INTRODUCTION 3
USAGE AGREEMENT 4
BACK-UP AND TRANSFER 4
COPYRIGHT 4
LIMITED WARRANTY ON DISK 4
PROGRAM AND MANUAL 5
LIMITATIONS OF REMEDIES 5
UPDATE AND CUSTOMER SUPPORT POLICY 5
ACKNOWLEDGEMENT 5
USER SUPPORT 5

SYSTEM CREATION 6

BOOT FILE/SYSTEM CONFIGURATION 12

SYSTEM FILE DESCRIPTIONS 18

MAIN SYSTEM COMMANDS 26

STACKED COMMANDS 27

WAITING FOR CALL SCREEN FUNCTIONS 28

FUNCTION KEYS 28

MAIL SYSTEM 29

MESSAGE BASE COMMANDS 29
GENERAL 29
SUB-OP/REMOTE ACCESS COMMANDS 31

UPLOAD/DOWNLOAD SYSTEM 31
GENERAL 31
UD-OP/REMOTE ACCESS COMMANDS 33

SYSOP/REMOTE MAINTENANCE SYSTEM 34

MINI-TERMINAL MODE SYSTEM 36

MODULAR TERMINAL PROGRAM 37

MODULAR SUBS 38

MODULAR UDS 38

MODULAR REMOTE COMMAND MAINTENANCE 38

MODULAR NETWORKING 38

MESSAGE MAKER 38
PERIOD COMMANDS 39
CTRL COMMANDS 39

MCI COMMANDS 40
OUTPUT 40
INPUT 40
FUNCTION 41
COLOR 42
SYSOP 42

STATS FILE DESCRIPTION 44

PROGRAMMING C_BASE AND MODULES 45

OUTPUT TO MODEM AND SCREEN 45
INPUT FROM MODEM AND SCREEN 46
PRINTING A FILE TO MODEM/SCREEN 47
OUTLINE OF V3.0 BBS PROGRAM 47
MODULE KIT 48
VARIABLES COMMONLY USED IN CBASE 49

CONVERTING MODULES 50
FROM V2.0 OF C_BASE 50
FROM OTHER BBS SYSTEMS 50

APPENDIX A - DRIVE SETUP 51
HOW TO ACCESS LOGICAL DRIVES FROM WITHIN THE BBS 52

APPENDIX B - ACCESS GROUP SETUP 54
CHANGING ACCESS GROUP PARAMETERS IN THE BOOT 54
SPECIFYING ACCESS CODES FOR ACCESS GROUPS 56
SECONDARY SECURITY LEVEL 56
REMOTE ACCESS LEVELS EXPLAINED IN DETAIL 58

APPENDIX C - TEXT PROMPT EDITING IN THE BOOT 59

APPENDIX D - HARDWARE CONSIDERATIONS 61
PRINTERS 61
1541 61
1571 62
1581 & 1581 PARTITIONING 62
RAM EXPANDERS (17XX SERIES) 64
SWIFTLINK CARTRIDGE/2400/9600+ BAUD 64
2400 BAUD/HAYES COMPATIBLE MODEM SETUP 66
XETEC LT. KERNAL HARDDRIVES 66
CMD HARDDRIVES 67
TURBOMASTER CPU 4.09 MHZ CARTRIDGE 68
ICT HARDDRIVES 69
USING JIFFYDOS 70
SFD 1001, D9060 PET HARDDRIVES, PARALLEL INTERFACES 70
D9060 AND OTHER COMMODORE PET HARDDRIVES 71
OTHER HARDWARE 71

APPENDIX E - MODULES SETUP 71
Empire 71
Murder Motel 71
Slots 71
ModsPak 71
UserRank 71
Little Shop Of Horrors 71

APPENDIX F - TROUBLESHOOTING AND BUG REPORTING 72
COMMON QUESTIONS 72
REPORTING BUGS 76

APPENDIX G - TRADEMARK ACKNOWLEDGEMENTS 77

NOTE: DO THE FOLLOWING THREE THINGS BEFORE GOING ANY FURTHER!

1: MAKE AN ARCHIVAL BACKUP OF THE DISKS! WHEN DOING THIS TRY NOT TO USE FAST HACKEM TYPE COPIERS AS THEY PROVE TO BE SOMEWHAT UNRELIABLE. TRY USING COPY-ALL PROVIDED ON THE DISK. WE ALL KNOW HOW ANNOYING IT IS WHEN THE CAT SPILLS THE SOFT DRINK ON THE DISK WE JUST PAID GOOD MONEY FOR!

2: READ THE DOCUMENTATION THOROUGHLY! READ IT ALL THE WAY THROUGH ONCE BEFORE SETTING UP THE SYSTEM. THIS WILL MAKE IT EASIER TO UNDERSTAND SETTING UP THE SYSTEM WHEN IT IS FINALLY DONE.

3: REMEMBER THE DONGLE (MECHANICAL PROTECTION DEVICE) BELONGS IN JOYSTICK PORT #2. THIS IS ONLY NECESSARY WHEN THE BBS IS ACTUALLY RUNNING AND ACCEPTING CALLERS.

WHO IS THIS DOCUMENTATION WRITTEN FOR?

The documentation for C_BASE 64 is written with the experienced user who has a decent working knowledge of Commodore DOS Commands and telecommunications in mind. No attempt is made to explain fundamental items, such as the following: What is a BBS? What is a sequential file? What is Punter? What is a Device #? How does a person scratch files? How can a person copy files from disk drive to disk drive? And so on. If the reader of this documentation does not have experience with the above subjects then it is advised that they not run C_BASE until they learn more about these elementary subjects. To gain such knowledge, a person may read the owner manuals for the disk drives, harddrives, modem, and any other equipment to be used on the BBS.

INTRODUCTION

Welcome to C_BASE v3.0! This program is the culmination of many years of work and effort not only from the author but from all the people that have supported C_BASE over the years. From modifying other BBS programs a long time ago, to 1.0s release many years ago, to 2.0s creation last year, and to 3.0s release in 1991, C_BASE has always progressed at a steady and fast pace! And now this is a result of that work.

Before getting into setting up the system, a few things should be said. First as stated earlier, read the documentation thoroughly! The documentation was not written so that it could be haphazardly skimmed. A good sysop always wants to understand what he is running anyway so please READ THIS DOCUMENTATION.

Most of the documentation is necessary to read. There are, however, parts of the documentation that may not be for the average person. These are parts that deal with modifying the basic, making modules, and other programming tasks. If a part of the documentation is not necessary to read but is there for informational purposes to programmers, it will be stated clearly at the beginning of that section.

If in the unlikely event that an error is ever encountered in the program contact me as soon as possible. This reporting of what is found in the program is vitally important. It is not only a responsibility but also an obligation as an owner to tell us of the quirks that occur so that they may be fixed for the benefit of not only one owner but others that run the program either in the present or the future. In fact, this documentation has been revised over ten times in an effort to provide a better understanding of the system to future owners already!

To set-up the system for the very first time, follow the instructions in the System Create and the Boot Configuration sections of this documentation.

PLEASE READ THE FOLLOWING WARRANTY AND PRODUCT INFORMATION SECTIONS!

USAGE AGREEMENT

You have the non-exclusive right to use the enclosed program. It may only be used for accepting incoming calls on a single computer at a time. Distributing the dongle, copies of the program, or documentation is not allowed. You may modify or translate the program for personal use only, and only in accordance with all other parts of this agreement. Copying the program with the intention of distributing it to others, whether or not for personal gain is illegal, and not in accordance with this agreement. Modifications may be distributed to other registered owners of C_BASE only.

BACK-UP AND TRANSFER

You may make a back-up of the program disk for back-up purposes only. The copyright notice must reside on all back-up copies of C_BASE.

If you are planning on a transfer of your registration to another party, you must notify me first and tell me whom you plan on giving or selling the BBS to. The other party must then submit a written request including the following:

(1) New owner's real name, alias, address, voice phone number, BBS phone number, and equipment listing in accordance with all the requested registration information.
(2) A transfer fee of $25.00 US Dollars.

If you transfer the program you must supply the new owner with your documentation booklet and you must destroy your backups or give the new owner your backups of the program.

If you transfer this software you will not receive any updates or support anymore since your name will be purged from our owner listings. Any updates that you paid for will go to the new owner instead.

COPYRIGHT

This program and all its documentation are copyrighted under the laws of the United States government. Copies of the program and its documentation except for back-up purposes or as a spare workdisk used to make and test your personal modifications on a spare computer are not allowed. **ANY COPYRIGHT NOTICES ON THE PROGRAM OR DISK MAY NOT BE REMOVED AT ANY TIME.**

LIMITED WARRANTY ON DISK

The disks upon which the C_BASE programs are furnished are warranted to be free from defects in materials and workmanship under normal use, for a period of ninety (90) days from the date of delivery as recorded in shipping records. To obtain a warranty service or replacement, the disk must be delivered pre-paid.

EXCEPT TO THE EXTENT PROHIBITED BY APPLICABLE LAW, ANY IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE ON THE DISK IS LIMITED TO THE DURATION OF THIS LIMITED WARRANTY.

PROGRAM AND MANUAL

The C_BASE manual and program are provided without warranty of any kind, either expressed or implied, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. Furthermore, the quality, correctness, accuracy, and reliability of the program are not warranted or otherwise guaranteed. The program's operation is not warranted nor guaranteed to be uninterrupted or error free.

LIMITATIONS OF REMEDIES

In no event will C_BASE BBS SYSTEMS be liable for any damages in excess of the license fee paid, including, without limitation, any lost profits, business goodwill or other special, incidental, or consequential damages arising out of the use or inability to use the program, or for any claim by any other party, even if C_BASE BBS Systems or the dealer has been advised of the possibility of such claims or damages. This warranty gives specific legal rights to the registered sysop and may also give other rights which vary from state to state.

UPDATE AND CUSTOMER SUPPORT POLICY

In order to be able to obtain any customer support or updates of the program, complete and return the enclosed registration card to C_BASE BBS Systems. If the registration information has not been provided or if C_BASE BBS Systems is aware of a breech of any part of this agreement, C_BASE BBS Systems is under no obligation to make available any customer support or updates to the program even though payment of any applicable fees have previously been made.

If there are any questions concerning this agreement, please contact CBASE BBS Systems as soon as possible IN WRITING.

ACKNOWLEDGEMENT

You acknowledge that you have read this agreement, understand it, and agree to be bound by its terms and conditions by completing and returning the enclosed registration card, or by your use of this software. You also agree that this agreement is the complete and exclusive statement or agreement between the parties and all other prior proposals and prior agreements, verbal or written, are superceded by this present agreement.

USER SUPPORT

User support may be obtained through writing the author at

Gunther Birznieks
7007 Georgia Street
Chevy Chase, Maryland 20815

Or by calling one of our information BBSes.

The Orchard 300/1200/2400 Baud
301-654-3228 (Washington DC Suburbs) 24 Hours
Sysop: Gunther Birznieks (Prof. Plum)

Vulgar Unicorn 300/1200/2400 Baud
301-323-WOLF (Baltimore Area) 24 Hours
Sysop: Kenneth Estes (Lone Wolf)

Please note that none of these systems is obligated to give access to CBASE Information areas. Such access is granted as a privilege, not a right, so please maintain a regard for the rules of the system while on each particular system. Such system rules may generally be found in a section of one of the libraries on the BBS.

In situations where it is impossible to contact the author via feedback on the main BBS then call C_BASE BBS Systems voice at 301-654-2935. In general though, feedback on the BBS is the preferred method of contact because then the communication does not rely on having two people with differing schedules connecting at the same time. Also, it is easier to explain things about a system verbosely in feedback. Read Me files are also generated periodically on the BBS to explain things that may have been brought up by users previously. It is a good idea to check the read me files for the answer to a question before leaving feedback.

SYSTEM CREATION

When starting up C_BASE v3.0 system for the very first time, load up the file called "system create" that resides on the C_BASE owners disk.

The system create file will create a stats file and userlog after it has finished asking the questions it needs to know about how the system should be configured.

Please read the sections of this documentation concerning **DRIVE SETUP** and **ACCESS GROUPS** because the setup sections of this documentation rely partly on this knowledge.

The first set of questions in the system create ask some fairly general questions about system setup. The system create asks what alias to have for the sysop of the system, and the current date in MM/DD/YR (Month/Day/Year) format.

Next, the configuration the sub boards, email, libraries and modules needs to be created. The defaults are device 8, drive #0, and DOS initialization command as -. The DOS Init command should be changed only if the BBS utilizes partitioning on the 1581, ICT HardDrive, CMD HardDrive, or any other drive that partitions via DOS commands. They may be changed also if the BBS uses double sided mode of a 1571 drive.

When entering a drive number always remember to include a ":" after the drive number. Also, since the prompts default to device 8, drive #0, and DOS command -, if these are changed do not merely additionally type the new information. Delete the old information first. For example at a prompt like

Drive #: 0:

Do not add 1: and press [RETURN]. This will result in the drive # becoming "0:1:". Delete the original 0: if 1: or another drive # is desired before typing the new drive #. Remember to add a ":" after the drive #.

LOGICAL DRIVE ASSIGNMENT WORKSHEET

LD#	SECTION NAME	DEVICE #	DRIVE #	DOS INIT COMMAND
0:	System Files			
1:	Temporary			
2:	Sub Files			
3:	Mail Files			
4:	Library File			
5:	Module Files			
6:	UD Section 1			
7:	UD 2			
8:	UD 3			
9:	UD 4			
10:	UD 5			
11:	UD 6			
12:	UD 7			
13:	UD 8			
14:	UD 9			
15:	UD 10			
16:	UD 11			
17:	UD 12			
18:	UD 13			
19:	UD 14			
20:	UD 15			
21:	UD 16			
22:	UD 17			
23:	UD 18			
24:	UD 19			
25:	UD 20			
26:	UD 21			
27:	UD 22			

Next, the program will ask to define the number of subs and subnames and security levels to access each sub. For the sub section, the system create will ask how many messages to have stored per message packet. A good number is five or ten. It should not be above 10 because this will slow down the board operation more than providing any file space advantage.

To explain what message packets are, the messages on C_BASE are generally not stored individually on the disk. They are stored in packets. When the system is first set up with the system create file, the number of messages per sequential file packet must be specified.

If a person chooses one message per packet in the system create, the message base will behave like a normal BBS without a message packet system. That is, when one message is stored per packet that means that there is now going to be only one message stored in a given sequential file.

The messages are stored on the packet and separated by markers followed by a minimal amount of information that is revealed in the message header when the file is read. The character that marks the separation of messages is chr$(255) or a "checkerboard" type character. Since the message header is stored in a compacted mode, this greatly increases scanning speed by as much as third over any other BBS that uses the sequential scanning method of storing messages.

The number that the system create asks for as the number of messages per packet is the amount of messages that is stored per sequential file on disk for each sub. If "5" is chosen then five messages will be stored on one sequential file before the BBS creates another message packet.

If the system is going to run the messages off of a slow serial drive like an unmodified 1541 or 1581, five (5) is an ideal message packet number. If the system is running on something faster such as a harddrive, then ten (10) will do equally well.

The maximum number of messages per sub board is directly related to the previously described message packet number. The maximum number of messages per sub must be a multiple of the message packet number minus one. For example, if the number of messages stored per sequential file is five then the only valid maximum number of messages to store on that sub would be 9,14,19,24,29, and so on. Keep in mind the limitations of the hardware that the messages will run off of such as the maximum number of files that can reside on the disk and the total number of blocks free. If the system is configured for five messages per packet, an estimate can be made of how much space the system needs for messages by assuming that each message packet will take up about ten blocks when full.

For each sub the access groups that can read each sub and post on each sub will need to be configured. This is done through a series of graphics menus in the system create.

SUB BOARD SETUP WORKSHEET

NUMBER OF PACKETS PER MSG		Example: 5		
SUB #	SUB NAME	GROUPS TO READ	GROUPS TO WRITE	MAX MSGS
0	SAMPLE(NO 0th SUB)	2;3;4;5;14	3;4;5;14	29
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				
21				
22				
23				
24				
25				
26				

The next section of the system create deals with setting up the upload download section and how many directories should be in the system. For each directory, the system create asks to name each, define the access group code needed to access each, the device #, drive #, and finally the DOS INIT command for each UD section. The access groups are defined using the same type of graphic menus used in defining the sub board access groups except that upload access, download access, and unlimited block credits access are defined separately.

UD SECTION SETUP WORKSHEET

UD#	UD NAME	GROUPS TO UPL	GROUPS TO DOW	W/UNLTD CREDITS
0	Sample	2;3;4;10	4;10	10
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				
21				
22				
23				
24				
25				
26				
27				

As a reminder, the DOS INIT command for each UD section is mostly intended for the use of people with drives that are able to be partitioned such as an ICT HD, CMD HD, Lt Kernal HD, and 1581s.

The last question that is asked to set up the stats file is the maximum number of users to have on the system. The recommended amount is between 100 to 200 users. Remember to limit the amount of users if the system is limited in storage capacity on the disk drives; each user takes approximately one block of space on the disk.

Finally, when the userlog is created, the sysop's account is created as User ID Number one (#1) after asking questions pertinent to that account.

The only thing left now is to create the files necessary to run the system. Further information on these are described in the C_BASE FILENAMES section of the documentation. There are sample seq system files on the back of the C_BASE owners disk.

The first basic step in setting up the BBS is now complete! Step two consists of creating the configure file in the boot program-- explained in the boot section of the documentation. Step three consists of defining the access groups in the boot file. Once this is done, the system is ready to go!

If the stats file needs editing to add or delete subs,uds,libraries or anything else, load the system create and use the stats editor portion of it.

BOOT FILE/SYSTEM CONFIGURATION

The first thing the BOOT file asks for is the TIME. This should be entered as HH:MM AM or PM where HH=Hours and MM=Minutes. If the boot does not detect any typing at this prompt for more than a minute, it will automatically load the stats file to see when the last user called and then use that as the current time to boot the BBS up. This is useful for sysops that run a XETEC LT KERNAL Hard Drive, because they can name the bootfile as AUTOBOOT so that when the system is first turned on, the BBS automatically loads both itself and the time in.

If a file on the boot disk called "cmdclock" exists then the BBS will skip the time prompt and get the time from the CMD Harddrive clock and load the BBS. For instructions on this, read the section of the documentation about the CMD harddrive.

A second menu will appear on the bottom of the screen that allows the use of built in fastloaders, changing the modem settings, miscellaneous settings, access group settings, and text prompt information. The last option allows the saving of the configure file once the system has been configured in memory.

If the amount of time to install the text prompts is a concern, the sysop may reduce this time by going to edit the text prompts and then pressing [RETURN] at every prompt until it finally asks to save the prompts as a file. The file will be saved as a binary program file called "text" and will not take as long to load. See the section of the documentation on text prompts for more information.

For the purpose of setting up the configure file information in memory, choose the option for setting up the modem and after that use the option for setting up the miscellaneous information.

When the system is already configured, all that has to be done is to choose the continue option. Before loading the BBS, the boot will ask whether or not user lock out mode should be on. This user lock out mode makes sure that no one from the sysop's side can affect the BBS by typing something on the keyboard. This can be turned on during occasions such as parties where the sysop may not want people to mess up the board by randomly pressing keys on the keyboard if someone gets majorly wasted. The next prompt asks if the fastloader will be used to load the main BBS program; this only refers to the built-in fastloaders and is in no way asking about external cartridges or other fastloaders that may be installed on the computer itself.

In the modem setup, modem type will be asked first. Depending on what modem type chosen, the BBS will automatically choose the appropriate AT Init string. However, this will pop up as a question anyway just in case it is necessary to add or delete things out of the already configured AT init string. Most people just need to press [RETURN] at this question since the AT Init string is usually appropriate for the modem.

It is imperative that the modem be turned on while configuring this section of the BBS because this is where the BBS determines the carrier status of the modem -- normal or inverted.

If the BBS does not work with this modem configuration and the modem has dip switches, the problem may lie in how the dip switches are set. The two most important dip switches to look at are the ones that control carrier detection and DTR if the modem has dip switches. Carrier detect should be configured so that it is NOT always on or always off but is on or off depending on whether someone is really connected or not. DTR should be switched so that it is modem controlled rather than always on or always off as well.

At the end of some of the AT Init strings there will be an "s10=30". This is done for a fairly unique feature called ANTICALL WAITING. What this does is allow the modem to IGNORE the CALL WAITING BEEP that might otherwise kick someone off of the BBS. This will work if a person has 300 baud or has a hayes compatible modem (or 1670 v2) and has previously initialized their modem with "ats10=30" to allow their own modem to ignore the call waiting tones as well.

Another item to keep in mind is that modems work best with CBASE if they are defaulted to give VERBOSE rather than NUMERIC RESPONSES. Some modems for example say **0** instead of **ok**. If your modem says **ok** instead of **0** after typing AT Commands then that will work. If it says **0** then the modem should be set to verbose responses. The modem's user manual should have a section on how to accomplish this.

If the modem is abnormally loud or turns itself on when a user logs off, go into terminal mode and type **at&m0** plus [RETURN] followed by an **at&w** plus [RETURN]. This should quiet the modem down and default it to speaker off.

Part of the modem setup asks a question about the delay with which the BBS picks up the phone. The range is 1-255 with 5 being the default delay. For some people with Caller ID, the modem needs to wait a longer time before the caller ID box will generate a phone number. There have been owners of CBASE who have had to wait as many as three rings before any information shows up on a caller ID box and other owners who do not have to wait for the information at all.

Note To BASIC Programmers: If it is necessary to experiment with the

delay timing for any reason, it may be easier to manipulate the variable directly after the board is loaded rather than constantly reconfiguring in the boot file and then reloading the BBS. The delay of 1-255 is stored in memory location 999 which can be changed via a POKE command.

The miscellaneous section of the boot file contains many questions which are somewhat unrelated to each other but are necessary when ultimately setting up the entire system.

In this section, if a mistake is made while typing the answer to a particular question, **[CURSOR UP]** will go back through the previous questions!

One of the first questions that is asked by the boot file in this section is what the Lt Kernal Device number is. Pressing [RETURN] here will indicate to the BBS that this type of harddrive is not connected to the system.

The question dealing with whether the callers log should be saved to printer or disk is concerned with where the commands that a user types while on line will be kept. The preferred option is "disk" since even if the output is set to go to the printer, the callers log information will still save on disk as well.

The system will ask whether the feedback should be stored as mail which is the more desirable option for most sysops or redirected to the printer. The reason to have feedback left as mail is that it is easier to validate users with the auto-edit user function in the mail section for users with remote/sysop command bit access.

The system will ask if the sysop is defaulted to be IN or OUT for the purpose of chatting. This defaulted option may be toggled at the WAITING FOR CALL screen upon pressing the appropriate Function Key described in the appropriate section of the documentation.

The choice of whether to validate the system disk everytime a user logs off is also asked. This question should never be answered affirmatively unless a 1581 exists on line and is experiencing severe problems that are described in the 1581 section of this manual.

The option is also open to have guests mandatorily taken to feedback after they apply to the BBS.

Call-Back validation is a feature that allows the BBS to call back the person who has applied and validate them to a different access level than access level 1 if a carrier is detected. This feature in the BBS is not recommended for use at all and was only installed because a select few people wanted it even though they do not mind the security risks it can pose to their BBS.

When call-back validation is activated, there is the option of defining which level the user is to be validated to after the call was successfully completed. Read how to setup the "callback" file in the description of filenames section of this documentation.

Additionally, there is also the option of having a closed system or a system open to guests.

The option of having Disk status messages displayed to the user is also configured here. The default is to have them on. These disk status

messages are not displayed to the user online; only the sysop sees them. Basically whenever a file is opened and a dos command is given to the drive, the dos command will be printed to the screen and the filename preceded by the drive number will be printed to the screen as well. This is useful for troubleshooting the program if the sysop needs to know what file is being opened on what drive. It is **strongly** suggested that this mode be left **on**.

Macros are one line messages that are printed before the main prompt in a sequential order and usually users prefer sysops to configure them "on" for the purpose of advertising other BBSes and local events.

If welcome bulletions are configured to be non-abortable, then users logging onto the BBS will not be able to abort the bulletins as they are prompted to read them. This is the advised way to set up the system since bulletins are normally important for users to read, and the bulletins autoabort anyway if the user has already read them in a previous call.

The system also allows the sysop to configure the number of open screens. If it is one (1) then the filename for the graphics open screen is "open". The number here is only for choosing the number of COMMODORE GRAPHICS open screens. The system will still look for only one ASCII/ANSI open screen as described in the filename section of the documentation. Likewise, the number of "end" message screens are also configured here. If the number of end and open screens needs to be configured more exotically, it is better to modify the BBS basic to accomplish this.

THE FOLLOWING SECTION ON SPACE ALLOCATION IS IMPORTANT! DO NOT SKIP!

The section in the configuration on DIMensioning is a more complex section. Basically for each section of the BBS, the BBS has to know in advance how large the section will be so that it can appropriate the proper amount of space.

Before, BBSes would simply dimension each section out to a number like 30 so the sysop would not have to worry about it and still be able to have 30 subs and 30 uds on his BBS.

However this is a waste of allocated space to a sysop who has less than 30 subs and 30 uds. C_BASE v3.0 is one of the first BBSes to allow the sysop to define how much DIMENSION space is allocated for each section of the BBS.

It is recommended that the system allocate about 2 dimensions larger than the particular section really is. This is because it makes it easier to edit the BBS later on to include more subs or other sections without having to also reconfigure this part of the boot file.

If a DIMENSION or BAD SUBSCRIPT ERROR occurs in the BBS while it runs then most likely one of these numbers is inappropriately matched to how large that section of the program really is.

For example, when it asks the number of libraries to dimension, if the system is configured for four libraries in the system create the system should have a number from around 4 to 6 entered in here.

The question referring to the number of modules and application questions should default to about 30 if the system will not have more than 30 online games in the mod file and 15 application questions in the app file. If the system has more than 15 application questions then the number here will need to be increased. For example, if the system had 18 application

questions then the number given here should be 18 times 2 (36).

The reason for the number of application questions needing to be at most half of the number stored here is that while the application questions are stored in the first half of the space allocated for both application questions and module files, the actual answers to the application questions are stored in the other half of the allocated space. Thus, for each application question there needs to be two allocated spaces -- one for the question and one for the answer.

C_BASE allows the system to be configured so that people can not go to certain sections of the BBS unless they have posted a certain number of times during that call first. There are two seperate questions which address this issue concerning the sections of the board dealing with UDS and Modules.

Another question the boot asks is how many credits each post is worth. This will be taken into account in the UD section to determine how much the user can download.

The maximum number of downloads per call puts a global cap on how many files a user can download in one particular call. Some sysops only want their users to be able to download a certain amount of files per call like 13 so that they do not go using up valuable board time for their own file leeching purposes.

The BBS logical unit to store the UD- x files is the logical drive number that the directory and description entries are stored for each UD. The UD- X files themselves are explained in the filenames section of the documentation. The concept of the BBS logical units is explained in the Drive Access appendix.

The BBS logical unit to store the UD-x file defaults to 0: logical unit. If the system is configured for 0: logical unit then the UD- x file will be read off of the disk drive that stores all the other system files. If the BBS is configured for 1: logical unit then the UD- x file will be read off of the disk drive that each particular UD is stored on. For example in this case if UD 1 is on device 9 and UD 2 is on device 10 then UD- 1 will be read on device 9 and UD- 2 will be read on device 10.

The system will also need to be configured for the device number, drive number, and dos init command for the system disk. Basically, the dos init command is the dos command that is sent out to the disk when a system file is activated. The Drive Access section of the documentation explains how this works in more detail. The device number, drive number, and dos init command that is entered here will be the parameters for the 0: BBS logical drive number.

The option to have UD title screens come up when people go to the various upload/download directories is asked for those people that wish to have a graphics file or a detailed descriptions of what each UD section is about. The name of these title screens is explained in the filenames section of this documentation.

The Boot file also allows the configuration of whether or not the mail files are to be seperated by seq file or by markers within one sequential file. The default is to have markers between mail. Basically C_BASE is defaulted to having all a person's mail stored in one file. When a user sends a person mail on C_BASE, and the person already has mail then instead of making a second file, the BBS simply appends the mail to the original file. This saves considerable file and directory space on the

disk! The only disadvantage is that it is slow if the user has more than 25 letters accumulated.

When the option for seperate mail is chosen and a second mail is sent to a user that already has mail, a second mail file will be created. The C_BASE filenames section has a description of what the mail files are named on the disk for whichever option is chosen.

It is only advised to set the option for having seperate sequential files per mail if the mail is stored on a harddrive like the CMD or Lt kernal. Otherwise, for nearly all BBSes, the default of using one mail file per user instead of many mail files per user is the preferred choice.

The next questions correspond to modules that may be added to the system in the future as FUTURE EXPANSION TO C_BASE v3.0.

Message Networking Module Is For FUTURE EXPANSION.
Sub Board Module Is For FUTURE EXPANSION.
Upload Download Module Is For FUTURE EXPANSION
Sysop Maintenance Module Is For FUTURE EXPANSION

When the questions corresponding to these modules come up that do not yet exist on the system, simply press [RETURN] to use the default settings.

With respect to the future message networking module, the system also asks for information as to when the system message networking module should call another BBS. This information consists of the starting hour and the number of hours past that starting hour that the message networking module will call out at.

When configuring access groups, the boot will ask whether the system should allow a particular access group to have primetime access or not. Primetime BBS hours are defined by the system in this miscellaneous section by choosing the hour in standard military time (0-23 for 12am to 11pm respectively) that PrimeTime for the board starts at. Then the system will ask for the number of hours that PrimeTime lasts. When a user logs on and does not have access to PrimeTime hours, that user will be logged off with a file called "prime" read out to the user.

The BOOT file will also ask what hour to have the c/network module load in at. This information is entered the same way the starting hour for primetime access is entered by typing the hour in standard military time (0-23). The next question will be how long the window of opportunity will remain open for the c/network module to load in. The default value is 2 for a 2 hour window. In otherwords, if the c/network module is due to load in at 3am, but someone is on the BBS an extraordinarily long period of time like 2:30am to 4:00 am then the window of time should be open so that the board can call the appropriate nodes in the network. Having a window of 2 hours means that it will still call up til the hour of 5am (if the start hour was 3am) if the BBS has not called out yet that evening.

The BOOT program then queries what Device # the Main BBS will be on for loading purposes. This is so the BBS knows what drive to load the main bbs off of when loading back in from a module.

It also asks whether or not to activate the currently loaded fastloader when the Main BBS loads back in. Usually if the BBS is loading off of a very fast drive or a harddrive, the fastloader should not be activated. The same questions are asked of the application module called "c/app mod" on the disk. Remember that the file loaded when a person applies as a guest is always **"C/APP MOD"**. If this file is not on the appropriate

device when a person applies, a "**FILE NOT FOUND**" error will occur.

The Boot file will then ask which device number the UD protocols should load off of when a person goes to TOGGLE Protocols in the BBS itself.

The question about saving sysops actions to callers log is usually answered with yes because the sysop usually has nothing to hide about the actions that he does on his own BBS.

The next few paragraphs have to do with configuring the colors of the BBS.

The Multi Color Basing System (MBS) is the option where the colors change randomly when a user is on the BBS from shades of Grey to Blue to Green to Red to Purple and so on.

If this function is not chosen, the colors of the system may be chosen seperately. An example of this is provided in the BOOT file itself.

The system will ask to choose the color that the cursor should have at prompts. Usually a neutral color such as yellow, light grey or cyan is chosen. Colors are entered by typing CTRL or COMMODORE key plus the number corresponding to the color desired. For example, a person who wanted a white cursor would press [CTRL] and 2 at the same time.

Upon completing these questions in the modem and miscellaneous setup, the basic system is now done! The final step is to save this configuration. If there is an existing configuration file, choosing the option to save the configure file at the main boot menu will erase the previous configuration. It is usually a good idea to save the configuration so that all this work does not have to be done over again the next time the BBS is booted up.

Upon continuing past this menu, a list of some of the major board information will pop up for a double check. The bottom of the screen will ask if this information is correct. If it is, then the Boot will ask whether the board should be loaded with a fastloader. This question is referring to the built in fastloader; it does not refer to any fastloaders that may be hooked up to the system such as jiffydos or cartridges! The third question the boot will ask is whether Keyboard lockout mode is on or not. Next, the BBS will start loading and C_BASE v3.0 is on its way to running. CONGRATULATIONS ON SETTING UP THE SYSTEM!

SYSTEM FILE DESCRIPTIONS

All the following files will be necessary on the system disk(s) in order to have a fully running system of C_BASE v3.0. At the end of this section, there is a list of the filenames that definately belong on the system disk if the system has those sections of the BBS open. The other files are configured to reside on other places in the program.

ANSI TAB - This is a program file that stores the ANSI Graphics character equivalents of Commodore graphics characters for the purpose of translating Commodore graphics to ANSI users. This file is only necessary if the system has ANSI users.

ASCII TAB - This is a program file that stores the ASCII equivalents of graphics characters for the purpose of translating Commodore graphics to ASCII users. Basically this is the same file as ANSI TAB except that it translates less of the Commodore graphics since there is no equivalent for some of the graphics characters in ASCII. This file is only necessary if the system has ASCII users.

PUNTER - This is one of two program files necessary to run the Upload Download Section of the BBS. It is the Punter Protocol file that loads into the BBS when someone toggles protocols at the Upload Download Section Prompt and X-MODEM is currently in memory instead of the Punter Protocol. This file is only necessary if the system has the upload/download section open.

XMODEM - This file is necessary to run the Upload Download Section of the BBS as it stores the XModem Protocol ML and loads when someone switches from PUNTER to XModem at the Upload Download Section Prompt. This file is only necessary if the system has the upload/download section open.

STATS - This file stores the basic stats of the BBS. The UD Names, Sub Names, Number of Users, Mail, Date, Name of Sysop, Security Level Accesses, Libraries, etc. More specific information about this file is found in another seperate section of this documentation. The reason there is a seperate section devoted to this file is because the stats file holds a great deal of information and the entire system relies on its integrity.

USERLOG - This is a relative file which stores each user's account in a corresponding record. This includes name, alias, location, phone number, credits, posts, and more. It is just as important as the stats file and should be backed up periodically using Copy-All or another file copier that copies the relative file type.

WAIT - This file is listed when the C_BASE system is waiting for a Call.

MTERM - This is the menu that is listed when the sysop goes to the built In Mini Terminal. This file is not necessary if the system does not have the built in mini terminal program.

SYSOP - This is the menu that is listed when someone press ? in the Sysop Maintenance section of the C_BASE program.

MENU - Main command menu.

NEW - This is the menu that users who have just applied. Users with access group 1 see this menu.

SUBS - This is the menu users see when they go to the subs section and press ?.

SUBOP - This is the menu that users see when they press ? in the subs section and they have either remote or subop (subboard operator) access.

UL/DL - This is the menu that users see when they go to the Upload Download section of the board and press ?. This menu is only necessary if UDs are open on the BBS.

UDOP - This is the menu that users see when they press ? in the ud section and they have either remote or udop (Upload/Download Directory Operator) access. This menu is only needed if the UDs are open on the BBS.

LOGON - This is the file that is read when a user first logs on. It lists stuff like their last call date, who the last caller was, how many posts they have, how many calls they have left that day, etc. An example of this file is provided on the C_BASE v3.0 owner disks.

USEREDIT - This file is read whenever someone with remote access edit a user and contains information about the user.

USER - This file is read whenever a "Y" for Status Check is typed at the main prompt.

END - This is the file users see when they log off. If the sysop has configured the system in the boot file for multiple ending screens then the filenames are actually "end x" where x is a number from 1 to however many end files he has chosen to have. For example the first end file in this scenario would be called "end 1".

BUL x - Where x is the number of the bulletin. When the BBS system is started, a "bul 1" file should be made for the first bulletin welcome message telling people about the initial news of the system.

If a remote access person wants to add anything to that bulletin, they may create a second file called "bul 2", and if they want to add to that, they can create "bul 3", and so on. However, for the most part the system will only need a "bul 1" file such that if the info on that file is outdated then it can simply be written over the previous "bul 1". These files may be created using the W command in the sysop maintenance section of the BBS. More information on the W command is found in the remote sysop maintenance section of the documentation.

A common error beginner CBASE sysops have is that they do not write the bulletin file using the W command in the remote sysop command prompt. This will invariably cause an ILLEGAL QUANTITY OR FILE DATA ERROR every time someone logs onto the BBS.

If the system requires that the bulletin file should only be seen by specific access groups then the W command may be suffixed with the Access Code which codes for letting only those access groups see that bulletin file. For information about calculating the access group code, read the access group section of the documentation. For example, if only access groups 2 and 3 are to see the bulletin that is being written with the W command, instead of just typing "w" to write the message, "w12" would be typed "w12" where 12 is 4+8 according to the correct way to set up access groups. It is not necessary to understand the concepts this paragraph to run the system. If it is understood however then it will allow usage of one more of the many functions in CBASE v3.

UD- x - This file does not need to be created. The BBS will do it. A sample name might be "ud- 1". Where X is the Number of the UD. For each directory entry, it stores several variables in order. The first is the caller number of the person who uploaded the file for comparing a new scan check. The second is the date the file was uploaded, the third is the program name, then the program type, then the amount of blocks the program is, and finally the description of the program. These files are not necessary unless the UD section is open.

UD-TITLE x - This is the title screen for the UDs. When a user goes to a UD directory then this file will be read to the user as a graphics picture of a description of that UD number x.

FORCE x - This is a forced mail file. Forced mail allows the sysop to send someone a message that they have no choice but to read. The BBS checks if the user have forced mail after it reads the bulletin files in. A remote access person writes a forced mail file by using the "w" command in the remote mode. This is not a file necessary to run the system. When W is typed, followed by a numerical code to write the forced mail file then this code will be stored in the forced mail, and it will tell the BBS what to do with the user after he or she has read the forced mail.

There are three basic codes.

1 = Kick The User Off After Reading
2 = Erase the forced mail file after the user has read it
4 = Delete the user after reading the forced mail file

Combinations of these codes may be used to produce different results. For example, to kick the user off after reading it and also erase the forced mail file afterwards, then one would use "w3" where 3 = 1 + 2 for the command to type at the remote sysop command prompt. This is similar to the way one adds a suffix when making "bul" files that are only accessed by certain access groups.

PRIME - If a user does not have access to primetime BBS hours and tries to log on during these hours then he will be logged off of the BBS and this file will be read out to him informing him of his problem. This is only a necessary file if the primetime access function has been activated in the BBS.

NAMELOG - This file does not need to be created. The BBS does it. It stores the names and ID's of all the users currently in the system and the BBS searches this when it searches for a handle in edit mode, sending mail, or logging on. It is usually a good idea to regenerate this file from scratch by typing ! at the Sysop Maintenance Section of the program once a week or so to keep it accurate since deleting a user will not remove his name from this list.

T S x - Where X is the sub number that there is a title page for. When someone presses "T" in the subs or they go to a sub they have never read msgs on before, this file will load up explaining what the sub is all about and what is appropriate subject matter to post on that sub if the intent of the sub is not already made clear by the title. These files are not necessary at all unless the sysop wants to give callers a more definitive idea of what a sub is for.

[SPACE]x - This file is the file that contains the mail for user number x, if and only if that user currently has mail waiting for him. If the option for seperating mail by file is turned on in the Boot file miscellaneous configuration then the filename is [SPACE]x[SPACE]n where n is the number of the mail file. So if a user number 10 has 2 mails waiting then filenames " 10 1" and " 10 1" would exist on the directory that contains the mail files.

CALLERS - This file does not need to be created. It stores a log of what each caller has done in the program while they were on the system.

CAT - This file stores validation information from the APP file. It should be cleared once a week or so by erasing it from the main system disk. It will automatically be created when a user applies and the BBS recognizes a cat file does not already exist on the disk for CATaloging Applications. The sysop can also read this file from the waiting for call screen and delete it when done reading it.

H - This is the Help File.

I - This is the Info File which contains information on what hardware the system is currently run on.

CHAT - This file loads up for the user when the sysop is "out" and the user tries to type C for Chat at the main prompt. When chat is pressed, the BBS will ask for a Chat Subject. This chat subject is stored in the

callers log after the user logs off so that if the user never left feedback the sysop still knows what the person wanted to talk about.

MF - This is the multi Feedback file. It should be created in the following format.

```
y
y1
y2
y3

[WHAT THE USER SEES]
```

Where Y is the number of Sysops/Co Sysops/Remotes to send mail to. y1,y2,y3 etc depending on how many the number was for Y are the user ID's in the order of the mail to be sent. [WHAT THE USER SEES] is a listing of sysops/remotes/etc according to what the sysop wants the user to see, as the y1,y2,y3 stuff will be HIDDEN from the user, the REST of the file is [DESCRIPTIVE] as to what the [USER ACTUALLY SEES]. If this concept is hard to understand, then experimenting with it may provide some illumination.

To be more specific, when writing the 'MF' file, the first line will be the number of people the system has in the multiple feedback section. The next line will be the ID of the first person in Multiple Feedback. Most Likely it will be a "1", because the sysop is usually the first person who wishes to be left feedback and the sysop's ID Number is "1". The next lines are the ID's of the rest of the people who are in the Multiple Feedback. When a user on the system types F at the main prompt for Feedback, the file is read to see first how many people are in the feedback and then the ID's are read into memory also. The rest of the file prints out to screen and modem normally. This last section contains any graphics and a little msg saying something like [1] For (First ID in the Multi Feedback file), [2] For (Second ID in the file), [3] For (3rd ID read in) and so on.

An example of this would be the following MF File.

```
2 (Equals Number of Sysops/Remotes)
1 (ID of first person to send feedback to...Usually 1 as that is the User
ID of the sysop)
2 (ID of second person to send feedback to)

[Descriptive Part]

Multi-Feedback System

[1] Prof. Plum   Sysop
[2] The Bard     Remote
```

BBS LIST - This follows the same format as "MF" in that there is information read in from the file before the actual printing of the file occurs on the screen. However, in this file, the first line is the number of Question prompts in the BBS lister. The following lines after the number of question prompts contain the questions themselves. For example, if the person creating the file only wanted the BBS lister to ask for "NAME of BBS:" and "PHONE NUMBER:" the first line would be a "2" for two question prompts, while the NEXT line would be "NAME OF BBS:" and the line after that would be "PHONE NUMBER:".

The following is an example for the above description of the BBS LIST file.

```
2 (Number of Questions)
NAME OF BBS : (First Question)
PHONE NUMBER: (Second Question)

[THE REST OF IT WOULD BE THE INFORMATION THAT IS PRINTED TO MODEM AND
SCREEN JUST LIKE A NORMAL FILE]
```

OPEN - This is the screen that is shown as soon as a user log ons to the system and before he types his/her LOGON ID. If the sysop has configured multiple opening screens in the boot file then the filenames will be "open x" instead of "open" where x is the number from 1 to the number of open files configured by the system.

MSGMENU - This file is the file that loads up and prints to the user when he/she is trying to write a message.

MSGHELP - This file loads when .H for Help File is activated by a user in the message maker.

MAIL - This is a menu of mail options that comes up when the user goes to the mail section of the program.

LIB x - When a user chooses which library section to go to, the BBS reads in the corresponding LIB x file for that section where X is the NUMBER of the LIBRARY SECTION. The Lib file for section one would be called "lib 1". The Lib file for section two would be called "lib 2" and so on

The Lib X file is read in just like the MF and BBS LIST files. It has information stored at the top of the file that the user never sees, and then it has a descriptive part that is printed to the screen after the first information part is read in.

The first line of the lib x file is the number of SEQ files that will be in that library section.

The rest of the lines contain the filenames of the SEQ files as they exist on the system disk itself. In the next example, the "lib X" file will have three SEQ files, the first of which is called 'billthecat', the second -- 'garfield', and the third -- 'garfield2' on the BBS Logical Drive for libraries.

```
3 (Equals NUMBER of SEQ Files)
billthecat (Filename of First File)
garfield   (Filename of 2nd   File)
garfield2  (Filename of 3rd   File)

(DESCRIPTIVE PART THAT IS PRINTED TO SCREEN STARTS HERE)

[1] Bill The Cat Picture
[2] Garfield Picture Number One
[3] Garfield Picture Number TwoÜfÜ
Please Choose A Number Now.
```

NOTE: To further clarify this, when the file is read, and the user chooses "1" for the File Number to Read, the file "billthecat" will be read to the user like a regular sequential file.

MOD - This is the module file that contains the information on the modules for loading. If the module access option is on for the access groups configured in the boot file, then the system will need to create a MOD file that is read everytime the user types "MOD" as a command at the

main command prompt. This file follows a format similar to lib x,mf, and bbs list files.

The FIRST line in the file is the NUMBER of MODULES to load.
The subsequent lines contain information on the modules themselves. These are called the info lines on each module.

The info line of the individual modules follows a strict format.

ACCESS GROUP CODE;DRIVE TO LOAD FROM;DRIVE TO USE FOR FILES;FILENAME

The information within the info line must be seperated with ";" (semi-colons).

The access group code is determined in the same way access group codes are determined elsewhere in the program. For an indepth discussion of how to calculate access group codes see the Access Group section of the documentation.

THE DRIVE TO USE FOR FILES, is the device # of the Drive from which seq files and other files will be used by the module. This is useful for a person who has a RAM EXPANSION Unit. They may wish to place modules on the RAM EXPANDER for loading, but not wish to have permanent seq and rel files associated with that module stored on it, so the sysop can choose to have another disk drive to store those files. For example, it would be a shame if a module such as EMPIRE were subject to a power loss and the rel file containing all the EMPIRE accounts on the ram expander would be lost.

The built in FastLoader may be used when Loading a Module. In order to do this a '-' must be placed in front of the filename as it appears in the MOD file. Note however that - is only an identifier to activate the fastload. The BBS will try to load the program as the filename without the '-' in front of it. It is never recommended to use the built in fastloader to load the modules, but if it works and does not cause the system any problems then that is great.

An Example File Setup For MOD:

```
2 (Equals Number of Modules)
32766;9;9;-c/wall   (1st Module)
32766;8;9;-c/empire (2nd Module)
```

(DESCRIPTIVE PART STARTS HERE)

MACROS - This file is set up a lot like the MF and Lib x files. Basically the first number on the line is the number of macros that the system should have. The subsequent lines are the macros themselves. Macros must never exceed 88 characters or the BBS will encounter an illegal quantity error when reading this file in.

MLOG - Whenever someone adds a macro to the macros list, their name followed by the macro they entered is stored on this file so the sysop will know who did what. This can be read conveniently from the wait screen and after reading the BBS will prompt as to whether to delete the file to start over again or not.

APP - This is the application file. It contains the descriptive portion of what the sysop expects of the user before he/she applies to the BBS system.

Before the descriptive file is read to the user, the questions and number

of application questions must be read from the beginning of the file. It follows the format of the number of question prompts in the first line, and then the subsequent lines contain the application questions themselves.

For the first seven question prompt lines there is no choice in the order of the questions that will be asked since these will be stored permanently in the userlog when a person applies. They must be stored as:

```
7 (Equals the Number of Questions)
Alias      :
Location   :
Phone #    :
Real Name  :
Password   :
CompType   :
Birthday   :
```

If the number of question prompts is greater than six then additional questions must be added to the APP file on the lines following the first seven questions. The answers to these extra questions as well as the answers to the initial seven questions will be stored in the CAT file. Only the answers to the initial seven questions will be stored in the userlog record of that user.

In the boot file one of the queries is how many questions to dimension for APP and for MOD use. There may be as many questions as was defined in this section divided by 2. So if it was dimensioned to 30 possible module files then the system will be able to have 15 application questions.

It is advised not to give people a lot of questions to respond to in their application. People generally start to become irate when they have to answer more than 10 questions and hang up.

CALLBACK - This file is only necessary if the system has activated the call back validation feature in the BBS. This file is made using the W command in the remote maintenance mode just like the app, mf, and other files like it. The first line of this file is the access group to validate the person to if the phone number proves to be valid. The second line of the file is for BBSes who have local dialing access to more than one area code. This should be the area code that the BBS resides in so that the BBS makes sure not to add the area code when dialing the person's phone number. This area code should be followed on the same line with the other area codes that are local to the BBS if any are available. The remaining lines contain all the valid exchanges in the area including areacode if necessary for the local call. An exchange is a 3 digit number in the beginning of the 7 digit phone number that specifies a certain city. Remember, the first area code listed in line two must be the area code the BBS itself lies inside! If it is not understood what I mean by exchange or area code, then read the local phone book for an explanation. A sample file for an area with only one local area code would be:

```
2 (Access Group to be validated)
301 (area code for the local dialing area)
365
467
761
etc... (All the 3 digit exchanges that are local)
```

A sample file for an area with three local area codes and 301 being the one the BBS resides in:

2 (Access group to be validated)
301703202 (Maryland, northern Virginia, Washington DC)
365
467
202365
202467
703751
etc.. (All the three digit exchanges that are local and includes area code for local exchanges that are not within the area code the BBS resides in)

If it was chosen in the BOOT to have seperate ASCII MENUS then the following files will have ASCII equivalents

OPEN
NEW
MENU
UL/DL
SUBS
MAIL

If the ASCII menus option is turned on as configured in the miscellaneous section of the BOOT file, then the BBS will search for the filenames above with an "/a" added onto the end of it. So an ASCII user who logs on would read the file "open/a" instead of "open". If there are multiple open screens then there will still only be one ascii screen called "open/a". The same is true for the other filenames.

To summarize, the following files are the ones that definately belong on the system disk if those sections are in use: ansi tab, ascii tab, stats, userlog, wait, mterm, sysop, menu, new, subs, subop, ul/dl, udop, logon, user, end, bul x, force x, prime, namelog, callers, cat, h, i, macros, mlog, chat, mf, bbs list, open, msgmenu, msghelp, mail, callback, and app. Any other files described in this section are sysop definable as to where they go on the BBS.

MAIN SYSTEM COMMANDS

MOD This command accesses the modules. When this command is typed, the file "mod" is read off of the system drive and a list of modules that are available are printed up on the system. If this command is typed within stacked commands, the menu will not show up in order to save time. Thus, a user may type "mod;1" to get to module number one faster than he normally would.

S Accesses the subboard section.

N Accesses the subboard section and automatically starts reading the new messages.

UD Accesses the upload download system

Q Quick Log off

G Regular Log Off Command

REM Accesses sysop maintenance mode if the user has RM% higher than one. See the section on remote access to find out more about the RM% settings.

Y Lists user stats from a file called "user" on the main system drive.

E Toggles Graphics/ASCII mode

C Chat Request Command

F Multiple Feedback

I Info file on what the system runs on

H Help file for the system

LOG Reads a log of what the previous callers have done on the system from the file on the system drive called "callers"

BBS Accesses the BBS lister command

LIB Accesses the libraries

M Accesses the mail section of the BBS. This includes reading saved mail, sending mail, user scanning, alias changing, location changing, password changing, and carbon copying.

ME Accesses the Macro Editor

OMN Accesses the Omni Message maker

O Reads all the welcome bulletins as well as the Omni Message

? Views the main prompt menu

B Alter baud rate. This is only for 300 Baud Callers.

P Toggle automatic pausing function. When toggled on the messages automatically pause every 25th line. This variable for each user is defaulted to off.

V Toggles column with which to read messages in. Every user has this variable defaulted to read and post messages in 40 columns. When toggled for the first time, it will throw the user into 80 column mode of reading and posting messages.

Probably one of the most unique features about C_BASE is the ability to output the messages and bulletins in either 40 or 80 columns. This feature is called true variable column outputting. On C_BASE the word wrap feature does not really wrap words in the message maker except superficially. The word wrapping is done upon reading a message instantaneously in machine code so that the message looks as if it was word wrapped for the user who is reading the message, not just the user who typed it!

The access groups that are allowed to use most of these commands can be changed in the boot file. See the section on altering access groups for more information.

STACKED COMMANDS

The BBS supports stacked commands. A string of commands may be entered at most prompts and seperated by the ';' character. The prompts that the stacked commands support are the logon prompts, the main prompt, remote prompt, subs, and UD prompt.

For Example:

"s;q;ud;q" at the Main Prompt would take the user to subs, quit to main,

go to UD's and then quit to main again. Stacked commands also work in the sysop command maintenance prompt foradded convenience to the sysop. Especially when deleting many users, the sysop may type 'd31;d34;d35..etc..etc' and then walk away in confidence that when he comes back several users will have been taken care of rather than just one.

WAITING FOR CALL SCREEN FUNCTIONS

[F1] -- Activates MiniTerminal Mode

[F2] -- Read Callers File

[F3] -- Guests Allowed Toggle

[F4] -- Simulated Logon

[F5] -- Goes To Remote Maintenance mode.

[F6] -- Sysop Available Toggle

[F7] -- Modular Terminal Program

[F8] -- Connect User to the BBS/Outputs ATA to modem

[C] -- Read CAT File log of applications

[V] -- View log of who entered what macro

[M] -- Read And Reply Mail As User #1

[CR] -- Logs on Sysop by pressing carriage [RETURN] key

FUNCTION KEYS

[F1] -- Increases security by 1, time limit by 10, blocks uploaded or blocks downloaded by 100 depending on which field in the window is shown in reverse light blue.

[F2] -- Does the same as [F1] except it decreases the values instead of increasing them.

[F3] -- Takes the user to the main system prompt.

[F4] -- Takes the user to the remote sysop command maintenance prompt.

[F5] -- Allows the sysop to highlight and choose which part of the window to change with [F1]/[F2]. Pressing it will cycle through highlighting the security field, the time limit field, blocks uploaded, or the blocks downloaded fields.

[F6] -- Places the current user on hold. During this time period what the user types will have no effect on the BBS and the user will be unable to see anything that is being done on the BBS by the sysop.

While the user is in hold mode, the sysop may freely give the user remote access by going into remote mode via [F4] and doing an edit of the users account. This should be noted as an important exception to the fact that normally only a user with remote access equal to 255 can alter the remote access bit during the edit of a user's account. This can be utilized as a method of giving someone remote command access while on-line rather than

having to edit the user off-line.

[F7] -- Toggles Chat Engaged/Disengaged

[F8] -- Kicks a user off the BBS with a flagged note in the callers log indicating that the user was F8'ed.

MAIL SYSTEM

After typing M from the main prompt the user will be taken to the mail system.

S Send a piece of E-Mail to a user number or handle.

C Does the same as S but this command stands for Carbon Copying. It allows the user to send the same mail that the user has sent previously to many people at once instead of having to retype it.

U User Scan lists the users aliases and corresponding ID numbers.

R Reads mail if the user has any

A Allows the user to change his alias graphics. It will not allow the user to change the actual lettering in the alias; it will only allow the user to change the graphics around the lettering. This is a security provision so that people can not log on and change their alias to the sysop and post an insulting message as "The Sysop" on the general base.

L Allows the user to change his location.

P Changes password.

MESSAGE BASE COMMANDS

GENERAL

First, the message base system on c*base is based on a RANGE of Messages. The messages the user has available to read at any one time on a sub is displayed as the range. For example, if the range were 20-30, the user could only read messages numbered from 20 to 30.

There are several basic functions which I will touch upon here only briefly as they are common to nearly all BBSs.

? This will print the subs menu for the user.

Pressing [RETURN] at the prompt will make the BBS read the next message.

Typing R will REREAD the current message that was lastread.

Typing Rx where X is a number will read message number X.

Typing Sx where X is a number will start scanning message headers at message number X.

Typing S alone will allow the user to SCROLL through the messages on one sub. That is, the messages will automatically read one after the other. Individual messages can be skipped by pressing "S" during the scroll, and the whole scroll can be aborted by typing

"A". If the user wishes to pause a scroll he can use one of 3 different keys ([SPACE],[ ], or [HOME]).

EXAMPLE: SCROLL FROM [30]:

Pressing [RETURN] at this prompt would automatically start scrolling the messages on that sub starting at message number 30 in the range of messages.

At the end of each sub, the user is asked whether he or she wishes to post, post anonymously, list subs, or change subs.

N automatically and conveniently scrolls through all the messages that the user has not read starting at the number of the sub that the user pressed "N" at.

Once again, at the end of each sub, the user is asked whether he or she wishes to post, list subs, or change subs. If the user presses return, the BBS will automatically read the new messages on the next sub. The user may type A instead of P on the sub to post anonymously at that prompt if the flag is turned on for that sub to allow anonymous posting.

If the user wishes the N scroll to automatically skip certain subs, the user may type J to toggle whether a particular SUB board is joined or unjoined. This command is described in more detail seperately in this section of the documentation.

L simply lists the subs in order with their corresponding number to the left of the name of the sub board. To the right of the subboard name, the amount of new messages to read on that sub are displayed as well.

X Typing number X switches the user to subboard number x.

> This takes the user to a subboard that is numbered one higher than the current subboard the user is on provided that the user has access to more subs.

T reads the title message of the sub to the user. The title is automatically read when a user goes to a sub he or she has never read messages on as well. The name of the file is "t x" for the title of sub number x and this file resides on the BBS logical drive where the posts are, not on the system logical drive.

Q Quits back to the main prompt.

P posts a message with a subject that the user types in. If anonymous posting is an open option on that sub then the BBS will ask whether the user wants to post anonymously.

O Allows the reposting of a message. If a user accidently abort his post while writing it or is cut off by call waiting and the user calls back right away, then typing O will allow the user to begin posting where he last left off in the message. This only works if no other message has been posted or written in the interim time.

* Posts a message as a reply to a post that was last read. The subject of this message become "(R)[Number of the message replied to]" of last message read.

M Allows the sending private mail to the user that wrote the last message that was read

J Toggles whether a sub is joined on a users account or not. All subs are initially joined on a users account. If a user unjoins a sub, then when they do a [N]EW SCROLL of messages, the board will skip the subs that are currently unjoined.

SUB-OP/REMOTE ACCESS COMMANDS

Ex Edits Users #x or with Handle = "x" just like at the remote maintenance command prompt.

K Kills the message that was last read.

V Allows the user to view the alias of who posted the last anonymous message read

? Views the subop menu called "subop"

% Allows the user to change the name of the sub that he are currently on

\ [British Pound Key] Lists all the access group names and their associated access group numbers

\w Lists all the access groups that have access to post on the subboard that the user is currently in.

\r Lists all the access groups that have access to read messages on the subboard that the user is currently in.

+rx Gives access group number x access to reading messages in the subboard that the user is currently in.

+wx Gives access group number x access to posting messages in the subboard that the user is currently in.

-rx This does the same thing as +rx but it takes away access to an access group that can already read messages on the sub the user is currently in.

-wx This does the same thing as +wx but it takes away access instead of giving access.

If the sysop wishes to edit a particular post, he should use ".L" to load the file into the C_BASE message editor and edit it as a normal seq file using the w command entered at the remote maintenance prompt. Since the only way the BBS knows how to seperate the messages from each other within a sequential file packet is the chr$(255) checkerboard pattern character, it is important that the user with remote access do not delete this feature out of the message packet while the user edits it so that the messages will remain separate from each other.

UPLOAD/DOWNLOAD SYSTEM

GENERAL

L Lists UD Directories available to the user.

X Typing a number X of the UD directory a user wishes to go to will take the user to that UD directory. For example if the user types "2" and the 2nd UD directory is called "Documentation" the BBS program will take the user to the sub directory called "Documentation".

> This takes the user to the UD directory that is one number higher than the one the user is currently on provided that the user has access to that UD directory.

$ This lists the directory of the files on the current directory being accessed which can either be a UD directory, or in eXchange mode it can be the directory of an entire disk.

$x This does the same thing as $ but starts reading out the directory at file number x. This is useful for reading particularly long directories.

N This does the same thing as $ but it only lists out files that are new since the user's last logon. This is useful for scanning the UDs for new files.

* This lists the directory of the files on the current UD directory including the description of each file.

Ax This does the same thing as * but only lists the description of file number x.

Q This Quits from the Upload Download section of the BBS to the Main Prompt.

? Reads out the menu "ul/dl". If the person is a subop or has remote access it will additionally read a menu called "udop" before reading "ul/dl".

Rx This Reads a SEQ file of number x on the current UD directory.

MD This activates the Multi Download Sequence. It will only work if the user has previously used the S command to select files.

MU This activates the Multi Upload Sequence.

CS This clears the previous selections made with the S command in case the user feels he made a mistake when he selected files earlier.

LS This lists the selections that have been made with the S command. The numbers with a # in front of it indicate the UD directory number in which the files were selected. The BBS automatically switches UD directories in the middle of a multidownload if files were selected off of multiple directories!

Sx Where X is a series of numbers seperated by ":" to select the files by number on the current directory for multidownloading.

S Simply typing S alone will list the files on the directory with a [Yes/No/Abort/Done] prompt so that the user can choose files to multidownload off of that particular directory. It is just like Sx, but is a different method of choosing files. In some cases Sx will be more convenient. In other cases, this method might be more convenient. C_BASE v3.0 supports both methods for added

convenience.

SN This will do the same thing as "S" but will automatically scan for new files on the directory and skip the old files since the last logon.

The user can select files on one directory, move to another directory, select files there, move to another directory and select files there. Then when he goes to multi download, the BBS will automatically switch directories while in the middle of multi downloading to accomodate the directory switching while selecting files.

U Uploads a file to the current UD directory

Dx Downloads a file number X off the current UD directory.

T This command toggles between the protocols Punter and XModem.

UD-OP/REMOTE ACCESS COMMANDS

^x This deletes a file number X off of the current UD directory but keeps the actual file on the disk. The filename is only erased off of the "UD- x" file but still exists on the directory. This is not for use in exchange mode.

/x This deletes a file number X off of the current UD directory and also scratches it off of the actual disk directory. This is not for use in exchange mode.

#x This does the same thing as the "/x" command previously described with one exception. This command additionally looks up who the original uploader of the file is and edits his blocks uploaded credits so that the credits that he was given for uploading this file are taken away.

Px This defines a pattern with which to view a directory in eXchange mode. For example, if the user was in eXchange mode and only wanted to view files starting with "c/" then the user would enter "pc/*". This is very useful when the user with remote access wants to regenerate a UD directory with files beginning with a certain prefix or for a quick look up of files if the user is downloading in eXchange mode.

{LEFT ARROW}x This is only for exchange mode use. It adds file number X onto the "UD- x" file for the current UD directory that the user is in.

! This regenerates a whole UD directory. While in eXchange mode, it will take the currently read directory and convert it to a "UD- x" file according to which UD number the user is currently in.

\ [British Pound Key] is used to list all the access groups and their corresponding access group numbers for easy reference.

\c [British Pound Key] followed by a "c" lists the access groups who have unlimited credits on that UD section.

\u Lists access groups that have access to upload to that particular UD section.

\d Lists access groups that have access to download from that particular UD section.

+ux Gives access group number x access to upload on the particular UD section the user is currently on.

+dx Gives access group number x access to download from the particular UD section the user is currently on.

+cx Gives access group number x unlimited credits on the particular UD section that the user is currently on.

-ux This is the same as +ux but it takes away access for that access group number x.

-dx This is the same as +dx but it takes away access for that access group number x.

-cx This is the same as +cx but it takes away the unlimited credits for access group number x.

@x Where X is a DOS command. This performs a DOS command "x" for the current directory that the user is on.

% This changes the name of the current UD that the user is on.

& This regenerates a whole UD directory just like "!" does but it only regenerates the file numbers that have been selected in eXchange mode using the "S" or "Sx" command.

X - Toggles eXchange mode on and off. If x is following by a logical drive number then it will go into exchange mode for that logical drive number. Otherwise, the BBS will go into exchange mode for the directory that corresponds to the current UD section the user is in. Access has to be specifically given to a user to allow access to exchange mode and even more access has to be specifically given to allow going into exchange mode of system drives other than the UD directories. See the the remote access section of the documentation describing RM% for more information.

The amount of credits a person has is equal to the amount of posts a person has times the amount of credits the sysop allows per post minus blocks downloaded plus blocks uploaded times the number of blocks awarded per upload (block ratio defined when defining access groups in the Boot).

SYSOP/REMOTE MAINTENANCE SYSTEM

Q Quit to Main. [RETURN] Does the same. When these are typed when going to remote mode from the wait screen, they return the sysop to the wait screen.

? Prints Sysop Maintenance Menu called "sysop"

T This changes the date on the system.

Dx Deletes user number X. User#1 may be deleted only in local mode or by user number one himself. The reason that the deletion of user #1 is allowed is that in the unlikely event that the record of user #1 ever gets corrupted, the sysop may need to delete it so that the sysop can edit the account properly.

Wx This writes a file. X is a number that is used for writing bulletins and for writing forced mail files. The Number X does not have to be there.

WO This writes a file starting where the user last left on in the most recently created message even if the user aborted out of it or had to runstop/restore the BBS in the middle of writing the previous message.

\ [The British Pound] like the one described in the MCI commands section is used for listing the names of the available access groups and their corresponding access group numbers.

$x This lists the directory of BBS LOGICAL DRIVE #x. If x is unspecified, then the BBS assumes the user means BBS logical drive 0:.

S For reading a SEQuential file. The user will be prompted for a BBS logical drive number followed by the filename. Absence of the BBS logical drive in the filename will make the BBS default to reading off of BBS logical drive 0:. A sample user input would be "2:t s 1" to read the filename "t s 2" off of the Subs Logical Drive #.

^ For going into PRINT mode of the BBS. This is only activated in local mode. This command prints a file to the printer.

Ex Edits user NUMBER X or if X is not a number, it will edit the user with the handle that X is. For example, if x = 23 then e23 will edit user number 23. If x = 'brinn', ebrinn will edit the id number of whoever Brinn is.

If the user edits himself then the stats that the user edits in edit mode will be appropriately copied to the variables that make up the user's account in memory.

+ '+' alone edits the next user in the userlog. For example, if the previously edited user was User ID Number 86, pressing + would edit User ID Number 87.

@x This does a dos command on the current directory that has last been read. X is a string like 's0:list' for scratching the file 'list' off of the directory. If no directory has been read then the remote prompt will tell the user which device and drive # he are currently on.

Px Allows the user to define a pattern to list directories with and perform file operations with. For example, type pbbs* will allow the user to look at all the files on a directory that begin with "bbs" when the user gives a directory command.

Cx Copies files number X off of the current directory read to another device. This may be used in conjunction with STACKED COMMANDS to copy multiple files. To select source and destination device,drive #, and dos commands see the explaination for .X below. As an example, if the user wanted file numbers 1 and 2 and 4, he would enter"c1;c2;c4" to stack the commands of the files he wanted to copy.

.X Period plus a Logical Drive number species what the destination drive should be in the copy file command. The source logical drive number for the copy file command is specified by listing the directory with the $x command of BBS logical drive number X that the user wishes to be the source.

! This creates a new NAMELOG file automatically which is a seq file

that stores the names and ID's of all the users for the SUPER FAST handle searching system. It is a good idea to use this command about once every week or two to keep the namelog file up to date. The namelog file is also added to automatically when a user applies.

{LEFT ARROW} This command prompts the user to type in a list of access group numbers seperated by ";". The BBS will then give the appropriate access group code that the BBS will recognize to give those access groups access to a specific section of the BBS. See the Access Group section of the documentation for more information on how access groups operate.

X "x" allows the user to specify the device #, drive #, and new DOS command to be associated with BBS logical drive number 1:. This is used in case the sysop wants to hook a disk drive up to the system that is not configured to fit any of the other BBS logical units already and the sysop wants to access that drive. For example, if the sysop hooked up a device #9, Drive #0 and wanted to view the directory of that drive, then the sysop would specify the Device # to be "9", drive # to be "0:" and the dos command to be "-" and would then type $1: to view the directory of that drive. This is also useful for configuring a destination BBS logical drive number or a source BBS logical drive number for copying files to and from a disk drive that normally are not hooked up to the system. Thus, the reason why BBS logical unit 1: is referred to as the temporary logical unit number is because it is used by the sysop to define disk drives that are only on the BBS temporarily.

.L Loads a file into the message maker when the user is in the "w" command message maker. This option is only for remote level users and users currently using the W file command regardless of their remote access (RM%) configuration.

If the file 'stats' is edited with the w command in the sysop maintenance section, then the variables in memory that correspond to the stats file will be changed as well.

MINI-TERMINAL MODE SYSTEM

This will be a brief description of the functions in the online BBS terminal mode.

When first going to terminal mode, the sysop will be presented with a menu of choices.

The first of these is R. Pressing this will reset the terminal mode back to the main bbs where it normally waits for a call.

The next two most important features are G and A. These take the sysop into the terminal mode itself. G takes the sysop into the graphics terminal mode and A takes the sysop to the ASCII terminal mode. The terminal mode itself can be exited back to the menu of options by simply pressing [F1].

B allows the sysop to change the current baud rate. If the sysop runs his BBS on a 1200 baud modem, the default baud rate is 1200. If he is using a 2400 baud modem, the default baud rate will be 2400 baud.

The {LEFT ARROW} key will take the miniterm into the UD section of the BBS. While in this area, go into the exchange mode of the UD where the uploading and downloading is to be done. Then do the opposite of what would be done if using a terminal program to download and upload. For example, if downloading from another BBS, a sysop would UPLOAD within the

UD section. And if uploading to another BBS, they would DOWNLOAD within the UD section. Typing Q at the UD prompt should place a person back in the terminal part of the BBS. It is advised to rather use the FULL MODULAR TERM for uploading and downloading rather than the miniterm -- this miniterm function exists only as a last resort for quick downloads and uploads in the miniterm.

Also with the {LEFT ARROW} command, the sysop must use the [F1],[F3] and [F5] windowing options to raise or lower the level to the access group(s) that have access to the UD sections that are being downloaded or uploaded to. Raising and lowering the access group in the window automatically places the appropriate security allowances and rights into memory for the BBS to understand which UD sections the person on line has access to.

If the sysop does not type anything in at the menu prompt in the terminal mode for more than 5 minutes, the BBS will automatically reset the BBS back to the waiting for call screen.

MODULAR TERMINAL PROGRAM

A great deal of thanks should go to CyberSage who wrote the public domain Modular Terminal Program on CBASE v2.0. It only had to modified and improved slightly upon porting it over to the v3.0 series of C_BASE.

This modular term is basically used in case the sysop wants to simply be able to use a more sophisticated terminal program that can do more things like upload, download, and use macros on a BBS.

To the v3.0 modular term we have added BUFFERing commands. The Buffer space is approximately 8K or 8192 characters long. The sysop activates the buffer by pressing the [F5] key and turns it off by pressing the [F3] key. The border turns red to indicate when the buffer has been activated. The [F7] key clears the buffer. From outside the terminal program there are options to view the buffer to screen, to screen and modem, to load a seq file into the buffer, and to save the buffer contents.

As an added enhancement, if a sysop returns to the BBS from the modular term, they may use **WO** at the REMOTE prompt to recall what was in the buffer and edit it and save it as a seperate file. This may be more convenient than saving the buffer in the modular term and then later going to remote mode in the BBS and .LOADing the buffer file.

The modular term loads primarily off of BBS Logical Drive number 0:. However, in the BOOT file the sysop can specify which device number to load it off of, so he has his choice of loading the system off of the device inherently associated with the 0: logical drive #, or a different device #.

There is an added function to the modular term that it lacked in v2.0 of C_BASE concerning the disk drives. Before, the sysop had to specifically choose the device to send DOS commands and do file transfers with before each transfer or command. Now, one of the last options on the main terminal menu is to set the files device and drive number. This was added to allow compatibility with Lt Kernals partitioning and also so that the sysop can set the terminal in the very beginning what device number the sysop wants to download or upload files to. This is the device and drive number that the buffering functions also act on for saving and loading buffers to and from the disk. The device number defaults to 8 and the drive number defaults to 0:. The drive number must have a ":" placed after it if the sysop chooses not to press [RETURN] to get the default 0:.

MODULAR SUBS

This has not been made yet but will be considered for future expansion possibilities. To create the modular subs, use the module kit and simply add the subs relevant programming from the main C_BASE program.

The modular programs load the same way the modular term does off of BBS logical drive 0: yet being flexible as to which device # to use.

MODULAR UDS

This is also reserved for future expansion possibilities.

MODULAR REMOTE COMMAND MAINTENANCE

This is reserved for future expansion possibilities.

MODULAR NETWORKING

This is currently being worked on but is also in the line of future expansion possibilities. Since the c/network file is loaded at a prespecified time, the sysop can also make a seperate module that performs housekeeping chores in the middle of the night such as for autopurging old users, checking mail and deleting old mail, checking ud directories and deleting old wares, and so on. The possibilities are endless for the creative programmer!

MESSAGE MAKER

This section of the documentation goes into the features of the C_BASE v3.0 message maker.

The C_BASE v3.0 editor can store up to 8K or 8192 bytes (Characters) of information in it and has an unlimited number of lines capacity. The text when entered by the user is stored automatically underneath KERNAL ROM of the C64 computer. This allows for an almost complete reduction of garbage collection which is common in most BBSes written in BASIC. The input routine for the message maker and for the regular prompts is the regular prompts is equivalent and in almost total machine language now in order to alleviate any occurance of garbage collection that might occur.

The word wrap is operated by COLUMN effect. The user may enter as many cursor movements or graphics that he wants WHILE WORD WRAP IS ON. This is because the BBS will NOT word wrap until the cursor is in the rightmost columns. So he is safe to do any graphics he wants on the screen as long as the cursor is not moved over to the rightmost column. If the user needs to do graphics in the rightmost column, then he can simply turn word wrap off. Note that when the message word wraps, it stores a fake carriage return. This fake carriage return is an [F7] character. This is done so that the BBS can tell the difference between real user initiated carriage returns in a message and word wrapped carriage returns when the BBS performs the variable column output routines to the file to output it in the proper columns.

Special Note: If you originally write a file in 80 columns, but load it in 40 columns using the .LOAD command, the file will remain in 80 columns because of the previous word wrapping that was done before the message was saved. The variable column output is just for Seq File reads to the screen and does not apply to actually loading the message into the editor.

Most commands are initiated by typing ".". The user may also initiate

them by typing "/" to maintain user friendliness and compatibility with other popular BBS systems.

"." or "/" Commands

PERIOD COMMANDS

R For Read (Normal)
V For Read (With Line Numbers Viewed)
M For Read (With MCI Codes Interpreted)
S For Save
D For Delete Range of Lines
E For Edit Line Number
H For Viewing A Help Menu
A For Abort
* For Search With Replace (Replaces first occurance)
G For Global Search And Replace (Replaces All occurances)
Q Quiet Mode Toggle. Turn Quiet Mode on when Buffering stuff up for ultra clear reception.
L For Loading A Seq file into memory (Remote Level or In Remote Maintenance mode)

The search with replace function is used in case the user mispells a word and does not want to go through the drudgery of editing the whole line. The user just simply searches for the word that is mispelled and replaces it with the word he wants!

Also, when the sysop is .Editing a line, he can recreate that line simply by typing CTRLU. Ctrl U recreates the line one character at a time for everytime he presses it! This is an added convenience for users who only want to change one thing in a line they are editing but dont want to have to retype specific graphics that they can simply reduplicate by typing ctrlu a couple of times. This can also be an added convenience when editing certain stats file lines and the user does not want to have to type in a lot of numbers over again just to get to the part of the line he really wanted to edit. CTRL U also operates at normal prompts to recreate information that was deleted but then wanted to recall later on like a list of stacked commands or file selections for example.

The HOME key acts as a recorded Delete within the message and is used to create a "pulling" effect with graphics much like the INSERT character is used to create a "Pushing" effect with c64 graphics mode.

The DELETE key within the message maker will delete past the current beginning of a line if there are more lines above the line that is being deleted. When the current line is fully deleted and the delete key is pressed, the last line reprints itself and becomes the current line that is being input into the message maker. When there are truly no more characters to delete in the message maker, only then will the delete key not do anything.

Other commands within the message maker can be accessed by typing CTRL along with a character.

CTRL COMMANDS

CTRL W Toggles Word Wrap
CTRL L Toggles Upper/Lowercase Graphics Mode
CTRL P Introduces a 1/10th Second Pause into the Message
CTRL U Recreates Line Character by Character
CTRL C Centers the current line

CTRL A Prints Alias
CTRL Y Prints Location
CTRL Z Erases the Last line. If the current line contains some input, CTRL Z instead erases the line in FRONT OF THE USER'S VERY EYES! This erasing of the line function is also available at regular prompts such as the main prompt.
CTRL D Defines a function key [CTRL F]
CTRL F Prints what was defined in the CTRL D function.
CTRL K TOGGLES KURSOR MODE. This is a very special part of the BBS which allows the user to do many things, provided that the user knows what he are doing.

CTRL K when activated will print the function defined in CTRL D everywhere that the cursor is moved. Also, when the user types regularly, the CTRL Function will be printed after whatever letter or character he typed! This can result in being able to use it as a drawing pencil or even animation! For example, if the user defined the function as a bird with its wings up, then cursor over, a bird with its wings down, and then cursor over and erase the last figure, and used this function in CTRL K cursor mode, then whereever the user moved the kursor, the 'bird' would seem to fly!

MCI COMMANDS

MCI stands for message command interpretor. This may seem like a vague name, but it is actually very descriptive. There are certain commands that the user can place in the messages on the BBS which will not output as normal text. They will instead become interpreted and display the functions that the commands stand for instead.

There are two types of MCI Commands in CBASE. The @ commands are accessable to sysop levels only. The \ MCI Commands are available to any user who is allowed to use MCI commands in his access group. \ Shall be used in order to replace the [POUND] symbol on the c64 keyboard that is normally used to mark an MCI Command. The [POUND] key may be found in the upper right hand corner of the keyboard to the direct left of the CLEAR/HOME key.

OUTPUT

\OA Alias
\OB Location
\OC Phone Number
\OD Real Name
\OE Access Group Name
\OF Last Date On
\OG Last Caller
\OH Time of Last Call
\OI Sysop In Or Out
\OJ Sysop Name
\OK BirthDay
\OL Current Date
\OM Current Time
\ON Computer Type
\OO Last Inputed Line (i$)

The output MCI Commands are interpreted properly during variable column outputting so that the message will appear word wrapped correctly no matter how long a variable is in the output MCI command list above.

INPUT

\I1 Waits For A Key To Be Pressed -- The ASCII Value of this key is placed in peek(52992) for future compares if necessary.

\I0 Inputs A Whole Line of Text terminated by a [RETURN] or 255 characters. It calls the Machine Language input routine that is used by the rest of the BBS for entering in data at prompts. Afterwards, the contents of \I0 are placed in i$ and may be viewed with \OO MCI Output command.

FUNCTION

\Sx With X Being a number from 0 to 9 turns slow mode on for reading a message. 9 is the slowest speed, while 0 is the fastest. SlowMode is turned off after a [RETURN] is outputed or a \N is encountered.

\P With two colors placed after it puts the output text into punctuation mode. Punctuation mode is where the text appears in one color and the punctuation is automatically made to look like the second color.

Additionally, if the user adds a **third color**, the BBS will detect this and Capital letters will be printed in that third color -- This is **capital punctuation mode.**

\R With 2 to 5 colors placed after it puts the output text into rainbow mode. The text alternates between these colors in order whenever a character is read in. This could otherwise be described as a character by character rainbow mode.

\L With 2 to 5 colors placed after it puts the output text into line rainbow mode. The text alternates the colors in order each time a line of text is read out.

\W With 2 to 5 colors placed after it puts the output text into word rainbow mode. The text alternates the colors in order each time a line of text is read out.

\N Clears all rainbow modes, punctuation mode, and slow typing back to normal output.

\Txx This performs a TAB on the text to column number xx. xx must be a double digit number. For example, to tab to column 5 the user would use \T05.

\B This is a branching MCI Command. If the true/false flag has previously been set, then the command will skip the output of text when it is reading a message until the label MCI Command is found if the result was false. Compares are only done using the SYSOP MCI commands described below.

\L This is a label MCI Command. If the text output was skipped due to a branch MCI Command, then this label will return the reading of the text to normal.

\Vx This is a Variable MCI Command. This really has no productive sense except for use in the text prompt editor. x is a number from 0 to 9 and basically v0 through v9 are temporary variables that the BBS uses and stores in memory for use in printing out stuff in the text prompts and then throws away again.

For example, in the text prompt that defines the header of a message, the subject of the message is accessed with \v0. However, in another text

prompt, \v0 would be something else. The only place the user will need to use the \V MCI Command is when he are editing text prompts, and even then keep note of what \Vx stands for for each text prompt. For example, if the user is editing the message header text prompt, then figure out in advance by the order the \V's appear in and what prints out normally, what each \Vx stands for. For example, if \V1 is listed after a big line of -'s and before a "[]" then the user can pretty much figure out that that stands for Alias. Likewise, the \V2 that is after the "[]" and before another set of "[]" can be thought of as the Time the message was posted.

As a note to programmers, \Vx calls information from from the input buffer at $CE00. X gives the routine information as to how many carriage returns to screen past before finding the proper thing to print out. 0 prints out the first piece of information in $CE00 terminated with a [RETURN] while 1 scans past that, and prints out the second piece of information terminated with a carriage return. This information is not necessary for knowledge of board operations but it might be helpful to people who are programming something new into the program.

COLOR

These MCI Commands are different from the others in that they are interpreted in the input parser rather than the output routine so that as soon as the user types the MCI Command, it is interpreted and the results displayed so that ANSI users may know what color they are currently typing in without having to perform an MCI Read of the message.

\CA White
\CB Red
\CC Cyan
\CD Purple
\CE Green
\CF Blue
\CG Yellow
\CH Orange (Dark Purple in ANSI/C128 80 columns)
\CI Brown
\CJ Light Red
\CK Dark Grey (Dark Cyan in ANSI/C128 80 columns)
\CL Medium Grey
\CM Light Green
\CN Light Blue
\CO Light Grey

\CP Reverse Video On
\CQ Reverse Video Off

SYSOP

The Sysop MCI Commands use the @ symbol instead of the \ to interpret the data. This is for security purposes. Only people with the appropriate remote bit set will be able to enter this MCI Command into the message editor.

@:x: This command takes a string expression "x" and outputs it. For example @:na$: would output the na$ (name) variable! @:dn$(1): would output the name of UD Directory number one. This command also accepts BASIC Tokens in "x"! This is what makes this command ultrapowerful! For example, the user can print out the left$(na$,2) if he wanted to by using the BASIC TOKEN CHARACTER that stands for LEFT$ in the @ MCI Command. The same holds true for MID$ and every other expression evaluator BASIC Command!

For example, the BASIC TOKEN Character that stands for MID$ is capital-J. So to print mid$(na$,2) which would output the variable na$ with the first character stripped off the user would use @:J(na$,2):

@:x: This command is the same as above except that this time "x" would be a numerical true/false expression. C_BASE automatically detects the difference between the formula between the parentheses being a NUMBER or a STRING. If it evaluates to a String of text, then the BBS outputs that string of text. If it is a numerical value then it does not output but instead stores the numerical value for future referencing with the BRANCH MCI Command \B.

The BASIC Token for the ='s operator is not "=" but we will use it in the following example. If the user wanted to compare i$ against a password in a message the user was typing so that only people that knew a password could see a certain part of the message he would use

```
@:i$="password":
```

This expression evaluates to either true or false. If i$ does currently equal "password" then the expression will be true. If it does not, the expression will be false. The BBS keeps track of TRUE/FALSE expressions with numbers. 0 is FALSE and -1 is TRUE.

In this past example, the user would have preceeded this compare MCI Command with \I0 for inputting a line of text into i$. After the compare command, there would be a \B for branch, then the secret message would be typed, and then this would be followed by a \L End of Branch label to indicate that the message can now be output again.

@:h[COLOR]x: format of the MCI Command is highly specific and prints out the text string "x" in a centered form surrounded by solid horizontal lines. Essentially this MCI Command is what prints out the line in the Message header that centers the subject. The Code "H" at the beginning of the :: expression indicates that this will be a header and [COLOR] indicates what color the user wishes the centered lines to be.

TABLE OF COMMON BASIC TOKENS/KEYBOARD EQUIVALENTS

STATEMENT	ASCII VALUE	KEYBOARD EQUIVALENT
INT	181	COMMODORE-J
ABS	182	COMMODORE-L
RND	187	COMMODORE-F
EXP	189	COMMODORE-X
PEEK	194	SHIFT-B
LEN	195	SHIFT-C
STR$	196	SHIFT-D
VAL	197	SHIFT-E
ASC	198	SHIFT-F
CHR$	199	SHIFT-G
LEFT$	200	SHIFT-H
RIGHT$	201	SHIFT-I
MID$	202	SHIFT-J

OPERATOR	ASCII VALUE	KEYBOARD EQUIVALENT
+ (ADD)	170	COMMODORE-N
- (SUBTRACT)	171	COMMODORE-Q
* (MULTIPLY)	172	COMMODORE-D
\ (DIVIDE)	173	COMMODORE-Z

^ (EXPONENT)	174	COMMODORE-S
AND	175	COMMODORE-P
OR	176	COMMODORE-A
> (GREATER)	177	COMMODORE-E
= (EQUAL)	178	COMMODORE-R
< (LESS THAN)	179	COMMODORE-W
NOT	168	COMMODORE-[POUND SIGN]

As a special HINT, if the user wants to reverse the value of an expression from TRUE to FALSE or vice versa then precede the expression x with a NOT Token and surrounded by parenthesis. For example, @:[NOT TOKEN](i$="password"): would now be TRUE only if the password was NOT equal to i$!

STATS FILE DESCRIPTION

The "stats" file that resides on the system disk contains almost all of the information that the system needs to run on a day to day basis. The file is set up fairly logically with major sections such as subs and uds and libraries being separately organized within the main stats file. As a sidenote, this section is more useful to people that actually know C64 BASIC although this should not be necessary to grasp the idea of what the variables stand for.

```
cl
vi$
da$,em,em%,nu%,mu%,dg%,ns%,nd%,lb%
dr(1);dr$(1);dc$(1)  (Device,Drive#,Dos Command For Temp)
dr(2);dr$(2);dc$(2)  (Device,Drive#,Dos Command For Subs)
dr(3);dr$(3);dc$(3)  (For Mail)
dr(4);dr$(4);dc$(4)  (For Libraries)
dr(5);dr$(5);dc$(5)  (For Modules)
su$(ns%)
rq%(ns%),rw%(ns%),z%(ns%),ra(ns%),wa(ns%),so%(ns%),ap%(ns%)
dr(nd%+5);dr$(nd%+5);dc$(nd%+5) (Device,Dr#,DOS Cmd for UDs)
u1(nd%),u2(nd%),u3(nd%),uo%(nd%),dn$(nd%)
lb%(lb%),lb$(lb%)
lc$
tc$
```

CL — This is the total number of calls made to the system

VI$ — This is the system name of the sysop

DA$ — This is the current date on the system

EM — Current number of mail files on the system

EM% — Maximum amount of mail files the system is allowed

NU% — Current number of users on the system

MU% — Maximum amount of users the system is allowed

DG% — Number of messages stored per packetÜfÜ

NS% — Number of subs

ND% — Number of UD sections

LB% — Number of library sections

DR()	Device #
DR$()	Drive #
DC$()	DOS Command
SU$()	Title of Sub
RQ%()	Current number of messages on a sub
RW%()	Max number of msgs on a sub
Z%()	Highest message number in range of sub
RA()	Access Code of Who can Read Messages on that sub
WA()	Access Code of Who can Post Messages on that sub
SO%()	ID Number of person who subops that sub
AP%()	If it equals 1, anonymous posting is available
DR()	Device Number of UD Dirs all in a row
DR$()	Drive number of UD Dirs all in a row
DC$()	DOS Command of UD Dirs all in a row
U1()	Access Group Code of who can upload on that UD dir
U2()	Access Group Code of who can download on that UD
U3()	Access Group Code of who has unlimited credits on the UD
UO%()	ID Number of UDop for that UD dir
DN$()	Name of that UD Directory
LB%()	Access Group Code of who can access that library
LB$()	Name of Library
LC$	Alias of last caller
TC$	Time of the last call

PROGRAMMING C_BASE AND MODULES

In order to modify CBASE or write modules the user should know the BASIC language on the Commodore 64 to some (no pun intended) basic degree. The 64 Manual that comes with the C64, C64C, 128,or 128D teaches what he needs to know in the first few chapters.

Also, the following pages contain basically the same type of information as the CBASE v2.0 manuel on modifying C_BASE. There is a seperate programmers manual to C_BASE v3.0.

OUTPUT TO MODEM AND SCREEN

In order to have a string printed to both the modem and the computer

screen, the character @ is used in place of the BASIC command "print" or its abbreviation "?". For example, if I wanted 'Hello I am user #1' to print to the screen I would type

print"Hello I am user #";un (Where un is equal to the current user number)

In the BBS program however, a person would use the following format:

@"Hello I am user 1"+str$(un)

Notice that in the print statement a programmer can print numbers with impunity. However, in the @ command the person has to make sure that the person is printing only string expressions. This means the person has to take numeric expressions and convert them to strings using str$ BASIC command if he wishes to print it. As a rule of thumb if the person would not be able to set any string variable equal to the whole expression that he is trying to print without an error, then he will not be able to use that expression with the @ command.

If the person wishes to print a string of text without a [RETURN] printed afterwards, he would use a semi-colon after the print statement in BASIC.

print"Hello I am user #";un;

In the BBS program however, in order to have the same thing print to BOTH the modem AND the screen the following would be used:

{LEFT ARROW}"Hello I am user #"+str$(un)

{LEFT ARROW} is just like the @ command except that it neglects to add a [RETURN] at the end of the text.

Certain characters that are printed with the @ or {LEFT ARROW} command are translated differently for the convenience of the sysop. The [F7] character is always translated as a [RETURN] character. So if a person wanted to print five [RETURN]s to the modem and screen a person would use "[F7][F7][F7][F7][F7]". The characters [F1], [F3], and [F5] are tied into the MultiColor Basing system. These characters translate to the colors defined for them. [F1] is the bold color, [F3] is the light color, and [F5] is the medium color.

Nearly all of the prompts that are inherent in the BBS itself are installed in a bubble of memory upon boot up. These prompts are numbered 0-120 and are visibly installed the first time the sysop boots up the system. Within the programming of the BBS itself, a person will see where a prompt is being printed by the format "poke997,x:sys51048". So if prompt number 100 is being printed out in the BBS, the person would see poke997,100:sys51048 in that area.

INPUT FROM MODEM AND SCREEN

Normally, in BASIC if a programmer wanted to input something one would type a statement similar to:

input"What do you want to do now";i$

And then what a person typed would be placed in i$.

The equivalent to this statement in the BBS and what should be done INSTEAD of using BASIC's Input statement is "gosub5"

The equivalent in C_BASE would be the following:

```
r$="What do you want to do now:":gosub5
```

Gosub5 goes to a routine that prints r$ to the modem and screen and then inputs text into i$.

A person could technically substitute "r$=" with the {LEFT ARROW} BASIC command and in most cases this is the more appropriate expression.

PRINTING A FILE TO MODEM/SCREEN

To print a SEQ file to modem and screen such as the seq file for the open title screen called 'open', a person would type

```
o$="open":gosub25
```

Within a module, a programmer would use the variable ft$ in place of o$ to maintain module kit compatibility with the v2.0 modules. o$=The Filename to Read into the computer.

OUTLINE OF V3.0 BBS PROGRAM

MAIN PROMPT COMMANDS RESIDE AROUND LINES 6710-6800.

REMOTE COMMAND PROMPT COMMANDS RESIDE AROUND LINES 6000-7000.

SUB BOARD COMMAND PROMPT RESIDES AROUND LINES 3990-4300.

MAIL SECTION RESIDES AROUND LINES 4300-5000.

UPLOAD DOWNLOAD COMMAND PROMPT RESIDES AROUND LINES 3430-5600.

MESSAGE MAKER ROUTINES RESIDE AROUND LINES 200-900.

LIBRARIES, OMNI MESSAGE, MACROS RESIDE AROUND LINES 8000-END.

CARRIER DETECT RESIDES AT 80-200.

WAITING FOR CALL/LOGON PROCEDURE RESIDES AT 7120-8000

If the person wishes to use the Multi Color Basing System, the colors are translated by using FKey characters in place of colors. The following is how they are translated:

F1 = Dark Color
F3 = Light Color
F5 = Medium Color

After the person modifies the BASIC source code, he will want to compile it using the compiler that comes with the program. Using any option that allows a person to compile the program WITHOUT a Run/Time Module is a good idea since it will shorten the compiled program by 24 blocks and also save compiling time.

If the person is compiling the program in 64 mode of a 128 it is advised that it be compiled in 2Mhz mode as that will speed up the

process by about 70%.

Look for a more detailed and extensive programmers guide to CBASE v3.0 to be released soon.

As a final note, in modifying the BBS, please do not take out the credits to Gunther Birznieks/Prof. Plum and the BBS Main infoline numbers in the BBS. Obviously, credit should go where credit is due. Also, do not change the version number of the BBS. Only the author is entitled to do that since if other owners call the BBS and see a higher version number they will get on the author's case thinking he has released a new version without saying anything, when the author in fact has not done that. It will also violate the ownership license if the authorship credits are removed or the version number is changed by anyone other than C-BASE BBS Systems.

MODULE KIT

Creating a module is simply as easy as starting to program the module at line 7500 in the module kit after it is loaded in. The module kit is an overlay program which contains rudimentary BBS routines including file reader, module loader, input, output, and many other types of routines. It is just like the BBS stripped of its subs, uds, remote maintenance, logon procedures, mail and other local BBS options.

Table Of Common Gosubs To Use In The Programming

GOTO205 Quits And Loads The Main BBS Program Back In.
GOSUB3200 Goes To The Module Menu List
GOSUB31 Explained Below In More Detail For Input#8,i$ from disk
GOSUB5 For Inputting (Explained above in How To Mod The Main Program)
GOSUB25 For File Reading (Explained above in How To Mod The Main Program)
GOSUB32 For taking the contents of i$ and striping graphics, and converting the contents to lowercase. This is the same GOSUB for the Main BBS program.

The Variables in the Module that are carried over AUTOMATICALLY to the module are as follows:

SL for Security Level
NA$ for Alias
BU for Blocks Uploaded
BD for Blocks Downloaded
UN for User Number
DA$ for Current Date
BR% for Baud Rate
PEEK(848) = 1 in Graphics Mode, = 0 In ASCII
PEEK(840) = 1 in Local Mode, = 0 when User is on line with modem

The Following Paragraph Is For More Advanced BASIC programmers and concerns the incorporation of actual system variables into the modules. This is used for the purpose of making modules that emulate the UD section, the subs section, or other sections of the BBS. It is possible to have the stats and userlog variables read into the module kit. This is accomplished by removing the statement that makes the module kit skip over the gosubs that read stats and userlog at lines 2080-2090. The removal of the skipping will allow the routine that would not normally be executed to read the stats

and userlog variables to accomplish its task. Likewise, if the person wishes for the stats and userlog variables to also save when the module is done, the routines that skip the stats and userlog save in the module kit can be taken out in lines 2050-2080.

One final note, if the user inputs from a disk device in the main program or in the module, it is useful to use GOSUB30 instead of INPUT#x,i$ because INPUT# only accepts up to 88 characters before crashing out with an illegal quantity error. GOSUB31 is equivalent to GOSUB30. These routines take input from channel #8 only and places the information into i$. Also, this routine only stops reading upon encountering a carriage return and skips past commas and colons. In summary, GOSUB31 (To Maintain CBASE v2.0/v1.0 compatibility) and GOSUB30 are equivalent to input#8,i$ but without the constraints.

VARIABLES COMMONLY USED IN CBASE

This includes the stats variables that are described in the stats file portion of the documentation.

UN	User Number
NA$	Alias
YL$	Location
BR%	Baud Rate
CD%	Calls Today
CL	Current Caller Number
PA$	Password
RN$	Real Name
PN$	Phone Number
MW%	Number of Mail
SL	Security Level
RM%	Remote Bit
JU	Joined/Unjoined
LL%	Pause On/Off
CL%	Column Length (39 or 79)
NP%	Number of Posts
FD	Files Downloaded
FU	Files Uploaded
BU	Blocks Uploaded
BD	Blocks Downloaded
BK	Blocks Free or Number of Blocks a File is
LO$	Date user was last on
CX	Holds caller number a user had when last on

The Following variables would be set before gosub5 for input is issued:

Use re = 1 for a password type prompt where the character's are echoed back as '' instead of the actual character.

Use rp = 1 When the person wants only one letter to be typed in gosub5 input.

Use rp = 2:gosub5 When the programmer wants only Y or N or RETURN to be typed in at Gosub 5 input and then when Y is typed, YES! is printed, and when N is typed, NO. is printed. When the person wants to do the same thing in the BBS program itself the person should use gosub221. Gosub221 already has a gosub5 within it so the programmer does not have to use gosub5 after doing gosub221.

CONVERTING MODULES

FROM V2.0 OF C_BASE

Converting modules from v2.0 of C_BASE is actually fairly easy and requires minimal reconstruction of a module file since the only thing that is different is the module kit overlay that is attached to the main program that starts at line 7500.

1: Load in the v2.0 module basic.

2: Delete lines 0-7499 (The Module Kit) Leaving the module itself intact.

3: Save this file to disk.

4: Load the v3.0 module kit.

5: Using the APPEND function available on some of the more advanced fastload cartridges or BASIC utility packages the person should now attach the file that was saved to disk in Step 3 to the v3.0 module kit.

If the person does not have a cartridge that supports the APPEND function then they may use the following technique. If the person does not understand this technique then they should not use it, because it involves changing some critical pointers in the memory of the C64 around.

[A]: Load The Module Kit
[B]: print peek(43), peek(44), peek(45), and peek(46)
[C]: Now record what each value is on a piece of paper
[D]: Then poke43,peek(45)-2:poke44,peek(46)
[E]: Then load the module with deleted 0-7499 lines. Load it with ",device" NOT ",device,1"
[F]: After that, poke43,original peek(43) value and poke44, original peek(44) value in step [B]
[G]: Now the two BASIC files will be appended to each other! Always make sure that this procedure worked before saving the file though! To do this the person can list lines 7120-7600 and see whether they merged smoothly at line 7500 where the actual module programming starts.

Also, in step [D], if peek(45)-2 is less than 0 then use the value 0 instead and when poking 44, poke 44 with peek(46)-1 instead.

6: Save this file.

Now the module is fully converted and can be compiled! That is all there is to it. Some modules may have a sysop remote maintenance procedure or a check for sysop level within the module itself. This will have to be changed by hand since v3.0 does not check for remote access using the variable "sl" but instead uses the remote bit variable "rm%". In general the best strategy is to look for places with "ifsl>10" and replace it with "ifrm%>0".

FROM OTHER BBS SYSTEMS

Converting game modules and other modular items from other BBS systems is not as easy a task as converting CBASE modules to higher versions. But the strategy essentially is actually still fairly

easy when keeping in mind that all BBSes operate essentially the same way as far as what types of routines it has to call.

For example, Some BBSes like CBASE 128 uses o$="String to output":sys4864 as its output. CMBBS uses o$="String to output":gosub3. Color uses sysc(x) after defining a$. And even other BBSes use extended homemade BASIC commands like CBASE 64's @"String to output" command. But they all have one thing in common; they all have a routine that takes some string and outputs it to the modem and screen. Converting a module from another BBS is merely a matter of figuring out the similarities.

And finally a few words should be said about Authorship etiquette. If a person plans on converting someone else's programming in a game module from another BBS to CBASE, the converter's name should not be in headlights at the expense of the original author credits. Please, above all else, always keep the proper credits of who actually did the programming for the module in the main module.

Afterall, if a person feels they actually did more work than the original programmer then they should have written the module from scratch for CBASE. It is that plain and simple.

Also, if a person sees a module has been written by someone else and wishes to convert it, first make an active effort to contact the original programmer. If a programmer is able to be contacted through a BBS number displayed in his or her module, then an active effort should be made to call that BBS and ask in a polite and nice manner whether he or she would mind having a conversion of his or her module as long as the credits are intact.

In fact, the original author may even be flattered at the chance of having his BBS number and name reaching more places than ever before. The original author may even be nice enough to help with the conversion and explain the logic of the program which would save a considerable amount of work! So please contact the person.

The one reason that an author might be hesitant to allow their module to be converted is if they are selling the module or plan to selling their module. As a corollary, do not sell other people's work; this is not ethical. Secondly, if the person is currently selling their module for X BBS that they currently run, then usually something can be negotiated. Most programmers would like the opportunity to sell their program on his X BBS for CBASE as well and increase their own profits while the CBASE owners gain a good module.

APPENDIX A - DRIVE SETUP

C_BASE 64 operates on a drive referencing system that is setup by sysop defined logical units. Logical Units within the BBS are not to be confused with logical units in a Lt Kernal harddrive although they operate on the same principle.

Basically each and every section of the BBS is assigned a NUMBER. This number is used to reference the information about the drives that that section of the BBS accesses. Throughout this documentation these numbers will be referred to as either BBS LOGICAL UNIT or LOGICAL DRIVE.

The logical drive numbers start at 0: and can continue all the way up to 99:.

0: System Logical Drive holds userlog, menus, callers file

1: Temporary Logical Drive. When a user goes to a UD, the UDs drive information is stored in this logical drive number. When a user enters eXchange mode in the UDs, the information about the drive he is accessing during exchange mode is passed here also.

2: Subs Logical Drive holds all messages, and sub title screens.

3: E-Mail Logical Drive holds all the e-mail.

4: Libraries Logical Drive holds all the library files.

5: Modules Logical Drive holds all the module/on-line game files

6: Upload/Download Directory Number 1 Logical Drive information stores files that are uploaded and files to be downloaded.

7: Upload/Download Directory Number 2.

8: Upload/Download Directory Number 3. All Logical Drive numbers from 6: to 99: correspond to a UD directory. 6: to UD1, 7: to UD2, 8: to UD3, 9: to UD4, 10: to UD5, etc.

For each logical drive the following parameters must be specified: DEVICE #, DRIVE #, and DISK INIT COMMAND.

DEVICE #: A 1541 is normally DEVICE #8. All drives must be hardwired to their proper device numbers prior to attempting to use the system. It is advised not to use any drives which have their device number changed via software. Also, do not have two disk drives being referenced as the same device number.

DRIVE #: All single drives have DRIVE # 0:. Dual drives are usually accessed as 0: and 1: to access to the two disk drives that exist with a single device #. CMD harddrive and Lt Kernal owners can enter the partition CP partition/Lt Kernal logical unit # here. The entered drive number should always be followed with a ":" character.

DISK COMMAND: Enables a command to be sent via the drive command channel before the drive is accessed. The main reason this option exists is to support various HARD DRIVE and 1581 partitioning systems. For drives which do not require a DISK COMMAND to be sent prior to access, this should be set to a dash ("-").

HOW TO ACCESS LOGICAL DRIVES FROM WITHIN THE BBS

Whenever a filename is asked for in C_BASE, merely prefix the filename with the logical unit number followed by a colon (":") to access the filename on the disk drive that is specified for the corresponding BBS Logical Drive.

EXAMPLE ONE:

Parameters of Example -- 0: logical drive for system is set to device #8, drive # 0:, DOS Command - and the user wishes to view the open screen.

When the BBS asks for the filename the user would enter "0:open". The prefix tells the BBS that it should open the file on BBS logical drive #0. Logical Drive #0 informs the BBS that the file should be opened on device 8, drive # 0:, and no dos command needs to be issued before viewing it.

EXAMPLE TWO:

Parameters of Example -- 2: logical drive for messages is set to device #9, drive #0:, DOS Command "/messages" (For 1581 partition) - and the user wishes to view the title screen of sub #1 named "title 1".

When the BBS asks for the filename the user would enter "2:title 1". The prefix tells the BBS that it should open the file on BBS logical drive #2. Logical Drive #2 informs the BBS that the file should be opened on device 9, drive # 0:. Furthermore, the DOS command "/messages" will be sent to the drive before accessing AUTOMATICALLY! So the file will be accessed on the partition easily without the need for typing dos commands to enter partitions when the user is editing and reading files!

These two examples displays the primary advantages of this BBS logical drive referencing system. The first is that a person no longer has to keep track of where to keep certain system files. A person no longer has to memorize what device number to store messages on, or which Lt Kernal Logical unit to store libraries on because it is all STANDARDIZED to PRESET BBS Logical Drive # assignments. Regardless of what device #, for example, that a person places the libraries in, a person will be able to access library files in remote mode on ANY CBASE system by merely Prefixing the filename with 4:! The second advantage is with dealing with complex partitioning drives. ICT HardDrives and 1581s both have a complex series of DOS commands that a user has to issue before accessing filenames within their partitions. BBS logical drive assignments ELIMINATE the need for dealing with those DOS commands because the BBS issues the DOS command for the user in advance since it already knows via the Logical Drive Prefix what DOS command needs to be assigned to the disk drive device!

EXAMPLE 3:

If a sysop wanted to read the directory of whatever device held the subs section, the sysop would type **$2:** at the REMOTE prompt. Likewise, **$3:** would read the EMAIL Directory, **$4:** would read the libraries directory, **$5:** would read the modules directory and so on. If the sysop wanted to read the directory of the system files like stats and userlog, they would type **$0: OR simply $** -- Remember that CBASE assumes 0: Logical Drive as the default drive. Notice that the numbers given after the $ command correspond to the sections of the board that is discussed in the table of logical drives above.

EXAMPLE 4:

A sysop wants to read the mail of user #130. According to the naming conventions of CBASE, the filename for mail is a [SPACE] + USER#. So in this case, the mail would be "[SPACE]130". To read the mail, the sysop would type **S** and return at the REMOTE Prompt.

This would bring up the **LU:FILENAME:** prompt. At this prompt, the sysop would type **3: 130** and press [RETURN]. The prefix of **3:** is used before the filename to tell CBASE v3.0 that the file may be found on the device that the EMAIL DRIVE (3:) is configured for.

EXAMPLE 5:

The sysop wants to read the stats file out to the screen. Once again, at the remote prompt, they would type **S** to read a sequential file. However at the **LU:FILENAME** prompt, they would type **0:stats** because 0: is the LOGICAL DRIVE NUMBER that corresponds to the system files. **Note**, that we could have left off the 0: because CBASE uses 0: as the DEFAULT LOGICAL

DRIVE if none is given.

EXAMPLE 6:

If the sysop wants to read a file off of the Device that UD #1 is stored on, they would use 6: as the LOGICAL DRIVE #. Likewise, if a sysop wanted to read a file off of UD #2, they would use 7:. A sysop wanting to read a file off of UD #3 would use 8:. **Notice**, that Logical Drives 6 and above correspond directly to the device's that are configured for UD sections 1 and above.

APPENDIX B - ACCESS GROUP SETUP

C_BASE 64 v3.0's security system is based on ACCESS GROUPS. An ACCESS GROUP is a series of settings defining what features of the BBS a user belonging to that ACCESS GROUP can access. C_BASE 64 supports FOURTEEN SYSTEM OPERATOR DEFINED ACCESS GROUPS! When editing a user's access only the ACCESS GROUP number has to be specified.

CHANGING ACCESS GROUP PARAMETERS IN THE BOOT

There are a few parameters that the BBS looks at that are hardwired within each access group and these are determined in the boot file. After completing the access groups, it is a good idea to save them. The file on the disk that contains the access group information after being saved is "ag" -- a program file. The access groups are stored as a program file to save time when loading.

The first question is what the name of the access group is. Each access group can only have 30 characters maximum for the group name. It is advisable that the names not be made too fancy or too complex since this may result in going over the required capacity. Also, do not use commas, semi-colons, or colons in an access group name. Pressing [RETURN] at this prompt will default to the current group name.

The other questions deal with system access. Library access determines whether the person is allowed to look at libraries or not. Mail System access allows a person into mail or not. The same holds true for Modules access, sub board access, UD access, Omni Msg making access, Macro Edit access, and MCI Command Access.

The question that asks whether the access group can bypass the post checks is asking whether or not the sysop wants the user to be able to get to UDs and Modules regardless of how many posts he or she has made on the system during that call. Recall that when having configured the BBS in the BOOT file, one of the questions asked how many posts are necessary before going into the upload/download area and in the modules. This allows a user to bypass the number that was entered here.

Other prompts deal with numerical constraints on their access. For example, Calls Per Day limits the amount of calls a person may make to the system in one day. Time Limit (1-9) is asking for how many minutes a person is allowed in one call multiplied by 10. UD Block Ratio wants to know how many blocks can one download for every block uploaded to the BBS. If the block ratio is 0 then that access group bypasses the credit system. Credits needed per module is multiplied by 10 when a person uses the module. If the credits needed per module is 0, then anyone with that access group can play and no credits are subtracted. Otherwise, ten times the amount entered here will be subtracted. For example, if the sysop puts 10 for the credits needed per module, 10 times 10 (100) credits will be subtracted everyime a person with that access group plays a module.

ACCESS GROUP CHART WORKSHEET

Access Group #	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14
Library Access															
Mail Access															
PrimeTime Access															
Module Access															
Sub-Board Access															
Read Callers Log															
BBS List Read/Add															
Back To Back Call															
300 bps UD															
1200 bps UD															
2400 bps UD															
OMNI-Msg Write															
Macro Write															
MCI Cmd Usage															
Bypass Post Check															
Max Calls/Day															
AM Time Limit*10															
PM Time Limit*10															
Creds Per Mod*10															
U/D Block Ratio															

SPECIFYING ACCESS CODES FOR ACCESS GROUPS

While some of the access group functions are defined in the boot for each particular access group, there are certain major parts of the board that require that a special access group code be given to determine which access groups can enter that area. These areas include reading messages on each sub, being able to post messages on each sub, uploading to each UD, downloading access to each UD, unlimited credits access for each UD, each library, and every module.

As an example, when a person sets up the "mod" file and only wants certain access groups to access a particular module, they will need to use the ACCESS GROUP CODE as the number entered here.

Group	Code
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024
11	2048
12	4096
13	8192
14	16384

The group code is equal to the number two to the group'th power. So if the group is 1, then the code is 2 to the 1st power (2). Likewise if the group is 7 then the code is 2 to the 7th power (128).

To calculate the access code for a particular area a person merely adds together the code numbers for the access groups that person wants to allow in that section. For example, if a sysop wanted access groups 1,2 and 14 to access a section that sysop would use 16390 (2+4+16384) as the access group code number.

The BBS itself also calculates the codes for a user if the user does not feel like adding them by hand. At the remote maintenance prompt the {left arrow} may be entered and this will give a prompt ">>" at which the user types the numbers of the access groups he wants to access a section seperated by ;'s. So for the previous example, a person would type "1;2;14" at the ">>" prompt, and the bbs would give the number 16390.

SECONDARY SECURITY LEVEL (SECURITY2)

When editing a user, one of the options to edit is security2. This was added several weeks after the release of 3.0 to supplement the ACCESS GROUP security levels discussed above.

Essentially, Access Groups grant RIGHTS to certain areas of the BBS subs, uds, libraries, and modules. There will be some cases where the sysop will need to allow a person who has one access group, the ability to see and do the things that another access group sees and does. This is done through the Secondary Security Level -- Hereinafter referred to as Security2.

Security2 is a granting of access group rights to a user that does not normally have access to what those access groups have.

Security2 is equal to the sum of the access group codes to the access group the sysop wants the user to have rights to. For example, if the sysop wants the user to access the combined subs, uds, modules, and libraries of both access groups 1 and 9, security2 would equal 2^1 + 2^9.

Or

security2 = 2 + 512 = 514. Where 2 = 2^1 and 512 = 2^9.

Note: Security2 must always include the original access group code as part of the calculation. Security2 is always set by default to equal 2^[ACCESS GROUP NUMBER] even if nothing was entered for security2. For example, if the user's primary access group was 10, security2 would equal 1024 (= 2^10) by default.

EXAMPLE:

ACCESS GROUP 4 = NORMAL LEVEL WITH 30 MINUTE TIME LIMIT
ACCESS GROUP 5 = HIGH LEVEL USERS WITH UNLIMITED TIME
ACCESS GROUP 6 = ROLE PLAYING GAMERS (HAS ACCESS TO ROLE PLAYING SUB BOARD)
ACCESS GROUP 7 = WOMEN ONLY (HAS ACCESS TO WOMEN ONLY SUB BOARD)
ACCESS GROUP 8 = CBASE OWNERS (HAS ACCESS TO CBASE OWNERS SUB)

CASE 1:

Sysop has granted a user access group 4 however, he discovers that the user is female and is a CBASE owner and wants to give her access to BOTH those subs. Instead of setting up a seperate access group that grants access to both CBASE AND WOMENS subs, he decides to grant her the rights to those subs by using security2!

Since the women's sub is ACCESS GROUP 7 and Cbase is 8, the security2 will include the sum of 2^7 + 2^8. It will ALSO include 2^4 because the original Access Group that person belongs to is access group 4. **Remember** to always include the main original access group a person is in the calculation of security2.

Security2 = 2^4 (FOR BASE ACCESS GROUP RIGHTS) + 2^7 (FOR WOMEN ONLY) + 2^8 (FOR CBASE OWNERS ACCESS)

CASE 2:

The Sysop has a user on the system who has access group 5 and unlimited time. He has just purchased a copy of CBASE for himself and the sysop wants to give him access to the CBASE owners sub. Therefore, security2 will include 2^8 (8 being the CBASE Access Group) as part of the calculation. The second part of the calculation will be 2^5 (For the BASE access group rights of that user).

Security2 = 2^5 (FOR BASE ACCESS GROUP RIGHTS) + 2^8 (FOR CBASE OWNER ACCESS).

CASE 3:

The sysop has a user of BASE access group 4 and security2 set to also access a women's sub. She has recently voiced an interest in joining

an online role playing game and the sysop wants to enter her into the game.

Her security2 before voicing interesting the role playing game was equal to 2^4 (FOR BASE ACCESS GROUP -- Always remember this) and 2^7 (For womens sub access).

Now, the sysop adds her to the Role playing sub by adding 2^6 (For role playing sub).

Security2 Now = 2^4 (For Base access group) + 2^7 (For women's sub) + 2^6 (For role playing sub).

HINTS ON SECURITY2:

As with the general setup of access groups for subs, uds, modules, and libraries, the sysop may use the {LEFT ARROW} command in REMOTE MODE in order to determine the number security2 should be. For example, at the >> in remote mode, type 2;5;7 for the number security2 should be in order to include the rights to access group 2, 5, and 7.

After reading through the three case examples of using security2, it should be almost obvious as to the motivation behind security2. WIthout security2, the sysop would have had to set up a seperate access group to control each permutation of who are the female users, role playing game people, and CBASE owners on his BBS. The access groups would have looked like

1: CBASE OWNERS
2: RPG
3: WOMENS
4: CBASE + RPG
5: CBASE + WOMENS
6: CBASE + RPG + WOMENS
7: RPG + WOMENS

Already, 7 access groups would be taken up by the different permutations. But with Security2, we need only have 3 access groups and merely combine the sum of the rights in one Security2 variable!

As a sidenote, when editing a user's level, if the new security2 does not have as part of its sum the new primary access group code, then security2 will be set to be equal to the new primary access group code and wipe out access to the other access groups until recalculated. This was done to avoid confusion when lowering levels for misconduct on a BBS -- Obviously a sysop would be quite annoyed to find a person lowered in level would still have access to, for example, a CBASE OWNERS SUB BOARD and UD when it was discovered that the user had actually pirated it instead.

REMOTE ACCESS LEVELS EXPLAINED IN DETAIL

When a person edits a user's main base security there are two numbers the user needs to look at. The first number is the actual access group that the user belong to. The second number is the remote access bit. This is configured for the user independant of the access group given. Throughout this documentation the value of this remote access bit will be called "RM%".

For BASIC programmers this is also the integer variable in the BBS basic that holds the value of the remote access bit.

Please note that RM% should never ever total above 255. This is because RM% is pased through to the modules as one poke and a value above 255 would cause an "illegal quantity error".

Each remote part of the BBS has its own remote access code that RM% must be set to. The codes are given in this section. If a person wants to give a person access to several remote parts of the board then they must add together the code numbers to get the correct value. An example is given at the end of this section of the documentation.

CODE NUMBER	RESULT
0	NO REMOTE ACCESS AT ALL
1	ALLOWS ACCESS TO REMOTE MODE BY TYPING REM AND ALSO ALLOWS ACCESS TO MOST UDOP AND SUBOP COMMANDS
2	ALLOWS ACCESS TO DOS COMMANDS IN REMOTE MODE AND UD SECTION
4	ALLOWS ACCESS TO FORMATTING THE DISK. FORMAT IS NORMALLY FILTERED WHEN ONLY CODE 2 IS PRESENT
8	ALLOWS ACCESS TO EXCHANGE MODE IN THE UDS
16	ALLOWS ACCESS TO ENTER EXCHANGE MODE IN SYSTEM DRIVES AS WELL AS LONG AS "X" HAS THE BBS LOGICAL DRIVE NUMBER AS ITS SUFFIX
32	ALLOWS ACCESS TO VIEWING PASSWORDS WHEN EDITING A USER. NORMALLY, BEING ABLE TO SEE AND EDIT A USER'S PASSWORD IS FILTERED OUT
64	RESERVED
128	RESERVED FOR NETWORK MODULE (INDICATES A NETWORK USER).

RM% CAN ONLY BE CHANGED BY A PERSON WHO HAS A REMOTE ACCESS OF 255 OR IN LOCAL MODE. 255 MEANS ALL THE BITS FOR REMOTE ACCESS ARE TURNED ON ALREADY (255=1+2+4+8+16+32+64+128). LOCAL MODE means that the sysop has logged onto the BBS without the aid of a modem connection.

LOCAL MODE may be SIMULATED for RM% changing purposes while a user is on-line by pressing the [F6] key to put the current user on hold. Keeping this in mind will help in the future.

As an example, the RM% value a sysop would give a person if that sysop wanted to give them access to remote mode, exchange mode without system drive access, dos commands without format, and no ability to view passwords would be a result of adding the CODES 1+2+8. So when asked to enter the Remt Bit Access, the sysop would enter 11 which is the result of adding 1, 2, and 8 together.

APPENDIX C - TEXT PROMPT EDITING IN THE BOOT

There are 121 (0-120) sysop definable prompts in CBASE. Prompts are the messages and questions that the BBS outputs to the user during the course of his or her stay on the BBS.

Please note when editing the text prompts, that if planning on using one of the noncommercial fastloaders built into the BBS itself, there is only a leeway of approximately 100 bytes or characters that may be added overall to the prompts. That is to say, each text prompt takes up a certain amount of memory and they all add up to a large total. If it is desired to make one prompt much longer than it is now, then it is

advisable that one prompt be made shorter unless the overall changes are within the 100 byte limitation. By the way, this limitation is generalized, not specific, so do not go around counting bytes unless it is considered good for the health.

Without using built in fastloaders then there is around a 1500 byte leeway. Basically, when going above the original 100 byte leeway then the prompts will start taking up the space that is normally occupied by the built in fastloaders. This is an incredible amount of space and will allow being able to make the text prompts just about as long and complex as desired.

When editing the text prompts, the [F7] character may be used to simulate a [RETURN]. For all intents and purposes the [F7] is cosmetically equivalent to the [RETURN] character. However, the reason to use [F7] is because prompts themselves are ended by the [RETURN] character already.

Remember that the [F1],[F3], and [F5] characters in the prompts are for the multicolor basing system. When the BBS reads in an [F1] character it replaces it with the current "dark" color. [F3] and [F5] characters get replaced with "light" and "medium" colors respectively.

Another suggestion that should be heeded is to not edit the prompts until becoming familiar with the BBS prompts. Try running the system for at least a week with the preinstalled text prompts and then edit them after a person can get a decent feel for which prompts belong where in the BBS program.

When finally getting around to editing the text prompts, a person could be intimidated by the MCI Commands and variables that make up some of them especially the message header, subs, ud, remote, and main command prompt. There are two ways to treat this. A person can either just not edit those prompts and only edit the ones that look straightforward or the user can experiment. Experimenting should not hurt the system; if a mess up occurs, simply scratch the "text" program file off of the disk to start with the preinstalled prompts again.

When editing the prompts that have MCI commands in them, use logic and common sense as the most valuable tools. For example, if a prompt said: "You have @:D(a): new messages!" and have no idea what "@:D(a):" means, then use common sense! Obviously a person does not have to be a genius to understand that @:D(a): is the computer's way of saying [The Amount of New Messages]. Just remember to type that part of the prompt in letter for letter and include the upper/lower case each character is written in since the MCI Commands are case sensitive.

When done, the text prompt file may be saved. If a mistake was made, the "text" file may be scratched off of the disk drive. The "text" program file stores the information as a binary machine language file to save space and time when loading.

Some users of a terminal program called "HandyTerm" may want the download of single files to automatically make the term go into a RECEIVE MODE. Modifying prompt #77 and inserting several CTRL-I's in front of the **@:d$:** in the text prompt will accomplish this. HandyTerm sees CTRL-I's as an automatic download initiator and the d$ is the variable that holds the current download name. There must be no other characters after the **@:d$:** except a [RETURN] (actually an [F7] character) if this modification is done.

We strongly advise against this Prompt #77 modification because it

disables HandyTerm users from also renaming the file before they download it since the HandyTerm will automatically accept d$ as the filename to download to. This makes it difficult for such people to download updates to programs that they already have on their directories.

The message header prompt is probably the most difficult prompt to actually edit. The person editing this prompt must always remember the specifications of the \V MCI Command and the @:h[COLOR]x: MCI commands. The reason that x does not equal anything in the message header is that when no variable is present to center, then the header MCI command AUTOMATICALLY uses **\v0** (SUBJECT OF THE MESSAGE) to center. Always use a real color for [COLOR] such as CTRL-1 for white (Do not use the MCI Color Commands). If a person no longer wants the subject of the message to be centered using the @:h[COLOR]: MCI command, the subject can be printed seperately using the **\v0** MCI Command.

APPENDIX D - HARDWARE CONSIDERATIONS

Perhaps one of the most common questions people have asked have pertained to how well the BBS will run on specific hardware and what sort of hardware upgrade one should get. Our reply is that we really can not fairly and objectively answer these questions. The author's BBS system runs off of a CMD 40 meg harddrive and a Turbomaster 4.09 Mhz CPU with JiffyDos. But this is not necessarily the best system setup for everyone.

Just because the author uses a CMD Harddrive does not mean that the CMD harddrive should be purchased over a Lt Kernal Harddrive. Both have their advantages and both have their disadvantages. We are not in a position to advocate either one to any owner. Likewise, just because we run a TurboMaster CPU does not mean a person should get that instead of a RAMLINK from CMD. A sysop should research the options himself and decide what is best for himself.

You should, however, shy away from purchasing the ICT type for the purpose of running on CBASE because of certain quirks in the DOS operating system which making it impossible to store messages and other sections of the board on it adequately. Also shy away from older Commodore Pet harddrives and the buscard interface. If a person has to use a Commodore Pet Harddrive, the Skyles Quicksilver interface is recommended and in the past their customer support has been above adequate.

PRINTERS

C_BASE supports the use of a printer as Device #4. Most commodore compatible printers will do nicely with the program.

It is not recommended to run the program with a printer on line for several reasons. The first is noise. Printers tend to be loud and this can get annoying since most person's BBSes are setup amidst public areas of a house or in a person's bedroom. Also, a printer is not recommended for use in conjunction with floppy drives since it increases the likelihood that the crucial timing between the computer and the disk drive serial bus will be messed up and thereby lock up the BBS while the BBS is running. If a printer online is necessary, then use it while running SFDs or Lt Kernals or other devices that do not require use of the serial bus.

1541

There has never been a problem running C_BASE on a 1541 except in regards to space. As a sysop running on a 1541, please keep note that when creating the system a person needs to be conscious of the fact that they

are using a space limited drive. 1541s have a maximum of 644 blocks free and only 144 file names. Do not go overboard and create a message base with thousands of messages on it.

Another helpful suggestion is to not validate your disk. Validating the a 1541 disk can easily result in a relative file which has one or more blocks mistakenly not allocating. This can cause some of the files to bleed into the userlog. The "safest" way to do the equivalent of a validation is to simply FILE copy (not full disk) each file over to another freshly formatted disk and use that backup as the new system disk.

1571

C_BASE supports the use of a 1571 in either double-sided mode or in 1541 emulation mode. If a sysop wishes to use the 1571 in double sided mode, then add the dos command "u0>m1" as the DOS Init command for the section of the BBS that uses the 1571 for its files in the system create.

It is recommended that double sided mode not be used with the bbs in favor of the double 1541 emulation mode. A 1571 in 1571 mode has 1328 blocks free but still has a maximum of 144 file names. A 1571 partitioned as two 1541s, one reading the upperside of a disk and the other reading the bottom side of the disk has 644 blocks free on each side (total of 1328) and 144 file names on each side (total of 288 file names). Clearly, utilizing the 1571 as two 1541s is more advantageous.

The DOS Init Command for using the top side of the 1541 disk is "u0>h0". This is the mode that the 1571 exists in when it is in 1541 mode normally. The DOS init command for using the bottom side of the 1571 is "u0>h1". To initialize a disk for using these two partitions use the following sequence of commands.

open15,de,15,"u0>m0" -- Makes sure that the 1571 is in 1541 mode.

print#15,"n0:top side,ts" -- Formats the top (normal) side of the disk.

print#15,"u0>h1" -- switches to using the bottom (opposite) side of the disk.

print#15,"n0:bottom side,bs" -- formats bottom side.

close15

It is important that the above steps are used to format the disk. The reason why is that a person can not simply format the bottom side by flipping the disk over because the disk will be formatted in the wrong direction. The disk is spinning in one direction and is formatted for use in that direction. If a person formats the bottom side in the same way they format the top side, then the 1571 will not be able to read the bottom side because it will have been formatted in the "wrong" direction.

The last advantage of formatting the disk in 1541 mode for both sides is that the 1541 fastloader built into C_BASE may be used with the disk drive now.

1581 & 1581 PARTITIONING

Using 1581 partitioning in C_BASE is not that difficult. The first step is to create the partitions. To do this either consult the 1581 Dos manual or use one of the 1581 test demo disk utilities that produce partitions. Also, most newer disk can create partitions as well. When

creating the partitions, only create them one level deep. Do not utilize partitions within partitions.

After that, in order to access each individial partition the sysop need only enter "/0:PARTITION NAME" as the DOS INIT command for subs, email, modules, libraries, or upload download sections depending on what the sysop wishes to use the partition for. The "/0:PARTITION NAME" is the format of the dos command as given by the 1581 Dos Manual. However, it has been our experience that in most cases "/PARTITION NAME" without the "0:" works just as well or even better in some cases than the DOS command.

If the sysop is using 1581 partitions it would be a good idea to make the system BBS LU 0: DOS INIT command, "i:" (initialize). This will take the BBS back to the root directory when the BBS accesses a main system file.

Older 1581s share a bug in their hardware having to do with the controller chip being out of date and a resistor missing within the 1581. The sysop may be able to set up the BBS correctly on a 1581 that has an older ROM chip and missing resistor, and may even be able to run the BBS for a long time without difficulty. However, these errors will surely cause problems in the long run and therefore should be corrected as soon as possible. If unsure whether the proper controller (WD1772 is the update) or resistor (J1 should have a 47ohm resistor closed circuit) is in the drive then run a 1581 diagnostic program on the drive. This program is public domain and it comes with the C_BASE Owner disks.

A second problem that 1581s share is an "append" bug. For most of the owners this firmware bug in 1581 DOS is never encountered, but invariably some people do have to deal with this problem. Basically, when there are a lot of files that get appended to on a 1581, there is a chance that sequential files will bleed into each other.

If this problem is encountered, there are two things that to do. The first is that messages and mail should go into their own 1581 partitions. This will seperate the files that get appended the most from each other (Callers file, Messages, and Mail). This will reduce the occurance of this bug in the DOS by a lot. The second thing yo do is configure the BBS to validate the disk after every call. Do this by answering yes to the question in the boot file miscellaneous configuration section concerning the validation of a disk after a user logs off. This second option is only recommended in severe cases since it will slow down the BBS considerably to have to go through a long validate procedure on the disk everytime someone logs off. The third and best option is to use a 1541 or other type of drive other than a 1581 as the system drive for storing the callers and namelog file.

If the system has the old rom chip (W1770) instead of the new one or the resistor is open when it should be closed, then this may actually serve to aggravate the bleeding problem described above. The other problem that these older versions of the 1581 cause is occasional device not present errors and many drive "lockups". I do not guarantee that C_BASE will run at all smoothly on 1581s with these faulty rom errors.

As far as fastloaders are concerned, most fastloaders are not recommended including the one that comes with the BBS. If it works, that is great. Otherwise do not use the fastload. Over the past years with C_BASE I have gotten mixed reactions about fastload cartridges being used in conjunction with 1581s and C_BASE. Action Replay seems to work most of the time, but there are some other owners who claim it is impossible to run their BBS off of an action replay. SuperSnapShot v5.0 seems to work very nicely with the BBS with few if any complaints from owners. To reiterate

however, no fastloader of any kind including my own are recommended for use with the BBS.

The only exception to this fastload rule is JiffyDos (tm of Creative Micro Designs). In fact, if a person runs their BBS with 1581 drives and other "soft" floppy drives, it is even recommended that they use this product since it makes it easier to back up the system disk with a built in fast relative file copier and makes the entire BBS run faster.

RAM EXPANDERS (17XX SERIES)

Using C_BASE v3.0 with a ram expander is fairly easy. Instead of the normal boot, merely load the RAMBOOT. Subsequently, copy the selected files to the ram expander using the built in file copy utilities. The sysop can also choose which device number they want the ram expander to be; 7 is the recommended device number.

Only enter one number for the ram expander device number. Do not enter things like ",8" or ",7" or ",7,1"! Use just plain numbers such as 7 or 8 or 15. 7 is the recommended number because normally nothing but a Ram Expander would ever use device 7. Plus 7 is easier to type than a two digit number like 15.

After this is done, The ramboot will be able to load the regular boot and then run it.

When setting up modules on the ram expander, set the device # of loading as the Ram Expander and the Device # for files to another device other than the ram expander because files are erased from a ram expander whenever the computer's power is turned off. For more information on this, see the section of the documentation labeled C_BASE FILES. This will show which number is the DEVICE TO LOAD and which is the DEVICE TO USE FILES OFF OF.

As a note to programmers, C_BASE v3 uses page $C500 as the page with which to interface the REU. This means that when using an REU with C_BASE, the sysop will not be able to use any of the built in fastloaders since they reside in this area.

One final note about ram expanders concerns compatibility with other equipment. C64s are set up to handle Input/Output through two seperate pages of memory known as DE00 or DF00.

Ram Expanders are set up to acknowledge DF00. Most interfaces and cartridges work in conjunction with ram expanders, because they occupy DE00 instead of DF00. If the ram expander does not work in conjunction with another type of cartridge or interface, this is probably due to it also residing at DF00 as an interface page.

SWIFTLINK CARTRIDGE/2400/9600+ BAUD

C_BASE is one of the few BBSes that works with the SwiftLink cartridge. Basically, this cartridge is an RS232 interface that allows the hook up RS232 modems. However, this interface differs from the standard interfaces in that it plugs into the cartridge port.

In order for CBASE to recognize a modem hooked up through the cartridge port in this manner, it is necessary to rename the appropriate "swift 64.o" file on the disk to "swift64". For most BBS sysops the appropriate swift 64.o file will be called "swift 64/de00.o". People that have conflicts with the default internal settings of their swiftlink cartridge,

use "swift 64/df00.o" if the person has switched the swiftlink cartridge to use page DF00 as an interface or "swift 64/d700.o" if it is switched to use page D700 as the input/output interface to the modem. The BBS Boot will load and install the file "swift64" in the proper place in memory when it finds it on the disk now that the filename "swift64" is there.

This cartridge way of interfacing a modem has several advantages to other standard interfaces. The first is that 2400 baud is actually smoother and faster using this cartridge than using a normal interface. This is because the modem information is traveling by parallel rather than serially through the normal USER PORT. Basically this allows the computer to spend less time servicing the modem and more time attending to normal tasks like running the BBS.

The second advantage besides smoother 2400 is the ability to hook modems of up to 38,400 Baud to the commdore computer. C_BASE is programmed to handle a baud rate of 38,400 baud being entered into the BBS. We have not tested this, but the makers of SwiftLink claim that it can inherently handle this speed so I assume it does.

Please note that if the sysop hooks up a 9600 baud modem to C_BASE via this cartridge that they should set it up in the same way this manual describes how to set up a 2400 baud modem with the exception that the "x" register should be set (See a modem manual for reference) so that the Modem will give the "connect 9600" message if someone calls in at that baud rate.

If the sysop wishes to use a modem with a higher baud rate on SwiftLink like 14.4K or 38.4K then choose the 9600 baud option and alter line 7175 which currently reads

```
7175 IF PEEK (838) = 8 THEN BR% = 9600
```

To

```
7175 IF PEEK (838) = 8 THEN BR% = 14400
```

(Or use whatever number corresponds to the highest baud rate of the modem such as 38400 if the modem supports 38.4K Baud)

Then simply recompile the basic code that has been modified after it has been saved.

The next step if the sysop plans on using a modem that supports above 9600 baud is to add lines to detect the connect signals of higher baud rates.

Normally C_BASE detects the following messages:

```
For 300  Baud:  "connect"
For 1200 Baud:  "connect 1200"
For 2400 Baud:  "connect 2400"
For 9600 Baud:  "connect 9600"
```

In each case, the BBS detects one or two characters that are unique to that baud rate connect signal. For example, if the BBS sees a "12" then it will go 1200 Baud and not bother reading in the rest of the "00" of "1200".

Obviously if the sysop uses a modem higher than 9600 Baud they will need to add their own detections for higher baud rates. Here are suggested line numbers.

```
7422 IF (C$+B$)="38" THEN BR% = 38400:GOTO7579
7423 IF (C$+B$)="14" THEN BR% = 14400:GOTO7579
```

That would be a sample way to detect higher baud rates. For the example above, line number 7422 detects 38400 baud by checking for the "38" in "connect 38400". As a sidenote, check the modem manual to find out and verify what the connect message is for the modem at that speed so that the proper characters are indeed being detected.

If the sysop wishes to add more connect messages they may use lines 7425 to 7429 since they are unused currently. For example, in the unlikely event that someone would connect at 4800 baud a person could add

```
7425 IF (C$+B$)="48" THEN BR%=4800:GOTO7579
```

As before, after making these changes merely save the basic source code to a disk and recompile it.

2400 BAUD/HAYES COMPATIBLE MODEM SETUP

In most cases the factory settings on a 2400 baud modem are inadequate to run a BBS on. Thus, the sysop has to go through the following procedure before setting up the "configure" file with the boot program. C_BASE is one of the few BBSes for the c64 that has error-free 2400 baud routines where there is no conflict between incoming and outgoing signals with "ate1" mode and the phenomena known slang-wise as "AT Thrashing".

Load in a terminal program and type the following AT commands in.

```
AT&F
AT&D3 (This is the most important register to have set)
AT&C1
AT&S0
ATX1
AT&E0
AT&W
```

Included on the C_BASE disk is a program called "2400 baud setup" that sets up the modem if adding these commands by hand feels uncomfortable. I documented the commands above so that if the 2400 baud modem was nonstandard then the sysop could play around with the settings for each of the parameters that is set and look them up in their modem manual.

As a final note on setting up the 2400 baud modem, if the modem has error correcting protocols such as ARQ or MNP then add the command AT&A0&W to the list of AT commands given above.

If planning on programming any modem output machine language coding for C_BASE, note that the v2.0 NMI has been reprogrammed to operate faster than before. Part of this reprogramming involved doing away with an inefficient RS232 send buffer in favor of quicker zero-page accessing of characters. There is still a receive buffer of course. The receive buffer is located at $CA00 and is the traditional 255 bytes long.

XETEC LT. KERNAL HARDDRIVES

Using C_BASE v3.0 with a Lt Kernal harddrive involves only a few easy and quick steps. Merely choose in the BOOT file configuration the DEVICE # the Lt Kernal is. This will speed up the Xetec Lt Kernal operations since the BBS will bring into play special Xetec speed saving features when accessing the drive.

Also, turn NMI traps OFF. This is the ONLY way to run the BBS at 2400 and 1200 baud error free. To turn NMI traps off, type 'config' at the "ready." Commdore BASIC prompt to reconfigure the system.

If partitioning use is desired, the BBS has provisions in the SYSTEM CREATE FILE and the STATS FILE to handle using ALL different LU's (Lt Kernal Logical Units) for SUBS, EMAIL, Modules, Libraries, and all the UD sections.

Please remember not to confuse Lt Kernal LUs with the logical drive notation within the BBS as described under the section of Drive SetUp. In this documentation, if Lt Kernal logical unit is meant, then it will be specified as such.

During the system setup, for each major section of the board, the system create will ask for device #, drive #, and dos command. Enter the device # of the Lt Kernal for the Device #. For Drive #, enter the Lt Kernal LU partition. For example, if the sysop wishes to use Lt kernal LU #5, they would enter 5: for the drive #. Normally make the DOS Init Command something like "i:", but if it is desired to use USER partitions within LU partitions (See Lt Kernal Owners Manual for a complete description), then use "LDEV#LU#USR#" as the DOS Init Command where LDEV, LU, USR are definable parameters.

For example, if the sysop wanted to go to LU 2, USER 1, and the Lt Kernal was device number 8, the DOS INIT COMMAND WOULD BE

"L821"

Finally, C_BASE supports the autobooting function of the Lt Kernal. If the power goes out, C_BASE will automatically reload itself from the Lt Kernal. If no one enters the time for an extended period of time in the boot file, the boot will automatically load the stats file and take the time of when the last user logged off and enter that in as the current time and loads the BBS. To take advantage of this feature all the sysop has to do is rename their "boot" file on C_BASE to "autostart" on the Lt Kernal.

CMD HARDDRIVES

C_BASE handles the CMD harddrive a lot like it would handle the Lt Kernal partitions. However, there are some differences.

First, if the sysop is planning on using partitions on the CMD harddrive then he should use the "CPx" DOS command as well as the appropriate drive # for each section of the BBS during system creation. For example, if the subs reside on partition #200 of the CMD harddrive, then put "200:" for the drive # and "cp200" for the DOS command. The BBS is CASE sensitive. The cp DOS command must be in lowercase. In the previous example, "CP200" would be incorrect and "cp200" would be the right format.

Lastly, If a person wishes to take advantage of the CMD harddrive built in clock so that they do not have to enter their own time when running the BBS, they can do so by creating a file called "cmdclock" on the booting device. When the boot file automatically detects the filename "cmdclock" it knows that there is a CMD harddrive on line and can get the current time from there. The sysop may create the "cmdclock" file by saving a file called "cmdclock" from the (w)rite file mode in the remote maintenance command prompt of the BBS. The **first** line of the cmdclock file must be the device number of the CMD harddrive preceeded by a [SPACE] character. Otherwise one may type a series of commands to create the file

from BASIC.

```
open8,de,2,"cmdclock,s,w"

print#8,"[SPACE][DEVICE # OF CMD HARDDRIVE]"

close8
```

Where "de" is equal to the device that the CMD harddrive is on.

In general, it is easiest if the CMD harddrive is set up for native partitions for each section of the BBS. However, there may be times when the sysop wants to use SUB DIRECTORIES within a partition on the CMD harddrive.

If a section of the BBS resides in a subdirectories within a partition use the CPx where x is the partition number as the DOS command for that section as described above. However, for the Drive # definition of that section of the bbs, use the "//[subdir name]/:" instead of the "x:" where x is the partition number.

Example:

If a subdirectory called libraries resides on partition 200, the dos command would be "**cp200**" and the drive # would be "**//libraries/:**" instead of "**200:**" for the section of the BBS that was being defined for libraries.

Do NOT use the **CMD CD** dos command as the dos command for going into sub directories. This will make it harder for the BBS to maneuver back out of the subdirectory when it wants to change sections of the BBS. Essentially using the "//[name]/:" as the drive number allows the BBS to still reside in the ROOT of the partition while accessing files in the subdirectory.

It is advisable to keep **UD- x** files (The UD Listings) in the root of any partition on the CMD harddrive.

TURBOMASTER CPU 4.09 MHZ CARTRIDGE

No special provisions are necessary for running this cartridge. C_BASE is already compatible with it. There are however some precautions to take if a person is planning on doing modifications to the BBS. The 4.09 Mhz cartridge speeds up FOR NEXT loops by 4 times. This is great if the for-next loop is doing a calculation, but horrible if the person is using the for next loop as a means of introducing a delay into the program. The best advice is to simply use the variable "ti" to count a specified difference between the original ti and the new ti value. "ti" is updated once every 60th of a second. So if a person wants a delay loop of one second they need only program a loop that waits for ti to be equal to or greater than the original ti value when entering the loop plus 60.

C_BASE already has a routine that does delays of this sort also. The format for calling this delay loop is setting "b" equal to the number of jiffies (60ths of a second) to wait, and then gosub74. To wait one second the programmer would use "b=60:gosub74".

After running the BBS for many months with the Turbomaster cartridge we have noticed that the 4Mhz cartridge sometimes fails to find files in 4Mhz mode that it is able to find in 1mhz mode. This is a rare occurance, and the files that this problem arises with always fails to be found. One of these files that seems to fail to be found is "plums ml" when loading the compiler to compile the BBS. Therefore, it is advisable if loading the

compiler in 4Mhz mode to load "plums ml" seperately with ",device,1" after having loaded the compiler and then run the compiler.

If this problem occurs within the BBS modifications by not being able to see read a sequential file somewhere then this problem may be alleviated by placing a small delay loop. One to a couple jiffy's is normally enough to solve the problem. We have only had this pop up in one area of the BBS and that was when loading the "term" menu in the miniterminal built into the BBS. It was solved by simply adding an extra delay by having the BBS issue a command to the modem channel. We are not really truly sure what causes this to happen actually or what really does fix it.

If a person runs into the problem of their sprite cursor doubling its normal width, then that is due to the TurboMaster CPU being turned on improperly. When powering down the computer, be sure to wait at least two to five whole minutes before powering it up again or there will be residue traces of electricity floating in the memory chips. If a person waits a good amount of time before turning on the TurboMaster cartridge and computer then that problem should never arise. Even if it does arise, the doubled width of the cursor has never affected the system operation for us except with it being a cosmetically undesirable trait.

Despite these tiny quirks in running with 4 times the speed, we highly recommend the TurboMaster cartridge to anyone running the BBS with equipment that is compatible with it. The turbomaster speeds up all BBS operations and makes compiling the basic 4 times faster for programming. The pausing phenomena known as "garbage collection" becomes an almost nonexistant blip while running with this speed demon. Our own personal system has run on a CMD HardDrive, JiffyDos, and the 4Mhz cartridge for several months now, and we have experienced a lot of compliments on the 4Mhz speed and have had no real problems with it.

ICT HARDDRIVES

Using C_BASE v3.0 with an ICT harddrive is merely a matter of playing around with the DOS INIT commands (See SystemCreate and BOOT file sections of the documentation for more information on the DOS Init Commands).

The BBS sends out a DOS INIT command when going to the subs, e-mail, modules, libraries, main system area, and upload/download sections. Since a person controls which partition they are currently on in an ICT Harddrive using DOS Commands, all there is to do is figure out the appropriate DOS Commands for each section.

Just figure out how the partitions will be chained on the ICT HardDrive per section. Do not make the chains too large or it will tend to slow the BBS down because the ICT drive become slower, the larger the partitions get. Make the chains too small though, and the BBS will not be utilizing the hard drive to the best of its abilities. With a little experimentation a happy medium should be found fairly quickly.

Whenever a person accesses a particular portion of the ICT harddrive, they will probably need to issue the proper DOS command that will go to the proper chain of partitioning for editing files on that partition. It would probably be a decent idea to keep a paper handy around the BBS displaying what DOS commands are issued for each section of the program including each ud section. Keeping track of the dos commands used to configure the system helps in case there is a system failure and a person needs to recover data from the chained partitions.

To partition an ICT HardDrive a person should understand how the drive is

set up in the first place.

The ICT Harddrive is really a 1571 with a harddrive which already comes partitioned as the equivalent of 120 664 block 1541 type drives. In order to connect these 120 1541's together, a person needs to "chain" them. The DOS command for accessing a chain is

'hm4 x y'. Where x is the starting drive and y is the ending drive in the chain.

So if a person did a DOS command such as

open15,8,15:print#15,"hm4 1 2" a person would be accessing the hard drive from partitions 1 through 2 (664*2 blocks = 1378 blocks free).

To signal the harddrive to access the built in 1571 instead of the harddisk itself, simply use the DOS command "h0".

The specifics of the ICT HardDrive should be in the ICT Manual. The above is just a brief overview in case it is not understood how to partition the ICT before reading the manual.

It is highly suggested that a person not run the system partition (BBS Logical Unit 0:) off of the hard drive itself because of quirks in the file handling capacity of the hard drive.

Also, do not run the message bases off of a chained partition on the harddrive. Doing this will result in not being able to use the Kx Kill message sysop command.

And as a final note in system setup, when configuring the system from the BOOT file and the system asks for what BBS LU to store the UD- files (Directory Listings) on, enter 0: and as suggested before, do not run the BBS LU 0: off of the hard drive itself.

USING JIFFYDOS

C_Base v3.0 has no problems running on JiffyDos. However, if RUNSTOP/RESTORE is hit for any reason and a person wants to re-enable the JiffyDos extended BASIC commands simply type "sys58451" at the ready prompt and press [RETURN].

SFD 1001, D9060 PET HARDDRIVES, PARALLEL INTERFACES

SFD 1001's, 2112 dual drives, and commodore 5/10 megabyte hard drives are all supported with C_BASE using an appropriate interface. The 2112 dual SFD drive can be accessed by setting the drive # during system setup as 0: for the first drive, and 1: for accessing the second drive. For some sections of the BBS, a person might enter the drive # as 0:, and for other sections the sysop could have the drive number as 1:.

If a person will be using the SWIFTLINK cartridge in conjunction with a QuickSilver interface they will need to change the internal settings in the SwiftLink so that it interfaces either at page $DF00 or page $D700. The Quicksilver occupying the $DE00 page and the SwiftLink in its default $DE00 setting conflict with each other.

If a person has an SFD but do not have an interface, the preferred interface to use with C_BASE is the QUICKSILVER interface. BUSCARDs interfaces are not recommended for use with C_BASE at all and we do not overtly support the BUSCARD type interface.

D9060 AND OTHER COMMODORE PET HARDDRIVES

The same notes that are listed for SFDs and their interfaces apply here. However, there is one very important change a person needs to make if they are using an old Commodore parallel harddrive.

Only boot the BBS using "9060boot". This program basically does a "poke882,8" and then boots up the normal boot for the BBS. If the Boot does not say something similar to "D9060 Provision Activated" then under no circumstances allow the text prompts to install beyond prompt 100. If a person does allow this to happen they alone will be responsible for the harddrive no longer working in a normal manner!

Also, when activating the D9060 provision, a person will not be able to use any of the fastloaders. This provision stops the text prompts from installing past $DC00 memory location and it overflows it into the fastload space. With other types of disk drives and harddrives, the program only stops the overflow at location $DE00.

OTHER HARDWARE

If a person ever runs across a piece of hardware that the BBS cannot support, and they feel it would benefit the users of C_BASE to support it, please feel free to send us the specifications and a xerox of a manual to that device to me so that I may incorporate its operation into future versions of the BBS.

APPENDIX E - MODULES SETUP

Empire -- To reset this module, erase the "{left arrow}empire" file and then run the module from the BBS. The game will detect that it is missing and prompt any person who has remote access if they wish to reset the game or not. The "{left arrow}empire" file contains the stats of the current players in empire.

Murder Motel -- To reset Murder Motel load and run the "mm.create" file from BASIC. This will set up the players logs and map.

Slots -- This module requires the filename "SLOTS.PIC" to be on the modules directory so that it may read this graphics screen in to play the game.

ModsPak -- This module automatically sets itself up for the most part. If a person wants to reset the module or do any maintenance to the trivia, polls, voting booth, wall, or jail, then they may type R if they have remote access at the main modspak prompt. The default password that allows a person to change the parameters in ModsPak is "23neurotic". This may be changed by editing the "mods info" file. This seq file contains the information as to which sections of the module are open and the module maintenance password.

UserRank -- In order to set up the ranking of the users in the userlog by blocks of credit and posts, a person merely has to load the module into the BBS and then it will allow remote access users to reset the UserRank mod. The reset procedure takes a very long time for this particular module. The UserRank files will appear on the modules disk itself even though the userlog is read in through BBS logical drive # 0:.

Little Shop Of Horrors -- To reset this game, the sysop needs to scratch the "LSstats" file off of the directory. When loaded into the BBS the game will realize the file does not exist and ask any person with remote

access if they wish to reset the game.

APPENDIX F - TROUBLESHOOTING AND BUG REPORTING

If a person follows the instructions in this documentation for setting up the BBS and it still does not seem to operate correctly, there are a few simple things to keep in mind when troubleshooting.

The first and most important way to troubleshoot the BBS is just plain common sense and logic. Scientific Method consists of formulating and brainstorming ideas of what might be causing the problem and testing the hypotheses out one by one. Scientific method does not include testing out several hypotheses at once unless necessary.

Very simply, please keep in mind that the possibility may exist that in the end a person may end up calling us for help. When a person does end up calling us they should have ready an idea of how they will describe the problem to us and they should have written down the things they have already done to try and fix the problem. The best way we can help anyone is if the person calling us can tell us in as specific detail as possible what is happening with the system.

The first hint in troubleshooting is to reconfigure the system so that the section of the BBS that the person is having trouble with is configured on another disk drive to see whether hardware incompatibility is the cause or whether the equipment is in need of repair such as realignment. The second hint is to go back and make sure that there are no disk errors on the disk. The third check a person always wants to do is to make sure the power supplies to the equipment are not overheating. Commodore is notorious for its poor power supplies and running a BBS for 24 hours a day can take its toll on a computer. These are just general hints, but in general use common sense in solving the problems.

The worst type of bad example scenario is "The program crashes." And then that is all that is said to us. Realistically, we have had people say that to us in the past. In fact, talking to the authors of other BBS programs, we find that this is not an isolated phenomena to C_BASE owners! It is our hope that this section will enlighten those that do not know how to report bugs in a specific manner other than "The program crashes." on how to report bugs and quirks in the program and save us both a lot of time and possibly long distance money.

COMMON QUESTIONS

Q: The BBS keeps locking up.

A: This can be caused by many things which are not related to the BBS at all. If the person just asked this question and did not give me information according to the instructions listed above, then a series of questions would be asked.

The first would be whether or not the person has a printer hooked up. When people run on serial disk drives such as 41s, 71s and 81s and the person also has a printer hooked through the back of one of them, this can cause lockup problems EVEN IF THE PRINTER IS NOT TURNED ON!

The second question would be "Where is the BBS Locking up". If the reply is "everywhere" then go on. If the reply is something like "In the subs" then the next consideration is "What sort of disk drive are the subs running on?". If it is an SFD, then the next consideration after that is "Does the BBS lock up in local mode or only when a person is calling and

reading messages?" If the answer is yes, then the possibility may exist that it is caused by using a BUSCARD type interface." In the v2.0 of CBASE the buscard interface was not supported. A patch in v3.0 has been made to support the BUSCARD but it still may not work 100% well and we recommend the use of a Quicksilver interface from Skyles Electric Works instead.

Lastly, a problem like this is encountered and there are many serial devices hooked up, copy all your main system files to only 1 or 2 disk drives, preferrably just 1, and see if it still locks up on running just one serial device. Then reconfigure the system over a matter of days adding one serial device at time until the system starts up again and through experimentation the configuration of drives that works best may be found.

And finally, The Transactor Magazine Volume 5 Issue 2, lists a series of bugs in the Kernal ROMs v2.0 and below for the 64 computer. One of these bugs is a bug in serial bus timing. The magazine article specifically states that when too many peripheral devices were connected together on the serial port the computer would misbehave (or lockup). KERNAL 3 and above fixes this problem. To test for which version of Kernal is in the 64, type print peek(65408) and [RETURN] at a BASIC READY prompt.

Q: In the middle of chat mode the modem lights start going haywire.

A: Transactor Magazine Volume 5 Issue 2 addresses another bug the KERNALs before v3 have. If a person deletes the last character in the last line of the screen and that line had previously scrolled up twice instead of once, the CIA timers that lie above the Color table become "disturbed". As an extension of what the Transactor article said, the CIA basically controls Modem driving routines. If the CIA becomes unfriendly, then so does the modem. Once again, this is fixed in KERNAL v3 and above for the c64.

Q: The BBS is stuck at the time that was entered in the Boot file.

A: This can be caused by several things that are all ultimately computer related. C_BASE BBS does not keep time by itself; it merely uses the real time clock that is built into the C64. If this clock fails, it is a symptom of something wrong inside the computer. Most people who have called up with this complaint have run their BBSes on a 128D series of computer. With this series of computer there is a fuse that is prone to blowing in the 128D. When this fuse is blown, all the operations of the computer are fine except this clock! Thus, in this case, the solution is to merely replace the fuse. If this problem occurs on a C64 or a regular 128 then the problem is a little bit more complicated. In some cases, using an interface in the cartridge port may cause this problem as well.

Some people have suggested that for people with this problem the real time clock be replaced with the ti$ clock. This should not be done. The reason the BBS uses the real time clock is that it is updated by the computer based on the 60Hz AC coming into the computer, and so it never loses time during operation. However, TI$ is updated differently and loses time constantly, especially during heavy serial drive access. Also, ti$ is used by the BBS already to keep track of a user's time limit to find out how many minutes the user has been on the BBS. And finally, if your real time clock in the computer is not working, this is a symptom of a larger problem and it should be checked out by an authorized commodore technician as soon as possible.

Q: There is a "file data error" when (A) the system is booted up or (B)

The BBS reads the stats file or (C) When a user logs on.

A: A file data error means that the BBS has tried to read a numerical input from disk but has received a normal text string instead.

(A) Erase the file on the boot disk called "configure". If the BBS has received a file data error here, it is because the "configure" file is not setup properly. This may be due to a disk corruption or it could be due to obtaining a system update where things have been added to the "configure" file. In both cases, the system will need to be configured over again in the BOOT file.

(B) This is caused by an incorrectly formatted "stats" file. Read the documentation of the stats file description to find out how it should be structured. Usually this is caused by a careless remote or sysop editing the stats file incorrectly such as adding a sub but not updating the number in the stats file that stands for number of subs. Always remember that the BBS keeps a running backup of the stats file called "backup" so if the old one is lost, it can be scratched and the "backup" file can be renamed to "stats". The best rule of thumb though is to always make a backup of the stats file before trying to do anything with it!

(C) This is caused when writing a bul x file from any place but the W Command in the remote maintenance mode. It can also be caused by someone having written an omni message file from remote mode instead of typing omn at the main prompt to write it.

Q: After the application module loads, the "app" file is not found and produces G's for the questions. After pressing [ENTER] a few times, the "app" file is found and everything seems to work ok again. At other times, when the application module is loaded, the BBS crashes with a DEVICE NOT PRESENT error but not all the time. Why?

A: In ALL these cases it seems that people are using either a large amount of serial drives, an old 64, and/or a 1581 (or 1571) with old ROMS in it.

The first time we encountered the problem was immediately after my very first v3.0 beta tester put up 3.0 last September -- Ken AKA Lone Wolf as listed in the documentation under support lines.

To the best of our knowledge, at the time he was running with 2 1581s, both with old ROMs and sometimes an additional 1541 or 1571 on line.

We were able to finally get around the problem by rehooking up and checking all the connections of the serial drives.

As an example, a sysop might have a device 8 1581 hooked into the computer first and then piggybacked a device 9 1541 and then after that a device 10 1581. If that sysop is having problems, they might try making both 1581s physically ahead of the 1541 in the serial chain or play around with different permutations.

Serial chain is meant to refer to the order in which the serial drives (1541s, 71s, 81s, etc) are hooked up in order to the computer.

The problem is not actually caused by CBASE; it is really a problem inherent in the fact that older rom drives and older ROM computers put out by Commodore have timing difficulties.

As background, serial devices like 1541s and 1581s are very very TIMING sensitive. The serial cables send information in the form of a stream of 8 bits at a time instead of the whole byte at once. Therefore, to keep things straight, the computer must be able to receive the bits one at a time and not get them out of sync. The way Commodore chose to sync the bits is using microsecond timers known as the CIA Clock.

While this had advantages a long time ago, it meant that if ANY ONE device in the serial chain of many serial drives has TIMING that is off or incompatible by even one microsecond compared to the other serial drives on the chain, then the other serial drives would be messed up in the manner that the G's appearing and DEVICE NOT PRESENT errors occur.

If the Commodore comptuer did not recognize incoming sync bits properly it might think a file was not found or is closed when it is actually open and ready to go.

If the problem is worse than just one bit being slightly off in timing, like several bits, then the Commodore computer will not even recognize the drive as being existant because of the "noise" on the serial cable. Thus, all of a sudden the device will appear to "go off line" until it is turned on again to resync its own timer in the serial chain for the computer to recognize it again.

Serial problems can also occur due to electrical noise on the line either with a power drain on the serial cable due to a fault in one of the drives or on the computer itself.

Kenny was able to solve the problem as our primary beta tester in less than a week and most of that time was figuring out what caused the problem in the first place. Since that time, Lone Wolf was able to run with even OLD ROM 81s and CBAE v3.0 for well over THREE MONTHS before he got his CMD Harddrive in January and the problem never occured again.

The reason the problem is more prone to exist on the older C64s is that the older 64s kernal rom chip had a tendancy to lock up in timing problems. The newer C64s have a greater tolerance for timing errors due to several serial devices chained together -- Commodore did not anticipate when they first released the 64 that 1541s would be a big item. In fact, for a full year after the author bought his 64 there was virtually no software available for the 64 on disk; it was all tape drive geared.

The best suggestion is to fool around with what things are hooked up where in the chain, and if a sysop is really pulling his or her hair out (never a pretty sight!), configure the system on just having ONE DISK DRIVE hooked up and turn on for the time being. Only then slowly experiment with successful configurations with TWO Devices, and then add the third device and so on. That way, the sysop can scientifically build their system up and get a better idea of which drive(s) may be the culprit.

The first type of serial chaining to try would be to hook up 1541's as first to be hooked to the computer, then 1581's and lastly 1571s. This configuration seems to work with the least problems among CBASE Owners.

Q: Sometimes my BBS gets a "**string too long**" error? What would cause this?

A: Usually this is caused by a corrupt file. It should be easy to tell which file got corrupted from where the error occured. When troubleshooting this type of error, always leave the mode on for the BBS

stating what file it is opening when it is opening it.

Case Example:

"0:namelog,s,r" has just been opened and then there is a "string too long" error.

The reason for this is obviously that somehow the "namelog" file was corrupted and requires regenerating.

Q: If you say you have an update to the system, which files should I generally get?

A: The files that should generally be redownloaded are

c/boot
c/bbs
ml 1.o
ml 2.o
ml 3.o

The other files will not generally change between updates. If there is a bug fix and the bug was blatantly in another file like the boot file, then only download the c/boot. If updates require other files, then there will be a note made to indicate that a given update requires more files.

As more questions and inquiries roll in, there will be sequential "read me" files made for downloading that answer these in more detail.

REPORTING BUGS

When a person wants to report a bug, please call or write us as soon as possible. Leave us the following: The entire system configuration including what device #s and what type of drives have each section of the BBS, and a detailed description of the problem. If applicable, also leave a line number that the error is occuring in. If possible please always include instructions on **how to duplicate the bug.** If we can not duplicate the bug that a person claims exists then it may be harder for us to figure out. Remember, the key word in reporting a problem with the program is **"specific".**

If the problem can not be duplicated by our customer support, then that will make it that much more difficult for customer support to understand and resolve the problem.

If the bug is occuring in a file on the BBS, upload the file to us in buffer in mail or on our BBS with a ^ in front of it so that it is hidden from other people.

When leaving feedback or writing a letter, please try to seperate the steps and explanation of the problem into logical sections or paragraphs that are seperated by several spaced lines to make it easier for customer support to read.

The most highly preferred form of customer support is to leave us feedback. Unless the problem is a dire emergency try to contact us on our BBS before contacting us voice. The second most preferred form of customer support is to write us a letter via US Postal service and a self addressed, stamped envelope. This makes it possible for to send a whole disk to us with the configuration and program files on it so we can take a look at the configuration right away. For most people this is less

expensive than calling long distance and leaving feedback and uploading the equivalent configurations. If all else fails or the problem is an emergency then by all means do contact us voice at reasonable Eastern Standard times.

APPENDIX G - TRADEMARK ACKNOWLEDGEMENTS

Commodore 64, 64c, C-128, C-128-D, SX-64, 1541, 1541-II, 1571, 1581, SFD-1001, D9060, 1700, 1764, 1750, RAMDOS, and 1670 are trademarks or registered trademarks of Commodore Electronics Limited. CMD HD Series Hard Drives, JiffyDOS, RAMLink, and SwiftLink are trademarks of Creative Micro Designs, Inc. Lt. Kernal is a trademark of Xetec, Inc. The ICT Minichief Hard Drive is a trademark of ICT, Inc. MSD is a trademark of MSD, Inc. TurboMaster CPU is a trademark of Schnedler Systems, Inc. Quicksilver is a trademark of Skyles Electric Works, Inc.