exists:

**4.** When the transfer is finished, press <Ctrl><D> on the keyboard to complete the capture of the source file on the UNIX side, . The character # should reappear.

The transfer of the source code file is now over and all it needs is compiling.

## 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 multiuser 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 (native or WinSock DLL)
- Dial-up networks using modems
- Interrupt 14h (only under Windows 3.x)
- Interrupt 6Bh (only under Windows 3.x)
- NVT over IPX (Novell Netware) (only under Windows 3.x)

Some of **Tun EMUL**'s main features include:

- Entirely configurable terminal emulation
- 21 asynchronous emulations are supported (VT220, VT320, VT520, ANSI, WYSE 60...)
- IBM (3270 and 5250) synchronous emulation
- Complete Microsoft Windows integration
- Revamping and function-key panels
- 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" terminals available on the market (ANSI, IBM 3151, VT220, VT320 & VT520, 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 emulation is 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 in order to customize emulations.

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

## Tun EMUL is compatible with the principal communication supports available

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 (only under Windows 3.x)
- Interrupt 6Bh (only under Windows 3.x)
- NVT over IPX (Novell Netware) (only under Windows 3.x)

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.

Win 3.X

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

Win 3.X

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

Win 3.X

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 previously mentioned communication supports in order to achieve several simultaneous sessions, even using different modes of communication and types of emulation.

TCP/IP and IPX allow several connections on a single server or on different servers. Using these protocols, **Tun EMUL** behaves as several 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). The MS-DOS version is supplied as a separate package. 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 Windows files that can be called automatically, even from within an application.

The supported protocols are:

- RTUNPLUS
- XModem
- YModem
- ZModem

## Tun EMUL provides emulation for IBM 5250 and 3270 terminals

In addition to traditional asynchronous emulations, **Tun EMUL** also offers separate 5250 and 3270 emulation modules for TCP/IP under Windows. 5250 and 3270 are synchronous emulations typically used in IBM AS/400 environments.

## Tun EMUL includes revamping functions

Under Windows, **Tun EMUL** incorporates new graphic attributes which, in association with the standard attributes (reverse video, underline...) give a graphical look to traditional UNIX applications. It is thus possible to totally revamp the look and feel of your character-based UNIX applications by changing the display settings of characters, colours and fonts, and by applying various 3-D effects.

Careful consideration should be given to the colour schemes, function-key panels and bitmaps used when revamping your application to create a corporate image. It is easy to make an application look worse by an inconsiderate choice of colours and it is worth scheduling time to ensure a professional result is achieved.

## TUN EMUL AND NIS

**Tun EMUL** allows access to the NIS server by calling the centralized resource management application **Tun NIS**. The installation of **Tun NIS** is proposed during the installation of **Tun EMUL**.

This functionality allows the user to view the resources available on the network and to select them directly with a simple click of the mouse. In the context of the terminal emulation proposed by **Tun EMUL**, the application **Tun NIS** lets you access the different servers present on the network conveniently and select the one you want to emulate. The network administrator, of course, must have previously configured the NIS server and defined the Servers resource table using the NIS Browser application.

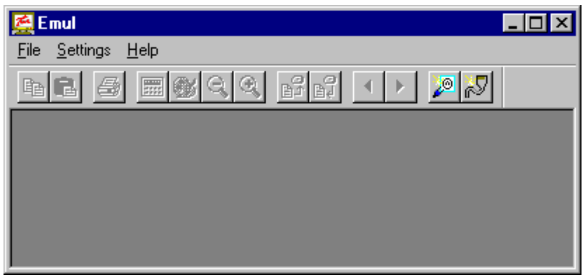
**Tun NIS** also allows access to objects shared on the network such as images (for the screen background, for example) and function-key panels.

For a full description of the NIS Browser, refer to the relevant section in the manual "**TCP/IP Network Services**".

# CHAPTER 3 - USING THE ASYNCHRONOUS EMULATOR

## STARTING THE EMULATOR

Run the program by clicking on the **Emulator** icon in **Tun EMUL** group in Windows 3.x or in the Windows 95 Start menu.



**Tun EMUL** offers users two ways of using terminal emulation:

1. Users may directly open sessions, and manually select parameters such as host name, terminal type and communication protocol.
2. Users can create configurations that contain all the details concerning the sessions they intend to use.

This section will explain the first option, covering the various aspects of using **Tun EMUL** under Windows. As configurations are based on the details given here, they are explained further in this chapter.

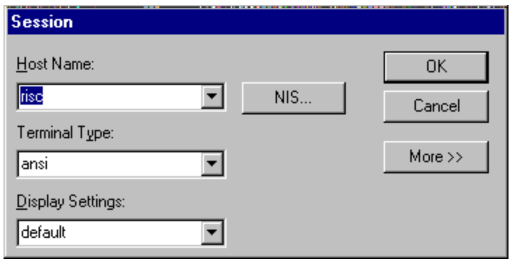
## OPENING SESSIONS

Select **File➔New Session** in order to begin a terminal emulation session. At the point, you will be asked to select the type of communication you intend to use. Several choices are available:

| | |
|---|---|
| **TCP/IP** | Telnet session over TCP/IP (WINSOCK.DLL) |
| **RS-232** | Emulation using COM ports (serial) |
| **Int14h** | Character-mode interface for x.25, ISDN (Windows 3.x) |
| **Int6Bh** | Packet-mode interface (Novell's NVT) (Windows 3.x) |
| **NVT** | Uses Novell's NVT/IPX protocol (with ESKER's NVT.EXE) (Windows 3.x) |

## EMULATION OVER TCP/IP

If you select TCP/IP, the following window will appear:



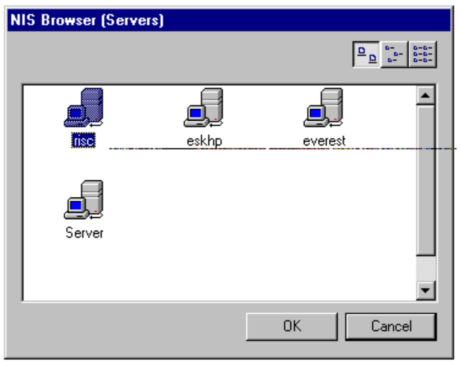
Terminal emulation over TCP/IP requires you to fill in two fields:

### Host Name

Enter the **name** or **IP address** of the host you wish to connect to in this field. If you are using **Tun KERNEL** TCP/IP stack, then you may click on the button next to the field to select a server from the current **Host table**.

You may use the **NIS...** button to view the servers installed on the network. Refer to the section "**Tun EMUL and NIS**'.

The following window is displayed:



Double-click the resource you wish to use.

### Terminal Type

Click on the button next to the field to choose from one of the available terminals:

| TERMINAL | DESCRIPTION |
|---|---|
| ANSI | SCO UNIX, SCO XENIX console |
| AT386 | Interactive UNIX(386) console, UNIVEL |
| ATO300 | ALCATEL APX |
| FT | Fortune |
| IBM3151, HFT | IBM RS6000 |
| HPTERM | Hewlett Packard console (2392A) |
| IMP | Printer (soft copy) |
| MINITEL | Minitel |
| TM266 | Phillips P90x0 |
| TO300 | UNISYS U6000 |
| TWS | BULL DKU 7102 |
| VT52, VT100, VT220, VT320, VT520 | DEC |
| WYSE50, WYSE60 | WYSE |
| 97801 | Siemens/Nixdorf |

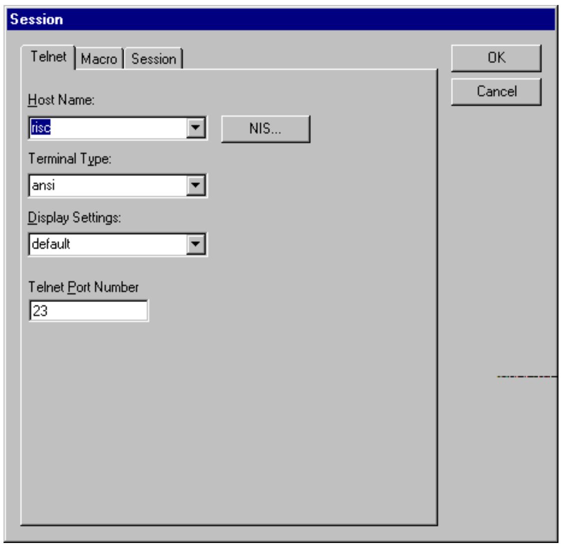
Choose the terminal type appropriate to the host machine or the application you intend to use.

### Display Settings

The **Display Settings** field takes the name of a file containing the emulator's setup parameters. This type of file can be created using the **Settings➔Display Settings** option in the main menu. The value **default.ctx** represents the default settings.

### More

The **More >>** displays a more detailed connection dialog box:



Host Name: risc
Terminal Type: ansi
Display Settings: default
Telnet Port Number: 23

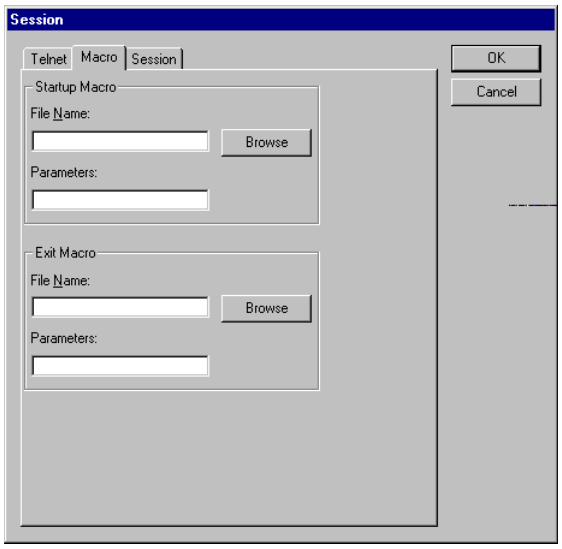
| | |
|---|---|
| **Note:** | This dialog box is also accessible with the option **Settings➔Session...** in the main menu. |

The dialog box contains three tabs, the first of which contains the fields from the connection dialog box.

The two other tabs let you associate with the session:

- First of all, a session startup macro and an exit macro (**Macro** tab).
- Secondly, a terminal configuration saved in a **.ses** configuration file. The settings in this file interact with the sequence files **.seq** (**Session** tab). Refer to the chapter "**Terminal Customization**" for more details.

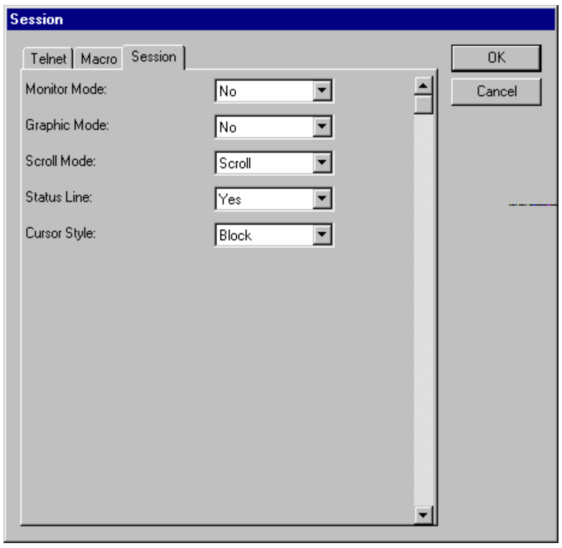
### Macro



You can run a macro at session connection and/or disconnection.

For one or both of these macros, enter the name of the macro in the **File Name** field, or select a name with the **Browse** button. Enter any macro parameters in the relevant**Parameters** field.

## Session



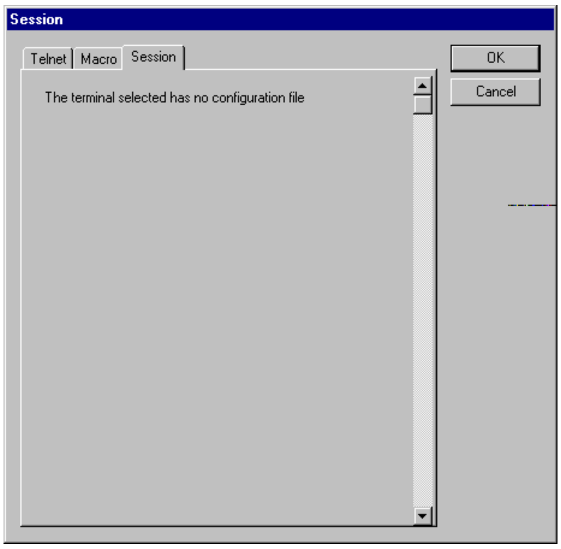
Monitor Mode: No
Graphic Mode: No
Scroll Mode: Scroll
Status Line: Yes
Cursor Style: Block

A terminal is associated with configuration parameters (e.g. cursor size, special character sets, etc.). When you select a terminal type (on the **Telnet** tab in this dialog box or in the Connection dialog box), you automatically associate a terminal configuration file to the session (if there is one). These configuration files are files with the extension **.ses** that the user can create himself (refer to the example of a **.ses** file in the chapter "**Terminal Customization**").

**Tun EMUL** immediately proposes a list of pre-defined **ses** files which can be used directly.

The **Session** tab lets you view the parameters in this file (if it exists), and modify them if necessary.

If there is no **.ses** file associated with the terminal, the **Session** tab appears as follows:



The terminal selected has no configuration file

## Starting an emulation session

After selecting the host and terminal type, click on **OK** to open the connection. If TCP/IP is running correctly on the PC and on the host, you should receive a login prompt.

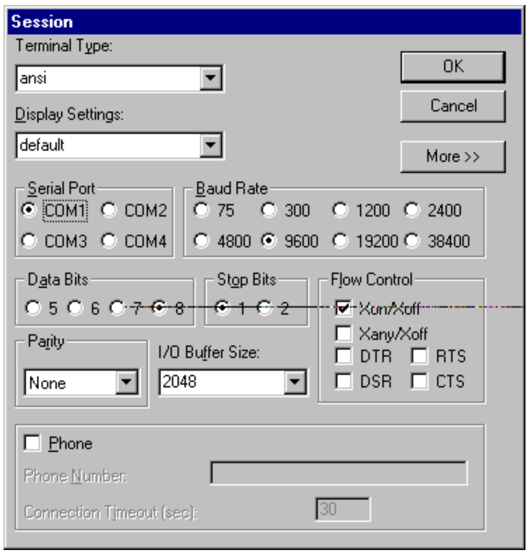
Win 3.X

| | |
|---|---|
| **Note**: | To establish the TCP connection, EMULWIN will try and load the dynamic link library file WINSOCK.DLL whose path should be indicated in the TUN.INI file situated in the WINDOWS installation directory. |

## EMULATION OVER SERIAL LINES

**Tun EMUL** can use serial ports COM1 through COM4, even simultaneously, in order to connect to several different hosts. However, only one connection per port is possible at a time.

Select **File➔New Session➔RS-232** for serial line settings:



Session

Terminal Type: ansi

Display Settings: default

OK | Cancel | More >>

Serial Port: [✓ COM1] COM2 COM3 COM4

Baud Rate: 75 300 1200 2400 4800 [✓ 9600] 19200 38400

Data Bits: 5 6 7 [✓ 8]

Stop Bits: [✓ 1] 2

Flow Control: [✓ Xon/Xoff] Xany/Xoff DTR RTS DSR CTS

Parity: None

I/O Buffer Size: 2048

Phone

Phone Number:

Connection Timeout (sec): 30

### Terminal

Click on the button next to the field to choose from one of the available terminals (Cf. list of terminals for "**Emulation over TCP/IP**" above).

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

### Display Settings

The field **Display Settings** contains the name of a file holding the emulator's working parameters. This type of file can be created using the option **Settings➔Display Settings** in the principal menu. The value **default.ctx** represents the default settings.

### Serial Port

This field determines which of the PC's communication ports (COM1 to COM4) is to be used. The use of ports COM3 and COM4 is only possible if they have been first of all configured.

### 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 to match 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

Select the values you need with the mouse.

#### More

The **More >>** button allows access to a more detailed connection dialog box. Refer to the preceding section dealing with a TCP/IP session to complete the fields in this box.

### Using a modem

For server connection using modems, select the check box **Phone**. Modem connections use all the same parameters as direct serial connections. More details are given on this type of connection in the next section.

### Starting emulation

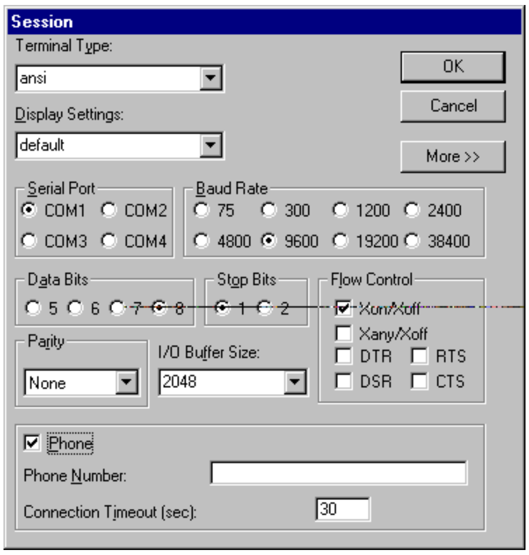
To connect to a host using the parameters described above, click on the **OK** button.

**Note**: Serial connections are more difficult than TCP/IP, and may require special attention to cabling and host configuration. Additional guidance is given in Appendix A in this manual.

## EMULATION USING MODEMS

Emulation using modems is almost identical to using direct serial connections (RS-232). After connection is established over telephone lines, a PC in emulation behaves the same as a local terminal would.

As with direct RS-232 connections, select **File➔New Session➔RS-232**:



Terminal Type: ansi

Display Settings: default

Serial Port: [✓ COM1] COM2 COM3 COM4

Baud Rate: 75 300 1200 2400 4800 [✓ 9600] 19200 38400

Data Bits: 5 6 7 [✓ 8]

Stop Bits: [✓ 1] 2

Flow Control: [✓ Xon/Xoff] Xany/Xoff DTR RTS DSR CTS

Parity: None

I/O Buffer Size: 2048

[✓ Phone]

Phone Number:

Connection Timeout (sec): 30

Most of the fields in the figure above are described in the preceding section.

| | |
|---|---|
| **Note**: | Do not forget to take the speed of your modem (and that of the host) into account when selecting **Baud Rate**. |

### Phone

Select this check box to indicate that a modem is attached to the specified COM port.

### Phone Number

You may enter the telephone number of the host system in this field, unless you prefer to dial manually.

Commas ( , ) may be used to insert a two second pause during dialing. This may be necessary for obtaining external lines in office buildings or for dialing internationally (i.e. 011,33,78934636).

If you entered the telephone number, then the emulator will dial automatically at startup. If you leave the field blank, you may use menu options **Connection➔Dial** in order to use the modem.

### Connection Timeout

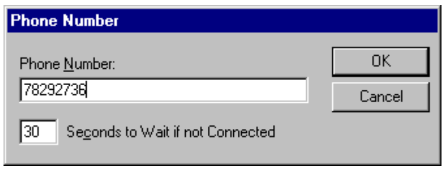
This field indicates the time (in seconds) after which, if no response is heard from the remote host, the emulator will consider the connection failed. Sixty seconds is usually a reasonable value.

| | |
|---|---|
| **Notes**: | **Tun EMUL** uses standard HAYES commands for modem control. If you need to modify the codes used for your modem, or for special configurations such as Minitel or 14400 baud, use menu option **Settings➔Modem Commands...**<br><br>When using Minitel emulation, use the Minitel font provided with **Tun EMUL**, as well as the Minitel function-key panel. Details on fonts and function-key panels are given further on in this chapter. |

## Starting modem sessions

After filling in the fields as described above, click on**OK** to begin a modem session.

If a number was specified, it will be dialed. If not, you will have to use the field **Connection➔Dial**, and then click on **OK**:


Phone Number

Phone Number: 78292736

30 Seconds to Wait if not Connected

## EMULATION USING INTERRUPT 14H (WINDOWS 3.X)

Win 3.X

| **Note**: | This type of emulation has been dropped from the 32-bit version of the program. |
|---|---|

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 a COM port, choose INT14-1, INT14-2, INT14-3, or INT14-4 depending on your software's Int 14h driver.

## Starting an emulation session

Click on **OK** to open a connection. If the Int14h driver is running properly, you should receive a login prompt.

## EMULATION USING INTERRUPT 6BH (WINDOWS 3.X)

Win 3.X

| **Note**: | This type of emulation has been dropped from the 32-bit version of the program. |
|---|---|

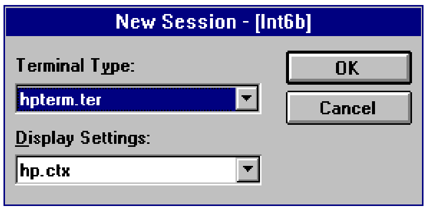
The "universal interface" concept with Interrupt 6Bh is very similar to Interrupt 14h, except that data is exchanged in **packet mode**. For example, NVT/IPX from Novell provides an Interrupt 6Bh interface with its resident program NVT.EXE.

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.


New Session - [Int6b]

Terminal Type: hpterm.ter

Display Settings: hp.ctx

The version of NVT.EXE supplied with the IPX package for UNIX requires the user to select a server for connection. Therefore there is no need to select a host name within **Tun EMUL**. The only fields which have to be filled in for this type of connection are**Terminal** (type) and **Display Settings**.

## Starting an emulation session

After selecting the terminal type, all you have to do is click on **OK** to begin an emulation session. If an Int6Bh connection was previously established, you should receive a login prompt.

## EMULATION OVER NVT/IPX (WINDOWS 3.X)

Win3.X

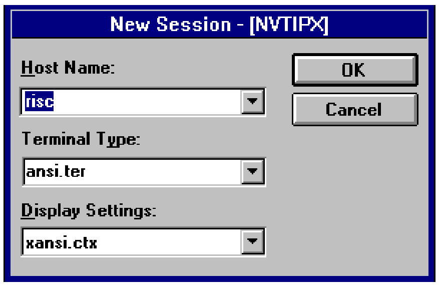
| **Note**: | This type of emulation has been dropped from the 32-bit version of the program. |
|---|---|

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. In particular, Windows users do not have to return to MS-DOS in order to open and close connections.

| **Tun EMUL** | |
|---|---|
| NVT.EXE (by ESKER) | |
| IPX (NIC Driver) | IPXODI LSL + MLID |
| Network Interface Card | |

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


New Session - [NVTIPX]

Host Name: risc

Terminal Type: ansi.ter

Display Settings: xansi.ctx

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.

After entering the host name and terminal type, click on**OK** to begin emulation.

## USING THE EMULATOR

### Managing multiple sessions

**Tun EMUL** runs in Multiple Document Interface mode (MDI), and can therefore handle several terminal emulation sessions at the same time.

In order to continue opening sessions, use **File➔New session** as described in the previous section. Additional sessions are opened as child windows in the **Tun EMUL** desktop. By default, they are presented in cascade form, but you may resize them using the mouse or change the layout using**Window➔Tile**.

To change from one session to another, you may either use the mouse or the menu option **Windows**. With the key-combinations <Alt><F1>, <Alt><F2>, <Alt><F3> and <Alt><F4>, you may quickly return to one of the first four sessions.

### Hard copy

This option may be used to print the contents of an emulation session. By selecting **File➔Print Screen**, you may use **Text** mode (faster), or **Graphic** mode, which transforms colors into different shades of gray.

The options **File➔Print Setup...** may be used to select the printer for printing screens.

### Toolbar

Win 95

The most frequently used Windows options may be displayed as buttons above the principal emulation screen. In an open session, click on **Settings➔Show Toolbar** in order to use buttons for actions such as *cut & paste* and *hard copy*. They can be removed by toggling off the same option.

A standard toolbar is proposed which can be customized by the user. The toolbar can be moved and is dockable in the main window, that is it can be placed along the side or bottom of the screen. If there are a lot of buttons, the toolbar can be wrapped, meaning that a second row or column of buttons is created depending on whether the toolbar is docked along the side or bottom of the screen.

If the cursor is left over a  button for a couple of seconds a tooltip will appear, a short text explaining the function of the button. At the same time, a description of the button's function is displayed on the status bar at the bottom of the screen.

If you prefer to see small buttons displayed in the toolbar, use the command **Settings➔Options** and select the check box **Small Toolbar Buttons**

## Toolbar Customization

In the 32-bit version, you can edit the toolbar directly by double-clicking on the toolbar itself (and not a button). A dialog box will allow you to choose which buttons you wish to display in the toolbar. As well as adding and removing buttons, you may add a separator (a space) to the toolbar between buttons or groups of buttons.

You can also choose your own toolbar bitmap. Specify the bitmap path in the Registry setting **BitmapPath**. Also in the Registry, you may change the order of the buttons (**Order**) and the number of buttons in the toolbar (**BitmapButtons**).

The following buttons are available:

New TCP/IP Session

New RS-232 Session

New NT14, NT6b, or NVT-IPX Session

Open a file

Open a configuration

Save a configuration

Cut

Paste

Print

Display function-key panel

Run the function-key panel editor, Panedit

Execute a macro

Automatically save a macro

Save an expected string

Edit a macro

Open the Display Settings window at the Attributes tab

Open the Display Settings window at the tab Terminal Size and Font

Open the Display Settings window at the tab Background

Open the Display Settings window at the tab Mouse

Open the Display Settings window at the tab Function-key Panel

Modify the keyboard

Set terminal

Set session parameters

Set emulator

Zoom out

Zoom in

Receive a file from the UNIX machine on the DOS machine

Send a file from the DOS machine to the UNIX machine

Open the line and dial a number (RS-232)

Change the phone number

Hang up

Return to previous session

Go to following session

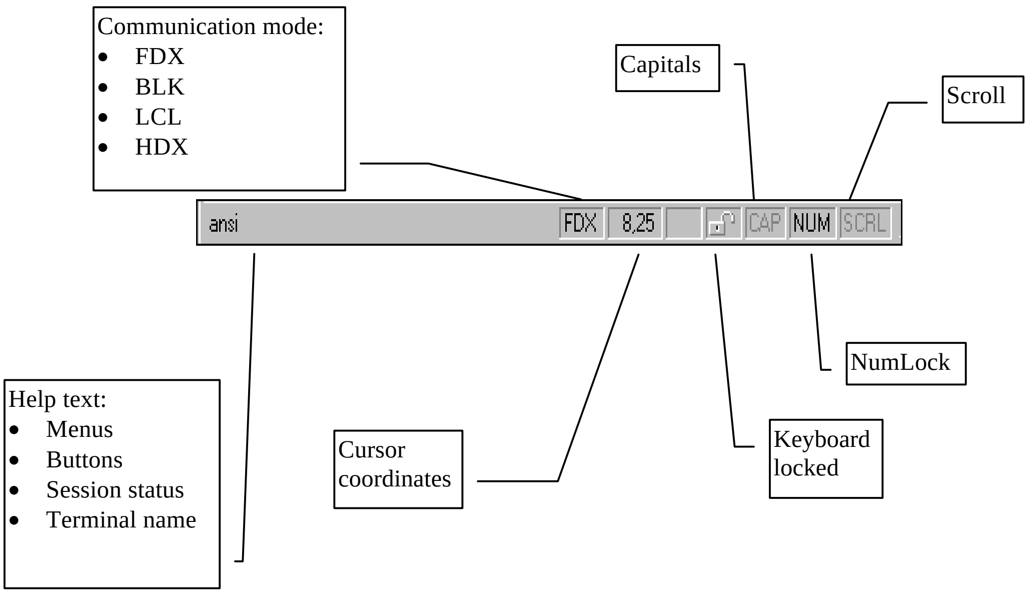
Cascade windows

Tile windows horizontally

Tile windows vertically

## Status Bar

The Status Bar may be displayed or hidden via the menu option **Settings➔Display Status Bar**. The following diagram shows the function of the indicators:



## Cut & Paste

With the mouse, you may select all or part of an emulation screen and use the standard **Copy** feature to place it in the Windows Clipboard. Either the command button , or menu option **Edit➔Copy** may be used.

You may also place the contents of the Clipboard onto the emulation session by using the command button or menu options **Edit➔Paste**.

By using **Edit➔Copy Option...**, you may control **CR/LF conversion**, the **wait state** (useful when Clipboard contents are very large ), and whether or not you wish to select rectangular blocks of an emulation screen. The field **Wait State** (expressed in milliseconds) delays the clearing of the clipboard during a voluminous paste operation. This avoids creating a bottleneck in the communication channel.

## Double-clicking in UNIX applications

Most character-based terminals do not support the use of a mouse in UNIX applications. However, application menus often react to the first letter of the word used to describe an option (i.e. pressing "Y" for yes, or "C" to continue...). **Tun EMUL** helps users take advantage of this by sending any letter on which you double-click with the mouse, as if it were typed on the keyboard.

## Changing the terminal display

There are many elements that make up the desktop of an emulation session. Any aspect of the terminal screen may be changed at any time:

- character font size and type
- colors
- size of the emulation screen
- function-key panels and command buttons
- scroll bars

These various components may adjusted by using **Parameters➔Display settings** during an emulation session (described in detail in **Customizing terminal display** later in this chapter).

## Function-key panels

Users may create custom visual function-key panels to automate many tasks. The buttons in function-key panels can send a wide variety of characters, strings and commands, and can serve as a way to create short-cuts for using UNIX from your PC. These mouse-driven function-key panels may be "attached" to any terminal session using **Display Settings** files (described in the next section).

The command button allows users to toggle on and off the use of function-key panels in the current session.

## Dynamic sizing

When **Dynamic Sizing** is activated (under **Display Settings**), and you use a character font available in several sizes (such as SystemPC), you may use the mouse to change the size of the emulation screen and font at the same time.

If you increase or reduce the emulation screen, the font changes size accordingly in order to always show a full terminal display with 80 columns and 25 lines. This option provides a more realistic display and easier control of multiple sessions.

**Note**: If your emulation uses two fonts at the same time, they both need to exist in the same sizes for **Dynamic Sizing** to function.

### Running macros

**Tun EMUL** allows users to create macros for automating complex tasks (described in the **Automation** chapter in this manual). Click on **File➔Macro** to run existing macros.

You can create simple macros using the option **Macro➔Record Macro**. Enter the new macro name in the dialog box displayed. Repeat the following operations as often as necessary:

- enter characters at the keyboard or perform a **Paste** operation, thus generating an instruction of the type "Send *String*";
- select an area of the screen with the mouse and click on the button **Wait**, thus generating an instuction of the type "Receive 60 *String*" for each line selected with an accompanying "IfError ERROR" instruction, ERROR being a label placed at the end of the file.

Any hesitation (over 4 seconds) on the part of the user generates a "Sleep *time*" instruction in the macro, where *time* is the duration in seconds of keyboard inactivity.

The macro can later be edited. Macros created in this way can also be included in function-key panels.

## PRE-CONFIGURING SESSIONS

**Tun EMUL** uses configuration files (.CFG) to memorize and re-use terminal parameters such as display settings, host names and number of sessions, etc., as well as session start and end macros and terminal configuration on a per session basis.

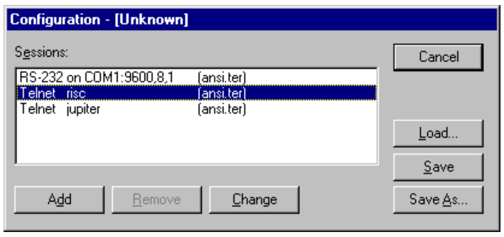
**Note**: The same configuration files are common to **Tun EMUL** under both Windows and MS-DOS.

### Saving current sessions

One way to create a configuration file is to open manually and set up the desired terminal sessions, and then to use the menu options **File➔Save configuration**. **Tun EMUL** will prompt you to enter a name for the .CFG file, and then will save all the parameters used in active sessions.

### Creating new configurations

Another way to create configuration files is to use **Settings➔Configuration...** to pre-define all the settings you would like to use, without actually opening each session first.



```
RS-232 on COM1:9600,8,1    (ansi.ter)
Telnet  risc               (ansi.ter)
Telnet  jupiter            (ansi.ter)
```

This option may also be used to modify existing configuration files (by pressing the **Load...** button).

Click on the **Apply** button to start the configuration.

### Starting emulation with configurations

After opening EMULWIN, you may use menu options **File➔Open configuration** to start the emulator with a pre-defined configuration.

You may also specify the name of the .CFG file directly on the command line, as shown below, in order to start emulation directly without going through the menu:

```
c:\tun\emul\emulwin demo.cfg
```

A windows icon can be made with the above command line in order to further simplify startup of frequently-used configurations.

## CUSTOMIZING THE TERMINAL DISPLAY AND REVAMPING

As mentioned earlier, **Tun EMUL** allows users to adjust the appearance and layout of emulation sessions. Instead of offering separate menu options for each parameter, **Tun EMUL** uses display settings files (.CTX) to group them together for each session.

Display settings files specify:

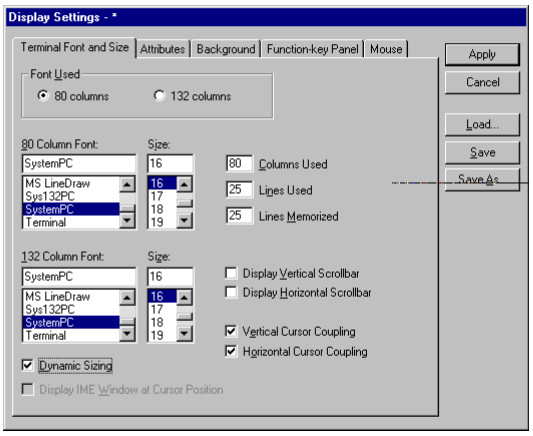
- character font size, type and attributes
- colors and effects
- background
- size of the emulation screen
- function-key panels
- customizable toolbar
- scroll bars

Display settings files may be stored and "attached" to sessions configurations (described in the preceding section). These .CTX files are common to the Windows and MS-DOS emulation modules.

The powerful facilities offered by the configuration mechanism give a graphic look and feel to traditional UNIX applications and amount in practical terms to a revamping of UNIX applications.

### Adjusting display settings

Click on **Parameters➔Display Settings...** in order to select the elements you would like to use:



Font Used: [✓ 80 columns] 132 columns

80 Column Font: SystemPC — Size: 16

132 Column Font: SystemPC — Size: 16

80 Columns Used

25 Lines Used

25 Lines Memorized

☐ Display Vertical Scrollbar

☐ Display Horizontal Scrollbar

☑ Vertical Cursor Coupling

☑ Horizontal Cursor Coupling

☑ Dynamic Sizing

☐ Display IME Window at Cursor Position

The dialog box displayed is based on five bookmarks which give access to five main types of parameter settings:

- Screen mode and font used
- Display attributes
- Screen background
- Function-key panels
- Mouse controls

## Character size and font

The parameters defining the terminal screen size and the character fonts to be used may be displayed by clicking on the bookmark **Terminal Font and Size**

## Dimensions

The default setting for EMULWIN.EXE emulates 80 x 25 screens. It is possible to change this setting by changing the fields Lines Used and TermCols. This is useful for emulating other terminals using different dimensions (for example, 132 columns or 43 lines).

Generally speaking, only 25 lines are simultaneously memorized by EMULWIN; however, this value may be changed using the field **Lines Memorized**. The maximum value for this field is 1024. Use the scroll bars on the right of the emulation screen to view the lines memorized by EMULWIN which have not yet been displayed.

## Scroll bars

The check boxes **Display Vertical Scrollbar** and **Display Horizontal Scrollbar** toggle the display of horizontal and vertical scroll bars along the side and bottom of the emulation screen. The scroll bars are useful if you have opted for more than 25-line storage and if you are not using **Dynamic Sizing**

## Character font

Any non-proportional character font available under Windows may be used in emulation. However, in order to be able to use all of the semi-graphic characters correctly, the font needs to be **OEM**, and not **ANSI**.

The font **SystemPC** is furnished with **Tun EMUL** for the best possible use of emulation under Windows. It is a fixed OEM font ranging in size from 2 to 30 developed especially for **Tun EMUL**. Use **Sys132PC** for emulation with 132 columns.

The fields **80 Column Font** and **132 Column Font** define the type and size of font which will be used for the emulator in either mode (80 or 132 columns).

The check boxes **80 Columns** and **132 Columns** make it possible to select the character font which will be used whatever the width of the emulated terminal.

## Dynamic sizing

This function causes the emulator to change the size of the font according to the size of the emulation window in such a way as to always display a full terminal screen (80 x 25 or 132 x 25 depending on the terminal).

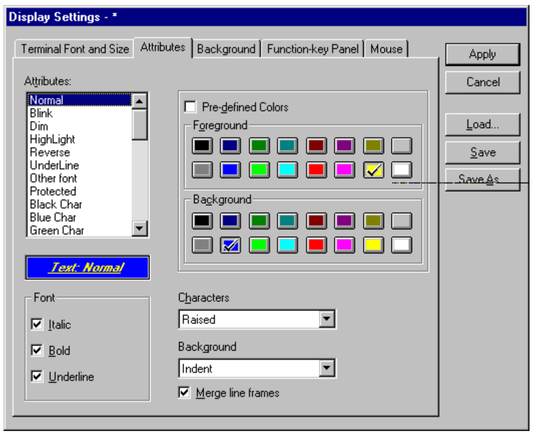
Dynamic sizing is best used with character fonts that are available in multiple sizes (as is the case with **SystemPC** or **Sys132PC**).

## Cursor coupling

The check boxes **Horizontal cursor coupling** and **Vertical cursor coupling** allow you to enable or disable cursor coupling in either direction. Cursor coupling scrolls the screen so that the cursor is always visible. If either check box is unmarked it is possible to move the cursor beyond the view of the terminal window.

## Attributes

Clicking on the bookmark **Attributes** displays the following screen:



Attributes:
Normal
Blink
Dim
HighLight
Reverse
UnderLine
Other font
Protected
Black Char
Blue Char
Green Char

☐ Pre-defined Colors

Font
☑ Italic
☑ Bold
☑ Underline

Characters: Raised

Background: Indent

☑ Merge line frames

This dialog box lists the attributes generally found in standard emulation sessions on the left at the top:

- Normal
- Blink
- Dim
- Highlight
- Reverse
- Underline
- Special characters
- Protected
- Standard colors

The remaining fields, buttons or tables in the dialog box can be used to remap all or part of the standard attributes. The effect is to give a graphic look to standard applications in character mode. For example, the attribute reverse video may be set in relief on the screen by using red characters on a gray background.

Proceed as follows to transform a standard attribute and give it special effects:

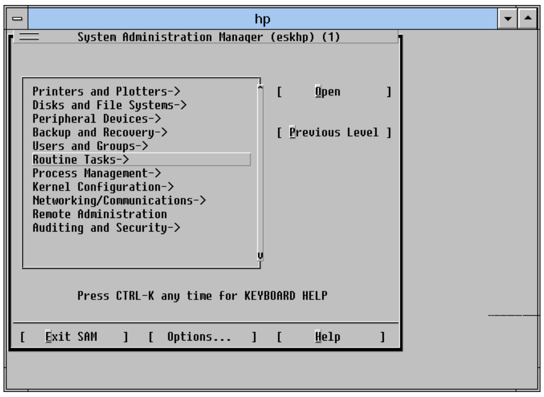
- Select a standard attribute in the attribute list
- Use the other fields to select the desired visual effects.

The following effects can be associated with the standard attributes using **Tun EMUL**:

| **Effect** | **Options** | **Examples** |
|---|---|---|
| Foreground color | 16 colors | red on green |
| Background color | 16 colors | green on red |
| Text style | Default<br>Normal or flat<br>Relief<br>Inset<br>Shadowed | flat text_<br>raised text_<br>inset text<br>shadowed text |
| Border style | Default<br>Normal or flat<br>Indent background<br>Outdent background<br>Outdent frame background | Indent background<br>Outdent background<br>Outdent frame background |
| Font style | Italics<br>Bold<br>Underlined | *italic text*<br>**bold text**<br>underlined text |

| | |
|---|---|
| **Note**: | To use the relief or shadow effects, it is better to choose a gray background.<br><br>All the options can be combined (Example: Bold + Inset + Outdent background) to produce the desired effect. |

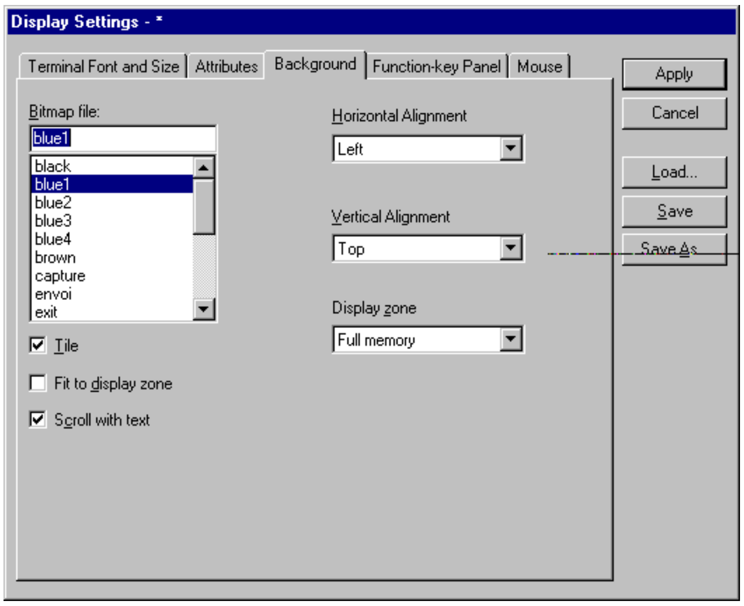
Here is an example of a screenshot obtained by mixing attributes (HP-UX System Administration Manager):



```
hp
System Administration Manager (eskhp) (1)

Printers and Plotters->                 [     Open       ]
Disks and File Systems->
Peripheral Devices->                    [ Previous Level ]
Backup and Recovery->
Users and Groups->
Routine Tasks->
Process Management->
Kernel Configuration->
Networking/Communications->
Remote Administration
Auditing and Security->

        Press CTRL-K any time for KEYBOARD HELP

[   Exit SAM    ]  [  Options...   ]  [     Help      ]
```

## Background

Clicking on the bookmark **Background** displays the following window:



Bitmap file: blue1

Horizontal Alignment: Left

Vertical Alignment: Top

Display zone: Full memory

☑ Tile

☐ Fit to display zone

☑ Scroll with text

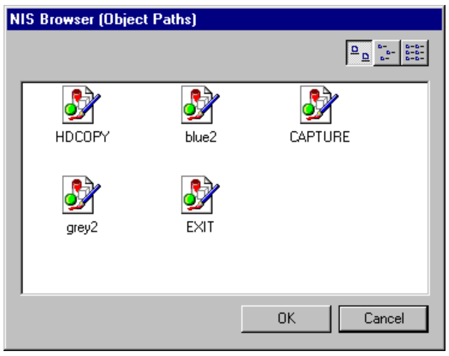
This dialog box allows the user to display a background image (wallpaper) during the current emulation session. To display an image, select a bitmap file in the list and add the desired parameters.

### File

This field should hold the name of the bitmap file (.BMP) with the required image. By default the program will look for the .BMP extension files in the **Tun EMUL** installation directory.

You can also use the **NIS...** button to view picture objects shared on the network. Refer to the section "**Tun EMUL and NIS**' for more information.

The following window is displayed:



Double-click the icon of the desired resource.

### Alignment

The fields **Vertical Alignment** and **Horizontal Alignment** allow the user to adjust the relative position of the bitmap if it is smaller than the emulation screen.

### Tile

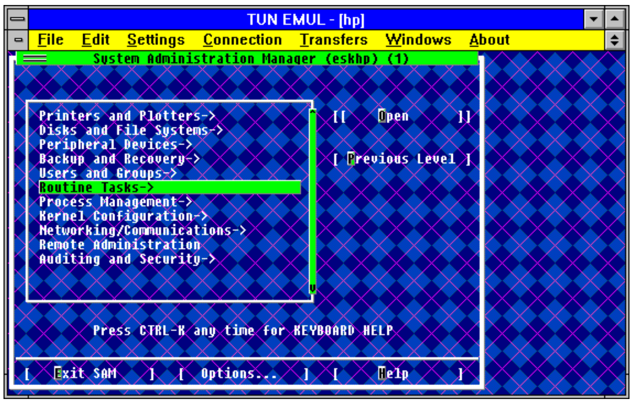
If the image is smaller than the emulation screen, it may be duplicated until it fits all the available space by selecting this check box.

### Fit to display zone

If the bitmap does not coincide exactly with the size of the emulation screen, it is possible to resize it until it occupies all the available space by selecting this check box.

### Display zone

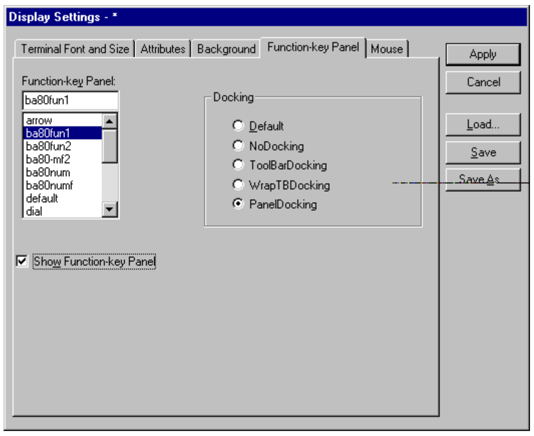
The display zone of the image can be the emulation screen or the whole of the emulation window. Here is an example of a screen obtained by displaying an image in tile mode in the background of the emulation window:



```
System Administration Manager (eskhp) (1)

Printers and Plotters->          [[    Open      ]]
Disks and File Systems->
Peripheral Devices->             [ Previous Level ]
Backup and Recovery->
Users and Groups->
Routine Tasks->
Process Management->
Kernel Configuration->
Networking/Communications->
Remote Administration
Auditing and Security->

       Press CTRL-K any time for KEYBOARD HELP

[  Exit SAM    ]  [  Options...   ]  [    Help     ]
```

## Function-Key panel

Clicking on the bookmark **Function-key Panel** displays the following window:



Function-key Panel: ba80fun1

arrow
ba80fun1
ba80fun2
ba80-mf2
ba80num
ba80numf
default
dial

Docking

- Default
- NoDocking
- ToolBarDocking
- WrapTBDocking
- [✓ PanelDocking]

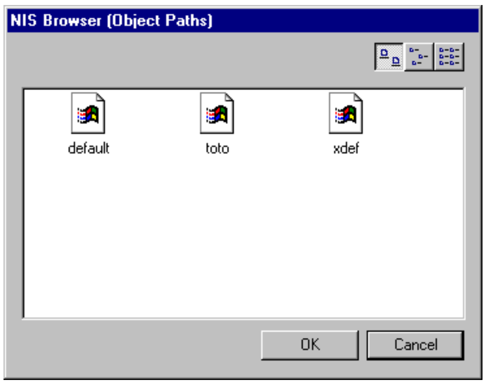
[✓] Show Function-key Panel

This option may be used to associate a visual function-key panel with emulation sessions. Function-key panels contain buttons that may be used with a mouse to start macros, programs, scripts, or to navigate within UNIX programs. Key-panels can be customized with the options **Settings➜Function-key Panel...** or by clicking on the icon **Panedit** (Panel Editor) in the **Tun EMUL** group under Windows 3.x or from the Windows 95 Start menu.

The field **Function-key Panel** in the dialog box should contain the name of the .PAN file that defines the panel.

You can use the **NIS...** button to view the function-key panel objects shared on the network. Refer to the section '**Tun EMUL and NIS**' for more details.

The following window is displayed:



Double-click the icon of the desired resource.

If the box **Show Function-key Panel** is selected, the panel will be displayed on the screen when emulation is started.

The function-key panel can be transformed into a toolbar. To do this, select the docking option you want to apply to the function-key panel. The possible docking options are as follows:

- **Default**: the type of docking used is that defined when the function-key panel was created, in the panel parameters. Refer to the chapter "Function-key Panels" for more details.
- **No Docking** the function-key panel will not be displayed as a toolbar.
- **Normal**: the function-key panel becomes a standard toolbar.
- **Wrap Toolbar**: if there are too many keys on the panel, they are arranged in more than one line or column.
- **Dock as Panel**: the function-key panel becomes a toolbar, but the layout of the keys remains unchanged.

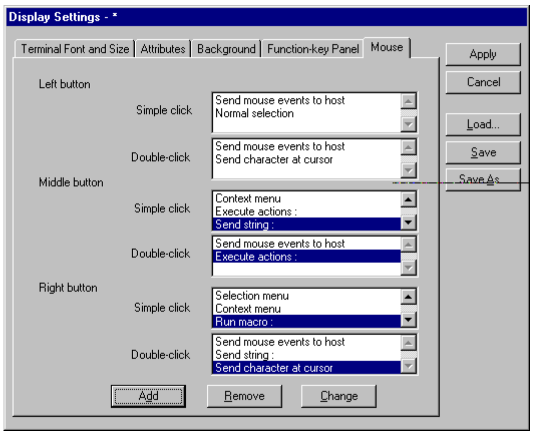
Please consult the chapter dealing with the creation and customization of function-key panels for further information.

**Note**: If the width or the height of the function-key panel is greater than the width or the height respectively of the screen, it cannot be docked.

## Mouse

Different actions can be associated with each mouse button. These actions can then be triggered with a simple click or double-click of the mouse. Refer to the section "**Mouse support in UNIX applications**", in the "**Special actions**" chapter.

Clicking on the **Mouse** tab displays the following window:



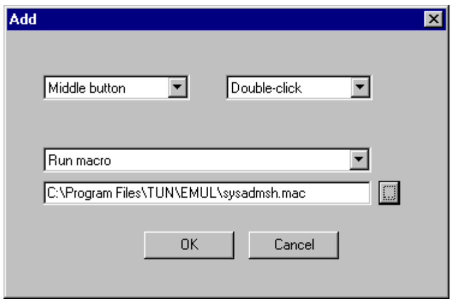
Left button
Simple click: Send mouse events to host / Normal selection
Double-click: Send mouse events to host / Send character at cursor

Middle button
Simple click: Context menu / Execute actions : / Send string :
Double-click: Send mouse events to host / Execute actions :

Right button
Simple click: Selection menu / Context menu / Run macro :
Double-click: Send mouse events to host / Send string : / Send character at cursor

For each of the mouse buttons (allowance is made for three) there are one or more corresponding instructions. Choose the desired behavior for each mouse button.

Instructions may be added, removed or changed by pressing the appropriate button at the bottom of the dialog box. The **Add** button displays the following dialog box:



Middle button
Double-click
Run macro
C:\Program Files\TUN\EMUL\sysadmsh.mac

The **Remove** button simply removes the instruction without displaying a dialog box. The instruction can be added again using the **Add** button.

The **Change** button allows the user to change the instruction selected for a particular button.

The following actions are possible:

- Sending mouse event to the host (blocking action if the mouse support is activated).
- Sending a string.
- Macro execution.
- Execution of actions.
- Function key (keyboard F-keys).
- Selection menu (i.e. context menu for a selection made with the mouse, a blocking action).
- General context menu (blocking action).
- Sending of the character at the cursor.
- Normal selection (blocking action).

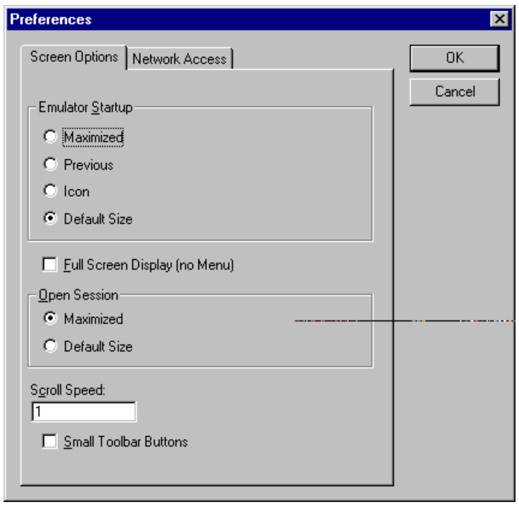
## Saving changes

In order to save any changes or load a display settings file, you may use the buttons **Save** (and **Save as...**) or **Load...**.

.CTX files may be attached to configurations using **Settings➔Configuration**.

## Options

The **Settings➔Options** menu allows you to control Screen Options and to implement a firewall security mechanism:



Emulator Startup
( ) Maximized
( ) Previous
( ) Icon
(●) Default Size

☐ Full Screen Display (no Menu)

Open Session
(●) Maximized
( ) Default Size

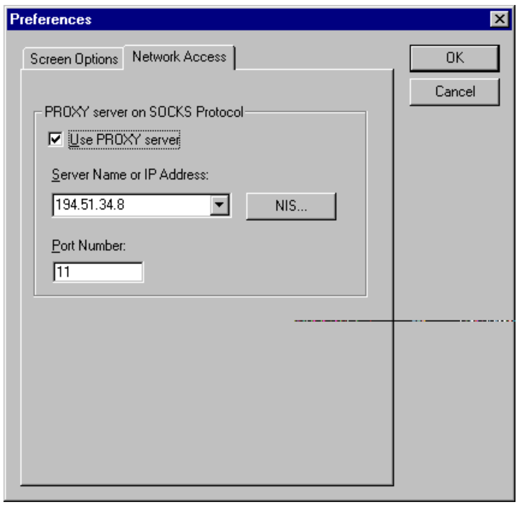
Scroll Speed:
1

☐ Small Toolbar Buttons

### Screen Options

On the first tab you can:

- Choose the startup screen size: this may be the maximum possible size, **Maximized**, the size of the screen on leaving the **Previous** session, reduction to **Icon** form, or the **Default Size** indicated on the **Terminal Font and Size**tab in the dialog box **Settings➔Display Settings...**
- Choose a full-screen display without menus, or other display controls.
- Choose the default screen size for the opening of a new session.
- Select the **Scroll Speed** defining the speed at which screen scrolling will take place. The default is 1.
- Choose whether to display small buttons on the toolbar or not. This is useful if you customize the toolbar with a lot of extra buttons (by clicking on an empty space in the toolbar).



PROXY server on SOCKS Protocol
☑ Use PROXY server

Server Name or IP Address:
194.51.34.8

Port Number:
11

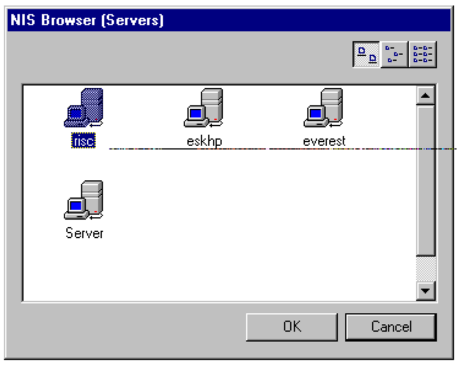
### Network Access

The **Network Access** tab can be used to implement a firewall. This means you can access an outside server by passing through a gateway machine of the Proxy type, which acts as a security filter to protect the local network.

Enter the name or IP address in the gateway field and check the box**Use Proxy Server**.

You can use the **NIS...** button to view the servers installed on the network. Refer to the section "**Tun EMUL and NIS**" for more details.

The following window is displayed:



Double-click the icon of the desired resource.

## TRANSFERRING FILES

**Tun EMUL** supports the following file transfer protocols:

- ASCII
- RTUNPLUS
- X-Modem
- Y-Modem
- Z-Modem

### ASCII file transfer

This is the most basic type of file transfer available in **Tun EMUL**. **Receiving** consists of capturing the characters sent across a connection into a file. The command button **Capture** also performs this function.

**Sending** consists of emptying the contents of a PC file onto the network connection, but does not provide any means of controlling reception. The host is responsible for capturing the data into a file.

The command most frequently used on hosts for this is:

```
ssty -echo ; cat>/tmp/file ; stty echo
```

### RTUNPLUS

ESKER's proprietary file transfer protocol, RTUNPLUS, has the advantage of being simple to use, and freely installable on any UNIX host. RTUNPLUS is provided in UNIX executable format for SCO UNIX and XENIX, and the source code (RTUNPLUS.C) may be compiled on other hosts.

Details on installing the RTUNPLUS server module are given in the Installation chapter of this manual.

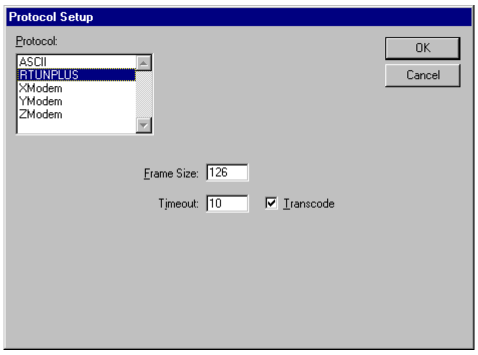
### X, Y and Z-modem

These are the most common file transfer protocols in use today, and are very frequently used when accessing BBS (Bulletin Board Systems).

The server programs for these protocols are usually not delivered as standard on most UNIX systems, but are generally available in the public domain as executables or source files for compilation.

## Protocol parameters

Each protocol provides different options for duplicate file replacement, frame size, retries and so on. To change the default settings, select **Transfers➔Protocol Setup...**

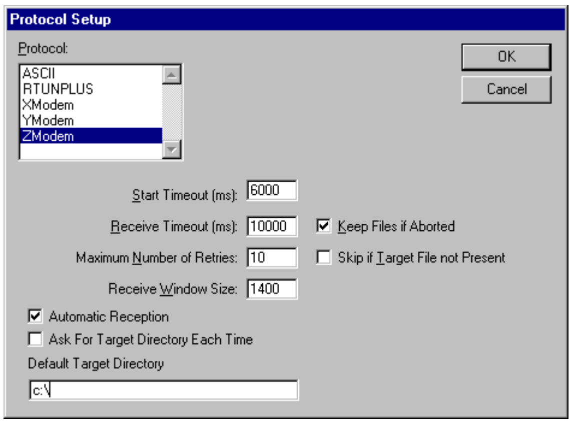


Protocol:
ASCII
RTUNPLUS
XModem
YModem
ZModem

Frame Size: 126
Timeout: 10
☑ Transcode

## Automatic Z-MODEM reception

The Z-MODEM protocol can be used to configure the automatic reception of files, thus avoiding the need to use the option **Transfers➔Get File...** for file transfer with this protocol.



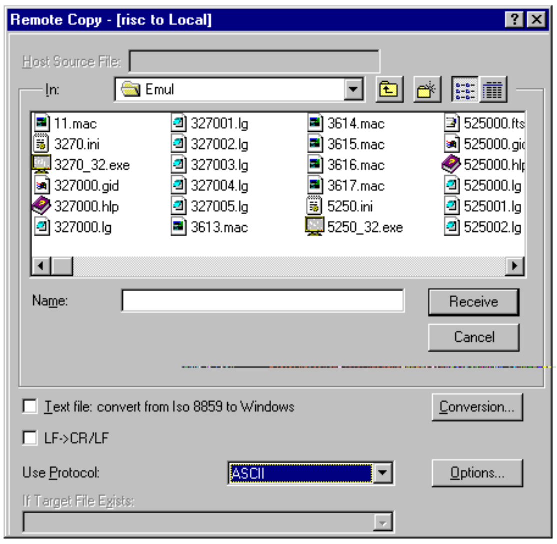
Protocol:
ASCII
RTUNPLUS
XModem
YModem
ZModem

Start Timeout (ms): 6000
Receive Timeout (ms): 10000 ☑ Keep Files if Aborted
Maximum Number of Retries: 10 ☐ Skip if Target File not Present
Receive Window Size: 1400
☑ Automatic Reception
☐ Ask For Target Directory Each Time
Default Target Directory
c:\

To use this function, select **Zmodem** in the protocol list and then select the check box **Automatic Reception** from the Z-MODEM protocol configuration fields. You can also enter the default target directory into which the file will be copied or else choose to be prompted for this directory each time you transfer a file. Select or clear the option **Ask For Target Directory Each Time** or enter the name of the default directory.

## Sending and receiving files

Use the options **Transfers➔Send File** (or **Get File**) to select and transfer files.



Several options are available when transferring files:

### Send

Click on this button to begin the transfer procedure. If you are using RTUNPLUS, then the UNIX host must be in the "shell" (at the #, %, $ prompt, on the command-line) in order to be able to run the server portion of the protocol. To use X, Y and Z-modem, you may need to manually start the server on the UNIX host (i.e. with a command such as **xmodem -r /tmp/tmp.file**).

### Host Target File: / Local Source File:

The names of the fields depend on the direction of the transfer. For receiving files the field names are reversed (**Host Target File** becomes **Host Source File**, etc). from those shown in the above screenshot. Enter the name the file should take when it arrives in its new location.

### Text file: convert...

This option converts the contents of the file from the local character set to the character set of the remote machine. The button **Conversion** displays a dialogue box which allows the two character sets to be selected.

### CR/LF -> LF

This option is used to apply or cancel CR/LF to LF conversions. It is useful when transferring text files from a PC to a UNIX server or vice-versa.

### Use Protocol

Specify one of the available transfer protocols.

### Options

The **Options** button can be used to select a protocol and set related parameters. The parameters vary in number and type depending on the protocol chosen. The button displays the same dialog box as the option **Transfers➔Protocol Setup...**

### If ... File Exists

This option determines what action to take in case you transfer a file that already exists on the destination machine. For example, you may choose to replace files of the same name, transfer the file if your version is newer, or change the name of the file.

| | |
|---|---|
| **Note**: | File reception can be rendered automatic for file transfer using the Z-MODEM protocol. Refer to the section "**Automatic Z-MODEM reception**". |

# CHAPTER 4 - 3270 AND 5250 EMULATIONS

## INTRODUCTION

The terminal emulations and features we have described up to this point have been for **asynchronous** terminals. In fact, EMULWIN.EXE and EMUL.EXE only support this type of terminal.

**Asynchronous** terminals use bi-directional data exchange (server➔PC, PC➔server) in which there is no "traffic control" mechanism. Users may type characters at the same time as the server sends others. **Characters** are the basic data "units" in asynchronous emulations.

Conversely, 5250 is a **synchronous** emulation used with IBM AS/400 machines, in which data traffic is controlled very strictly in **packet** mode. 3270 emulation as encountered on IBM mainframes and as implemented by the program **Tun 3270** (3270.EXE) is also **synchronous** emulation. Characters no longer flow freely between terminals and servers, but rather circulate in **packets**. With this type of communication, terminals do not send data until the server has responded to the previous request, and vice versa. Data exchange between the two machines is structured and organized.

In general, synchronous emulation is less flexible than asynchronous emulation, and offers fewer possibilities for customization and personalization with respect to the user interface. Both **Tun 3270** and **Tun 5250** now have powerful revamping functions which allow the user to define a more attractive environment.

The main advantages to synchronous emulations is that they are perfectly-adapted to "transactional" type architecture (the forerunner of client/server), and offer excellent performance in packet-mode networks such as x.25 and x.32.

Because of the fundamental differences between 3270 emulation and the standard UNIX emulations described in this manual, ESKER decided to develop specific programs to perform this type of emulation.

The 3270 and 5250 emulators are used in a similar manner. Therefore all the information given in this chapter applies to both of them, unless the differences are explicitly stated.

### HLLAPI (only for 3270 emulation)

HLLAPI (High Level Language Application Interface) is a set of functions which makes it possible to program in high level languages such as C, Pascal, Basic and even COBOL.

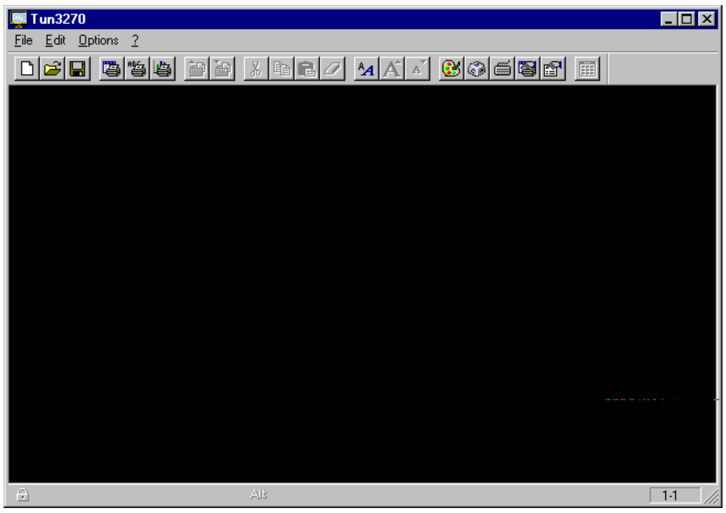
With HLLAPI functions, programs written by the user behave in the same way, as far as the host is concerned, as the user of a 3270 terminal. The HLLAPI interface identifies the functions and the data structures that are used, and carries out the operations defined in the user's program on the remote machine.

Each session, without exception, must be associated with a short name, that is to say a letter of the alphabet, to identify it uniquely. This means there can be a maximum number of 26 sessions. HLLAPI only works with one session at a time.

The HLLAPI delivered with **Tun 3270** is totally compatible with IBM EHLLAPI as well as Microsoft's WinHLLAPI, that is to say all the functions included in these interfaces are supported. The HLLAPI functions are incorporated in the DLL WHLLAPI.DLL.

## USING 5250 OR 3270 EMULATION

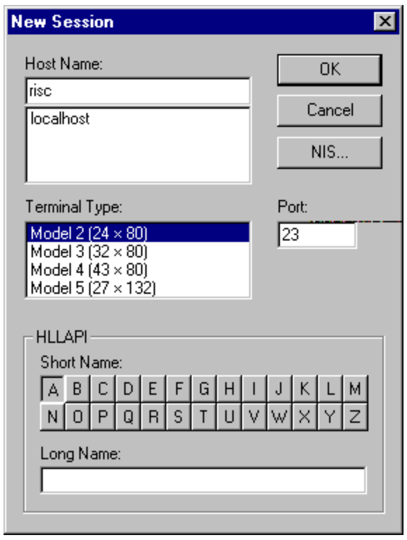
Run the program by clicking on the **Tun 3270** or **Tun 5250** icons in **Tun EMUL** group in Windows 3.x or in the Windows 95 Start menu.



*3270 emulation*

## OPENING SESSIONS

By selecting **File➔New Session** from the principal menu you may select the name or IP address of the host with which you wish to connect, and the terminal type:


New Session

Host Name: risc

localhost

Terminal Type:
Model 2 (24 × 80)
Model 3 (32 × 80)
Model 4 (43 × 80)
Model 5 (27 × 132)

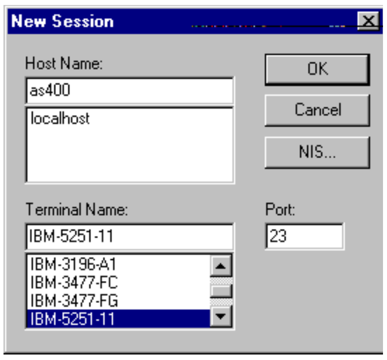
Port: 23

HLLAPI
Short Name:

| A | B | C | D | E | F | G | H | I | J | K | L | M |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| N | O | P | Q | R | S | T | U | V | W | X | Y | Z |

Long Name:

*3270 emulation*


New Session

Host Name: as400

localhost

Terminal Name: IBM-5251-11
IBM-3196-A1
IBM-3477-FC
IBM-3477-FG
IBM-5251-11

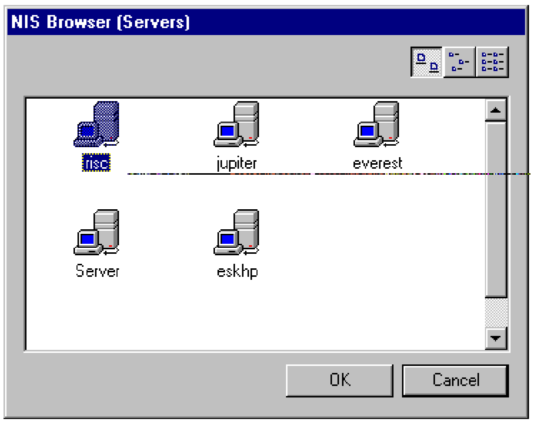
Port: 23

*5250 emulation*

### Host Name

Enter the host name or the IP address.

You can use the **NIS...** button to view the servers installed on the network. Refer to the section "**Tun EMUL and NIS**'.
The following window is displayed:



Double-click the icon of the desired resource to open the connection.

### Terminal Type (3270 emulation)

This field sets the type of 3270 terminal displayyou wish to use. There are four possible models:

- Model 2 (24 x 80)
- Model 3 (32 x 80)
- Model 4 (43 x 80)
- Model 5 (27 x 132)

### Terminal Name (5250 emulation)

This field sets the type of 5250 terminal display you wish to use. All of the options are able to handle 24 lines and 80 columns, but only three of them, IBM-3180-2, IBM-3477-FC and IBM-3477-FG, display 27 lines with 132 columns:

- IBM-3179-2
- IBM-3180-2
- IBM-3196-A1
- IBM-3196-A1
- IBM-3477-FC
- IBM-3477-FG
- IBM-5251-11
- IBM-5291-1
- IBM-5292-2

### Port

The default port is number 23. The user may enter a different number if necessary.

### HLLAPI (3270 emulation)

Clicking on a letter links an HLLAPI short name to the current session. A long name may be added in the field below.

## Starting sessions

After choosing a host and a terminal, click on **OK** to connect. If TCP/IP is running correctly on the PC and on the host, you should receive the sign-on screen.

## 3270 and 5250 use WINSOCK.DLL

**Tun 3270** and **Tun 5250** use the WINSOCK.DLL (Dynamic Link Library) standard interface to TCP/IP, and attempts to load it even if a search path is not specified. The WINSOCK.DLL path is automatically copied to TUN.INI during the installation process.

Win 95

Windows 95 automatically looks for the*.DLL* file and copies its path to the registry.

## USING THE EMULATORS

### Introduction

Esker's synchronous emulators include extensive revamping facilities which allow the user to customize the terminal. The possibilities include changing the color of the characters, changing the character style and borders, and using background images.

### Changing fonts

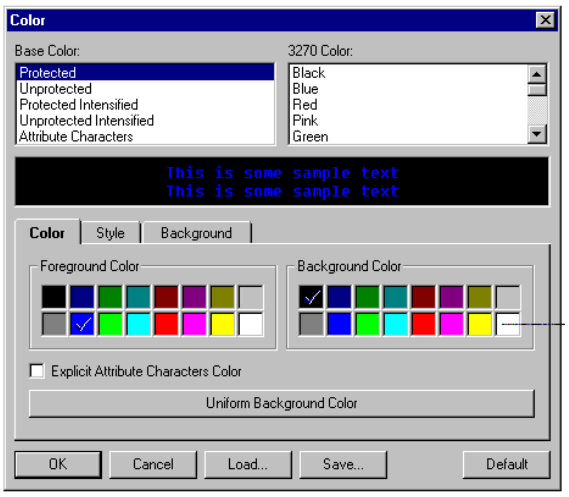
Any non-proportional character font available under Windows may be used in 3270 and 5250 emulation. The font **SystemPC** is furnished with **Tun EMUL** for the best possible use of emulation under Windows. It is a non-proportional OEM font developed especially for **Tun EMUL**. Use **Sys132PC** for emulation with 132 columns.

Synchronous emulators run with **Dynamic Sizing**. This means that the size of the font will always be adjusted in relation to the emulation window, thereby always showing a full terminal screen.

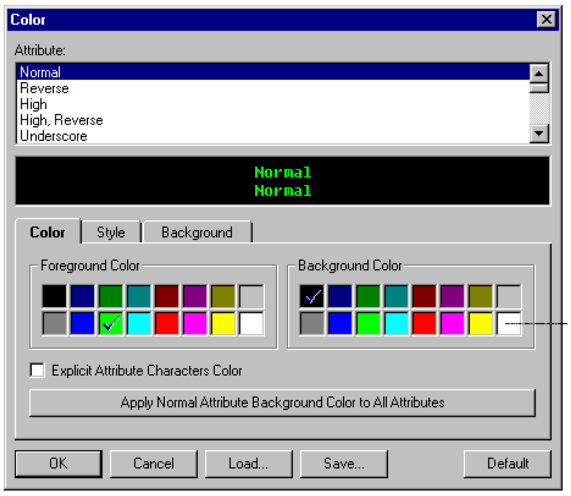
Use **Options➔Font...** or press the button in order to select a different font type or size. To increase or reduce the size of the font used, press the buttons or respectively. Notice, however, that this only works if the font used comes in different sizes (SystemPC, Sys132PC, Courier New, etc.).

### Changing colors

By default, the colors used in synchronous emulation are the same as those used by IBM on its own color terminals. You may assign different colors if you wish by using the menu option **Options➔Colors** or the button which displays the screen shown below. The top section of the screen relates to the host attributes.



*3270 emulation*



*5250 emulation*

The three tabs in the lower part of the screen allow the user to customize the screen attributes.

## Uniform Background Color

For 3270 emulation, click the button **Uniform Background Color** to assign the selected color to all the attributes.

For 5250 emulation, click the button **Apply Normal Attribute Background Color to All Attributes** so that all the attributes have the same background color, that of the normal attribute.

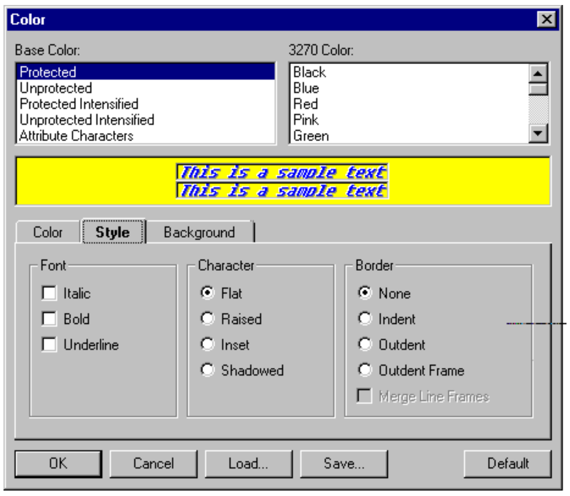
## Explicit Attribute characters

In a string of characters there are special characters, represented on the screen by a blank which contain information on the characters that follow (e.g. on the format of a character string). Normally, these characters are the same color as the background and therefore do not appear on the screen.

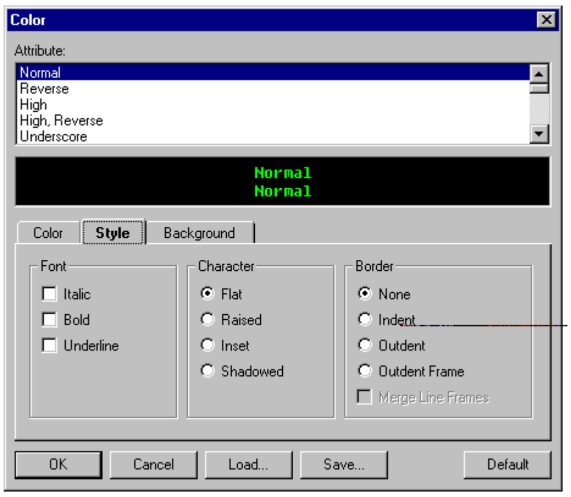
However, in both 3270 and 5250 emulation these characters can be made to appear in a different color to make them visible. To do this, assign the color of your choice to the attribute *Attribute Characters* (at the end of the **Attribute** list), and select the option **Explicit Attribute Characters Color** to apply the color to these characters.

## Style

Click on the Style tab to display the different styles available:



*3270 emulation*


Color

Attribute:
Normal
Reverse
High
High, Reverse
Underscore

Normal
Normal

Color | **Style** | Background

Font: Italic, Bold, Underline

Character: [✓ Flat] Raised Inset Shadowed

Border: [✓ None] Indent Outdent Outdent Frame Merge Line Frames

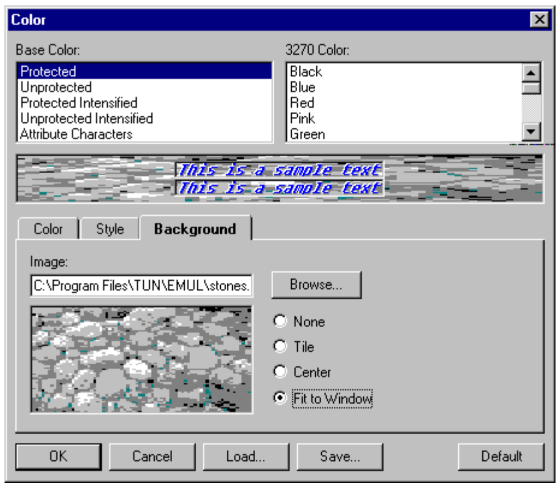
OK | Cancel | Load... | Save... | Default

*5250 emulation*

Enter the required font, character and border details. Check the results in the sample text.

## Background

The **Background** tab allows you to display an image on the screen during a particular session:


Color

Base Color:
Protected
Unprotected
Protected Intensified
Unprotected Intensified
Attribute Characters

3270 Color:
Black
Blue
Red
Pink
Green

This is a sample text
This is a sample text

Color | Style | **Background**

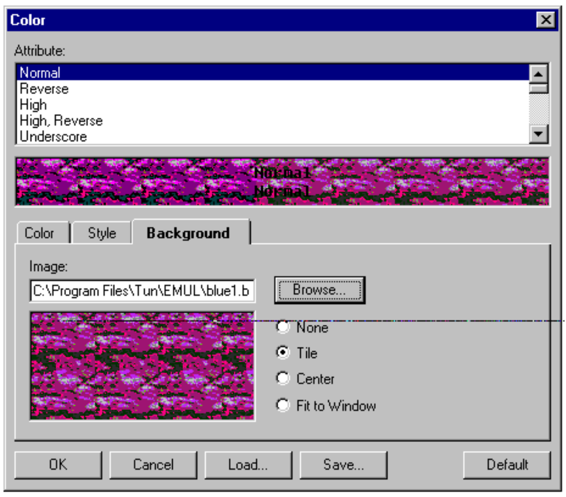
Image: C:\Program Files\TUN\EMUL\stones.

Browse...

None Tile Center [✓ Fit to Window]

OK | Cancel | Load... | Save... | Default

*3270 emulation*


Color

Attribute:
Normal
Reverse
High
High, Reverse
Underscore

Normal
Normal

Color | Style | **Background**

Image: C:\Program Files\Tun\EMUL\blue1.b

Browse...

None [✓ Tile] Center Fit to Window

OK | Cancel | Load... | Save... | Default

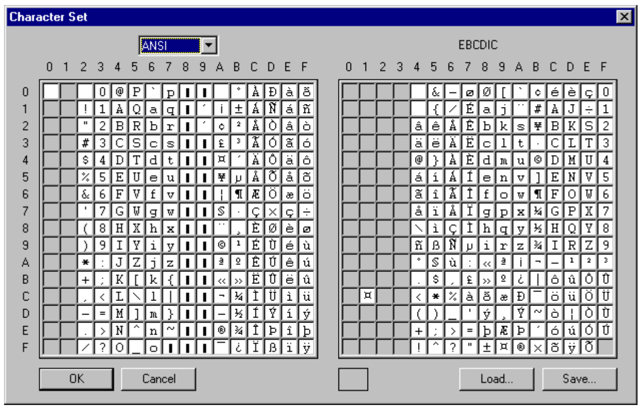
*5250 emulation*

Use the Browse... button to select the bitmap image you wish to use as a background. The image may be tiled on the screen, centered or made to fit the entire window.

The options available in the **Color** dialog box can be saved for future use. Press the **Save...** button and give the file an appropriate name. Use the **Load...** button to load a previously saved file from the standard directory browser window.

## Modifying the character table

Synchronous emulation uses the EBCDIC extended character set, which can vary from country to country. If the current table does not meet your particular needs, you may change it using **Options➔Character Set...** or the button . This will open the window shown below:

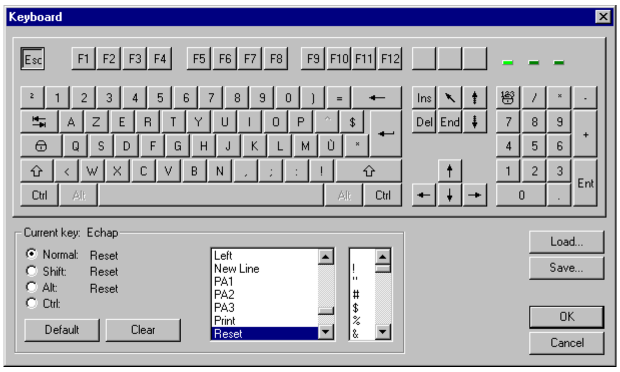


The current EBCDIC table is shown on the right, and the character set available on the PC is on the left. Notice the small window above the PC character set giving the choice between the ANSI and OEM character sets. Only the EBCDIC table may be modified.

To change a character, click on the character you wish to modify in the PC character set, and then drag it across to the position at which you would like to place it in the EBCDIC character table. You may record any changes by clicking on**Save**.

## Remapping the keyboard

The parameters used by the keyboard in 5250 and 3270 emulation are the same as those typically used by true IBM terminals. However, if you do need to remap the keyboard, click on **Options➔Keyboard...** or press the button  which displays the following screen:



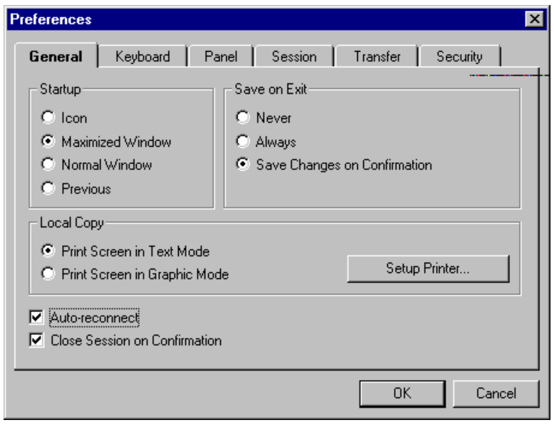
Current key: Echap

[✓ Normal: Reset]
Shift: Reset
Alt: Reset
Ctrl:

You may change any key by first clicking it, then selecting a value from one of the two tables at the bottom of the window. The check boxes **Normal**, **Shift**, **Alt** and **Ctrl** allow you to define the different levels of key-combinations. Redefinition of the **Ctrl** key will only take effect if the check boxes **Left Ctrl Key** and/or **Right Ctrl Key** are selected in the **Preferences** dialog box (in the **Keyboard** tab).

## Preferences

The Preferences dialog box (displayed by choosing the menu option **Options➔Preferences**) shown below can be used to change **General**, **Keyboard**, **Panel**, **Session**, **File Transfer** and **Security** options:



Preferences

General | Keyboard | Panel | Session | Transfer | Security

Startup
- Icon
- [✓ Maximized Window]
- Normal Window
- Previous

Save on Exit
- Never
- Always
- [✓ Save Changes on Confirmation]

Local Copy
- [✓ Print Screen in Text Mode]
- Print Screen in Graphic Mode

Setup Printer...

- [✓ Auto-reconnect]
- [✓ Close Session on Confirmation]

OK | Cancel

### General

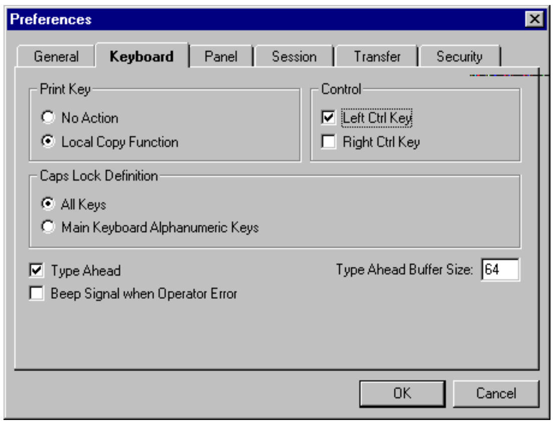
The **General** tab is used to define window options on **Startup** and to set the save options on **Exit**.

The **Local Copy** options define which mode the screen will be printed in if the option **Local Copy Function** is chosen on the **Keyboard** tab for the **Print Key**. The hard copy may be requested by a host application or by the operator.

If the **Auto Reconnect** check box is selected, this means that the program will automatically re-establish the connection if it is interrupted by the host. Selecting the **Close Session on Confirmation** check box will display a confirmation dialog box on exiting the program.

### Keyboard

Use the **Keyboard** tab to define such options as the use of the print key, the use of the control keys, and the **Type Ahead Buffer Size** The latter field is only active if the **Type Ahead** option is selected.



Preferences

General | Keyboard | Panel | Session | Transfer | Security

Print Key
- No Action
- [✓ Local Copy Function]

Control
- [✓ Left Ctrl Key]
- Right Ctrl Key

Caps Lock Definition
- [✓ All Keys]
- Main Keyboard Alphanumeric Keys

- [✓ Type Ahead]
- Beep Signal when Operator Error

Type Ahead Buffer Size: 64

OK | Cancel

Notice that the first option in the **Print Key** section applies the normal operation of a 5250 terminal for that key. The second option activates the **Local Copy** option chosen on the **General** tab.

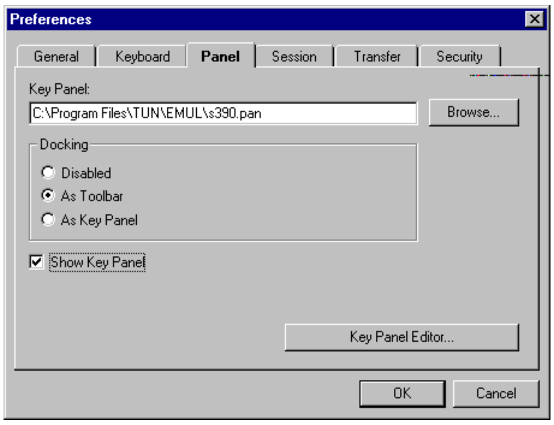
**Caps Lock Definition** determines whether the Caps Lock key will be allowed to effect the whole keyboard or simply the main alphanumeric keyboard, excluding the numerical keypad. In this way, the symbolic keys on the keypad, Up, Down, Home, etc., are freed for other purposes.

The operator may find it useful to select the checkbox **Beep Signal when Operator Error**. For example, if the operator presses the Enter key when the cursor is not in the right place, he will be alerted by a beep.

## Panel

The third tab, **Panel**, can be used to assign a function-key panel to the active session.

The key panel may be treated as a dockable toolbar, a key panel proper or may be disabled. It will not be displayed unless the check box **Show Key Panel** is selected.



Preferences

General | Keyboard | Panel | Session | Transfer | Security

Key Panel: C:\Program Files\TUN\EMUL\s390.pan [Browse...]

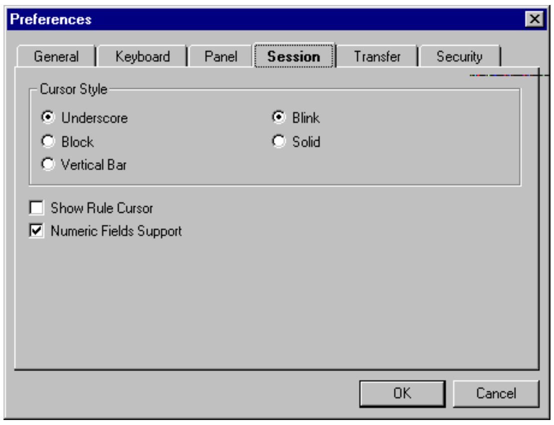
Docking: Disabled [✓ As Toolbar] As Key Panel

[✓ Show Key Panel]

The button **Key Panel Editor** opens the Function-key Panel Editor for the creation or modification of a function-key panel.

## Session

Define **Cursor** style and **Numeric Fields Support** on the fourth tab:



Preferences

General | Keyboard | Panel | Session | Transfer | Security

Cursor Style: [✓ Underscore] Block Vertical Bar — [✓ Blink] Solid

Show Rule Cursor

[✓ Numeric Fields Support]

Note that if the check box **Show Rule Cursor** is selected, the line at which the cursor is positioned is underlined to facilitate, for example, the viewing of a dense text file.

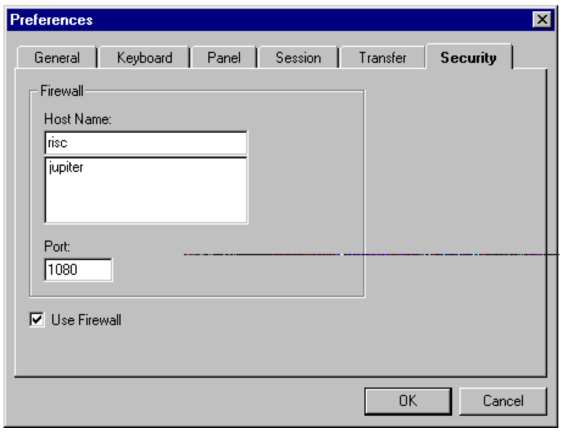
In 3270 emulation, check the box **Numeric Fields Support** if you wish to restrict user input to numeric characters. If the check box is left unselected, any alphabetical character may be entered in numeric fields.

## File Transfer (only for 3270 emulation)

The fifth tab, **File Transfer**, defines the settings for the transfer of files to and from different types of environments. Since the information on this tab is essentially concerned with the sending and receiving of files, please refer to the section **Sending and Receiving Files** below.

### Security

The **Security** tab allows the user to enter system security details.

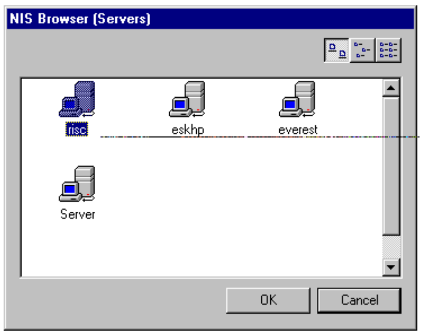


Firewall
Host Name: risc
jupiter
Port: 1080
☑ Use Firewall

Enter the name of the proxy server which acts as the firewall in the **Host Name** field and change the **Port** number if necessary. Data transmission will then be protected by the safety mechanism of the proxy server.

You can use the **NIS...** button to view the servers installed on the network. Refer to the section "**Tun EMUL and NIS**'.

The following window is displayed:



Double-click the icon of the desired resource to enter it as the firewall.

## Cut & paste options

You may use the mouse to select any zone on the screen for copying to the windows clipboard (using **Edit➔Copy**).

To place the contents of the clipboard onto the screen, use**Edit➔Paste**.

You may also select a zone with the mouse and then erase it using**Edit➔Delete**.

It is also possible to carry out these operations for the whole screen and not only a selected part of it by using the option **Edit➔Select All** from the main menu.

## Hard copy

This option may be used to print the contents of an emulation session. By selecting **File➔Print Text...** or **File➔Print Graphic...**, you may use **Text** mode (faster), or **Graphic** mode, which transforms colors into different shades of gray.

It is also possible to print the whole screen by selecting the option**Edit➔Select All** from the main menu.

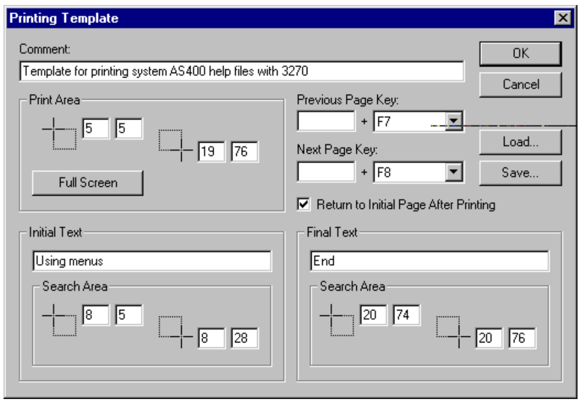
## PRINTING WITH TEMPLATES

It used to be the case that the user had to print each emulation screen page individually if he wanted a hard copy. For long document files, this could require an enormous amount of time. With **Printing with Templates**, the user indicates:

- the screen zone he wishes to print, thus omitting server-supplied information displayed at the bottom of each emulation screen, such as command keys,etc.
- the starting text, i.e. the character string at which printing will begin
- the final text or terminating string, at which the emulator will stop printing

The template may be saved and re-used.

To print with a template, select the option **Options➔Printing Template** or click the button in the toolbar to display the following dialog box:


Printing Template

Comment: Template for printing system AS400 help files with 3270

Print Area: 5 5 — 19 76

Previous Page Key: + F7

Next Page Key: + F8

☑ Return to Initial Page After Printing

Initial Text: Using menus — Search Area: 8 5 — 8 28

Final Text: End — Search Area: 20 74 — 20 76

The print area is identified by the cursor coordinates in the section**Print Area**.

- Enter an optional comment. This is useful since it will appear in the**Print With Template**dialog box.
- Enter the coordinates for the screen area to be printed in the order "top left" and "bottom right". Click on the screen at the top left-hand corner of the area you wish to print and note the cursor coordinates on the right-hand side of the status bar.
- Click the bottom right-hand corner of the area to print and note the coordinates. Enter them in the dialog box.
- Enter the **Previous Page Key** and the **Next Page Key** in the appropriate fields. These are **system-dependent** and are often shown at the bottom of the emulation screen where the commands are displayed. If they are not shown at the bottom of the emulation screen page, you should consult the documentation for the type of host you are connecting to. The codes for these keys are sent to the host which reacts accordingly. If these are not entered, only the current page will be printed.

  Note that the system might require that an alphabetical key be pressed with the enter key. In this case use the extra field provided for this purpose (the first field).

- If the check box **Return to Initial Page After Printing** is selected, the program will return to the initial page when printing is completed.
- In the **Initial Text** and **Final Text** fields, enter the text which the emulator will consider as marking the first and last pages to be printed. In the respective **Search Area** boxes, enter the coordinates of the screen zone where the emulator will search for the initial and final character strings.

**Note that all the coordinates may be entered more simply by clicking on the position icon (the square formed with a dotted line) and dragging the cross-hairs to the required position on the emulation screen and releasing the mouse button. It might be necessary to move the dialog box first.**

Also, the Full Screen button may be used to apply the coordinates of the full screen.

Press **Save** to open the **Save Printing Template** dialog box and save the file under a name of your choice.

## Printing with a template

To print using a printing template, load the template using the dialog box described above and click **OK**. Use the option **File➔Print with Template** or click the button in the toolbar. The **Print with Template** dialog box which opens has the look and behavior of a normal Windows print dialog box with one or two differences.

Instead of a print selection option, there is a **Current Page to End** option which prints from the current page to the **Finishing Text** indicated by the template. This may be the end of the file or an intermediary point.

Also, the template may be changed at the time of printing by using the **Browse...** button in the **Printing Template** section and loading a different template file (extension **ptp**).

The printout consists of sheets containing the emulation screen pages without any unnecessary emulation screen information.

## Saving sessions for re-use

You may save all the parameters defined for the current session in a .CFS file by using **File➔Save**. Doing this will record information such as the host name, fonts and colors. Initialization files may then be reloaded at any time by using **File➔Open**, allowing users to find a familiar terminal every time they enter emulation.

To open an emulation session directly with a corresponding initialization file, use the following command:

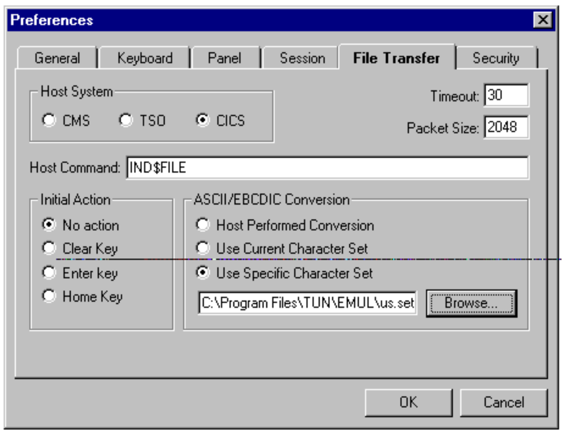
```
C:\TUN\EMUL\3270.EXE -c session.cfs
```
or
```
C:\TUN\EMUL\5250.EXE -c session.cfs
```

## Closing the current emulation session

Click on **File➔Close** to end the current emulation session.

## SENDING AND RECEIVING FILES (ONLY FOR 3270 EMULATION)

In order to use the **Tun 3270** file transfer option, the IBM file transfer program IND$FILE must be installed on the mainframe. To specify general file transfer options, use the **File Transfer** tab in the **Preferences** dialog box:



Host System: CMS TSO [✓ CICS]
Timeout: 30
Packet Size: 2048
Host Command: IND$FILE
Initial Action: [✓ No action] Clear Key Enter key Home Key
ASCII/EBCDIC Conversion: Host Performed Conversion Use Current Character Set [✓ Use Specific Character Set]
C:\Program Files\TUN\EMUL\us.set

The following options are available:

### Host System

Defines the type of host operating system you will be connecting to.

### Timeout

Indicate the length of time the program will wait for a response from the host to the file transfer request. The default is 30 seconds.

### Packet Size

The default packet size is 2040 bytes. The speed of the file transfer is directly proportional to the size of the packet.

### Host Command

Indicate the command that will handle the transfer. The default is IND$FILE.

### Initial Action

Defines the initial action to be provoked before the transfer is started. Hence, it is possible to clear the terminal screen, if the **Clear** key is set, or send the Enter or Home key code first.

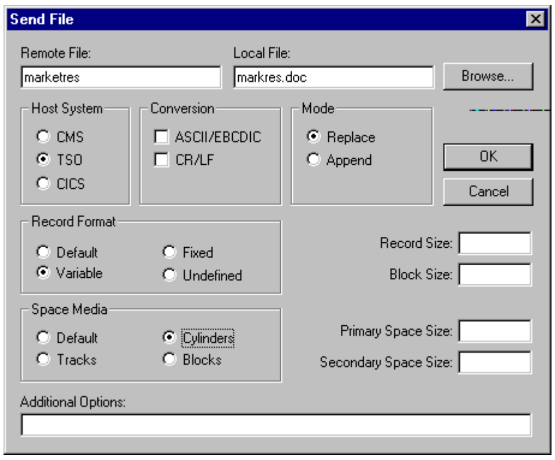
### ASCII/EBCDIC Conversion

Choose the appropriate option for the file transfer in preparation. **Host Performed Conversion** means characters will be converted to the host code page by the host; **Use Current Character Set** converts received characters to the character set loaded; this is carried out by the PC. A different character set may be chosen by browsing the **.set** files, thus making allowance for the differences in national character sets.

To send or receive a file, use either the menu options in the **File** menu or click on the buttons ( or ) in the toolbar.

## Sending

If the **File➔Send File...** option is chosen, the following dialog box will be displayed:



Remote File: marketres
Local File: markres.doc
Host System: CMS [✓ TSO] CICS
Conversion: ☐ ASCII/EBCDIC ☐ CR/LF
Mode: [✓ Replace] Append
Record Format: Default [✓ Variable] Fixed Undefined
Record Size:
Block Size:
Space Media: Default Tracks [✓ Cylinders] Blocks
Primary Space Size:
Secondary Space Size:
Additional Options:

Perform the following steps to send a file:

- Enter the local and remote names of the file to be sent.
- Select the host environment of the remote machine.
- Set the conversion type: choose ASCII/EBCDIC to effect the conversion to EBCDIC, and CR/LF to replace CR/LF codes with end of record codes.
- Choose whether the file will replace an existing file of the same name or will be appended to it.

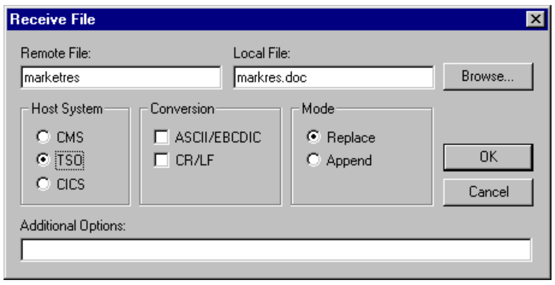
For file transfer to a TSO or CMS host, complete the newly activated fields as necessary. In the case of the **Record Format** options, the default option causes the record length to be controlled by the host environment's defaults. Otherwise, select **Variable** or **Fixed** for a CMS host environment, or **Variable**, **Fixed** or **Undefined** for a TSO host environment.

For a TSO host environment, complete the fields in the **Space Media** section. If either **Tracks** or **Cylinders** is chosen as the unit of measure, enter the required number in the **Primary Space Size** box; use the **Secondary Space Size**box to allocate additional units to the data set when the primary space is filled.

If you are unsure of the values to enter for the different sizes, contact your system administrator.

## Receiving a file

Clicking on the **Receive** button or activating the menu option (**File➔Receive...**) displays the following dialog box:



Remote File: marketres
Local File: markres.doc
Host System: CMS [✓ TSO] CICS
Conversion: ☐ ASCII/EBCDIC ☐ CR/LF
Mode: [✓ Replace] Append
Additional Options:

Perform the following steps to download a file:

- Enter the names of the Remote and Local files. An existing local file of the same name will be replaced.
- Choose the type of host environment in the System section.

- In the Conversion section, select the ASCII/EBCDIC check box to perform the conversion to ASCII and the CR/LF to add the carriage return and line feed characters.
- For Mode, choose Replace to replace the file named in the field Local File or Append to add the data to the end of the file.

### Additional Options

Additional options may be defined when both sending and receiving files. Enter any additional IND$FILE parameters that may not have been catered for in the dialog box options.

## Command buttons

It is possible to display the command buttons at the top of the emulation window to accelerate access to the most frequently used commands (size of font, colors). To do this, use the option **Options➔Toolbar**:

Opens a new emulation session and closes the current session if there is one.

The user may load all the settings used in a session (host address, colors, character set, keyboard, etc.) from a setup file with a .cfs extension.

The user may save all the settings used in a session (host address, colors, character set, keyboard, etc.) in a single file with a .cfs extension.

Prints screen page or pages with template.

Prints the screen or a selected area of the screen in text mode.

Prints the screen or a selected area of the screen in graphic mode.

Displays the Send dialog box.

Displays the Receive dialog box.

Cuts the selection highlighted with the mouse and copies the contents to the clipboard.

Copies to the clipboard the selection highlighted using the mouse.

Copies the contents of the clipboard to the cursor position.

Deletes the selection highlighted using the mouse.

Selects the screen display font.

Selects the next font size up to the current one. For this option to be possible, the current font must exist in different sizes.

Selects the next font size down to the current one. For this option to be possible, the current font must exist in different sizes.

Associates a color with each of the attributes used during emulation.

Allows the modification of the correspondence table for the PC and extended EBCDIC character sets.

Allows the user to change the values returned by the keyboard.

Allows the user to print with a template.

Allows the user to modify the preferences.

Displays the function-key panel

## Dockable Toolbar

Win 95

The toolbar for the Windows 95 version of the program is dockable, i.e. it can be positioned on the side or bottom of the screen.

## Mouse Menu

The right button of the mouse can be used to display a menu with, notably, options for changing or viewing **Session**, **Field** and **Character Properties**. **Cut**, **Copy** and **Paste** operations can also be carried out using this menu.

## Function-key Panels

Function-key Panels are now available for the **Tun 3270** and **Tun 5250** emulators. The panels can be prepared using **Tun PANEDIT**and loaded from disk. These floating panels are also dockable.

# CHAPTER 5 - FUNCTION-KEY PANELS

## INTRODUCTION

A function-key panel is a window consisting of buttons which the user can summon during an emulation session. If the user clicks on one of the buttons, one of the following operations is executed:

- Transmission of a pre-defined string of characters on the communication channel (simulates a keyboard entry).
- Execution of a macro function.
- Execution of a specific Tunaction (launch an application, make a hard copy...).

The key panel buttons may contain:

- Text (centered, left or right justified, on several lines).
- An image (centered, or tiled).
- An image and text.

Just like the keys on the keyboard that are affected by the Shift or AltGr keys, the buttons on a key panel can have three levels. Each button level can display a different value and carry out a specific operation. The level is defined by a special button called the **Lock** button.

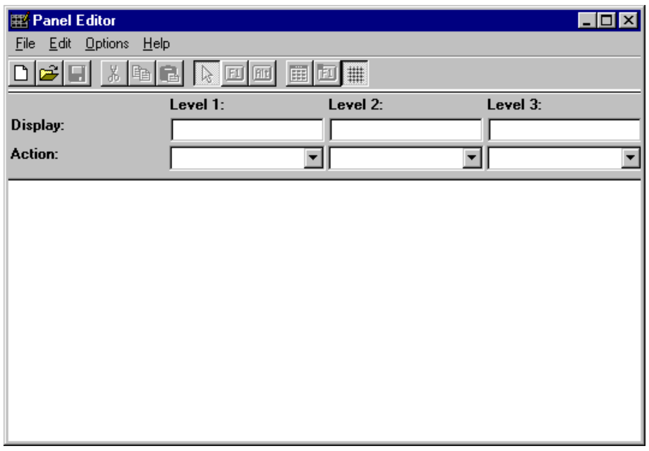
The purpose of the key panel is to replace the keyboard with the mouse as much as possible. With well-designed key panels, traditional UNIX operations may offer the same ease of use as native Windows applications.

The aim of the present chapter is to explain how to design and define a function-key panel.

## CUSTOMIZING A FUNCTION-KEY PANEL

To create a function-key panel, click on the **Panel Editor** icon in **Tun EMUL** group in Windows 3.x or in the Windows 95 Start menu.

The function-key panel editor displays the following window:



After creating or loading a new function-key panel by using the options **File➔New** or **File➔Open...**, a model key panel appears. It shows the dimensions of the key panel. During the emulation session, the key panel will take the relative size and position of that shown in the panel edit window. It is possible to change the size and position of the panel edit window using the mouse and the drag handles.

# CREATING A PANEL

A key panel needs at least one button for it to be functional.

## Creating a button

There are two types of button:

- Simple buttons which control three levels (just like the keys of an ordinary keyboard)
- **Lock** buttons which change the state of all the simple buttons on the same key panel.

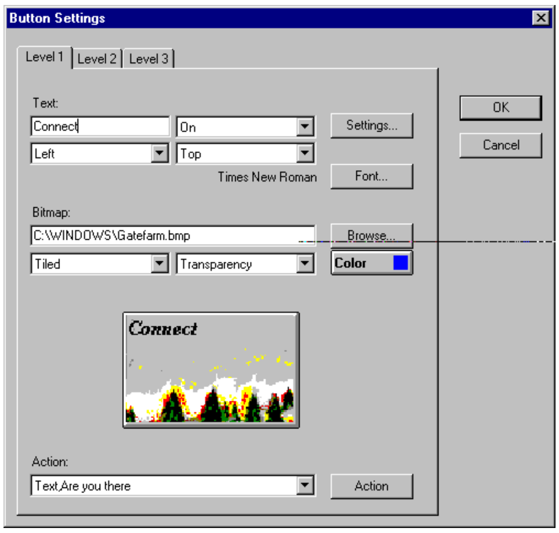
There can only be one **Lock** button per key panel.

## Simple button

To create a simple button, carry out the following operations:

- Create a new file (**File➔New**).
- Press the button [F1] on the toolbar in the principal window.
- Click in the key panel definition window and slightly displace the mouse with the button held down to draw the size and shape of button required.
- Release the mouse button.

This series of operations creates a blank button on the screen. Now properties have to be linked to the button. To do this, double-click on the blank button to display the following dialog box:



Button Settings

Level 1 | Level 2 | Level 3

Text: Connect | On | Left | Top | Times New Roman

Bitmap: C:\WINDOWS\Gatefarm.bmp | Tiled | Transparency

Action: Text,Are you there

The dialog box for the definition of simple buttons has three levels indicated by the tabs Level 1, Level 2 and Level 3. Each level represents one of the three possible states the button may take. The same information is requested at each level.

### Text

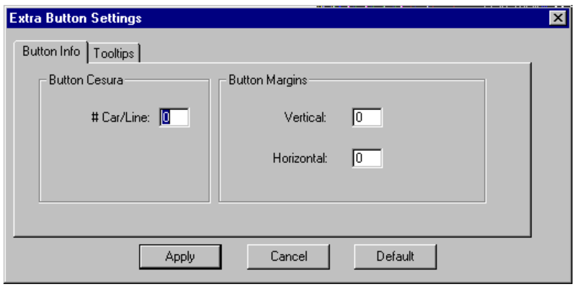
The field **Text** contains the text to be displayed on the button itself. Type the required text into the blank field below the field label **Text**. Use one of the options in the box immediately below the new text field to position the text horizontally, i.e. to center the text or to left- or right-justify it. The box to the right of the new text contains three options to position the text in relation to a bitmap (if one is used on the button).

The text may be placed above the bitmap, below it or directly on the bitmap. The last text option box (Top, Center, Bottom) defines the vertical position of the text. If the text has been placed above or below a bitmap, then the vertical area is reduced and there is little room to maneuver the text vertically.

Character style, font and color can also be changed by pressing the button **Font...**. If there is no text associated with a button, it will remain shaded during the display of the function-key panel. The current font is displayed to the left of the button.

## Settings

The **Settings...** button opens the **Extra Button Settings** dialog box which allows the user to define the number of characters per line allowed on the button, and the margins for the text on the first tab shown **Button Info**:



# Car/Line: 0
Vertical: 0
Horizontal: 0

On the tab **Tooltips**, enter the text which you like to see displayed as a tooltip for the button.

## Bitmap

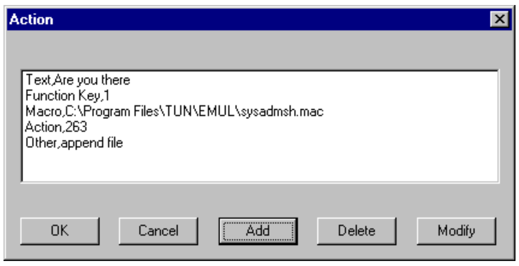
An image of the **paintbrush** type (.BMP) may be shown on the button if the full path name is entered in the field **Bitmap**. To select the image, use the button **Browse...**. The image may be centered on the button or duplicated so that it occupies all the available space on the button by pressing the appropriate button in the **Button Settings** dialog box, **Center** or **Tiled**. If both an image and text are linked to a button, the position of the bitmap will depend on the options chosen for the text. If **On** has not been chosen for the relative position of the text, the image occupies three-quarters of the button space and the text the remainder.

## Color

Use the **Color** button, if desired, to apply a color mask to the bitmap (**Transparency**). The default is **None**. The **Color** button becomes active when the option **Transparency** is chosen. Press it to display a color dialog box from which to choose the mask color.

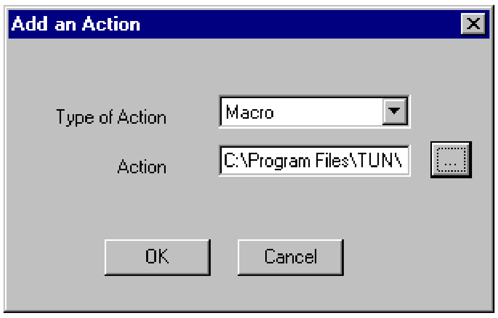
## Action

The field **Action** indicates which string of characters is to be emitted or which action is to be taken when the user presses the button. Pressing the button displays the following dialog box:



Text,Are you there
Function Key,1
Macro,C:\Program Files\TUN\EMUL\sysadmsh.mac
Action,263
Other,append file

Press the **Add** button to add one of the five possible types of action,**Text**, **Function Key**, **Macro**, **Action** or **Other**, to the button:



Type of Action: Macro
Action: C:\Program Files\TUN\

- For a button to send a string of characters, select the Type Of Action **Text**; you may use different mnemonics as well as the coding used for the configuring of the keyboard (fkey1, fkey2...).
- For a button to send a Function Key code, choose **Function Key** in the Type of Action field and put the number of the keyboard F key in the**Action** field.
- If you want the button to execute a macro (see the chapter "Emulator Automation"), the corresponding command field (for the Type Of Action **Macro**) must contain the name of the macro to run.(e.g. C:\\TUN\\EMUL\\AUTO.MAC).
- If you wish the button to start off a **Tun EMUL** action (see Action.doc), the corresponding command field (**Action**) should contain the number of the action (e.g. 51 to clear the screen).
- The Type of Action **Other** is only available for commands used in earlier versions of asynchronous emulators.

Actions may be modified or deleted.

## Lock button

To create a **Lock** button, carry out the same operations as for a simple button but first of all click on the button in the toolbar of the principal window. In the same way as for a simple button, properties have to be associated with the lock button by clicking on it. The dialog box displayed is the same as for a simple button except that no **Action** is required. The sole purpose of the **Lock** button is to change the state of the other buttons on the function-key panel.

The **Lock** button will only work in the following circumstances:

- A text or a bitmap is placed on the button for each of the three levels.
- A text or a bitmap is placed on the **Lock** button for level 3 only. This indicates to the program that the other two levels are also active.

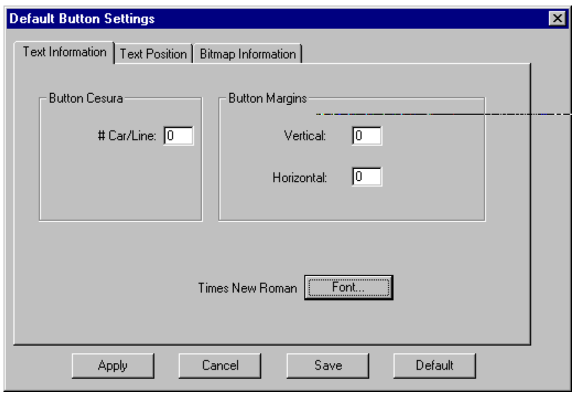
The use of a lock button in a key panel divides the number of buttons otherwise necessary by three.

## Additional buttons

Up to 128 buttons may be contained on the one key panel.

## Default Button Settings

The user can define default button settings (**Options➔Default Button Settings...**) which will be used automatically by the **Button Settings** dialog box (displayed by double-clicking on the prototype button). The default values can always be modified as necessary when a button is being defined.


Default Button Settings

Text Information | Text Position | Bitmap Information

Button Cesura
# Car/Line: 0

Button Margins
Vertical: 0
Horizontal: 0

Times New Roman

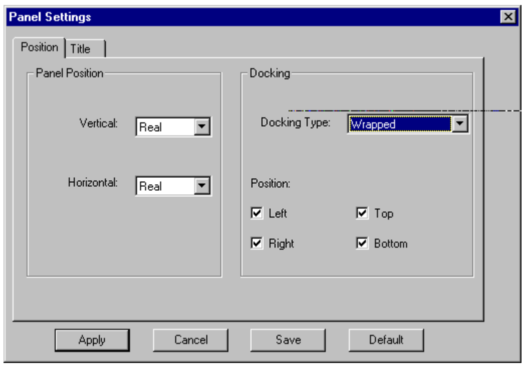
The information contained on the first tab, **Text Information**, may be changed by using the **Settings...** button in the **Button Settings** dialog box. If the number of characters per line field, **Char/Line**, contains the value 0, the text on the button will be wrapped.

For all three tabs, pressing the **Save** button will register the values as the default values. The values are only applied to newly created buttons. The**Default** button recalls the previously saved values.

## Panel Settings and Positioning

### Positioning a panel

Use the option **Options➔Panel Settings** to define the initial panel position, docking options and title format. On the **Position** tab, set the initial panel position. The Vertical and Horizontal option **Real** corresponds to the original position of the panel in the panel editor screen (default).


Panel Settings

Position | Title

Panel Position

Vertical: Real

Horizontal: Real

Docking

Docking Type: Wrapped

Position:

☑ Left ☑ Top

☑ Right ☑ Bottom

### Docking

Select the type of docking you wish to apply to the function-key panel:

- **None:** the function-key panel cannot then be transformed into a toolbar.
- **Normal:** the function-key panel becomes a normal toolbar.
- **Wrapped:** if there are too many keys, they are re-organized on more than one line or column.
- **Default:** the function-key panel is changed into a toolbar, but the layout of the keys remains the same.

The type of docking selected here corresponds to the **Default** option proposed by the emulator when the function-key panel is selected. Refer to the chapter on the definition of the display settings for an asynchronous emulation session.

### Title Bar

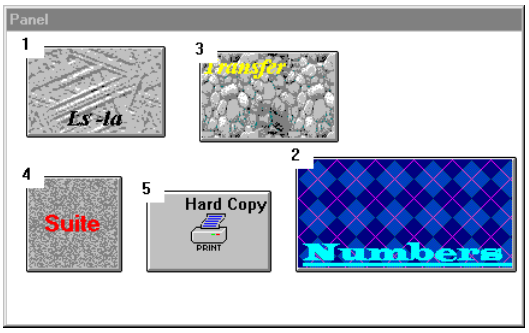
The **Title** tab is used to define the format of the panel Title Bar:**Normal**, **Half-height** or **None**.

Test the panel to see the results of the various options.

### Set Tab Order

The tab order controls the numerical order in which the buttons are selected or highlighted when the user presses the Tab button. The Tab order of the buttons may be set through the option **Options➔Set Tab Order**. This assigns to each of the buttons a number (not displayed) which may be used later if the button is modified. For example, Action 291 in **Tun EMUL**, which changes the text of a button, takes two parameters; the first is the tab order number and the second the new text.

To assign tab order to a panel of buttons, select the option **Set Tab Order** and click on the panel button which is to be number one with the *right* mouse button. Then click on the remaining buttons with the *left* mouse button in the desired tab order. The order may be changed by recommencing the operation. When you are satisfied with the order, deselect the menu option.



Lock buttons are always take the same number as the number of buttons.

## TESTING A KEY PANEL

To effect an immediate test of the key panel without having to load it into the terminal emulator, press the button ▦ in the toolbar or use the menu option **Options➔Test Mode**. This function simulates the use of the key panel, just as if it were being used in the terminal emulator. To stop the test, press the same button again ▦.

## SUPPLEMENTARY FUNCTIONS OF THE PANEL EDITOR

### Moving and changing the size of a button

To move a button or change its size, select the button with the mouse. This displays dimensioning handles at each corner. Click on the inside of the button and move the mouse with the mouse button held down to move the panel button correspondingly.

It is possible to select several buttons at the same time by pressing the button ↖ in the toolbar and then drawing a rectangle big enough to contain them. All the buttons totally or partially contained in the rectangle thus drawn will be selected. Several buttons may thus be moved, for example, in one operation.

### Duplicating a button

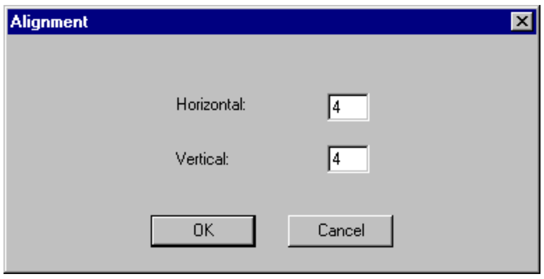
To duplicate a button, select it and move the mouse while holding down the **Ctrl** button on the keyboard. On releasing the mouse button, a new panel button similar to the selected one will appear. All the data linked to the button, text as well as bitmap, are copied.

#### Copy and Paste

It is also possible to duplicate a button using the **Copy** and **Paste** buttons on the toolbar. The button ⧉ copies the button, selected with the mouse, to the clipboard. The button 📋 places the copy in the panel next to the original. All the data linked to the button, text as well as bitmap, are copied.

## Aligning the buttons

The menu option **Options➔Align** was devised to help the user align the buttons. Click on the option, then select **Active** to activate the function and **Options...** to choose new displacement values in the **Alignment** dialog box:



Horizontal: 4

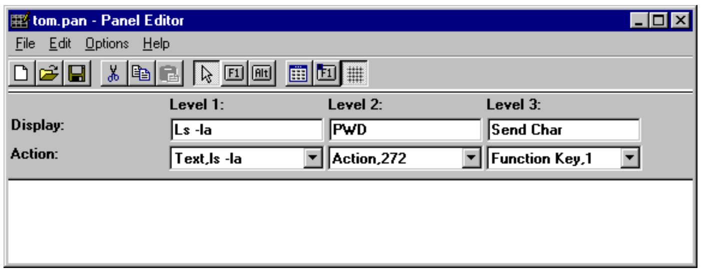
Vertical: 4

Alignment is also rendered active by clicking on the button ▦ in the toolbar. By default, the newly created buttons are placed in a position which corresponds to a multiple of 4 pixels. The width and length of the buttons are also multiples of four. This constraint facilitates the alignment of the buttons in relation to each other. The dimensions of new buttons are multiples of the alignment values. To cancel this constraint click on ▦.

Select a button and holding the mouse button down, move it gently vertically or horizontally. The button will move the exact number of pixels entered in the **Horizontal** and **Vertical** fields of the **Alignment** dialog box.

## Changing the description of a button quickly

To change the descriptions and actions of a button, it is possible to update the table at the top of the panel editor screen. Enter the new **Display** and **Action** values for the level in question:



| | Level 1: | Level 2: | Level 3: |
|---|---|---|---|
| Display: | Ls -la | PWD | Send Char |
| Action: | Text,ls -la | Action,272 | Function Key,1 |

## MOUSE MENUS

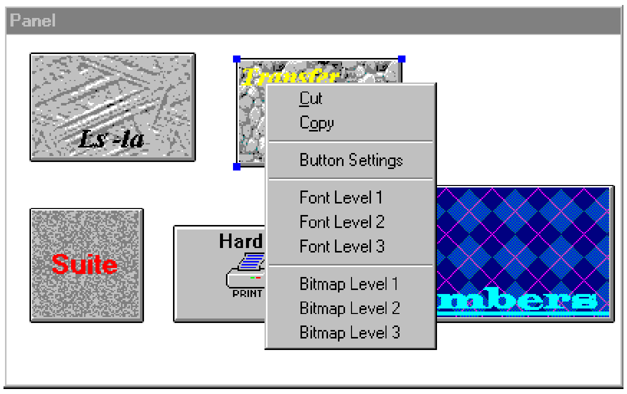
Most of the above options are accessible through mouse-driven menus.

### Panel Settings

Click on an empty space in the prototype panel with the right mouse button to access the **Panel Settings** dialog box.

### Button Settings

To display the button menu, first of all select the button by clicking on it with the left mouse button. Then click once with the right mouse button. If you click directly on a button with the right mouse button, keep the mouse button pressed.



### Toolbar

The button functions are described below:

Opens a new panel file and closes the current panel file if one is open.

Opens an existing panel file.

Saves the panel file. The default extension, which it is convenient to use, is .pan.

Cuts the selected button and copies it to the clipboard.

Copies the selected button to the clipboard.

Copies the button in the clipboard to the current panel. The copy is placed in the same place each time. The user must then position it as he wishes.

Normal selector arrow. Can be used to select one or more buttons in the prototype panel.

Button arrow. Press to draw a new button.

Lock button arrow. Use to draw a lock button. Only one lock button is allowed per panel.

Toggles test mode on and off.

Sets tab order. Click on the first button with the right mouse button and number the remaining buttons by clicking on them with the left button.

Activates alignment.

## LINKING A FUNCTION-KEY PANEL TO AN EMULATION SESSION

### Tun EMUL

To associate a particular function-key panel with an emulation session, start the emulator and then link the key panel configuration file to the session using the options **Settings➔Display Settings➔Function-key Panel**. A key panel may be summoned or removed by pressing the button in the toolbar.

### Tun 3270 and Tun 5250

Choose the panel required in the **Options➔Preferences...** dialog box on the **Panel** tab. Once chosen, the panel may be displayed or removed by using the menu option **Options➔Panel**.

# PART II
# ADVANCED USE OF THE ASYNCHRONOUS EMULATOR

# CHAPTER 6 - 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" (e.g. 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 supply them as parameters on the command line when starting the emulator. Refer to chapter '**Program Syntax**'.

## SAMPLE MACRO

The macro given below as an example automates access to a 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.25PAD, 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. There is only one instruction per line. The first word in the 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 (e.g. "*SendAndReceive* may be written "*SENDANDRECEIVE*" or "*sendandreceive*").

If the first character of a line is "#", the entire line will be treated as 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 (e.g. "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" "%tata").

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 if 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 (only asynchronous emulation) |
| **ClearMessage** | Erase messages in a dialog box |
| **ClearScreen** | Clears the screen (only asynchronous emulation) |
| **ClosePanel** | Erases and unloads the current key panel from memory |
| **Connect** | Connects to a server (only 3270 and 5250 emulations) |
| **Continue** | Unconditional return to a loop |
| **Disconnect** | Disconnects from a server previously connected (only 3270 and 5250 emulations) |
| **Display** | Displays characters received by the emulator (only asynchronous emulation) |
| **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 (only asynchronous emulation) |
| **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 |
| **PrintScreen** | Prints the screen (only 3270 and 5250 emulations) |
| **PrintTemplate** | Prints using a template (only 3270 and 5250 emulations) |
| **RcopyGet** | File transfer from UNIX to MS-DOS (only asynchronous emulation) |
| **RcopyPut** | File transfer from MS-DOS to UNIX (only asynchronous emulation) |
| **Rcoupe** | Disconnects modem communication (only asynchronous emulation) |
| **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 |
| **Receive** | Waits to receive one or more strings (only asynchronous emulation) |
| **ReceiveFile** | Receives a file from the remote server (only 3270 emulation) |
| **Repeat** | Starts a block of reiterative instructions |
| **Return** | Unconditional exit from macro and return to the emulator |
| **Rdial** | Dialing using a Hayes-compatible modem (only asynchronous emulation) |

| | |
|---|---|
| **Rhangup** | Ends modem communication (only asynchronous emulation) |
| **Rnumero** | Dialing using a Hayes-compatible modem (only asynchronous emulation) |
| **SearchString** | Searches a character string on the screen (only 3270 and 5250 emulations) |
| **SearchStringIn Rect** | Searches for a string of characters in a rectangular area of the screen (only 3270 and 5250 emulations) |
| **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 (only asynchronous emulation) |
| **SendFile** | Sends a local file to the remote server (only 3270 emulation) |
| **SendFunKey** | Sends a function key code to the remote server (only 3270 and 5250 emulations) |
| **SessionTitle** | Assigns a name to the emulation session |
| **Set** | Defines and assigns a variable |
| **SetCursor** | Moves the cursor (only 3270 and 5250 emulations) |
| **SetHelpFile** | Defines a help file (only asynchronous emulation) |
| **SetToCursor** | Moves the cursor to the last position found (only 3270 and 5250 emulations) |
| **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 |
| **Wait** | Waits until a timeout is finished (only 3270 and 5250 emulations) |

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

# CHAPTER 7 - 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.

As 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 actions no. **271** and **272** initiate file transfers between the PC and the server without user intervention. The parameters associated with these actions 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 server in emulation session number one (X:).

In the default ANSI.SEQ file, the following escape sequences have been added in order to perform these types of file transfer:

```
\033\033put%p0%s p 271
\033\033get%p0%s p 272
```

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 **%p0%s p 271 and 272** to the relevant .SEQ file. To trigger these actions from the UNIX host, you could use a shell script such as:

```
echo "\033\033put'C:\\\\AUTOEXEC.BAT/tmp/autoexec.bat'"
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 "\033\033get' /tmp/termcap C:\\\\\TUN\\\\\EMUL\\\\\termcap'"
read A
$A
```

A shell script file **transfile.sh** illustrating the possibilities described in this section is supplied with the product.

## PC PROGRAMS STARTED BY SERVER

Parametered actions nos. **295** and **296** may be used to start MS-DOS or Windows 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 **write.exe** under Windows you could give **C:\WINDOWS\WRITE** 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:\\\\\\WINDOWS\\\\\\WRITE\""
```

> **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 to do this 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

### Using Windows Print Manager

Verify that the file **c:\tun\emul\xxxx.seq** contains the following ESC sequences:

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

The first sequence activates transparent printing mode and the second inhibits it. If these sequences are not to be found in the file, add them to the end.

### Unix shell

The following is an example of using the UNIX shell to transparently print the file **/etc/passwd** on the PC using the default printer:

```
echo "Start transparent printing test(xxxx.seq)"
echo -n "\033[5i"
echo "File PASSWORD"
echo "-----------------------------"
cat /etc/passwd
echo "-----------------------------"
echo "End of file"
echo "\033[4i"
echo "End of test"
```

### Bypassing the Windows Print Manager

The following example can be used to print transparently on LPTn or COMm without using the Windows Print Manager.

Check that the following ESC sequences appear in the**.seq** file corresponding to your type of emulation in the directory **c:\tun\emul**:

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

The first line switches to transparent copy mode and the second inhibits this mode (transparent printing in latter case is effected with the Windows Print Manager.

If the above sequences are in the file, replace them with the following sequences, otherwise simply add the following sequences to the end of the file:

```
[5i s 260 262("LPT1")
[4i s 259 263
```

The first switches to direct transparent copy mode on the parallel port LPT1 and the second inhibits this mode.

### Example

In the following example, the UNIX shell transparently prints a copy of the file /etc/passwd on the LPT1 port of the PC:

```
echo "Start transparent printing test"
echo -n "\033[5i"
echo "Transparent copy test"
echo "-----------------------------"
cat /etc/passwd
echo "End of test"
echo "\033[4i"
```

The objective of this script is to print in transparent mode without going through the Windows Print Manager.

N.B.: If a Windows application is printing at the same time as **Tun EMUL** is printing transparently, the print jobs will be mixed up because of the direct printing on the port.

## 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 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\""
```

## CHANGING SESSIONS AUTOMATICALLY

As **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 31).

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

```
\033S%p0%1d 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:

**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_ALL** | Activates all the events defined by initialization |
| **EVENT_MOVE** | 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 8 - 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**: A more simple solution consists in defining the display settings (see "Customizing the Terminal Display").

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 (15,1).

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).

## 132-COLUMN EMULATION

### 132 Columns under Windows

It is very easy to configure emulation with 132 columns under windows. However, in order to be able to see the entire screen, you should assign the font Sys132PC to the display settings file (.CTX).

### 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 option **Parameters➔Display Settings➔Size**

#### 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 both MS-DOS and Windows.

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).

Under Windows, you may create an alternate font using SDK. 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:

- Switch the emulator to scancode mode by sending the sequence **\033~5**
- Change the tty on the UNIX host using this command:

```
stty isscancode xscancode
```

To switch back to **Tun EMUL**'s native mode:

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

```
stty -isscancode -xscancode
```

## USING COM3 AND COM4 UNDER MS-DOS

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.

Under Windows, COM3 and COM4 are defined using the Control panel. 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 the **Settings➔Modem Commands...** parameters to define the codes that are required.

# PART III
# REFERENCE GUIDE

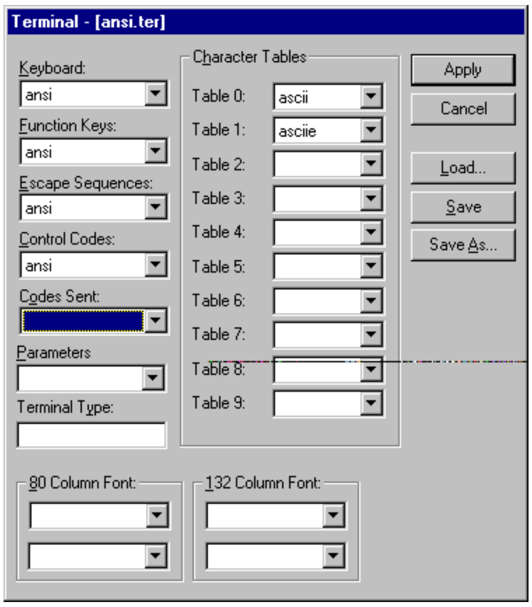
# CHAPTER 9 - TERMINAL CUSTOMIZATION

## 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.

## COMPOSITION OF A TERMINAL

To load the configuration of a terminal, select the option **Settings➔Terminal...** in the principal menu and then select one of the terminals proposed (e.g. ansi.ter):



Terminal - [ansi.ter]

Keyboard: ansi
Function Keys: ansi
Escape Sequences: ansi
Control Codes: ansi
Codes Sent:
Parameters
Terminal Type:

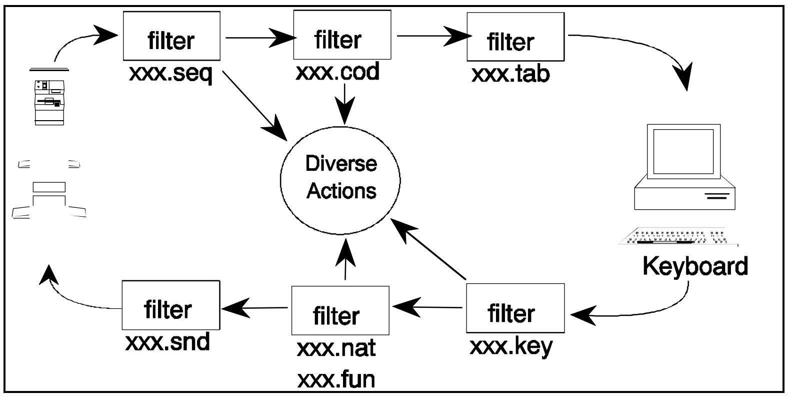
Character Tables
Table 0: ascii
Table 1: asciie
Table 2:
Table 3:
Table 4:
Table 5:
Table 6:
Table 7:
Table 8:
Table 9:

80 Column Font:
132 Column Font:

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 pre-defined suffixes (.key, .nat, .snd, .fun, .seq, .cod, .tab, .ses) 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).

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, .snd, .ses usually do not need to be changed.

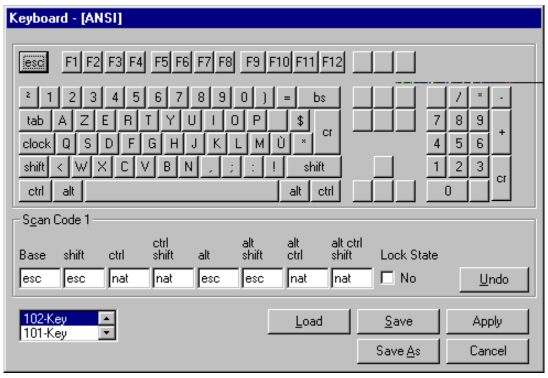
### Terminal configuration

A terminal configuration file with the extension **.ses**, whose contents interact with the **.seq** and **.cod** files, can be associated with each type of terminal. A number of configuration files are supplied with **Tun EMUL** so that standard configuration parameters for the chosen terminal can be associated with each session.

These parameters can be changed on the **Session** tab in the dialog box obtained by clicking **More >>** when a new session is opened.

## DEFINING KEYBOARD FILES (.KEY)

Keyboard definition is managed through **Parameters➔Keyboard** in the principal menu and selecting the keyboard to be modified.


Keyboard - [ANSI]
esc F1 F2 F3 F4 F5 F6 F7 F8 F9 F10 F11 F12
² 1 2 3 4 5 6 7 8 9 0 ) = bs
tab A Z E R T Y U I O P $ cr
clock Q S D F G H J K L M Ù *
shift < W X C V B N , ; : ! shift
ctrl alt alt ctrl
/ * - 7 8 9 + 4 5 6 1 2 3 cr 0
Undo
102-Key
101-Key
Load Save Apply
Save As Cancel

Scan Code 1

| Base | shift | ctrl | ctrl shift | alt | alt shift | alt ctrl | alt ctrl shift | Lock State |
|---|---|---|---|---|---|---|---|---|
| esc | esc | nat | nat | esc | esc | nat | nat | ☐ No |

All the physical keys on the keyboard are identified by a unique value: the scan code. Clicking on one of the keys displays the associated values.

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 check box **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)

With this dialog box, 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:

- enter the character directly (if it is printable, as long as the character is not a number or a space):
  *example:*
  **`a      B       &`**

- place it between quotes:
  *example:*
  **`'a'    '1'     '?'`**

  *exceptions:*
  ' is coded '\''
  \ is coded '\\' or \\

- give the character's decimal value between 0 and 255
  *example:*
  **`203   48      53`**

- give the hexadecimal value (0 to FF, preceded by 0x:
  *example:*
  **`0x0a  0x30       0x4f`**

- 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. 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 check box **Lock State** indicates 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 keyboardfor France (FR.NAT):

```
 2       &    '1'   nop    nop    nop    nop    nop    nop
 3       Θ    '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       Φ    '7'   nop    nop      `      `      `   nop
 9       _    '8'   nop    nop   '\\'   '\\'   '\\'   nop
10       τ̄    '9'   nop    nop      ^      ^      ^   nop
11       α    '0'   nop    nop      @      @      @   nop
26  cflexe  trema   esc    esc    nop    nop    nop    esc
27       $      ú    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 .fun files contain the definition of each of the function keys used during an emulation session. For example, here is a part of the file ANSI.FUN:

```
\033[M       F1
\033[N       F2
\033[O       F3
.
.
\033['       Ctrl/Shft F11
\033[{       Ctrl/Shft F12
\033[H       Home
\033[A       Up arrow
\033[I       Page up
-            -
\033[D       Left arrow
del          Suppr
\033[C       Right arrow
+            +
\033[F       End
\033[B       Down arrow
\033[G       Page down
\033[L       Insert
nop          \1`\0xd
nop          \1a\0xd
nop
nop
nop
nop
nop
nop
```

There are two columns in the file:

- The character chain produced by the function key
- The name of the function key

To change the value sent by a function key, change the corresponding value in the first column. The encoding of the sequences follows the same rules as those defined for the character chains in**key** files.

It is possible, using function-key encoding, to perform all the actions available with **Tun EMUL**. For example:

| Encoding | F-Key | Action |
|---|---|---|
| a.51.250 | **F2** | executes actions 51 (clear screen) and 250 (beep) |
| a.264("c:\\tun\\emul\\macessai") | **F3** | runs the macro "macessai" |
| root\n | **F5** | sends the username for the connection |
| password\n | **F6** | sends the password |
| a.296("c:\\windows\\write") | **F7** | runs the program "write" |
| a.141(20,72,2) | **F8** | draws a rectangle |

## 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 in the following cases:

- if the current initialization strings are not appropriate,
- 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).

### Contents of a .SEQ file

The .SEQ files contain the definition of the function keys used during an emulation session. For example, here is part of the file ANSI.SEQ:

```
62 74
\033
[6n s 227(%"\0233"%V%{1}%+%s%$%";"%$%H%{1}%+%s%$%"R"%$)
[R%p0%c%p1%dm p 290
[%p0%{1}%-%3d;%p1%{1}%-%3dH p 91
[%p0%{1}%-%3d;%p1%{1}%-%3df p 91.
.
.
.

\033Me%p0%dX p 283
\033Mf%p0%s p 284
\033put%p0%s p 271
\033get%p0%s p 272
```

Escape sequence files are composed of three parts:

- An initialization sequence (the first line)
- A sequence header (the second line)
- Defined escape sequences (all remaining lines)

## 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 divide the initialization sequence into several lines terminating each line, except the last one, with the backslash character\ (e.g. the 2nd line in the file WYSE60.SEQ).

Here, for example, is an initialization line:

```
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 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.

**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:

- simple sequences that do not change
- 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:

- definition of the parameters themselves
- calculations and controls to be carried out on the parameter
- 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 pop a variable off the stack

*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. Ifthis 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]** :

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

- 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 EMUL**, control codes are configured in files with a **.COD** extension. Here, for example, are the contents of the file ANSI.COD:

```
nul       0
soh       0
stx       0
etx       0
eot       0
enq       0
ack       261
bel       250
bs        96
ht        99
lf        113
vt        0
ff        51
cr        97
so        0
si        0
dle       0
dc1       0
dc2       0
dc3       0
dc4       0
nak       0
syn       0
etb       0
can       0
em        0
sub       0
esc       0
fs        0
gs        0
rs        0
us        0
```

The first column contains the control code mnemonic and the second column contains the number of the action to be carried out on reception of the corresponding code. Three options may be used in the second column:

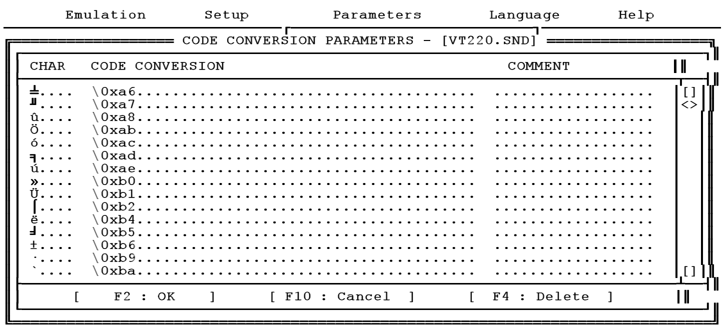
- Leave the value blank (In this case, the character is displayed on the terminal).
- Respond with 0 (In this case, no action is carried out, and the character is not listed).
- Select an action from the list in the **.eng** action file or the **.doc** action file.

Only simple actions can be associated with the control codes.

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:



```
     Emulation      Setup          Parameters       Language      Help

            CODE CONVERSION PARAMETERS - [VT220.SND]
 CHAR   CODE CONVERSION                              COMMENT
 ╧....  \0xa6..................................  ..................  []
 ╜....  \0xa7..................................  ..................  <>
 û....  \0xa8..................................  ..................
 Ö....  \0xab..................................  ..................
 ó....  \0xac..................................  ..................
 ╕....  \0xad..................................  ..................
 ú....  \0xae..................................  ..................
 »....  \0xb0..................................  ..................
 Ü....  \0xb1..................................  ..................
 ⌠....  \0xb2..................................  ..................
 ë....  \0xb4..................................  ..................
 ╛....  \0xb5..................................  ..................
 ±....  \0xb6..................................  ..................
 ·....  \0xb9..................................  ..................
 `....  \0xba..................................  ..................  []
    [    F2 : OK    ]     [ F10 : Cancel  ]     [  F4 : Delete  ]
```

## 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 |

A .TAB file looks like this:

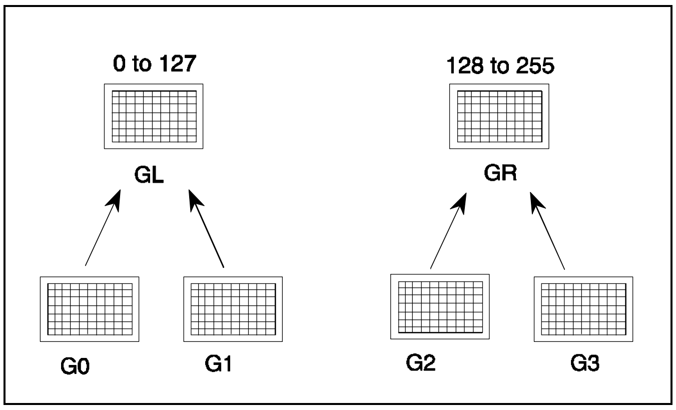
```
80 81 82 83 84 85 86 87 88 89 8a 8b 8c 8d 8e 8f
90 91 92 93 94 95 96 97 98 99 9a 9b 9c 9d 9e 9f
a0 a1 a2 a3 a4 a5 a6 a7 a8 a9 aa ab ac ad ae af
b0 b1 b2 b3 b4 b5 b6 b7 b8 b9 ba bb bc bd be bf
c0 c1 c2 c3 c4 c5 c6 c7 c8 c9 ca cb cc cd ce cf
d0 d1 d2 d3 d4 d5 d6 d7 d8 d9 da db dc dd de df
e0 e1 e2 e3 e4 e5 e6 e7 e8 e9 ea eb ec ed ee ef
f0 f1 f2 f3 f4 f5 f6 f7 f8 f9 fa fb fc fd fe ff
```

There are 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 have to be entered in this table.

To change a particular character, its hexadecimal code must first be looked up in the ASCII table and then entered in 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 the tables 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.

## CONFIGURATION FILES (.SES)

A terminal configuration file with the extension **.ses**, whose contents interact with the **.seq** and **.cod** files, can be associated with each type of terminal. A number of configuration files are supplied with **Tun EMUL** so that standard configuration parameters for the chosen terminal can be associated with each session.

These parameters can be changed on the **Session** tab in the dialog box obtained by clicking **More >>** when a new session is opened.

It is also possible to create **.ses** configuration files directly. Here is an example of this type of file:

```
[Intro]
ParamNb=4......................................  Total number of parameters
Param1=id_MonitorMode..........................  Each parameter has an identifier.
Param2=id_GraphMode............................  ...
Param3=id_NbLine...............................
Param4=id_NbCol................................

[id_MonitorMode]                                 Title of the parameter section.
Label=SetupMonitorMode.........................
ItemNb=2.......................................  Number of choices in the combo box
Item1=SetupON..................................  Setup choices
Item2=SetupOFF.................................  ...
Action1=155("MonitorMode","1")                   List of actions to be executed according to
        160("MonitorMode").....................  the choice
Action2=155("MonitorMode","0")                   ...
        160("MonitorMode").....................
InitDefault=2..................................  Designates the default choice
InitAction=%g"MonitorMode"%{1}%+...............  Action which selects the correct option when
                                                 the dialog box is opened (when modifying
[id_GraphMode]                                   an active session).
Label=SetupGraphMode
ItemNb=2
Item1=SetupON
Item2=SetupOFF
Action1=155("GraphMode","1")
160("GraphMode")
Action2=155("GraphMode","0")
160("GraphMode")
InitDefault=2
InitAction=%g"GraphMode"%{1}%+

[id_NbLine]
Label=SetupNbLine
ItemNb=2
Item1=Setup24Li
Item2=Setup25Li
Action1=155("NbLigne","24")
160("NbLigne")
```

```
Action2=155("NbLigne","25")
160("NbLigne")
InitDefault=1
InitAction=%g"NbLigne"%{23}%-

[id_NbCol]
Label=SetupNbCol
ItemNb=2
Item1=Setup80Col
Item2=Setup132Col
Action1=155("NbCol","80") 160("NbCol")
Action2=155("NbCol","132") 160("NbCol")
InitDefault=1
InitAction=If(%g"NbCol"%{80}%=)    %{1}
Else %{2} EndIf
```

## Description

### Identifiers

The parameter identifiers (e.g. in the above example *id_MonitorMode*, etc) are also used to set the configuration file for a particular session. The names of combo-boxes and the listed items are stored in the **.lg** files. Therefore the file **.ses** contains the related message file identifiers (e.g. *SetupMonitorMode*, *SetupOn*, etc. in the example) as parameters.

### Order of the combo-boxes

The combo-boxes appear in the order of their definition in the section *[Intro]*. The items listed in a combo-box appear in the order of their definition.

### Actions

The field "InitAction" must contain a number from 1 through N corresponding to the initial choice in the combo-box when it is first opened and when it relates to an active session. In other cases, the field "InitDefault" is used. "InitDefault" is set in accordance with the initialization parameters of the **.seq** file, so that the **.ses** and **.seq** files correspond.

If the dialog box relates to the active session, the actions relating to the choices of each combo-box (*Action1...ActionN*) will be executed when the dialog box is validated (**OK** is pressed).

### Loading order of the *ses* file

The **.ses** file is loaded after the **.seq** file but before the user's choice of **.cfg** file. The choice of **.cfg** file (session configuration) therefore receives priority.

# CHAPTER 10 - PROGRAM SYNTAX

## INDEX

## EMULWIN EMULWIN

EMULWIN (EMUL32 under Windows 95) is the command used to start the emulator.

*Syntax*

```
emulwin config.cfg      [-u [h][c][m][d][t][r][s][w]]
        [-m in_macro]   [-q out_macro]
        [-h hostname]   [-t term.ter]
        [arg]
```
under Windows 3.x

```
emul32  config.cfg      [-u [h][c][m][d][t][r][s][w]]
        [-m in_macro]   [-q out_macro]
        [-h hostname]   [-t term.ter]
        [arg]
```
under Windows 95

or

```
emulwin [-k Niskey [-y Nistable]]
        [-u [h][c][m][d][t][r][s][w]]
        [-m in_macro]   [-q out_macro]
        [-h hostname]   [-t term.ter]
        [arg]
```
under Windows 3.x

```
emul32 [-k Niskey [-y Nistable]]
        [-u [h][c][m][d][t][r][s][w]]
        [-m in_macro]   [-q out_macro]
        [-h hostname]   [-t term.ter]
        [arg]
```
under Windows 95

| | |
|---|---|
| **config.cfg** | configuration file (".cfg" file). The suffix is optional. If used, this argument starts **Tun EMUL** using the supplied terminal emulation configuration. |
| **-k** | NIS resource (must be associated with-**y**) |
| **-y** | NIS table (host or emulation configuration table, must be associated with **-k**). |
| **-m** | followed by the name of a macro to be executed when emulation is started |
| **-q** | followed by the name of a macro to be executed when quitting emulation |
| **-h** | followed by the name of the host. Used with a config.cfg if config.cfg has @ as host name. |
| **-t** | terminal file (".ter" file). Used with a config.cfg if config.cfg has @ as terminal file. |
| **-u** | used to restrict users' rights to emulation under MS-Windows. It serves to disable the following options to ensure user security:<br><br>**File➔New Session**<br>**File➔Open Configuration...**<br>**File➔Save Configuration**<br>**File➔Save Configuration As...**<br>**File➔Close Session**<br>**File➔Close All Sessions** |

**File➔Print Screen**
**File➔Print Setup...**
**Settings, Connection, Transfers, Macro, Windows**

The characters h, c, m, d, t, r, s, and w are used in conjunction with **-u** to selectively reactivate features:

| | |
|---|---|
| **h** | reactivates **File➔Print Screen** |
| **c** | reactivates **File➔Print Setup...** (with H). |
| m | reactivates **Macro** |
| **d** | reactivates the **Connection** option |
| **t** | reactivates **Transfers➔Protocol Setup...** |
| **r** | reactivates **Transfers➔Get File...** |
| **s** | reactivates **Transfers➔Send File...** |
| **w** | reactivates **Windows** |

**arg** represents the different parameters that are sent to the host machine when the emulation is loaded.

*Example*

```
EMULWIN conf1 -u
```

## 3270 3270

3270 emulation under Windows

*Syntax*

**`3270 [-c config_file] [-h hostname] [-t term]`**
**`[-k Niskey [-y Nistable]]`**
under Windows 3.x

**`3270_32 [-c config_file] [-h hostname] [-t term]`**
**`[-k Niskey [-y Nistable]]`**
under Windows 95

*Description*

3270.EXE (3270_32.EXE) is a Windows terminal emulator that provides 3270 emulation for connecting to IBM mainframes.

| | |
|---|---|
| **`-c`** | starts emulation with an initialization file that defines:<br>• host name<br>• 3270 terminal type<br>• color assignment<br>• font size and name<br>• ASCII / EBCDIC correspondence<br>• keyboard definition |
| **`-h`** | followed by the name of the host |
| **`-t`** | followed by the name of a terminal file |
| **`-k`** | NIS resource (must be associated with-**y**) |
| **`-y`** | NIS table (host or emulation configuration table, must be associated with **-k**). |

Initialization files are created by entering emulation, and then saving the current configuration using **File➔Save**.

## 5250 5250

5250 emulation under Windows

*Syntax*

**`5250 [-c config_file] [-h hostname] [-t term]`**
**`[-k Niskey [-y Nistable]]`**
under Windows 3.x

**`5250_32 [-c config_file] [-h hostname] [-t term]`**
**`[-k Niskey [-y Nistable]]`**
under Windows 95

*Description*

5250.EXE (5250_32.EXE) is a Windows terminal emulator that provides 5250 emulation for connecting to IBM AS/400 servers.

| | |
|---|---|
| **-c** | starts emulation with an initialization file that defines:<br>• host name<br>• 3270 terminal type<br>• color assignment<br>• font size and name<br>• ASCII / EBCDIC correspondence<br>• keyboard definition |
| **-h** | followed by the name of the host |
| **-t** | followed by the name of a terminal file |
| **-k** | NIS resource (must be associated with-**y**) |
| **-y** | NIS table (host or emulation configuration table, must be associated with **-k**). |

Initialization files are created by entering emulation, and then saving the current configuration using **File➔Save**.

## PANEDIT PANEDIT

Function-key panel editor.

*Syntax*

**`PANEDIT [file]`**
under Windows 3.x

**`PANED32 [file]`**
under Windows 95

*Description*

PANEDIT is a Windows program which creates and customizes key panels for use with **Tun EMUL** emulators. The only argument which can be passed to the program is the name of a key panel file, extension .PAN, which contains the definition of the key panel.

# CHAPTER 11 - MACRO LANGUAGE SYNTAX

## INDEX

| | |
|---|---|
| **Break** | Unconditional exit from a loop |
| **ChangeTerminal** | Instant change of terminal type (only asynchronous emulation) |
| **ClearMessage** | Erase messages in a dialog box |
| **ClearScreen** | Clears the screen (only asynchronous emulation) |
| **ClosePanel** | Erases and unloads the current key panel from memory |
| **Connect** | Connects to a server (only 3270 and 5250 emulations) |
| **Continue** | Unconditional return to a loop |
| **Disconnect** | Disconnects from a server previously connected (only 3270 and 5250 emulations) |
| **Display** | Displays characters received by the emulator (only asynchronous emulation) |
| **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 (only asynchronous emulation) |
| **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 |
| **PrintScreen** | Prints the screen (only 3270 and 5250 emulations) |
| **PrintTemplate** | Prints using a template (only 3270 and 5250 emulations) |
| **RcopyGet** | File transfer from UNIX to MS-DOS (only asynchronous emulation) |
| **RcopyPut** | File transfer from MS-DOS to UNIX (only asynchronous emulation) |
| **Rcoupe** | Disconnects modem communication (only asynchronous emulation) |
| **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 |
| **Receive** | Waits to receive one or more strings (only asynchronous emulation) |
| **ReceiveFile** | Receives a file from the remote server (only 3270 emulation) |
| **Repeat** | Starts a block of reiterative instructions |
| **Return** | Unconditional exit from macro and return to the emulator |
| **Rdial** | Dialing using a Hayes-compatible modem (only asynchronous emulation) |
| **Rhangup** | Ends modem communication (only asynchronous emulation) |
| **Rnumero** | Dialing using a Hayes-compatible modem (only asynchronous emulation) |

| | |
|---|---|
| **SearchString** | Searches a character string on the screen (only 3270 and 5250 emulations) |
| **SearchStringIn Rect** | Searches for a string of characters in a rectangular area of the screen (only 3270 and 5250 emulations) |
| **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 (only asynchronous emulation) |
| **SendFile** | Sends a local file to the remote server (only 3270 emulation) |
| **SendFunKey** | Sends a function key code to the remote server (only 3270 and 5250 emulations) |
| **SessionTitle** | Assigns a name to the emulation session |
| **Set** | Defines and assigns a variable |
| **SetCursor** | Moves the cursor (only 3270 and 5250 emulations) |
| **SetHelpFile** | Defines a help file (only asynchronous emulation) |
| **SetToCursor** | Moves the cursor to the last position found (only 3270 and 5250 emulations) |
| **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 |
| **Wait** | Waits until a timeout is finished (only 3270 and 5250 emulations) |

## 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

**Only available for asynchronous emulation.**

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.

*Syntax*

```
ClearMessage
```

*Description*

This instruction erases the contents of a dialog box.

*See also*

HideMessage, ShowMessage

## CLEARSCREEN CLEARSCREEN

**Only available for asynchronous emulation.**

Clears the screen.

*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

## CONNECT CONNECT

**Only available for 3270 and 5250 emulation.**

Connects to a server.

*Syntax*

```
Connect server [terminal] [-port]
```

*Description*

This instruction lets you establish a connection with the remote server **server**. The parameter **terminal** refers to the name of the terminal for 5250 emulation and the terminal model for 3270 emulation. The option **port** specifies the IP port that is to be used.

*Example*

```
IfConnected Return
Connect AS400 "IBM-5251-11" -1023
IfError Error
Return
Label Error
Echo "Connection error!"
```

*See also*

IfConnected, Disconnect

## DISCONNECT DISCONNECT

**Only available for 3270 and 5250 emulations.**

Disconnects from a server previously connected.

*Syntax*

```
Disconnect
```

*Description*

This instruction closes the current connection.

*Example*

```
IfConnected Disconn
Label Conn
Connect MAINFRAME "5"
IfError Error
Return
Label Disconn
Disconnect
Goto Conn
Label Error
Echo "Connection error!"
```

*See also*

IfConnected, Connect

## DISPLAY DISPLAY

**Only available for asynchronous emulation.**

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*. For internal commands, the following string must precede the command: "**C:\\COMMAND.COM".**

*Examples*

Internal command:

```
dos "c:\\command.com /c dir"
```

External command:

```
dos "xcopy c:\\tun\\emul\\*.seq 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. The 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

**Only available for asynchronous emulation.**

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.

*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 '*hbsec*" seconds.

## PRINTSCREEN PRINTSCREEN

**Only available for 3270 and 5250 emulations.**

Prints the screen.

*Syntax*

```
PrintScreen [-t|-g] [-s]
```

*Description*

This instruction prints the screen. The option **-t**, which is the default value, prints the screen in text mode and the option **-g** prints the screen in graphic mode. The option **-s** displays the dialog box for selecting the printer.

*See also*

PrintTemplate

## PRINTTEMPLATE PRINTTEMPLATE

**Only available for 3270 and 5250 emulations.**

Prints using a template.

*Syntax*

```
PrintTemplate [template] [-c|-e|-a]
```

*Description*

This instruction prints from a template. The parameter template, of type "string" is the name of the print template file you want to use. The option **-c** prints the current screen, **-e** prints from the current page to the end of the file, and the option **-a** prints all the pages, otherwise if there are no options specified the dialog box **Print with Template** is displayed.

*See also*

PrintScreen

## RCOPYGET RCOPYGET

**Only available for asynchronous emulation.**

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

**Only available for asynchronous emulation.**

Transfers files from MS-DOS to UNIX.

*Syntax*

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

*Description*

This command executes file transfer from Ms-DOS to UNIX. 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** 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 currentlyhas a UNIX prompt on the screen. Wildcard characters ? and * may be used to transfer several files at once.

*See also*

**RcopyGet**

## RCOUPE RCOUPE

**Only available for asynchronous emulation.**

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

## RDIAL RDIAL

**Only available for asynchronous emulation.**

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 phone book 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

## RECEIVE RECEIVE

**Only available for asynchronous emulation.**

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

## RECEIVEFILE RECEIVEFILE

**Only available for 3270 emulation.**

Receives a file from the remote server.

*Syntax*

```
ReceiveFile local_file recv_cmd
```

*Description*

This instruction starts a file transfer from the remote server to the local machine using the protocol IND$FILE. The parameter **local_file**, of type "string" is the name the file will be given on the local machine. The parameter **recv_cmd**, of type "string" contains the standard reception parameters of the command IND$FILE.

*Example*

The instruction:

```
ReceiveFile "C:\\PUBLIC\\SALES.RPT" "SLS REPRT A (ASCII CRLF"
```

transfers the file "SLS REPRT A" from a CMS system to the local file "C:\PUBLIC\SALES.RPT" with ASCII/EBCDIC and CRLF conversion.

*See also*

SendFile

## REPEAT REPEAT

Starts a 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

## 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

**Only available for asynchronous emulation.**

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

## RNUMERO RNUMERO

**Only available for asynchronous emulation.**

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 phone book 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

## SEARCHSTRING SEARCHSTRING

**Only available for 3270 and 5250 emulations.**

Searches a character string on the screen.

*Syntax*

```
SearchString string [startx starty stopx stopy]
```

*Description*

This instruction is used for searching for a string indicated by the parameter **string**. The search can be carried out over the whole screen or in a part of the screen, in which case the parameters **startx**, **starty**, **stopx**, **stopy** are used to specify the start and end positions of the search area. When the instruction has finished, and if the string has been found, the screen coordinates of the start of the string found will be stored in memory and can be used by the instruction **SetToCursor**.

*Example*

```
SearchString "Program"
IfError Error
SetToCursor
Return
Label Error
Echo "The screen does not contain the string 'Program'"
```

*See also*

SearchStringInRect, SetToCursor

## SEARCHSTRINGINRECT SEARCHSTRINGINRECT

**Only available for 3270 and 5250 emulations.**

Searches for a string of characters in a rectangular area of the screen.

*Syntax*

```
SearchStringInRect string [left top right bottom]
```

*Description*

This instruction can be used for searching for a string, designated by **string**, on the screen. The search can be carried out over the whole screen or in a rectangular area of the screen, in which case the parameters **left**, **top**, **right**, and **bottom** are used to specify the coordinates top-left and bottom-right of the search area. When the instruction has finished, and if the string has been found, the screen coordinates of the start of the string found will be stored in memory and can be used by the instruction **SetToCursor**.

*Example*:

```
SearchStringInRect "Program" 3 4 50 16
IfError Error
SetToCursor
Return
Label Error
Echo "The string 'Program' cannot be found in the search area"
```

*See also*

SearchString, SetToCursor

## 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

**Only available for asynchronous emulation.**

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

## SENDFILE SENDFILE

**Only available for 3270 emulation.**

Sends a local file to the remote server.

*Syntax*

```
SendFile local_file send_cmd
```

*Description*

This instruction starts a file transfer to the remote server from the local machine using the protocol IND$FILE. The parameter **local_file**, of type "string", is the name of the local file that is to be sent. The parameter **send_cmd**, of type "string", contains the standard send parameters of the command IND$FILE.

*Example*

The instruction:

```
SendFile "C:\\PUBLIC\\SALES.RPT" "SLS REPRT A (ASCII CRLF"
```

transfers the local file "C:\PUBLIC\SALES.RPT" to the file "SLS REPRT A" on a CMS system with ASCII/EBCDIC and CRLF conversion.

*See also*

ReceiveFile

## SENDFUNKEY SENDFUNKEY

**Only available for 3270 and 5250 emulations.**

Sends a function key code to the remote server.

*Syntax*

```
SendFunKey key
```

*Description*

This instruction simulates the pressing of a function key just as if it had been typed on the keyboard. The parameter **key**, of type string, must be one of the function keys listed in the **Keyboard** dialog box.

*Example*

```
ReadVar "Enter the name of the user:" user
Send %user
SendFunKey Enter
Wait
Return
```

*See also*

Send

## SESSIONTITLE SESSIONTITLE

Assigns a name to the emulation session.

*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

## SETCURSOR SETCURSOR

**Only available for 3270 and 5250 emulations.**

Moves the cursor.

*Syntax*

```
SetCursor x y
```

*Description*

This instruction moves a cursor to the position defined by the **x** and **y** parameters.

*See also*

SetToCursor

## SETHELPFILE SETHELPFILE

**Only available for asynchronous emulation.**

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"
```

## SETTOCURSOR SETTOCURSOR

**Only available for 3270 and 5250 emulations.**

Moves the cursor to the last position found.

*Syntax*

```
SetToCursor
```

*Description*

The search commands such as SearchString store the coordinates of the last position found if the search was successful. The command **SetToCursor** moves the cursor to this last position.

*Example*

```
SearchString "Program"
IfError Error
SetToCursor
Return
Label Error
Echo "The string 'Program' is not on the screen"
```

*See also*

SearchString, SearchStringInRect, SetCursor

## 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.

*Syntax*

**`ShowMessage [-h]`**

*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.
The option [-h] can be used to display the dialog box without the **Cancel** button.

*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.

## WAIT WAIT

**Only available for 3270 and 5250 emulations.**

Waits until a timeout is finished.

*Syntax*

```
Wait [timeout]
```

*Description*

This command puts the system on hold until the end of a timeout or until the time defined by the parameter **timeout** is finished. The parameter **timeout** is in milliseconds.

*Example*

```
SendFunKey Enter
Wait 1000
IfError Timeout
Return
Label Timeout
Echo "The remote system is not responding after 1 second."
```

*See also*

Sleep, SendFunKey

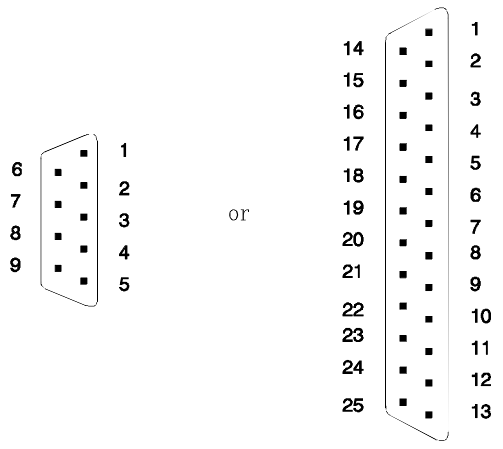
# APPENDICES

# 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 following figure:



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:

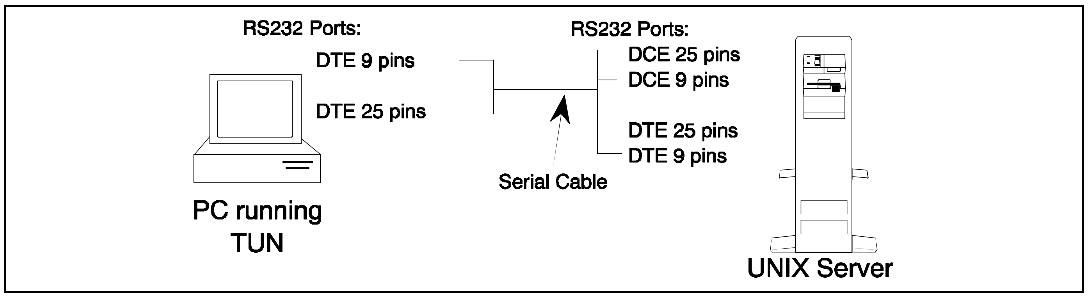
| **PC 25-pin male DTE** | | **Host Machine 25-pin female** |
|---|---|---|
| TD 2 | ⟷ | 2 RD |
| RD 3 | ⟷ | 3 TD |
| GRD 7 | ⟷ | 7 GRD |

## 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 figure below:



**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 |
|---|---|---|
| **25-pin DTE** |  | **25-pin DCE** |
| 2 | ↔ | 2 |
| 3 | ↔ | 3 |
| 7 | ↔ | 7 |
| **25-pin DTE** |  | **9-pin DCE** |
| 2 | ↔ | 3 |
| 3 | ↔ | 2 |
| 7 | ↔ | 5 |
| **25-pin DTE** |  | **25-pin DTE** |
| 2 | ↔ | 3 |
| 3 | ↔ | 2 |
| 7 | ↔ | 7 |
| **25-pin DTE** |  | **9-pin DTE** |
| 2 | ↔ | 2 |
| 3 | ↔ | 3 |
| 7 | ↔ | 5 |

| **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 a 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 for "sysadmsh |
|---|---|

The following macro is an example of remote communication via modem. It connects the user to Esker's BBS. It assumes a modem is connected to the serial port of a host running SCO UNIX and that the following lines are included in the file **/usr/lib/uucp/Devices** on the UNIX machine.

```
ACU tty1a - 1200 hayes1200 \D
ACU tty1a - 9600 hayes9600 \D
```

The user may elaborate the basic macro by adding the loop required to test the presence of several modems or ports and choose the next available one).

```
# Start
Label BEGIN

# Send login
Hide
        SendAndReceive 20 "root\r" "assword"
                IfError NOCONNECTION
        SendAndReceive 20 "support\r" "TERM"
                IfError NOCONNECTION
        SendAndReceive 20 "\r" "#"
                IfError NOCONNECTION
        send "clear\r"
        pause 2


# Check if the port is busy

ReadVar "Enter the port you want to use: " PORT

ClearMessage
Echo "Please wait, ... connecting ... , this will take a few seconds"
             SendAndReceive  3  "cu  -s9600  -l  %PORT  dir\r"  "connected"
   "LOCKED"

#   You can insert the name of the device directly, if known
#              SendAndReceive  20  "cu  -l  /dev/tty1a  dir\r"  "connected"
   "LOCKED"
#   Then you do not need the variable PORT

        IfEqual "LOCKED" BUSY

# Connection
        Send "atz\r"
        pause 4
        Send "atdt 0,72449472\r"
sleep 20
HideMessage

Display

Receive 0 "Thank you" "NO CARRIER"
pause 1
exit
```

```
# No connection
Label NOCONNECTION
Echo "Communication failed"
ReadVar "Press Return to quit: " ANSWER
# Exit the emulator
exit

Label Busy
ClearMessage
ShowMessage
echo "MINITEL IS ALREADY IN USE, PLEASE TRY AGAIN LATER"
sleep 5
ReadVar "Press return to quit: " ANSWER
#exit the emulator
exit
```

# INDEX