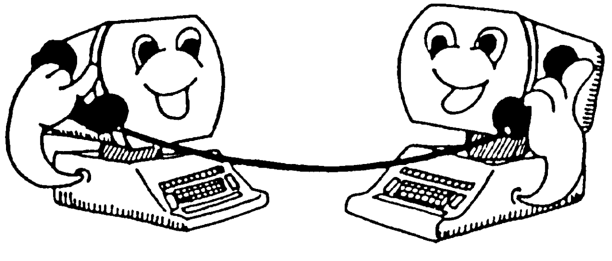
OS9 LEVEL II BBS



LICENSE AGREEMENT

OS9L2BBS and the accompanying documentation are governed by the terms of the following license agreement.

You, the user, are granted a paid up license to use OS9 Level II BBS and accompanying documentation for an unspecified amount of time. You in no way become the owner of the package, nor do you have the right to sell or re-sell the package. You are allowed to make copies of this package for archival use only, any other copying is a violation of this agreement.

Help keep the price of software down...DON'T PIRATE!

DISCLAIMER

Although this software package is designed to prevent the accidental loss of data, the deleting of important files is still possible.

The author of this software is in no way responsible for the loss of time, money, or data as a direct or indirect result of the use of this software package.

OS9 Level II BBS
Release 3.00

TABLE OF CONTENTS

INTRODUCTION

OS9L2BBS is a set of commands and utilities that, when used together, create a sophisticated Bulletin Board System. In this manual a basic knowledge of OS9 will be assumed, if you do not know how to use OS9, or if you are confused about any references that this manual makes to OS9, please refer to your OS9 manual for assistance. For the most part, you must understand the following: How to use OS9 directory tree structure, how to edit, examine, and create text files, how to use the CMDS directory.

Also included in the package is a Bulletin Board disk. This disk contains a complete bulletin board setup for you to examine and use as you please. The board can be used "as is", modified, or completely changed for your board; we leave the creativity up to you. We do, however, suggest that you create your own menus and board design as this gives your board a personal touch. If you have problems, use this setup as a reference. By looking at the various files and how they interact you can get a better idea of how the system works.

OVERVIEW

OS9L2BBS is not a single program, but a set of commands that can be used in conjunction with the OS9 Level II Operation System to create a very sophisticated Bulletin Board System. To efficiently use OS9L2BBS you should be able to use the OS9 operating environment to your advantage.

The way a typical BBS, set up with this software, should run is as follows:

The command <u>Tsmon</u> should be run with the port name as a parameter. When Tsmon recognizes a caller it will automatically run two programs, <u>Login</u> and <u>Monitor</u>. <u>Login</u> will first list the 'motd' file to the caller, and then prompt him to log in. <u>Monitor</u> will simply monitor the carrier detect status on the port. Once the user logs in, <u>Login</u> will run the program specified for that user in the BBS.users file. Note that this can be any program, even a shell for sysop and superuser passwords. This program can be either a single command or a script file name. If it is a script file name, the command file should call the <u>Menu</u> program. The <u>Menu</u> program is the cornerstone of the system. This command displays a menu text file, then waits for users to input options. When the <u>Menu</u> program is finished, login will list the 'eotd' file to the caller and hang up.

If you do not know how to create and use script files, you should read your OS9 documentation. A script file is, quite simply, a file that contains a list of commands to be executed one after the other, like a program. To run a script file, all you have to do is type the name of the file. The biggest problem that you will encounter with script files is that the file becomes the primary input device. This means that if a program in a script file requires input it will take it from the file. This can be rectified, however, by putting a '</1' after every command in the file. This will cause it to take it's input from the terminal.

SOFTWARE INSTALLATION

To install OS9 Level II BBS Rel. II, using the complete built in set of menus, follow these steps:

```
(users with 2 disk drives)
Format 2 new disks (Format /d0)
Back up the original system disks (Backup /d0 /d1 #56k)
Put the backup 'OS9 Level II BBS System' disk in drive 0
Type the following:
        chd /d0
        chx /d0/cmds
        Install2

(users with 3 disk drives)
Format 3 new disks (Format /d0)
Back up the original system disks (Backup /d0 /d1 #56k)
Put the backup 'OS9 Level II BBS system' disk in drive 0
Type the following:
        chd /d0
        chx /d0/cmds
        Install3

(users with a hard disk drive)
Format 2 new disks (Format /d0)
Back up the original system disks (Backup /d0 #56k)
Put the backup 'OS9 Level II BBS System' disk in drive 0
Type the following:
        chd /d0
        chx /d0/cmds
        InstallH
```

NOTE: This installation procedure produces a working BBS system onto your backup disks. It is recommended that you back up these disks regularly.

HARDWARE INSTALLATION

Some hardware installation is necessary to run a bulletin board system. You should familiarize yourself with the hardware aspect of setting up a BBS system. There are many good books on telecommunications. The following instructions should help you get up and running as quickly as possible.

Materials needed: Auto answer modem
RS-232 Pak
25 pin RS-232 cable
Multi-Pak interface

1. Install the RS-232 Pak in slot 1 of the Multi-Pak
2. Connect one end of the RS-232 cable to the RS-232 Pak.
3. Connect the other end of the RS-232 cable to the modem (see modem manual for further instructions)
4. Connect the modem to the telephone line. (see modem manual for further instructions)

The modem is now connected to the computer.

NOTE: You should make sure that the following conditions are met.

1. Make sure that the following pins are connected on the RS-232 Cable: 2, 3, 8, 20.

2. Make sure that the modem is set so that the carrier detect line reflects the status of the phone line. (this is usually accomplished through a dip switch on the modem or a command sent to the modem).

3. Make sure that the modem is set so that it returns to command mode and hangs up the phone on loss of DTR. (This is usually accomplished through a DIP switch on the modem or a command sent to the modem)

FILES LIST

The following is a list of all files and commands that come on the OS9L2BBS distribution disk, and a small description of each.

Name	Description
System setup commands:	
Tsmon	autobaud terminal monitor
Login	login program
Monitor	carrier detect monitor
Menu	menu program
New_user	program to get new user info
BBS.build	simple file creator
BBS.list	simple file list command
BBS.validate	new user validation program
BBS.form	create user forms
BBS.stat	View user statistics
At	Event scheduler program
Prompt	Create user prompts
ViewBBS	Allows you to view BBS while running
AnsiCode	ANSI graphics code generator
AnsiEd	ANSI graphics editor
AnsiFilt	ANSI graphics -> OS9 filter
Suser	Set user number
Pause	Wait for a keypress
Chat mode commands:	
BBS.chat	chat request
Answer	answer chat request
Conference mode commands:	
BBS.who	who is currently on line
BBS.page	page another user
BBS.conf	enter conference mode
BBS.conf.who	who is in conference
Message retrieval commands:	
BBS.create	message base create command
BBS.post	message post command
BBS.convert	message conversion (from 2.0) utility
BBS.delete	message deletion command
BBS.pack	message pack command
BBS.read	message read command
BBS.reply	message reply command
BBS.forward	message forward read
BBS.new	message read new command
BBS.scan	message scan command
BBS.search	message search command
Mail retrieval commands:	
BBS.mail.post	mail posting command
BBS.mail.check	mail checking command
BBS.mail.read	mail reading command
BBS.mail.readd	mail read & delete
BBS.mail.delete	mail delete command

File retrieval commands:

BBS.upload	upload prompting command
Uloada	upload (ascii)
Uloadx	upload (xmodem)
Uloadxc	upload (xmodem CRC)
Uloady	upload (ymodem)
BBS.download	download prompting command
Dloada	download (ascii)
Dloadx	download (xmodem)
Dloadxc	download (xmodem CRC)
Dloady	download (ymodem)
DLD.validate	new uploads validation
DLD.add	add new files (without upload)
DLD.unvalidate	un-validate old uploads
DLD.search	search for a download
DLD.list	list downloads
DLD.read	read file information

These commands are further explained in the following pages.

SYSTEM SETUP COMMANDS

TSMON

Syntax: Tsmon [-m] <port>

Tsmon is the first command run to start the execution of the board. The Tsmon command monitors a port (such as /t2) for an <enter> keypress. If a character is received that isn't an <enter> tsmon assumes that the baud is set wrong and switches baud rates. It then waits again for an <enter>. Tsmon will continue cycling like this through baud rates until it gets an <enter> keypress. The baud rates cycle as follows:

1200 Baud
2400 Baud
300 Baud
1200 Baud
.
.
.

Once tsmon receives an <enter> character it sends a message stating what baud, parity, and stop bits are operating. Tsmon then automatically runs two commands, Login and Monitor. If the -m option is specified, the Monitor command will not be executed. These commands are explained further in the section for that command. Tsmon should be run as a background task as follows:

```
Tsmon /t2 &
```

That way the board runs completely in the background. If Tsmon is run in the foreground, it will not allow you to use that window any longer. For this reason it is suggested that you run Tsmon in the background.

You can run the board in a window for debugging. To do this enter the following commands:

```
iniz /w7
display c >/w7
Tsmon -m /w7 &
```

NOTE: I used '/w7' here because it automatically defaults to 80 columns. You can use any window you like. Once you have run this command, simply press <CLEAR> until you see the window, then press <ENTER>. The board should respond with '1200 baud 8 bits no parity'. You can now access the board as if you were on line.

Another feature of Tsmon is it's modem setup feature. With this feature you can have Tsmon reset your modem every time someone logs off. When Tsmon is first run, and every time a user is logged off, it will look for a file called "modem.set" in the current working directory. If the file does not exist, this feature is ignored, if the file does exist, Tsmon sends whatever information is in that file to the modem AT THE BAUD RATE THE PORT WAS XMODED TO. This means that if you use Tsmon you MUST Xmode your port to the modem's baud rate. This file should contain any commands that you need to send to your modem to have it initialized. My modem.set looks as follows:

```
ATS0=1
```

This is because I don't always run the BBS, but when I do, it automatically sets it to answer on the first ring.

NOTE: 2400 baud user MUST have this file and must have at least the text "AT" in the file. This is due to the fact that 2400 baud modems will not automatically jump to their highest rate.

LOGIN

Syntax: Login

The Login program allows you to maintain a list of users and their passwords and user numbers. This program is automatically run by Tsmon and need not be run by the sysop. This is the program that prompts the person logging on for his name and password.

The Login program requires several files to run. The first file it looks for is 'motd'. This file is listed to the user's screen immediately after he calls. It should be used as a welcome message, and should provide any instructions necessary for logging in. A default motd file is provided, but it is suggested that you create your own. You can use any editor to create this file. This file should be in the same directory as Tsmon is run from, usually '/dd/bbs'.

The Login program next looks for a file called BBS.users. This file contains the name, password, program to run, and user number for every user of the bulletin board. This file is a simple text file that can be created and edited by any editor. The format each line in this file is as follows:

```
user name,password,program,number[,time]
```

Where user name is the user's name, password is the user's password, program is the name of a program or script file to run, number is a number that you assign to each user (You should assign yourself as SYSOP 0), and finally the time (which is optional) is the time the user is allowed on the system PER DAY. If this parameter is omitted or is set to 0, the user will not have a time limit and can access the system as long as he wants.

Note that the program to run can be a script file. I would suggest this as you could have the file list special messages and bulletins that you create, read and check mail, and then run the menu program. Here is a sample command file:

```
-x
BBS.list bulletins
chd mailroom
bbs.mail.read
chd ..
BBS.list sysop_message
menu bbsmenu bbscmds </1
```

This command file would list the file called "bulletins", read the mail, list a file called 'sysop_message' and run the menu program. The -x stops the board from hanging up if an error occurs.

Another file that Login looks for is a file called 'Userlog'. If the file does not exist, login creates it and puts the user's name and time logged on in it. This allows you to maintain a list of all callers and the time they called.

Login also prints the person's name and the time they called on the printer (if connected).

The last file that Login looks for is a file called 'eotd'. This file is displayed after the command that was executed is finished. In the case of the menu program, this happens when a specific option (that you define) is pressed. This file should contain an appropriate sign off message of your choosing.

Still another file that Login looks for is "/dd/bbs/BBS.userstats". If this file does not exist, Login will create it. If it does exist, Login will look for the current user's entry in the file. If the current user's entry is not found, an entry is created for him. Note that this means that you NEVER have to fool with the userstats file, it will be maintained automatically. Login will then store the time and then run the program you specified. When you exit from the menu, Login will update the amount of time remaining for the next login. This allows users to only access the board for a preset amount of time each day. If a time of 0 is given, or no time at all is given, the user will not have a time limit and can access the board as long as he likes. Notice that this is compatible with release 2.0 of the BBS, so you do not have to change your BBS.users file from release 2.0 if you don't want to give time limits.

MONITOR

Syntax: Monitor

The monitor program will monitor the carrier detect status of the port. If the carrier detect is lost the program will kill all processes started by it's parent process and any of it's children. This means that if someone hangs up, any processes started by that person will be killed.

NOTE: This program is automatically run by Tsmon and need not be run by the system operator. This program is also very hardware dependent. It works with the Deluxe RS-232 Pak from Radio Shack as well as many 'compatible' paks (including the Orion Technologies TELEpak).

MENU

Syntax: menu <menufile> <commandfile>

The menu command allows the sysop to create his own menus to be used by the users of his bulletin board. This command is the cornerstone of the whole BBS system. It allows the sysop to define what keys will be pressed to execute what commands. It also allows the sysop to define all of his own menus. The menu is a text file that is 'echoed' or sent to the user's screen. This file can even contain special graphics characters, such as the popular ANSI standard.

The menu command requires two files as follows:

A menu file, which is simply a text file that will be listed to the user's screen. This file can be created using any editor (edit, scred, tsedit, BBS.build, or AnsiEd (for ANSI graphics)). Control codes can also be inserted into this file for special effects. The control codes will appear as dots when you type them in. To clear the screen the control code is <CNTRL><L>. To ring the bell the code is <CNTRL><G>. You can experiment to find other special codes.

Example menu file:

```
^L
+----------------------------+
|  Files   menu              |
+----------------------------+
|                            |
| L> List files              |
| U> Upload a file           |
| D> Download a file         |
+----------------------------+
```

The next file required is a commands file, which is a text file that instructs the menu program how to use options. Each line in this file contains either a single letter option followed by the OS9 commandline to execute, or a user priority condition, followed by a single letter option followed by the OS9 commandline to execute. The user priority condition is in the form <#####, >#####, or =##### where ##### is a five digit user number. The user number MUST be 5 digits. If the number is less than 5 digits it must be padded with 0's. The <, >, and = symbols stand for less than, greater than, and equal to respectively. If the user priority is specified, the menu program first checks that the user can access the option, then if the user's number meets the specified condition (<, >, or =) he is allowed to execute the option.

Anything that is legal on a command line is legal on these lines, including semicolons, memory modifiers, i/o redirection, etc. On the last line of this file you should put a slash (/) and the prompt that you want to use.

Example commands file:

```
L dir downloads
u uloadx
d dloadx
=00000 s shell
>01000 v dld.validate
/Enter your choice:
```

The command file has some special commands that can be used instead of regular commands. These special commands are:

```
chd  <dirname>
chx  <dirname>
chm  <menufile> <commandfile>
ex
```

These commands must be on a line by themselves, they cannot be executed with any other commands.

The change directory command (chd) changes to the specified directory, reads in the new menu file (in the new directory) and this becomes the new menu. This allows you to have sub-menus simply

by putting them in sub-directories and giving them the same name. If you look on the sample disk you will see that this is how the menus are set up. Changing directories therefore changes menus as well. With this system you can easily remember how your system is set up by looking at how your directories are set up.

The change execution directory command (chx) will change the current execution directory and re-display the current menu.

The change menu command (chm) will change the menu file names. This allows you to have more than one menu in each directory.

The exit command (ex) exits from the menu program and returns to the calling program. If the calling program was Login, the user will be disconnected after receiving the 'eotd' file.

Example special commands:

```
c chd coco
o chd os9
=00000 s chd SYSTEM
m chd /dd/bbs
8 chm bbsmenu_80_col bbscmds
4 chm bbsmenu_40_col bbscmds
g ex
/Enter Your Choice?
```

Another feature of the Menu program is timekeeping. The Menu program, when it is first run, checks the time remaining. It then keeps track of this time and will log the user off when his time is up. The time remaining is set up by the Login command using the value given in the BBS.users file (see Login for more information).

BBS.BUILD

Syntax: BBS.build <filename>

The BBS.build command is a simple file creator/editor similar to the editor used for posting messages. This editor is provided to the sysop to add his own features to the board. The build command will not edit an old file, it will only create a new one.

One example use for this command is to create feedback to the sysop on his printer. To do this simply have as a menu option '1 BBS.build /p'. This will allow someone to type data directly onto the printer.

BBS.LIST

Syntax: BBS.list <filename>

The BBS.list command is a simple file list command similar to the 'List' command that comes with OS9. The important difference is that the person on-line can press the <SPACE> bar and abort the listing. This allows a person to cancel a long listing.

BBS.VALIDATE

Syntax: BBS.validate

The BBS.validate command allows you to easily add entries to your BBS.users file. This command will prompt you for all necessary information for a new user, including his alias (to update the /dd/bbs/BBS.alias file. This command can be used as an alternative to using a text editor to add new users.

BBS.FORM

Syntax: BBS.Form <formpath> <pathname>

The BBS.Form command allows you to design your own forms for users to fill out. This command is useful for getting information from your system users. It can be used to get the necessary information from new users. Note that this is used in the new_user menu for that purpose. The first file, the <formpath>, is a text file that describes the format of the form. You create this file with any text editor. All you have to do is enter one question on each line of this file. This question will be sent to the users's terminal and he will be allowed to type a response. The question asked, and the user's response is then sent to <pathname>, the second parameter. Note that this can be '/p' as well as a file. If the specified file does not exist, it is created, if it does exist, it is appended to.

BBS.STAT

Syntax: BBS.stat

The BBS.stat command allows users to view their user statistics. This command outputs the user's statistics in an easily readable format. The user statistics are maintained in a file call "/dd/bbs/BBS.userstats" and this file must exist for the BBS.stat command to work. This file does NOT need to be created by you. It will be automatically created by Login. This file will be updated by the BBS commands. The following statistics are maintained: Last logged on, times logged on, number of files uploaded and downloaded, the number of messages left and read, and the amount of time remaining for this login.

AT

Syntax: At yy/mm/dd hh:mm <commandline>

The At command allows you to execute commands at a given date and time. The <commandline> parameter is any valid OS9 command line (that you would type at the OS9: prompt). The date is specified in OS9 format, year/month/day hour:minute (note that the time is 24 hour (military) time). If a date of 00/00/00 is specified, the event will occur EVERY day at the specified time. Note that this is useful for things like automatically deleting the daily userlog at 12:00 midnight every night (whether your there or not). It is also useful for packing message bases late at night when the BBS traffic is low.

Example:

```
At 00/00/00 00:00 del /dd/bbs/userlog
```

NOTE: the time 00:00 is 12:00 midnight in 24 hour time.

PROMPT

Syntax: Prompt <command> "Parameter Prompt" <other parameters>

The prompt command is a very powerful command that allows you to create prompts for your BBS. You can prompt the user for parameters to be passed to a command. For instance, suppose you wanted a Co-Sysop menu to include the DIR command, but didn't want to give shell access to the Co-Sysop. You could implement this in the menu commands file as follows:

```
.
.
d prompt dir "Enter sig name:"/files
.
.
```

This would list the files directory in the SIG specified. This command is provided as a convenience to the Sysop and can increase the power of the BBS's menus.

VIEWBBS

Syntax: ViewBBS

The ViewBBS command allows you to view the board while it is running IF YOU HAVE INSTALLED THE PROPER DRIVERS! In order for this command to work, you MUST have installed the /Td device as well as the /Wb window (included). This command will effectively do a <CLEAR> to /Wb. See the section on the /Td driver for more information.

ANSICODE

Syntax: AnsiCode <codes>

The AnsiCode command allows you to send ANSI code sequences through the BBS. ANSI codes are used to control colors etc. on ANSI terminals. Some of the more important codes are listed below.

Code	Meaning	Code	Meaning
01m	Bold Face On	04m	Underline On
05m	Blink On	07m	Reverse Video On
08m	Invisible On	00m	Reset Graphics
30m	Foreground Black	40m	Background Black
31m	Foreground Red	41m	Background Red
32m	Foreground Green	42m	Background Green
33m	Foreground Yellow	43m	Background Yellow
34m	Foreground Blue	44m	Background Blue
35m	Foreground Magenta	45m	Background Magenta
36m	Foreground Cyan	46m	Background Cyan
37m	Foreground White	47m	Background White

Example:
```
AnsiCode 33;44m
```

Notice how the codes can be stacked, this command would set the foreground to yellow and the background to blue.

ANSIFILT

Syntax: AnsiFilt <inputfile or <command> ! AnsiFilt

AnsiFilt is a filter that convert ANSI codes into their OS9 screen code equivalents. This command is a filter so that you can either use it by redirecting its input or by piping the output of another command through it. This command is useful for testing the BBS's menus.
Example:
```
        AnsiFilt <BBS.menu_am
```

Example:
```
        AnsiCode 33;44m ! AnsiFilt
```

Note that the second example allows you to check the results of the AnsiCode command so that you can see how it will look.

ANSIED

Syntax: AnsiEd [<inputfile>] [<outputfile>]

AnsiEd is an ANSI graphics editor that allows you to create great looking screens and menus with ANSI graphics. The complete description on how AnsiEd works is discussed at some length later in the AnsiEd section of this manual.

SUSER

Syntax: Suser <number> [<commandline>]

Suser allows you to change the current user number. This allows you to check for mail etc., without being user 0. Note that this command can only be run by the Sysop (user 0). If no commandline is specified, a shell will be started.

CHAT MODE COMMANDS

BBS.CHAT

Syntax: BBS.chat

The BBS.chat command allows a person on a remote terminal to communicate with someone (usually the sysop) at the local BBS site. The BBS.chat command will cause a bell to ring on the local BBS's terminal. If no one answers the bell (see the answer command) the person issuing the chat request will be informed that the sysop is not available.

NOTE: The chat command requires that the local terminal has at least one unused window available. (this is to ring the bell).

ANSWER

Syntax: Answer

The answer command is used to answer a corresponding chat request. Simply enter 'Answer', and you will be able to communicate with the person on-line. When you are finished chatting, you can terminate the chat session by pressing <ALT><X>. The person issuing the chat request cannot terminate the chat. If no one answers the chat request in 10 rings the chat is terminated.

CONFERENCE MODE COMMANDS

BBS.WHO

Syntax: BBS.who

The BBS.who command gives a list of the users currently on line. This command can be used in a menu to allow users on line to see who is currently on the system. This command can be used together with BBS.page to initiate conference modes. This command is also useful to the Sysop when he wants to know who is on-line.

NOTE: This command requires that the file '/dd/bbs/bbs.alias' exist.

BBS.PAGE

Syntax: BBS.page

The BBS.page command allows a user to send a brief one-line message to another user that is logged on at the same time (but a different port). The command prompts the user for the person to page. The program then checks to see if that user is currently on-line. If that user is on-line he will get the message at the next available time. Note that this may take a few seconds, the user initiating the page will be informed to wait.

NOTE: This command requires that the file '/dd/bbs/bbs.alias' exist.

BBS.CONF

Syntax: BBS.conf

BBS.conf is a system command used to create a 'conference mode'. When BBS.conf is run by two separate people at the same time, each user can see what the other types. This is similar to the conference mode on DELPHI and other multi-user systems. Once the program is run you will be prompted for a handle. This handle will be displayed before each line of text as you type. If you press <CTRL><X> while in conference mode, a list of users currently in conference will be displayed. If you press <CTRL><Z>, conference mode will be exited. Each user in conference mode should have a different user number. If more than one person is in conference with the same number, only one may see the messages.

NOTE: This command requires that the file 'Conf.dat' be in the commands directory and that the file '/dd/bbs/bbs.alias' exist.

BBS.CONF.WHO

Syntax: BBS.conf.who

BBS.conf.who is a system command that displays a list of the users currently in conference mode (running BBS.conf). This allows users to see who is in conference without actually entering the conference mode.

NOTE: This command requires that the file '/dd/bbs/bbs.alias' exist.

MESSAGE RETRIEVAL COMMANDS

BBS.CREATE

Syntax: BBS.create

The BBS.create command creates the files necessary to start a message base. By executing this command you create a message base with 0 messages. If a message base already exists, BBS.create will return in error. This command should be run in each sub-directory that is intended to be used for a message base.

BBS.POST

Syntax: BBS.post

The BBS.post command will add a new message to an already existing message base in the current directory (see BBS.create). The person issuing this command (usually a BBS user) will be prompted for a message subject. After the subject of the message is entered, the user will be prompted for who the message is addressed to. Note that the user can address a message to all users by simply pressing <ENTER> at this prompt. After entering who the message is addressed to the user will be allowed up to 100 lines to enter his message. The post facility uses a simple, user-friendly, editor for posting this message. This editor will word-wrap at the end of each line.

NOTE: This message base is not completely compatible with release 2.0. Release 2.0 does not allow a message to be addressed to a specific user. A utility for converting message bases is provided with the board. This utility (BBS.convert) will convert a release 2.0 compatible message base to a release 3.0 compatible message base. See the BBS.convert command for more information.

NOTE: This command requires that the file /dd/bbs/BBS.alias exist with a list of user names or aliases and their numbers.

Example:
Sysop,0
The main man,1
Joe Smoe,2
.
.
.

This allows messages to be posted by alias names instead of real names. If you wish to use the real name as the alias, that's fine.

NOTE: This command also updates the /dd/bbs/BBS.userstats file.

BBS.CONVERT

Syntax: BBS.convert [<dirname>]

The BBS.convert utility is specifically for those people who own release 2.0 of this system and are upgrading to release 3.0. If you do not own Release 2.0 of this system, disregard this command. If you do own release 2.0 of this system, you should update your message bases. Note that it is not ABSOLUTELY necessary, but if you don't all previously posted messages will be addressed to random users (possibly unknown). When you run BBS.convert it addresses all previously posted messages to "All Users". Here are some examples:

BBS.convert /dd/bbs/sigs/coco

NOTE: the <dirname> is optional, current dir is assumed.

BBS.DELETE

Syntax: BBS.delete

The BBS.delete command allows someone to delete his own messages. The sysop is allowed to delete anyone's message. If someone tries to delete someone else's message he will be given the message 'You cannot delete that message.' When the BBS.delete command is issued, it gives the total number of messages and asks which message number to delete. If that person owns the message or is the sysop (user 0) then the message will be deleted.

NOTE: When a message is deleted it still occupies space until it is packed. (See the BBS.pack command).

BBS.PACK

Syntax: BBS.pack <dirname>

The BBS.pack command removes all deleted messages from the specified message base. When a message is deleted, it is really only flagged for deletion. To be completely removed the message base must be packed. This is done in order to save time for the BBS user. Packing a message base requires a lot of time for large message bases, I therefore recommend that packing be done during board off hours (early in the morning, late at night, etc.) to reduce user inconvenience. If the directory name parameter is omitted, the message base in the current directory (if any) will be packed.

NOTE: A message base should be packed when it has a significant amount of deleted messages or when more disk space is needed.

BBS.READ

Syntax: BBS.read

The BBS.read command allows someone to read messages previously posted (see the BBS.post command). The BBS.read command will tell the user how many current messages exist, what his current message # is, and prompt him for what message number to read. The user can also specify that he wants to read the next message, previous message, next thread message (next message with the same subject as the current message) or reply to the current message. The message will then be listed. The user can then continue entering message numbers, or enter 'q' to quit.

NOTE: This command also updates the /dd/bbs/BBS.userstats file.
NOTE: This command requires that the /dd/bbs/BBS.alias file exists.

BBS.FORWARD

Syntax: BBS.forward

The BBS.forward command works much the same as the BBS.read command, except that it reads all messages following a certain message number.

NOTE: This command also updates the /dd/bbs/BBS.userstats file.
NOTE: This command requires that the /dd/bbs/BBS.alias file exists.

BBS.NEW

Syntax: BBS.new

The BBS.new command automatically reads all messages with a number higher than the last message read by that user. The BBS.read, BBS.forward, and BBS.new commands keep track of the last message read by each user. The BBS.new command can then read any message not previously read.

NOTE: This command also updates the /dd/bbs/BBS.userstats file.
NOTE: This command requires that the /dd/bbs/BBS.alias file exists.

BBS.SCAN

Syntax: BBS.scan

The BBS.scan command is used to read the headers of all messages in the message base. When the BBS.scan command is executed it tells the user how many messages exist. It then prompts for the first message number to scan. The message number, person who posted the message, date, and message subject are then displayed for those messages with a number higher than the number entered.

BBS.SEARCH

Syntax: BBS.search

The BBS.search command is used to search for a certain message subject. When the BBS.search command is executed it prompts the user for the text to search for. The text is then searched for in the message subjects. A listing of what message subjects contain that text is then displayed.

The text does not have to be the entire message subject, it can be only a part. For example, if you wanted to find all messages dealing with modems you could enter 'modem' as the text to find. All message subjects containing the word 'modem' would then be found.

MAIL RETRIEVAL COMMANDS

BBS.MAIL.POST

Syntax: BBS.mail.post

The BBS.mail.post command allows a BBS user to send mail to another BBS user. When the command is implemented it prompts for the name of the person to whom the mail should be sent. If the person's name entered is found in the '/dd/bbs/bbs.aliad' file, the user is then prompted to enter the message subject. Once this is entered the user is given a maximum of 100 lines to leave his message. The mail post command uses the same user friendly editor as BBS.post and BBS.build. Once the message is entered the mail is put on file. When the receiver asks to read mail he will receive the message. Mail is private, therefore only the person receiving the mail can read his mail. The Sysop can however list the entire mail file if he pleases. The sysop can also use the 'suser' command to set the user number, and check a specific user's mail.

NOTE: BBS.mail.post requires that a file exists called /dd/bbs/BBS.alias which has a list of all user names or aliases and their user number.Example:

```
Sysop,0
The main man,1
Joe schmoe,2
.
.
.
```

This allows alias names to be used instead of regular name, but still allows the sysop to maintain a list of real names.

BBS.MAIL.CHECK

Syntax: BBS.mail.check

The BBS.mail.check command allows a BBS user to see if the mail he sent has been received. The BBS.mail.check command immediately tells the user what messages have not yet been received. A user can only check the mail that he has sent.

BBS.MAIL.READ

Syntax: BBS.mail.read

The BBS.mail.read command allows a BBS user to read any mail that has been sent to him. The message will be listed along with who sent it. The user will then be asked if he wished to re-read the message (in case it scrolled off of his screen). If not the command will continue with the next message etc.

NOTE: The messages will not be deleted with this command (See BBS.mail.delete and BBS.mail.readD)

BBS.MAIL.READD

Syntax: BBS.mail.readD

The BBS.mail.readD command is exactly like the BBS.mail.read command, except that if any messages were present BBS.mail.delete is automatically called to delete them.

BBS.MAIL.DELETE

Syntax: BBS.mail.delete

The BBS.mail.delete command will delete any mail sent to the user WITHOUT READING IT. This command should only be executed after the mail has been read (this can be done automatically with BBS.mail.readD.)

FILE RETRIEVAL COMMANDS

BBS.UPLOAD

Syntax: BBS.upload [dirname]

BBS.upload is an integrated file upload system. It prompts the user to enter the upload protocol desired. The program then runs the appropriate upload program. If the file is uploaded without error, the user is prompted for a one line description of the file. This description is then placed in a list file for later validation. This program requires that the programs Uloadx, Uloadxc, and Uloady be in the current execution directory.

NOTE: This command also updates the /dd/bbs/BBS.userstats file.

ULOADA

Syntax: Uloada [<filename>]

Uloada uploads a file using a simple ASCII protocol. With this protocol, you simply send the information as an ascii text file. When you are finished uploading, simply type <CTRL><T> and upload will be terminated.

ULOADX

Syntax: Uloadx [filename]

Uloadx uploads a file using the Xmodem protocol as implemented by Ward Christensen. This program is automatically run by BBS.upload and need not be run by the sysop. If the filename is not specified, the user is prompted for one.

ULOADXC

Syntax: Uloadxc [filename]

Uloadxc uploads a file using the CRC version of the Xmodem upload program. This command is automatically run by BBS.upload and need not be run by the sysop. If the filename is not specified, the user is prompted for one.

ULOADY

Syntax: Uloady [filename]

Uloady uploads a file using the Ymodem protocol. This command is automatically run by BBS.upload and need not be run by the sysop. If the filename is not specified, the user is prompted for one.

BBS.DOWNLOAD

Syntax: BBS.download [dirname]

BBS.download is an integrated file download system. It prompts the user to enter the download protocol desired. The program then runs the appropriate download program.

NOTE: This command also updates the /dd/bbs/BBS.userstats file.

DLOADA

Syntax: Dloada [filename]

Dloada is a download system command that allows the user to download files using the ASCII protocol. With this protocol, the ASCII text is simply sent through the BBS and the user can use a capture buffer to receive the information.

DLOADX

Syntax: Dloadx [filename]

Dloadx is a download system command that allows the user to download files from the board using the Xmodem protocol implemented by Ward Christensen. If the filename is not specified, the user is prompted for one. This command is automatically run by BBS.download.

DLOADXC

Syntax: Dloadxc [filename]

Dloadxc is a download system command that allows the user to download files from the board using the CRC version of the Xmodem protocol. This command is automatically run from BBS.download and need not be run by the sysop. If the filename is not specified, the user is prompted for one.

DLOADY

Syntax: Dloady [filename]

Dloady is a download system command that allows the user to download files from the board using the Ymodem protocol. This command is automatically run from BBS.download and need not be run by the sysop. If the filename is not specified, the user is prompted for one.

DLD.VALIDATE

Syntax: DLD.validate [dirname]

DLD.validate is a download system command that allows the system operator to validate uploads. When this program is run, the user is given a list of files not yet validated for download. The user can then either validate the file, download the file, or go on to the next file. If the user decides to validate the file, he will be prompted for a one line description of the file (64 character max.), a list of keywords on which the file can be searched, and a paragraph description of the file. The BBS users can then use this information to find and access the file. If dirname is not specified, the current directory is assumed. This is true of all up/down load commands.

DLD.ADD

Syntax: DLD.add [dirname]

DLD.add is a download system command that allows the sysop to post downloads, without having them be uploaded. This allows the sysop to add new files to the current database without having to upload them.

DLD.SEARCH

Syntax: DLD.search [dirname]

DLD.search is a download system command that allows users to search for files to download using keywords. These keywords are the keywords entered by the sysop when he validates the program. When this command is run the user is prompted for a keyword to search for. The user will then be given a list of all files that are associated with that keyword.

DLD.LIST

Syntax: DLD.list [dirname]

DLD.list is a download system command that allows users to list all files currently validated for downloading in the specified directory. When the command is run the user is given a formatted listing of all files currently available.

DLD.READ

Syntax: DLD.read [dirname]

DLD.read is a download system command that allows users to read the paragraph description of a file validated for download. When this program is run the user will be prompted for the name of the file to read. If the file has been validated for download, the paragraph description entered by the sysop will be displayed.

MENU SYSTEM

OS9 Level II BBS Release II comes with a complete set of menus already created and ready to run. You can change these menus as you like, or create completely new ones! Have fun adding new sections to your board and making it better. You should take special notice of how the menu system is currently set up, as this will help you to create your own additional menus. For more information on menus, refer to the Menu command explanation.

In an effort to help you understand how the menu structure works, the following is an explanation of the sample menu setup.

There are 3 individual sets of menu files in each directory that come with the system. There are also 3 special sets of menu files in the BBS directory for special features. These menus are defined as follows:

Menu file	Commands file	Description
bbs.menu_32	bbs.cmds	user with a 32 column screen.
bbs.menu_80	bbs.cmds	user with an 80 column screen.
bbs.menu_am	bbs.cmds	user with ANSI graphics
bbs.menu_sl	bbs.cmds_sl	menu to select type of menu to use
bbs.menu_sys	bbs.cmds_sys	Sysop menu
bbs.menu_new	bbs.cmds_new	New user menu

The bbs.menu_32, bbs.menu_80, and bbs.menu_am are the menu text files for various terminal types. Note that all of these menu text files use the same (bbs.cmds) commands file. This is because they all execute the same options, they just look different!

The other three menus are explained as follows:

bbs.menu_sl allows users to select what type of menus to use when they log on. It lists the three types, 32 column, 80 column, and ANSI graphics. This allows users to log in from different terminals and not run into problems because of the wrong terminal type.

bbs.menu_sys allows the sysop to log in and perform various functions of the SYSOP from a remote location. Note that these files to DO NOT have the public read attribute set. This helps to prevent other users from using this menu, as only user 0 can read this menu.

bbs.menu_new is a special menu for new users. This menu allows only limited access to the BBS and is used mainly for new user validation. A line in your BBS.users file should give access to this menu for new users.

The bbs.cmds file uses user numbers to determine priority. A user number less than 1000 is determined to be a normal user. Any numbers greater than 1000 is considered an elite user and is allowed access to the elite user menu. This standard is compatible with that set forth in Release 2.0 of this system. NOTE that this is only a standard and is in NO WAY ENFORCED! If you want to change your user numbering system, possibly to include many different levels of users, this is completely up to you. The Menu command provides you with MAXIMUM flexibility in this area, you can set up as many user levels as you like!

The way you set up the board is completely up to you. I do suggest that you add your own special touch, as this gives the board a personal touch.

CREATING/CHANGING THE BULLETIN BOARD

If you do not wish to use the built in menu structure, you can create your own menu structure. The procedure for doing this is described below. You may want to read this section even if you intend to use the built in structure, as it will give you more insight into how the system works.

MAKE A BOOT DISK

The first thing that must be done to create the new BBS system is to create a boot disk. You can do this using either 'Cobbler', 'OS9gen', or 'Config' all of which are provided with your OS9 Level II disk. The only thing you have to make sure of is that your boot disk has the proper ACIA drivers to run the ACIAPAK that you have, and that it has the proper /dd descriptor for your default disk drive. The board requires that these two devices be available.

CREATE THE DIRECTORY STRUCTURE

The next thing that has to be done is to create the directory structure for the board. The board requires at least two directories on your disk to run. These are the CMDS directory and a directory on your default drive (/dd) named BBS. Other directories and drives can be used as you please, but here are some ideas that you can use.

Usually you would want to either create a new directory or use another disk drive to hold the user mail. You can name this directory anything you want, I named mine MAILROOM. Any time you want to deal with mail you should change to this directory (chd).

Another directory that you might want to create is a directory to put the main message base. This can also reside on another disk if you like. I named my major message base directory BOARD, but the naming is strictly up to you.

Another possibility for message retrieval is to create several Special Interest Group directories. This again you might want to put on a different disk drive, or possibly on the same drive as the main message base, this is entirely up to you. I called mine COCOSIG and OS9SIG.

Uploads and downloads will also have to be put somewhere. There are several ways of doing this as well. You could have a directory under each Special Interest Group for that groups files, or you could have one large file storage section for all files, or some combination of both. I created one directory for all files called FILES.

COPY COMMANDS TO THE DISK

The next step is to copy the commands from the distribution disk to your boot disk. To do this you can use the Dsave command that comes with OS9, or if you have the OS9 toolkit ($19.95) you can use the Wcopy command. I would recommend purchasing the toolkit, as it will save you a lot of labor in managing the board.

BUILD THE MOTD FILE

You are now ready to build the 'motd' file. This file is listed to the users screen when he logs on and must exist in the directory that tsmon is run from. You can put whatever text you want in this file, but it should contain the name of your board and any instructions necessary to log in.

BUILD THE EOTD FILE

You should now build the 'eotd' file. This file is listed to the users screen when he logs off. You can put whatever text you want in this file, but it should give some friendly log off message.

BUILD THE MENU FILES

This is the most difficult and laborious process in creating the board, but is also the one that allows you to be the most creative.

You should create a menu file in each directory that you want users to be able to access. In that menu you should have options for every action that you want to take place in that directory. Options can be any OS9 command, BBS command, or even your own programs.

You can name the menu files anything that you want, but I suggest that you use names that have meaning to you. Any menus in sub-directories MUST have the same name as menus in a previous directory that calls it. This means that your menu structure should follow your directory structure and the menu files should all have the same name, but be in different directories. For example, if you named your menu 'BBS.menu' (as I did), and your command file 'BBS.cmds' (as I did), then every directory that you want to call should have those two files in it. You then call the new menu by simply doing a 'chd' command to the new directory. The menu program will then automatically use the new menu files.

To actually create the menu, you can use any editor that you have available (Build, Edit, Scred, Tsedit, or even BBS.build). You should create the menu file exactly as you would like it to appear to the user, with the addition of control commands for screen formatting (See Menu for more information).

Once the menu file is created you must create the command file. This also can be done with any available editor. The command file should have, on each line, a single letter option followed by the command line to execute for that options. The command line can be any OS9 command, BBS command, command file, or anything else that would be valid on an OS9 command line. The only difference is the special commands (See Menu for more information). These special commands must be on a line by themselves. Typically the commands that you would put in a message base would be BBS.post, BBS.delete, BBS.forward, BBS.scan, BBS.new, etc.

These two menu files must be created for every sub-directory that you want to have. These sub-directories then become sub-menus to perform different functions.

CREATE ALL MESSAGE BASES

You must now go to every directory that will contain a message base and execute BBS.create. This will initialize the message base so that new messages can be posted. If this is not done, any calls to BBS.post will return in error.

CREATE USER LIST FILES

The next step in setting up the board is to construct the user list files. These files have to be BBS.users and /dd/bbs/BBS.alias. The BBS.users file must be in the same directory as tsmon is run from, I suggest using /dd/bbs to avoid confusion. The BBS.users file must have the user name IN CAPS, the user password IN CAPS, the command to execute (usually menu or a command file), and the user's unique number. You must assign each user a number. Once you have created the user file you must create the alias file. This file contains the user's name or alias followed by a comma and the user's number. This is the name that the user will be referred to when posting mail and messages. If you do not want to use aliases, simply put the user's real name for his alias (this name does not have to be in caps).

ATTRIB ALL USER FILES

Once you have all of these files created you must use attr to give public read access to any file that a user will need to access. This does NOT include BBS.users, motd, or eotd. It does however include all menu files. If you have the OS9 toolkit ($19.95) you can use the Wattr command to attrib all files quickly.

CREATE A STARTUP FILE

Although it isn't necessary, you should create a startup file on your boot disk. This startup file should set the time, change to the BBS directory, and do a 'Tsmon <port> &'. That way all you need to do to run the board is place the proper disks in their drives and press reset.The board should now be ready to run. If you have problems step through the creation process again and make sure that you have created all of the proper files. (NOTE: A major source of problems is when files have the wrong attributes, make sure all files have appropriate attributes)

MAINTAINING THE BULLETIN BOARD

Once the board is set up it should be a fairly simple task to maintain it. There are only a few things that the sysop should make sure he takes care of when operating the board.

SET THE DATE AND TIME

The sysop should always set the date and time properly when he boots the system. All message files and mail files use the system date to determine when messages get posted. If the system date is wrong it could cause confusion to the users of the board.

READ THE USER FEEDBACK

The best source of information about the board will come from it's users. There may be problems with the board (such as files having the wrong attributes) that you never noticed. As a sysop it is your responsibility to read your mail, messages, and feedback as often as possible.

A WORD ABOUT UPLOADS

As a sysop it is also your responsibility to maintain the files section of your board. It is a good idea to check out all programs that are uploaded before posting them for download. This way you can help prevent the spread of virus and trojan horse programs.

INITIATING NEW USERS

You should set up some way of initiating new users onto your board. This can be achieved by putting a special logon in the users file called 'NEW USER'. You should also assign some appropriate password. You can then give the instructions for new users in the 'motd' file. The new user logon should run the command 'New_user'. This command will get all of the person's information and store it in a file or on the printer. (See New_user for more information)

ANSI - EDITOR

Syntax: AnsiEd [<inputfile>] [<outputfile>]

<inputfile> is an ANSI graphics file to be loaded in for editing

<outputfile> is an ANSI graphics file to be output when finished editing.

The ANSI graphics editor allows you to create colorful and dynamic ANSI color screens on your COCO III. ANSI codes are a set of special codes used to format output on a computer terminal screen. The ANSI codes include codes for changing colors, positioning the cursor, and setting character attributes (such as underlining and blinking). To use the ANSI editor, you do not need to know the codes, they will be automatically generated.

The ANSI editor has a special 'Animation' feature that allows you to do special effects with ANSI codes. This allows you to have dynamic color displays that move and change. This special feature, however, makes the editing somewhat more difficult.

To implement the animation effects, two buffers are used; a screen buffer, and an output buffer. The screen buffer can be edited at will, the output buffer cannot be edited. To enable you to put information in either of these buffers, the editor has two modes, EDIT mode and RECORD mode. In addition to this, the editor has a way to put the screen buffer into the output buffer. This allows you to edit information on the screen until you get it looking right, then send it to the output buffer. When you save a file, the screen is automatically put into the output buffer before saving. In the RECORD mode, every key that you press gets converted to it's ANSI code equivalent and put into the output buffer. In the EDIT mode you can move about the screen and change it at will, the information will not be sent until you tell it to.

In order to use this editor well, you MUST be able to distinguish between the two buffers and use them to your advantage. Information in the output buffer should be considered already sent, information in the screen buffer should be considered information yet to be sent.

When loading files you will always be prompted whether you want the file read into the output buffer or the screen buffer. If the file has some sort of animated effect it MUST be loaded into the output buffer if you wish to retain the animation. This means that once an animation effect is saved, it cannot be edited.

An editing session may go as follows. First create an animation effect (using the RECORD mode). An example of this animation effect is a border that comes down from the top of the screen to surround a menu. Once the animation effect has been created, change to the EDIT mode and fill in the rest of the picture. When everything looks good, save the image to a file. This file will then have the ANSI codes needed to do both the animation and the rest of the screen.

It should be noted that both information going to the record buffer AND information going to the screen buffer BOTH get displayed on the screen. In most cases this will work fine, but it is possible to get strange results if you are not careful. For instance, suppose you load an animation effect into the output buffer. You then switch to EDIT mode, clear the screen and begin editing an image. When you save this image the screen will NOT (I repeat NOT!) be cleared after the animation effect! The clearing of the screen in EDIT mode clears your screen and clears your EDIT buffer, but does NOT send a clear screen code to the file. If you want to send a clear screen code to the file you must first turn on RECORD mode, then clear the screen. As you can see this could cause some confusion if you are not careful.

Ok, now that I have explained how the editor uses it's buffers, now let's see what editing features it has.

EDITOR FUNCTIONS

[ALT][G] - Set Graphics

This function allows you to set the various graphics parameters of the screen. This is the command used to change colors and attributes (underline etc.). This is the most commonly used function. When in RECORD mode the ANSI codes used for each function will be sent to the output buffer. In the EDIT mode the current editing attributes will be changed. When you press [ALT][G] you will get a menu with the following options.

[R] - Reset graphics
This option turns off ALL attributes and sets the colors to white on black.

[O] - Bold on
This option turns on bold face.

[U] - Underscore on
This option turns on underlining.

[L] - Blink on
This option turns on blinking.

[V] - Reverse Video on
This option turns on reverse video.

[I] - Invisible on
This option makes all characters invisible.

[F] - Set foreground color
This option allows you to set the foreground color. A menu of all colors and their numbers will be displayed.

[B] - Set background color
This option allows you to set the background color. A menu of all colors and their numbers will be displayed.

[D] - Done
This option returns you to the main editing screen.

[ALT][R] - **Record mode**

This function changes from the EDIT mode into the RECORD mode. The first thing that happens when this key is pressed is that an ANSI code to position the cursor is sent to the output buffer. Then it switches to RECORD mode. In RECORD mode every key you press is converted into it's ANSI code equivalent. This includes all color change, cursor movement, and character codes.

[ALT][E] - **Edit mode**

This function changes from the RECORD mode into the EDIT mode. In EDIT mode you can move around the screen wherever you please. No information will be sent to the output buffer until you either execute the 'save' function or the 'put screen into buffer' function. Note that what you are editing is the screen buffer. Not all information on the screen is in the screen buffer. Some information may still be on the screen that is in the output buffer, but is not in the screen buffer.

[ALT][C] - **Clear the screen**

This function clears the screen. If you are in the RECORD mode, the screen is cleared and a clear screen code is sent to the output buffer. If you are in the EDIT mode, the screen is cleared AND the screen buffer is cleared.

[ALT][N] - **Clear to end of line**

This function clears from the current cursor position to the end of the current line. If you are in the RECORD mode, the line is cleared and a clear line code is sent to the output buffer. If you are in the EDIT mode, the line is cleared AND the line in the screen buffer is cleared.

[ALT][S] - **Save cursor position**

This function stores the current cursor position for later retrieval. If you are in the RECORD mode, the position is saved and a save cursor code is sent to the output buffer. If you are in the EDIT mode the current cursor position is saved, but there is NO change in the status of the screen buffer. In the EDIT mode this function is used for convenience only.

[ALT][A] - **Restore cursor position**

This function restores the cursor position previously saved with the Save cursor position function (above). In the RECORD mode the position is restored and the restore cursor code is sent to the output buffer. In the EDIT mode the cursor is simply restored to its saved position.

[ALT][P] - **Put screen into buffer**

This function saves the information currently in the screen buffer to the output buffer. The information is scanned, starting from the top of the screen and moving from left to right down the page. If no information was typed into a position, that position will be skipped. Cursor positioning characters will be used to move the cursor to the proper locations on the screen. By using these cursor positioning characters information already on the screen will NOT be disturbed. This allows you to fill in spaces not already filled by some sort of animation effect without disturbing the characters made with the effect.

[ALT][L] - **Load from file**

This function will load information from an ANSI graphics file into either the screen buffer or the output buffer. The information will be displayed on the screen as it is read in. Note that the information read in does NOT have to be a file created with ANSI Editor, it can be any text file. If the file has ANSI codes in it, they will be converted and displayed properly on the screen. If the file has any animation effects, and you want to retain these effects, you must read the file into the output buffer. If you read a file with animation effects into the screen buffer, only the final result of the animation will be retained in the buffer.

[ALT][K] - **Save to file**

This function will save information from the editor to a file. Before the information is saved the screen buffer will be put into the output buffer, the output buffer will then be written to disk. You will be prompted to enter the file name to save to. The output from this function can be listed to any ANSI compatible terminal and the result will be what you created with the editor. This file can be used to make dynamic displays for your BBS system, or for your own use. This file can also be loaded into the editor later for further editing.

[ALT][Z] - **Zap buffers**

This function allows you to ZAP all or part of one of the buffers. You will be asked if you want to zap:

[S] The ENTIRE screen buffer
[R] The ENTIRE record buffer
[D] The character under the cursor in the screen buffer
[T] The character last typed in RECORD mode

NOTE: for the 'T' option, the last character typed MUST have been a regular ASCII character (a letter or symbol), not a special ANSI character sequence. (arrows, set graphics, save position, etc.). If you try to zap an ANSI sequence, you will get unpredictable results. BE CAREFUL!

[ALT][X] - Save and exit

This function first saves the current file being edited (see save option above) then exits back to the system.

[ALT][Q] - Quit without save

This option exits the editor and returns you to OS9. This option does NOT save any information.

THE DISPLAY

The ANSI-Editor screen has 2 sections, the screen display and the status line. The screen display shows what the final screen result should look like. Note that this display may be different from the final result (see the discussion on the use of buffers). The status line of the display shows some helpful information about the current position of the screen.

The first part of the status line shows the current X and Y coordinates of the cursor on the screen. This is useful when moving about the display and for centering etc.

The next part of the status line is the current mode of operation. This will say either EDITING or RECORDING according to whether you are in EDIT mode or RECORD mode.

The next thing on the command line is the current character. This is the current character IN THE SCREEN BUFFER. This is NOT neccessarily the current character on the screen. This allows you to see directly into the screen buffer.

Next is the current attributes. This gives a list of the attributes currently set. This is the attributes that will be used whenever you enter a character.

Finally, the current colors in use are given. This is the colors that are currently active. These colors will be used whenever you enter a character.

USAGE EXAMPLES

There are several different ways to use ANSI - Editor. It can be used to create animation effects or it can be used simply to generate screens.

The simplest case is when you simply want to generate a screen with no animation. The only problem with this is that when you create a screen there is no way to make sure that the person receiving the screen has a clear screen. To make sure that you clear his screen you must first RECORD a clear screen character. To do this you should first boot up ANSI - Editor, then put it in RECORD mode, then press [ALT][C] for clear screen. You can then put it in EDIT mode and create your screen. When you have your screen looking like you want it, simply save it to a file. That's all there is to it!

NOTE: The reason that the clear screen code is not automatically sent is so that you can combine several ANSI graphics files into one file. If a clear screen code were always inserted before your screen automatically you would not be able to do this.

A more complex graphic would include some animation effects. To generate these follow this example. Boot up ANSI - Editor. Put it in RECORD mode, then type in whatever animation effect you want (including a clear screen if desired!). If you want to reposition your cursor on the screen you can use the arrow keys, but this has a side effect. Wherever you move, the end result will have the same move. In other words, if you press right arrow 10 times, then down arrow 5 times, the final result will have 10 move right characters followed by five down line characters. This is OK if you want this effect, but if you simply want to move to another part of the screen (without seeing the cursor move) simply switch to EDIT mode, move the cursor to the new location, and switch to the RECORD mode again. You will now be in the new location IMMEDIATELY. Once you are finished making the animation effect, you can switch to the EDIT mode, fill in whatever needs to be filled in, then save. Now you will have the animation effect followed by the screen fill. Note that the screen will fill in AROUND the animation effect. It will not overwrite or rewrite the animation effect.

You may want to combine files in creating your graphics. To do this simply load in the files one after another. They will all be combined into one file when you save. You can read them into the output buffer or the screen buffer. This allows you to maintain animation effects as well be able to edit what the final output looks like.

I hope you enjoy using ANSI - Editor. Your friends and fellow BBS users will enjoy the dynamic screens that you create.

VIEWING THE BBS ON - LINE

Installing the Necessary Drivers

Special drivers are included with release 3.0 of OS9L2BBS to enable you to see what your BBS users are doing and to interact with them. These special drivers are on the "BBS Commands Disk" in a directory called MODULES. These modules MUST be included in your OS9Boot file if you want to see the board on line. To install the new drivers, follow these instructions:

Put a blank formatted diskette in drive 1
Put your system BOOT disk in drive 0
Type: chd /d0
Type: chx /d0/cmds
Type: OS9gen /d1

The disk will spin for a while then stop. When it stops...

type: /d0/OS9Boot

The disk will again spin and then stop. When it stops...

Take the BOOT disk out of drive 0.
Insert your "BBS Commands Disk" into drive 0.
type:/d0/modules/DDriver

The disk will again spin and then stop. When it stops...

type: <CTRL><BREAK>

The disk will again spin and then stop. When it stops...

type: dsave /d0 /d1 ! shell

You will see commands begin to be automatically typed and executed. This will copy all files from drive 0 to drive 1. It may take a LONG time, so go get something to drink and wait...

When the disk stops spinning, you have a new boot disk. Put the new disk in drive 0 and press the RESET button.

When the disk has booted up, the new drivers will be installed.
NOTE: This is your new system BOOT disk, keep it in a safe place and use it to boot whenever you want to run the BBS.

If you get a BOOT FAILED error with this new BOOT disk, your boot file may be too big. Try eliminating some of the modules you don't need (by CONFIGing a new disk).

If you would like to use CONFIG to generate a completely new disk, you can copy the needed modules from the MODULES directory on the "BBS Commands Disk" to the MODULES directory on the CONFIG disk. These modules are "DblDrv.dr", "Td.dd", and "Wb.dw". Each of these modules is explained on the next page.

The Double Driver

The double driver is a special device driver that drives two devices, a ACIAPAK and CC3IO. The driver then reads all of its input and sends all of its output to the devices called '/Td' and '/Wb'. These two device descriptors are included in the MODULES directory. Td becomes the new ACIAPAK descriptor (replaces /t2) and Wb is a special window that acts like any other window (/w1, /w2, etc.). The only difference is that DblDrv, the device driver, will try to access /Wb when it accesses /Td. If it cannot, it will return in error.

Note that ALL reads and writes to /Td are passed through /Wb and /Td, but Set and Get status calls are sent directly to ACIAPAK. What this means is that some things don't work in the window that do work on the port. One of these is Tsmon. Tsmon will NOT accept input from the window to establish baud rate. It will also NOT accept characters from the window while in chat or conference mode. This turns out to be perfectly reasonable in that someone in the window should not be allowed to interfere with the Tsmon, and should not need to interfere with the chat or conference modes.

The modules in the MODULES directory are described as follows:

DblDrv.dr	- The double device driver.
Td.dd	- The ACIAPAK device descriptor.
Wb.dw	- The CC3IO device window descriptor.
Ddriver	- All 3 modules merged into 1 file (for convenience).

If you wish to use /Td with another driver (other than ACIAPAK), this is ok. All that needs to be done is to patch DblDrv.dr where it says 'ACIAPAK' to the name of your driver. Everything else should be the same. I purposely left plenty of room in the DblDrv driver for longer names, so if the device driver you want to use is longer, it can still be patched it. If you don't know how to do the patch yourself, call us at the number given below and we can give you the patch over the phone.

Viewing The BBS

Viewing the BBS while someone is on-line is very simple, simply run the ViewBBS command, and a window will pop up displaying what the user is doing. Note that you would think that because the BBS runs in a window, you would be able to <CLEAR> to that window. But because of the way OS9 handles windows, this is NOT the case. The window will only be on the <CLEAR> key some of the time. This is a problem with OS9 and NOT with the BBS system. Using the ViewBBS command will always show you the window.

TROUBLESHOOTING

There are many problems that can arise in setting up a bulletin board system. Here are some tips to help when things don't seem to be working properly.

Problem:	People report that they press <ENTER> and nothing happens.
Solution:	Make sure that the DCD (Data Carrier Detect) dip switch on the modem is set to reflect the status of the phone line (not forced on).
Problem:	User gets the 'xxx baud 8 bits no parity' message, but then the computer seems to lock up.
Solution:	Make sure that the MOTD file is in the directory that Tsmon was run from. Make sure that the CMDS directory contains the LOGIN command. Make sure that the file BBS.users is in the directory that Tsmon was run from. Make sure that there is enough remaining memory.
Problem:	The user gets in, but at a certain menu they get logged off.
Solution:	Check the attributes of the menu files (make sure they have public read access).
Problem:	A certain menu option does not run (returns to menu).
Solution:	That menu option is returning an error, make sure it runs by running it 'stand alone'. Make sure there is enough free memory to run the program.
Problem:	People report that they cannot post messages. Everything seems okay, but the message doesn't get posted.
Solution:	Make sure that the file '/dd/bbs/bbs.alias' file exists and has the proper user's aliases in it (verify with the bbs.users file).
Problem:	When someone hangs up without logging off, they stay logged on when the next user calls.
Solution:	Make sure the carrier detect line on the modem is set to reflect the status of the phone line (not forced on).

I hope these tips will help you solve most problems. If you cannot solve problems with these tips, phone support is available at (601) 266-2773. We regret that we cannot pay for the phone call, but we feel that keeping the price of software down is important to you.