"Make ProBoard The Choice of Professionals Again!"

Version 2.30

++ November 2, 2023 ++

TABLE OF CONTENTS

INTRODUCTION

IMPORTANT INFORMATION - READ THIS FIRST

- ProBoard (to include all executables and documentation files) is copyrighted material of Jason Bock and http://proboardbbs.com.

- You can use ProBoard free of charge.

- We reserve the right to quit giving support or releasing updates of the software.

- The software and other materials included in the distribution archive are provided “AS IS” without warranty of any kind. We do not guarantee the correct functioning and/or reliability of the software. The authors, or any agent of the authors will not be liable for any direct or indirect damages resulting from the use of the software.

- You may not reverse-engineer ProBoard in any way and you may not add, change or delete any copyright information in the distribution file archive or on the ProBoard.com GitHub.

- You are free to distribute the original, unmodified ProBoard archive provided no fee is charged for its distribution. This excludes charges for online time on electronic bulletin boards or other communication services.

Copyright & Trademark Statement

The following names/products mentioned in this documentation are copyrighted material, trademarks, or registered trademarks:

Bimodem	Erik Labs
BNU	David Nugent/Unique Computing Pty Ltd.
Borland C++, Turbo C++	Borland International, Inc.
DESQview	Quarterdeck Office Systems
DoorWay	Marshall Dudley
FastEcho	Technik Burchhardt
FidoNet	Tom Jennings
GEcho	Gerard van der Land
GChat/GEdit	Chris Patterson
IBM PC/XT/AT	International Business Machines, Inc.
LANtastic	Artisoft Inc.
MBUTIL	Gerard van der Land
Microsoft C	Microsoft, Inc.
Microsoft Windows	Microsoft, Inc.
Multi-Edit	American Cybernetics, Inc.
Net2BBS	PC Micro (Mike Ehlert)
NetFoss	PC Micro (Mike Ehlert)
NetSerial	PC Micro (Mike Ehlert)
Opus	Wynn Wagner III
QEMM	Quarterdeck Office Systems
QuickBBS	The QuickBBS Group, Inc.
QuickEd	Tirosh Bros.
RIPscrip	TeleGrafix Communications, Inc.
RemoteAccess (RA)	Wantree Development and Andrew Milner
Squish	Scott J. Dudley
TheBank	Alain Schellinck
TLIB	Burton Systems Software
FrontDoor	Joaquim H. Homrighausen
Turbo Assembler/Debugger	Borland International, Inc.
X00	Ray Gwinn

Description

ProBoard is a computer program that allows you to run a BBS.

A BBS is a computerized Bulletin Board System, where files, messages and other useful items may be exchanged between users.

The operator of a BBS is called the System Operator, or SysOp.

The SysOp is responsible for setting up and maintaining the BBS, therefore, this manual is primarily geared towards the SysOp, or people who are interested in becoming SysOps.

ProBoard has everything you need to efficiently run a BBS, and more! It's state of the art technology that allows you to run a BBS without consuming large amounts of your hard disk space.

ProBoard provides the ability to run a multi-node BBS, allowing more than one user to be online at a time. It's small size and lightning fast speed make it ideal for running under a multi-tasker like DESQview, or on a network.

ProBoard also fully supports the ability to interface with mail networks such as FidoNet, UseNet, EchoNet, and mail processors such as SQUISH, FastEcho, GEcho, Fmail, and others.

The greatest asset of ProBoard is its ability to be enhanced by you the user, through the use of programs written in C/C++ using the provided ProBoard SDK (Software Development Kit). You can do virtually anything with your BBS using the SDK and a C/C++ compiler. Programs written with the ProBoard SDK are called PEX (ProBoard Executable) files. This built in “PEXability” assures you that there's nothing ProBoard can't do!

You do not have to be a C/C++ programmer to enjoy the benefits of ProBoard's SDK or these PEX files. There are many 3rd party PEX files available for ProBoard right now with more being released every day.

ProBoard has the ability to run most, if not all doors programs and other utilities written for other BBS systems such as PCBoard, QuickBBS (QBBS), SuperBBS, RemoteAccess (RA), etc.

When running an external program, ProBoard can swap itself to disk/EMS, and stay resident in approximately 2KB of memory!!

ProBoard is highly configurable because it allows you to use either the SQUISH, Fido Compatible *.MSG, JAM or the HUDSON message base formats for the contents of your message base. In addition to supporting all four of these formats for the message base, ProBoard allows you to configure your BBS to use all four at the same time. Name one other BBS that allows you this flexibility!

Not flexible enough yet? Consider then that ProBoard allows you to configure up to 10,000 message areas, and additionally 10,000 file areas. ProBoard also supports the use of CD-ROM drives. Flexibility!!!

Most, if not all, known utilities for RemoteAccess (RA) v1.11 as well as RemoteAccess v2.x, will work with this version of ProBoard.

ProBoard is remarkably easy to setup with the supplied PROCFG.EXE file. Most users are up and running within a few short painless (even enjoyable) hours.

We hope you will have as much fun running ProBoard as we did developing it.

Thank you for choosing ProBoard, “The Choice of Professionals”.

Features

The following is a list of the most important features found in ProBoard.

- ProBoard is THE fastest QuickBBS-style BBS program around!! It's completely written in highly optimized C++ and assembly language.
- Full multi-line/multi-user support. Up to 255 nodes can share the same user database, message base and file database. Each node can have its own modem configuration, welcome screens, menu structure, etc.
- Built-in fullscreen message editor.
- Support for 4 message base types: Squish, Hudson JAM and *.MSG, all at the same time.
- Support for up to 10,000 message areas and 10,000 file areas.
- Compatible with most, if not all, doors written for other BBS software such as RemoteAccess, QuickBBS, PCBoard, etc.
- A software development kit is included with ProBoard! You can write your own extensions to ProBoard using C or C++. Programs written with the SDK run “inside” ProBoard, for maximum flexibility and speed. The programmer doesn't have to worry about modem communications, user files, etc. THIS IS TOTALLY UNIQUE!! No other BBS software offers anything that even comes close.
- Integrated, lightning-fast duplicate file checking on uploads.
- Hooks for external upload checking programs.
- Extremely fast indexed file system, while still using the standard FILES.BBS-based file system.
- Full CD-ROM support. The CD-ROM drive is not accessed until a file is actually downloaded. It is only accessed to copy files to a local drive of your choice freeing up the CD-ROM for other requests.
- Full alias (handle) support.
- Flexible protocol configuration. You can install any protocol directly in ProBoard (even bi-directional protocols are supported). Protocols that can be installed in ProBoard include X/Y/Zmodem, MPT, Bimodem, HS/Link, etc.
- Local up/downloads
- Supports all types of modems. All known connect rates are directly supported, and you can specify up to 6 user-defined connect rates.
- Very flexible security system with over 65,000 security levels and 32 security flags as well as “reverse” flags.
- TTY, ANSI, Avatar (0 and 0+) terminal emulation.
- Swaps itself to disk or EMS when shelling to an external program (door), leaving only 2 (two) Kb resident!
- REAL-TIME multi-line chat built in. No stupid line-per-line chat.
- Direct support for all high speed modems (up to 115,200 bps).
- Extremely user-friendly configuration/maintenance utilities.
- Full RIP support, and unlike other BBS packages that leave the SysOp looking at cryptic

RIP codes, we even display the menus your non-RIP callers see, on your end so you can see what's happening on your BBS.

- Built in file tagging system.
- Users (and the SysOp) can send “personal files” to other users. ProBoard automatically deletes the files after they have been downloaded.
- Automatic validation of users using a sophisticated “template” system.
- Powerful User Editor including filter function, allows you to easily find and work with only the users you specify.
- Ability to specify an external message editor either as door or as a PEX file.
- Ability to specify an external chat program, either as a door or as a PEX file.
- Fully “data driven” system usage graphs, even in RIP mode.
- Ability to limit menu selections by age, baud rate, gender of user, time online, or time of day.
- Full Language support. Every prompt in the system can be changed (including color) and saved in a language file, as well as multi-language support for menus and textfiles. You can even replace any prompt with ANSI files or PEX files.
- Internal QWK support, provided as a PEX file for seamless integration into your BBS.
- User name can optionally be added to FILES.BBS when user uploads files.
- Powerful “Free Files” functions, great if you're running a support BBS, or a BBS to promote your company.
- Fully buffered I/O for optimum system performance.
- Fully definable paging hours for every day of the week.
- Support for OS/2 2.x time-slicing.
- “SysOp Next” function, with definable “alert” music, as well as semaphore file support.
- Ability to call menu editor while user is online (or in local mode) and edit the current menu. Great for testing, or in case you see something that just HAS to be fixed while a user is online.
- Built in file counters, shows which of your files are the most popular.
- Built in “flag description” editor, helps you remember just what all of those flags you have are for.
- Lots and lots more…

Changes

From what we know, existing v2.16 compatible PEX modules shouldn't be affected.

If something worked in v2.22B, but doesn't in v2.30, please let us know!

Below, you'll find changes to recent versions of ProBoard. Each change has a notation to the left of it which identifes the type of change made to the software. Those notations are as follows:

(-) Means a bug fix.
(*) Means a new feature.
(%) Means a change.
(D) Means a documentation change.
(I) Means information.

ProBoard v2.30 (Changes since ProBoard v2.22B)

(-) Fixed the dates in the JAM message bases

(-) Fixed the LastCaller and DOB dates in the current caller status screen.

(-) Updated the ENGLISH.PBL file to fix some issues displaying Y2K bugs

(*) Added an ANTIBOT.PEX call that loads ANTIBOT.PEX to force people upon first connection, to hit escape to verify that they are a human caller. The full source is included as well so you can modify it.

(*) Updated the _LC.PEX file to provide a more graphical LastCaller screen with full source included.

(%) Updated the Version screen with more information

(%) Updated the stock menus to have more features out of the box

(%) States ANSI detection upon connection

(*) Added a Python script to be called from a menu item to notify the ProBoard Discord channel who connected to your BBS

(*) Added a new menu Function 7: SHELL Special Code: *J which Displays the configured BBS name.

(*) Added a new menu Function 7: SHELL Special Code: *K which Displays the version of ProBoard.

(D) Migrated the documentation to www.proboardbbs.com/wiki

(-) Fixed other miscellaneous Y2K bugs that were left over

(%) Removed all forced registration notifications that were still present

ProBoard v2.21.00 & v2.22B (Changes since ProBoard v2.20.00)

(-) Fixed other miscellaneous Y2K bugs that were left over

(%) Version 2.21.00 was based on ProBoard v.2.20.0 which has now become open sourced,

thanks to John Riley who purchased it from Jeff Reader of Telegrafix and released it to the public. This is a minor update built by Mike Ehlert @ PCMicro to check if the code successfully compiles and executes.

(%) All the Telegrafix registration and time-bomb code has been removed.

(I) Testing revealed there are serious Y2K bugs in the source code that John was provided with. John was told that he will be receiving 3 CD backup discs from Jeff shortly containing other versions, but it seems unlikely that these discs will contain a more usable version.

(%) The Y2K bug(s) effect dates stored in the user record, messagebase and filebase. When rolling the date back to 2009 or earlier, these bugs no longer occur.

(%) The FOSSIL.ASM communication object has been updated by Mike to improve I/O performance in a virtual environment such as NTVDM, DOSBox, VMware etc. This is done by bypassing INT 14h calls and calling the FOSSIL directly.

ProBoard v2.20.00 (Changes since ProBoard v2.16)

(%) ProBoard now has a new registration scheme. It no longer uses the REGKEY.PB file to store your registration information. Now, your registration information is burned into the PROBOARD.EXE file itself. I know there are going to be a lot of people out there who won't like this change in the slightest, but it's really for new customers of ProBoard. With ProBoard v2.2, new customers don't automatically get an unlimited node license when they buy a copy. They purchase the number of nodes they want to run. The key is the serial number and the activation code which is used to unlock the software. This information tells ProBoard how many nodes it will allow. Now, for all those people who already own existing copies of ProBoard, you will automatically be upgraded to an unlimited node license as part of your upgrade.

How do I upgrade to v2.2, and get it registered? It's a four-step process:

Step 1: Download ProBoard v2.2 (which you've already done, or you wouldn't be seeing this file).

Step 2: Run the program included with ProBoard 2.2 called REGINFO.EXE. This will read your REGKEY.PB file, and extract out the old registration information. It will then display an eight-digit code on the screen which is your Upgrade-ID code. This code will be crucial in the next step.

Step 3: Visit the TeleGrafix Web site at http://www.telegrafix.com, and purchase your upgrade. While you're filling out the upgrade form, you'll be asked for your Upgrade-ID code, the name of the SysOp and the BBS name as they appear in ProCFG. The Web site will compare this information against the data encoded in your Upgrade-ID code, and if they match, you'll automatically be given your new registration information.

NOTE: A single BBS can only be upgraded once, to prevent pirates from stealing your registration information!

Step 4: Run the REGISTER.EXE program, and key-in the newly assigned serial number and activation code.

That's all there is to it. You're ready to go online with a fully functioning ProBoard system.

(%) New look and feel for the main console.

(-) Fixed a minor centering problem in the “last caller” field.

(*) Added a new option to the status bar (ALT-J) to “Jump to DOS”. This option was already available in the software, but it wasn't documented on the main console screen.

(*) Added five new functions to the ProBoard PEX SDK:

fileno()
getcurdir()
getdisk()
_dos_getftime()
_dos_setftime()

(%) Changed message 361 in the language files to make room for the expanded year string.

(-) Updated ProCFG.EXE to properly handle four-byte year data entry. The expiration date can be set to “00/00/0000” to indicate no expiration date.

(-) The new user signup now allows you to select from the updated date formats:

MM/DD/YYYY
YYYY/MM/DD
DD/MM/YYYY

(-) The new user signup now forces you to enter four-digit years when asked to enter your date of birth.

(-) The file system now forces you to enter four-digit years when entering a date to search for files by.

(-) Fixed a potentially serious bug with the birthdate asking code. There were circumstances when it might generate a garbled birthdate.

(*) Added a new macro code to “.A?? File Control Codes”. The new code is “^F;” (i.e., 06-59). This code displays the user's password, or if the “Hide Password” option is enabled in ProCFG, then this option will display the password hiding character as defined in ProCFG.

(-) The default SETUP.* menu files have been changed to use the new “^F;” macro code instead of “^FC” for the password display to properly support the Hide Password option.

(*) Changed the look and feel of the SysOp functions F1 and Shift-F1 to be more consistent with

the main console appearance.

(%) Changed the color of the status line displayed on the main console when a user is logged in. The new color is more like the main console.

(%) Changed the entire look and feel of ProCFG to be more similar to the BBS main console.

(-) Fixed a bug with ProCFG where if you make changes to the system configuration data, the inappropriate action was taken when you chose to save or cancel the changes when you exit. I don't know if this was present in the release version of v2.16, but in the code obtained from the author, the exact opposite operation was performed. If you decided to abandon the changes, they got saved instead, and vice-versa.

(-) Updated the PB_SDK header, and the library. Several of the data structures in PB_SDK.H have been updated to be accurate with regard to ProBoard v2.2 compatibility. The .LIB file has also been rebuilt to be v2.2 compliant.

ProBoard v2.16 (Changes since ProBoard v2.15)

NOTE: the new features in this release are not documented in PB216.DOC.

(-) When showing GIF file information, the more prompt is now displayed correctly.

(*) ProCFG allows you to insert, delete, move and sort message areas and file areas.

(*) Menu functions 49 and 54 (Select file and message area), now accept a new parameter to set the number of columns to display the areas in:

(-) Erasing of the “More Y/N/Tag” prompt should now work ok.

(-) ProBoard would say “No files received” twice after a null upload. This has been fixed.

(-) When editing tagged files, a 'C' would be shown when typing a 'U' to untag a single file. Fixed.

(-) The “inactive for xx seconds” prompt is now only shown once.

(-) Tabs in FILES.BBS are converted to spaces instead of waiting for <Enter>.

(*) The color of the letters typed when tagging files is no longer hardcoded to yellow.

ProBoard v2.15 (Changes since ProBoard v2.12)

(*) The biggest change in this version is the inclusion of an internal fullscreen editor. It's a stripped down version of TheEdit, and it will be distributed with ProBoard free of charge and fully functional. To enable it, you have to toggle “Use internal fullscreen editor” in ProCFG/Options/Paths.

*** (NOTE: if you're using a PEX named FSED.PEX (like TheQuote), the internal full-screen editor will NOT work! Delete FSED.PEX from your PEX directory to make it work)** *

(*) The default language can now be specified. This is the language that will be used to prompt the users for their name at login. It's also used when the user's language is not installed.

(*) A semaphore file for shutting down ProBoard is now supported. Just create a file called

DOWN.#, and ProBoard will exit with a specific errorlevel.

(-) When a user's screen length is set to 0, ProBoard no longer locks up when listing files.

(*) Default protocols have been implemented. There's a new menu function to change the default protocol. The user can select any protocol or “No default protocol”. 3 extra language prompts have been added for this feature.

(*) It is now possible to password-protect every menu item (see the menu editor)

(%) The default date for the new file check is no longer the date of the last call, but the date of the last new file check.

(*) 2 new menu functions have been added: “Change Message Area Group” and “Change File Area Group”. This allows users to select a new file/msg group without having to select a file/msg area.

(*) The functions “Keyword Search”, “Filename search” and “New file search” now accept a new parameter:

```
/FG    Only searches areas in the current file group
/FG=x  Only searches areas in the specified file group
```

(*) For each menu item, you can now specify what nodes are allowed to access it.

(%) When you only have one language installed, or when all languages share the same menus, ProCFG won't ask for a language in the menu editor.

(*) The background color for prompts can now be changed. It's a global setting in ProCFG/Options/Display.

ProBoard v2.12 (Changes since ProBoard v2.10)

(-) The installation utility will ask you if you want to overwrite your old language files or not. If you select “No”, any prompts that are missing in your old language files will be updated with the default English prompts that come with ProBoard.

(-) The local protocol is no longer selectable when RIP is enabled.

(-) The local protocol is no longer selectable in the QWK PEX.

(-) The language editor no longer shows funny characters when pressing <PgDn> at the last screen.

(*) When a user logs off, any files that were tagged and not downloaded will still be tagged at the next logon.

(%) The maximum number of AKAs has been increased to 50 (used to be 20)

(%) The RIP support has been reduced: the internal RIP prompts and screens are no longer used by ProBoard. There were too many problems with it. You can only have RIP MENUS now.

(-) Putting a Ctrl-L in a language prompt no longer messes up the screen.

(-) When user's alias and real name are the same, ProBoard would sometimes leave the “From”

field blank when posting a message. This has been fixed.

ProBoard v2.12 (Changes since ProBoard v2.01)

(*) Modem connect strings have been added for the following baud rates:

1200/75
21600
26400
31200
33600
36000
57600
64000
115200

(*) It is now possible to run a PEX or display an ANSI/ASCII file in any language prompt.

(*) You can now define up to 6 user-defined baud rates and connect strings.

(*) Separate modem configurations for each node are now possible.

(*) New option added to Menu Function 27 (Write a message) to send netmail to a specified address/person.

(%) The maximum # of security levels you can specify in ProBoard is now 50 instead of 25.

(*) Periods ('.') are now allowed as date separators, for example, 06.14.57 is now considered a valid date.

(-) On multi-node systems (running ProBoard standalone, without a mailer) sometimes the modem failed to initialize properly or would fall asleep while waiting for a call. This has been fixed by sending the initialization commands every 5 minutes.

(*) Uploads can now be scanned for viruses by specifying an external upload scanner. You should be able to integrate any available upload scanner into ProBoard.

(*) New Switch in PBUTIL UK. You can now specify a -L<level>, and PBUTIL UK will only work on users with the level you specify.

(-) The CONFIG.RA file created by CONVERT SIMUL2 was sometimes not recognized by external utilities because file area names in it were being truncated. This has been fixed.

(%) The QWK offline mail PEX will now update the users lastread pointers BEFORE it waits for the user to press <ENTER>.

(-) When compiling a nodelist (using PBUTIL NC), ProBoard would display a negative number when the “xxx nodes compiled” message was displayed. This has been fixed.

(%) The field lengths for the Message Areas have been increased. The maximum length for an “area name” and “echomail tag” is now 80 characters.

(%) All of ProBoard's message area configurations are now stored in the file “MESSAGES.PB” instead of, as in previous versions, the file “MSGAREAS.PB”. This means that any external

utilities or programs you have that use the file “MSGAREAS.PB” can not be used until they are updated to use the new “MESSAGES.PB” file structure.

(%) Any PEX files or doors that use the “MSGAREAS.PB” structure will NOT work correctly. They will have to be recompiled for this version to support the new “MESSAGES.PB” file structure!

(%) Any files contained in your file lists (FILES.BBS, etc.) that are larger than 1 megabyte, (for example, a 1.4 megabyte file), are now displayed as “1.4M” instead of “1400k” when the file list is shown to your users.

(%) When a “guest” user (GUEST flag set in their user record) logs in, and he/she forgot the password, ProBoard will no longer write a warning message.

(%) ProBoard will no longer ask new users to select a language if there's only one language file installed.

(*) There is a new option in PROCFG/Options/System Options: “RIP Graphics”. When set to “Disabled”, ProBoard will not use RIP, even if it was detected at login.

(*) ProBoard now allows you to change the colors of your the file listings and decide whether or not not you want to display the file counters to your users. (Refer to “PROCFG/Options/File List Format” for more details).

(*) ProBoard will now automatically enter the '/' for the users when they enter a date.

(*) While editing message/file areas in PROCFG, you can now jump to the next or previous message/file area by pressing the <PgUp> and <PgDn> keys.

(-) Lastread pointers are now correctly maintained for message areas greater than 1,000.

(%) The “CD-ROM” flag in each file area's configuration has been removed. ProBoard will now determine the format of your FILES.BBS itself (whether or not the date/time and size are listed). Refer to the PROCFG/FileAreas section of the 2.10 manual for more information.

(*) A new flag has been added for each file area: “Copy Local”. When enabled, ProBoard will copy the file to a local directory before starting a download. Refer to the PROCFG/FileAreas section of the 2.10 manual for more information.

(*) The time-consuming process of creating the file “TOPS.PRO” when the first caller of each new day logs in, has been moved to PBUTIL. As a result, there's a new option in PBUTIL: “DM” (Daily Maintenance). You should run this at least once a day (as part of your nightly maintenance).

(*) Menu Function 34 (View Archive) now supports RAR archived files.

(*) Handles (Aliases) are now supported in menu functions 13 (List Users), 50 (Show Users Online), 51 (Last Callers) and 53 (MultiLine Chat).

(%) Files listed in “FILES.BBS” can now start with a “!”. ProBoard will determine if a line in “FILES.BBS” starting with a “!” is a comment or a file name.

(-) PROCFG will no longer lock up the system if you press the <DEL> key when there are no entries shown in the Personal Files window.

(*) When running ProBoard in standalone mode (without a front-end mailer system), when you're at the “waiting for call” screen, you can now shell to DOS by pressing <ALT+J>.

(%) The .A?? file control code for the user's address line 2 has been changed from ^F% to ^F' (single quote).

(%) The .A?? file control code for the user's country has been changed from ^F@ to ^F“ (double quote).

(-) When you are replying to netmail from a point address, ProBoard now correctly addresses the message to that point address.

(%) The file MODEM.PRO has been renamed to MODEM.PB It is safe to delete your old MODEM.PRO file -AFTER- you have run the CONVERT PB routine (or the installation utility) as outlined in UPGRADE.TXT.

(-) The “In TOPFILES” flag is now saved correctly when editing file area configurations.

(-) The ”@<LASTDATE>@“ and ”@<LASTTIME>@“ text macros now work correctly.

(-) The “Hidden” flag in validation templates now correctly toggles the “Hidden” flag in the user record instead of the “Deleted” flag.

(-) Files that are not valid .GIF files are not shown when using Menu Function 55 (Show .GIF File Info).

(%) The “BEFOREPW.A??” and “AFTERPW.A??” files are now also displayed (if they exist) for BBS systems who are NOT using a system password.

(-) File counters for files that have been downloaded more than 99 times, are now displayed properly (alignment) when displaying file lists.

(-) The JAM lastread pointers in are now correctly updated when running PBUTIL MP (Message Pack).

(%) Users no longer can press <CTRL+G> to make the BBS beep when they enter messages using ProBoard's internal message editor.

(*) Personal files can now be sent from directories other than the personal files directory as specified in ProCFG.

(*) For every personal file, you can set whether ProBoard will delete the file after it has been downloaded or not.

(*) ProCFG now checks if directories you entered exist or not, and asks the user if he/she would like to create them. Paths are created multiple levels deep. For example, if you enter “C:\PB\RIP\ICONS”, and C:\PB\RIP does not exist, ProCFG will create “C:\PB\RIP” first, and then “C:\PB\RIP\ICONS” !

(*) Menu function 44 (Global boards selection) now accepts the parameter ”/M“ which allows the user to globally select/unselect areas for mail check.

(%) For each message area, you can now choose between the following for “Name Options”:

- Real Names Only
- Free Alias
- Fixed Alias (or real name)
- Fixed Alias (enforced)

(*) Local Up/Downloads are now supported. (by using the supplied PEX file LOC_PROT.PEX)

(%) When searching for files, the area names are now shown on one line, unless a match in found in an area.

(*) You can now replace the login procedure with a PEX. The old LOGIN.PEX is no longer supported.

Technical Information

ProBoard is entirely written in C++ and Assembler. No third-party libraries are used (except for the Squish MSGAPI), so we have TOTAL control over the code!

For the development of ProBoard, we used the following tools:

Compiler	Borland C++ v3.1
Linker	TLink, part of Borland C++ v3.1
Assembler	Turbo Assembler v3.1
Debugger	Turbo Debugger v3.1
Editor	Multi-Edit Professional 7.00

Credits

ProBoard is owned by: Jason Bock

ProBoard and all the included utilities have been / are updated by:

Modifications by:

- Mike Ehlert (v2.21.00)
- Lawrence Stockman and Rushfan (v2.22B)
- Jason Bock (v2.30+)

Other ProBoard help and guidance:

- Ozz Nixon
- George De Vries

ProBoard and all the included utilities were originally written by: Philippe Leybaert

Portions of ProCFG are also written by:

- Alain Schellinck
- Modifications by Jeff Reeder
- Modifications by Mike Ehlert
- Modifications by Lawrence Stockman and Rushfan
- Modifications by Ozz Nixon
- Modifications by Jason Bock

The ProBoard documentation is written by:

- Originally written by: Jim Biggs & Philippe Leybaert
- Reformatting and v2.2 additions by: Jeff Reeder
- ProBoard v2.22 Additions and Telnet Information: John Riley, Sr.
- The documentation has been cloned and updated to wiki format by Jason Bock

Note from Jim: This version of the documentation is dedicated to my late Father, Richard Merle Biggs. Dad, you will never ever be forgotten.

Note from John: The ProBoard version 2.22 and beyond document is dedicated to my two BBS Brothers, Robert King of Decatur, Illinois, and Eli Sanford of Gastonia, North Carolina.

First and foremost, Robert King. ProBoard SySop of both Planet Caravan and God and Country Node 2. Robert King, you introduced me to ProBoard. I fell in love with ProBoard BBS software Package so much, I just had to buy the complete and total rights and source code to it! I will forever remember you and the years (K-12) we went to school. I will remember our late nights hanging out and building computers. How I wish you could be here on the ProBoard Development Ninja Team. You would love it! You will always and forever be my brother. I love and miss you daily, Rob.

Secondly but surely not least, Eli Sanford. Eli Sanford, you have gone down in BBS History as one of the, if not the greatest WWiV BBS Software support of all time. You and I were brothers separated by different moms and dads. El, you and I caused hell in the thebbs.org IRC chatrooms, raised hell on the internet radio air waves. We caused trouble in your IRC Network, oddnet.net. We took knowledge and founded Osprey Networks. We once again raised hell over the internet radio air waves. We fought the good fight. No, matter what though, you supported me and my love for ProBoard in every way. Even if you did try to convince all the time to walk away from a ProBoard BBS and switch to WWiV. I love you, Eli. You will always be my brother.

From Jason: Thank you, John Riley for giving me this great opportunity, Dawn Bock and my children for understanding that this process takes a ton of time.

SUPPORT

Local support sites

Support for ProBoard can be obtained via the following locations:

Jason Bock SiliconUnderground

BBS: siliconu.com
Fidonet: 1:267/310
Email: jason@proboardbbs.com
Website: http://www.proboardbbs.com

John Riley, Sr. Slasher BBS

BBS: slasherbbs.com
Cell: (205) 570-5029
Email: john@proboardbbs.com
Website: http://support.proboardbbs.com

Support on the internet

There are many WWW and FTP sites available on the internet. Many of them are maintained by independent ProBoard gurus. Links to these sites can be found on http://www.proboardbbs.com, the official website of ProBoard BBS Software.

The “official” ProBoard web site is: http://www.proboardbbs.com

Bug Reports

Bug reports should be sent to: support@proboardbbs.com

If you find a bug, please try to describe every possibly relevant piece of information about the bug, including what data entry fields don't work right, what quirky situations bomb-out, or any other relevant information. Remember, if we can't duplicate the problem, it'll be all that much harder to fix!

Registration

You no longer **have** to “register” ProBoard BBS.

We are working on getting already registered ProBoard systems to display their license keys in the “Version” screen.

We are also working on getting a registration system in place so you can have a free registration key to display your information for nostalgia reasons. **AGAIN**, you **DO NOT** have to register ProBoard.

INSTALLATION / UPGRADING

Installation Overview

If you currently operate a RemoteAccess bulletin board system, Refer to the chapter “RemoteAccess to ProBoard Conversion” for specific instructions on how to convert your existing system.

*** Prudence and courtesy dictate that we ask you to back up your system before installing and/or upgrading ProBoard.** *

*** Although every effort has been made to provide you with software that will install/upgrade easily, due to the many different number of possible hardware and software combinations, it's virtually impossible to guarantee that a problem won't occur. It's your system, do what you like, but remember we did ask you to backup first.** *

Now, on to the good stuff!!

Unpack the distribution archive in a temporary directory or to a diskette and run INSTALL.EXE. You will be asked in which directory you want to install ProBoard. After confirmation, you will get an input screen where you can enter the different directories for ProBoard. After you have entered them, ProBoard will create all non-existing directories and start the installation. When completed, you will be asked to add a line to set the ProBoard environment variable in AUTOEXEC.BAT.

Now run PROCFG (the ProBoard configuration utility) to review and change all ProBoard settings (e.g. your BBS name, sysop name, etc.). Don't forget to set your modem parameters (explained in the documentation, chapter “Configuration”)

The ProCFG menu editor and user editor will not display correctly until you set the Paths and save.

Run ProCFG then:

- Go to Options –> Paths then exit to main
- Exit ProCFG and save changes.
- Go back into ProCFG and finish setting up your BBS.

At this stage, you can run ProBoard locally by running PROBOARD.EXE without any command line parameters.

To be able to accept modem callers, you have to install a FOSSIL driver if you don't have one installed already. X00 is a very good one to try. It can be found on any ProBoard support BBS. To install it, follow the documentation of X00. A command line that will almost always work is (assuming your modem is on COM1):

X00 B,0,38400 E

After you have done this, you will have to install an external protocol driver called CEXYZ which

can be downloaded from any support BBS. Place CEXYZ.EXE in your ProBoard system directory. DO NOT SET THE CEXYZLOG variable (or any other environment variable). In other words, don't follow the installation instructions of CEXYZ. Just copy the file CEXYZ.EXE to the ProBoard system directory. Nothing more, nothing less.

If you want to be able to use the fullscreen editor for entering messages on your BBS, you will have to download an external full- screen editor from a BBS and install it according to the installation instructions that come with it. A great full-screen editor PEX (ProBoard EXecutable) is TheEdit by Alain Schellinck.

Refer to the chapter “Starting ProBoard” in the manual for instructions on how to run ProBoard.

*** The following section is intented for people who don't want to use the automatic installation program for installing ProBoard.** *

We will assume that you created the directory C:\PB as ProBoard's main system directory.

1) Create the following directories:

C:\PB\MENUS
C:\PB\TXTFILES
C:\PB\PEX
C:\PB\MSGBASE
C:\PB\RIP
C:\PB\RIP\ICONS
C:\PB\SDK

2) Unpack ProBoard in the directory C:\PB.

3) Move all *.ANS, *.ASC and *.AVT files to the directory C:\PB\TXTFILES.

4) Move all *.PBM files to the directory C:\PB\MENUS.

5) Move all *.CPP, *.C, *.H, *.HPP, *.OBJ and *.LIB files from PB_230.ZIP in the directory C:\PB\SDK.

6) Move the files _GRAPH.PEX and _LC.PEX to C:\PB\PEX.

7) Move all *.RIP files to the directory C:\PB\RIP.

8) Move all *.ICN files to the directory C:\PB\RIP\ICONS.

9) Run ProCFG and enter the correct modem parameters for your modem (explained in the documentation, chapter “Configuration”)

10) Mark the file PROBOARD.EXE “read only” using the DOS ATTRIB command. This is important because PROBOARD.EXE is an overlayed executable and to run properly it needs this file attribute set to “read only”

To do this, change to the C:\PB directory and type the following command.

ATTRIB +R PROBOARD.EXE

11) Next, you have to install a FOSSIL driver if you don't have one installed already. X00 is a very

good one to try. It can be found on any ProBoard support BBS. To install it, follow the documentation of X00. A command line that will almost always work is (assuming your modem is on COM1):

X00 B,0,38400 E

After these 11 steps you can run ProBoard, but you will not be able to use the fullscreen editor or perform file transfers.

To log in locally, execute PROBOARD.EXE without any parameters.

To make ProBoard answer incoming calls, run PROBOARD -S.

12) Run ProCFG and create some file and message areas. (explained in the documentation, chapter “Configuration”)

13) Download an external full-screen editor from a BBS and install it according to the installation instructions that come with it. A great full-screen editor PEX (ProBoard EXecutable) is TheEdit by Alain Schellinck.

14) Download CEXYZ from a BBS and copy the file CEXYZ.EXE in the directory C:\PB. (CEXYZ can be found on any ProBoard Support BBS). DO NOT SET THE CEXYZLOG variable (or any other environment variable). In other words, don't follow the installation instructions of CEXYZ. Just copy the file CEXYZ.EXE to the ProBoard system directory. Nothing more, nothing less.

Now, you will be able to run ProBoard, use the fullscreen editor, and allow users to perform file transfers using X,Y & Zmodem.

Upgrade Overview

If you currently operate a RemoteAccess bulletin board system, Refer to the chapter “RemoteAccess to ProBoard Conversion” for specific instructions on how to convert your existing system.

*** Prudence and courtesy dictate that we ask you to back up your system before installing and/or upgrading ProBoard.** *

*** Although every effort has been made to provide you with software that will install/upgrade easily, due to the many different number of possible hardware and software combinations, it's virtually impossible to guarantee that a problem won't occur. It's your system, do what you like, but remember we did ask you to backup first.** *

Now, on to the good stuff!!

Unpack the distribution archive in a temporary directory or to a diskette and run INSTALL.EXE. The installation utility will detect if ProBoard is already installed and will ask you if you want to update to the new version. That's all there is to it. The update process will run automatically.

*** The following section is intended for people who don't want to use the automatic installation program for upgrading to ProBoard v2.30.** *

1) Remove the “read-only” attribute of your existing PROBOARD.EXE by issuing the following

DOS command:

ATTRIB -R PROBOARD.EXE

If your existing PROBOARD.EXE file was not marked “read-only” to start with, remember that this is necessary for ProBoard to run correctly since it's an “overlayed” .EXE file.

You should issue the above command even if you're not sure about the present attribute of your current PROBOARD.EXE file.

Important: If you do not remove the “read-only” attribute from your PROBOARD.EXE file, then you will not be able to copy the new PROBOARD.EXE file to the directory you are running ProBoard from, and the upgrade will fail. Read the above, and be sure you understand it before you proceed.

2) Copy all executables (*.EXE) from PB_230.ZIP over your old executables

3) Set the read-only attribute of PROBOARD.EXE

ATTRIB +R PROBOARD.EXE

4) Copy all *.PEX files from PB_230.ZIP in your PEX directory (overwriting the existing PEXes that came with the previous version)

5) Copy all *.CPP, *.C, *.H, *.HPP, *.OBJ and *.LIB files from PB_230.ZIP in your SDK directory

6) Copy all *.230, *.DOC and *.TXT files from PB_xx.ZIP in your documentation directory (or your system directory)

7) Copy all *.PBL files from PB_230.ZIP in your system directory (overwriting the existing language files that came with

the previous version)

8) Copy DEFLANG.PB to your ProBoard system directory.

9) Run “CONVERT PB” in your ProBoard system directory.

Windows 10 Information

Users setting up ProBoard with GameSrv or Net2BBS on Windows XP through Windows 8.1 can skip this information.

Windows 11 and other x64 bit Windows OS Information

When Microsoft released Windows 11, they decided not to make any more operating systems that support running 16-bit applications by using their NTVDM (NT Virtual DOS Machine) code.

If you want to run your BBS on Windows 11, you must do one of the following:

- Install a hypervisor and create a virtual machine on top of it
- Install DOSBox or DOSBox-X
    - DOSBox - https://www.dosbox.com/

    - DOSBox-X - https://dosbox-x.com/
    - This will only allow you to run one node per DOSBox/DOSBox-X console session
    - You could run DESQView in a console under DOSBox or DOSBox-X
    - You could install Windows 3.1, 95 or 98 under DOSBox or DOSBox-X
- Install NTVDMx64 - http://www.columbia.edu/~em36/ntvdmx64.html
- Install winevdm - http://www.columbia.edu/~em36/otvdm.html

Installing NTVDMx64

This method has been tested but is not supported by the ProBoard authors or test team.

Information: http://www.columbia.edu/~em36/ntvdmx64.html

Only 32-bit editions of Windows include the “NT Virtual DOS Machine” known as NTVDM, which allows DOS applications to run in emulation mode. Microsoft did not include NTVDM in their 64-bit editions of Windows.

NTVDMx64 is a patched version of Microsoft's NTVDM, for 64-bit Windows. It is based on leaked Windows 2000 source code, which was updated by the OpenNT Project and includes modern build tools. A programmer known as Leacher1337 has ported it to support 64-bit Windows by creating code-injection loaders that convert the 32-bit and 64-bit structures back and forth without altering the protected Windows system files.

NTVDMx64 an open-source project on Github: https://github.com/leecher1337/ntvdmx64

The full binaries and installer can be downloaded from the University of Columbia here: http://www.columbia.edu/~em36/ntvdmx64.html The current version is dated April 12, 2021

Before installing NTVDMx64 on Windows 10, it is important to disable Windows Defender “SmartScreen” and if you use Microsoft Edge to download it, you will also need to disable “SmartScreen for Microsoft Edge”. These can both be disabled by clicking on: * Settings> Update and Security> Windows Security> App & Browser Control

Antivirus software should also be disabled during the installation, and if your Computer's BIOS has an option called “Secure Boot” then this must also be disabled in order for NTVDMx64 to function. If “Secure Boot” is detected, the installer will open a Microsoft web page that explains how to disable it.

To install NTVDMx64, first extract the desired language folder from the .7-zip Archive into a folder on your computer such as c:\NTVDMX64 and right-click on the install.bat file and select “Run as Administrator”. The installer will ask if you wish to install support for 16-bit Windows applications (WOW32), which you can answer “No” to and it will just install support for DOS applications.

If you encounter issues installing NTVDMx64, you can open a support ticket on the Github page. A recent sysop encountered install issues and was assisted here: https://github.com/leecher1337/ntvdmx64/issues/101 Since then, this issue was resolved in the October 4, 2020 release of NTVDMx64.

You can optionally install the HAXM edition of NTVDMX64, which supports Intel Virtualization

Technology (VT-x) hardware acceleration so it is scientifically faster in textmode. HAXM requires a modern (2017 or later) Intel CPU with VT-x. To determine if your CPU supports Intel Virtualization Technology, visit https://ark.intel.com and select Processors > Find Processors by feature, or download the Intel Processor Identification Utility. Using HAXM, NTDVMx64 will run textmode applications nearly as fast as Microsoft's original NTVDM for 32-bit Windows editions.

Note that when using the HAXM edition of NTVDMx64, the WOW32 feature can not be used, WOW32 allows support for 16-bit Windows applications made for Windows 3.1 and earlier. If such support is desired, there is a much better solution is available here: https://github.com/otya128/winevdm

Otvdm/winevdm: run old Windows software in 64-bit Windows

This has NOT been tested by the authors or testers of ProBoard. This is only here for informational purposes.

Information: http://www.columbia.edu/~em36/otvdm.html

Enable Legacy Console Mode

- Open a command prompt, elevated command prompt, PowerShell, or elevated PowerShell window. Another way is you could also just directly right click on the console window shortcut or file, click/tap on Properties, and go to step 3 below.

Right click or press and hold on the title bar of the console window, and click/tap on Properties.
Click/tap on the Options tab, check the Use legacy console box for to turn on the Legacy Mode. This is actually needed to run any DOS BBS software or DOS Door software in Windows 10!
Click/Tap on OK to apply.
Close and reopen the console window to apply.

This information was given by Mike Ehlert of PCMicro.com

ArcaOS

ArcaOS is…

- …Arca Noae’s OS/2-based operating system
- …targeted specifically at booting, installing, and running on modern PC hardware
- …fully supported by Arca Noae’s team of engineers and developers
- …an upgrade path for customers currently using OS/2 or eComStation

ArcaOS is not just a repackaged, re-branded OS/2 Warp distribution. ArcaOS has undergone significant enhancements to address long-outstanding issues with prior OS/2 distributions, as well as to ensure its compatibility with the widest range of modern hardware available. Many portions of the OS have been completely rewritten to more fully leverage the power and performance of modern PCs. This isn’t your grandfather’s OS/2.

ArcaOS runs…

- …OS/2 applications (32-bit and 16-bit)
- …16-bit Windows applications
- …DOS applications
- …ported Linux applications
- …select 32-bit Windows applications
- …Java applications
- …Qt applications
- …REXX applications
- …much more…

Yes, everything that you were able to run under IBM's OS/2 installs and runs on ArcaOS.

Do you have a custom written OS/2 application? Favorite, classic OS/2 productivity, word processing, spreadsheet, or graphics applications? OS/2, Windows 3.1, or DOS games? ArcaOS runs all of those just like OS/2 Warp 4, only better. ArcaOS is more compatible with modern hardware, makes more efficient use of memory and system resources, and installs more easily than any other OS/2 distribution…ever. Really.

Do you have a system with 16GB of RAM in it? Want your apps to really fly? Configure ArcaOS to utilize all memory above 4GB as a RAM disk, and at bootup, copy your most frequently used applications there. It's like running your OS/2, Windows, DOS, REXX, Java, and ported Linux applications on air.

Visit our store to license your copy of ArcaOS today, view some screenshots, read some comments by others, or visit our wiki to learn more.

Have some quick questions? Check out our FAQ.

Want even more info? Create an account and subscribe to our blog for regular updates.

Installing SIO

These instructions have been provided by Sean Dennis!

Sean's Quick 'n Easy SIO/VModem Setup FAQ v0.0.6 Updated on 25 October 2023 Written by Sean Dennis (sysop@outpostbbs.net)

* NOTE: This FAQ assumes you know how to set up your BBS and OS/2 correctly. If you need help with that, please look elsewhere. This FAQ covers SIO v1.60d as I have never used SIO2K.

1) What is SIO/VMODEM?

SIO stands for Serial Input/Output. It is a serial communications driver for OS/2. VModem is a

virtualized modem engine also for OS/2. Note that these are NOT FOSSIL drivers, but replacements for OS/2's VCOM.SYS/COM.SYS and ArcaOS' PSCOM.SYS/VCOM.SYS. OS/2 doesn't need FOSSIL drivers but if you use DOS doors/BBS software, VXOO.SYS, SIO's virtual VDM FOSSIL driver, is required.

Basically, VMODEM fools the BBS into thinking that it's connecting to a standard serial port(s) and SIO handles the real communications end. I've used this with both OS/2 and DOS based BBS software and SIO/VMODEM work flawlessly.

2) Where can I download this? Is it registerable?

Look for SIO160D.ZIP. It's available from my BBS in file area 16, “Cheepware” (guest access available to download).

SIO is no longer registerable. Ray Gwinn, the author of SIO, has said publicly on Facebook to use the keygen that is available for 1.60d. No keygen is available for SIO2K.

3) OK, I got it on my system, now how do I install it?

Put the ZIP file into a temporary directory, unZIP it and then type INSTALL. SIO will then install itself into C:\SIO and run REREG.EXE (to check to see if you have a registration). Now, here's where things get fun! Get out your favorite text editor and load up CONFIG.SYS. Under OS/2 and eComStation, the lines referring to COM.SYS and VCOM.SYS have been commented out and the lines referring to SIO.SYS and VX00.SYS have been installed.

For ArcaOS, you must manually comment out the following lines in CONFIG.SYS:

```
DEVICE=C:\OS2\BOOT\PSCOM.SYS
DEVICE=C:\OS2\MDOS\VCOM.SYS
```

For all systems, make sure all of this is after the MOUSE.SYS line. If not, your mouse may quit working.

4) It's in the directory…how do I configure it for my system?

You'll need to define each COM port.

(Before you go any further, open the docs and read them. Yeah, they're dry, but there's a lot of good information in there! However, there is one little piece of info that you'll need…)

Basically, your SIO line should look like:

```
DEVICE=C:\SIO\SIO.SYS
```

You'll then want to add something like:

```
(COM1,3F8,IRQ4)
```

to the line… but that's not complete either. It will get you basic comm support, but you will not be able to run DOS doors over telnet! Why? SIO does not give up the COM port to any other program unless specified with a - in the command line. So, with that addition, the command line would look like:

```
DEVICE=C:\SIO\SIO.SYS (COM1,3F8,IRQ4,-)
```

If you want to lock your ports (I do and recommend it for the most consistency with data streams), you put it directly after the COM port like so:

```
DEVICE=C:\SIO\SIO.SYS (COM1:57600,3F8,IRQ4,-)
```

That's all there is to it for your regular dial-up modem. Now, if you're planning on using a virtual modem, things are a wee bit different. Since you're really not using a modem, you'll need to fool the computer into thinking there IS one on that COM port. You don't need the real IRQ and base addresses here, you can make them up… however, for this example, I'll use COM2's standard calling conventions here.

To use COM2 as a virtual modem, here's the line you could use:

```
DEVICE=C:\SIO\SIO.SYS (COM2,INTERNET:2F8,NONE:3)
```

To run DOS doors on the virtual modem:

```
DEVICE=C:\SIO\SIO.SYS (COM2,INTERNET:2F8,NONE:3,-)
```

If you want to lock the COM port AND run DOS doors:

```
DEVICE=C:\SIO\SIO.SYS (COM2:57600,INTERNET:2F8,NONE:3,-)
```

See a pattern emerging here? :)

This is my personal SIO command line in CONFIG.SYS (the first line is wrapped for this document):

```
DEVICE=C:\SIO\SIO.SYS (COM1:57600,3F8,IRQ4,-) (COM2:57600,
INTERNET:2F8,NONE:3,-) (COM3:57600,INTERNET:3E8,NONE:4,-),
 (COM4:57600,INTERNET:2E8,NONE:3,-)
DEVICE=C:\SIO\VSIO.SYS
DEVICE=C:\SIO\VX00.SYS
DEVICE=C:\OS2\MDOS\ANSI.SYS
```

If you are going to run DOS doors, make sure to include the VX00.SYS line as that is the DOS FOSSIL driver. VSIO.SYS is required in all cases. ANSI.SYS is also needed if you are going to use ANSI graphics.

Make sure that SIO's directory is in both your PATH and DPATH statements. Very important.

When you're done, save the file and reboot. You will see SIO's little “I'm alive” screen while booting up the system.

5) Great! Now, HOW do I set up VModem?

VModem is blessedly simple to use… just type in VMODEM at a VIO prompt. Vmodem will pop up and say at the bottom a message about being alive and well. The window will look somewhat like a modem with lights: when a particular port is being used, the black text will turn yellow when accessed.

If you're not online, VModem will still fire up. It doesn't care as long as those ports are defined in CONFIG.SYS.

You can make VMODEM load after the TCP/IP stack is loaded by creating a CMD file called TCPEXIT.CMD in C:\TCPIP\BIN (applies to all systems).

Something like this will work:

```
@echo off
c:
cd \sio
start "VModem" /win /min vmodem
```

You can also start your entire BBS via that batch file.

6) So, just what do I do to my BBS to have it access those virtual modems?

Set it up normally as if those modems were really there. However, the init string is ATZ. That's all that is needed. DO NOT USE +++ in the init string as it is not implemented in VModem. Please read the SIO documentation for more information.

That's all for a simple SIO/VModem setup. However, RTFM DOES APPLY. PLEASE READ THE MANUALS! There is mportant information in there that you need to know!

If you have any questions, I would be happy to try to help you.

As of this writing, you may contact me via:

BBS : telnet://bbs.outpostbbs.net:10323 or
https://outpostbbs.net/connect.html
Echomail: MIN_OS2 (Micronet), OS2 (Fidonet)
Netmail : 1:18/200@Fidonet and 618:618/1@Micronet
Email : sysop@outpostbbs.net
WWW : http(s)://outpostbbs.net

I greatly prefer to be contacted via my BBS or in echomail.

I hope this document helps you. This is gained from over twenty-five years of running SIO and VModem. My current BBS configuration is running ProBoard 2.17 under ArcaOS PE 5.0.8 with SIO/VMODEM v1.60d.

<eof>

Fix Issues with ProBoard Running On ArcaOS

Rick Smith found an issue with running ProBoard (Any Version) in ArcaOS 5.

The ProCFG menu editor and user editor would not display the names correctly or no data.

The fix is to run ProCFG then:

- Go to Options –> Paths then exit to main

Exit ProCFG and save changes.

SYSTEM CONVERSIONS

RemoteAccess 2.02 to ProBoard Conversion

If you are operating a RemoteAccess 2.02 BBS system, you can convert your existing userfile, message files, message areas, file areas and menus to the format used by ProBoard. This is done by running your RAFILE EXPORT in your RA system directory, and then running CONVERT.EXE in the ProBoard system directory.

But, before you do ANYTHING, we highly recommend that you back up at least your existing RA files, if not your entire system.

Here's the procedure to follow:
- Backup your existing files.
- Install ProBoard as explained in the installation instructions.
- Change to the RA system directory
- Run RAFILE EXPORT
  (Note: for areas where the FILES.BBS cannot be created in the file area's directory, like for CD-ROM areas, you have to run "RAFILE EXPORT" for each area separately. See your RA documentation for details)
- Change to the ProBoard system directory (usually C:\PB).
- Copy the following files from your RA message base directory to the ProBoard message base directory: USERS.BBS, USERSXI.BBS, LASTREAD.BBS, MSGINFO.BBS, MSGHDR.BBS, MSGTXT.BBS, MSGIDX.BBS, MSGTOIDX.BBS.
- Run CONVERT RA <ra-dir>
  eg. CONVERT RA C:\RA
- Run PBUTIL with the UF parameter.
  eg. PBUTIL UF
- Run PBUTIL with the UP parameter.
  eg. PBUTIL UP

Things you will have to do manually:
- Enter all system parameters in ProCFG (like paths).
- Enter the user levels & download limits in ProCFG
- Enter any events you have in PROCFG.
- Copy any .Q-A files to the ProBoard directory.

RemoteAccess Compatible Files

ProBoard uses different files than RA to store the file area and message area configurations. Therefore, to run any of your RA utilities that use the files CONFIG.RA, FILES.RA or MESSAGES.RA you must first run a conversion program to create compatible versions of these files. Which conversion you need to run depends on which version of RA the utility you want to run was written for.

Use the CONVERT utility (Supplied with PB_220.ZIP) to create the following files from your ProBoard system files.

FILES.RA
MESSAGES.RA
CONFIG.RA

If the utility you want to run was written for RA v1.1x then you need to run “CONVERT SIMUL1”. If the utility you want to run was written for RA v2.xx then you need to run “CONVERT SIMUL2”.

The CONVERT SIMUL<x> routine will place these files in the directory where you run CONVERT SIMUL<x> from. We strongly suggest running it from your ProBoard system directory (usually C:\PB).

You must run this conversion before you attempt to use your favorite RA utilities. Since it is possible for the wrong version of the files CONVERT SIMUL<x> creates to be present from an earlier conversion we suggest running CONVERT SIMUL<x> from a batch file each time you call a utility that needs these files. An example is as follows:

CONVERT SIMUL2 run RA 2.xx utility here…

later…

CONVERT SIMUL1 run RA 1.1x utility here…

You should run the correct “CONVERT SIMUL<x>” each time before you run one of your RA utilities.

Option: When running CONVERT SIMUL<x>, you can specify the “-U” parameter (CONVERT SIMUL<x> -U). This will allow CONVERT to create up to 10,000 file areas in FILES.RA instead of the first 200 only.

STARTING PROBOARD

Starting ProBoard

ProBoard looks for its data files in the directory where PROBOARD.EXE is located. You can, however, tell ProBoard to look for the data files in another directory by setting the environment variable PROBOARD (eg. SET PROBOARD=F:\PB). The environment variable DSZLOG used by most protocols is not needed. ProBoard will set this variable prior to starting the protocol.

You should start ProBoard with a batch file. ProBoard should ALWAYS be run with a batch file, because it returns to DOS when a user logs off.

Below is a basic batch file for stand-alone operation (without a mailer). It is called P.BAT and is included in the ProBoard installation archive.

```
:again
   PROBOARD -S
   if errorlevel 99 goto out
   if errorlevel 1 goto fatal
   goto again
:fatal
   echo A fatal error occurred.
   goto x
:out
   echo Normal exit.
:x
```

Note that errorlevels 2-4 generate a fatal error. These error- levels are used for mail networking, and ProBoard should not return one of these errorlevels in a standalone environment.

The PROBOARD.EXE command line options are discussed in the “Reference” chapter later in this manual.

IMPORTANT In order for PROBOARD.EXE to function properly and protect the file, it's DOS file attribute MUST be set to “read-only”. This is done using the DOS ATTRIB command.

Example:

```
ATTRIB +R  PROBOARD.EXE  (this marks PROBOARD.EXE as "read only")
```

IMPORTANT You will possibly need to setup errorlevels when exiting ProBoard for scanning of mail, running maintenance, updating bulleting, etc. If you have questions, please feel free to reach out on the support forums.

CONFIGURATION

Configuration Overview

ProBoard is completely configured by PROCFG.EXE. This program can be executed from any directory as long as PROCFG.EXE is in your path. In most of the menus, just press <Ins> to add an item, press <Del> to remove an item, and press <Enter> to select an item.

Options (F1)

This is the first option to choose when running PROCFG.EXE. A second menu will be displayed with the following items:

- Paths
- New Users
- Security
- Yelling
- System Options
- File Transfer
- Display Options
- Site Info
- QWK Options
- File List Format

A description of these fields follows.

Paths

Text Files

Directory where ProBoard's textfiles are stored (drive included!).

RIP Files

Directory where ProBoard's RIP files are stored (drive included!).

RIP Icons

Directory where ProBoard's RIP icons are stored (drive included!).

Menus

Directory where ProBoard's menus are stored (drive included!).

Message Base

Directory where the message base will be stored (drive included!).

Uploads

Directory where the users' uploads will be stored (drive included!).

Private Uploads

Directory where personal files are stored. (for file exchanges between users).

Nodelist Directory

Directory where the nodelist is located. For use by the nodelist compiler. Leave this field blank if you do not have a nodelist.

PEX Files

Directory where the PEX files will be stored.

Editor Command

The external editor's filename and path (eg. C:\GEDIT\GEDIT.EXE). You can also specify a PEX file here by using a "@" as the first character.

You can also use shell options here.

It appears that if you are using GEdit v2.00 or v2.01 that you will need to write an RA 1.1x style EXITINFO.BBS file to make GEdit work with ProBoard 2.16 and higher. This is done with the Menu Function 7 "*E" parameter. Refer to Menu Function 7 for more info.

Example: *SGEDIT\GEDIT.EXE -N*#*E
Example: @MYEDIT (for a PEX file).

If you are not using an external editor, leave this field blank and ProBoard will default to it's internal line editor (you can also use ProBoard's internal fullscreen editor by enabling the next option).

NOTE: this command will be ignored if ProBoard's internal fullscreen editor is enabled with the next option.

Use internal fullscreen editor

Set this to "Yes" to enable the built-in editor fullscreen editor instead of the external editor defined in "Editor Command". When the internal fullscreen editor is enabled, the external editor will be ignored.

External Chat

Path to your external chat program. If one is defined here it will be executed instead of ProBoard's internal chat routine. You can specify either a chat door (.EXE) or a chat PEX (.PEX) file here.

If you want to specify an .EXE file, enter the full path to the program. (see example below).

If you want to specify a PEX file, you must place a "@" character before the command line. (see example below).

You can also use the "Shell" (Menu Function 7) parameters (like *X etc.) or the text macros (@<…>@).

To use the built-in chat routine in ProBoard, leave this line blank.

Example: *E*SGCHAT.EXE D E (.EXE chat door)
Example: *@TheChat (.PEX chat door)

New Users

New User Level

The level a user will have upon his or her first login.

New User Flags

The flags a user will have upon his or her first login.

New User Loglevel

The loglevel a user will have upon his or her first login. This controls what information about the users session will be written to the system log (more info about this in the chapter titled USERS).

Allow ANSI

Allows New Users to select ANSI terminal emulation.

Allow AVATAR

Allow new users to select either AVATAR/0 or AVATAR/0+ terminal emulation.

Ask Voice Phone

Determines whether ProBoard should ask fora user's voice phone number at the time of their first login.

Ask Data Phone

Determines if ProBoard should ask for a users data phone number at the time of their first login.

Ask FAX Phone

Determines if ProBoard should ask for a users FAX phone number at the time of their first login.

Ask Birth Date

Determines if ProBoard should ask a new user for their birth date. If this is set to “Allow Blank” then users can decide whether they want to leave the birth date field empty or provide it for their user record.

Ask State

Determines whether ProBoard should ask the user for the State that they live in.

Ask Country

Determines if ProBoard should ask users for the Country they live in.

Ask Address

Determines if ProBoard will ask users for their address. If this is set to “Yes” ProBoard will

present the user with 3 lines to enter their address.

Ask Sex

If this is set to “Yes” ProBoard will ask the user if they are male or female.

Ask Date Format

If set to “Yes” ProBoard will allow the user to select the date format they prefer.

Choices are:

MM/DD/YY
YY/MM/DD
DD/MM/YY

Security

Allow Quick Login

If this is enabled, the SysOp can login locally by pressing [Enter] at the login prompt without entering a password. If you don't like this, just turn it off.

Write Pwd-Failure Messages

If a user fails to log in because he/she exceeded the maximum number of password retries, ProBoard can write a security message to that person and to the SysOp, telling him/her what happened. This can be turned on or off with this option.

Allow Login With Alias

Determines if ProBoard will allow users to login with their alias (handle). Users can also select or change their alias using Menu Function 59. If this option is set to “No”, ProBoard will not ask new users to select an alias.

Log Local Calls

If this option is enabled, all local logins will be logged in the file PROBOARD.LOG (ProBoard's system log).

Hide SysOp Activity

If enabled, ProBoard will hide all SysOp activity from the Last Callers list (Function 51), Show Users Online (Function 50), and from the display of the User List (Function 13).

Max. Password Retries

Maximum number of incorrect password attempts ProBoard will allow, before logging the user off.

Min. Password Length

The minimum length of a password. Should you change this option at a later time to less than the length of any existing users passwords, ProBoard will not check for Minimum Password Length when an existing user logs on. ProBoard will however enforce this setting again when

any existing users decide to change their password.

Security Message Area

The message area number where ProBoard's Security Manager should write it's security related messages.

Level For Crashmail

Level needed to send crash mail.

Flags For Crashmail

Flags needed to send crash mail.

Level For Fileattach

Level needed to do a file attach for Netmail.

Flags For Fileattach

Flags needed to do a file attach for Netmail.

Use System Password

(Yes/No). If set to “Yes”, ProBoard will require all incoming callers to know and enter a 'System Password' before asking them for their user name and user password. Great for running a 'Private' BBS in a corporation, etc. If set to “No” users will not be prompted for a 'System Password'.

System Password

If the above option 'Use System Password' is set to 'Yes', this is the password users are required to enter to gain access to the BBS.

Yelling

Max. SysOp Pages

Number of times a user can page the SysOp during one session.

PageBell Length

Number of seconds the SysOp paging bell will ring.

Paging Hours

Pressing <ENTER> on this option opens a window which allows you to specify during what hours a user can page you. Hours are selectable for each day of the week, in half hour increments.

Message Area #

When a user pages you and you don't respond, they will be asked if they want to leave you a message. If they choose to do so, the message area number you define here will be where their message to you is placed.

System Options

Check Mail At Login

Determines whether ProBoard should check for new mail and files upon login. If set to “Ask”, ProBoard will ask the user if they want to check for waiting mail and/or personal files.

Allow One-Word Names

Determines whether a user's name can be a single word.

Date Format

Determines if ProBoard should display all dates in European format (DD/MM/YY) or American format (MM/DD/YY).

Use File Sharing

Set this option to “Yes”, if you want to share the message base with FrontDoor or any other utility that uses the same locking scheme as RemoteAccess/FrontDoor

Swap to Disk

Determines the default shelling mode. If this option is set to ON, ProBoard will be swapped to disk/EMS, leaving only 2000 bytes resident!

Fast Mode

When this option is enabled, ProBoard will use about 10Kb more memory (depending on the number of file-areas), but the system will run faster.

Kill Netmail When Sent

Controls whether netmail will be killed after a message has been exported from the message base.

Confirm Validate

If this is set to “Yes”, ProBoard will prompt you to confirm any validation on a user record when applying a validation template. If set to “No”, no confirmation will be required once a validation template is selected to be applied to a user record.

Activity Log Size

ProBoard creates and maintains a file called BINLOG.PB in your ProBoard System Directory. This file contains information for generation of system graphs etc. Set the number of days worth of data the file BINLOG.PB should contain. Entering a 0 (zero) means the BINLOG.PB file will grow forever.

Log Local Activity

If set to “Yes”, ProBoard will add your local login activity to the BINLOG.PB file.

I/O Buffer Size

Specifies the amount (in bytes) that ProBoard will use for it's internal serial communication

buffering. Recommended buffer size is 32 for single line systems. If you are running a multi-node system under a multi-tasker such as DESQview, OS/2, or Windows, you probably will want to increase this value. To disable I/O buffering completely set this field to the value of 1.

Default language file

Name of the language file to be loaded when ProBoard starts. This language will also be loaded when a user logs in who's language could not be found.

Inactivity Limit

Number of seconds a user is allowed to remain idle. If the user hasn't typed anything when this limit is exceeded, they will be logged off.

Quote String

This string is used to when ProBoard quotes a message for a reply. A '@' character is replaced by the initials of the user who wrote the message quoted from.

Password Display

Enter the character that you want ProBoard to display to users, as they enter their password. The character you enter here will also be used in place of displaying the password if you have “Hide Password” (below) set to “Yes”. You may use any ASCII character (1-255) as a Password Display character.

Hide Password

If this option is set to yes, when you bring up user records in the User Editor, the users password will not be displayed. Instead, the character you entered in “Password Display” (above) will be shown. If this option is set to “No” then you will see each users password when using the User Editor.

Screen Blanking

Enter the number of seconds that should elapse before ProBoard's built-in screen saver should blank your screen.

Fuzzy Search Rate

Determines how extensive ProBoard's fuzzy search feature should work. The fuzzy search will help your users locate other users from your user file when posting messages and a misspelled name is encountered. It is also used in the User Editor when you perform searches. The higher the value you enter here, the closer a user record must match the search criteria before it is considered a find by the fuzzy search engine.

RIP Graphics

Determines if ProBoard will display RIP Graphics (both internal and user created) to the user or not. If set to “Disabled” (use the space bar to toggle the setting) RIP Graphics will not be displayed to the user even if RIP is detected when the user logs in.

File Transfers

Minimum Upload Space

Space needed on the upload drive for uploads to be allowed (in Kb).

Download Hours

Pressing <ENTER> on this option opens a window which allows you to specify during what hours a user can download files from your system. Hours are selectable for each day of the week, in half hour increments. The value of the “Ignore DL” flag in the user's record will override any hours you may have set here.

Check Duplicate Uploads

If this option is set to “Yes”, ProBoard's Menu Function 33 (Upload a File) will check upon completion of any upload(s) for duplicate files. ProBoard can not check for duplicates prior to any upload(s) due to any batch protocols the user may choose to use. Use this option along with the next two options (below) to decide how you want your system to deal with any duplicate files your users may upload.

Kill Duplicate Uploads

If set to “Yes”, when a user uploads a file that already exists on your system (in the file index) then ProBoard will delete the file from your upload directory (as specified in the data line of your upload menu entry or in PROCFG). If you want to save the duplicate files rather than have them deleted, set this option to “No”. When does ProBoard delete the file? Great question!!

ProBoard will delete any duplicate files immediately after checking for dupes, even before the user is prompted to enter a file description, so think twice before you set this option to “Yes”.

You can also create a text file to be displayed to your users in the event that they do upload duplicate files. The file that ProBoard will display must be named

DUPESFND.A?? and be placed in your "Textfiles" directory (usually C:\PB\TXTFILES).

Ignore File Extensions

If this option is set to “Yes”, then ProBoard's duplicate checking will consider NEWFILE.ZIP and NEWFILE.ARJ to be the same file. So if you have NEWFILE.ZIP on your hard drive or CD-ROM drive and a user uploads NEWFILE.ARJ, ProBoard will consider this to be a duplicate upload.

Be sure you understand this before setting this option to “Yes”.

Upload scanner

When uploads are received, ProBoard will run the command line specified here FOR EACH FILE. The macro @<FILE>@ will be replaced by the full name of the file to be scanned (including drive and directory). When the file is suspicious, the upload scanner should do one of 2 things:

Return with an errorlevel different from 0.
Create a semaphore file with a name specified in "Semaphore".

If the file failed the virus scan (so if a semaphore file was created or an errorlevel was returned),

ProBoard will perform an action on the file (defined in “action”)

None : No action (but upload is not credited to the user's account)
Move : The file is moved to a different area (specified in "bad file area")
Delete: The file is deleted

If you don't want to use the upload scanner, just leave the command line for the upload scanner blank.

Add Uploaders Name to FILES.BBS

If set to “Yes” the uploaders name will be added to the file name and description in FILES.BBS.

Display Options

Use 43/50 Line Mode

Enable local display of ProBoard in 43 or 50 line mode.

Show User Info Window

Should ProBoard display the User Info Window, when users are online.

Site Info

System Name

The name of your BBS. Will be written to EXITINFO.BBS. Be sure this matches any “keys” you might have for any door programs. This information is also used for your key file when you register, so be sure whatever you put here matches what you fill out on your registration form EXACTLY!

Location

The City where your BBS is located. This is used for the QWK packets created by your BBS so it's a good idea not to leave it blank when you set up your BBS.

Phone Number

The primary phone number of your BBS. Also used for QWK packets so be sure to not leave this field empty.

SysOp Name

The name of the SysOp. This also will be written to EXITINFO.BBS This information is also used for your key file when you register, so be sure whatever you put here matches what you fill out on your registration form EXACTLY!

Default Origin Line

Default origin line, used for Echomail if none is specified when setting up your echomail areas. (more about this later).

Number of Nodes

Used for multi-line systems. Enter the maximum number of users allowed to log in at the same time (max. 255)

QWK Options

QWK File Name

This is the name that QWK packets generated by ProBoard's QWK PEX will be given.

File List Format

It is here in PROCFG that you define how you want your file listings displayed to your users. Pressing <ENTER> on any of the following options (except for Hide/Show file counters which is an on/off toggle) will invoke ProBoard's color editing window, which will allow you to customize the colors of the items displayed in your file listings. As you begin to define the colors for your file listings, and whether or not to display file counters, ProBoard shows you in the bottom window how your file listings will appear to your users.

Hot Tip!

Remember, you can change the colors, and then if you decide you don't like the colors you have selected, you can return to the previous color configuration by selecting “Cancel Changes” when exiting from PROCFG. Just be aware that any other changes you made while in PROCFG will also be lost if you select “Cancel Changes”.

A brief explanation of each field follows:

File Tag

The letter to the left of the file name. This letter is what the user will press to tag the file for download.

File Name

The name of the file.

File Size

The size of the file.

File Date

The date of the file

File Counter

How many times the file has been downloaded from your system. (Refer to PBUTIL FC (File Counters) for more information on how the file counters are maintained).

File Description

The description of the file.

Extended Descriptions and Separators

Any lines of text in your file listings beginning with a space, are considered to be extended descriptions (or comments). This is not to be confused with extended file descriptions which begin with a “+” character, and may actually span several lines in your file listings.

Missing Files

A file name which is in your file listing but the file does not physically exist on the hard drive or CD-ROM where you told ProBoard to find it.

Hide/Show File Counter

Toggles the display of file counters on/off by pressing <ENTER> on this selection.

Protocol Configuration - (F2)

The second option in PROCFG.EXE is “Protocols”. You may want to skip this option when doing a first-time installation of ProBoard, as many protocol configurations are included in the file PROTOCOL.PRO (which can be found in the file EX_CFG.ZIP). You should select each protocol listed and set 'Enabled' to 'No' (you won't lose the protocol configuration) unless you have the actual protocol file itself physically on your hard drive. The protocols themselves are NOT included with ProBoard, but can be obtained from any ProBoard Support BBS, as well as many others.

Downloading and uploading files has always been one of the most important activities of BBS's. Most BBS programs have file transfer protocols pre-installed and do not allow additional protocols to be configured. ProBoard's philosophy is entirely different: no protocols are actually embedded in the code, all protocols are external and are being called by ProBoard.

As of yet, we don't know of any external protocol incompatible with ProBoard, Bimodem included! If you happen to find one that you think is an incompatible external file transfer protocol, please let us know!

Upon selection of the 'Protocols' option, a submenu is displayed containing the protocols already present. Add a protocol by pressing <Ins>, remove a protocol by pressing <Del>.

A protocol is entirely defined by the following parameters:

Protocol Name

Protocol's name, to be displayed in the down/upload menu. This can include a short description if you like.

Hotkey

Key to be pressed by the user to activate the protocol.

Batch

Determines whether the protocol can handle batch-mode (whether it can handle multiple files). Xmodem for example can handle only one file at a time, while Zmodem can handle multiple files.

Enabled

ProBoard comes with several pre-configured protocols for which you may not have the necessary files. It would be useless to have these protocols displayed in the menu. You can prevent this by setting Enable to 'No', without having to lose the configuration for this protocol.

Both-Way

Determines whether the protocol is a full-duplex protocol, i.e.. whether it can send and receive files at the same time. (eg. Bimodem)

Bimodem

The Bimodem protocol uses an odd format for its control file. Setting Bimodem to 'Yes' causes the control file to be written to disk in Bimodem-format.

Log File

Name of the log file created by the protocol. After the file transfer, the information needed to update the user- records will be obtained from this file by ProBoard. Most protocols write a file specified in the environment variable DSZLOG. ProBoard will set this variable to the right filename for you. ProBoard checks for the logfile in the directory where it was started from, so if some external protocol writes a different logfile than the one specified in the DSZLOG-variable, make sure it writes the file in the startup-directory!

Control File

Protocols that can handle batch-mode usually allow parameters to be passed not only on the command line, but also (should the command line grow too long) by means of a control file. ProBoard must know of this file, to be able to pass the filenames to the protocol.

Download Command

Command needed to start the protocol in download-mode. You may want to use the shell options of menu function 7 here. If the first character of this command is a '@', the named pex-file will be run. Note that no shell parameters (*x) are supported when calling a pex-file. You can use string macros though.

IMPORTANT: The command should be independent of the path it is called from.

Batch-mode protocols also require a control file to be specified. Should you, anywhere in this field, fill in a '#', then this character will at run-time be replaced by the filename of the file to be sent (only for non-batch protocols).

Upload Command

Command needed to start the protocol in upload-mode. Here also, the command should be independent of the directory it is called from and a '#' will be replaced by the filename of the file to be received (for non-batch protocols), or by the directory where files should be received into (for batch-protocols). A '@' as the first character will execute a pex-file (see DL Command).

Download String

Determines what should be written in the control file when downloading. A '#' character is replaced by the path and filename of the file to be sent to the user. Most often, a single '#' is the

only character in this field. This works for most of the protocols.

Example:

In case a user wants to download 3 files, entering 'Send #' in this field causes the following to be written in the control file:

```
Send C:\PB\FILES\COMM\TM.ZIP
Send C:\PB\FILES\COMM\TBILL.ZIP
Send C:\PB\FILES\UTIL\SHEZ55.ZIP
```

You can always take a look at the pre-configured protocols, to lighten things up for you.

Upload String

<reserved for future use>

Download Keyword

In order to allow ProBoard to update the user-records from the protocol's log file, a keyword must be specified to indicate a file has successfully been sent. If a protocol writes 'Sent <filename>' in the logfile, you should specify 'Sent' as the keyword. This keyword is CASE SENSITIVE!

Upload Keyword

Same as the previous field, for uploads.

File Word Number

This is the number of the sent file's filename, counting from the keyword, but NOT including the keyword. This is used for both uploads and downloads.

```
Eg. Sent 12/05/90 12334 PB_220.ZIP
```

In this case, you should enter '3' as the word number, because PB_220.ZIP is the third word counting from, but not including, the keyword ('Sent').

Efficiency

A percentage that gives the throughput efficiency for this protocol. This value is used to estimate the time needed to perform a file transfer.

Local Only

If set to “Yes”, this protocol will only be available when logging in locally.

CONCLUSION:

Correctly installing the protocols may seem somewhat difficult at first, but you will soon get used to it. And don't forget that the most commonly used protocols are already pre-configured in ProBoard!

The Local Protocol PEX

Included with ProBoard is a PEX which allows you to perform local file transfer (to/from your

own hard disk). This PEX is called LOC_PROT.PEX and should be located in your ProBoard PEX directory.

The protocol is already configured in the PROTOCOL.PRO file that comes with ProBoard. If you have upgraded from an earlier version of ProBoard, you can add the protocol manually.

This is how you should set it up:

```
================================================================
Protocol Name: Local Download Protocol
Hotkey       : L
Batch        : Yes                          Efficiency: 100  %
Enabled      : Yes
Both-way     : No                           Local only: Yes
Bimodem      : No
Log-file     : DSZLOG.TXT
Control-file : DSZCTL.TXT
Download cmd : @LOC_PROT D @@<STARTDIR>@DSZCTL.TXT
Upload cmd   : @LOC_PROT U #
Downl. string: #
Upload string:
Downl. keyw. : D
Upload keyw. : U
File word nr : 1
================================================================
```

Message Areas - (F3)

ProBoard is one of the only BBS packages that allows you to use more than one message base format to hold your messages. You can use Hudson, Squish, JAM, or Fido (.MSG) or a combination of these at the same time!

ProBoard's message base limits are as follows:

Message Type	Number of Message Areas	Maximum File Size
Hudson	200	16 Mb/15000 messages (all areas combined)
Squish	10,000	5400 msgs. per area.
Fido *.MSG	10,000	Unlimited (disk space available).
JAM	10,000	Unlimited (disk space available).

Each message area in ProBoard has its own name and properties.

For example, you could have a message area for public messages only, several message areas for Echomail, or an area where you create “Announcement” messages for your users.

Selecting “Message Areas” from the Main Menu, gives you a list of the available areas (which may initially be empty if you are setting up your BBS for the first time.)

To edit a Message Area, press <Enter>. To add a Message Area, press <Insert>, or enter a

number for the area you want to create.

For example, entering the number 10000 will create message area 10000. Keep in mind that if you are using the HUDSON format for any of your message areas, that any areas specified to use the HUDSON format will have to be specified as areas 1 - 200. This is a limit of the existing HUDSON message base format and is not a limit of ProBoard.

A message area has the following fields:

Name

Name of this message area.

Path

The path where this Message Area is located. This is used for SQUISH, JAM, and *.MSG, but NOT for Hudson.

Message Kind

The kind of message. You can have:

- Local (Local messages)
- EchoMail (Echomail)
- NetMail (Netmail)
- Pvt EchoMail (Private Echomail)

One word about the difference between the “Echo” type and the “Pvt Echo” type: In “Echo” areas, it is not allowed to delete messages that have been exported by an echomail processor (as specified by the FTSC, the FidoNet Technical Standards Committee). In a “Pvt Echo” area, this restriction is not imposed.

Message Type

One of the following:

- **Private only:** Only private messages allowed.
- **Pvt/Public:** Private or public messages allowed.
- **Public only:** Only public messages allowed.
- **To All:** This message type should be used in a LOCAL message area only.

The “To All” message type is intended for a SysOp to leave messages to all users. Any messages entered in this area will be considered “To All” regardless of the whom the “To” is addressed to. Messages entered in this area will be shown when ProBoard checks for waiting mail. You may not want to allow users to reply to these messages since their replies will be sent to all users. A better idea is to set the “Reply Area” to a different message area number if you decide to let users reply to your announcement messages.

Name Options

Determines which names can be used to write messages in this area. This can be:

- **Real Names Only:** Obvious I guess
- **Free Alias:** The user can choose any alias he likes, as long as it's not used by another user.
- **Fixed Alias (or real name):** The user can choose between his real name and his alias
- **Fixed Alias (enforced):** The user can only use his alias to write a message

It is recommended that areas where aliases are allowed, are made “Public Only”.

Message Base

The type of message base to use for this message area. Valid choices are Hudson, Squish, JAM, and Fido (*.MSG).

Use the <SpaceBar> to toggle between available choices. Remember, if you select Hudson that this message area's number (see top bar of the window) should be message area 200 or lower since Hudson only supports up to 200 message areas.

Read Level

Security Level needed to read messages in this area.

Read Flags

Flags needed to read messages in this area. You can now specify flags that a user MUST NOT have in order to READ from this message area. These flags are shown in reverse. To set a reverse flag, press the flag letter twice (it will appear reversed). To clear a reversed flag type the flag letter one more time.

Write Level

Security Level needed to write messages in this area.

Write Flags

Flags needed to write messages in this area. You can now specify flags that a user MUST NOT have in order to WRITE to this message area. These flags are shown in reverse. To set a reverse flag, press the flag letter twice (it will appear reversed). To clear a reversed flag type the flag letter one more time.

SysOp Level

Security Level needed to be allowed access to all functions in this message area.

SysOp Flags

Flags needed to be allowed access to all functions in this message area. You can now specify flags that the SysOp MUST NOT have in order to be allowed access to all functions in this this message area. These flags are shown in reverse. To set a reverse flag, press the flag letter twice (it will appear reversed). To clear a reversed flag type the flag letter one more time.

Origin Line

Only for Echomail: if you do not specify this, the default origin line will be used (refer to F1, Site Options, to specify your default origin line).

Use AKA

The network address for this area (for Echomail & Netmail only).

EchoMail Tag

This field is used by ProBoard to create a file called ECHOTOSS.LOG You should enter the Echomail Tag name for this message area (if this message area has the “Message Kind” field set to “EchoMail”). When a user enters a message into this area the tag name you specify here will be written to the file ECHOTOSS.LOG in the ProBoard system directory, causing your mail tosser to scan for outbound mail in this message area upon the callers logoff.

QWK Area Name

(Optional) You can enter the name of this area for QWK mail packers. QWK only supports up to 12 characters for area names.

Reply Area

The area where replies to messages in this area should be posted. Useful for allowing users to reply to “To-All” messages. Set this to “0” (zero) if you want replies to go to the current message area.

SysOp

The user name for an “Area SysOp”. A user with this name will have full SysOp access to this message area. When users write messages to “SysOp” in this area, the messages will be sent to the user name specified here.

Group #1

The first group number this message area belongs to (1-255) if any.

Group #2

The second group number this message area belongs to (1-255) if any.

Group #3

The third group number this message area belongs to (1-255) if any.

Group #4

The fourth group number this message area belongs to (1-255) if any.

All Groups

Does this message area belong to all message groups? Press the <SpaceBar> to toggle between Yes/No.

Kill after <xx> days

When running the PBUTIL message packer with the -D parameter, all messages that have been

in the message base for <xx> days will be deleted. This is for both HUDSON and JAM message base formats.

Kill rcvd after <xx> days

When running the PBUTIL message packer with the -D parameter, all messages that have been received for <xx> days will be deleted. This is only for HUDSON or JAM message base formats.

Max # messages

The maximum number of messages allowed in this area. When this number is exceeded, PBUTIL MP -D (message pack & delete) will delete the oldest messages from the message base (Hudson and Squish).

File Area Groups - F6

ProBoard allows you to configure up to 255 File Area Groups for your system. File groups can be used to place files of similar interest into a group users can select from.

If you run a BBS that caters to programmers, and also to people who enjoy gardening, you may want to create two separate file area groups so programmers aren't downloading files about pruning roses, and gardeners aren't downloading files about array elements

When you select “File Area Groups” a window is displayed showing any installed file groups (if any). To add a file area group press <Ins>. To delete a file area group, press <DEL>. To edit a file area group that already exists, press <Enter> with the highlight bar on the group you want to edit.

When you press <Ins> to add a new file group, a dialog will be displayed asking for the group number. Press <ENTER> to accept the next available area, or type a number (between 1 and 255) for the file area group you want to create.

The fields that make up a file group are as follows:

Area Name

The area name that will be displayed to users when selecting from your file area groups.

Access Level

The minimum security level needed for a user to be able to see/select this file area group.

Access Flags

The access flags needed (or not needed) to access this file area group.

Once a file area group has been defined, refer to Menu Function 54 (Select New File Area) for the options needed to allow your users access to it.

File Areas - (F5)

File areas are used to setup and categorize your downloadable files. You can prevent groups of users from accessing certain file areas with the use of Access Levels and/or Access Flags. ProBoard uses an advanced file index to locate files as defined here in your file areas. Once you have configured your file areas, you MUST run PBUTIL FI (file index) to create the index in order for ProBoard to find the files for users to download.

A file area has the following fields:

Area Name

Name of this file area. Displayed to users on the BBS on the file selection list.

File Location

Directory where the files for this area are physically located. It is also possible to specify multiple directories per file area. To do this, enter the first file directory here, then create a file in your ProBoard system directory (usually C:\PB) called FA_<area>.CTL. Place each additional file directory (one directory per line) in this file to tell ProBoard where to find the additional directories for this file area.

Example: for file area #10 (Games), which has a total of 4 directories, place the first directory in the “File Location” slot in PROCFG, then create a file called FA_10.CTL. In this file, place the additional directories one per line like this:

```
D:\DLOAD\GAMES2
D:\DLOAD\GAMES3
D:\DLOAD\GAMES4
```

Use an ascii editor like QEDIT, or the DOS editor to create this file.

Listing File

Full path & filename of the file in which the downloadable files are described. This files is often called FILES.BBS. Refer to Menu Function 31, in the chapter on Menus.

Sample: C:\PB\DLOAD\FILES.BBS)

Access Flags

Flags needed to download files in this area. You can now specify flags that a user MUST NOT have to access this file area. These flags are displayed in reverse. To set a reverse flag - type the flag letter twice until you see it display in reverse. To clear a reverse flag type the flag letter again.

Access Level

Level needed to download files in this area.

Copy Local

Setting this option to “Yes” (use the <SPACEBAR> to toggle), tells ProBoard whether or not to use it's CD-ROM specific file listing formats. In addition, it tells ProBoard to copy the files in this

file area to a different location, usually a local drive before a download of the file begins. This is ideal for CD-ROM or Network drives since it frees up these resources for other users to access them.

It's important to understand that ProBoard now determines the format of “FILES.BBS” itself. This will assist you in using CD-ROM's on your BBS that have different “FILES.BBS” formats.

The “FILES.BBS” can have any of the following formats:

<filename> <description>
<filename> <size> <date> <description>
<filename> <date> <size> <description>
<filename> <size> <description>
<filename> <date> <description>

Any missing information (<date> or <size>) will be retrieved from the CD-ROM or Network drive by ProBoard. If only the <date> is specified in your “FILES.BBS”, ProBoard will need to retrieve the <size> of the file from the disk in order to display your file listing for this area to your users. If both the <date> and <size> are specified, (the preferred format for your “FILES.BBS”) ProBoard will not need to check the CD-ROM or Network drive for this information.

Hot Tip!

Use “FILES.BBS” listings that include <date> -and- <size> whenever possible, to prevent ProBoard from accessing your CD-ROM or Network drives. This will give your BBS optimum performance when users choose to list your files.

ProBoard will also copy files from CD-ROM or Network drives to a local drive when the “Copy Local” flag is set to “Yes”. Files will be copied to a directory called “CD_TEMP”, which will be created by ProBoard, off of the directory from which ProBoard was started from. You can specify your own directory by creating a DOS environment variable called CDTEMP containing the name of the drive -and- directory.

How to create your own “Copy Local” directory:

- Decide on and create the directory where you want the files from the CD-ROM or Network to be copied into. For example, C:\MYDIR\CDFILES
- Add the following line to your “AUTOEXEC.BAT” file:

SET CDTEMP=C:\MYDIR\CDFILES

If ProBoard can not find the directory as specified it will attempt to create it, and if it can't will revert to using the CD_TEMP directory off of the directory the ProBoard was started from.

In TOPFILES

Determines whether or not this file area is included when TOPFILES.A?? is created. Refer to PBUTIL FC for more information.

Free Area

Specifies whether or not ALL files in this file area are FREE files. If this is set to 'Yes', any files a user downloads from this area will not be deducted from their download limits.

Group #1

The first group number this file area belongs to (1-255) if any.

Group #2

The second group number this file area belongs to (1-255) if any.

Group #3

The third group number this file area belongs to (1-255) if any.

Group #4

The fourth group number this file area belongs to (1-255) if any.

All Groups

Does this file area belong to all file groups? Press the <SpaceBar> to toggle between Yes/No.

Max. files

Maximum number of files that can be downloaded from this area per user per day (0 means unlimited).

Max. Kb

Maximum number of Kbytes that can be downloaded from this area per user per day (0 means unlimited).

FILES.BBS Date Format

The date format that is used in this area's FILES.BBS file. You have to change this if FILES.BBS contains dates in a different format than “MM/DD/YY” of “MM-DD-YY” (CD-ROMs from Europe for example). If you don't set this so it matches the date format in FILES.BBS, ProBoard will not be able to correctly read the dates listed in your file list.

File Area Groups - F6

ProBoard allows you to configure up to 255 File Area Groups for your system. File groups can be used to place files of similar interest into a group users can select from.

If you run a BBS that caters to programmers, and also to people who enjoy gardening, you may want to create two separate file area groups so programmers aren't downloading files about pruning roses, and gardeners aren't downloading files about array elements

When you select “File Area Groups” a window is displayed showing any installed file groups (if any). To add a file area group press <Ins>. To delete a file area group, press <DEL>. To edit a file area group that already exists, press <Enter> with the highlight bar on the group you want to edit.

When you press <Ins> to add a new file group, a dialog will be displayed asking for the group

number. Press <ENTER> to accept the next available area, or type a number (between 1 and 255) for the file area group you want to create.

The fields that make up a file group are as follows:

Area Name

The area name that will be displayed to users when selecting from your file area groups.

Access Level

The minimum security level needed for a user to be able to see/select this file area group.

Access Flags

The access flags needed (or not needed) to access this file area group.

Once a file area group has been defined, refer to Menu Function 54 (Select New File Area) for the options needed to allow your users access to it.

Time/Download Limits - (F7)

In ProBoard you can grant different groups of users different rights concerning download limits and maximum online time per day. Additionally, you can limit downloading in a very powerful and flexible way. This is done by defining user levels.

Editing user levels is done in by pressing <Ins> to add a security level, <Del> to delete a security level, and <Enter> to edit an existing security level.

These are the fields to be specified for each level:

Security Level

The user level you are editing.

Time Per Day

Time a user with this level can spend on your system each day.

Kb Download Per Day

Daily download-limit associated with this user level (in Kbytes/day).

Download Delay

Time a user must spend online each call before a download can be made. (Great to calm down excessive downloaders).

Usergroup ID

String of max. 5 characters that identifies this level, eg. NEW, REG, VIP, etc. This is optional. These ID's will be shown when the userlist is displayed (Refer to Menu Function 13).

Free Download

The amount that can be downloaded by users with this level, without having to upload or write messages.

Upload Needed

The percentage of total downloads the user has to upload. Eg. if the upload factor is 15%, and a user has downloaded 1000Kb, he will have to upload 150Kb. Setting this to 0 allows the users to download as much as they want, until the download limit (see below) is reached. Of course, it is impossible to download more than the daily maximum each day.

Free Download/Msg

The amount of Kilobytes that can be downloaded free for each message written. This rewards busy message-writers by increasing their download-limit. The amount of kilobytes is added to the free download number (see above)

Maximum Download

When this limit is set to a positive non-zero value, and a user reaches this limit, his/her level will be changed to the "Fall To" level. Be careful when changing this to a value other than 0 (zero). Doing so may lock out a user if the "Fall To Level" (below) is set to zero. Your user may not be too happy with your BBS should this happen.

Fall To Level

Security Level which user should be assigned when they reach the (above) "Maximum Download" setting.

User Editor (F8)

Working with your users records using the User Editor is easy. You can use the user editor to locate/view/change information in a users record. You can also edit a users record while they are on-line by pressing <Alt-E>.

Both ways of editing a users record in ProBoard are virtually identical. ProBoard keeps a user's record in memory while they are online and then writes it back to the user file once they log off. If you want to make changes to a user's record who is online, it is important to use the <Alt-E> method since you will be editing their record in memory. If you use the user editor in PROCFG.EXE to edit the record of a user who is online, any changes you make will be written over by the user record in memory when they log off.

You can use the following keys in the user editor:

<PgUp>

Go to the previous user-record.

<PgDn>

Go to the next user-record.

<Ctrl-PgUp>

Go to the first user-record.

<Ctrl-PgDn>

Go to the last user-record.

<ESC>

Exits the User Editor.

<Alt-A>

Add a new user record.

<Alt-D>

Toggle the 'deleted' flag of the current user.

<Alt-F>

Shows flag descriptions for flags you have defined on your BBS. After pressing <Alt-F> another window will appear showing the flags and any descriptions you have defined for them. You can add/edit descriptions for your flags by pressing <Alt-E> in this window, then using the Up and Down arrows to position on the flag. Then simply type the description you want and press <ESC> to exit that field, and <ESC> once again to exit the flag description window.

<Alt-L>

Opens a secondary window listing your user records. Several powerful functions are available here to make finding and editing your users much easier. Press the <F1> key, and ProBoard will display a help screen containing information on the following items.

<PgUp> ..... Moves to the previous screen page of user records.
<PgDn> ..... Moves to the next screen page of user records.
<Home> ..... Moves to the first record in the user file.
<End> ..... Moves to the last record in the user file.
<Esc> ..... Closes the user list window and returns to the user editor, positioned on the first record in the user file.
<Enter> ..... Closes the user list window and returns to the user editor, positioned on the user record you were on in the user list window.
<1...9> ..... Begin typing the record number you want to go to in the user file. A window will appear showing you the number as you type it. Press <ENTER> to jump to the record you have entered, or <ESC> to return to the user list.
<A...Z> ..... Begin typing a name or part of a name to search for in the user file. This activates the search dialog window. Once you have entered the name you want to

search for, press <ENTER> to begin the search. You can press <ESC> at any point during the search, to abort the search and return to the user list. If the search finds a record matching what you entered, you can press <ENTER> to edit that record, or press <Alt-N> to find the next occurrence of your search criteria.

<Alt-F> ..... Activates a window asking if you want to enable the user list filter. If you select "No", you are returned to the user list. If you select "Yes", the user list filter dialog window is activated. By filling in various items in the filter dialog, you can limit the scope of the user records displayed in the user list. Once a filter is in effect, only the user records matching the filter condition are displayed in the user list. You can then move thru the records displayed, editing them and using any other user list keys as you normally would. To disable a filter, press <Alt-F> and select "No" when asked "Enable filter?". Closing the user editor also cancels any filter in effect.

<Alt-G> ..... Activates a window asking what user record you want to go to in the user list. Either enter a number and press <ENTER> to go to that user record, or press <ESC> to return to the user list.

<Alt-S> ..... Activates the search dialog window. Once you have entered the name you want to search for press <ENTER> to begin the search. You can press <ESC> at any point during the search, to abort the search and return to the user list. If the search finds a record matching what you entered, you can press <ENTER> to edit that record or press <Alt-N> to find the next occurrence of your search criteria.

<Alt-N> ..... Finds the next occurrence (if any) of search criteria specified with the <Alt-S> (search) option (above).

<Alt-N>

Searches for the next user-record matching the search criteria specified with <Alt-S>.

<Alt-P>

Toggles the Password Hide/UnHide option when viewing user records. The character that is displayed when passwords are hidden is defined in PROCFG, [F1] Options, System Options, in the “Password Display” field. When viewing a user record where the password is hidden, simply press <ENTER> when positioned on the hidden password, and ProBoard will display the user's password in a window.

<ALT-R>

Restore User Record. If you are editing a user's record and change your mind, you can use this option to restore the user record to it's original values.

<Alt-S>

Searches for a user-record. You may optionally specify either part of, or the full name of the user you are searching for. The search uses “fuzzy” logic, meaning that it will find user names containing strings that are close to what you type. How accurate the fuzzy search is depends on the setting in PROCFG, [F1] Options, System Options, in the “FuzzySearch Rate” field.

<ALT-V>

Validate User. Opens a window containing any validation templates defined in the PROCFG, [Shift-F6] Validate Template Editor. (Refer to the “Configuration” section of this manual under “Validate Template Editor” for more information on setting up your templates.) Select one of the templates, and ProBoard will update the user record with the values defined in the template, asking you for confirmation first. Press <ESC> at any point during the validation to abort the validation.

<F10>

Opens a secondary window containing additional information about the current user. Once this window is displayed, you may display/edit the following items.

- Times Called.
- Total Number of Downloads.
- Total KB of Downloads.
- Total Number of Uploads.
- Total KB of Uploads.
- Number of Messages User has posted.
- Total Time user has spent online to date.
- Amount of Time user has spent online today. If the user has not been online today, there may still be a value in this field. This field is cleared by ProBoard upon a users first call of each day.
- Total KB downloaded today.

- Time Balance in the users bank account.
- Time Withdrawn from users bank account.
- Time Deposited in users bank account.
- Time user has borrowed from TheBank.
- Payback date for time user borrowed from TheBank.
- Kbyte Balance user has banked.
- Kbytes user has withdrawn from TheBank.
- Kbytes user has deposited.
- Kbytes user has borrowed.
- Payback date for Kbytes borrowed.
- Date last used the time bank.

When you have found the user-record you are looking for, move between the fields by using the arrow keys. To edit a field, just move the selector to that field and begin typing.

A user record has the following fields:

User Name

User's name.

Alias

Each user can have a unique 'alias' (nickname). In selected message areas, the user can use this alias to write messages or (if enabled) can login using this alias.

Password

User's password.

City

City the user lives in.

State

State the user lives in.

Country

Country the user lives in.

Address 1

Users address, line 1 of 3

Address 2

Users address, line 2 of 3

Address 3

Users address, line 3 of 3

Forward To

(not implemented yet)

Level

User's security level (ranging from 0 to 64000). A 0 (zero) means that the user has NO access to the BBS.

Flags

User's flags. Edit them by pressing <Enter>, and then pressing any character ranging from A thru Z and 1 thru 6 to toggle that flag. Press the <F1> key to view/edit any flag definitions for your BBS.

Expiration Date

When this date is specified (non-zero), the user's security level will drop to the level specified in the “Expiration Level” field (below).

Expiration Level

The new security level assigned to a user when the “Expiration Date” (if other than 0) is reached.

Expiration Flags On

User flags to turn on when expiration date is reached.

Expiration Flags Off

User flags to turn off when expiration date is reached.

Comment

This field allows you to enter comments about this particular user.

Voice Phone

User's voice phone number (not restricted to a certain layout).

Data Phone

User's data number (if any).

FAX Phone

User's FAX number (if any).

Birth Date

The date of birth of this user. This will be recorded if you turn on the option to ask new users for their birth date upon initial login. Directly to the right of the birth date field is a number in brackets. This number is the users age, as calculated by ProBoard based on the date entered into the birth date field. If you change the users birth date, you must save the record in order for the calculation of the users age to take place.

Sex

User's sex. Toggle options with the space bar.

Date Format

The date format this user prefers to use. Toggle through the allowable formats by pressing the right-arrow and left-arrow while positioned on this field.

Screen Height

Number of screen lines for this user.

Default Protocol

The default file transfer protocol this user has selected.

Language

The language file the user selected from the list of available languages on your BBS. If no languages are available or the user pressed <ENTER> when asked to select a language then ENGLISH is used as the default. You can edit the users language selection but you must type the language name.

Log Level

The way in which entries about this users calls will be written to PROBOARD.LOG Use the Space Bar to toggle between “Normal”, “Suspicious”,“Extensive”, and “Friend”.

Terminal

Users Terminal Setting. Press the <Spacebar> to toggle between “ANSI”, “Avatar”, “Avt/0+”, and “TTY”.

NetMail Credit

Number of credits the user has left to write Netmail messages.

First Call

The date of the user's first call to your system. This field is display only.

Last Call

The date of this users last call to your system. This field is display only.

Deleted

If this is set to “Yes”, this users record will be removed from your user file the next time you run PBUTIL UP (User Packer) to pack the user file.

More Prompt

Pause after each screen page?

Clear Screen

Send screen clearing codes to the user?

Hot Keys

Hot Keys enabled? If set to “Yes” user can hot key thru your menus. If set to “No” then command-stacking will be used instead.

IBM

If set to “Yes” all IBM-specific characters will be filtered out and converted to standard ASCII. If set to “No” then IBM characters will be sent.

Full Screen Editor

If set to “Yes”, then the fullscreen message editor (defined in PROCFG, F1, Paths, Editor Command) will be used when this user enters messages. If set to “No” then ProBoard will use it's internal line editor instead.

NoKill

If set to “Yes” the user's record CAN NOT be removed from the userfile when using PBUTIL UK (User Killer) or PBUTIL UP (User Packer).

Ignore Download

Does this user have UNLIMITED download access? If set to “Yes”, ProBoard will ignore download limits for this user.

Attention

If set to “Yes”, ProBoard will play the file “ATTEN.MUS” when this user logs on. The file “ATTEN.MUS” must be located in your ProBoard System Directory (usually C:\PB).

No Tops

If set to “Yes”, then this user will be excluded from any of the “tops” lists that ProBoard creates. (Refer to Menu Function 48 for more info).

Hidden

If enabled, ProBoard will hide this users activity from the “Last Callers List”

(Menu Function 51), "Show Users Online"
(Menu Function 50), and from the "User List (Menu Function 13).

Guest

Allows this user to have “guest account” status. This means their daily totals for minutes online, files downloaded, etc., will be reset on each call of the day, and not just on their first call of each day.

Free Chat

If set to “Yes”, ProBoard will freeze the system timer when you and this user are chatting. This means no time will be deducted from a users online time when chatting.

Local Only

If enabled, this user is not allowed to log in remotely, they can only use the local console to login.

RIP

If set to “Yes” then RIP graphics will be sent to this user. If set to “No” RIP graphics will not be sent to the user even if RIP is detected.

Menu Editor (F9)

Creating menus for ProBoard is extremely easy using the built-in Menu Editor.

When you select the “Menu Editor” option in PROCFG's main menu, a new window is opened containing the defined languages. ProBoard allows you to have a language specific set of menus for each language you define. Refer to the “Language Editor” portion of the manual for more information on specifying paths for your language specific menus. If no path is given, the default menu path defined in PROCFG, (F1) Options, Paths, Menus is used.

Once you have selected the language whose menus you want to edit, a second window is opened displaying the menu names that are available to edit. A menu must already exist to be displayed in this window.

You can type the first few letters of a menu name to jump to that menu in the choice list, or use the cursor keys to position on the menu that you want to edit. Then press <ENTER> to invoke the menu editor.

In this window, you can use the following keys:

- Up/Down - Scroll up/down through existing menus.
- Enter - Select an existing menu to edit.
- Insert - Add a new menu.

Once you have selected a menu, you will notice the name of the menu you are editing is displayed on the top line of the Menu Editor and a line-by-line list of the items on the selected menu is displayed.

To add a menu item, press <Ins>, to remove a menu item, press <Del>. Your menus can contain up to 255 menu items for each menu.

To edit a menu item, move the selector to that item and press <Enter>. You can change the menu prompt and highlight colors, or specify a RIP menu to be displayed to your users by pressing <Alt-P>. If you want to see how a menu will look to your users (ASCII/ANSI), press <Alt-S>. You can also copy and paste menu items. To copy an item, move to the item you want to copy, and press <Alt-C>. To insert the copied item somewhere else, move to the place where you want the menu item, and press <Ctrl-P>. You can even copy and paste items across different menus.

Hot Tip!

You can press <Alt-M> while logged on locally and a menu is displayed, or when a user is logged on and sitting on a menu. This launches the menu editor with the current menu loaded so you can edit it. You will need to tell ProBoard which language you want the menu loaded for. This

makes it very easy to correct/add items to a menu you are viewing on the screen, while it is still fresh in your mind.

Basically, each line of your menus that your users will see contains the following attributes.

Text Line

Text line to be displayed to the user.

Color

Color of this menu item. Press <ENTER> to edit the color for this menu item.

HotKey

The key the user is to press to activate this menu option.

Function

Menu Function to be executed. Press <ENTER> to display and select from a list of all ProBoard Menu Functions.

Data

Data associated with this menu item. Refer to the “Menu Functions” section of this chapter for more info on Data Items that you can use or that may be needed with some Menu Functions.

Minimum Level

Minimum security level a user needs to access this menu item.

Maximum Level

Maximum security level a user can have to access this menu item.

Flags

The flags that the user needs in their user record to access this menu item. Press <Enter> when on this field to invoke the menu flag editor. Simply press the character (A-Z or 1-6) to toggle the flags. While you are in the menu flag editor you can press <F1> to view/modify notes about what your flags are used for. You can also specify flags that a user MUST NOT have in order to access this menu item. These flags are shown in reverse. To set a reverse flag, press the flag letter twice (it will appear reversed). To clear a reversed flag type the flag letter one more time.

Minimum Age

The minimum age a user has to be in order to access this menu item. Set this to “0” to disable age checking.

Maximum Age

The oldest age that a user can be to access this menu option.

Sex

The sex the user must be to access this menu function. Set this to “Don't Care” to allow users of either sex to use this menu option.

Time Left

The minimum amount of time a user must have in order to use this menu function. Great for doors that don't check a users remaining time, etc.

Time Online

The amount of time a user must be online before they are allowed to select this menu option.

Time Frame

The hours during which users can select this menu item. Default is “fully enabled” meaning that this menu item is available at all times during a 24 hour period. Press <Enter> to bring froward a window that will allow you to select times for this menu item to be accessed. Times are in 1/2 hour increments, based on a 24 hour format.

Minimum Speed

Minimum baud rate at which user must be connected in order to see this menu item.

Maximum Speed

Maximum baud rate at which this menu item will be displayed. Great for suggesting users with 300 baud modems buy a faster modem.

Nodes

Determines on what nodes this menu item will be available. Press [Enter] to change.

Password required

If enabled, a password is required to execute this menu item (see next item)

Password

Password required to executed this menu item (if enabled). The password is not case sensitive.

RIP Options:

Show Remote

If set to “Yes” then menu item or A?? file will be sent to remote. The default value for this setting is “No”.

Show Local

If set to “Yes”, when the RIP sequence is sent to the user, the SysOp will see this as a normal ANSI menu-line or ANSI file. The default value for this setting is “Yes”

Reset Screen

If set to “Yes”, ProBoard will reset (clear) the RIP windows before executing this menu function.

Matrix Addresses (F10)

This is where you enter your net address. If you have more than one net address, enter them here also.

Modem Parameters - (Shift-F1)

When selecting “Modem Parameters”, you get a choice between the default configuration or a node-specific configuration. The default configuration is used whenever there is no node-specific configuration defined.

You do not need to change these parameters when using ProBoard with a front-end mailer, since it is in the mailer's setup where you would define your modem(s) parameters.

Hardware Setup

COM port

Enter the port that your modem is configured to use. For example, enter a 1 here to specify COM1, etc.

Locked Speed

Set to “Yes” if your modem's bps rate is fixed (see below)

Max. Bps Rate

Maximum baud rate your modem can handle. If you are locking your COM port with the fossil driver, this setting is ignored.

Modem Delay

Number of 1/10 seconds to pause between each character that is sent to the modem (some modems can't handle fast input).

Answer Delay

Number of 1/10 seconds to wait before the answer command (see below) is sent to the modem after a ring is detected (only when “Manual Answer” is enabled)

Command Strings

You can insert special codes in the modem command strings:

^	Set DTR high.
` or v	Set DTR low.
|	Sends a <CR> to the modem.
$	Sends a break to the modem.
~	Pauses for 1/2 second.

Make sure you add a '|' to send a return for all modem commands. The modem result messages also need a '|' if the modem sends a CR/LF after the message. (i.e. “ATA|” or “RING|”)

Init Command 1-3

Modem initialization commmands. These are strings that have to be sent to the modem to initialize it and to have it ready to answer the phone. It is possible to have up to 3 init commands. The second one won't be sent until the first one results in “OK”, and the third one won't be sent until the second one returns “OK”. Every 5 minutes, the modem will be re-

initialized (some modems fall asleep after a while)

Ok Message

What the modem sends back when a command was accepted (“OK”)

Off Hook Command

Command to be sent to the modem when the SysOp is logging in locally.

Down Command

Command to be sent to the modem when the SysOp presses <Esc>

Manual Answer

Set to “Yes” if ProBoard should send the answer command when a ring is received. Do NOT set your modem in auto-answer mode when using this option.

Ring Message

What the modem sends when the phone rings.

Answer Command

What ProBoard should send to answer the phone.

External/Fax Msg

ProBoard will exit immediately when this message is received from the modem. This can be used to receive faxes when running ProBoard in a standalone environment (for some modems this string should be set to “CONNECT FAX”). ProBoard will then IMMEDIATELY exit with the given errorlevel.

External Errorlevel

The errorlevel ProBoard will exit with when the above message is received from the modem.

Connect Strings

<xxx> Bps Call

String returned by the modem upon an <xxx> bps call. This is a 'partial' string. So if the modem sends 'CONNECT 2400/ARQ', the string 'CONNECT 2400' will match. A '|' can be used to specify a CR/LF. It HAS to be used for the 300 bps connect string, because “CONNECT” without a '|' would match “CONNECT 2400” or “CONNECT 9600”. So the correct string is: “CONNECT|”.

You can define up to 6 user-defined connect strings. For each one you can specify the bps rate and the connect message that goes with it.

LOCKING BAUD RATES

Note: If you use an error-correcting modem (MNP/V42), you may have to lock your serial port speed using the fossil driver.

A “locked” baud rate refers to the transfer rate between the computer and modem. When locked, the baud rate will remain constant regardless of what the application program, such as

ProBoard, requests the baud rate to be. The modem MUST support a constant computer to modem speed, otherwise any baud rate change requested by an application will be ignored, resulting in an incorrect setting between your system and your users. The result will be garbled input and output.

Most high speed modems do support a fixed baud rate, and by locking the baud rate you will obtain higher throughput. A fossil program like X00 or BNU will handle this for you and is HIGHLY RECOMMENDED (see the respective fossil doc file for command syntax).

Note: Experience shows that a faster locking baud rate than actual phone line baud rate will yield the best results.

Examples: X00 B,0,9600
BNU /L0=9600

See your FOSSIL documentation for details. If possible, install your fossil for “quiet” or “no commercial” mode so the screen display from the fossil does not bleed onto the ProBoard main screen (stand-alone systems with no mailers in particular).

SysOp Macros (Shift-F2)

There are 2 kinds of SysOp macros:

* **Key macros:** With this type of macro, it is possible to assign many keystrokes to a single key. When the macro key is pressed, all keys specified will be passed to ProBoard, as if you typed them yourself.

Special chars: '|' is replaced by <Enter>
'^' is replaced by <Esc>

* **Shell macros:** You can link any DOS command to a macro's hotkey. A shell-definition MUST start with a '@'. The string following the '@' should contain the DOS command to be executed. You can, of course, use the special shell options from menu function 7.

Take this for an example:

Suppose you have set 'Swapping' to 'No' in ProCFG. Should you, however, need ALL your system memory in the shell, you could define the following macro:

@*C*N*Q*X (COMMAND.COM, NoLog, NoMsg, Swapping).

* **String macros:** You can display a line of text to the user with a hotkey. A string macro should start with a back quote (`) character. When the hotkey is pressed, the line of text will be sent to the user and displayed on your local screen.

Events - (Shift-F3)

It is possible to instruct ProBoard to perform some action at a set time and day (an “Event”). This action can be: exiting with a specific errorlevel (for a batch file) or executing a DOS command. This is useful for example, to pack the user file, or to pack the message base at a specified time each day.

You can define up to 30 events.

An event is described by the following fields:

Enabled

If this is set to 'No', the event will be ignored.

Active Days

Determines on what days of the week this event will run. To edit this field, press <Enter> and toggle the 'Yes/No' fields displayed next to each day.

Event Time

The time at which the event has to run on the selected days (in 24h format).

Duration

How long this event has to stay active. During the event, no users are allowed to log in on ANY node.

Event type

'Command' : A DOS-command will be executed when the event is activated. You can use any of the shell options from menu type 7. Remember to use the *Z option to run a batch file.

'Errlevel': When the event is activated, ProBoard will exit with an errorlevel, specified in the next field.

Errorlevel

The errorlevel to use for this event if the event type is set to 'Errorlevel'.

DOS-command

The DOS-command to execute for this event if the event type is set to 'Command'.

Node number

An event will run on ONE node. You can specify the node here.

When using a frontend-mailer, you only have to specify the time, days and duration, because the events have to be executed by the mailer.

Personal Files - (Shift-F4)

ProBoard allows you and other users to send “Personal Files” to each other on the BBS. ProBoard keeps track of each uploaded file. You can add/edit/delete Personal Files with this option in ProCFG. In the list of files, a [+] means that the file physically exists. This assures you that a file you define as a Personal File for another user actually is in the directory waiting for them upon their next call to your BBS. When the file name is shown between parentheses, it means that the file is not located in the personal files directory (as setup in ProCFG), but in another directory.

To edit an entry, press <Enter>,

To Delete an entry, press <Del>,
To add an entry, press <Ins>.

Filename

Name of the uploaded file. Press <Alt-L> to select any file which is in your personal files directory. You can also enter a complete path and file name if the file you want to send is not located in the personal files directory. Keep in mind that when the “Delete” flag is set (see below), the file will be physically removed from your hard drive after the user has downloaded the file.

From User

The user who sent this file. Press <Alt-L> to select a name from your user file, or type the name in yourself.

To user

The destination user for this file. Press <Alt-L> to select a user from your user file, or type one in.

Delete

If this flag is set to “Yes”, the file will be physically deleted after the user has downloaded it. If set to “No”, only the entry in the personal files list will be removed, not the file itself.

It is possible to add files to this list. This way you can send a file to a specific user. ProBoard will handle deleting files after a user transfers “personal files” that are waiting for them (if the “Delete” flag is set). You can also manually delete files in the “personal file” editor by positioning the highlight bar on the filename and pressing the <DEL> key. You will be asked to confirm the deletion of the file before it is deleted from the “personal file” list as well as from your hard drive.

For more information on how to implement this option on your BBS, refer to Menu Function(s) 22, 33, 32 explained in detail in the section on menus.

Language Editor - (Shift-F5)

ProBoard allows you to change the text and the color for every prompt in the system to create a custom “look and feel” for your BBS. ProBoard also allows you to display your system prompts to users in several different languages.

Language files end with the file extension “.PBL” and are stored in your ProBoard system directory (usually C:\PB). You can create and install up to 30 different language files at any one time.

To create a different language file, first you must select one to work with as a model. You then copy the model to a language file with a new name. Refer to the following example:

COPY ENGLISH.PBL STARTREK.PBL

This would create a new language file called “STARTREK.PBL” from the “ENGLISH.PBL” file

(supplied with ProBoard). All of the system prompts in “STARTREK.PBL” would at this point be exactly the same as the ones in ENGLISH.PBL.

You should then start up PROCFG.EXE and select “Language Editor” (Shift+F5). A window will be displayed showing the currently installed languages. Position the highlight on STARTREK (notice that the name in the right column still says “English”) and press <ENTER> to begin working with it.

You will now see a full screen window with all of the prompts for your new STARTREK language file. Note on the top line the following options:

Alt-D

Restores the default prompt for the system prompt you are currently positioned on. So if you change the “Please enter your first and last name” prompt to read “Enter your name”, and then decide you liked the default prompt better, press Alt-D while positioned on the “Enter your name” prompt and the default prompt will be restored. The default prompt is always displayed on the bottom line of the Language Editor.

Alt-P

Opens the “Language Properties” dialog. This is where you define the properties for this language file. You'll notice the description for the STARTREK language file still says ENGLISH in the description. This needs to be changed to STARTREK so your users can select the new language, otherwise they will have two ENGLISH languages to choose from.

Alt-S

Opens the “Search” dialog. Enter a text string to search for in the language and ProBoard will move to the first prompt in the language file that matches. To repeat the search, in the prompts for the next occurrence of the text string you entered, press <Alt-N>. This allows you to easily search your entire language file for specified text, making changes to the prompts as the text is found.

A description of the fields in the “Language Properties” dialog is as follows:

Description

his is the name your users will see when choosing from language files they have access to.

Available

Whether this language is available to callers with the proper security level.

Level

The minimum security level required to select this language file.

Flags

The security flags needed in a users record to be able to select this language file.

Menu Path

The full path to your language specific menus. If specified ProBoard will look for language

specific menus in this path when users select this language. If left blank, ProBoard will display the menus in the path specified in PROCFG, Options, Paths, Menus.

Text Path

The full path to your language specific text files. If specified ProBoard will display language specific versions of your A?? text files if you create them. RIP files need to be in one common directory (in other words, ProBoard does not support RIP language specific text files at this time. If left blank, ProBoard will display the textfiles in the path specified in PROCFG, Options, Paths, Textfiles.

Questionnaire Path

The full path to your language specific questionnaire files. If specified ProBoard will use language specific versions of your questionnaires if you create them. If left blank, ProBoard will look for questionnaire files (files with the .Q-A file extension) in the ProBoard System Directory.

Once you are positioned on a prompt that you want to change, press <ENTER> to invoke the "Edit Language String" dialog. Each language string can be up to eight lines long. The color for each string is fully configurable (just like in the menu editor). You can insert the following control codes in your language string.

@a<file>@

Shows <file>.ANS/ASC/AVT (depending on the user's terminal setting). You can put this code anywhere in the prompt. (Example: "@aP_ENTER@" displays P_ENTER.A??)

@p<file>@

Runs <file>.PEX when this prompt is displayed. Parameters are allowed (seperated by spaces). You can put this control code anywhere in the prompt.

(Example: "@pTEST 2@" will run TEST.PEX with the parameter 2)

^

Toggles the highlight color on/off. The actual highlight color can be changed by selecting the "Highlight" option in the Language String editor.

****

Single Backspace.

\^

Generates a caret (the ^ character).

\<B><F>

Set background color to <B> and foreground color to <F>. The color has to be entered as one hex character (0-9,A-F).

The colors are:

0 - Black

1 - Blue
2 - Green
3 - Cyan
4 - Red
5 - Magenta
6 - Yellow
7 - White (Grey)
8 - Bright Black
9 - Bright Blue
A - Bright Green
B - Bright Cyan
C - Bright Red
D - Bright Magenta
E - Bright Yellow
F - Bright White (Grey)

For the background color, bright colors are not available. When you use 8-F for the background color, the color will have the blinking attribute.

Other color codes you can use are:

\HR -> Bright red
\HG -> Bright green
\HY -> Bright yellow
\HB -> Bright blue
\HP -> Bright purple
\HC -> Bright cyan
\HW -> Bright white
\LR -> Dark red
\LG -> Dark green
\LY -> Dark yellow
\LB -> Dark blue
\LP -> Dark purple
\LC -> Dark cyan
\LW -> Dark white

Some examples:

\0F Bright white on black
\1B Bright cyan on blue
\8E Bright yellow (blinking)
\70 Reverse (black on grey)
\LC Dark cyan on black

A description of the fields in the Language String dialog is as follows:

String #1 - String #8

The actual text that will be displayed to your users. Some prompts can be quite simple to understand (refer to Prompt #7) and some can seem quite difficult at first (refer to Prompt #207) with many options needing to be passed to the prompt, as well as hotkey values. A careful study of the default prompts, and how they display will help you to gain a better idea of how they work.

Hotkeys

The hotkeys needed for the prompt. If you study Prompt #207 you will see the hotkeys defined are “SCRDILAE”. If you look at the text strings for this same prompt, you will see why these hotkeys are needed. Be sure to specify the correct number and order of hotkeys for each prompt (if applicable). Any prompt which does not use hotkeys will have “n/a” displayed in the hotkey field and you will not be able to enter anything into the hotkey field.

Color

Allows you to define a color for the prompt text displayed to your users. Press <ENTER> to select the color you want to use for your prompt text.

Highlight

Allows you to define a color for the highlight. Remember, highlight is toggled on/off with the ^ character.

Prompt Color

Allows you to select a color for the users reply to this prompt. For example, if your prompt reads “Enter your first AND last name” and this color is set to cyan, when a user types their name, what they type will be displayed in cyan (or whatever color you select here).

The Language Editor will allow you to see what your prompts will look like when a user sees them without showing you the control codes. Move the highlight thru the language editor until you are close to, but not on, Prompt #191. You will notice it displays “(DUPLICATE- DELETED”) and the word “DELETED” is flashing. Now move the highlight selector to prompt #191 and press <ENTER>. You will now see the prompt with the control codes that make it flash, etc.

Validate Template Editor - (Shift-F6)

ProBoard allows you to easily validate users on your BBS using the values defined here in the Validate Template Editor.

The Validate Template Editor makes validation of your new users with any desired security level, loglevel, or combination of flag settings, and complete subscription support very easy.

Once your templates have been defined, all you need to do to validate a user using the templates, is to press <ALT-V> while positioned on their record in the User Editor. Then, simply select which validation template you want to apply.

Before you can use your validation templates, you must first define them. This is done here, in

the Validate Template Editor.

To add a validation template, press <Alt-A>. You can also press <F1> for help at any time while in the Validate Template Editor.

The following keys can be used while in the Validate Template Editor.

Page Up

Go to the previous record in the template editor.

Page Down

Go to the next record in the template editor.

CTRL+PgUp

Go to first template record.

CTRL+PgDn

Go to last template record.

ESC

Leave the template editor.

Alt-A

Add a template record.

Alt-D

Toggles the current template record for deletion.

Alt-F

Show/Edit your flag descriptions.

Alt-L

Shows a list of all currently defined templates.

Alt-S

Searches currently defined templates by descriptions.

Alt-N

Searches for next record containing matching description as in previous search.

The following fields make up a template "record" in ProBoard's Validate Template Editor:

Description

This is the description for this template. Be sure to describe your templates so you remember what security level they assign to a user at a later time when you validate them using the templates.

Security Level

This is the security level you want assigned to a user when validating them using this template.

Subscribe

How many days of “subscription” access a user will receive when validated using this template. ProBoard will use this value to calculate the Expiration Date in the users record.

Expiration Level

The security level you want to assign when the users subscription level reaches the Expiration Date.

Timebank Time

How many minutes of time you want to place into the users time bank account.

Timebank Kbyte

How many KBytes of file credit you want to place into the users time bank account.

Netmail Credit

How many netmail credits the user should receive when validated.

Validation Flags On

The flags to enable in a users record when validating them using this template.

Validation Flags Off

The flags to disable in a users record when validating them using this template.

Expiration Flags On

The flags to enable when a user reaches their expiration level.

Expiration Flags Off

The flags to disable when a user reaches their expiration level.

Comment

A comment to place in the users record when validating them using this template.

Loglevel

What loglevel to assign a user when validating them using this template. Use the <Space Bar> to cycle through the choices.

Hidden

f enabled, ProBoard will hide this users activity from the Last Callers list (Function 51), Show Users Online Function 50), and from the display of the User List (Function 13).

No Tops

If set to “Yes”, then this user will be excluded from any of the “Tops” lists that ProBoard creates. Refer to Menu Function 48 for more info.

Attention

If set to “Yes”, ProBoard will play the file ATTEN.MUS when this user logs on. The file ATTEN.MUS must be in your ProBoard System Directory (usually C:\PB).

NoKill

If set to “Yes” the user's record CAN NOT be removed from the userfile when using PBUTIL UK (User Killer) or PBUTIL UP (User Packer).

Ignore Download

Does this user have UNLIMITED download access? If set to “Yes”, ProBoard will ignore download limits for this user.

Deleted

If this is set to “Yes”, this means this Template Record is marked for deletion and will be cleared from your list of templates as soon as you leave the Validate Template Editor.

About ProCFG - (Shift-F9)

Displays a window which contains copyright information about ProCFG.

SECURITY

Levels & Flags

All security procedures toward a user are being done through the user's LEVEL and FLAGS. ProBoard can have levels ranging from 0 to 64000, and provides 32 flags (A-Z, and 1-6) that can be ON or OFF.

If a menu needs a certain level and flags, then only the users with a level equal to or higher than that level and with all the needed flags will be able to SEE and CHOOSE this menu option. It will remain invisible to all the other users.

An example will be given in the next chapter.

Trashcan

It is possible to specify names that cannot be used to log on to your system. Often used fake names are: “SysOp”, “BBS”, etc…

You can specify these names in a textfile called TRASHCAN.CTL. Each line in this file specifies an unwanted or illegal name.

An example file is included.

MENUS

Menus

Menus are one of the most important parts of a BBS. Menus are the direct interface between a user and your BBS. They are used to execute all the BBS functions, and they can have their own submenus. They take care of security, by disabling or hiding certain functions from certain users or groups of users.

ProBoard allows your BBS to have a very personal look, as your menus can be built and displayed in a VERY flexible way.

Setting Up Menus

A menu is basically line-oriented. Every line is linked to a function to be executed and to a textline to be shown to the user. Every line/function has its own level and flags, to make sure that not all of the menu items are available to every user.

A menu line has the following fields:

Textline..... Textline to be displayed to the user.
Hotkey..... Key to be pressed by the user to activate this menu item.
Function..... Function to be executed.
Data..... Data associated with this menu item.
Level..... Level needed to access this menu item.
Flags..... Flags needed to access this menu item.
Color..... Color of the menu line.

Menu Security

To clarify menu security, here's a simple example:

Suppose we have 4 users with the following levels and flags:

Name	Level	Flags
Pete	10	Z
Jerry	100	P
Al	100	R
Charlie	200	P & R

Let's define a menu with textlines only:

Text	Level	Flags
Good morning;	10	
,;	50	
Jerry	100	P
and;	150	
Al;	100	R
send their best wishes	200	P & R
.	300	
:-)	10	X

This would give the following result when the menu is displayed:

Pete	Good morning
Jerry	Good morning,Jerry
Al	Good morning,Al
Charlie	Good morning, Jerry and Al send their best wishes

The last 2 lines of the menu will NEVER be displayed, because none of the 4 users have the required level AND flags.

Creating Menus

Your BBS's main menu MUST! be stored in a file called TOP.PBM. All the other menus can have any file name you want to give them.

When you select the 'Menu Editor' option in PROCFG's main menu, a new window will be opened containing the menus already available (if any).

In this window, you can use the following keys:

Up/Down	Scroll up/down.
Enter	Select a menu.
Ins	Add a menu.

When you have selected a menu, a list of all the menu items will be displayed. To add a menu item, just move past the last item and press <Enter>. To insert an item, press <Ins>, to remove one, press <Del>. To edit a menu item, move the selector to that item and press <Enter>. You can change the menu prompt and highlight colors by pressing <Alt-P>. If you want to see how a menu will look, press <Alt-S> to simulate what your users will see when they view this menu.

You can also copy and paste menu items.

To copy an item, move to the item you want to copy, and press <Alt-C>. To insert the copied item somewhere else, move to the place where you want the menu item inserted, and press <Ctrl-P>. You can even copy and paste items across different menus.

A menu item has the following fields:

Textline..... This is the string to be displayed.
Leaving this field blank causes a blank line to be displayed to the user. A CR/LF will be sent after the menu line. To avoid this, just enter a ';' as the last character. This will cause the next textline of the menu to be appended to this one.
Special textline characters:
^ : Switches between normal and highlighted color.
~ : Replaced by the number of minutes the user has left in this session.
` : Replaced by the name of the current message area (more about this later).
@ : Replaced by the name of the current file area (more about this later).
This textline can also contain text macros like @<NAME>@ or @<NODE>@. More about this in the "Text macros" section.
You are not limited to two colors in menu lines. You can use the codes \1 to \7 to change colors, or \0 to return to the original color.

Hotkey..... Most of the menu functions must be chosen by the user, so ProBoard must react to certain key-presses from the user. Receiving the hotkey for a certain menu item will cause ProBoard to execute the function associated with this item. The hotkey can be any ASCII character or digit, but one character has a special meaning to ProBoard: <Ctrl-A> will make this function AUTOEXEC, which means that this function will be executed as soon as

this menu item is displayed (without really selecting this option).

Function..... The function associated with this menu item. Pressing the <Enter> key will cause a complete list of all the menu functions to be displayed in a separate window.

Data..... This field is optional with some of the functions. Basically, in this field you enter the parameters of a menu function. Eg. function 40 (Display ASC/ANS file) needs a filename as data (parameter). Function 1 (Goto menu) needs the name of the menu to be jumped to. Text macros can be inserted (see "Text macros" section).

Level..... Level needed to access this menu item.

Flags..... Flags needed to access this menu item.

Color..... Color for the menu's textline to be displayed in.

Hints

You can create a menu by using textlines, but you can also create menus that display a file to the user, in which all the options are shown. This technique works as follows:

The first item in the menu should have ';' as textline and <Ctrl-A> (autoexec) as hotkey. The function to be executed should be function 40 (Display file with hotkeys).

When the user accesses this menu, the specified file will immediately be sent. Setting all the remaining textlines of the menu to ';' will cause NOTHING BUT this file to be displayed.

Combined with the use of text macros, this allows you to show different looking menus for each node you are running. For example, if you use function 40, and enter “MAIN@<NODE>@” in the data field, node 1 will see file MAIN1 , node 2 will see MAIN2, etc.

This feature can be used to create a completely different set of menus for each node on your board!

Another powerful feature is the ability to create a special menu called GLOBAL.PBM This is the same as any other ProBoard menu except that all items defined in this menu will be appended to the bottom of each and every menu on your BBS. This can help convey a standard look and feel for each of your menus to your users. The prompt from the original menu and not from the GLOBAL.PBM is the prompt that will be displayed to your users. Be careful when using GLOBAL.PBM that you do not append hot-keys that are already in use on existing menus.

If you need more information about all the possibilities, take a look at the example menus supplied with ProBoard.

Menu Function Summary

The following is a summary list of the ProBoard Menu Functions.

Each Menu Function is explained in detail in the following section titled “Menu Functions Overview”.

Function	Action Performed
Function 1	GOTO Menu
Function 2	GOSUB Menu
Function 3	GOTO Previous Menu
Function 4	GOTO Menu and Clear Menu Stack
Function 5	Show ANS/ASC File
Function 6	Change State
Function 7	Shell
Function 8	Show Version Information
Function 9	Log Off
Function 10 ...	Show System Usage Per Hour (Graph)
Function 11 ...	Chat Request
Function 12 ...	Execute Questionnaire Script
Function 13 ...	Display User List
Function 14 ...	Display Time Info
Function 15 ...	Show A?? File and Wait
Function 16 ...	Change City
Function 17 ...	Change Password
Function 18 ...	Change Screen Length
Function 19 ...	Toggle Screen Clearing Codes
Function 20 ...	Toggle More Prompt
Function 21 ...	Toggle ANSI Codes
Function 22 ...	Check for Personal Mail / Files
Function 23 ...	Read Messages
Function 24 ...	Scan Messages
Function 25 ...	Quick Scan Messages
Function 26 ...	Show System Usage by Day (Graph)
Function 27 ...	Write a Message
Function 28 ...	Combined Boards Select
Function 29 ...	Display System Usage Per Week (Graph)
Function 30 ...	Show Directory
Function 31 ...	List Files
Function 32 ...	Download a File
Function 33 ...	Upload a File
Function 34 ...	View an Archive
Function 35 ...	File Search by Keyword
Function 36 ...	File Search by File Name

Function 37 ... Show New Files
Function 38 ... View a File
Function 39 ... Display Named File
Function 40 ... Display A?? File with Menu Hotkeys
Function 41 ... Toggle the Full Screen Editor
Function 42 ... Toggle Command Stacking/Hotkeys
Function 43 ... Clear Marked Messages
Function 44 ... Global Combined Boards Selection
Function 45 ... Display Text File and Wait
Function 46 ... Change User Level and/or Flags
Function 47 ... Make a Log Entry
Function 48 ... Show Hit Parade
Function 49 ... Select Message Area
Function 50 ... Show Users Online
Function 51 ... List Last Callers
Function 52 ... Remote User Editor
Function 53 ... Multi-Line (Internode) Chat
Function 54 ... Select File Area
Function 55 ... Show .GIF File Information
Function 56 ... Toggle IBM Characters
Function 57 ... Change Phone Number
Function 58 ... Change Data/Fax Phone Number
Function 59 ... Change User Alias (Handle)
Function 60 ... Run ProBoard SDK File
Function 61 ... Bulletin Menu
Function 62 ... Toggle AVATAR/0
Function 63 ... Toggle AVATAR/0+
Function 64 ... Show Graph (General)
Function 65 ... Display A??/RIP file with Hotkeys
Function 66 ... Change RIPscrip Font
Function 67 ... Toggle RIPscrip Graphics
Function 68 ... Edit Tagged Files
Function 69 ... Select a new Language
Function 70 ... Change Date Format
Function 71 ... Change Mailing Address
Function 72 ... Change FAX Number
Function 73 ... Change Country
Function 74 ... Change Default Protocol
Function 75 ... Select Message Area Group
Function 76 ... Select File Area Group

Menu Functions Overview

In this function overview, parameters between <> are REQUIRED, and parameters between [] are optional.

For example: <blank> is a required parameter.
[/M] is an option you might specify in addition.

Function 1: GOTO MENU

DATA: <menu name> [/M=<nr msgarea>] [/F=<nr filearea>] [/P=<pwd>]

This function makes ProBoard jump to menu <menu name>.

Option /M can be used to define one single menu for several message areas. You could jump to a 'general' message area menu and pass option /M=3, to make message area 3 active for the selected menu. Please refer to function 23 (Read messages) for more information about this option.

Option /F works exactly the same, but applies to file areas.

It is possible to protect a menu with a password by using the /P= option. For example: “SysOp /P=Test” would protect menu “SysOp” with the password “Test”. The user will have to enter this password to be allowed to move to the menu.

It is also possible to select the next available message or file areas with the use of the following parameters.

/M=+ (set next accessible Message Area)
/M=- (set previous accessible Message Area)
/F=+ (set next available File Area)
/F=- (set previous accessible File Area)

Data examples:

- **MSG /M=5:** This makes ProBoard GOTO the menu named “MSG” and sets the Message Area to Area #5.
- **FILE /F=10:** This makes ProBoard display the menu named “FILE” and sets the File Area to Area #10
- **MSG /M=+:** This makes ProBoard display the menu named “MSG” and selects the next Message Area that is available to this caller.
- **FILE /F=-:** This makes ProBoard display the menu named “FILE” and selects the previous File Area that is available to this caller.

Function 2: GOSUB MENU

DATA: <menu name> [/M=<nr msgarea>] [/F=<nr filearea>] [/P=pwd]

This function is largely the same as the Menu Function 1 (it even has the same parameters), but the menu this function is called from is pushed onto an internal stack. ProBoard will jump back to this menu when returning from menu <menu name> (by executing Menu Function 3).

Data examples: (see Menu Function 1 for complete parameters)

- **MSG /M=5:** This makes ProBoard display the menu named "MSG" and sets the Message Area to Area #5. It also places the previous menu on a stack. The previous menu can be returned to by using Menu Function 3.
- **FILE /F=10:** This makes ProBoard display the menu named "FILE" and sets the File Area to Area #10. It also places the previous menu on a stack. The previous menu can be returned to by using Menu Function 3.
- **MSG /M=+:** This makes ProBoard display the menu named "MSG" and selects the next Message Area that is available to this caller. It also places the previous menu on a stack. The previous menu can be returned to by using Menu Function 3.
- **FILE /F=-:** This makes ProBoard display the menu named "FILE" and selects the previous File Area that is available to this caller. It also places the previous menu on a stack. The previous menu can be returned to by using Menu Function 3.

Function 3: GOTO PREVIOUS MENU

DATA: (none)

This function makes ProBoard go back to the previous menu if you used Menu Function 2.

Function 4: GOTO MENU & CLEAR STACK

DATA: <menu name> [/M=<nr msgarea>] [/F=<nr filearea>] [/P=pwd]

This function performs the same tasks as Menu Function 1, but will clear the existing menu stack, thus preventing the user from returning to the previous menu.

Data examples: (see Menu Function 1 for complete parameters)

- **MSG /M=5:** This makes ProBoard display the menu named “MSG” and sets the Message Area to Area #5. The menu stack will be cleared preventing return to any previous menus already on the stack.
- **FILE /F=10:** This makes ProBoard display the menu named “FILE” and sets the File Area to Area #10. The menu stack will be cleared preventing return to any previous menus already on the stack.
- **MSG /M=+:** This makes ProBoard display the menu named “MSG” and selects the next Message Area that is available to this caller. The menu stack will be cleared preventing return to any previous menus already on the stack.
- **FILE /F=-:** This makes ProBoard display the menu named “FILE” and selects the previous File Area that is available to this caller. The stack will be cleared preventing return to any previous menus already on the stack.

Function 5: SHOW ANS/ASC FILE

DATA: <filename>

This function displays a file to the user with the extension .ANS or .ASC, depending on the user's ANSI-setting. The file must be stored in ProBoard's textfiles directory; <filename> should NOT contain an extension (max. eight characters). If ProBoard doesn't find the .ANS file, it will look for the .ASC file.

Function 6: CHANGE STATE

DATA: (none)

Allows a user to change the state in their user-record.

Function 7: SHELL

DATA: <command line>

With this function, you can instruct ProBoard to execute an external program. This program has to do its own serial I/O, so ordinary programs will only work if run locally.

Parameter <command line> needs full path and filename (.COM or .EXE extension included), and may contain some special codes. These codes will (at run-time) be replaced by a value or a string.

Special codes:

(*)

** Replaced by an asterisk ('*')

*# Replaced by the node number.

*\ Sends the message "SysOp is shelling..." to the user before the shell is executed, and the message "SysOp has returned..." afterwards.

*! Freezes ProBoard's system timer when shelling.

*= Do not call any fossil functions when shelling. Great for use in INIT.PEX files.

*A Writes a user's handle (alias) to DORINFOx.DEF instead of his/her real name.

*B Current baud rate.

*C Replaced by the full name & path of the command interpreter. It is the contents of the environment variable "COMSPEC". This usually is C:\COMMAND.COM.

*D Writes a full 52-line DOOR.SYS drop file to the current directory before shelling.

*E Writes an RA 1.1x EXITINFO.BBS to the current directory before shelling and reads it back afterwards. ProBoard creates an RA 2.xx by default so if you need the older EXITINFO.BBS format be sure to specify the *E Parameter.

*F User's first name.

*G Indicates whether user has ANSI (1) or ASCII (0) set.

*H Tells ProBoard not to disable the fossil driver when shelling.

*I Maximum user-inactivity (seconds).

*J Displays configured BBS name.

*K Displays current version of ProBoard.

*L User's last name.

*M ProBoard's start-up directory (including trailing '\')

*N Shell will NOT be logged in PROBOARD.LOG.

*O (not zero!) is replaced by the path of the current file area.

*P Com-port used by ProBoard (1-8).

*Q Don't let user know that ProBoard is shelling (!).

*R User's record number in USERS.BBS.

*S ProBoard's system directory (incl. trailing '\')

*T Time left for the user today (minutes).

*U Displays current user's handle (alias).

*V Disables adding 2 to the graphics capability line in DORINFOx.DEF when the user has AVATAR enabled.

*W Runs the shell in a window , so the status on the last line will not be cleared. This option only works with programs that send their output to the standard output device (no direct screen writes).

*X ALWAYS SWAP to disk/EMS, even if swapping is disabled in ProCFG.

*Y DO NOT SWAP to disk/EMS.

*Z Execute the command as if you typed it from the command line. The main use for this option is to execute DOS batch files. It is exactly the same as entering "*C /C <command>". It can also be useful to execute programs that are located in your path, without having to specify the exact directory name.

*_ (asterisk underscore) Tells ProBoard not to write underscores instead of spaces when writing a user's last name in DORINFOx.DEF. For example, users name is Harvey Van Hooten. Without this parameter, users name will be written to DORINFOx.DEF as follows:

HARVEY VAN_HOOTEN

By using the *_ (asterisk underscore) the users name will be written to DORINFOx.DEF as follows:

HARVEY VAN HOOTEN (with no underscore in last name)

*0 (zero) ProBoard will write a DORINFO1.DEF instead of a DORINFO<node>.DEF - great for running doors which require a DORINFO1.DEF file on a multi node ProBoard system.

*1 Installs a timer-function when shelling, which continuously redisplays the user's status line on the first screenline. This can be used with ALL programs. (Works great most full- screen protocols!)

*2 Same as *1, but uses the bottom screenline (try this with QuickEd or GEdit!)

Suppose ProBoard is in directory D:\PB.

Data example: *Q*STEST.EXE *B

ProBoard will execute D:\PB\TEST.EXE 2400 and the user won't see this happening.

To execute a batch file, you have to use the following syntax:

"*Z<filename>.BAT <parameters>". This is expanded to: "*C /C <filename>.BAT <parameters>".

When shelling, ProBoard writes a standard DORINFOx.DEF file, where x stands for the node-number. An RA-compatible EXITINFO.BBS file and a DOORWAY-compatible DOOR.SYS can also be created by specifying the *E and/or *D options. All these files are created in the directory where ProBoard is started from.

Function 8: SHOW VERSION INFORMATION

DATA: (none)

This function shows information about ProBoard's version number. If you have a registered copy of ProBoard, the SysOp's name and BBS-name will be shown. With this function, you can show off to your users that you're a nice SysOp: one that registered!

Function 9: LOGOFF

DATA: (none)

Shows the file GOODBYE.ANS/ASC and hangs up the phone.

Function 10: SHOW SYSTEM USAGE PER HOUR (GRAPH)

DATA: (none)

Shows a bar graph of the average system usage per hour. From the day you install ProBoard, it

will keep track of a usage rate per hour and per day. If you want to change the usage values, just edit the values using ProCFG.

Function 11: CHAT REQUEST

DATA: (none)

This function allows the user to make a request to chat with the SysOp. The user will be prompted for a reason why he wants to chat. The minimal length of this reason must be 5 characters. ProBoard will not allow the user to enter 5 spaces as a reason for chatting. If the SysOp does not answer, the status line will start flashing, indicating that the users wants to chat. You can view the reason for the chat by pressing <F8>.

You can define your own page-tune by creating a RemoteAccess- compatible description file. The format of this file is described in the reference section.

Function 12: EXECUTE QUESTIONNAIRE SCRIPT

DATA: <scriptname>

This function executes a questionnaire. A questionnaire is a common ASCII-file containing several commands to be executed by ProBoard. A questionnaire scriptfile has an extension .Q-A, and the user's answers will be stored in a file with the same filename, but with extension .ASW. Starting from version 1.15, you can write very powerful questionnaires by using the ProBoard SDK.

You can use the following commands in a scriptfile:

ASK <length> <number of variable>
eg: Ask 10 1

Asks the user something. The user has <length> characters to answer, and the answer will be stored in the variable <number of variable>. The maximum number of variables in ProBoard is 20.

CHANGECOLOR <color>
eg: ChangeColor 3

Changes the color in which the following text will be displayed. The number <color> is the ANSI color code, which means that this function works only for users who use ANSI.

CLEARSCREEN

Well, what do you think?

DISPLAY "<string>"
eg: Display "Hi there!"

Displays a string <string>. The string must be contained in double quotes; a '|' in a string will be replaced by CR/LF (new line).

IF <number of variable> = "<string>"
  eg. If 1 = "Y"

Directs the .Q-A file according to the user's answers. If the condition is met, all commands after the IF-statement will be executed, until an ENDIF is encountered.

ENDIF

Ends an IF-statement.

```
eg: Display "Do you have a hard disk?"
    GetChoice YN 1
    If 1 = "Y"
      Display "Storage space in Mb?"
      Ask 2
    EndIf
    Display "..."
```

GETCHOICE <options> <number of variable>
  eg: GetChoice YN 1

Forces the user to give a proper answer, chosen from <options>.

OUTPUTANSWER ["<description>"] <number of variable>
  eg: OutputAnswer "Name: " 1

Writes [“<description>”] and the value in <number of variable> to the .ASW-file. <description> is optional.

POSTINFO

Writes a header to the .ASW-file. The header contains some general information about the user (“Peter Piper answered on …”).

QUIT

Ends execution of the questionnaire script.

SETFLAG <flag> <ON/OFF>
  eg: SetFlag C ON

Sets/clears user flag <flag>. For compatibility with RA, it is also possible to use the QuickBBS/RA flags (A1-D8)

SETSECURITY <level>
  eg: SetSecurity 20

Changes the user's security level.

CAPITALISE [On|Off]
  eg: Capitalise On

Changes the way input from the user is displayed. When ON, all characters typed will be converted to uppercase.

DISPLAYFILE <filename>
  eg: DisplayFile TEST

Displays <filename>.ANS/ASC. It is identical to menu function 5.

EXEC <command>
  eg: Exec "*ZECHO TEST"

Execute a shell. It is identical to function 7. Note that you MUST use quotes (“) if the command contains any spaces.

LISTANSWER <var-num>
  eg: ListAnswer 4

Displays variable <var-num>, with a CR/LF at the end.

MENUCMND <num> [data]
  eg: MenuCmnd 11 "Yelling SysOp..."

Execute any menu command. If the data field contains spaces, you MUST use quotes (”).

WAITENTER
  eg: WaitEnter

Waits for the <Enter> key.

Function 13: DISPLAY USERLIST

DATA: (none)

Prompts the user for a name (or part of a name) and goes looking for it in the name fields of USERS.BBS. If the user doesn't specify a string (if he presses <Enter>), the entire userlist will be displayed. ProBoard uses a built-in “fuzzy search” algorithm, which will locate specific users even if you misspell their name.

Function 14: DISPLAY TIME INFO

DATA: (none)

Displays information about the current time, time online, remaining time, etc.

Function 15: SHOW ANS/ASC FILE & WAIT

DATA: <filename>

Displays an ANS/ASC-file (like function 5) and waits for the user to press <Enter>.

Function 16: CHANGE CITY

DATA: (none)

Allows the user to change the city in his user-record.

Function 17: CHANGE PASSWORD

DATA: (none)

Allows the user to change his password.

Function 18: CHANGE SCREEN LENGTH

DATA: (none)

Allows the user to change his screen's length (# lines).

Function 19: TOGGLE CLEAR SCREEN CODES

DATA: <blank> or <parameter>

Allows the user to decide whether he wants clear screen-codes sent or not.

In the data field, you can enter these parameters:

Ask	Will ask for new status
On	Toggles option on
Off	Toggles option off
Toggle	Toggles option (reverse of current condition)
Save	Save current status for this option
Restore	Restore status saved by "Save"
/Q	Quiet. Do not display "xxxx is now enabled/disabled"

Example: “Ask /Q”

This will ask the user if they want to enable or disable screen clearing and ProBoard will not display: "Screen Clearing is now enabled/disabled".

Note: Leaving the data field blank is the same as specifying the “Toggle” parameter.

Function 20: TOGGLE MORE PROMPT

DATA: <blank> or <parameter>

Allows the user to decide whether scrolling should pause when the screen is full.

In the data field, you can enter these parameters:

Ask Will ask for new status
On Toggles option on
Off Toggles option off
Toggle Toggles option (reverse of current condition)
Save Save current status for this option
Restore Restore status saved by "Save"
/Q Quiet. Do not display "xxxx is now enabled/disabled"

Example: “Ask /Q”

This will ask the user if they want to enable or disable the More? prompt, and ProBoard will not display: "Page Pausing is now enabled/disabled".

Note: Leaving the data field blank is the same as specifying the “Toggle” parameter.

Function 21: TOGGLE ANSI CODES

DATA: <blank> or <parameter>

Allows the user to choose ANSI graphics/colors or not.

In the data field, you can enter these parameters:

Ask Will ask for new status
On Toggles option on
Off Toggles option off
Toggle Toggles option (reverse of current condition)
Save Save current status for this option
Restore Restore status saved by "Save"
/Q Quiet. Do not display "xxxx is now enabled/disabled"

Example: “Ask /Q”

This will ask the user if they want to enable or disable ANSI, and ProBoard will not display: "ANSI codes enabled/disabled".

Note: Leaving the data field blank is the same as specifying the “Toggle” parameter.

Function 22: CHECK FOR PERSONAL MAIL & FILES

DATA: <blank> or [/F] or [/M] or both.

Searches all the message areas for messages addressed to the user that have not yet been received by him/her. If ProBoard finds an area with new mail, which the user does not have read access to, it will inform the user, and the SysOp.

This function also checks for personal files addressed to the user.

Specifying the [/F] option will force ProBoard to check only for new personal files.

Specifying the [/M] option will force ProBoard to check only for new personal mail.

ProBoard will also find mail addressed to a users alias.

Function 23: READ MESSAGES

DATA: <area> or <*> or <0>

Allows the user to read a message. If <area> is specified (a number ranging from 1 to 10,00), then only messages from message area <area> can be read. If <*> is specified, only messages from the active area can be read (refer to the goto/gosub menu functions and to function 49). If <0> is specified, only messages from areas selected in the combined boards can be read. When reading messages in combined message areas, messages are read “per area”, not in the order of the message number.

The user has several options when reading messages:

Forward : First to last.
Reverse : Last to first.
New : New messages not read by the user. Messages are read per area. If you have new messages in Area 1 and 5, ProBoard will first show all new messages in Area 1, then all new messages in Area 5.
Selected : Messages selected by name or subject.
Marked : Marked messages only.

When a message is read, the user has the following options:

Next : Next message.
Previous : Previous Message
Again : Show this message again.
Stop : Stop reading messages.
Mark : Mark this message for later use.
Reply : Reply to this message. The entire message will be passed to the external fullscreen editor, thus making it possible to quote text from the original message in your reply.
Unread : Set this messages status to 'Not received'. Moves to the next message (if any).
Delete : Delete this message from the message base.
Forward : Copy the message and address it to somebody else. This will only work in a LOCAL message area.
Move : Move this message to another message area.
Private : Toggles the private/public status of the message.
Export : Export this message to any file.
+ / - : Show the next/previous message in the reply-chain.
Original : Go back to the message where you first used +/- to follow replies.

Function 24: SCAN MESSAGES

DATA: <area> or <*> or <0>

Gives an overview of the messages. Only the message-header will be displayed, and the user has the possibility to mark messages for later retrieval. The options follow the same rules as in function 23 (Read messages).

Function 25: QUICKSCAN MESSAGES

DATA: <area>/<*>/<0>

Same as the previous function, but only an overview in short form is displayed. The user does not have the possibility to mark messages. The options follow the same rules as in function 23 (Read messages).

Function 26: USAGE GRAPH BY DAY (GRAPH)

DATA: (none)

Displays a graph of your system's use for the last <xx> days. The parameter <xx> is determined by the setting in PROCFG, under Options (F1), System Options, # Usage Graph Days.

Function 27: WRITE A MESSAGE

DATA: <area> or <*> [/L] [/N=Netmail address] [/T="Name"]
[/S="Subject"]

Allows the user to write a message. The area <area> (a number ranging from 1 to 10,000) can be replaced by <*> (refer to function 23). ProBoard's “fuzzy search” will assist users who enter another users name improperly when they are writing messages in local areas. ProBoard will search the user file, and give the user a choice of all users whose names are close to what they typed.

Optional parameters:

/L The user will be logged off after writing this message.
/N Writes a netmail message to the destination address specified. This could also be used to send netmail to a UUCP gateway.
/T The "To" (the addressee's name) is automatically specified. It is not necessary to use quotes (") when the username does not contain spaces.
/S The subject of the message is automatically specified. It is not necessary to us quotes (") when the subject does not contain spaces.

Examples:

/N=1:282/30 This would address a netmail message to Fidonet node number 1:282/30.

/T=SysOp Writes a message to the SysOp.

/T="Peter Piper" Writes a message to Peter Piper.

/S="ProBoard" Writes a message with the subject line already filled in. The user will not be able to change the subject.

Function 28: COMBINED BOARDS SELECT

DATA: <blank> or [/M]

Allows the user to select multiple message areas when using the 'combined boards read/scan' function. This is useful when a user is not interested in certain message areas.

Also allows users to select which areas ProBoard will check [/M] for new mail. This is very useful for visiting SysOps who don't want to be forced to read echomail they already received on their own system.

Function 29: SYSTEM USAGE PER WEEK (GRAPH)

DATA: (none)

Displays a graph of the system-usage in percent for the last 24 weeks.

Function 30: SHOW DIRECTORY

DATA: [directory name]

Shows a list of all the files stored in the specified directory. If no directory is given, the user will be prompted for one.

Function 31: LIST FILES

DATA: <area> or <X>

Lists the files in the specified file area. If <X> was specified, the files in the active file area will be shown. The list of these files must be stored in a textfile created by the SysOp (refer to file area configuration). This textfile should contain the names and descriptions of the files. The description can be of any form you like.

A line in the textfile should look like this:

a) <filename> <blanks> <description>

Filename + date + size + description will be displayed in the appropriate colors.

b) <+> <description>

The <+> will cause the description to be placed at the same horizontal cursor position and in the same color as the description from (a). This line will be displayed when searching for files.

c) <!> <description>

The description will be placed at the left of the screen, in the same color as the descriptions above. This line will be displayed when searching for files.

d) <;> <description>

Same as in (c), but the color is white.

e) <description>

Same as in (d), but this line will NOT be displayed when searching for files.

Example:

```
============================================================
PB_220.ZIP ProBoard version 2.20
+Best BBS program in the world
!Original Belgian Product
;(made possible by Burger King in the USA)
============================================================
```

This will be output as:

```
============================================================ [c2]
PB_220.ZIP   07/08/93 600583    ProBoard version 2.20         [c1]
                                Best BBS program in the world [c1]
Original Belgian Product                                      [c1]
(made possible by Burger King in the USA)                     [c2]
============================================================ [c2]
```

[c1] stands for color 1, [c2] stands for color 2.

If a file area is configured as being CD-ROM, the file listing should look slightly different. Option (a) will become:

a) <filename><blanks><date><blanks><filesize><blanks><description>

There is one more thing you can do to make your file listings more colorful (previously undocumented) : Inserting Ctrl-A to Ctrl-G characters in your file listing will change the color to:

Ctrl-A Red

Ctrl-B Green
Ctrl-C Yellow
Ctrl-D Magenta
Ctrl-E Blue
Ctrl-F Cyan
Ctrl-G White

Function 32: DOWNLOAD A FILE

DATA: <area selection> and/or </parameter>

Allows the user to download a file. The file areas a user can download from are defined by the area selection. This is a list of area-specifications, separated by blanks. Each specification has the form:

[+]<area>
<-><area>
'+' stands for 'Include this area in the area-list', and is optional.
'-' stands for 'Exclude this area from the area-list'.

The parameter <area> should be one of the following:

(*)

*	All areas.
C	CD-ROM areas only.
X	Currently active area.
<n>	Area number <n>.
<n1-n2>	Areas <n1> to <n2>.

Examples:

(*)

*	All areas
* -C	All non-CD-ROM areas.
* -3-9 +5	Areas 1,2,5,10,11,...
X +2	Active area + area 2.
C -2	All CD-ROM areas, except area 2

Of course, a user must have the necessary download security in an area to be allowed to download files from it.

You can also specify several other parameters:

/A ..... Allows the user to download ANY file accessible by DOS. When using this option, the full path and filename must be specified. This option is intended for

remote SysOps.

/F=<file> ..... The user will not be prompted for a file, but the file <file> will immediately be sent to the user (useful for textfiles describing all the files on the BBS). Parameter <file> should contain the full path name. (eg. /F=C:\PB\PB.DOC). Can also be combined with the </T> option (see below).

/F=[<area>]<file> ...Download the <file> from [<area>] directly. The following example would download the file PB_220.ZIP directly from file area #13. e.g. /F=[13]PB_220.ZIP

/F=@<filelist> .....Download the files listed in the textfile <filelist>. e.g. /F=@C:\PB\ALLFILES.RSP

/I ..... Ignore DL kbytes.

/K=<key> .....Use the protocol key <key>.

/L=<log> .....Write to an additional log file <log>. The format for each line is: <D/U> <area> <file> <size> <free YES/NO>.

/N ..... No log. Don't log anything.

/P ..... Download all personal files for the user.

/Q ..... Start download right away without prompting the user. You must use the /F= and the /K= parameters so ProBoard knows what file to send, using what protocol when using this option.

/T ..... Allow the user to download the specified file without checking the users remaining time left. Refer to "Free Files" for more details. Let's say you want to let users download the latest version of ProBoard and you don't care about the time it takes or how much time they have left. In the data line put the path and file name followed by the </T> parameter. An example of doing this is as follows: C:\FREEFILE\PB_220.ZIP /T

A Small hint regarding free files: if you want one particular area to be completely free (time and Kb), you can make a menu choice like this:

Function: 32
Data : 41 /T

This will allow the user to download anything from area 41, regardless of their time remaining.

ProBoard will now also copy files from CD-ROM to a local drive when the “Copy Local” flag is set to “Yes”. It will copy files to a directory called CD_TEMP, created in the directory where ProBoard was started from. You can specify your own directory by creating an environment variable called CDTEMP containing the name of the directory (+drive!!) If ProBoard can not find the directory as specified it will attempt to create it. Refer to the chapter “CONFIGURATION” under File Areas, for more information on the “Copy Local” and CDTEMP environment variable options in ProBoard.

Important ProBoard uses a highly optimized index to find your files when users want to download them. Be sure to run PBUTIL FI (File Index) at least once a day to create the index, or ProBoard will not be able to find the files. You may want to run it more often than once a day if you add new files to your system.

Function 33: UPLOAD A FILE

DATA: <uploadpath> or </parameter>

Allows the user to upload a file to the BBS. If <uploadpath> is specified, the upload will be placed in that directory. If not, it will be placed in the default upload-directory specified in PROCFG. (Unregistered versions of ProBoard will always use the default upload directory).

Upon successful reception of the file, the user will be prompted for a description of that file. The description can be several lines. If it begins with a '/', this description will be written to the file FILES.PVT in the upload-directory, else it will be written to the file FILES.BBS.

All uploads are now logged to a file in the ProBoard System Directory called UPLOAD.LOG. The log file is in the following format:

01-Feb-94 14:52:38 Upload of PB_220.ZIP by John Riley
01-Feb-94 14:56:01 Upload of PROCFG.EXE by Jason Bock

The parameters you can specify are: ———————————————————

/D ..... Do not ask user for upload descriptions.
/F=<file> .....Upload <file> (for non-batch protocols).
/K=<key> ..... Use protocol key <key>.
/L=<log> ..... Write to an additional log file <log>. The format for each line is: U O <file> <size> NO.
/N ..... No log. Do not log anything.
/P ..... Upload private files. If the data field contains "/P", the user can upload a personal file to another user. In this case the upload will be placed in the private upload directory (specified in ProCFG).
/Q .....Start upload right away, without prompting the user. You must specify a /K=<key> for this to work.

Function 34: VIEW ARCHIVE

DATA: <area selection>

View the contents of a ZIP/LZH/ZOO/ARC/ARJ/RAR-file. The file specification input by the user will be looked for in the areas specified in <area selection>. ProBoard's Archive Viewer looks at the contents or “header” of the file the user selects to view to determine it's archive type rather than the file's extension (.ZIP/.ARJ etc…) This function now uses the file index to locate the archive to view so it is much faster than in previous releases of ProBoard. Please refer to Menu Function 32 (Download) for more information.

Function 35: KEYWORD SEARCH

DATA: <area selection> [/FG] [/FG=group]

Looks for a character string in the file descriptions. If the string is found, the related file and its description will be displayed. The character string will be looked for in the areas specified in <area selection>. ProBoard now supports keyword searches in lines starting with | (PCBoard CD-ROM lists) such as the popular Night Owl CD-ROM's. This means a keyword will be found in these extended descriptions as well. Please refer to Menu Function 32 (Download) for more information.

Option /FG will search only in the areas in the currently selected file group. You can define a file group to search in by specifying a group number in /FG=<group number>.

Function 36: FILENAME SEARCH

DATA: <area selection> [/FG] [/FG=group]

Looks for a filename in the file listings (wildcards allowed). The filename will be looked for in the areas specified in <area selection>. Refer to function 32 (Download) for more information.

Option /FG will search only in the areas in the currently selected file group. You can define a file group to search in by specifying a group number in /FG=<group number>.

Function 37: SHOW NEW FILES

DATA: <area selection> [/FG] [/FG=group]

Shows a list of files more recent than a date specified by the user. If the user does not specify a date, then that date will be the last time this user logged in. The files will be looked for in the areas specified in <area selection>. Please refer to function 32 (Download) for more information.

Option /FG will search only in the areas in the currently selected file group. You can define a file

group to search in by specifying a group number in /FG=<group number>.

Function 38: VIEW A FILE

DATA: <directory>

This functions asks for a filename, and then looks for (and shows) the file stored in <directory>.

Function 39: DISPLAY NAMED FILE

DATA: <full filename>

Displays a file to the user. The data field <full filename> must contain path, name and extension.

Function 40: DISPLAY ANS/ASC FILE WITH MENU HOTKEYS

DATA: <filename without extension>

This function is the same as function 5, but can only be used in 'autoexec-menus', because it shows an ANS/ASC-file AND checks for menu

hotkeys at the same time. Please refer to the section about menus for more information about the autoexec-concept.

Function 41: TOGGLE FULLSCREEN EDITOR

DATA: <blank> or <parameter>

Lets the user decide whether they want to use ProBoard's line editor or an (external) fullscreen editor.

In the data field, you can enter these parameters:

Ask Will ask for new status
On Toggles option on
Off Toggles option off
Toggle Toggles option (reverse of current condition)
Save Save current status for this option
Restore Restore status saved by "Save"
/Q Quiet. Do not display "xxxx is now enabled/disabled"

Example: “Ask /Q”

This will ask the user if they want to enable or disable the full
screen (external) editor, and ProBoard will not display:

"Full Screen editor is now enabled/disabled".

Note: Leaving the data field blank is the same as specifying the “Toggle” parameter.

Function 42: TOGGLE COMMAND STACKING

DATA: (none)

Lets the user decide whether he will work with hotkeys or with combinations of hotkeys and command stacking (à la Opus). The command stack execution is initiated by pressing <Enter>. A ';' in the command stack will be replaced by <Enter>.

Eg. M1WSysOp;Subject;Y

This would write a private message in message area 1 to the SysOp, with subject “Subject”.

Function 43: CLEAR MARKED MESSAGES

DATA: (none)

Clears all marked messages.

Function 44: GLOBAL COMBINED BOARDS SELECTION

DATA: (none)

Allows the user to select or deselect all areas in his combined boards selection.

Function 45: DISPLAY TEXTFILE & WAIT

DATA: <full filename>

Displays a textfile to the user, then waits for the user to press <Enter>. The filename requires full path, name and extension.

Function 46: CHANGE USER LEVEL AND/OR FLAGS

DATA: [level] [flag +/-] [flag +/-] ...

Changes a user's level and/or flags. [level] can occur only once in the parameter array, flags can be toggled on/off by specifying the flag, followed by +/-.

Data example: 10 A+ 3-

This would set the user's level to 10, set flag A and clear flag 3.

Function 47: MAKE A LOG ENTRY

DATA: <log entry>

Writes <log entry> to PROBOARD.LOG, thus allowing the SysOp to customize his log.

Function 48: SHOW HITPARADE

DATA: <Mn>/<Kn>/<Tn>/<Un>/<Fn>/<Cn>/<On>

This functions returns an overview of the most active users in several fields.

The 'n' with the parameters stands for the number of users to be displayed in the hitparade. (unregistered versions default to 5)

M Best message-writers
K Best downloaders (Kb)
T Best downloaders (# downloads)
U Best uploaders (Kb)
F Best uploaders (# uploads)
C Best callers (# times called)
O Total time online

Data example: U15

Function 49: SELECT MESSAGE AREA

DATA: <area selection> or </parameter>

Lets the user select a new message area. This function is to be used in cooperation with other message-related functions.

Valid area selection parameters are:

X Current Area
E All Echo Areas
N All NetMail Areas
L All Local Areas
* All Areas
<n> Area #<n>
<n1>-<n2> Areas #<n1> to #<n2>

Other valid parameters are:

/MG ..... Makes the user select a message group if they haven't done so already.
/MG=<num> ..... Only allows user to select from message areas belonging to message group <num>.

/N ..... ProBoard will run even faster since it will
not check for waiting mail (no "*" will be
displayed indicating new mail) if you
specify this parameter. This can be useful
if your BBS has many large message areas,
in particular, Squish or JAM. Remember,
ProBoard can be configured for 10,000
message areas so a full "new" mail check in
each area can be time consuming.

Examples:

(*)
* -E All non-echomail areas
1-20 50-60 Areas 1..20 and 50..60
N 10-20 -15 All netmail areas, plus areas 10..20, except area #15
/N Do not have ProBoard perform the new mail check in each
message area. Great for Squish and JAM.

Note: ——— Since ProBoard parses only the first letter of an alpha area selection, the keywords from prior versions, [Local / Net / Echo] can still be used and will still work. It is recommended however, that you change to the new parameter(s) since the keywords [Local / Net / Echo] may not be supported in upcoming versions.

Function 50: SHOW USERS ONLINE

DATA: <blank> or /H

Displays users who are logged in on other nodes. Of course, this function is only useful on multi-user systems. To protect the SysOp's health, it is possible not to display the SysOp's name here (Refer to the ProCFG section).

Valid parameters are:

/H ..... Show the user's handle (alias) instead of
their real name.
<blank> ..... Will display the user's real name.

Function 51: LIST LAST CALLERS

DATA: <number of users to be displayed> [/H]

Gives an overview of the users that have most recently logged in (for ALL the nodes). In the non-registered version, the number of users shown is forced to 5. Specifying the parameter /H will show the user's aliases instead of their real names.

Function 52: REMOTE USER EDITOR

DATA: (none)

Allows you to adjust a user's level (or even delete him), without you having to be at the computer (VERY handy for co-SysOps!).

Function 53: MULTILINE CHAT (INTERNODE CHAT)

DATA: - <blank> or /H

This is certainly one of the nicest Menu Functions in ProBoard! This function allows two users on different nodes to chat with each other IN REAL TIME!!! You will actually see the other user type (mistakes?), as if you were in chat-mode with the SysOp. This is UNlike other systems, where entire LINES are sent to the other node.

User A on node X will have to specify the node he wants to chat with, then user B on node Y will be prompted if he wants to chat. It is VERY (!VERY!) important that your system supports full file & record locking to use this option. When you are not running a LAN, you MUST install SHARE.EXE! ProBoard will definitely lock up when SHARE.EXE is not installed on a stand-alone system.

Valid parameters are:

/H ..... Will show the users handle (alias) instead of their real name to the users on other nodes, when prompting them to engage in a chat.
<blank> ..... Will display the users real name instead of their alias when prompting users on other nodes to engage in a chat.

Function 54: SELECT FILE AREA

DATA: <area selection> or </parameter>

Lets the user select a new file area. This function is to be used in cooperation with other file-related functions. Refer to functions 1 and 32 for more information.

Valid parameters are:

/FG ..... Makes a user select a file group if they haven't done so already.
/FG=0 ..... Makes the user always have to select a file group.
/FG=<num> ..... Only display file areas belonging to file group number <num>.

Function 55: SHOW .GIF FILE INFO

DATA: <area selection>

Prompts the user for a .GIF-filename (wildcards allowed) and displays resolution and number of colors for the file(s). The files will be looked for in <area selection>. Refer to function 32 (Download) for more information.

Function 56: TOGGLE IBM CHARACTERS

DATA: (none)

Allows the user to disable/enable extended IBM characters. When disabled, all IBM-specific characters are converted to standard ASCII characters (“+-|”)

Function 57: CHANGE PHONE NUMBER

DATA: (none)

Allows the user to change his phone number stored in his user record.

Function 58: CHANGE DATA/FAX PHONE NUMBER

DATA: (none)

Allows the user to change his data/fax phone number stored in his user record.

Function 59: CHANGE USER ALIAS

DATA: (none)

Allows the user to change his alias. It is not allowed to use an alias that is being used by another user.

Function 60: RUN PROBOARD SDK FILE

DATA: <program> [data]

Loads & executes a ProBoard Executable (PEX file) created using the ProBoard Software Development Kit (SDK). The PEX file can be run from any directory. It is the responsibility of the PEX programmer to search for the appropriate data files when their PEX is run. If you are not sure if the PEX you are running does this then it's a good idea to place the PEX file directory (Specified in PROCFG, F1-Options, Paths, PEX Files).

Important

Any PEX files that were written for versions of ProBoard prior to 2.00 will have to be recompiled with the new 2.1x SDK before they can run under ProBoard 2.16.

Function 61: BULLETIN MENU

DATA: <filename> [prompt]

Displays <filename>.ANS/ASC (like function 5), and prompts the user for a file suffix. This suffix is appended to <filename>, and the file with the resulting filename is displayed. Obviously, <filename> should not be longer than 7 characters. The optional [prompt] parameters defines a prompt to be shown to the user. For example: Enter a bulletin.

Data example: BULLET Enter a bulletin:

Function 62: TOGGLE AVATAR/0

DATA: <blank> or <parameter>

Allows the user to toggle AVATAR/0 on/off in their user record.

In the data field, you can enter these parameters:

Ask	Will ask for new status
On	Toggles option on
Off	Toggles option off
Toggle	Toggles option (reverse of current condition)
Save	Save current status for this option
Restore	Restore status saved by "Save"
/Q	Quiet. Do not display "xxxx is now enabled/disabled"

Example: “Ask /Q”

This will ask the user if they want to enable or disable AVATAR/0 screen display codes, and ProBoard will not display: "Avatar codes are now enabled/disabled".

Note: Leaving the data field blank is the same as specifying the “Toggle” parameter.

Function 63: TOGGLE AVATAR/0+

DATA: <blank> or <parameter>

Allows the user to toggle AVATAR/0+ on/off in their user record.

In the data field, you can enter these parameters:

Ask	Will ask for new status
On	Toggles option on
Off	Toggles option off

Toggle Toggles option (reverse of current condition)
Save Save current status for this option
Restore Restore status saved by "Save"
/Q Quiet. Do not display "xxxx is now enabled/disabled"

Example: “Ask /Q”

This will ask the user if they want to enable or disable AVATAR/0+ screen display codes, and ProBoard will not display: "AVT/0+ codes are now enabled/disabled".

Note: Leaving the data field blank is the same as specifying the “Toggle” parameter.

Function 64: Show Graph (General)

DATA: <graph type> [/N=<days>] [/T="text"]

This menu function is a direct system hook to the _GRAPH.PEX file supplied in PB_220.ZIP

<graph type> "Hourly", "Weekly", "LastDays" or "Speed".
/N=<days> Number of days to read from BINLOG.PB (Optional)
/T="text" Text to display on the top bar of your graph. A "%d" (no quotes) is replaced by the number of days/weeks

Example:

Hourly /N=14 /T="Usage graph for the last 2 weeks"
Speed /N=60 /T="Cool baud rate statistics for the last %d days"

The C++ source code for _GRAPH.PEX is included with ProBoard.

You can add your own <graph type> by editing and compiling the _GRAPH.CPP file.

Function 65: Display A??/RIP File with Hotkeys

DATA: (none)

Allows you to display an A??/RIP file to users. Very similar to Menu Function 40 except that RIP files are supported.

Function 66: Change RIPscrip Font

DATA: (none)

Allows the user to select either the large RIP font (80×24) or the small RIP font (80×43) for displaying system prompts etc.

Function 67: Toggle RIPscrip Graphics

DATA: (none), <blank> or <parameter>

Allows the user to toggle RIP graphics on/off.

In the data field, you can enter these parameters:

Ask	Will ask for new status
On	Toggles option on
Off	Toggles option off
Toggle	Toggles option (reverse of current condition)
Save	Save current status for this option
Restore	Restore status saved by "Save"
/Q	Quiet. Do not display "xxxx is now enabled/disabled"

Example: “Ask /Q”

This will ask the user if they want to enable or disable RIP graphics, and ProBoard will not display: "RIPscrip graphics are now enabled/disabled".

Note: Leaving the data field blank is the same as specifying the “Toggle” parameter.

Function 68: Edit Tagged Files

DATA: (none)

Allows users to edit the list of tagged files.

Function 69: Select a New Language

DATA: (none)

Allows users to select a new language file for the system prompts.

Function 70: Change Date Format

DATA: (none)

Allows the user to select a new date format for system date displays. Valid choices are: MM/DD/YY, YY/MM/DD, and DD/MM/YY.

Function 71: Change Mailing Address

DATA: (none)

Allows the user to change their mailing address. Displays users current mailing address (if any) and presents a new line for them to begin typing. If nothing is typed on this first line by the user, the change mailing address function is terminated.

Function 72: Change FAX Number

DATA: (none)

Allows the user to change the FAX Number in their user record.

Function 73: Change Country

DATA: (none)

Allows the user to change the country in their user record.

Function 74: Change Default Protocol

DATA: (none)

Allows the user to change the default protocol for downloads and uploads. ProBoard will no longer ask for a protocol when a download or upload session is started.

Function 75: Select Message Area Group

DATA: (none)

Selects a new message area group. See function 49 for more information.

Function 76: Select File Area Group

DATA: (none)

Selects a new file area group. See function 54 for more information.

RIPscrip GRAPHICS

RIPscrip GRAPHICS (RIP)

ProBoard auto-detects and supports RIPscrip (referred to as RIP) Graphics.

A flashing “R” on the right side of the status bar indicates the RIP was detected. ProBoard will display the file INTRO?.RIP (if it exists) instead of INTRO?.ANS/ASC (only if RIP is detected).

Missing Icons are automatically sent to the caller. You must have a Zmodem protocol installed with a hotkey of “Z” for this to work.

ProBoard's built in message reader is fully RIP capable as are the “More” prompt, the “Enter” prompt and several other internal ProBoard functions. Almost all of the internal ANS/ASC/AVT files can be replaced with RIP files. You need only create a RIP file with the proper name and copy it into your RIP directory.

To add your own RIP menus to ProBoard, do the following:

Once you have designed a RIP screen for your menu, you can add it to a menu by pressing Alt-P (Prompt) in the menu editor. You will be able to set “RIP menu” to “Yes”, and enter a file name for a RIP screen. Enter the name of your RIP screen here. That's all there is to it!

There are some additional settings you can change for each menu item, which makes the RIP support in ProBoard extremely flexible:

Show remote
  * Send menu-line/ANSI-file to remote when RIP enabled
  (Default No).
Show local
  * Send menu-line/ANSI-file to local screen when RIP
  enabled (Default Yes).
Reset screen
  * Reset RIP windows before executing this function.
  (Default Yes).

These options are very powerful. When the user has RIP enabled, the SysOp will still see the “normal” ANSI screens. Don't touch the defaults if you want it like that. You will see that the RIP support in ProBoard is a lot friendlier towards to SysOp than in other BBS software. On the local end, the SysOp will see everything as if the user had no RIP enabled. This allows the SysOp to follow what the user is doing. (Other software like RA and Wildcat show the RIP “bang” commands, which are totally unreadable for the SysOp).

Two new menu functions have also been added to support RIP:

Function 66: Change RIP text font (large or small) for non-RIP screens.
Function 67: Enable/Disable RIP. Allows the user to disable RIP even it
        was detected.

FILE TAGGING

File Tagging

ProBoard supports file tagging when users are browsing the file list, performing a keyword or file search, or performing a new file search. File tagging allows your users to easily select files they want to download as they browse your available files.

QWK

QWK

Internal ProBoard QWK support has been added! It is provided as a PEX file called QWK.PEX. The only things necessary to configure are the following:

- The packet file name your BBS will create (ProCFG/Options/QWK).
- The QWK area name in each message area configuration.

Call the QWK PEX as any you would any other PEX file. No configuration files are necessary. One parameter can be given: “D” (ie “QWK D”). This will pack all mail and start the download right away.

USERS

Users

The file USERS.BBS contains all the data about the users, such as level, number of times called, combined boards access, messages last read, etc. Use ProBoard's User Editor to add/edit/delete users in the USERS.BBS file. If you delete users you must run PBUTIL [UK]. More info on this in the section on PBUTIL.

Log Levels

Each user has a certain loglevel, to determine how information about that user will be written to ProBoard's log (PROBOARD.LOG).

This can be useful if you suspect a user of logging in using 2 different names. If you give those users a higher log level, you can track their behavior.

The following levels are possible:

Friend	* NOTHING BUT login, logoff and errors will be logged
Normal	* Login
	* Logoff
	* Errors
	* Writing SysOp messages
	* SysOp paging
	* Downloads
	* Uploads
	* Questionnaires
Suspicious	* Everything from 'Normal'
	* Reading messages
	* Hitparades
	* Last callers
	* Graphics
Extensive	* Everything from 'Suspicious'
	* ALL the movements between menus (this can make the logfile HUGE)

ECHOMAIL & NETMAIL

Echomail & Netmail

ProBoard fully supports Echomail and Netmail, according to the FTSC (FidoNet Technical Standards Committee) specifications.

Echomail

When a user enters a message in a message area configured as an Echomail area, an origin line will be added to that message. This origin line is obtained from the message area's configuration. If there is no origin line specified for this message area, the default origin line will be used. The default origin line is specified in PROCFG under Options (F1) - Site Info, in the field “Default Origin Line”.

To import/export Echomail, you need an Echomail processor. No such program is included, but since ProBoard uses the .MSG (Fido), Squish, or Hudson message-base structures, a large number of Echomail processors are available that will work with ProBoard.

Some of the echomail-processors compatible with ProBoard are:

Squish FastEcho IMail
GEcho FMail FreeMail

Netmail

A Netmail message (in short: Netmail) is a message that has a fixed destination within the network. This destination is defined by a node-number of the form “Zone:Net/Node.Point”. ProBoard will first find out whether this address exists, and will then (if it exists) tell you the name of the node where the Netmail is to be sent to.

When entering a node number in ProBoard, it is possible to look up the nodes you want by entering a '?'.

eg: ? Shows a list of all zones.
2:? Shows a list of all nets and regions in zone 2.
2:292/? Shows a list of all nodes in net 2:292/

ProBoard needs a nodelist to use Netmail. This nodelist is usually available on every node in the network. ProBoard generates an index file for its own use (NODE_IDX.PRO) by running PBUTIL NC.

For more information about PBUTIL's NC-option, please refer to the section on PBUTIL.

To import and export Netmail, you need an external utility like MAILSCAN, MBUTIL or ZmailH.

PBUTIL (The ProBoard Utility Program)

PBUTIL

PBUTIL is an extra utility provided with the ProBoard package. It performs all of the maintenance your ProBoard BBS system needs.

Some PBUTIL Operations have Options and/or Parameters that can be specified on the command line when running them. Operations which have command line Options and/or Parameters are identified in the list below with a “(+)” after their “Operation Identifier”, (the two letters necessary to begin each PBUTIL operation).

PBUTIL is called along with an “Operation Identifier” (a two letter sequence specifying the operation to be executed (along with any options):

PBUTIL <operation> [option]

PBUTIL Operation	Description	Options/Parameters
DM (+)	Daily Maintenance	-H
FB	Fix BINLOG.PB	
FC (+)	File Counters	-N<x> -F -R
FI	FILES.BBS (re)-Indexer	
HF (+)	Hatch Personal File	<files> -T= -F= -D -C
MI	Message (re)-Indexer	
ML	Message Linker	
MP (+)	Message Packer	-R -D -F -K -H -J
MU	Music Player	
NC	Nodelist Compiler	
UF	Userfile Fixer	
UI	Userfile (re)-Indexer	
UK (+)	Userfile Killer	-C<n> -D<n> -L<evel>
UP (+)	Userfile Packer	-R -K
US	Userfile Sorter	

PBUTIL can be run from any directory provided that it (PBUTIL.EXE) is found in your DOS path. Refer to your DOS manual for more information on the PATH statement.

[DM] Daily Maintenance

At this time, the DM function of PBUTIL has only one function: it (re)creates TOPS.PRO. You should run it as part of your daily maintenance. If you specify the -H option, the file TOPS.PRO will be created using the user's aliases instead of their real names.

[FB] Fix BINLOG.PB

You should run PBUTIL FB if you ever receive the message that your BINLOG.PB file is corrupt. This “fix” routine will remove any and all unreadable data from the BINLOG.PB file. For more information on BINLOG.PB, refer to Menu Functions 10, 26, 29 and 64.

[FC] File Counters

After each download, ProBoard adds a line to the file DOWNLOAD.LOG. This file's only purpose is to be used by the FC module of PBUtil. FC reads the file DOWNLOAD.LOG and updates file counters in every file listing, to keep up with the number of times every file has been downloaded.

Specifying option -N<xx> instructs PBUtil to create a list of the top-xx downloaded files. This list is named TOPFILES.A?? and is stored in the textfiles-subdirectory. The number of files to be written in this list is <xx>, (eg. -N15).

Specifying option -F instructs PBUtil to ALWAYS create TOPFILES.ASC/ANS/AVT/AVP, even if DOWNLOAD.LOG is empty or doesn't exist.

Option -R rewrites all file listings with the appropriate file count added to each file. This is useful when you have added some new files.

The file counters will be placed before the description of the files.

Eg. PB_220.ZIP [809] Superb BBS program from Belgium !!!!
^^^^^

File Counter - shows how many times file has been downloaded

You can create a file called NOTOPS.CTL to exclude files from being included when the TOPFILES.A?? is created. The format of this file is just <filename>.<extension> Example as follows:

BLACK.ZIP
NOWAY.EXE
RA_111.ARJ

[FI] File Indexer

When you run PBUTIL FI, ProBoard reads each of your file listings defined in your file areas and creates an index (FILESIDX.PB). ProBoard now only supports indexed FILES.BBS. ProBoard will use the index to find files users specify they want to download. This dramatically speeds up locating files on systems with many different file areas, especially those on CD-ROM drives. Since files that are not in your file index will not be found for users to download, be sure to run PBUTIL FI at lease once each day.

[HF] Hatch Personal File

Personal files can be setup to be sent to users (hatched) from the command line (instead of

using the PROCFG/Personal Files menu interface) using PBUTIL's “Hatch File” (HF) function.

Any personal files you setup (hatch) using the PBUTIL HF method will appear in the PROCFG/Personal File menu interface for later review and administration.

The syntax for PBUTIL HF is:

PBUTIL HF files [files...] /T=ToName /F=FromName /D /C

The following are valid options for PBUTIL HF:

<files> A list of the file(s) to be sent as personal file(s). Wildcards and paths are allowed. If the /C option is not used, the files will NOT be copied to the personal files directory specified in PROCFG Options/Paths. If no path is given, PBUTIL will search in the current directory first. If no files could be found in the current directory, the personal files directory will be searched.

/T=ToName The name of the user to send the files to. Underscores must be used for spaces in names. ProBoard does not check the user file for valid user names to be entered here so be careful when typing user names that you spell them correctly.

/F=FromName The name of the sender of the files. If not specified, the name of the SysOp is used. Again, underscores must be used for spaces. ProBoard does not check the user file for valid user names to be entered here so be careful when typing user names that you spell them correctly.

/D Sets the "KILL" (delete) flag for the hatched file(s). This means that the files will be deleted after they have been downloaded by the user if this is specified on the command line.

/C Copy the specified file(s) to the Personal Files Directory (specified in PROCFG/Options/Paths). Make sure if you use this option, that you also use the /D option in addition, otherwise the files won't be deleted after download from your Personal Files Directory.

Examples:

```
PBUTIL HF C:\PB\PB_220.ZIP /T=Jason Bock /C /D
PBUTIL HF *.TXT /T=John_Doe /F=Joe_Sysop
```

[MI] Message Indexer (Hudson and JAM Only)

Recreates the message base index files (MSGIDX.BBS,MSGTOIDX.BBS and MSGINFO.BBS) for Hudson and (*.JDX) files for JAM areas.

[ML] Message Linker (Hudson Only)

Completely rebuilds reply chains in all areas. This operation has to be performed when echomail is imported into the message base and after using the message packer with the -D parameter.

[MP] Message Packer (Hudson and JAM Only)

Packs the message base. This means that the deleted messages are effectively removed from the message base.

The following are valid options for PBUTIL MP:

(-)
-D ..... Removes old messages. Check the section of the manual about the message areas for more information.
-F ..... Forces the pack to be executed, even if there are no deleted messages (Hudson Only)
-H ..... Instructs the message packer to pack the Hudson areas only.
-J ..... Instructs the message packer to pack the JAM areas only.
-K ..... Deletes the .BAK files created when packing the message base. (Hudson Only!)
-R ..... Instructs the message packer to renumber the message base. Renumbering the message base is done automatically when the highest message number exceeds 25000. Lastread-pointers in the userfile will be adjusted when renumbering.

Do NOT pack the message base when a user is online!!

[MU] Music Player

Use this to play/test any music files you create for ProBoard.

Music files MUST have .MUS as the file extension. The file played when a user “Yells” for the SysOp (paging) is called PAGE.MUS and the “Attention” music file is called ATTEN.MUS

Examples: PBUTIL MU PAGE (plays the paging music file)
PBUTIL MU ATTEN (plays the attention music file)

Note: It is not necessary to include the file extension when using PBUTIL MU to play a music file.

[NC] Nodelist Compiler

Reads a FidoNet-compatible nodelist and creates a ProBoard index- file (NODE_IDX.PRO) in the

system directory. This file will be less than 10K in size, but your original nodelist has to stay in the nodelist directory.

The compiler will look for the latest standard nodelist (NODELIST.xxx) in your nodelist directory.

Specifying extra files as parameters instructs the nodelist compiler to compile extra nodelists. Several extra nodelists can be given as parameters. If you don't specify an extension, ProBoard will look for the latest file with extension .nnn (nnn=day number). If you want to compile a file without an extension, use <FILE>.

Eg. PBUTIL NC MYNL MYLIST. FDNET.PVT

This would compile the latest NODELIST.nnn, the latest MYNL.nnn, MYLIST and FDNET.PVT in the standard nodelist directory.

To determine the costs of sending Netmail, a textfile will be read that you will have to create. This textfile is called COST.PRO and should be in ProBoard's system directory. The lines in this file have the following format:

<Command> <Param1> [Param2]

Commands:

MYZONE <zone>	All commands after this command act on zone <zone>. You need at least 1 MYZONE command. If not, ProBoard will assume you are in zone 2. This is used to specify your own zone.
DEFAULT <cost>	Defines the default Netmail cost.
ZONE <zone> <cost>	Defines the Netmail cost for zone <zone>.
REGION <region> <cost>	Defines the Netmail cost for region <region> within your own zone.
NET <net> <cost>	Defines the Netmail cost for net <net> within your own zone.

A simple example for a node in Belgium, where the BBS is part of only ONE network.

```
MYZONE   2       I'm in zone 2
DEFAULT  100     Default = 100 credits
ZONE     3 50    Zone 3 = 50 credits (Australia)
ZONE     1 40    Zone 1 = 40 credits (North-America)
REGION  28 10    Region 28 = 10 credits (Netherlands)
REGION  29  0    Region 29 = FREE (Belgium)
NET    512  5    Net 512 = 5 (HCC Netherlands)
```

We give another example for a node which is part of 2 networks, so this node has nodenumbers 2:292/1900 and 89:120/40

```
DEFAULT   100    Default = 100 credits
ZONE     1  50   Zone 1  =  50 credits
ZONE     2  20   Zone 2  =  20 credits
```

```
ZONE     3  70     Zone 4  =  70 credits
ZONE    89  10     My private network = 10 credits
MYZONE   2         Following definitions are for zone 2
REGION  29   1     Region 29 in zone 2  = 1 credit
NET    292   0     Net 292   in zone 2  = 0 credits
NET    512   5     Net 512   in zone 2  = 5 credits
MYZONE  89         Following definitions are for zone 89
REGION  12  2      Region 12 in zone 89 = 2 credits
NET    120  0      Net 120   in zone 89 = 0 credits
```

[UF] UserFile Fixer

Run this after you use a user file packer/sorter from another source (like RAUSER or RACE). It assures that the extensions in ProBoard's user file are updated properly.

[UP] UserFile Packer

Removes all deleted users from USERS.BBS, except the users with the NoKill flag set. Corrupted user records are also removed.

Specifying option -R instructs the user packer to reset all Last Read pointers to zero.

Specifying option -K will delete the .BAK file created when packing the user file.

Do NOT pack the userfile when a user is online!!

[US] UserFile Sorter

Sorts the user file by security level and by first name. Also updates the lastread pointers for the new sorted user file as well as reindexing the user file.

REFERENCE

Multi-user Operation

Since ProBoard is a multi-line BBS package, it allows you the ability to have more than one user use the userfile and the message base at the same time. ProBoard doesn't do any internal multi-tasking to make the program act as flexible as possible.

This makes sure that you can set up a multi-line BBS via a network and multiple computers, or by running ProBoard under a multitasking system. ProBoard is also DESQview aware.

ProBoard can handle up to 255 nodes. Each node needs its own directory, because ProBoard supports the use of many different external message editors and most of these editors are designed to be used on single-node systems. One editor that does not need such a configuration is GEDIT. With GEDIT, you simply place one copy of it in your ProBoard startup directory (or any other directory of your choice) and then specify where the editor is located in PROCFG, [F1], Paths, under the “Editor Command” option. A few other newer style editors also support multi-line use without needing a copy in each nodes directory. To find out more, ask one of the support BBS's.

Each node MUST be started from its own private directory.

No ProBoard-related files have to be placed in this directory but certain doors and certain message editors may need this directory for their own use.

If you want to install 3 nodes in a network, you could create the following structure:

C:\PB\MSGBASE	Message base directory
C:\PB\TXTFILES	Textfiles directory
C:\PB\MENUS	Menus directory
C:\PB\PEX	PEX files directory
C:\PB\NODE1	Start-up directory for node 1
C:\PB\NODE2	Start-up directory for node 2
C:\PB\NODE3	Start-up directory for node 3
C:\PB\NODE4 ... 255	Start-up directory for node 4 - 255

ProBoard should be started from C:\PB\NODE1 for node 1, from C:\PB\NODE2 for node 2, etc. These directories may require the files for the external editor. All the other files that ProBoard uses, should be stored in ProBoard's system directory (usually C:\PB).

If you run ProBoard under a multitasker like DESQview, the different nodes MUST be run on different com-ports. It should be noted that the new DESQview v2.6 supports networks like Novell Lite and Lantastic directly, making it possible to run the BBS on a machine that is also a server. Refer to your DESQview v.26 manual/setup program for more details.

IMPORTANT	If you are running ProBoard multi-line using a multitasker, the DOS program SHARE.EXE MUST! Be installed.

SysOp Keys

While the user is on-line, the SysOp can perform several actions, by using the SysOp keys:

<Ctrl-Left-Arrow> <Ctrl-Right-Arrow>	Lower or Raise the security level of the user who is currently online. Only the levels you previously configured in PROCFG can be selected.
Up/Down	Raise/Lower the current user's time left. The time subtracted/added is not restricted to this session!
Alt-C [Chat]	Start a chat with the user. The chat may be ended by pressing <Esc>.
Alt-J [Jump]	Jump to DOS. If 'Swapping' was set to 'Yes' in PROCFG, the ProBoard session will be swapped to disk or EMS, thus making all memory available to the DOS commands you want to execute. You can return from the shell by entering EXIT at the DOS prompt.
Alt-H [HangUp]	Hangs up the phone, throwing the user off-line immediately (very unfriendly)!
Alt-L [LockOut]	Hangs up the phone AND sets the user's level to 0, thus making sure he/she cannot log in any more (very very unfriendly)!
Alt-N [SysOp Next]	Creates a semaphore file called SysOpNXT.SEM in the ProBoard system directory. Also will play the music file SysOpNXT.MUS when user logs off so you are aware your BBS is free.
Alt-E [Edit]	Allows you to edit the user online. The editor is similar to the editor in ProCFG.
Alt-I [Image]	Appends an image of the screen to the file IMAGE.TXT in the ProBoard system directory.
Alt-R [Reset]	Resets the chat request status. This will stop the flashing of the status line.
Alt-S [Static]	What can I say. Useful in combination with the ALT-H to help you, the friendly SysOp, free up your board when needed.

Alt-Y [No Chat]	Makes ProBoard not sound the page music if the user online when you press this, pages you. Comes in very handy when you see that user who ALWAYS wants to chat is logging on to your BBS.
Shift-F1	Shows a help-screen with all the SysOp macros available. (registered version only)
F1	Shows a help-screen with all the SysOp keys available.
PgUp/PgDn	Shows additional information about the current user on the status line. You can also display a particular status by pressing F2-F10.
Home	Shows the normal status line after using PgUp/PgDn.
F2	Shows the user's name, level, time left and time online.
F3	Shows the user's handle and flags.
F4	Shows the user's city and phone numbers.
F5	Shows statistics about the current user.
F6	Shows the user's comment line.
F7	Shows system information.
F8	Shows the chat reason if the user tried to page the SysOp earlier.
F9	Shows the name and login time of the last caller.
F10	Shows the time used and Kbytes downloaded today. It also displays the date of birth of the current user.

Furthermore, there are 10 programmable hotkeys (SysOp macros). They can be configured in ProCFG using the F10 option.

Command line parameters & Errorlevels

PROBOARD.EXE accepts following command line parameters:

PROBOARD [-B<baud>] [-P<port>] [-N<node>] [-L<level]
[-T<time>] [-S] [-Q] [-X] [-V<mode>]

These parameters have the following meaning:

(-)
-B<baud> Specifies the baud rate. You can specify the following baud rates: 300, 1200/75 2400, 4800, 7200,

9600, 12000, 14400, 16800, 19200, 21600, 24000,
26400, 28800, 38400, 57600, 64000, and 115200.

-P<port> Specifies the com-port (1-8).

-N<node> This node's node number (1-255).

-L<level> Prevents people with a security level lower than this level to access this node. Great for only allowing subscribing user access to one or more nodes.

-T<time> <time> is the time until the next event in minutes. Useful with FrontDoor's DOBBS batch file.

-S Start ProBoard in stand alone mode. ProBoard will use as a default the PORT and the BAUD specified in your configuration, but you can override them by using the -P<port> and the -B<baud> command line parameters along with the -S parameter.

-Q Quiet mode. Great if your BBS is in a location such as your bedroom etc. The only noise ProBoard will make is playing the ATTEN.MUS file (if a user who has this set in their user record logs on), and playing the PAGE tune when a user pages during valid paging hours.

-X Tells ProBoard not to use EMS.

-V<mode> Runs ProBoard in video mode <mode>. This can be useful if you have a monitor capable of displaying 132 columns, and you would like to run ProBoard in such a mode (you will see an extra information window on your screen if ProBoard is run in 132 cols mode). The <mode> parameter is a decimal number specifying the video mode, as it is used with INT 10H, function 00. (only programmers will understand this though :-). This is different for every video card.

When no '-B<x>' and no '-S' parameter is given, ProBoard will be started in local mode.

ProBoard returns an errorlevel when a user has logged off.

The errorlevels of PROBOARD.EXE are:

0 Everything OK, normal logoff.
1 FATAL error, something 'terrible' happened, or the modem could not be initialized.
2 Not used.
3 Netmail entered by the user.
4 Echomail entered by the user.
5 Echomail AND Netmail entered by the user.
99 SysOp pressed <Esc> at the "Waiting for call" screen.
100 ProBoard was requested to go down by a semaphore file.

AVATAR/0 and AVATAR/0+ Terminal Emulation

ProBoard now supports AVATAR terminal emulation. It allows users to select either AVATAR/0 or AVATAR/0+, providing that the terminal package they use to call your BBS is compatible with either AVATAR/0 and/or AVATAR/0+. Examples of terminal software packages that support AVATAR/0+ are, FrontDoor 2.02 and TinyTerm. QModem and Telix also support AVATAR, but only in the AVATAR/0 mode. AVATAR has the advantage over ANSI of being much faster.

A word about the files extensions (.A??) that are used for AVATAR screens. There are two file extensions that are associated with AVATAR text files that you will display to your users. They are .AVT and .AVP. When displaying (.A??) textfiles to your users, ProBoard looks for them in the following order.

1.) .AVP
2.) .AVT
3.) .ANS
4.) .ASC

Refer to the following section on “Hard Coded .A?? files” for more information on the default .A?? files that ProBoard will display to your users.

To create files with the .AVT and .AVP file extensions you will need a file utility called AVTCONV.EXE. This file was distributed with RemoteAccess, and is also available from most of the ProBoard beta sites, as well as any of the ProBoard Support BBS's. It is very simple to use this utility. Just copy it to your ProBoard TEXTFILE directory and run the following commands:

AVTCONV *.ANS AVT -- this will create a copy of all of your ANSI (.ANS) screens with the extension .AVT These are the textfiles for displaying to users that have AVATAR/0 terminal emulation selected.

AVTCONV /C+ *.ANS AVP -- this will create a copy of all of your ANSI (.ANS) screens with the extension .AVP These are the textfiles for displaying to users that have AVATAR/0+ terminal emulation selected.

Hard-coded .A?? files

In certain situations, ProBoard will display default .A?? files.

Just what is an A?? file?

All .A?? files are files that have the file extensions of .AVT, .AVP, .ANS, or .ASC. If the .ANS/AVT/AVP file cannot be found, the .ASC file will be displayed (if it exists).

Note: You have to insert your own “Press [Enter] to continue” prompt if necessary!

AFTERPW.A?? Displayed to a user after they have entered their password

BEFOREPW.A?? Displayed to a user right before ProBoard asks them for their password.

BIRTHDAY.A?? If a user logs in on his/her birthday, this file will be shown after the news file.
Note: you can run a pex-file to congratulate a user on his/her birthday.

DLDELAY.A?? Displayed to a user who has to wait <n> minutes before downloading. Refer to Configuration - Time/DownLoad Limits (F5) for more information on setting the number of minutes for the DownLoad Delay.

DLHANGUP.A?? Displayed to a user who has selected [G]oodbye after download, after the 10 second timer to abort the hangup, has expired.

DUPESFND.A?? Displayed to a user who uploads one or more duplicate files to your BBS if "Check Duplicate Uploads" is set to "Yes" in PROCFG's File Transfer Options. The file will be displayed to the user once the duplicate check is completed if any duplicate files are found.

EVENTDUE.A?? Shown if a user can't login because of an event that has to run soon.

EXPIRED.A?? When a user's level has expired, this file is shown.

EXP_WARN.A?? Displayed when the user's level expires within less than 30 days.

GOODBYE.A?? Displayed when the user is logging off, just before the carrier is dropped.

INTRO.A?? Displayed when a user logs in, BEFORE asking a user's name and password.

INTRO<n>.A?? This is the same as INTRO.ASC/ANS except that this file that can be displayed to the specified node number. For Example: to display it to node 2 you would create a file called INTRO2.A??. If no INTRO<n>.A?? exists, ProBoard will then show the file INTRO.A?? or LOGO.A??

MAXPAGE.A?? Displayed when the user has tried to page the SysOp too many times.

MSGHELP.A?? Replaces the built-in message reading help if this file exists.

NEWS.A?? Displayed AFTER the user has read his/her new mail.

NEWUSER.A?? Displayed to user before ProBoard asks "Are you a New User?". When the user enters their name and ProBoard does not find their name in the user file, ProBoard asks "Did you enter your name correctly?" If the user indicates they did, this file (if you create it) will display to the user.

NEWUSER1.A?? Displayed when a new user is logging in, BEFORE he/she has started the questionnaire.

NEWUSER2.A?? Displayed when a new user is logging in, AFTER he/she has completed the questionnaire.

NOACCESS.A?? Displayed when user is not allowed to log onto one or more lines because their access level is lower than the one specified in the command line PROBOARD -L<level> (used to start ProBoard on that node). Read the "Reference Section" of this manual, under "Command Line Options & Errorlevels" for more information on this command line.

NOTAVAIL.A?? Displayed when the user tries to page outside paging hours

PAGED.A?? Displayed when the SysOp does not respond when the user tries to page.

PRIVATE.A?? Is displayed when your system is configured as a private system, and a new user tries to log in.

SECxx.A?? 'xx' stands for a userlevel. If a user with level xx logs in, this file will be displayed (eg. SEC25.ANS). The file is shown after all WELCOME<x> files and before the mailcheck.

TRASHCAN.A?? Shown when a user tries to use a name listed in TRASHCAN.CTL file.

WELCOMEx.A?? Displayed at login, after the user entered his/her name. The 'x' stands for a digit ranging from 1 to 9; these files (if they exist) will be displayed in ascending order, 1-2-...-9, BEFORE a mailcheck is done.

.A?? File Control Codes

You can use several codes in ProBoard's textfiles. These codes will be replaced by internal variables, or will perform special actions.

In the code list, you will see a code's ASCII value, the control code and a description of the code.

A '^' means Ctrl, so ^D means Ctrl-D.

ASCII	CODE	DESCRIPTION
65	^A	Waits for the user to press <Enter>.
66	^B	Disables interruption by pressing <S>.
67	^C	Enables interruption by pressing <S>.
68	^D	Enables 'More'-prompt.
69	^E	Disables 'More'-prompt.
71	^G	Rings a bell on the user's computer.
76	^L	Clearscreen.
87	^W	Pauses for 1 second.

ASCII	CODE	DESCRIPTION
06-65	^FA	User's full name.
06-66	^FB	User's City.
06-67	^FC	User's password.
06-68	^FD	User's data/fax phone number.
06-69	^FE	User's phone number.
06-71	^FG	Time of last login.
06-76	^FL	Netmail credit left.
06-77	^FM	Number of messages written.
06-78	^FN	Message last read.
06-79	^FO	User's level.
06-80	^FP	Number of calls by user.
06-81	^FQ	Number of uploads by user.
06-82	^FR	Kbytes uploaded by user.
06-83	^FS	Number of downloads by user.
06-84	^FT	Kbytes downloaded by user.
06-85	^FU	Number of minutes online today.
06-86	^FV	User's screen length.
06-87	^FW	User's first name.
06-88	^FX	ANSI codes ON/OFF.
06-89	^FY	Screen pausing ON/OFF.
06-90	^FZ	Clearscreen codes ON/OFF.
06-48	^F0	Fullscreen editor ON/OFF.
06-49	^F1	User's Alias.
06-50	^F2	Command stacking ON/OFF.
06-51	^F3	IBM Characters ON/OFF.
06-52	^F4	User's State.
06-53	^F5	User's birth date.
06-54	^F6	User's expiration date (if any)
06-55	^F7	Day's until expiration date.
06-56	^F8	AVATAR/0 - on/off.
06-57	^F9	AVATAR/0+ - on/off.
06-40	^F"	User's Country.
06-58	^F:	User's first call (date).
06-36	^F$	User's address (line 1).
06-37	^F'	User's address (line 2).
06-38	^F&	User's address (line 3).
06-60	^F`	User's sex.
06-91	^F[	Download Kb left today.
06-126	^F~	Download delay in minutes.
06-33	^F!	# Minutes remaining until allowed to download.
06-40	^F(	Current file group name.
06-41	^F)	Current message group name.
06-42	^F*	Current file group number.
06-43	^F+	Current message group number.

06-61	^F=	User's fax number
06-59	^F;	Show password, or password character is passwords hidden

ASCII	CODE	DESCRIPTION
11-65	^KA	Total number of calls to the BBS.
11-66	^KB	Name of the last user on the BBS.
11-67	^KC	Number of active Hudson message base messages.
11-68	^KD	Number of first message.
11-69	^KE	Number of last message.
11-70	^KF	Number of times user has paged the SysOp.
11-71	^KG	Day of the week (full).
11-72	^KH	Number of users on the BBS.
11-73	^KI	Current time.
11-74	^KJ	Today's date.
11-75	^KK	Minutes online during this session.
11-77	^KM	Minutes online today.
11-79	^KO	Minutes online left today.
11-80	^KP	Version number of ProBoard (x.xx)
11-81	^KQ	Daily online limit.
11-82	^KR	Baud rate.
11-83	^KS	Day of the week (short).
11-84	^KT	Daily download limit (Kbytes).
11-87	^KW	Node number.
11-88	^KX	Hang up phone.
11-89	^KY	Active message area name.
11-90	^KZ	Active file area name.
11-48	^K0	# Messages in active message area
11-49	^K1	Current message area #
11-50	^K2	Current file area #

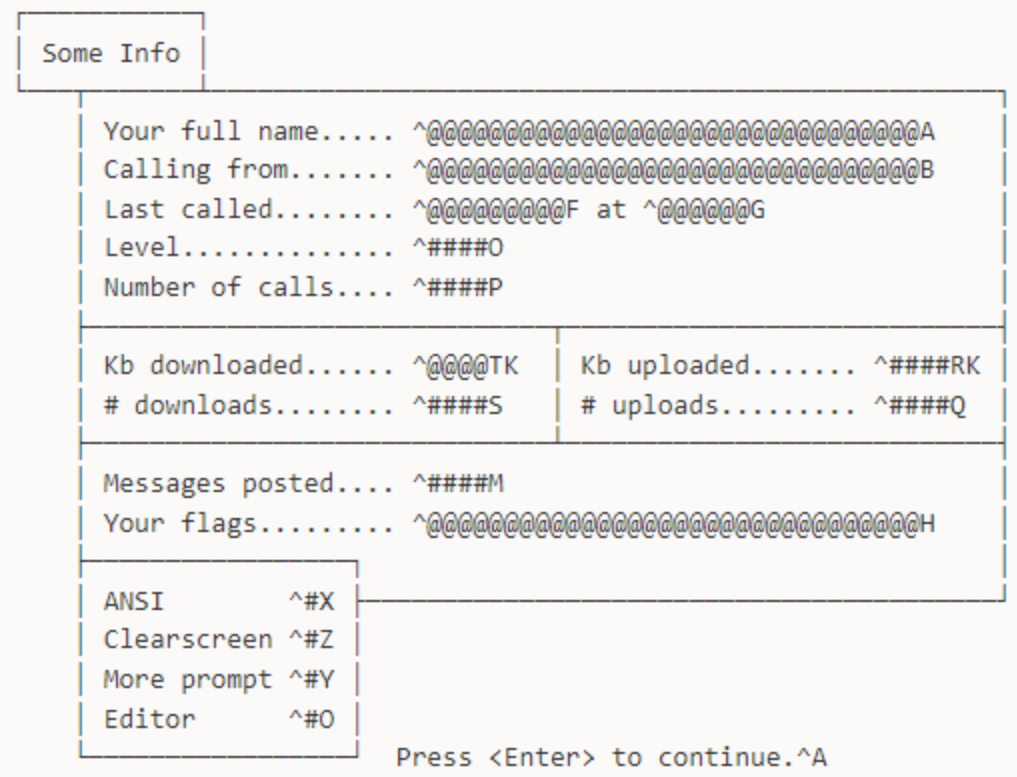
You can also inform ProBoard about the length of a string to be placed in a textfile. This is done is the following way:

Between the first and last code, you can place '@' or '#' or '%' codes.
The field's length will be defined by the number of characters, first and
last control code included. Use '@' to align (justify) a field to the
left, use '#' to align to the right and use '%' to center a field.

```
  Eg. ^K@@@@@@@@@@@@@@@@@@@@@@@@@B
        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
                                  23 X '@'
```

Thus, the total amount of characters equals 25, the user's name (^KB) will be placed in a 25-character field, left justified. You now can easily draw 'boxes' around this field, without having to worry about the actual length of the user's name.

An example of a textfile using the control codes: (the '^' stands for '^F') :



Shell Menu (Option 7) Special Codes

(*)

** Replaced by an asterisk ('*')

*# Replaced by the node number.

*\ Sends the message "SysOp is shelling..." to the user before the shell is executed, and the message "SysOp has returned..." afterwards.

*! Freezes ProBoard's system timer when shelling.

*= Do not call any fossil functions when shelling. Great for use in INIT.PEX files.

*A Writes a user's handle (alias) to DORINFOx.DEF instead of his/her real name.

*B Current baud rate.

*C Replaced by the full name & path of the command interpreter. It is the contents of the environment variable "COMSPEC". This usually is C:\COMMAND.COM.

*D Writes a full 52-line DOOR.SYS drop file to the current directory before shelling.

*E Writes an RA 1.1x EXITINFO.BBS to the current directory before shelling and reads it back afterwards. ProBoard creates an RA 2.xx by default so if you need the older EXITINFO.BBS format be sure to specify the *E Parameter.

*F User's first name.

*G Indicates whether user has ANSI (1) or ASCII (0) set.

*H Tells ProBoard not to disable the fossil driver when shelling.

*I Maximum user-inactivity (seconds).

*J Displays configured BBS name.

*K Displays current version of ProBoard.

*L User's last name.
*M ProBoard's start-up directory (including trailing '\')
*N Shell will NOT be logged in PROBOARD.LOG.
*O (not zero!) is replaced by the path of the current file area.
*P Com-port used by ProBoard (1-8).
*Q Don't let user know that ProBoard is shelling (!).
*R User's record number in USERS.BBS.
*S ProBoard's system directory (incl. trailing '\')
*T Time left for the user today (minutes).
*U Displays current user's handle (alias).
*V Disables adding 2 to the graphics capability line in DORINFOx.DEF when the user has AVATAR enabled.
*W Runs the shell in a window , so the status on the last line will not be cleared. This option only works with programs that send their output to the standard output device (no direct screen writes).
*X ALWAYS SWAP to disk/EMS, even if swapping is disabled in ProCFG.
*Y DO NOT SWAP to disk/EMS.
*Z Execute the command as if you typed it from the command line. The main use for this option is to execute DOS batch files. It is exactly the same as entering "*C /C <command>". It can also be useful to execute programs that are located in your path, without having to specify the exact directory name.
*_ (asterisk underscore) Tells ProBoard not to write underscores instead of spaces when writing a user's last name in DORINFOx.DEF. For example, users name is Harvey Van Hooten. Without this parameter, users name will be written to DORINFOx.DEF as follows:

HARVEY VAN_HOOTEN

By using the *_ (asterisk underscore) the users name will be written to DORINFOx.DEF as follows:

HARVEY VAN HOOTEN (with no underscore in last name)

*0 (zero) ProBoard will write a DORINFO1.DEF instead of a DORINFO<node>.DEF - great for running doors which require a DORINFO1.DEF file on a multi node ProBoard system.
*1 Installs a timer-function when shelling, which continuously redisplays the user's status line on the first screen line. This can be used with ALL programs. (Works great most full- screen protocols!)
*2 Same as *1, but uses the bottom screenline (try this with QuickEd or GEdit!)

Suppose ProBoard is in directory D:\PB.

Music Files

You can make your paging-sound and attention-sound more attractive by defining your own music file. A music file is a text file in which you can use 2 keywords:

TONE [hz] [1/100's sec]	Sounds a tone of [hz] Hz during the specified period.

WAIT [1/100's sec] Waits for the specified period.

The format of this textfile is compatible with the RemoteAccess music files.

The paging-musicfile is named PAGE.MUS, and the attention-music- file is named ATTEN.MUS.

You can use this frequency-table to write your own masterpiece:

	0.1	0.2	0.3	0.4	0.5	0.6
C	45	134	268	536	1071	2145
C#	71	142	284	568	1136	2273
D	75	150	301	602	1204	2408
D#	80	159	319	638	1275	2551
E	84	169	338	676	1351	2703
F	90	179	358	716	1432	2864
F#	95	190	379	758	1517	3034
G	100	201	402	804	1607	3215
G#	106	213	426	851	1703	3406
A	113	225	451	902	1804	3608
A#	119	239	478	956	1991	3823
B	127	253	506	1012	2025	4050

Text Macros

You can enter text macros in any user-definable string. Text macros are defined as “@<NAME:n>@”, and are replaced by internal ProBoard variables at runtime. You can insert these macros in the following strings:

- Textlines in menu items

Data fields in menu items
Menu prompts
Modem initialization string

'NAME' is the name of the macro. A list of available macros will follow.

You can also specify the field width for the resulting string. This is done by appending the macro name with a ':' and a number. A positive number will result in a left aligned string, while a negative number will result in a right aligned string.

For example:

@<NAME:30>@ will display the user's name in a left aligned field of 30 characters wide.

Note that the field width specifier is optional.

Here is a list of the available macros:

BAUD	Current baud rate
CITY	The city of the current user
COUNTRY	User's country/state
CURFILEAREA	Current file area name
CURFILEAREA#	Current file area number
CURFILEAREADIR	Current file area path
CURFILEGROUP	Current file group name
CURFILEGROUP#	Current file group number
CURMENU	Current menu name
CURMSGAREA	Current message area name
CURMSGAREA#	Current message area number
CURMSGGROUP	Current message group name
CURMSGGROUP#	Current message group number
DATAPHONE	User's data phone number
DATE	Current date (xx-xxx-xx)
FIRSTNAME	The first name of the current user
HANDLE	The user's handle (fixed alias)
HIGHMSG	Highest message number in message base
ID	The user level ID for this user
LANGUAGE	The language of the user (base file name)
LASTDATE	Date of user's last call
LASTTIME	Time of user's last call
LEVEL	The level of the current user
LOWMSG	Lowest message number in message base (Hudson)
MNUDIR	Menu directory
MSGDIR	Message base directory
NAME	The name of the current user
NLDIR	Nodelist directory
NODE	Current node number
NUMMSG	Number of messages in current area
NUMUSERS	Total number of users in userfile
NUMYELLS	The number of times the user yelled
PASSWORD	The user's password
PEXDIR	PEX-files directory
PORT	Current com-port nr (1-8)
PVTDIR	Private uploads directory
STARTDIR	Startup-directory
SYSDIR	ProBoard system directory
SYSOPNAME	Name of the SysOp
TIME	Current time (xx:xx)
TMLEFT	Number of minutes left
TMONLINE	Number of minutes online
TOTALCALLS	Total # of calls to the system

TOTALMSG	Total # of messages in message base (Hudson)
TXTDIR	Textfiles directory
UPDIR	Upload directory
USERREC	Record number of this user's user record
VERSION	ProBoard version number (eg. 2.20)
VOICEPHONE	User's voice phone number
%NAME	Will be replaced by the contents of DOS environment variable 'NAME'. Example: in DOS or a .BAT file you would put SET INITSTR=ATDT5551212. Then when you use a text macro like this: @<%INITSTR>@ it would be replaced with "ATDT5551212". Note the "%" symbol in the macro.

Other special characters are:

#1 - #7	Change color (only valid for displayable strings).
#B1 - #B7	If a 'B' is in front of the color number, the "blink" attribute will be turned on.

1 = Red
2 = Green
3 = Yellow
4 = Blue
5 = Magenta
6 = Cyan
7 = White

Example:

Hi @<#3>@@<FIRSTNAME>@@<#7>@, how is the weather in @<CITY>@?—

Other single examples: @<NAME:35>@
@<TMONLINE:-5>@
@<CITY>@

Example Batch Files

This is an example for standalone-operation with 2 errorlevel events defined. Event 1 returns level 10, event 2 returns level 20. No echomail and netmail is used.

```
:Loop
PROBOARD -S
if errorlevel 100 goto End
if errorlevel 99 goto End
if errorlevel 20 goto Event2
if errorlevel 10 goto Event1
```

```
if errorlevel 1 goto FatalError
goto Loop
:Event1
echo Perform some actions
echo --------------------
goto Loop
:Event2
echo Perform some more actions
echo ------------------------
goto Loop
:FatalError
echo FATAL ERROR - ProBoard Down
:End
```

Using ProBoard with a frontend-mailer is a little more complicated. This is an example for use with FrontDoor.

In this example FrontDoor uses the following errorlevels:

```
300 bps call : 50
1200 bps call : 51
2400 bps call : 52
Local call    : 60
Mail received : 100
Exit          : 150
```

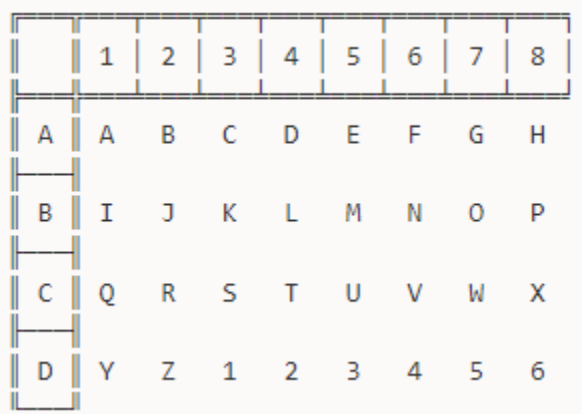
```
SET FD=C:\FD
:Loop
cd \fd
fd
if errorlevel 150 goto End
if errorlevel 100 goto MailRcv
cd\pb
if errorlevel  60 goto Local
if errorlevel  52 goto Call2400
if errorlevel  51 goto Call1200
if errorlevel  50 goto Call300
:FatalErr
echo FATAL ERROR
goto End
:Call300
proboard -b300 -p1
goto CheckPBErr
:Call1200
proboard -b1200 -p1
```

```
goto CheckPBErr
:Call2400
proboard -b2400 -p1
goto CheckPBErr
:Local
proboard
:CheckPBErr
if errorlevel 100 goto End
if errorlevel 5 goto NetEcho
if errorlevel 4 goto EchoEntered
if errorlevel 3 goto NetEntered
if errorlevel 1 goto FatalErr
goto Loop
:NetEcho
REM ************************************
REM ** Export netmail & echomail here **
REM ************************************
Goto Loop
:NetEntered
REM ************************************
REM **      Export netmail here       **
REM ************************************
Goto Loop
:EchoEntered
REM ************************************
REM **      Export echomail here      **
REM ************************************
Goto Loop
:MailRcv
REM ************************************
REM **  Unpack and Import mail here   **
REM ************************************
Goto Loop
:End
```

Flag Cross-Reference Chart

The following cross-reference chart will help you to easily determine which flags to use for your user records and menus using the ProBoard style (A-Z and 1-6) of flags.

Note: The flag structures in ProBoard are actually no different than those in QuickBBS or RA, they are just expressed in a more “user friendly” format.



	1	2	3	4	5	6	7	8
A	A	B	C	D	E	F	G	H
B	I	J	K	L	M	N	O	P
C	Q	R	S	T	U	V	W	X
D	Y	Z	1	2	3	4	5	6

A few flag examples follow:

QuickBBS/RA	ProBoard
A1	A
C3	S
D7	5
C8	X

well…. you get the idea.

FREE Files

Many SysOps have a master file list containing all the files available on their BBS for users to download, or perhaps some virus detection software that they would like to make available to their users without affecting their users download ratio. Other SysOps may run a support BBS and have certain files they want to make available to callers without regards to how long it takes to download. ProBoard's powerful “Free File” option(s) can easily accommodate both of these things.

It is possible (in addition to marking entire file areas) to specify a list of files that users can download without ProBoard deducting the kilobyte amount of the download from their daily limit. This would allow you to make one or two (or several) files in a file area as “Free” without making the entire area “Free” to your users. To accomplish this, create a file called FREEFILE.CTL in the ProBoard system directory (usually C:\PB). Create this file with an ascii editor such as QEDIT or DOS editor. Each line in the file should contain the name (do not include the drive and path to the file) and extension of any files you want to make available for FREE. Wildcards are allowed.

An example follows:

```
PB_220.ZIP
FILELIST.ZIP
*.TXT
```

SCANV86.ZIP
PB_STRUC.*
LIST.LZH

It is also possible to specify individual files that you want to make available to your callers with no regard for the time it takes a user to download them. This means that if you have a file that is 1024k, and the user has 1 minute left, the user would still be able to download the file. This is a valuable feature for SysOps who run a support BBS and want to make individual files available to users without getting hate mail from those users who call long distance and then don't have enough time left or a high enough security level to download support files. Refer to Menu Function 32 for more details.

Semaphore files for shutting down ProBoard

It is possible to shut down ProBoard by creating a semaphore file in the ProBoard system directory. The file should be called:

DOWN.<node#> (node# is the node number)

When ProBoard finds this file, it will hang up and immediately exit with errorlevel 100. It will then create a file called ISDOWN.<node#> in the same directory to indicate that ProBoard has been shut down successfully.

TIPS & TRICKS

Navigating Through Menus

ProBoard has been created for the novice and experienced user. It supports Opus-like commands and RA/QBBS style hotkeys. Using hotkeys is the easiest way to navigate through the menus. While a menu is being displayed, you can enter any menu command. The displaying of the menu will be stopped, and the corresponding menu function will be executed immediately.

The other way to enter menu options is command stacking. This way you can enter several menu commands at a prompt, and execute them all at once by pressing [Enter]. You are not limited to menu commands only. You can enter any key that should be “stacked”. A ';' stands for <Enter>.

This asks for an example I guess. Suppose you are in a menu where option [E] selects the “Enter Message” function. Now, if you want to write a private message to the SysOp, using “Test” as a subject, you could enter:

ESysOp;Test;Y

E	"Enter Message"
SysOp	"Write message to" prompt
;	Enter
Test	"Subject" prompt
;	Enter
Y	Answer to "Private [Y/N]" prompt

If you don't like to use command stacking all the time, but want to use it occasionally, you can enter a ';' at any menu prompt, and you will be able to enter a “command stack”.

The [S] & [P] key

You can stop almost any incoming text with the [S] key, and pause with [P] key. To resume a paused text, press [P] again or press the <Enter> key. The SysOp can disable this though.

Online help in each menu

You can create help files for each of your menus by doing the following:

Create a menu item in your global menu (GLOBAL.PBM) with the hotkey '?', function 5 (Show ASC/ANS file), data “@<CurMenu>@”. Now when a user hits '?' in any menu, the file <MENUNAME>.A?? will be displayed to the user.

Uploading files to the current file area

Normally, uploads are sent to the predefined upload directory (set in ProCFG/Options/Paths). You can override this in the upload function by entering a path in the data field. You can use text

macros to send uploads to the CURRENT file area by entering @<CurFileAreaDir>@ in the data field.

SOFTWARE DEVELOPMENT KIT

Software Development Kit (SDK)

Included with ProBoard is the ProBoard Software Development Kit (SDK). It consists of the files PB_SDK.H, PB_SDK.OBJ and PB_SDK.LIB.

The SDK allows C and C++ programmers to write extensions to ProBoard. The ProBoard Software Development Kit is bundled with ProBoard as free software. This means that you may use it to write extensions for ProBoard.

The extensions you can write for ProBoard are called “PEX” (ProBoard Executable) files. You may distribute any ProBoard PEX files you write royalty free.

Any PEX files you create with the SDK are run from within ProBoard using Menu Function 60 (Run ProBoard SDK file).

For more information on using the ProBoard Software Development Kit (SDK) and the functions that it contains, please refer to the file included with this release of ProBoard, called “PB220SDK.DOC”.

Requirements

To create your own programs you need:

- An ANSI compatible C or C++ compiler. Compilers known to be compatible with the ProBoard SDK are:
  - Zortech C/C++ 2.0 - 3.0
  - Microsoft C/C++ 5.1 - 7.0
  - Borland C++ 2.0 - 4.5
  - Turbo C++ 1.0
  - Turbo C 1.0 - 2.0
  - Other compilers may work too, but were not tested.

- An MS-compatible linker (usually the one that came with your compiler).

- Linkers known to work with the ProBoard SDK are:
  - All Microsoft Linkers (LINK.EXE)
  - Borland's Turbo Link (TLINK)
  - ortech's BLINK

- The ProBoard SDK include file PB_SDK.H

- The ProBoard SDK library PB_SDK.LIB (for non-Borland compilers)

- The ProBoard SDK file PB_SDK.OBJ (for Borland 3.x compilers)

Creating ProBoard Executables (PEX-files)

To create your own ProBoard executable (.PEX file), you first create a source file, say MYPROG.C. When ProBoard loads a PEX file, it executes the function main() in your program. This function is defined as:

```
void main(int argc,char *argv[])
{
    ...
}
```

The parameters 'argc' & 'argv' contain optional data that can be given to your program at load time (exactly like C's argc and argv parameters). Creating this main() function is the only thing you need to do to write a valid ProBoard executable. From this point on, it's just like writing a regular C program. Most of the ANSI standard library functions are available, plus many ProBoard- specific functions and global variables, used for interfacing between your program and the ProBoard environment. As in a regular executable, you can build your program from several source files.

This is a tiny sample ProBoard program:

```
#include "pb_sdk.h"
void main()
{
    printf( "You have %d minutes left.\n", TimeLeft() );
    AddTime( 10 );
    printf( "Now you have %d minutes left.\n", TimeLeft() );
}
```

Included with ProBoard are a few example programs (in source form of course).

Compiling and Linking

ProBoard executables have to be compiled using the LARGE memory model.

For example (Turbo C++) : tcc -c -ml myprog.c

Once you have your .OBJ file (in this case MYPROG.OBJ), you have to link it with the ProBoard SDK library (PB_SDK.LIB). You have to specify a filename as output filename. The linked executable must have extension .PEX. It is important to disable linking of the default compiler libraries!! It is also required to turn on case sensitivity.

e.g. Turbo C++ : TLINK MYPROG,MYPROG.PEX,,PB_SDK /N /C
Microsoft : LINK MYPROG,MYPROG.PEX,,PB_SDK /NOD /NOI;
Zortech : BLINK MYPROG,MYPROG.PEX,,PB_SDK /NOD /NOI;

That's all there is to it. Now you can run your program through ProBoard's menu function 60.

I will now give the compiler syntax for some popular compilers:

```
Zortech C/C++ : ztc -c -a -ml myprog.c
Microsoft C   : cl -Zep -Gs -c -AL myprog.c
Turbo C/C++   : tcc -c -ml myprog.c
Borland C++   : bcc -c -ml myprog.c
```

And linker syntax:

```
Borland     : TLINK PB_SDK MYPROG,MYPROG.PEX /N /C
Microsoft   : LINK MYPROG,MYPROG.PEX,,PB_SDK /NOD /NOI;
Zortech     : BLINK MYPROG,MYPROG.PEX,,PB_SDK /NOD /NOI;
```

Note that the syntax for Borland C++ has changed from previous versions of the SDK. It is required that you specify PB_SDK.OBJ BEFORE your object files, and don't link PB_SDK.LIB.

Restrictions

There are few restrictions in writing ProBoard executables. For C programs there are actually only two major restriction: long math & floating point operations.

Due to the fact that all compilers generate function calls to multiply, divide and shift long integers, and that every compiler vendor uses its own array of functions, it is impossible to include these functions in the ProBoard SDK. As a result, it is not directly possible to perform these operations on long values (eg. long1 = long2 * long3). However, we did include functions to perform these operations, but you will have to call these functions explicitly (eg. long1 = l_mul(long2,long3)). People who know their compiler well, can work around this problem by extracting the long math functions from the standard library, and linking them with the PEX file. For example, the long math functions used by the Zortech compiler are in the module LMATH.OBJ, located in the library ZLC.LIB. Extracting this module is as simple as : ZORLIB ZLS *LMATH;

These are the ProBoard long math functions that you can use:

```
long l_mul(long1,long2)   Multiply 2 long values   = long1 * long2
long l_div(long1,long2)   Divide 2 long values     = long1 / long2
long l_mod(long1,long2)   Modulo of 2 long values  = long1 % long2
long l_shl(long1,int1)    Left shift a long value  = long1 << int1
long l_shr(long1,int1)    Right shift a long value = long1 >> int1
```

The functions l_div(), l_mod(), l_shl() and l_shr() are also available for unsigned long values: ul_div(), ul_mod(), ul_shl() and ul_shr().

Floating point operations are NOT supported. So it is not possible to use 'double' and 'float' variables & constants.

Another major restriction only affects C++ programs that are compiled with some other compiler than Borland C/C++ 3.x or Zortech C++ 3.0. In a ProBoard C++ program that is not compiled with one of the compilers mentioned, you can't declare global variables that have constructors or destructors (ie. no static constructors & destructors). So this will compile fine,

but will certainly crash your machine:

```
class myclass
{
    int a;
  public:
     x() { .... some action .... }
    ~x() { .... some action .... }
};
myclass some_global_object;
main()
{
    ...
}
```

You can, however, use objects with constructors and destructors, as long as they are declared within a function. So this would work:

```
main()
{
    myclass some_local_object;
    ....
}
```

Library functions supported

Almost all standard library functions are supported. Note that all functions that write to the standard output device (printf, putchar,…) write their output to the ProBoard window and to the user (through the modem). Functions that read from the standard input device and from the keyboard are NOT supported. (scanf,getch,gets,…) We provide substitutes for these functions.

Functions supported:

fopen, freopen, fseek, ftell, fgets, fgetc, fflush, fclose, fputs,
getc, getchar, gets, fputc, putc, putchar, puts, fread, fwrite,
printf, fprintf, vfprintf, vprintf, sprintf, vsprintf, setbuf,
setvbuf, remove, rename, rewind, clearerr, feof, isalpha, iscntrl,
isdigit, isgraph, islower, isprint, ispunct, isspace, isupper,
isxdigit, toupper, tolower, int86x, intdos, intdosx,
dos_findfirst, dos_findnext, write, open, creat, close, unlink,
chsize, dup, dup2, lseek, access, filesize, filelength, isatty,
atoi, atol, strtol, rand, srand, calloc, free, malloc, realloc,
putenv, getenv, abs, labs, memcpy, memmove, strcpy, strncpy,
strcat, strncat, memcmp, strcmp, strncmp, memchr, strchr, strcspn,
strpbrk, strrchr, strspn, strstr, strtok, memset, strlen, memicmp,
stpcpy, strcmpl, strnicmp, strdup, strlwr, strupr, strnset, strrev,
strset, swab, strncmpl, strnicmp, clock, time, mktime, asctime,

ctime, localtime, gmtime, strftime, sleep, usleep, msleep, difftime,
mkdir, chdir, rmdir, fileno, getcurdir, getdisk, _dos_getftime,
_dos_setftime.

For information on these functions, consult your compiler's library reference manual.

In the string output functions (printf and puts), you can use the following control codes in your string:

```
\001 or \x01   Set output color to red
::::    ::::
::::    ::::
\007 or \x07   Set output color to white
\x11 to \x17   Same as \1 to \7, but blinking colors
\t             Stops output of string until user presses [Enter]
\f             Clears the user's screen (if enabled in the user's
               record)
```

NOTE: Do NOT include any standard include files from your compiler. The only include file you can use is PB_SDK.H. THAT'S IT!!

ProBoard Interface Functions

This is a list of all the functions that can be used to interface with ProBoard.

Several types have been defined in the file PB_SDK.H

```
bool    : a boolean value that can have the values TRUE or FALSE
byte    : an unsigned character (8 bits)
word    : an unsigned short integer (16 bits)
dword   : an unsigned long integer (32 bits)
KEY     : used for key scan codes (for sysopkey handlers and the
          ScanKey() function).
```

void AddTime(int min);

Adds 'min' minutes to the user's time left. If 'min' is negative, the number of minutes will be subtracted from the time left.

Examples:

```
AddTime( 10 );     /* Adds 10 minutes     */
AddTime( -5 );     /* Subtracts 5 minutes */
```

int TimeLeft(void);

Returns the time left in minutes for the current user.

int TimeOnline(void);

Returns the time in minutes that the current user is online.

void SuspendTimer(void);

Suspends the online-timer for the current user. No time will be subtracted until RestartTimer() is called.

void RestartTimer(void);

Restarts the timer after a call to SuspendTimer().

void AdjustTime(void);

Call this function if you changed the user's level. This recalculates all limits (time,download,…).

void MsgEd(void);

Fires up the internal message editor or the fullscreen editor, depending on the settings of the user. The message will be stored in the file MSGTMP in the current directory. If a message is aborted, the file MSGTMP will be deleted by the message editor. If you create a MSGTMP file before invoking the message editor, this file will be used as a 'quote' message.

int PostMessage(char *from, char *to, char *subject, int area, bool pvt);

Posts a message to a user.

from = Name of the sender
to = Name of the user you're sending the message to
subject = Subject of the message
area = Message area number where the message has to be posted in
pvt = Private status (TRUE=private, FALSE=public)

No checking is performed on any of the parameters!

Return value: -1 on error
0 if ok

int PostNetmail(char *from, char *to, char *subject, int area, FIDO_NODE *address, bool attach, bool crash, bool kill);

Posts a netmail message.

from = Name of the sender
to = Name of the user you're sending the message to
subject = Subject of the message
area = Message area number where the message has to be posted in
address = Destination node number of this message.
attach = TRUE for a file attach message

crash = TRUE if this message is a crashmail message
kill = TRUE if the message has to be deleted after it is sent

FIDO_NODE is a structure with the following fields:

```
struct
{
   unsigned zone;     /* Zone number  */
   unsigned net;      /* Net number   */
   unsigned node;     /* Node number  */
   unsigned point;    /* Point number */
}
```

No checking is performed on any of the parameters!

Return value: -1 on error
0 if ok

bool ReadMsgArea(int area, MSGAREA *ma);

Reads data about a message area.

area = Message area number (1-200)
ma = Pointer to a MSGAREA structure to be filled

Check the file PB_SDK.H for details about the MSGAREA structure definition.

Return value: 6
FALSE if not defined
TRUE if read ok

bool ReadMessage(MESSAGE *msg, long msgid, int areanum);

Reads message id <msgid> from area numer <areanum> into the MESSAGE structure. To access the text of the message, use the function CreateMessageText(). See the file PB_SDK.H for a description of the MESSAGE structure.

Return value: FALSE if the message does not exists
TRUE if the message is read ok

void WriteMSGTMP(char *text);

Writes the string <text> to the file MSGTMP. The file is always recreated when using this function. To append text to MSGTMP, use the function AppendMSGTMP(). You have to insert your own CR/LF pairs in the text if you want to force a newline. Lines can be longer than 80 characters. The message processing functions will do a word-wrap if necessary. Do NOT insert a single CR or LF in the text. (\r or \n).

void AppendMSGTMP(char *text);

Same as WriteMSGTMP(), but adds text to an EXISTING file.

void ShowMessage(MESSAGE *msg);

Shows the message <msg> to the user, including header and message text.

See the file PB_SDK.H for an description of the MESSAGE structure.

void CreateMessageText(MESSAGE *msg);

Reads the text that belongs to message <msg>, and stores it in a file called MSGTMP in the current directory. This file can then be accessed as an ordinary textfile.

void CreateMessageTextString(MESSAGE *msg, char *text, int max);

Reads the text that belongs to message <msg>, and stores it in the string <text>. The text will be terminated by a '\0' character. You have to specify a maximum number of characters that can be stored in <text>.

bool FirstMessage(MESSAGE *msg, int area, int order, long first);

bool NextMessage(MESSAGE *msg, int area, int order);

These functions are used to read messages in forward or reverse order. To start scanning messages, you have to call the FirstMessage() function first, and the NextMessage() function to read more messages.

```
msg   = MESSAGE structure where the message has to be stored.
area  = area number to read messages in
order = 1 for forward order, -1 for reverse order
first = message to start reading from. This message does NOT need to
        exist. The first non-deleted message following this message
        will be read first.
```

See the file PB_SDK.H for an description of the MESSAGE structure.

```
Return value:  FALSE  No more messages found.
               TRUE   Message was found and read ok.
```

This is an example on how to use these functions:

```
result  =  FirstMessage( msg,
                   5,
                   +1,
                   1 );
while ( result == TRUE )
{
   /* Do something with the message */
   result  =  NextMessage( msg,
                     5,
                     +1 );
```

}

void DeleteMessage(MESSAGE *msg);

Deletes message <msg>.

void MarkMessage(long msgid);

Marks message <msgid> for later retrieval. Up to 200 messages can be marked.

void ReadMarkedMessages(void);

Shows all marked messages to the user, with the option to wait after each message. If the user selected to wait after each message, the standard message options menu will be shown to the user. (Next,Prev,Stop,…).

In fact, this function is identical to menu function “Read Messages”, with the user selecting “Marked”.

void ListMarkedMessages(void);

Is similar to ReadMarkedMessages(), but this function acts as a “QuickScan Messages” with the user selecting “Marked”.

void UnMarkAllMessages(void);

Unmarks all messages that were previously marked. You have to call this function before marking any messages, because the list of marked messages is remembered until a user logs off.

int NumMsgAreas(void);

Returns the highest message area number that is not empty.

long GetLastRead(int areanum, long user_recno);

void SetLastRead(int areanum, long user_recno, long msgid);

Reads / Sets the lastread pointer for a specific user and message area. If GetLastRead() returns -1, an error occurred.

<areanum> ..... The area number (1-10000)
<user_recno> .. The record number of the user (0 - ...)
<msgid> ..... The id of the message you want to set the lastread
pointer to.

int FuzzySearch(char *text, char *string, int degree);

Performs a fuzzy search for <string> in <text>. <degree> is the percentage of the characters in <string> that have to match.

Return value: -1 if not found
>0 percentage of characters matched

IO_... functions

All functions starting with IO_ are low-level functions to access the serial port directly. Do NOT use these functions for ordinary I/O operations. They are intended for writing file transfer protocols. See the file PB_SDK.H for the prototypes of these functions.

char WaitKey(void);

Waits for a character to be read from the modem or local keyboard.

Return value: the character read.

char WaitKeys(char *keylist);

Waits for any character specified in <keylist>.

Return value: the key from the list that was pressed (converted to uppercase if the key is a letter).

Example:

```
c = WaitKeys( "ABC\r\x1b" );
/* Waits for A,B,C,<Enter> or <Esc> */
```

void Input(char *buf, int len, int readmode);

Asks for a string from the user. The string is stored in <buf>, and the user will not be allowed to enter more than <len> characters.

```
<readmode> : INPUT_ALL      : All characters are allowed
             INPUT_UPFIRST  : First character of each word is
                              converted to uppercase.
             INPUT_UPALL    : All characters are converted to
                              uppercase.
             INPUT_DIGITS   : Only digits are accepted.
             INPUT_NOFIELD  : OR together with any of the previous
                              modes if you don't want an input-
                              field background to be displayed.
```

bool Ask(bool default);

Asks for a Yes or No response from the user. Pressing 'Y' returns TRUE and outputs “Yes”. Pressing 'N' returns FALSE and sends “No” to the user. Pressing <Enter> returns <default>, and sends “Yes” or “No”, depending on <default>.

char PeekChar(void);

Checks if a character is available in the input buffer, and returns that character if there is.

Return value: 0 if no character available
!= 0 the character read (key pressed by user)

void SetColor(char color);

Changes the current output color. All text that is output after this function is called will be displayed in the specified color. This function does nothing when the user does not have ANSI or AVATAR enabled.

<color> can be one of the following:

BLACK, RED, GREEN, YELLOW, MAGENTA, BLUE, CYAN, WHITE

void SetfullColor(unsigned char color);

Changes the current foreground & background color. The color code is an IBM-type screen color code. This function does nothing when the user does not have ANSI or AVATAR enabled.

Some examples:

0x1E : Bright yellow on dark blue background.
0x87 : Blinking white on a black background.

void GotoXY(int x, int y);

Moves the cursor to position (x,y). The upper left corner of the screen is (1,1). This function does nothing when the user does not have ANSI or AVATAR enabled.

void ClrEol();

Clears from the current cursor position to the end of the line. This function does nothing when the user does not have ANSI or AVATAR enabled.

void EnableStop(void);

void DisableStop(void);

bool Stopped(void);

Enables stopping of output by pressing 'S'. When enabled, a user can press 'S' to stop incoming text. When this happens, any string output function will immediately return to the caller, en the function Stopped() will return TRUE to indicate that the user requested to stop output.

EnableStop() Enables this feature and sets the "stopped" status to FALSE .
DisableStop() Disables the stop feature.

Stopped() Returns the "stopped" flag (TRUE or FALSE).

Example:

```
EnableStop();
for ( i = 0;  i < 100;  i++ )
{
    printf( "...something..." );
    if ( Stopped() )
    {
        break;
    }
}
```

char ShowHotkeyFile(char *filename, char *hotkeys);

Shows the file <filename> to the user, and watches for incoming keys that are specified in <hotkeys>. If one of these keys is detected, the function stops output, and returns the hotkey. The “stop” feature will be enabled when sending the file.

Return value: 0 : File sent completely and no hotkeys pressed.
1 : 'S' pressed (output stopped)
2 : File not found
!=2 : Hotkey detected

char ShowHotkeyANSIFile(char *filename, char *hotkeys);

This function is identical to ShowKotkeyFile(), except for the fact that no extension and path can be specified. Depending on the setting of the user, <filename>.ANS/ASC in the textfiles directory will be sent. If no .ANS file is found, it will try to send the .ASC file. If that fails too, 2 will be returned.

Return value: 0 : File sent completely and no hotkeys pressed.
1 : 'S' pressed (output stopped)
2 : File not found
!=2 : Hotkey detected

void InitLineCounter(void);

bool LineCounter(void);

These functions are used to support page pausing (“More Y/N”). Calling InitLineCounter() initializes the line counter to 0. Every time you call LineCounter(), the line counter is incremented by one. If the counter reaches the screen length, the user will be asked if he wants to continue reading. If he selects “No”, the LineCounter() function will return FALSE.

Example:

```
InitLineCounter();
```

```
for ( ; ; )
{
    printf( "...something...\n" );
    if ( ! LineCounter() )
    {
        break;
    }
}
```

bool ExternalInput(void);

Returns TRUE when the last character that was read through any of the input routings originated from the remote keyboard (ie. the sysop did not type it).

int ReadUser(USER_REC *rec, int recnr);

Reads user record number <recnr> (0-…) from the user file and stores it in <rec>.

Return value: -1 if record does not exist.
0 if record is read ok.

void WriteUser(USER_REC *rec);

Writes user record <rec> to the user file. The record number is stored in the USER_REC structure. If you want to write a user record to a different position in the user file, you have to change the 'record' field in the USER_REC structure.

bool SetUser(long recnr);

Sets the internal ProBoard user record to record number <recnr>. This function should ONLY be called from _LOGIN.PEX !!

Return value: TRUE on success
FALSE on error

char PlayMusic(char *filename, char *hotkeys);

Plays a .MUS file. The file has to be located in the ProBoard system directory. Do not include a path and extension in <filename>. The music can be stopped if the sysop presses one of the keys specified in <hotkeys>. ^^^^^

Return value: 0 Music file played till the end
1 Music file not found
>1 The hotkey pressed by the SYSOP
^^^^^

void PostInfo(char *filename);

Does exactly the same as the POSTINFO statement in a .Q-A file. No path and extension is

allowed in <filename>. The file where the information will be written to will have the extension .ASW and will be placed in the ProBoard system directory.

int ReadFileArea(int areanum, FILEAREA *fa);

Reads information about file area number <areanum> in the <fa> structure.

Check the file PB_SDK.H for details about the FILEAREA structure.

Return value: -1 File area does not exist
0 Ok

void MenuFunction(int function, char *data);

Executes menu function <function> with <data>. For details, see the menu functions description in this file.

There are named constants declared for each function in the file PB_SDK.H. It is recommended that you use these constants instead of numbers.

void Log(int loglevel, char *fmt, ...);

With this function you can write information to the ProBoard logfile. The log entry will be written if the user has a loglevel that is greater than <loglevel> .

<loglevel> can be one of the following:

LOG_FRIEND, LOG_NORMAL, LOG_SUSPICIOUS, LOG_DANGEROUS

If you want a log entry to be written regardless of the user's loglevel, use LOG_FRIEND.

The Log() function works like the printf function: you can use format specifiers in the <fmt> string, and pass extra parameters.

Example:

```
Log(LOG_NORMAL,"User's name = %s",CurUser->name);
```

void HangUp();

Hangs up the phone, logging off the user. It is exactly the same as pressing Alt-H.

bool CheckAccess(int level, long flags);

Checks if the current user can access something that requires <level> and <flags>. Returns TRUE if the user has access, FALSE if not.

KEY ScanKey(void);

Checks the LOCAL keyboard for a key, and returns the scan code for that key. If no key is available, 0 will be returned.

For a list of defined values for scan codes, check the file PB_SDK.H.

bool GetFlag(long flags, char flagchar);

void SetFlag(long flags, char flagchar);

void ClearFlag(long flags, char flagchar);

Access flags are stored in a long integer (32 bits). You have 3 macros at your disposal to manipulate access flags:

GetFlag : Returns TRUE if flag <flagchar> is set
SetFlag : Turns flag <flagchar> on
ClearFlag : Turns flag <flagchar> off

<flagchar> must be a value from 1 to 32. 1-26 correspond to A-Z, while 27-32 are the same as 1-6.

int ErrorLevel(void);

Returns the errorlevel of the last executed shell command (Through MenuFunction(MENU_SHELL,…))

bool GetIniVar(char *fname, char *varname, char *value, int max);

bool SetIniVar(char *fname, char *varname, char *value);

These functions can be used to read and write to .INI files. These files are similar to the files used by MS Windows. They can be used to store settings for your application (PEX). An example of a .INI file:

```
DataPath = C:\PB\PEX\MYPEX\DATA
ProgPath = C:\PB\PEX\MYPEX
MaxUsers = 10
```

This is a standard ASCII file, with each line in the form:

<varname> = <value>

The value of a specific variable can be read by calling the function GetIniVar(). Setting a variable (writing to the .INI file or createing it if it doesn't exist yet) can be done by calling SetIniVar().

The parameters for the functions are:

<fname> Name of the .INI file. The extension of the file will always be changed to .INI by ProBoard.
<varname> Name of the variable (case INsensitive)
<value> For GetIniVar() : a pointer to a buffer where the variable's contents will be stored.

For SetIniVar() : a pointer to the value of the
variable.
<max> Maximum number of characters that can be copied in
<value> (including trailing '\0')

Return value: FALSE if the file could not be opened or the variable does not exist.

void exit(void);

Exits the current pex-program, and unloads it from memory.

void ExitTSR(void);

Exits the current pex-program, and leaves it resident. This has some important implications:

- All global and static variables keep their values for subsequent executions.
- When the same pex-file is run again through menu function 60, the resident copy will be executed. This will result in faster loading.
- Any handlers that were installed will be called.

void InstallHandler(int handler, function);

This is probably the most advanced and complicated function in the ProBoard SDK. With this function you can set up a “handler” for a specific action. Right now, only two types of handlers are supported: a replacement for the sysopkey-handler and a handler to intercept loss of carrier. This way you can create your own sysopkeys, or change the behaviour of existing sysopkeys. You could create a handler that intercepts the Alt-J key, and write your own flavor. In future versions you will be able to set up handlers for low-level I/O operations. So you could replace all I/O routines by your own functions. This way you can support your own exotic hardware or something like that. Anything goes!

handler : Handler type. At this time, only HANDLER_SYSOPKEY and
HANDLER_HANGUP are supported.
function : A pointer to the handler function you created.

You can install as many handlers as you want. When the PEX is removed from memory (when the PEX exits), all handlers that were installed will be removed by ProBoard.

The handler function is a function that returns an integer. The parameters depend on the type of handler you are creating.

For HANDLER_SYSOPKEY the handler function has to be declared like this:

```
int sysopkeyhandler( KEY k )
{
    ...
}
```

A sysopkey handler intercepts any special key that is pressed by the sysop. (like F1,Alt-H,…). You

can redefine any key, or add your own.

For HANDLER_HANGUP, the handler functions has to be declared like this:

```
int hanguphandler( void )
{
    ...
}
```

The hangup handler works a little different: it is called whenever ProBoard exits (carrier lost, sysop hung up,…). It can be used to perform some cleanup for your running PEX file (like closing files, writing data, etc..). You can install the handler at the start of your program and remove it before the program is finished.

If a handler function decides to do anything, it has to return HANDLED. If the handler wants to leave it up to ProBoard, it must return NOT_HANDLED.

This asks for an example I suppose:

```
int keyhandler( KEY k )
{
    if ( k == KEY_ALTJ )
    {
        printf( "\7\rHang on %s! I'm shelling to DOS...",
              UserFirstName );
        MenuFunction( MENU_SHELL, "*N*Q*X*!" );
        printf( "I'm back!!\n" );
        return HANDLED;
    }
    else
    {
        return NOT_HANDLED;
    }
}
main( int   argc,
     char *argv[] )
{
    InstallHandler( HANDLER_SYSOPKEY, keyhandler );
    ExitTSR();    /* IMPORTANT */
}
```

So what does it do? It intercepts the sysopkey-function, and checks if the key pressed is Alt-J. If it is, it shells to DOS with swapping ON, and it writes its own messages to the screen. When the sysop returns, the handler returns HANDLED to let ProBoard know that it does not have to process the key. You could use this example in an INIT.PEX file. Is this powerful or what?

void RemoveHandler(int handler, function);

Removes handler number <handler> which has been installed by InstallHandler(). <function> is the pointer to the handler that should be removed.

long MsgNum(int area, long id)

Returns the message number for message 'id', in area 'area'. For Hudson and *.MSG, this is identical to the message id (but DO NOT rely on that). Returns a value < 1 if there is no message number for 'id'.

long MsgId(int area, long n)

Returns the message id for message number 'n', in area 'area'. For Hudson and *.MSG, this is identical to the message number (but DO NOT rely on that). Returns a value < 1 if there is no message number 'n'.

long HighMsg(int area)

Returns the message id of the last message in the given area. Note: if the lastread pointer is higher than what this function returns, DO NOT use FirstMessage()! The FirstMessage() function does not support reading past the end of the area. (if you know what I mean

long NumMsgs(int area)

Returns the total number of active messages in the given area.

void LocalDisplay(bool showlocal)

Enables/Disables local echo of text. A parameter of TRUE means “enable local display”. FALSE means “disable local display”. See _GRAPH.CPP for an example.

void RemoteDisplay(bool showremote)

Enables/Disables remote echo of text. A parameter of TRUE means “enable remote display”. FALSE means “disable remote display”. See _GRAPH.CPP for an example.

bool RIP()

Returns TRUE if RIP was detected and enabled.
FALSE if not...

void ShowRIPscrip(char *name)

This will send a RIP file located in the RIP directory. No file extension should be given. (.RIP is assumed) The file will not be shown on the local screen.

void ResetInactivity(void)

Resets the internal inactivity timeout counter. This function can be used when returning from a shell or after a time-consuming function has been called. It will prevent the user from being logged off because of inactivity.

bool AddTaggedFile(TAGGED_FILE *tag)

Adds a file to the internal list of tagged files.

Returns TRUE if the was successfully added.
FALSE if the file was already tagged.

bool RemoveTaggedFile(TAGGED_FILE *tag)

Removes a file from the internal list of tagged files. Returns TRUE if the tagged file was found and deleted from the list, or FALSE if the tagged file was not in the list.

void ClearTaggedFiles(void)

Clears the internal list of tagged files (all tags are removed).

bool IsTagged(TAGGED_FILE *tag)

Returns TRUE if the given file is in the internal list of tagged files,
FALSE if not.

int GetTaggedFiles(TAGGED_FILE *tagarray, int maxitems)

Fills the given array <tagarray> with all the files in the internal list of tagged files (up to a maximum of <maxitems>). This function returns the number of tagged files copied to the array <tagarray> (⇐ maxitems).

bool PutTaggedFiles(TAGGED_FILE *tagarray, int nitems)

Fills the internal list of tagged files with the files in the given array <tagarray>. <nitems> is the number of items in the array.

Returns TRUE on success
FALSE on error

Global ProBoard Variables

You have access to some important ProBoard system variables through global variables defined in PB_SDK.H.

word PBVersion;

The version number of ProBoard. The high byte is the major version number (eg. 1). The low byte is the minor version number in hex (eg. 0x30). So if you're using ProBoard v1.30, PBVersion

will be set to 0x0130 .

word Beta;

The beta version number of ProBoard. If this number is 0xFFFF, it means that the release version is running. Beta numbers start from 1.

For example, when running ProBoard 1.40 Beta/19, PBVersion will be set to 0x0140, and Beta will be set to 19.

long BaudRate

The current bps rate of the caller. 0 means local.

USER_REC *CurUser;

Pointer to the current user record. You can change any of the field values in the record, but this will not result in any immediate action by ProBoard. For example, if you change the user's level, ProBoard will not know that you changed it, so you will have to tell it by calling AdjustTime();

Check the file PB_SDK.H for a description of the USER_REC structure.

int UserRecNr;

The record number of the current user's record in the file USERS.BBS, starting from 0. This value cannot be changed.

int NumLimits;

The number of user levels defined in ProCFG. You can access the limit-values through the global array 'Limits'. This value cannot be changed.

LIMIT *Limits;

Is an array of all user levels defined, with download and time limits. Check the file PB_SDK.H for information on the LIMIT structure.

You are not allowed to change any values! (A good compiler should generate an error or warning if you try to do so)

char *LoginDate;

char *LoginTime;

This is the login date & time of the current user.

LoginDate[0] : Day portion of login date
LoginDate[1] : Month portion of login date
LoginDate[2] : Year portion of login date (00-99)
LoginTime[0] : Hour portion of login time

LoginTime[1] : Minute portion of login time
LoginTime[2] : Second portion of login time

bool NetEntered;

bool EchoEntered;

These READ-ONLY variables tell you if netmail and/or echomail has been entered during this session.

int NumUsers;

This READ-ONLY variable is the number of users currently available in the file USERS.BBS.

int NodeNumber;

This READ-ONLY variable is the current node number.

char *CurMenu;

char *UserFirstname;

char *PrevUser;

char *StartupPath;

char *SysPath;

char *PageReason;

CurMenu : Current menu name
UserFirstName : First name of current user
PrevUser : Name of previous user
StartupPath : Name of the directory where ProBoard was started from
(with trailing '\')
SysPath : Name of the ProBoard system directory (with trailing '\')
PageReason : Reason for paging the sysop (as entered by the user)

These are READ-ONLY strings! (except PageReason)

word *PageCount;

POINTER to the number of sysop pages done by the user. This value can be changed by changing the unsigned integer pointed to by 'PageCount'.

CONFIG *Config;

A pointer to the current ProBoard configuration structure. See the file PB_SDK.H for a

description of the CONFIG structure.

Special-purpose PEX-files

Several pex-files will be loaded automatically (if present). They perform certain actions like edit a message, set up handlers, etc.

INIT.PEX : Will be loaded and executed before any I/O has been done. Use this to set up any handlers (sysopkey handlers, ...)

INIT_1.PEX -

INIT_9.PEX : Is the same as INIT.PEX, but will be loaded in the order of the numbers. So you can have up to 10 initialization pex-files.

_LOGIN.PEX : Is run instead of the normal login procedure. See the section explaining this PEX for more information

LOGIN.PEX : No longer supported!

BIRTHDAY.PEX : Is run after showing NEWS.ANS/ASC if today is the user's birthday.

NEWUSER.PEX : Is run before the file NEWUSER.A?? is shown.

NEWUSER1.PEX : Is run before the file NEWUSER1.A?? is shown.

NEWUSER2.PEX : Is run before the file NEWUSER2.ANS/ASC is shown.

WELCOME.PEX : Is run before the file WELCOME.ANS/ASC is shown

WELCOMEx.PEX : Is run before the file WELCOMEx.ANS/ASC is shown (x = 1 to 9)

SECxx.PEX : Executed when a user with security level xx logs in. Is run before SECxx.ANS/ASC is shown.

EXPIRED.PEX : Is run before EXPIRED.ANS/ASC is shown, and after a user's level is lowered because the subscription date was reached. ProBoard will send one parameter to this PEX: the original level, before it was changed by ProBoard.

GOODBYE.PEX : Is run before GOODBYE.ANS is displayed.

GOODBYE2.PEX : Is run after GOODBYE.ANS is displayed.

MSGED.PEX : Is a replacement of the built-in message editor. It will be executed with no parameters. The message that you create has to be written to a file called MSGTMP. ProBoard will then read this file and create a message from it. To let ProBoard know that the user has aborted a message, delete MSGTMP.

FSED.PEX : Is similar to MSGED.PEX, but will be executed if the user has selected the fullsrceen editor. It works the same way as MSGED.PEX, but ProBoard will create a MSGTMP file with the original message if a user is

replying to a message.

Using the _LOGIN PEX

By writing a _LOGIN.PEX, ProBoard allows you to replace the normal login procedure (not including the new user procedure) with your own. It is the PEX's responsibility to check for validity of the name entered and to do password checking!

Once you determined the record number of the user, you have to call the function SetUser() with the record number as parameter, and exit the PEX immediately.

If the user is not in the user database yet, you should exit without calling SetUser(), and set the name that was entered by the user in a memory region supplied by ProBoard to your PEX as the only parameter. This parameter is a 'dword' in decimal form, so you have to convert it to a pointer using the following procedure:

```
/*----------------------------*/
char  *pName;
dword  dwName = 0;
char  *s;
for ( s  =  argv[ 1 ];  *s;  s++ )
{
   dwName  =  l_mul( dwName, 10L )  +  (dword) ( *s - '0' );
}
pName  =  (char *) dwName;
/*----------------------------*/
```

ProBoard File Structures

This document describes the file structures used by ProBoard v2.15-v2.30. For information on the structures of the RemoteAccess compatible files, refer to the RemoteAccess structures document. It can be obtained from http://pcmicro.com/ra/ra-y2k-2.zip

The data-structures are C-structures, this means:

Strings are stored as an array of characters, terminated by a 0.
A date is stored as 3 bytes (day,month,year)
A time is stored as 3 bytes (hour,min,sec)

The 'accessflags' are stored in a long integer, with bit 31 being flag 'A', bit 30 flag 'B' , … Bit 5 is flag '1', bit 4 is flag '2', etc… I know this doesn't sound logical, but we had some good reasons for doing this  .

The combined boards are stored in 125 bytes (1000 bits). One bit per message area.

```
typedef unsigned short word;
typedef unsigned long  dword;
typedef unsigned char  byte;
typedef unsigned long  accessflags;
```

```
typedef unsigned char  combinedboards[125];
typedef unsigned char  bool;
typedef unsigned char  Time[3];
typedef unsigned char  Date[3];
typedef unsigned char  TimeFrame[7][6];

typedef struct
 {
  byte periods[7][6];    7x48 periods of 1/2 hour = one week
 }
 TimeFrame;

typedef struct
 {
  word        minLevel;             0 =          don't care
  word        maxLevel;             0 =          don't care
  accessflags  flags;              All off =     don't care
  accessflags  flagsNot;             All off =     don't care
  byte        minAge;              0 =          don't care
  byte        maxAge;              0 =          don't care
  byte        sex;                 0 =          don't care
  word        minTimeLeft;          0 =          don't care
  word        minTimeOnline;        0 =          don't care
  TimeFrame    timeFrame;            All on =      don't care
  byte        terminals;           0xFF =       don't care
  dword       minSpeed;             0 =          don't care
  dword       maxSpeed;             0 =          don't care
  byte        extra[50];           Extra space
 }
 AccessRecord;
```

USERSPB.BBS

This file is tied to the files USERS.BBS and USERSXI.BBS. The records are stored in the same order as the records in USERS.BBS. The name of the user is duplicated in this file, so it is possible to fix the userfile after using a third-party userfile packer/sorter, which does not know about USERSPB.BBS.

```
struct UsersPbBBS
  {
    char         name[36];
    char         country[26];
    char         state[26];
    char         faxPhone[16];
    char         passWord[16];
```

```
    char          language[9];
    Date          lastPwdChange;
    Date          lastNewFilesCheck;
    short         logLevel;
    short         tbTimeBalance;
    short         tbKbBalance;
    short         tbTimeWithdrawn;
    short         tbKbWithdrawn;
    word          tbTimeDeposited;
    word          tbKbDeposited;
    word          tbTimeLoaned;
    word          tbKbLoaned;
    Date          tbTimePayback;
    Date          tbKbPayback;
    Date          tbLastUsed;
    word          expLevel;
    accessflags   expFlagsOn;
    accessflags   expFlagsOff;
    dword         uFlags;
    combinedboards mailCheckBoards;
    dword         totalTimeUsed;
    word          qwkMaxMsgsPerArea;
    word          qwkMaxMsgs;
    byte          qwkArchiver;
    byte          ripFont;
    byte          checkMail;
    byte          checkNewFiles;
    byte          extra[398];
  };
```

These are the extra user-flags for 'uFlags':

```
#define RA_UFLAG_NOIBM       1
#define RA_UFLAG_NOTOPS      2
#define RA_UFLAG_AVTPLUS     4
#define RA_UFLAG_ATTEN       8
#define RA_UFLAG_NORIP      16
#define RA_UFLAG_MULTILOGIN  32
#define RA_UFLAG_FREECHAT    64
#define RA_UFLAG_LOCALONLY  128
```

CONFIG.PRO

```
struct CONFIG_PRO
  {
    char       shellmsg[81];         Message to show when shelling
```

```
char        sysopname[36];        Name of sysop
char        txtpath[61];          Path for textfiles
char        mnupath[61];          Path for menu-files
char        msgpath[61];          Path for message base
char        uploadpath[61];       Uploadpath
char        editorname[61];       Name of external editor
word        newuserlevel;         Level for new user
short       newuserloglevel;      Loglevel for new user
accessflags newuserflags;         New user flags
short       max_passinput;        Maximum attempts for password entry
short       min_passlength;       Minimum password length
short       inactivity_time;      Inactivity time-out limit
short       max_sysop_pages;      Maximum times sysop can be paged
short       pagebell_length;      Length of page-bell (secs)
short       mailcheck;            Check for mail at logon?
short       europe;               European date format?
short       numnodes;             # nodes
bool        allowansi;            Allow ANSI?
bool        allowavatar;          Allow AVATAR?
short       discrete;             Hide sysop activity?
short       askphone;             Ask for phone number?
short       allowoneword;         Allow one-word names
unsigned    crashlevel;           Level needed for crashmail
accessflags crashflags;           Flags needed for crashmail
word        attachlevel;          Level needed for file attach
accessflags attachflags;          Flags needed for file attach
short       allowmsgupload;       Allow message uploads
short       allowstacking;        Allow command stacking
Time        page_start;           Paging hours start
Time        page_end;             Paging hours end
short       handshaking;          I/O Handshaking
short       allowalias;           Allow alias for login
short       loglocal;             Log local calls
short       doswap;               Allow swapping
char        originline[61];       Origin line
char        nodelistdir[61];      Nodelist directory
char        sysopkeys[10][60];    Sysop hotkeys
Time        dl_start;             Download hours start
Time        dl_end;               Download hours end
short       uploadspace;          Space needed for uploads
char        pvtuploadpath[61];    Directory for files uploads
char        quotestring[6];       String used for quoting
bool        fastmode;             Use fast mode
```

```
bool       extra_bool_1;
bool       killsent;            Kill netmail after sent
bool       multiline;           Use message base sharing?
bool       egamode;             Use 43/50 line mode
bool       showuserinfo;        Show user info while in EGA mode?
char       pexpath[61];         Directory for PEX files
bool       allowquicklogin;     Allow quick sysop login?
bool       suspendmsgtime;      Suspend time when writing msgs
short      securityboard;       MsgBoard for security messages
bool       pwdmessages;         Write security-messages?
bool       extra_bool_2;
char       bbsname[36];         Name of the BBS
char       pwdchar;             Password character
short      tb_maxtimedeposit;   Max time deposit per day (TimeBank)
short      tb_maxkbdeposit;     Max Kbytes deposit per day (TimeBank)
short      tb_maxtimewithdrawal; Max time withdrawal per day (TimeBank)
short      tb_maxkbwithdrawal;  Max Kbytes withdrawal per day (TimeBank)
short      usage_days;          Days to keep usage graphs
char       systempwd[16];       System password
bool       usesystempwd;        Use system password?
bool       askbirthdate;        Ask Birth Date?
short      binlogdays;          # days to log in BINLOG.PB
bool       binloglocal;         Log local calls to BINLOG.PB yes/no
short      pageArea;            Area number for page messages
bool       indexfiles;          Use file indexing
bool       checkdupes;          Check for dupes
bool       killdupes;           Kill duplocate files
bool       ignore_ext;          Ignore file extensions for dupe checking
char       RIPpath[61];         Path for RIP scripts
char       iconspath[61];       Path for RIP icons
char       location[36];        BBS Location (city)
char       phone[26];           BBS Phone #
char       QWKid[9];            BBS QWK id
word       IObuffersize;        I/O buffer size in bytes
TimeFrame  pagingHours;         Paging hours
char       defaultLanguage[9];  Default language
bool       addUploaderName;     Add uploader's name to FILES.BBS
TimeFrame  downloadHours;       Download hours
bool       askdataphone;        Ask data phone #
bool       askfaxphone;         Ask fax phone #
bool       askaddress;          Ask mailing address
bool       asksex;              Ask sex
bool       askdateformat;       Ask date format
```

```
    bool       askstate;           Ask state
    bool       askcountry;         Ask country
    short      fuzzyRate;          Fuzzy search percentage for user editor
    bool       hidePassword;       Hide password in user editor
    bool       valConfirm;         Confirm user validation
    char       extra_char_1[17];
    char       extChat[81];        External chat program
    char       virScanCommand[61];   External upload scan command
    byte       virScanType;        External upload scan command (type)
    char       virScanSemaphore[13]; External upload scan command (semaphore)
    byte       listColorTag;       File list color (tag char)
    byte       listColorFileName;    File list color (file name)
    byte       listColorSize;      File list color (file size)
    byte       listColorDate;      File list color (file date)
    byte       listColorCounter;     File list color (file counter)
    byte       listColorDesc;      File list color (description)
    byte       listColorSeperator;   File list color (description seperator)
    byte       listColorMissing;     File list color (missing indicator)
    bool       hideFileCounter;      File list setup - hide file counter
    bool       disableRIP;         TRUE = disable RIP completely
    char       extra_char_2[81];
    byte       virScanFailedAction;  External upload scan command (action)
    word       virScanFailedArea;    External upload scan command (failed area)
    byte       promptColor;        Prompt color (background)
    bool       internalfsed;       Set to 1 if internal fsed enabled
    char       extra[3];
  };
```

TIMELOG.PRO

```
struct TIMELOG_PRO
  {
    short      numdays;            Number of days active
    Date       lastdate;           Last update
    long       hours[24];          # minutes per hour usage (obsolete)
    long       days[7];            # minutes per day usage  (obsolete)
    long       weeks[53];          # minutes per week usage (obsolete)
    long       totalcalls;         Total calls to system
  };
```

FILECFG.PRO

```
struct FILECFG_PRO
  {
    char       name[80];           Name of file area
```

```
    char       listpath[80];        Path for file-list
    char       filepath[80];        Path for files
    word       level;               Level needed for access
    long       flags;              Flags needed for access
    char       type;                1 = CDROM File listing type
    short      maxfiles;            Max files per day in this area downloadable
    short      maxkb;               Max Kbytes per day in this area
    bool       notops;              Set to 1 -> should not be listed in TOPFILES
    bool       free;               Set to 1 -> free area
    byte       groups[4];           Groups belonging to
    bool       allGroups;           Belongs to all groups
    byte       minAge;              Minimum age
    long       flagsNot;           Access flags not allowed
    byte       dateFormat;          Date format (same as in user file)
    byte       extra[2];
 };
```

MESSAGES.PB

```
struct MESSAGES_PB
  {
    word       areaNum;             # of message area (1-10000)
    word       hudsonBase;          Number of Hudson message base
    char       name[81];            Name of message area
    byte       msgType;             Kind of message area (Net/Echo/Local)
    byte       msgKind;             Type of message (Private only/Public only/.)
    byte       msgBaseType;         Type of message base
    char       path[81];           Path to Squish or *.MSG
    byte       flags;              Alias allowed/forced/prohibited
    word       readLevel;           Minimum level needed to read msgs
    accessflags readFlags;          flags needed to read msgs
    accessflags readFlagsNot;       flags non-grata to read msgs
    word       writeLevel;          Minimum level needed to write msgs
    accessflags writeFlags;         flags needed to write msgs
    accessflags writeFlagsNot;      flags non-grata to read msgs
    word       sysopLevel;          Minimum level needed to change msgs
    accessflags sysopFlags;         flags needed to change msgs
    accessflags sysopFlagsNot;      flags non-grata to read msgs
    char       origin[62];          Origin line
    short      aka;                AKA
    word       rcvKillDays;         Kill received after xx days
    word       msgKillDays;         Kill after xx days
    word       maxMsgs;             Max # msgs
    char       sysop[36];           Area Sysop
```

```
    short      replyBoard;          Area number where replies should go
    char       echoTag[61];         Echomail Tag Name
    char       qwkTag[13];          QWK Area Name
    byte       groups[4];           Groups belonging to
    bool       allGroups;           Belongs to all groups
    byte       minAge;              Minimum age for this area
    byte       extra[112];
  };
#define MSGTYPE_BOTH     0  Private/Public
#define MSGTYPE_PVT      1  Private Only
#define MSGTYPE_PUBLIC   2  Public Only
#define MSGTYPE_TOALL    3  To All
#define MSGKIND_LOCAL    0  Local
#define MSGKIND_NET      1  NetMail
#define MSGKIND_ECHO     2  EchoMail
#define MSGKIND_PVTECHO  3  Pvt EchoMail
#define MSGBASE_HUDSON   0
#define MSGBASE_SQUISH   1
#define MSGBASE_SDM      2
#define MSGBASE_JAM      3
```

ONLINE.PRO

```
struct ONLINE_PRO
  {
    char       name[36];            Name of user online
    char       city[16];            City of user online
    word        baud;               Baud rate of user
    short      status;              0 -> online , <0 -> not online or unavailable
    char       extra[8];
  };
```

TOPS.PB

```
struct TOPS_PB
  {
    char       name[36];            Name of user online
    dword       n;                  Data (# kb/minutes/files/etc...)
  };
```

PROTOCOL.PRO

```
struct PROTOCOL_PRO
  {
    char       name[50];            Name of protocol
    char       key;                 Hotkey for Protocol
    char       flags;               Protocol behavior
```

```
    char       logfile[60];         Logfile from protocol
    char       ctlfile[60];         Control file (redirection file)
    char       dlcom[128];          Download command line
    char       ulcom[128];          Upload command line
    char       dlstring[80];        String to place in download control file
    char       ulstring[80];        String to place in upload control file
    char       dlkeyword[40];       Download keyword to search for in logfile
    char       ulkeyword[40];       Upload keyword to search for in logfile
    short      wordnr;              File name word nr after keyword (1-..)
  };
```

Flags:

```
#define PROT_BATCH   1     Batch protocol
#define PROT_ENABLED 2     Enabled
#define PROT_BOTH    4     Full-duplex protocol
#define PROT_BIM     8     Bimodem-type ctl-file
#define PROT_LOCAL  16     Local only
```

LIMITS.PRO

```
struct LIMITS_PRO
  {
    word       level;               Level
    short      timelimit;           # minutes per day
    short      daily_klimit;        Kbytes per day allowed
    short      pre_download;        # minutes online before download
    char       id[6];               Usergroup ID
    word       free;                Free upload in Kb.
    byte       factor;              Percentage upload required
    word       max_download;        Max download for this level
    short      fallto;              Fall to level x when max. reached
    short      msgfactor;           # Kbytes granted per message written
    char       extra[5];
  };
```

MODEM.PB

```
struct MODEM_PB
  {
    long       maxBps;              Maximum Baud Rate
    long       lockedBps;           Locked Baud Rate  (not used)
    dword      flags;               Attributes (see below)
    short      port;                COM port (1-8)
    short      commandDelay;        Delay in 1/10s between characters sent
```

```
short     answerDelay;        Delay in 1/10s before answer string sent
short     blankTime;          Time in seconds for screen blanker
char      msgCon300    [80];  Connect 300 string
char      msgCon1200   [80];  Connect 1200 string
char      msgCon1275   [80];  Connect 1275 string
char      msgCon2400   [80];  Connect 2400 string
char      msgCon4800   [80];  Connect 4800 string
char      msgCon7200   [80];  Connect 7200 string
char      msgCon9600   [80];  Connect 9600 string
char      msgCon12000  [80];  Connect 12000 string
char      msgCon14400  [80];  Connect 14400 string
char      msgCon16800  [80];  Connect 16800 string
char      msgCon19200  [80];  Connect 19200 string
char      msgCon21600  [80];  Connect 21600 string
char      msgCon24000  [80];  Connect 24000 string
char      msgCon26400  [80];  Connect 26400 string
char      msgCon28800  [80];  Connect 28800 string
char      msgCon38400  [80];  Connect 38400 string
char      msgCon57600  [80];  Connect 57600 string
char      msgCon64000  [80];  Connect 64000 string
char      msgCon115200 [80];  Connect 115200 string
char      msgConExternal[80]; External/Fax connect string
char      msgRing      [80];  Ring string
char      msgOk        [80];  "OK" string
char      cmdInit1     [80];  Init command string 1
char      cmdInit2     [80];  Init command string 2
char      cmdInit3     [80];  Init command string 3
char      cmdAnswer    [80];  Answer commnad string
char      cmdOffHook   [80];  Off Hook command string
char      cmdDown      [80];  BBS Down command string
char      cmdAttention [80];  Attention command string (usually "AT|")
short     externalErrorLevel; Errorlevel for external/fax connect
char      msgCon31200  [80];  Connect 31200 string
char      msgCon33600  [80];  Connect 33600 string
char      msgCon36000  [80];  Connect 36000 string
char      msgConUser1  [80];  User connect string #1
long      userConBps1;        User connect bps #1
char      msgConUser2  [80];  User connect string #2
long      userConBps2;        User connect bps #2
char      msgConUser3  [80];  User connect string #3
long      userConBps3;        User connect bps #3
char      msgConUser4  [80];  User connect string #4
long      userConBps4;        User connect bps #4
```

```
    char       msgConUser5   [80];    User connect string #5
    long       userConBps5;          User connect bps #5
    char       msgConUser6   [80];    User connect string #6
    long       userConBps6;          User connect bps #6
    byte       extra[920];
  };

#define MODEM_LOCKED       (0x00000001L)
#define MODEM_MANUAL_ANSWER (0x00000002L)
```

BINLOG.PB

```
struct BINLOG_PB
  {
    Date       date;
    Time       timeIn;
    Time       timeOut;
    char       name[36];
    char       city[26];
    char       country[26];
    long       baud;
    word       node;
    long       kbDown;
    long       kbUp;
    word       yells;
    word       level;
    dword      uflags;
    char       alias[36];
    char       extra[45];
  };
```

FILESIDX.PB

```
struct FILESIDX_PB
  {
    char       filename[13];         Name of the file, with extension
    word       area;                 File area number where file is located
  };
```

FGROUPS.PB / MGROUPS.PB

```
struct GROUPS_PB
  {
    char       name[80];
    word       level;
    long       flags;
    long       flagsNot;
    byte       extra[10];
```

```
  };
```

PVTFILES.PB

```
struct PVTFILES_PB
  {
    Date       date;
    char       fname  [80];
    char       to     [36];
    char       from   [36];
    char       desc   [80];
    dword      attr;
    byte       extra  [61];
  };

#define PVTFILE_KEEP (0x00000001L)
```

***.PBM (ProBoard Menus)**

```
#define MENU_RIP 1          menu header attribute
struct MENU_HEADER
  {
    dword attr;   // bit 0 = RIP menu
    char prompt[200];
    byte color;
    byte highlight;
    char RIPname[9];
    byte extra[100];
  };

#define MENU_SHOWREMOTE 1    menu item attribute
#define MENU_SHOWLOCAL  2    menu item attribute

struct MENU_ITEM
  {
    dword attr;     // Show remote/local - reset RIP
    char text[160];
    char data[160];
    byte color;
    byte hotKey;
    word function;
    char password[16];
    AccessRecord access;
    byte extra[50];
  };
```

***.PBL (ProBoard Language Files)**

```
struct LANG_HEADER
```

```
 {
   char       name[41];         Name of the language
   bool       avail;            TRUE = Available
   word       level;            Level needed
   accessflags flags;           Flags needed
   char       menuPath[61];     Path for menus
   char       txtPath[61];      Path for text files
   char       questPath[61];    Path for Q-A files
   char       copyright[81];    Copyright notice
   byte       attrib;           Attribute (not used)
   byte       extra[500];
 };

#define LANG_PROMPT  1
#define LANG_NOCOLOR 2

struct LANG_PROMPT
 {
  word len;              Length of string (excl. terminating '\0')
  word numHotkeys;       # Hotkeys defined
  byte color;            Main color
  byte highlight;        Highlight color
  byte promptcolor;      Prompt color
  byte flags;            Prompt/NoColor
 };
```

Each prompt consists of a LANG_PROMPT header, followed by 'len' characters for the prompt (not including the terminating zero), followed by 'numHotkeys' characters for the hotkeys (if any).

So a language file has the following structure:

```
    LANG_HEADER
    LANG_PROMPT 1
    string 1
    hotkeys 1
    lANG_PROMPT 2
    string 2
    hotkeys 2
    LANG_PROMPT 3
    string 3
    hotkeys 3
    ....
```

RemoteAccess Y2K File Structures

The original document cam be downloaded from: http://pcmicro.com/ra/ra-y2k-2.zip (*

Developers notes and Turbo Pascal structures for RemoteAccess 2.60 GAMMA Preliminary

If you develop software which accesses any of the following files, you should read the following notes. They refer to important structure changes in this release (2.60) and are indicated with the “{}” characters in the first two column positions of the effected line.

Data Types LIMITS.RA { NOT changed for y2k - not necessary } HMBMSGHDR { NOT changed for y2k - see note below } FILESHDR { NOT changed for y2k - not necessary } FILESIDX { NOT changed for y2k - not necessary } USERS { changed for y2k } TIMELOG { changed for y2k } EVENT { format change for y2k } EXITINFO { Added support for y2k - see below for details } DOOR.SYS { Not changed, but marked for future changes }

```
{ with boolean flag to permit 4 digit year info }
{ when implemented by 3rd party developers.     }
```

MENU { changed for y2k }

Primary Purpose of data structure change

The primary change in structures is to create a viable, working solution to “Year 2000 Concerns”. There were several alternatives to accomplish this including “testing the date”.

The primary one chosen revises the “Date” data type from “MM-DD-YY” to “MM-DD-KCYY” “MM”, “DD” and “YY” are the same as they have been. “K” and “C” is for the added two digits of the year - “K” for the thousands digit and “C” for the century digit and will translate to the new displayed and internal format thus: “MM-DD-KCYY”.

A modified y2k date format used is of the format “MMKDDCYY” and will fit within the existing 'OldDate' string descriptor lable.

All internal date coding will use that format and use that format for date storage except as noted.

With the exception of the usage in the EVENTS record, this decision is DIFFERENT than the one previously announced due to of the availability of larger hard drives at lower cost and the simplicity of “decoding the date format”.

Sample of how date will be stored

Example: January 3rd, 2000
will be stored by RA v2.6 as "01-03-2000" and at a new locations!
The old date coding (in CONFIG.RA) will be at the old location
and using the old format. However, the program will not read
from those dates but will write to them for compatibility.
RemoteAccess will properly extract the date as "01-03-2000"
and carry it in that format throughout the internal workings
of the program.

CAUTION *** CAUTION** * CAUTION *** CAUTION** ** FOR THE TIME BEING, the current date storage and location will also be used and updated by RA. +HOWEVER+ RemoteAccess will not read from or use that date IT WILL ONLY WRITE THAT DATE TO ASSIST 3rd PARTY UTILITIES THAT HAVE NOT YET BEEN UPGRADED!

If your program EXRACTS the date from the file, then it will display (in mm-dd-yy) as “01-03-00”

If your program has been upgraded, it will display the date, either as “01-03-00” or “01-03-2000” depending if it displays the date as a 2- or 4-digit year and what options you may have selected.

Each of YOU will need to test your utilites and programs with the data files to see HOW it handles them and if it does so to your satisfaction.

For those other programs that extract the date from the file typically EXTRACT the MONTH (from the first two characters), the DAY (from the third and forth characters) and the YEAR from the LAST two characters will not have any problem dealing with this revised format. THEY WILL, HOWEVER, run into difficulties if some dates are 19yy and others are 20yy and attempt to sort or delete based on the perceived “age” of the record! Since all dates will be assumed, BY THAT PROGRAM, to be 19yy dates. Once all dates have been transitioned into the year 2000, the user's program will believe it still to be in the 1900's.

NOTE:

Hudson Message Base

Due to the common usage of the HUDSON MESSAGE BASE (HMB) format, The date format will remain in the 'MM-DD-YY' format. However, Coding within RemoteAccess will take that into consideration and assume dates PRIOR TO (but not including) January 1st, 1950 refer to dates in the year 2000 and beyond.

At some later date, we suspect that the HMB (if continued to be used) will handle date structures in a different manner and appropriate changes to RemoteAccess will be made.

BBS User options

The BBS user will have additional date display format choices that include all the previous 2-digit year options and the addition of new 4-digit year options.

We will display the date for the SysOp as a 4-digit year formatted date.

Additional Added features

At the request of you sysops, the Default settings for the New User's Message Area, File Area and File Group will be configurable in RACONFIG in the New User Configuration section.

(end of Version 2.60's description of changes - see below for areas) (affected)

The following text is from the STRUCT.250 file

The area number to which records in the above files refer is no longer dependent upon the position of the record in the file. This is to allow utilities to move, copy, insert and delete areas and groups at will, without the need to renumber every reference to every moved area or group.

The following applies to the record structure of all the above files as published in this document.

If you look at each record definition, you will notice that the MESSAGErecord and FILESrecord structures each have 4 unused bytes at the beginning, and that the GROUPrecord structure has 2 unused bytes at the beginning.

In RA 2.60, the first two bytes of each of these records are redefined as a single word value (2 byte unsigned integer). This word holds the area number of that record - the name of the new field is 'AreaNum'. This has the following implications:

1. To get the Hudson message-base board number for a record from MESSAGES.RA, do NOT rely on its position. The AreaNum field contains the actual board number.

2. To access the correct FDB files from a file area record in FILES.RA, use the value contained in the AreaNum field, NOT the number of the position of the record.

3. Utilities are now free to move any record to any location in these files without having to worry about renumbering FDB files or menu entries.

4. When ADDING a new record, utilities MUST assign a valid area number to the record in the AreaNum field. Valid numbers are in the range 1 to 65535, and MUST be unique. See the note about index files below for information on how to determine the next available unique number.

The INDEX files

In order to provide the fastest possible access to the above four files, each is now indexed. The index files have the same path and base name with an extension of 'RDX' - for example, the index file for C:\RA\FILES.RA would be C:\RA\FILES.RDX.

The index files are automatically generated by RACONFIG after editing areas. They must also be generated by any utility which:

1. Adds, deletes or moves a record within its file, or 2. Changes the AreaNum field of any record.

FAILURE TO REINDEX WILL CAUSE UNPREDICTABLE RESULTS FOR THE SYSOP! RA and its utilities all check to make sure that the index files are up to date.

The format of the index files is very simple. It is a flat file of word (2 byte unsigned integer) records which contains one record for each AreaNum value in the corresponding file. The

physical position of each index record is equal to the AreaNum value minus 1. The value of the index record is equal to the physical position of the record for that area number in the main file PLUS 1. A value of zero indicates that the area number is not currently used.

For example:

File offest	FILES.RA AreaNum	FILES.RDX record value
0	1	1
1	2	2
2	10	0
3	11	0
4	12	0
5	13	0
6	-	0
7	-	0
8	-	0
9	-	3
10	-	4
11	-	5
12	-	6

In the above example table, to look up the correct record in FILES.RA for a particular area number, the code might look something like this in Pascal:

```
Seek(RDX, AreaNum-1);
 Read(RDX, Index);
 Seek(FILESRA, Index-1);
 Read(FILESRA, FILESRAinfo);
```

Determining the next free unique area number is a simple matter of scanning the index file until a value of zero is encountered.

*)

```
type
  AskType       = (Yes, No, Ask, Only);
  VideoType     = (Auto, Short, Long);
  MsgType       = (LocalMail, NetMail, EchoMail, Internet, Newsgroup);
  MsgKindsType  = (Both, Private, Public, ROnly, NoReply);
  OrphanType    = (Ignore, Create, Kill);
  FlagType      = array[1..4] of Byte;
  ResetType     = (Never, Week, Month, Year);
  Time          = String[5];
{}OldDate       = String[8];
{}OldLongDate   = String[9];
{}Date          = String[10];
{}LongDate      = String[11];
```

```
 ByteArray32    = Array[1..32] of Byte;
 NetAddress     = record
                   Zone,
                   Net,
                   Node,
                   Point        : Word;
                  end;
 LIMITSrecord   = record
                   Security,
                   Ltime,
                   L300,
                   L1200,
                   L2400,
                   L4800,
                   L7200,
                   L9600,
                   L12000,
                   L14400,
                   L16800,
                   L19200,
                   L38400,
                   Llocal,
                   RatioNum,
                   RatioK       : Word;
                   PerMinCost     : Real;
                   L21600,
                   L24000,
                   L26400,
                   L28800,
                   L57600,
                   L64000       : Word;
                   FlexiTime      : Real;
                   LsessionTime   : Word;
                   ResetAmt       : Word;
                   ResetPeriod    : ResetType;
                   ResetOffset    : Word;
                   L31200,
                   L33600       : Word;
                   FreeSpace      : Array[1..13] of Byte;
                  end;
LANGUAGErecord = record
                   Name         : String[20];
                   Attribute      : Byte;
```

```
            DefName,
            MenuPath,
            TextPath,
            QuesPath      : String[60];
            Security      : Word;
            Flags,
            NotFlagsMask  : FlagType;
            FreeSpace     : Array[1..190] of Byte;
          end;

MSGINFOrecord  = record
            LowMsg,
            HighMsg,
            TotalMsgs     : Word;
            TotalOnBoard  : array[1..200] of Word;
          end;

HMBMSGIDXrecord = record
            MsgNum        : Integer;
            Board         : Byte;
          end;

MSGTOIDXrecord = String[35];

HMBMSGHDRrecord= record   {not changed for y2k version}
            MsgNum        : Integer;
            PrevReply,
            NextReply,
            TimesRead     : Word;
            StartBlock    : Word;
            NumBlocks,
            DestNet,
            DestNode,
            OrigNet,
            OrigNode      : Word;
            DestZone,
            OrigZone      : Byte;
            Cost          : Word;
            MsgAttr,
            NetAttr,
            Board         : Byte;
            PostTime      : Time;
  PostDate      : OldDate;    { see note above }
            WhoTo,
            WhoFrom       : MSGTOIDXrecord;
            Subject       : String[72];
```

```
        end;
MSGTXTrecord   = String[255];
USERONrecord   = record
          Name,
          Handle       : MSGTOIDXrecord;
          Line         : Byte;
          Baud         : Word;
          City         : String[25];
          Status,
          Attribute    : Byte;
          StatDesc     : String[10];
          FreeSpace    : Array[1..98] of Byte;
          NoCalls      : Word;
         end;
         { Status byte - 0 : Browsing (in a menu)
                         1 : Uploading/downloading
                         2 : Reading/posting messages
                         3 : In a door/external utility
                         4 : Chatting with sysop
                         5 : Answering questionnaire
                         6 : RTC
                         7 : New user logon
                       255 : User-defined - display StatDesc
          Attribute   - Bit 0 : Hidden
                        1 : Wants chat
                        2 : Reserved for RANETMGR
                        3 : Do not disturb flag
                        6 : Ready (0=busy) }
LASTCALLrecord = record
          Line         : Byte;
          Name,
          Handle       : MSGTOIDXrecord;
          City         : String[25];
          Baud         : Word;
          Times        : LongInt;
          LogOn        : String[5];
          LogOff       : String[5];
          Attribute    : Byte;
         end;
        { Attribute - Bit 0 : Hidden }
FILESHDRrecord = record
```

```
            Name         : String[12];
            Size,
            CRC32        : LongInt;
            Uploader     : String[35];
            UploadDate,                {not affected by y2k}
            FileDate,                  {not affected by y2k}
            LastDL       : LongInt;
            TimesDL      : Word;
            Attrib       : Byte;
            Password     : String[15];
            KeyWord      : Array[1..5] of String[15];
            Cost         : Word;
            LongDescPtr  : LongInt;
            FreeSpace    : Array[1..20] of Byte;
          end;

 {Attrib - Bit 0 : Deleted
                1 : Unlisted
                2 : Free (don't adjust ratio) - Does NOT affect "Cost"
                3 : Not available (don't allow downloads)
                4 : Locked (no kill)
                5 : Missing/offline
                6 : No time restrictions - always allow DL
 }
FILESIDXrecord = record
            Name         : String[12];
            UploadDate   : LongInt;     {not affected by y2k}
            KeyWordCRC   : Array[1..5] of LongInt;
            LongDescPtr  : LongInt;
          end;

LASTREADrecord = array[1..200] of Word;

USERSIDXrecord = record
            NameCRC32,
            HandleCRC32  : LongInt;
          end;

COMBINEDrecord = array[1..200] of Word;

USERSrecord  = record  {Format Change only}
            Name         : MSGTOIDXrecord;
            Location     : String[25];
            Organisation,
            Address1,
            Address2,
```

```
        Address3      : String[50];
        Handle        : String[35];
        Comment       : String[80];
        PasswordCRC   : LongInt;
        DataPhone,
        VoicePhone    : String[15];
        LastTime      : Time;
{} OldLastDate : OldDate; {MM-DD-YY format}
        Attribute,
        { Bit 0 : Deleted
                1 : Clear screen
                2 : More prompt
                3 : ANSI
                4 : No-kill
                5 : Xfer priority
                6 : Full screen msg editor
                7 : Quiet mode }
        Attribute2    : Byte;
        { Bit 0 : Hot-keys
                1 : AVT/0
                2 : Full screen message viewer
                3 : Hidden from userlist
                4 : Page priority
                5 : No echomail in mailbox scan
                6 : Guest account
                7 : Post bill enabled }
        Flags         : FlagType;
        Credit,
        Pending       : LongInt;
        MsgsPosted,
        Security      : Word;
        LastRead,
        NoCalls,
        Uploads,
        Downloads,
        UploadsK,
        DownloadsK,
        TodayK        : LongInt;
        Elapsed       : Integer;
        ScreenLength  : Word;
        LastPwdChange : Byte;
```

```
            Group         : Word;
            CombinedInfo   : COMBINEDrecord;
{}      OldFirstDate,
{}      OldBirthDate,
{}      OldSubDate     : OldDate;   {continues to be MM-DD-YY}
            ScreenWidth,
            Language,
            DateFormat     : Byte;

 { DateFormat   Returns Date Name }
 {  Value      Format Picture    }
 {   1        'DD-MM-YY '      }
 {   2        'MM-DD-YY '      }

{ 3 'YY-MM-DD ' } { 4 'DD-Mmm-YY' }

{} {   5        'DD-MM-YYYY '    }
{} {   6        'MM-DD-YYYY '    }
{} {   7        'YYYY-MM-DD '    }
{} {   8        'DD-Mmm-YYYY'    }
{} { Values of 5 - 8 added at Version 2.60 }

            ForwardTo      : String[35];
            MsgArea,
            FileArea       : Word;
            DefaultProtocol: Char;
            FileGroup      : Word;
            LastDOBCheck   : Byte;
            Sex            : Byte;
            XIrecord       : LongInt;
            MsgGroup       : Word;
            Attribute3     : Byte;

             { Bit 0 : Mailbox check: scan selected areas only }

            Password       : String[15];
            PrefixLastDate,
            PrefixFirstDate,
            PrefixSubDate  : PrefixDate;
            FreeSpace      : Byte;
           end;

USERSXIrecord  = record
            FreeSpace      : Array[1..200] of Byte;
           end;

SYSINFOrecord  = record
            TotalCalls     : LongInt;
```

```
    LastCaller,
    LastHandle     : MSGTOIDXrecord;
    ExtraSpace     : array[1..92] of Byte;
  end;

TIMELOGrecord  = record {This file changes size}

{ } StartDate : Date; {changes to 4-digit year }

                    { MM-DD-KCYY format     }
    BusyPerHour    : array[0..23] of Word;
    BusyPerDay     : array[0..6] of Word;
  end;

OldTIMELOGrecord  = record    { this is a dummy record for use with }
                    { the EXITINFO.BBS file             }
    StartDate      : OldDate;       { MMKDDCYY format }
    BusyPerHour    : array[0..23] of Word;
    BusyPerDay     : array[0..6] of Word;
  end;

MNUrecord      = record {format change}
    Typ          : Byte;
    Security,
    MaxSec       : Word;
    NotFlagsMask,
    Flags        : FlagType;
    TimeLeft,
    TimeUsed     : Word;
    Age,
    TermAttrib   : Byte;

    {Bit 0 : ANSI
         1 : AVT
         2 : RIP}

    MinSpeed,
    MaxSpeed,
    Credit,
    OptionCost,
    PerMinCost   : LongInt;
    Node,
    Group        : ByteArray32;
    StartTime,
    StopTime     : Array[1..7] of Word;
    Display      : String[135];
    HotKey       : String[8];
    MiscData     : String[135];
```

```
            Foreground,
            Background    : Byte;
{}      Y2kDropFile   : Boolean;  {New field }
{}      FreeSpace     : Array[1..49] of Byte; {decremented}
          end;

EVENTrecord   = record  { Format changes }
            Status        : Byte; { 0=Deleted 1=Enabled 2=Disabled }
            StartTime     : Time;
            ErrorLevel    : Byte;
            Days          : Byte;
            Forced        : Boolean;
{}      LastTimeRun   : OldDate;  {Old 'MM-DD-YY' New 'MMKDDCYY'}
          end;

EVENTrecordArray = array[1..20] of EVENTrecord;

MESSAGErecord  = record
            AreaNum,
            Unused        : Word;
            Name          : String[40];
            Typ           : MsgType;
            MsgKinds      : MsgKindsType;
            Attribute     : Byte;

            { Bit 0 : Enable EchoInfo
                    1 : Combined access
                    2 : File attaches
                    3 : Allow aliases
                    4 : Use SoftCRs as characters
                    5 : Force handle
                    6 : Allow deletes
                    7 : Is a JAM area }

            DaysKill,   { Kill older than 'x' days }
            RecvKill      : Byte; { Kill recv msgs, recv for more than 'x' days }
            CountKill     : Word;
            ReadSecurity  : Word;
            ReadFlags,
            ReadNotFlags  : FlagType;
            WriteSecurity : Word;
            WriteFlags,
            WriteNotFlags : FlagType;
            SysopSecurity : Word;
            SysopFlags,
            SysopNotFlags : FlagType;
            OriginLine    : String[60];
```

```
    AkaAddress     : Byte;
    Age            : Byte;
    JAMbase        : String[60];
    Group          : Word;
    AltGroup       : Array[1..3] of Word;
    Attribute2     : Byte;
      { Bit 0 : Include in all groups }
    NetmailArea    : Word;
    FreeSpace2     : Array[1..7] of Byte;
  end;

GROUPrecord    = record
    AreaNum        : Word;
    Name           : String[40];
    Security       : Word;
    Flags,
    NotFlagsMask   : FlagType;
    FreeSpace      : Array[1..100] of Byte;
  end;

FILESrecord    = record
    AreaNum,
    Unused         : Word;
    Name           : String[40];
    Attrib         : Byte;
  { Bit 0 : Include in new files scan
            1 : Include in upload dupe scan
            2 : Permit long descriptions
            3 : Area is on CD-ROM
            4 : All files are FREE
            5 : Allow DLs not in FDB
            6 : Allow users to password uploads
            7 : Scan uploads }
    FilePath       : String[40];
    KillDaysDL,
    KillDaysFD     : Word;
    Password       : String[15];
    MoveArea       : Word;
    Age,
    ConvertExt     : Byte;
    Group          : Word;
    Attrib2        : Byte;
      { Bit 0 : Include in all groups }
    DefCost,
```

```
    UploadArea,
    UploadSecurity : Word;
    UploadFlags,
    UploadNotFlags : FlagType;
    Security       : Word;
    Flags,
    NotFlags       : FlagType;
    ListSecurity   : Word;
    ListFlags,
    ListNotFlags   : FlagType;
    AltGroup       : Array[1..3] of Word;
    Device         : Byte;
    FreeSpace      : Array[1..13] of Byte;
  end;

CONFrecord     = record
    Name,
    Parent         : String[8];
    Desc           : String[70];
    Attr           : Byte;
  { Bit 0 : Private
          1 : Unlisted
          2 : Global
          3 : Permanent
          4 : Use handles
                          }
    Moderator      : String[35];
    Language       : String[20];
    Password       : String[15];
    Security       : Word;
    Flags          : FlagType;
    NumNodes       : Byte;
    Active         : Array[1..250] of Byte;
    Child          : Array[1..250] of Boolean;
    NotFlagsMask   : FlagType;
    FreeSpace      : Array[1..96] of Byte;
  end;

MODEMrecord  = record     {no change at version 2.60 }
    ComPort,
    InitTries      : Byte;
    BufferSize,
    ModemDelay     : Word;
    MaxSpeed       : LongInt;
    SendBreak,
```

```
            LockModem,
            AnswerPhone,
            OffHook         : Boolean;
            InitStr,
            InitStr2,
            BusyStr         : String[70];
            InitResp,
            BusyResp,
            Connect300,
            Connect1200,
            Connect2400,
            Connect4800,
            Connect7200,
            Connect9600,
            Connect12k,
            Connect14k,
            Connect16k,
            Connect19k,
            Connect38k,
            ConnectFax      : String[40];
            RingStr,
            AnswerStr       : String[20];
            ErrorFreeString  : String[15];
            Connect21k,
            Connect24k,
            Connect26k,
            Connect28k,
            Connect57k,
            Connect64k      : String[40];
            Connect31k,
            Connect33k      : String[40];
            FreeSpace       : Array[1..100] of Byte;
          end;

ARCrecord = record
          Extension : String[3];
          UnpackCmd,
          PackCmd   : String[60];
    end;

CONFIGrecord = record
  VersionID         : Word;
  xCommPort         : Byte;
  xBaud             : LongInt;
  xInitTries        : Byte;
```

```
xInitStr,
xBusyStr          : String[70];
xInitResp,
xBusyResp,
xConnect300,
xConnect1200,
xConnect2400,
xConnect4800,
xConnect9600,
xConnect19k,
xConnect38k       : String[40];
xAnswerPhone      : Boolean;
xRing,
xAnswerStr        : String[20];
xFlushBuffer      : Boolean;
xModemDelay       : Integer;
MinimumBaud,
GraphicsBaud,
TransferBaud      : word;
SlowBaudTimeStart,
SlowBaudTimeEnd,
DownloadTimeStart,
DownloadTimeEnd   : Time;
PageStart         : Array[0..6] of Time;
PageEnd           : Array[0..6] of Time;
SeriNum,
CustNum           : String[22];
FreeSpace1        : Array[1..24] of Byte;
PwdExpiry         : Word;
MenuPath,
TextPath,
AttachPath,
NodelistPath,
MsgBasePath,
SysPath,
ExternalEdCmd     : String[60];
Address           : Array[0..9] of NetAddress;
SystemName        : String[30];
NewSecurity       : Word;
NewCredit         : Word;
NewFlags          : FlagType;
OriginLine        : String[60];
QuoteString       : String[15];
```

```
Sysop             : String[35];
LogFileName       : String[60];
FastLogon,
AllowSysRem,
MonoMode,
StrictPwdChecking,
DirectWrite,
SnowCheck         : Boolean;
CreditFactor      : Integer;
UserTimeOut,
LogonTime,
PasswordTries,
MaxPage,
PageLength        : Word;
CheckForMultiLogon,
ExcludeSysopFromList,
OneWordNames      : Boolean;
CheckMail         : AskType;
AskVoicePhone,
AskDataPhone,
DoFullMailCheck,
AllowFileShells,
FixUploadDates,
FreezeChat        : Boolean;
ANSI,                { ANSI: Yes, no, or ask new users    }
ClearScreen,         { Clear:      "                      }
MorePrompt        : AskType;   { More:       "            }
UploadMsgs        : Boolean;
KillSent          : AskType;   { Kill/Sent    "          }
CrashAskSec       : Word;      { Min sec# to ask 'Crash Mail ?'    }
CrashAskFlags     : FlagType;
CrashSec          : Word;      { Min sec# to always send crash mail. }
CrashFlags        : FlagType;
FAttachSec        : Word;      {      "    ask 'File Attach ?'    }
FAttachFlags      : FlagType;
NormFore,
NormBack,
StatFore,
StatBack,
HiBack,
HiFore,
WindFore,
WindBack,
```

```
ExitLocal,
Exit300,
Exit1200,
Exit2400,
Exit4800,
Exit9600,
Exit19k,
Exit38k          : Byte;
MultiLine        : Boolean;
MinPwdLen        : Byte;
MinUpSpace       : Word;
HotKeys          : AskType;
BorderFore,
BorderBack,
BarFore,
BarBack,
LogStyle,
MultiTasker,
PwdBoard         : Byte;
xBufferSize      : Word;
FKeys            : Array[1..10] of String[60];
WhyPage          : Boolean;
LeaveMsg         : Byte;
ShowMissingFiles,
xLockModem       : Boolean;
FreeSpace2       : Array[1..10] of Byte;
AllowNetmailReplies : Boolean;
LogonPrompt      : String[40];
CheckNewFiles    : AskType;
ReplyHeader      : String[60];
BlankSecs        : byte;
ProtocolAttrib   : Array[1..6] of Byte;
xErrorFreeString : String[15];
xDefaultCombined : array[1..25] of Byte;
RenumThreshold   : Word;
LeftBracket,
RightBracket     : Char;
AskForHandle     : Boolean;
AskForBirthDate  : Boolean;
GroupMailSec     : Word;
ConfirmMsgDeletes : Boolean;
FreeSpace4       : Array[1..30] of byte;
TempScanDir      : String[60];
```

```
ScanNow          : AskType;
xUnknownArcAction,
xFailedUnpackAction,
FailedScanAction   : Byte; {Bit 0:Mark deleted, 1:Mark unlisted, 2:Mark notavail}
xUnknownArcArea,
xFailedUnpackArea,
FailedScanArea     : Word;
ScanCmd          : String[60];
xDeductIfUnknown   : Boolean;
NewUserGroup       : Byte;
AVATAR           : AskType;
BadPwdArea         : Byte;
Location         : String[40];
DoAfterAction      : Byte; {0 = wait for CR, > 0 = wait for x seconds}
OldFileLine        : String[40];
CRfore,
CRback           : Byte;
LangHdr          : String[40];
xSendBreak         : Boolean;
ListPath         : String[60];
FullMsgView        : AskType;
EMSI_Enable        : AskType;
EMSI_NewUser       : Boolean;
EchoChar         : String[1];
xConnect7200,
xConnect12000,
xConnect14400      : String[40];
Exit7200,
Exit12000,
Exit14400          : Byte;
ChatCommand        : String[60];
ExtEd            : AskType;
NewuserLanguage    : Byte;
LanguagePrompt     : String[40];
VideoMode          : VideoType;
AutoDetectANSI     : Boolean;
xOffHook         : Boolean;
NewUserDateFormat  : Byte;
KeyboardPwd        : String[15];
CapLocation        : Boolean;
NewuserSub         : Byte;
PrinterName        : String[4];
HilitePromptFore,
```

```
HiLitePromptBack    : Byte;
xInitStr2           : String[70];
AltJSwap            : Boolean;
SemPath             : String[60];
AutoChatCapture     : Boolean;
FileBasePath        : String[60];
NewFileTag          : Boolean;
IgnoreDupeExt       : Boolean;
TempCDFilePath      : String[60];
TagFore,
TagBack             : Byte;
xConnect16k         : String[40];
Exit16k,
FilePayback         : Byte;
FileLine,
FileMissingLine     : String[200];
NewUserULCredit     : Byte;
NewUserULCreditK    : Word;
ArcInfo             : Array[1..10] of ARCrecord;
RAMGRAltFKeys       : Array[1..5] of String[60];
ArcViewCmd          : String[60];
xConnectFax         : String[40];
ExitFax             : Byte;
UseXMS,
UseEMS              : Boolean;
CheckDOB            : Byte;
EchoCheck           : AskType;
ccSec,
ReturnRecSec        : Word;
HonourNetReq        : Boolean;
DefaultCombined     : COMBINEDrecord;
AskForSex,
AskForAddress       : Boolean;
DLdesc              : AskType;
NewPhoneScan        : Boolean;
Exit21k,
Exit24k,
Exit26k,
Exit28k,
Exit57k,
Exit64k             : Byte;
TagLogoffWarning,
LimitLocal,
```

```
  SavePasswords      : Boolean;
  BlankLogins        : Byte;
  ripiconpath        : string[60];
  Exit31k,
  Exit33k            : Byte;
  IncludeNewCDareas  : Boolean;
  FutureExpansion : Array[1..513] of Byte;
end;

EXITINFOrecord = record
              Baud           : Word;
              SysInfo        : SYSINFOrecord;   { not effected  }
{}       OldTimeLogInfo   : OldTIMELOGrecord; { Century prefix below }
              UserInfo       : USERSrecord;     { Format Change }
              EventInfo      : EVENTrecord;     { Format Change }
              NetMailEntered,
              EchoMailEntered  : Boolean;
              LoginTime      : Time;
{}       OldLoginDate    : OldDate;        { Century prefix below }
              TimeLimit      : Word;
              LoginSec       : LongInt;
              UserRecord     : Integer;
              ReadThru,
              NumberPages,
              DownloadLimit   : Word;
              TimeOfCreation  : Time;
              LogonPasswordCRC : LongInt;
              WantChat       : Boolean;
              DeductedTime    : Integer;
              MenuStack      : Array[1..50] of String[8];
              MenuStackPointer : Byte;
              UserXIinfo     : USERSXIrecord;
              ErrorFreeConnect,
              SysopNext      : Boolean;
              EMSI_Session    : Boolean;      { These fields hold  }
              EMSI_Crtdef,                  { data related to an }
              EMSI_Protocols,               { EMSI session      }
              EMSI_Capabilities,
              EMSI_Requests,
              EMSI_Software   : String[40];
              Hold_Attr1,
              Hold_Attr2,
              Hold_Len       : Byte;
              PageReason     : String[80];
```

```
            StatusLine      : Byte;
            LastCostMenu    : String[8];
            MenuCostPerMin  : Word;
            DoesAVT,
            RIPmode         : Boolean;
            RIPVersion      : Byte;
{}      PrefixTimeLogInfo_StartDate, { 'KC' From TimelogRecord.StartDate }
{}      PrefixLoginDate : String[2]; { 'KC' From USERSrecord.LoginDate }
            ExtraSpace      : Array[1..81] of Byte;
          end;

PROTOCOLrecord = record
            Name          : String[15];
            ActiveKey     : Char;
            OpusTypeCtlFile,
            BatchAvailable : Boolean;
            Attribute     : Byte; { 0=Disabled, 1=Enabled }
            LogFileName,
            CtlFileName,
            DnCmdString,
            DnCtlString,
            UpCmdString,
            UpCtlString   : String[80];
            UpLogKeyword,
            DnLogKeyword  : String[20];
            XferDescWordNum,
            XferNameWordNum : Byte;
          end;
```

TELNET

Windows 10 Information

Users setting up ProBoard with GameSrv or Net2BBS on Windows XP through Windows 8.1 can skip this information.

Enable Legacy Console Mode

1. Open a command prompt, elevated command prompt, PowerShell, or elevated PowerShell window. Another way is you could also just directly right click on the console window shortcut or file, click/tap on Properties, and go to step 3 below.
2. Right click or press and hold on the title bar of the console window, and click/tap on Properties.
3. Click/tap on the Options tab, check the Use legacy console box for to turn on the Legacy Mode. This is actually needed to run any DOS BBS software or DOS Door software in Windows 10! Click/Tap on OK to apply.
4. Close and reopen the console window to apply.

This information was given by Mike Ehlert of PCMicro.com

Setting Up ProBoard with Net2BBS

NetFoss was tested with ProBoard BBS for DOS version 2.17 and higher. Replace C for your actual hard drive letter, if you did not use the C Drive.

Here is how to configure it:

1) Extract ProBoard in c:\pb

- Create directories for each node such as c:\pb\node1 and c:\pb\node2 etc. *THIS IS A MUST!*
- Your directory tree should look like this:

```
C:\PB
C:\PB\NODE1
C:\PB\NODE2
```

2) Create a RUNPB.BAT in the ProBoard Directory, which looks like this:

```
set proboard=c:\pb
cd\pb\node%1
\pb\proboard.exe -B115200 -N%1
```

- The -B115200 switch tells ProBoard to assume that the caller is already connected to the

modem at that speed. The -N%1 passes the node number, since %1 is replaced with the node number when the batch file is run.

3) Unzip the NetFoss files into a directory such as c:\netfoss\ and also unzip the included NET2BBS.ZIP into the same directory. Copy NETFOSS.DLL into the \windows\system32\ directory. This can ONLY be done from a Command Prompt which was opened using “Run As Administrator” (right click on its icon to select this) in Windows Vista and above. However this can also be done from dragging NETFOSS.DLL in Windows Explorer to the \windows\system32\ folder.

4) Configure a Telnet Server to run the NF.BAT and the RUNPB.BAT batch files. If you are using the included Net2BBS Telnet Server, then edit your Net2BBS.INI file to use a command line like this:

```
Command=c:\netfoss\nf.bat /n*N /h*H c:\pb\runpb.bat *N
StartPath=c:\pb\node*N
```

5) For maximum file transfer speed, install Public Domain Zmodem (PD ZModem) as an external protocol in ProBoard. This runs several times faster then the ProBoard internal Zmodem or FDSZ.

6) Run NET2BBS.EXE, and it is ready to accept telnet connections.

EXTERNAL PROGRAMS / DOORS

ProBoard Internally Supported DOOR Drop Files

DOOR.SYS
EXITINFO.BBS
DORINFO<node>.DEF

When configuring menu's for DOORS, you have parameters <command line> that need the full path and filename (.COM or .EXE extension included), and may contain some special codes. These codes will (at run-time) be replaced by a value or a string.

Special codes:
*D Writes a full 52-line DOOR.SYS drop file to the current directory before shelling.
*E Writes an RA 1.1x EXITINFO.BBS to the current directory before shelling and reads it back afterwards. ProBoard creates an RA 2.xx by default so if you need the older EXITINFO.BBS format be sure to specify the *E Parameter.
*M ProBoard's start-up directory (including trailing '\')
*P Com-port used by ProBoard (1-8).
*0 (zero) ProBoard will write a DORINFO1.DEF instead of a DORINFO<node>.DEF - great for running doors which require a DORINFO1.DEF file on a multi node ProBoard system.

ProBoard will soon support other Drop File formats like:

PCBOARD.SYS
DOOR32.SYS

NFU and DOOR32.SYS under ProBoard for DOS

The following was written by Mike Ehlert and John Riley, Sr.

NetFoss Utilities for Win32 Doors version 1.14

NFU allows DOS BBS software using NetFoss to run Win32 doors. NFU currently has three functions:

1. Builds a DOOR32.SYS drop file, needed by most Win32 BBS Doors. It does this by reading user data from the DOOR.SYS drop file, and also reading the Telnet Socket Handle from the Windows Environment Variable

%SOCKET%.

2. Allows a Win32 application to run from a DOS BBS Shell, which must be passed on the NFU command line.

3. It has an option to also pass keystrokes to the Console when the door launches, but this does not work for non-Console GUI doors. This can be useful for doors that ask if ANSI is supported, or asks the user for their name. It can also be used to allowing a DOS BBS to shell to a Win32 BBS that uses the same userbase.

NFU [path to DOOR.SYS] ["Door Command line + parameters in quotes"] <Opt par>

Examples:

NFU c:\proboard\node1 "c:\proboard\dark32\dark32.exe /n1"
NFU . "c:\proboard\dark32\dark32.exe /n1" (dot uses current directory)

Shortcuts:

$ - Replaced by path+\ only of DOOR32.SYS
^ - Replaced by path+\filename of DOOR32.SYS

Optional parameters:

/Q - Quiet Mode: No console output except for error messages
/R - RIP Enable: Without this, RIP Emulation in DOOR.SYS is ignored
/YN - Send Yes/No character to a door that asks user if they have ANSI
/K="x" - Stuff keyboard buffer with one character or a string of characters
Keyboard macros: |=Enter, !N=User Name, !A=User Alias, !P=Password

The second parameter can be a batch file (.BAT or .CMD), or an executable. Keyboard macros are case sensitive.

IMPORTANT INSTRUCTIONS

If you are using Net2BBS version 1.14 or later then skip the next two steps and continue on to the the "Adding your DOOR32 to your BBS" section. Net2BBS 1.14 and later automatically updates the %SOCKET% environment variable with the Telnet Socket Handle.

Changes to your telnet server command line:

If your current telnet server command line may look something like this:

Command=c:\netfoss\nf.bat /n*N /h*H c:\pb\proboard.bat *N

As you can see, the BBS batch file (in this case proboard.bat) is only being passed the node number, (using the *N macro) which it will be able to be accessed as variable %1 since it is the first parameter passed.

You will need to change the command line to also pass the Socket Handle (using the *H macro), as the second parameter being passed to proboard.bat, so it will be accessible as variable %2.

The new telnet server command line will look like this after the change:

Command=c:\netfoss\nf.bat /n*N /h*H c:\proboard\proboard.bat *N *H

Changes to the BBS batch file:

Add the following line to proboard.bat before it runs the BBS software:

set SOCKET=%2

This will store the socket handle into the environment variable %SOCKET%, which can later be accessed by NFU so the socket handle can be placed in the DOOR32.SYS drop file it creates.

Adding your DOOR32 door to your BBS:

In order for a DOS BBS to run a Win32 Door, the Door command line or batch file must be passed through NFU on the second command line parameter shown above. The Door command line must be contained within quote marks to allow spaces between any passed parameters.

Below are setup examples for the following 32-bit Windows doors:

- BBSLink
- MannIRC
- LORD32
- Darkness
- DoorMud
- Ambroshia
- Pimp Wars
- Jezibel
- Food Fight 2004

BBSLink

BBSLink is an InterBBS Game Server using a modified version of Manning's Telnet Door which automatically logs the user into their Server and directly into the desired door using the BBS Sysops pre-assigned credentials. You can join for free at http://bbsnet.net

How to install BBSNet for a DOS BBS:

1. Extract BBSlink to a directory/folder. (In this example we will assume C:\BBSLINK) and install Microsoft DotNet (.NET), which is required by the RM Library .DLL

2. Edit your BBSLINK.BAT file, and fill in the 3 codes provided by the BBSLink admin. While you are editing this batch file, you will need to add a “Change Directory” command to change to its directory somewhere before the bbslink.exe line:

CD\BBSLINK

If you are running your BBS from a different drive then BBSLink is on, then you may also want to add a “C:” command on the line above it.

3. In your BBS software, configure your door to run a Command Line such as this example, which will run NFU.EXE and pass the path-only of your BBS Nodes dropfile directory, as well as the BBSLINK.BAT that NFU will execute along with its needed parameters. Say for example, that your BBS software is ProBoard and your BBS directory structure is

C:\PB (main BBS directory)
C:\PB\NODE1 (Node 1 dropfile directory)
C:\PB\NODE2 (Node 2 dropfile directory)

ProBoard BBS uses the following Door Command Line macros:

(*#) is replaced by the Node number
(*D) tells ProBoard to create a DOOR.SYS drop file, and is filtered out.

So the following type-7 Command Line could be configured for this door in ProCfg: So the following type-7 Command Line could be configured for this door in ProCfg:

c:\netfoss\nfu.exe c:\pb\node*# "c:\bbslink\bbslink.bat lord C:\pb\node*#\DOOR32.SYS" *D

Note that for node 1, the above command line would be translated by the BBS as so:

c:\pb\nfu.exe c:\pb\node1 "c:\bbslink\bbslink.bat lord c:\pb\node1\DOOR32.SYS"

Since ProBoard and most other BBS software sets the current directory as the node's dropfile directory before executing a door, you could shorten the size of the first parameter by replacing the path to the dropfile directory shown above with just a ".", telling NFU to use the current directory to read door.sys from and to create the DOOR32.SYS in, like so:

c:\netfoss\nfu.exe . "c:\bbslink\bbslink.bat lord c:\pb\node1\DOOR32.SYS" *D

You could also shorten the second parameter by replacing the "Full path+\DOOR32.SYS" with a "^" character, like so:

c:\netfoss\nfu.exe . "c:\bbslink\bbslink.bat lord ^" *D

Note: the *D in this example is only used to tell ProBoard to create a DOOR.SYS drop file. Other BBS software will have different ways of configuring this which are often not command line macros. Many BBS programs create the DOOR.SYS by default.

Check your BBS documentation on what macro it uses to pass the node number. While Proboard uses *#, most BBS software use *N and many use other macros.

4. Once you have BBSLink working successfully with the above configuration for the LORD door, simply create additional menu options as above for the other door games available on BBSlink, replacing the word "lord" with one of the other door codes, such as lord2, teos, ooww, tw, luna, etc. A complete list of codes is available at:

http://www.bbslink.net/sysop/

Avoid these common mistakes:

- Don't leave out the quote marks on the Command Line passed to NFU. NFU requires two parameters (1: The dropfile path to the door.sys, 2: the full command line to execute), and this second parameter must be in quotes if it contains its own set of parameters.
- Do not include the name of DOOR.SYS in parameter 1, just the path where it is located NFU will read the DOOR.SYS dropfile here, and will create the DOOR32.SYS dropfile in the same directory before executing the Command Line in parameter 2.
- BBSLink requires NOT ONLY the path to a DOOR32.SYS file as stated in it's documentation, it also requires the DOOR32.SYS filename to be included in that path.

DoorParty

DoorParty is an InterBBS Game Server using a modified version of Rick Parrish of R&M Software's Telnet Door which automatically logs the user into their Server and directly into DoorParty using the BBS Sysops pre-assigned credentials. You can join for free at

http://wiki.throwbackbbs.com/doku.php?id=start.

How to install DoorParty for ProBoard:

Before installing DoorParty, make sure you register for an account. Here is the information from DoorParty on how to join from their website:

- If you're looking to sign up to DoorParty, I've moved the application process to the BBS. I figured this gives it a more “authentic” feel to the spirit of DoorParty, and also simplifies the entire thing for everyone.

- You can find it here using the fTelnet app on the BBS website http://www.throwbackbbs.com/, or telnet://bbs.throwbackbbs.com with your favorite installed telnet terminal.

- Once you create an account on the BBS, the application form and info can be found on the main menu as: # % apply for DoorParty!

- Logon to Throwback and hit '#' from the main menu

Once you have been accepted to DoorParty, create and download your custom install package:

- In order to get your custom package, please log onto Throwback BBS (telnet://bbs.throwbackbbs.com) where you signed up. If you've been approved, you'll have a new menu option in the DoorParty application main menu that says:

- –>.P. Custom Connect Package

- You can download your custom package there.

1. Extract the DoorParty zip file (dpConnect_windows.zip) that you downloaded from Throwback at http://www.throwbackbbs.com/ to a directory/folder. (In this example we will assume C:\dpConnect_windows).

2. Create a folder: c:\doorparty and copy the following files from c:\dpConnect_windows into it:

- c:\dpConnect_windows\bbs_files*.* (All files)
  - DoorParty.exe
  - RMLib.dll
  - <Your System Code>_DP.bat
- c:\dpConnect_windows\doorparty-connector*.* (All files)
  - doorparty-connector.exe
  - doorparty-connector.ini

3. You will not have to edit ANY DoorParty files if you properly configured your DoorParty customized installer files.

4. In ProCfg, configure your door to run a Command Line such as this example, which will run

NFU.EXE and pass the path-only of your BBS Nodes dropfile directory, as well as the BBSLINK.BAT that NFU will execute along with its needed parameters. Say for example, that your BBS software is ProBoard and your BBS directory structure is

C:\PB (main BBS directory)
 C:\PB\NODE1   (Node 1 dropfile directory)
 C:\PB\NODE2   (Node 2 dropfile directory)

ProBoard BBS uses the following Door Command Line macros:

(*#) is replaced by the Node number
 (*D) tells ProBoard to create a DOOR32.SYS drop file, and is filtered out.
 (*U) tells ProBoard to use the user's handle.
 Replace <your system code> with the code you were assigned.
 The batch file should already be named correctly in the zip file.

So the following type-7 Command Line could be configured for this door in ProCfg:

c:\netfoss\nfu.exe c:\pb\node*# "c:\doorparty\<your system code>_dp.bat *# *U" *D

Note that for node 1, the above command line would be translated by the BBS as so:

c:\pb\nfu.exe c:\pb\node1 "c:\doorparty\<your system code>_dp.bat 1 Massacre c:\pb\node1\DOOR32.SYS"

Note: the *D in this example is only used to tell ProBoard to create a DOOR32.SYS drop file.

Avoid these common mistakes:

- Don't leave out the quote marks on the Command Line passed to NFU. NFU requires two parameters (1: The dropfile path to the door.sys, 2: the full command line to execute), and this second parameter must be in quotes if it contains its own set of parameters.
- Do not include the name of DOOR32.SYS in parameter 1, just the path where it is located NFU will read the DOOR.SYS dropfile here, and will create the DOOR32.SYS dropfile in the same directory before executing the Command Line in parameter 2.
- DoorParty requires NOT ONLY the path to a DOOR32.SYS file, it also requires the DOOR32.SYS filename to be included in the node path.

MannIRC Door

MannIRC is a Bitch-X style IRC client for BBS systems by Rick Parrish of R&M Software, and is available from http://randm.ca

Here is how to install it:

1. Extract MANNIRC.RAR to a directory/folder. (In this example we will use C:\PB\MANNIRC)

2. Open MannIRC.ini in your favorite text editor, and edit the options as described in MANNIRC.HTML

3. In your BBS software, configure your door to run a Command Line such as this example,

which will run NFU.EXE and pass the path-only of your BBS Nodes dropfile directory, as well as the MANNIRC.EXE that NFU will execute along with its needed node parameter. Say for example, that your BBS software is ProBoard and your BBS directory structure is:

C:\PB (main BBS directory)
C:\PB\NODE1 (Node 1 dropfile directory)
C:\PB\NODE2 (Node 2 dropfile directory)

ProBoard BBS uses the following Door Command Line macros:

- # is replaced by the Node number
- D tells ProBoard to create a DOOR.SYS drop file, and is filtered out.

So the following type-7 Command Line could be configured for this door in ProCfg:

c:\netfoss\nfu.exe c:\pb\node*# "c:\pb\mannirc\mannirc.exe c:\pb\node*#\door32.sys" *D

Note that for node 1, the above command line would be translated by the BBS as so:

c:\netfoss\nfu.exe c:\pb\node1 "c:\pb\mannirc\mannirc.exe c:\pb\node1\door32.sys"

If your BBS software automatically sets the current directory as the node's dropfile directory before executing a door, you can shorten the size of the first parameter replacing the path to the dropfile directory shown above in parameter 1 with just a ".", telling NFU to use the current directory to read door.sys from and to create the door32.sys in, like so:

c:\netfoss\nfu.exe . "c:\pb\mannirc\mannirc.exe c:\pb\node*#\door32.sys" *D

To make the second parameter shorter, you can use the "^" shortcut character to replace the path and filename of the DOOR32.SYS like so:

c:\netfoss\nfu.exe . "c:\pb\mannirc\mannirc.exe ^" *D

Darkness

Darkness was one of the first DOOR32 door games released. It is a LORD style game, by Jack Phlash of Demonic, and is available from Distortion BBS at http://d1st.org. Here is how to install it:

1. Extract DRK100B3.ZIP the D112001.ZIP update to a directory/folder. In this example we will assume C:\PB\DARK32

2. Run DARKCFG.EXE, and select [c] BBS and modem config. Use the "[" and "]" keys to go forward and back one node, and for each node page, you will need to select "[d] Drop file directory" and change it to the correct directory for your BBS. For example, C:\PB\NODE1\ for Node 1 once you have configured all your nodes, press [q] twice to quit and save changes.

3. In your BBS software, configure your door to run a Command Line such as this example, which will run NFU.EXE and pass the path-only of your BBS Nodes dropfile directory, as well as the DARK32.EXE that NFU will execute along with its needed node parameter. Say for example, that your BBS software is ProBoard and your BBS directory structure is:

C:\PB (main BBS directory)
C:\PB\NODE1 (Node 1 dropfile directory)
C:\PB\NODE2 (Node 2 dropfile directory)

ProBoard BBS uses the following Door Command Line macros:

- # is replaced by the Node number
- D tells ProBoard to create a DOOR.SYS drop file, and is filtered out.

So the following type-7 Command Line could be configured for this door in ProCfg:

c:\netfoss\nfu.exe c:\pb\node*# "c:\pb\dark32\dark32.exe /N*#" *D

Note that for node 1, the above command line would be translated by the BBS as so:

c:\netfoss\nfu.exe c:\pb\node1 "c:\pb\dark32\dark32.exe /N1"

If your BBS software automatically sets the current directory as the node's dropfile directory before executing a door, you can shorten the size of the first parameter by replacing the path to the dropfile directory shown above in parameter 1 with just a ".", telling NFU to use the current directory to read door.sys from and to create the door32.sys in, like so:

c:\netfoss\nfu.exe . "c:\pb\dark32\dark32.exe /N1" *D

DoorMud

DoorMud - Land of the Forgotten is a Multi User Dungeon (MUD) game by Evan Elias, and it's available from http://dmud.thebbs.org

Here is how to install it:

1. Extract the DOOR32 version of DoorMud (DMUD099D.ZIP) to a directory/folder.

(In this example we will assume C:\PB\DMUD)

2. You must first run the door in local mode by executing "dmud32d.exe -l". The game will automatically finish its installation the first time you run it in local mode.

3. In your BBS software, configure your door to run a Command Line such as this example, which will run NFU.EXE and pass the path-only of your BBS Nodes dropfile directory, as well as the DMUDD32.EXE that NFU will execute along with its needed parameters. Say for example, that your BBS software is ProBoard and your BBS directory structure is

C:\PB (main BBS directory)
C:\PB\NODE1 (Node 1 dropfile directory)
C:\PB\NODE2 (Node 2 dropfile directory)

ProBoard BBS uses the following Door Command Line macros:

- # is replaced by the Node number
- D tells ProBoard to create a DOOR.SYS drop file, and is filtered out.

So the following type-7 Command Line could be configured for this door in ProCfg:

c:\netfoss\nfu.exe c:\pb\node*# "c:\pb\dmud\dmudd32.exe -N *# -D c:\pb\node*#" *D

Note that for node 1, the above command line would be translated by the BBS as so:

c:\netfoss\nfu.exe c:\pb\node1 "c:\pb\dmud\dmudd32.exe -N 1 -D c:\pb\node1"

If your BBS software automatically sets the current directory as the node's dropfile directory before executing a door, you can shorten the size of the first parameter by replacing the path to the dropfile directory shown above in parameter 1 with just a ".", telling NFU to use the current directory to read door.sys from and to create the door32.sys in, like so:

c:\netfoss\nfu.exe . "c:\pb\dmud\dmudd32.exe -N *# -D c:\pb\node*#" *D

You can also shorten the second parameter by replacing the path (only) to the DOOR32.SYS file (without the filename) with the "$" character like so:

c:\netfoss\nfu.exe . "c:\pb\dmud\dmudd32.exe -N *# -D $" *D

Ambroshia

Ambroshia - The Test of Time is a Multi User Dungeon (MUD) game by Dominic Carretto, and it's available from http://ambroshia.thebbs.org

Here is how to install it: (Assuming that your BBS uses *# as the node macro)

1. Extract the Win32 version of Ambroishia (AMB4031.ZIP) to a directory/folder.

(In this example we will assume C:\PB\AMBV4)

2. In your BBS software, configure your door to run a Command Line such as this example:

c:\netfoss\nfu.exe c:\pb\node*# "c:\pb\amb\amblaunch.exe /n *# /d c:\pb\node*#\door32.sys" *D

Note that for node 1, the above command line would be translated by the BBS as so:

c:\netfoss\nfu.exe c:\pb\node1 C:\PB\amb\amblaunch.exe /n 1 /d c:\pb\node1\door.sys"

If your BBS software automaticly sets the current directory as the node's dropfile directory before executing a door, you can shorten the size of parameter 1 like this:

c:\netfoss\nfu.exe . "c:\pb\amb\amblaunch.exe /n *# /d c:\pb\node*#\door32.sys" *D

You can also shorten the second parameter by replacing the path (only) to

c:\netfoss\nfu.exe . "c:\pb\amb\amblaunch.exe /n *# /d ^" *D

Pimpwars

PimpWars is another classic RPG in which you play a pimp. It was updated to Win32 by James Coyle of MysticBBS fame.

James left one annoying "feature" from the original game, in which it ignores the ANSI graphics

setting found in the drop file, and instead asks the user “Use ANSI? (Y/N)” when it start up. NFU can automaticlly answer this question, so it's never even seen by the user by adding the /YN switch as the third parameter.

Here is how to install it: (Assuming that your BBS uses *# as the node macro)

1. Extract PW_153W.ZIP to a directory/folder. In this example we will assume C:\PB\PIMPWARS)

2. In your BBS software, configure your door to run a Command Line such as this example:

c:\netfoss\nfu.exe c:\pb\node*# "c:\pb\pimpwars\pimpwars.exe c:\pb\node*#\door32.sys *#" /YN *D

Note that for node 1, the above command line would be translated by the BBS as so:

c:\netfoss\nfu.exe c:\pb\node1 "c:\pb\pimpwars\pimpwars.exe c:\pb\node1\door32.sys 1" /YN *D

If your BBS software automaticly sets the current directory as the node's dropfile directory before executing a door, you can shorten the size of parameter 1 like this:

c:\netfoss\nfu.exe . "c:\pb\pimpwars\pimpwars.exe c:\pb\node*#\door32.sys *N" /YN *D

You can also shorten the second parameter by replacing the path+/DOOR32.SYS with “^”:

c:\netfoss\nfu.exe . "c:\pb\pimpwars\pimpwars.exe ^ *#" /YN *D

Jezabel

Jezabel is an RPG in which you are a virus and battle other malware written by DreamMaster. It's available from http://dreamlandbbs.org.

Here is how to install it: (Assuming that your BBS uses *# as the node macro)

1. Extract JZBL25.ZIP to a directory/folder. In this example we will assume C:\pb\jez)

2. In your BBS software, configure your door to run a Command Line such as this example:

c:\netfoss\nfu.exe c:\pb\node*# "c:\pb\jez\jezw32.exe c:\pb\node*#\door32.sys" *D

Note that for node 1, the above command line would be translated by the BBS as so:

c:\netfoss\nfu.exe c:\pb\node1 C:\pb\jez\jezw32.exe c:\pb\node1\door.sys"

If your BBS software automatically sets the current directory as the node's dropfile directory before executing a door, you can shorten the size of parameter 1 like this:

c:\netfoss\nfu.exe . C:\pb\jez\jezw32.exe c:\pb\node*#\door.sys"

You can also shorten the second parameter by replacing the path+/DOOR32.SYS with “^”:

c:\netfoss\nfu.exe . C:\pb\jez\jezw32.exe ^" *D

Food Fight 2004

Food Fight 2004 was a Win32 remake of the Classic DOS Food Fight door, written by Chris

Martino.

1. Extract FF2K4V24.ZIP to a directory/folder. In this example we will assume C:\PB\FOOD)

2. In your BBS software, configure your door to run a Command Line such as this example:

c:\netfoss\nfu.exe c:\pb\node*# "c:\pb\food\ff2k4.exe -Dc:\pb\node*#\door32.sys" *D

Note that for node 1, the above command line would be translated by the BBS as so:

c:\netfoss\nfu.exe c:\pb\node1 "c:\pb\food\ff2k4.exe -Dc:\pb\node1\door32.sys"

If your BBS software automatically sets the current directory as the node's dropfile directory before executing a door, you can shorten the size of parameter 1 like this:

c:\netfoss\nfu.exe . "c:\pb\food\ff2k4.exe -Dc:\pb\node*#\door32.sys" *D

You can also shorten the second parameter by replacing the path+/DOOR32.SYS with “^”:

c:\netfoss\nfu.exe . C:\pb\jez\jezw32.exe ^" *D

(Leave out the *D if not using ProBoard - this tells ProBoard to create a DOOR.SYS)

Other Win32 doors that were tested but not found to be worthwhile documenting

- **MyCroft doors:** sherlock Holmes, Ficitois Stock Exchange, Steller Quest, U-Boat.
- **Atlantis doors:** Kentucky Derby, Vegas Slots
- Tournament Trivia (from DoorMud author)
- KFTO Boxing
- BBS Simulator (Zoob)
- Usurper - Great classic game but the Win32 version is still in beta and it requires upgrading from other versions currently.

Win32 doors that appear not to work

The Online Pub for Windows by Paul Sidorsky only supports Windows COM port handles, and not TCP Sockets, so it will not work.

The Telnet Door by Enigma/Dink released with his Citrisoft kit appears to only support Windows COM Port handles, and not TCP Sockets, so it will not work.

The Telnet Door by Mercyful Fate will get as far as making a remote connection, but once it displays the remote logon screen, it will disconnect (or perhaps the remote BBS disconnects due to a telnet negotiation issue). If anyone is able to get this to work correctly, please let me know.

The only telnet door that appears to work so far is the one by Rick at R&M, which uses settings simular to the one for BBSLink without needing a batch file.

Setup ProBoard with GameSrv

1.) First thing to do is install ProBoard just like the help/docs in the archive tell you. I believe you just unpack the ZIP into any directory of your choice. Run the INSTALL.EXE and follow the prompts!

2.) Next click on the GameSRV setup executable! However make sure you uncheck the FOSSIL drivers and the DOORS!

3.) Next install NetFoss 1.X into the same directory you told GameSRV to install into. However, you have to copy or move NETFOSS.DLL into C:\WINDOWS\SYSTEM32 under Windows XP or higher!

4.) Now create a RUNBBS.BAT to look just like this:

```
@ECHO OFF
REM IF YOU DID NOT INSTALL INTO C:\GAMESRV THEN YOU WILL HAVE TO CHANGE
REM THE THREE LINES BELOW WHICH MAKE REFERENCE TO THAT DIRECTORY
REM DO _NOT_ CHANGE ANYTHING ELSE
C:
C:\GAMESRV\NETFOSS.COM /n%1
IF ERRORLEVEL 1 GOTO END
C:\GAMESRV\NETCOM.EXE /n%1 /h%2 C:\GAMESRV\BBS.BAT %1
C:\GAMESRV\NETFOSS.COM /U
:END
```

5.) Now edit/create you a BBS.BAT. It should look like this:

```
@ECHO OFF
REM IF YOU DID NOT INSTALL INTO C:\PB THEN YOU WILL HAVE TO CHANGE
REM THE THREE LINES BELOW WHICH MAKE REFERENCE TO THAT DIRECTORY
REM DO _NOT_ CHANGE ANYTHING ELSE
C:
C:\WINDOWS\SYSTEM32\SHARE.EXE /f:4096
SET PROBOARD=C:\PB
cd\PB\node%1
C:\PB\PROBOARD -n%1 -b115200
:END
```

6.) Now edit GameSRV's General Information Tab under the Configuration Editor…

7.) Click the SAVE Button!

8.) Now test it out and see if it works! Load a program such as SyncTERM, Hyperterminal, Qmodem Pro for Win95/NT, or even mTel32! Tell any of these programs to connect to your ISP Address, your Domain Name, or Sub-Domain Name. Which ever one is pointed to your computer with ProBoard and GameSrv on it. Your IP address will always be pointed to your computer!

9.) If for some reason you have problem with echo when signing on with one of the programs above, then go into ProCFG and edit the modem Init String for each Node you have, to read this:

ATH0 S1001=1 S1002=1 S1003=7 S1005=0 S1008=23 &D2H0S0=1|

That should be all you need to install ProBoard with GameSrv and NetFOSS! You can download latest version of GameSrv at https://www.randm.ca/