Color 64 version 8.0 Manual
written by Anthony Tolle and Fred Ogle
Updated 01-Nov-2005 by Anthony Tolle

Foreward by Michael “Nuke” Newkirk
Sysop of Itchy Butt BBS in King George VA (Color 64 v8.1)
Dated March 30th 2024

Credits:

Color 64 v7.37 was written by Greg Pfountz
The Network 64 v1.26 modification for Color 64 was written by Adam Fanello

Reformatted by Mike Newkirk Mar-2024
Sysop: **Itchy Butt BBS itchybutt.org:6502**
Thanks to my wife for her patience with me as I worked diligently on this document of which she has no understanding as to why!

Contents

Tables:

Foreward

I grew up in the 1970's and 80's where I got to see the evolution of telecommunications come to the homes of America and beyond, giving us computer nerds a whole new world that very little people understood; well, my parents and sister sure didn't, none of their friends, nor most kids in my school. But for a select few of us, we got it, and to us it was a shot of freedom! No longer were we bounded by computer meet-ups at the pizza parlors in our hometowns to find the latest software that is out for our computers or to meet other like-minded nerds, although those still did happen! The advent of the Bulletin Board System (BBS) gave us the opportunity to promote the idea of sharing software, ideas, thoughts and maybe some competition from the comfort of our home, perhaps while cranking a Kansas or .38 Special album in the background. And if you had a few extra dollars for long-distance, your resources would double and triple.

Today, I sit in front of my home computer typing this, and my current modem speed test I just ran from it indicates the modem speed is "2,814 MBPS". This means every second on average, 2,814,000,000 binary bits (1s and 0s) come through my modem. In the beginning, this was a tad bit slower... OK; painfully slow; 300 bits of 8-bit computer words per second (300 baud) was the "crawl" of the "crawl/walk/run" phase of our telecommunication abilities. On the first BBS I worked on, one of our programmers had learned how to hack the Vic Modem to crank up to a blazing 345 baud, which we proudly advertised on our BBS. Trying to download a program before one's sibling picked up the other phone in the house though was a real byte in the rear. 1200 Baud was a dream for a while until it happened, and if you had the money to spend, you were a God. Then came 2400.... The glory days of Commodore 64 BBSing, where one could download a disk easily before their time is up!

*The funny thing about all of this and maybe the "ha!" moment for all us nerds and geeks – who were all misunderstood and ridiculed in our local schools – is that while it was so foreign to others and considered "GEEK", you look around today and nearly *every single person* over the age of seven is using what was once GEEK. Every mobile phone, every laptop connecting to the internet, every TV using streaming services... We, the geeks, were the pioneers; and thank God to the programmers who were smart enough to produce these products to take advantage of the telecommunications capability. Those that laughed at us - that we would run a BBS back then or get on a BBS and play Empire instead of going out and partying in our Pintos - are using Facebook or Twitter today and I bet ten floppy disks that a lot of them probably dabbled in Angry Birds or Farmville.*

I started my BBS experience around 1984 on "Hal's BBS" which was a basic program (compiled with Blitz!) – and to think of how that BBS could be SO fun with only one or two 1541s as a resource is mind boggling to me. Here today with Image 3.0 or Color 64 with 3-5 disk drives (or emulated drives) working to support... that would have been this geek's dream come true back when I was 15. What am I saying? It's a dream come true today for this 56-year-old.

A few years later around 1988, I learned about Color 64 from a BBS running it in the San Diego area where I was stationed for the Navy. The System Operator (SYSOP) was "Color Fan" (Hi, if you're still out there!!). I would call in to his BBS each Saturday morning, invoke chat, and he and I would talk (type) about whatever while having our morning coffee. Eventually, I got a copy of the software from him and started my first Color 64 BBS. My military pay supported my geek appetite and allowed me to obtain the coveted 1581 Disk Drive as well as the 1750 RAM Expansion unit and a 1541 disk as the upload/download drive. For those that don't understand what that really means: I devoted nearly two weeks of work pay just to have that one 1581 disk

drive that would run my BBS, then another two weeks of pay went towards the RAM expander that would support. ALL IN THE NAME of BBS'ing!

I tip my hat to my fellow sysops and the amazing programmers of then (The Pioneers!) that not only made all this a possibility and to those that are out there today to ensure our 1980 machines can still "call" each other... even if the landlines that were once the most crucial component of the architecture are near extinct. As us dinosaurs start to transition to whatever is next beyond this earthly plain, the exposure of these systems to the younger generations are crucial to their survivability and is a great stepping-stone for them to understand and appreciate the systems that drive their lives today.

This document is a capture of the last Color 64 v8.0 txt that is widely available on the internet. I wanted to reformat it and make it an easier table-driven reference as much as possible for my own personal use, and the idea grew to make this something more with distribution to the interested communities in mind. I tried to keep to the author's original personality and content, but invoked many grammatic changes and some technical that I felt would better serve the new generation of SYSOPs coming in.

At the time of this writing, Color 64 8.1 has long hit the streets. There was a promised update of the manual to reflect changes incorporated for 8.1, but it has yet to materialize. That said, this manual does remain indispensable for the Color 64 v8.1 sysop but note that updates to ML and any of the programs listed as "8100" version are not captured in this material.

Peace & Love. =--Nuke--=
Michael Newkirk – 2024

Chapter 1, Introduction

This BBS program is comprised of two parts. The first part is the set of BASIC programs, called overlays, which are responsible for the operation of the BBS program. The second part is a section of machine language code, which extends the functions of BASIC to allow it to operate with a modem and to allow BASIC to run as fast as possible. The main purpose of writing the overlays in BASIC is for ease of modification by you the SYSOP. If you can do a little BASIC programming then you should be able to make almost any changes to the BBS that you may want, but if you ever need or want any help in making a modification feel free to ask us about it. Also, even though Color 64 was written to be easily modified, the stock system already includes enough features that you don't need to make a lot of modifications to run an excellent BBS system.

The Color 64 BBS program is a very intricate program. Before setting out to install your system, you should read through all the installation documentation carefully. Also, do not start the installation late at night; make sure that you have plenty of time and no distractions when you start the installation process. Finally, after reading the installation instructions, do some planning on paper so you will have almost everything decided BEFORE you begin installation.

Here are some of the features of Color 64. It supports:

- Both public and private messages with ANSI, ASCII or CBM graphics
- Uploads and downloads using Xmodem or Punter protocol (including multi-receive mode) from any number of drives
- Text files and on-line help files
- Variable baud rates (300 to 2400 without SwiftLink and 300 to 38,400 with)
- Self-maintaining online caller log
- Real-time clock with date & time stamping on all messages
- Daily time limits
- Variable access levels including full remote SYSOP capabilities
- Chat mode with word wrap
- All feedback stored on disk in your private mailbox
- Autoreply to public and private messages
- Automatic deletion of old unread mail
- Automatic purging of members who have not called for a certain number of days
- Hidden uploads & downloads
- SYSOP terminal mode with upload and download capabilities (for calling other boards)
- Local mode (for calling your own board)
- A download directory system that keeps all files dated and in chronological order
- Use of the Creative Micro Designs SwiftLink RS-232 cartridge
- Use of the Schnedler Systems 4.09 MHz Turbo Master CPU

The program has been written to be as crash resistant as possible, but if a caller does happen to find a way to crash the system causing a BASIC error, then a small machine code routine will automatically rerun the BBS program.

1.1 Hardware Support

This BBS program will operate with almost any combination of disk drives. Of course, some disk drives work better than others. Currently, with the bugs in the 1581 disk drives, it is recommended that this drive be used only as an upload, download, or text files drive--unless you have access to a fixed ROM (JiffyDos ROMS usually have all the latest fixes). Color 64 will support most any type of Commodore printer (or other printer with a Commodore interface), but a printer is not required.

- Commodore Ram Expansion Units
  The BBS can also be used with a Commodore 17XX series Ram Expansion Unit (REU) from Commodore. However, the unit must have at least 256K of storage to be usable with Color 64. Color 64 uses a Ram Expander by installing RAMDOS, a standard REU software interface that simulates a CBM disk drive in the REU's RAM. The version of RAMDOS included with Color 64 supports REU's expanded up to 2 megabytes of memory.

- CMD RamLink
  Color 64 supports the use of the RamLink device from Creative Micro Designs. RamLink may be used one or both of the following ways: If you also have a CMD HD you can take advantage of the parallel interface, or if you have RamCard and RAM installed then you can take advantage of RamLink's automatic ram-disk function.

- Skyles 1541 Flash!
  This program supports 1541 Flash! (on device number 8 only). This DOS speedup kit from Skyles Electric Works triples the speed of a 1541 making the BBS much more responsive. If you have 1541 Flash!, all you need to do is follow the instructions in the 1541 Flash! section for installing the code. The BBS program will then automatically detect if device 8 is being used and will switch to slow mode if the current device is not 8.

- Modems
  This BBS will work with any 100% Hayes-compatible modem, and most modems that support an AT command set like Hayes modems. Some modems can work with Color 64 only if special modifications are made to the system. One example is the Avatex 1200 Low-Cost modem (not the Avatex 1200 HC or the Avatex 2400). Instructions for the modifications will be provided as you follow the installation instructions.

  This BBS program will not work with a modem that doesn't support any kind of command set. This includes any Commodore 300 baud modems that require you to dial manually (picking up the receiver and dialing on the in-line phone).

The best type of modem to use is one that can be programmed not to auto-answer (ATS0=0) and when the phone rings which will accept the command ATA to answer the phone. For 1200 baud operation a Hayes Smartmodem 1200, Volksmodem 12, the NEW Commodore 1670 modem (the version that has 4 switches on the back of it) and most other Hayes-compatibles modems will operate in this mode. If you are using a Hayes Smartmodem 1200, set the internal DIP switches 3, 5, and 8 down, all others up. If you are using the NEW 1670, switch 3 up and all other switches down.

For 2400 baud operation, a Hayes Smartmodem 2400 or good compatible is required. Without SwiftLink the C64 must be programmed very carefully to be able to support 2400 baud, and the quality of modem and phone line are very important. Please refer to the section in "Specific Hardware Requirements" for instructions on setting up for 2400 baud use.

- CMD SwiftLink Interface
  For communication rates above 2400 BPS, the SwiftLink RS-232 interface produced by Creative Micro Designs, Inc. is required. This affordable piece of hardware plugs into the cartridge port of your Commodore computer. It allows communication rates up to 38,400 BPS and the information transfer is more reliable than otherwise possible. Also, SwiftLink will allow you to take advantage of the special data compression modes of many 2400 BPS modems that offer MNP (Microcom Networking Protocol). Color 64 should support any Hayes-compatible high-speed modem.

- Schnedler TurboMaster CPU
  For the fastest BBS operation possible, Color 64 supports use of the Schnedler Systems 4.09 MHz Turbo Master CPU. This nifty device plugs into your computer's cartridge port and increases your computer's speed by 400 percent. Although as of now the Turbo Master is no longer produced by Schnedler Systems, you may be able to find a used one somewhere. For the ultimate BBS speed, you can use the SwiftLink RS-232 cartridge along with the Turbo Master. To do this, you need the Master Adaptor that is a companion device to the Turbo Master. Configure the Master Adaptor to GEORAM mode and plug the SwiftLink into the extension port. We have not tested the possibility of changing the SwiftLink's I/O address, so if you have made a hardware modification, it might be possible that the SwiftLink will work with Turbo Master only if the I/O address is at the factory default of $DE00.

==========

1.2 The Program Overlays

To get the most out of the Commodore 64's limited amount of memory, the BBS program has been divided into several program overlays. All the overlay file names begin with "√bbs." and have a suffix in the form "80XX", where "80" means it's part of the version 8.0 system and "XX" is the current revision number of the individual overlay. The reason for this is to allow for upgrades and fixes to individual overlays without having to upgrade the entire BBS system. Most of the files should have "8000" (meaning no revisions) at the time of this writing. If I refer to an overlay without the suffix in

this documentation, you should still assume that the file name on disk still has the version number suffix. For example, if I refer to “√bbs.ov3", I really mean the "√bbs.ov3 8000" overlay.

The "√bbs.init" overlay is responsible for initial variable setup, caller logon, and password maintenance. "√bbs.msgs" is responsible for all public and private messages. "√bbs.xfer" is responsible for single file uploads uploads, single and multi-downloads, and directory maintenance. "√bbs.ovl” is responsible for chat mode, multi-uploads, help and text files. The "√bbs.ov2" overlay handles loading your online games and modules. "√bbs.ov3” is an empty skeleton file, allowing you to create a custom overlay. You do not need this last file because it can be completely switched out. Use of the module loader in "√bbs.ov2" is not required either, but it is recommended. Also included is the "√bbs.term" overlay, which is a full featured terminal program. The use of this program is optional, but there should be enough room on any system to store this handy program.

If you are going to use Network, then there are two other overlays that Color 64 will use. The first is the "√bbs.nw1" overlay, which contains the Network message editor, and handles calling and receiving calls. The other is the "√bbs.nw2" overlay, which handles the Network maintenance, and the distribution of messages. If you do not wish to run Network, then these files are not required.

1.2.1 Disk Speed Enhancers

The average size of these overlays is about 90 blocks each and would require a long time to load on an unmodified disk drive, so it is required that you use a speed enhancer. JiffyDos is highly recommended, although there are many other types of fastloader systems that are compatible with Color 64. You can also use the Xetec Lt. Kernal hard drive with its own Host Adapter (make sure you set the NMI trap off - 00 - in the Lt. Kernal setup). If any load time takes more than 15 seconds, you will probably get complaints from your callers.

Another solution to the slow disk access is to add a C1764, C1750, or compatible Ram Expansion Unit (REU) expanded to at least 256K. These units add 256K or 512K (or more if expanded) of ternal RAM to your 64 and will eliminate the need for program loading delays in the BBS. Color 64 will support up to 2 megabytes of external RAM. Separate boot programs are used that will install RAMDOS and then load a BASIC "script" which will copy all the necessary files onto your REU. Even if you plan to use an REU for your system, I suggest that you first install it without the REU and run it off a disk device. Then once you are comfortable with the operation of Color 64, you can easily switch to REU use.

As you may have noticed, each of the program overlay filenames begins with a “√“. To type this character on Commodore 64 & 128, you must press and hold either shift key, then press the @ key. This special character is used to hide all the system files from being seen or downloaded by your callers.

1.3 File Storage Drives

Now that you have all your hardware together, it is time to decide how Color 64 will be able to store all its files.

The BBS program divides up external storage into "drives" (short for "disk drive"). In these instructions, a drive is a generic term used to refer to any type of external storage. For example, a drive can be a 1541 floppy disk drive, a CMD HD 16-megabyte partition, a 2-megabyte REU, or any other separate and isolated area where files can be stored.

On the drives are stored "groups" of files. A files group is a collection of files that have a common function, and which share the same disk space. For example, the Public Messages group is the group of all the public messages posted by the users on a BBS system.

All the groups of files can be totally independent of each other (i.e. on separate drives), or you can have several groups stored on one drive. The BBS system will be able to keep track of all the files in a group no matter how you have them stored, if all the files in the group are in the same disk directory. It is recommended that you keep the groups separate if you can, to allow room for expansion and because the system will run faster. Also note that all file groups must always be accessible to Color 64, which means that you cannot not use a "disk-swapping" method of changing disks in a drive as the system is running.

If I refer to a "Program Files drive" this means the DRIVE that the Program Files are stored on. This same terminology applies to the other files groups, because it allows for the possibility that you are storing more than one group of files on a single drive. For example, the System Files drive could also be the Help Files drive, but only if both groups are stored on the same drive.

Here are all the file groups that Color 64 uses. Some groups have a "recommended minimum storage space" listed, which is just a recommendation on how many free blocks you should start out with for that group:

- Program Files
  This contains all of your main overlays, your "√bbs.parms" (parameters) file, plus any extra ML files required for use with your system. Usually this means all the files that are preceded with the "√bbs" prefix. It is also recommended, but not required, that you include the BBS Boot Files with your Program Files. If you are going to run a full system, the Program Files should have at least 1000 blocks of total space on it (a double sided 1571 disk would be the minimum floppy disk); Otherwise, you *may* be able to fit your overlays onto a 1541 floppy, but only if you do not use either the "√bbs.ov2" and "√bbs.ov3" overlays, or you do not use the Network overlays. It is required that the Program Files be stored only on drive 0 or drive 1 (LU 0 or LU 1 of a Lt. Kernal) of a disk drive. Also, the files should NOT be stored in a subdirectory (as on CMD drives).

- Boot Files

This group contains all the files necessary to start up the Color 64 program, along with all the utilities like SETUP, password tools, menu maker, etc. If you have enough room (e.g. a 1571 or 1581 partition on an HD), you should put these files with your Program Files. It is required that these files be accessed only on drive 0 or drive 1 (LU 0 or LU 1 of a Lt. Kernal) of a disk drive. Also, the files should NOT be stored in a subdirectory (as on CMD drives).

- System Files
  This group contains all the non-program files that are used by the main BBS system. This includes menu files, data files, relative files, etc. The drive that the System Files are stored on must be able to support storage of relative files (e.g. a ICT HD partition chain cannot hold relative files). Recommended minimum storage space: 300 blocks.

- Help Files
  This group contains all the files which a caller can read to get help on using the BBS system.

- Text Files
  Like Help Files, this group contains all the other files (such as Commodore Graphics files) that a caller can read for information or for recreation. If you wish, you can combine this section and the Help Files section to keep all your text files in one area, or you can even choose not to have any Text Files.

- Password File
  This group contains your password file, the file that contains the information for all your BBS system callers. When the file is first created, enough space is opened to allow for the maximum number of callers that you think you will ever have. You should always have enough space in the password file to allow for new users, otherwise you can get unpredictable results. About 1 block is allocated for each caller, so if you were to set the maximum to 300, the password file will be more than 300 blocks. Thus, you may wish to put this file on its own drive. The drive that you put the password file on MUST be able to store relative files. For example, a chain of ICT HD partitions cannot hold a relative file.

- Private Files
  This group contains all the private E-mail that callers send to each other. You can also add special modules to your system which will use this area for anything that is privately assigned to a specific caller. Recommended minimum storage space: 300 blocks.

- Public Messages
  This contains the public messages that are posted by callers. Public messages networked to your system are also stored here. Recommended minimum storage space: 500 blocks.

- Caller Log
  This contains the caller log, which is a text file containing the record of all activity on the system, including your own. The activity is stored in the form of the commands and text that is entered by a caller. You can decide on two ways to store the caller log. You can have the log

stored separately for each day of activity, or you can have just one file with the most recent activity printed on it. Either way, you will be able to set the maximum size of the file, along with the option of trimming the file when it is too large. This file must be stored on drive 0 of a disk device. This means drive 0 of a dual drive such as the MSD-SD2, or it means LU 0 of the Lt. Kernal HD. Devices such as the 1541 or CMD HD are unaffected because the 1541 is a single drive device and the CMD HD uses 0 to refer to the current selected partition.

- Network Files
  If you choose to run Network, this contains all the files that are pertinent to Networking. Recommended minimum storage space: 1000 blocks.

1.4 Different System Configurations

The smallest possible system would be two 1541 disk drives, with three disks: Program disk, Boot disk, and system disk (all other file groups). The first drive would be the Program Files, and the drive from which you boot the system. The second drive would have your system disk in it and would be used for all the other file groups (System Files, Public messages, etc.).

With larger amounts of space, there are a few guidelines to remember:

(1) You should always keep your Program Files on a separate drive, but perhaps you can put the password file with them. Also, it is customary to have the Program Files stored on device number 8, drive number 0.
(2) For a new system, the System Files will not take up very much room, so this can probably be combined with the Help Files, Text Files, or another small group.
(3) The Caller Log section can usually be combined with the System Files if you do not need a daily copy of activity (but remember that the Caller Log must be stored on drive 0 of a device).

Other than that, it is up to you how you allocate space on your system. If at first, you are not sure how to do it, then don't worry. The SETUP program will allow you to change the location of all these files, which allows you to balance out the file load on each drive. You will be able to see if a certain drive has too many files, so all you will need to do is change the location of the groups and move the files. At this point you should plan out on paper how you will store the files groups so you will be ready when you begin the installation process.

During the rest of the documentation, I will refer to swapping disks while booting certain programs. If your system does not require swapping disks (as on a Hard Drive system), then you must make sure that all the files that are required will be accessible when booting Color 64 programs. This can be accomplished by putting all the Program Files and Boot Programs in the same area, or by putting them in separate partitions on the same hard drive.

Chapter 2, Installation

The installation process can be divided into three parts: copying the necessary program files, running the SETUP program, and then installing the system files. A detailed explanation of the function of each of the options in SETUP will be provided, as well as a description of all the necessary system files.

You should never use your original Color 64 disks as anything but the source of all the stock BBS files that come with Color 64. It is advised that you make a backup of your Color 64 disks for your own use in the remote case that something happens to them during the installation process.

Now make sure that you have plenty of time and there are no distractions, and you can begin the installation process.

2.1 Copying the BBS Files

Once you have decided how to divide up your storage, it is time to copy all the files necessary to boot the BBS system. See the insert titled "Disk Contents" to find out which of your Color 64 system disks contains the appropriate files.

If you are using floppy drives, format 2 or 3 blank disks for use. If you are using a hard drive, set up blank partitions for your Program Files and/or Boot files.

1. Copying the Program Files
   The first thing you will need to do is the Program Files. For floppy disk users this will be known as your Program disk, but some of the included programs may refer to it as the Parms disk because the BBS parameters file is stored with your Program Files. The files listed in Table 1 are required to be in your Program Files. If there is not enough room for all of these files, then you can omit the "√bbs.term" program.

Table 1 - Color 64 Program FIles to Copy

√bbs.init
√bbs.msgs
√bbs.xfer
√bbs.ovl
√bbs.ov2
√bbs.term
√bbs.punt
√bbs.xmo/c
√bbs.ansi
√bbs.ascii

2. Program Files Needed for Network
   If you have enough space left on your Program Files drive and you wish to use Network, then include the files in Table 2. If these files are missing and you try to use Network, the BBS system will not work.

Table 2 - Program Files to Copy if planning to participate in Color 64 Network

√bbs.nw1
√bbs.nw2

3. Specific System Requirements
   At this point, you may also need to make some changes to your program overlays for your system to work correctly. You should consult the section on "Specific Hardware Requirements" if you are using a fastloader cartridge, an ICT hard drive, or an Avatex 1200 modem (not the Avatex 1200 HC). These three hardware devices require special code in the overlays to make the BBS system work correctly. The code was not built into the overlays for one or both of the following reasons: The code may be large and would be unnecessary on other systems, or the code may not be compatible with other types of hardware.

4. Copying the Boot Files
   Next you want to do your Boot Files, the files which not only allow you to start up the Color 64 system, but which also contain the SETUP program and other utility programs. For 1541 floppy disk users, this will be your Boot disk. For 1571 floppy drive or hard drive users, you have a couple of options: If you have at least 700 blocks free on your Program Files drive, then you can copy all the Boot Files onto it; Otherwise, you will need another blank drive for the boot files. The files listed in Table 3 are needed in the Boot Files:

Table 3- Core Boot Files to Copy

bootmaker
+bbs
√sys.loadml
√sys.mlinit
√sys.setup
√sys.edit
pswd tools
dir tools
menu maker
bbs convert

In addition, you need to copy the specific BBS ML file to your Boot Files, depending on your setup identified in Table 4:

Table 4 - Machine Language File Selection Based on Configuration

Swiftlink	TurboMaster CPU	ML File to Copy
No	No	√sys.mlnorm
Yes	No	√sys.mlswft
No	Yes	√sys.mltmno
Yes	Yes	√sys.mltmsw

If you do not have the correct ML file copied to your Boot Files, then your system will not work correctly. If you have enough room, you can copy all four of the ML files to the Boot Files, allowing you to easily change the type of system you are running in the future.

5. Boot Files Needed for Network

If you plan to run Network, then the following additional files listed in Table 5 as part of your Boot Files:

Table 5 - Network Boot Files to Copy

√sys.net
prscrn52750

6. Boot Files Needed for Ram Expander

If you plan to use a 17XX Ram Expansion Unit for faster loading of your program files, then you need the additional files listed in Table 6 as part of your Boot Files:

Table 6 - Additional Boot Files to Copy for REU Use

√sys.ramove
√sys.ramcpy
√sys.rdrein
√sys.smerge
√sys.ramdos

7. A Shortcut for Booting

If you want to be able to boot your system from the Program Files drive, then there are two methods.

a) If you have enough space on the Program Files drive, then you can put the Boot Files with the Program Files (see above on creating Boot disk); or

b) Copy just the files necessary to boot the BBS system, identified in Table 7, to your Program Files. This is only intended if you are not using the 17XX series REU.

Table 7 - Minimum Boot Files Required (Boot FIles / Program Files combined on disk)

bm small
+bbs
√sys.loadml
Applicable ML file (see Table 4)

2.2 Creating the Boot Programs

The next step to accomplish is to use the included "bootmaker" utility to create the boot programs that start the essential BBS operations. All the boot programs begin with a "+" (plus sign) symbol; the +bbs program you copied to the boot files is an example of this. Creating these boot programs is very important because you set some important parameters for the system, such as whether you are using the SwiftLink RS-232 cartridge.

The bootmaker utility creates your boot programs by loading in the +bbs boot program, modifying it, and then storing it back to disk along with the other boot programs. For this reason, it is necessary that you already have the +bbs program present in the Boot Files.

2.2.1 Running the Bootmaker Program

The first thing you will need to do is insert your boot disk, then LOAD and RUN the "bootmaker" program. The following questions will occur:

1. Device, Drive and Init command for boot programs:
   The first question you will be asked is the device, drive, and init command for the drive your boot programs are stored on.

   When using the bootmaker program, you will be asked several times for device, drive, and init commands for disk drives. In all the questions, these parameters are limited to specific ranges. The device number must be in the range of 8 to 30, the drive number must only be 0 or 1, and the init command can be a maximum of 16 characters. Read the section entitled "Drive Initialization Commands" for information on the proper init command for your disk drives. Note that the drive init command is always preceded by a quotation mark in the

bootmaker questions. This allows special characters like colons to be entered in the init command, but the quote mark will NOT be part of the drive init command. Do not delete the quote mark, or enter any other quote marks, or you may cause an error. One more note about the drive init command: You can separate multiple commands with the “!” (exclamation point) character, but the "i" command will not change the default drive number as described in the section "Drive Initialization Commands". If you are not familiar with this feature, just remember that the boot programs can only access drive 0 or 1 or a disk device. In the drive parameter questions you can simply press RETURN to accept the default. You can also edit the line by retyping it or by using the left and right cursor keys.

For this question, you should enter the information for the drive that you copied all the boot files to; it must have the "+bbs" program stored on it. Edit this if necessary, and press RETURN. The program will then notify you that it is loading the "+bbs" program into memory.

2. Running system with Ram Expander
   Next, you will be asked if you are going to run your system with a Commodore 17XX series Ram Expansion Unit. I suggest that when you are first installing your system that you configure it for use without a Ram Expander. Then once you are comfortable with running Color 64 you can re-run bootmaker and your SETUP program to reconfigure your system. Otherwise, answer "Y" to this question if you have an REU and you want to run your overlays on it.

3. Boot Drive
   Next, you will be asked to enter the device, drive, and init parameters for your Boot drive. This is the drive that you will be running your Boot Files on, with all the files you use to create your Boot disk. This can be the same as your Program Files drive if you decided to store your Boot Files on the same drive as the Program Files. The boot programs will use this information when loading your utilities like SETUP, so this must be the drive that you will always use to load your boot programs from.

4. Program Drive
   After this, you will be asked to enter the parameters for the Program drive. This is the drive that the Program Files (overlays) are stored on. If you have a separate Program disk, and you will need to swap the disks during the boot operation, then this should be the same as the Boot drive question. Also, if you are using an REU on your system, this is the drive that your Program Files will be loaded from, as well as the drive your BBS parameters will be read from.

5. External drive
   If you answered "Y" to the Ram Expander question, then at this point you will be asked to enter the parameters for the External storage device. In the case of the REU, this is the device, drive (always 0), and init of your REU. Of course, since the device number of the REU is user defined in the first place, this is your chance to set what device number that YOU want the REU to be. 15 is a common device number, but you can use any device

number that isn't already occupied. The boot programs will use these parameters when initializing the REU. ICT HD users note: The included ICT utility overlay assumes you have you REU set to device 15.

6. Need to swap disks
   The bootmaker program will recognize if you entered the same parameters for your Boot drive and Program drive and if so, it will ask if you need to swap disks when booting your BBS system. If you have separate Boot and Program disks and only one disk drive to boot from then you need to answer "Y" to this question. If you are using the "bm small" program, then this question will not be asked.

7. Using TurboMaster CPU
   Next, you will be asked if you are using the Schnedler Systems 4.09 Mhz TurboMaster CPU. If you have one and you wish to run your system with the Turbo Master device, answer "Y" to this question.

8. Using SwiftLink Cartridge
   The next question asks if you are using the Creative Micro Designs SwiftLink RS-232 cartridge. If you do have the cartridge and you want to use it with your system, answer "Y" to this question.

9. Using Lt.Kernal HD
   The next question depends on your answer to the SwiftLink question. If you answered "N" to the SwiftLink question, then you will be asked if you are using a Lt. Kernal Hard Drive on your system. You must answer "Y" to this question IF you are using Lt. Kernal. Otherwise, the system will not run properly.

10. SwiftLink address
    If you answer "Y" to the SwiftLink question, then you will be asked a technical question regarding the installation of your SwiftLink cartridge. The SwiftLink cartridge must use a certain portion of the computer's memory to allow programs to interface with the cartridge. The stock SwiftLink cartridge shipped by CMD is set to occupy the section of memory known as page $DE00. There is an optional hardware modification that users can perform to change this address to either $DF00 or $D700. If you have never modified your SwiftLink cartridge, then simply use $DE00. Otherwise, those who have modified their cartridge will know what the address is set to. If you are not sure then just use $DE00, and then if the BBS program is not sending or receiving from the modem you will know that this needs to be changed to one of the other two addresses (a little experimentation may be necessary).

After answering the last question, the program will then immediately start creating the boot programs on the disk with the "+bbs" program on it. Table 8 lists the files either replaced or created by the Bootmaker Utility.

Table 8 - Bootmaker Created / Replaced Files

+bbs
+reboot
+setup
+net setup
+editor
+shell
+ram.start (1)
+ram.restart (1)
+ram.reinit (1)
+ram.bbs (1)
+ram.reboot (1)
(1) If REU option is selcted

2.2.3 Using the "bm small" Program

If you are not using an REU and you have separate Program and Boot disks, you can also use the "bm small" program to create the necessary boot programs in your Program Files. This will allow you to boot your BBS program without having to swap the Boot and Program disks during the process. Refer to Table 7 under Section 2.1.4 for information on which other files you need in your Program Files to do this. You MUST still run bootmaker so you will be able to use your BBS SETUP program, but after that is done you can run "bm small".

Once you run the "bm small" program you should enter the drive parameters for the PROGRAM FILES at the very first question, because this is the disk you will be creating the extra boot programs on. Also, you will not be asked if you need to swap disks, because the necessary boot programs are going to be on the same disk as the Program Files. Once you run the program, the “+bbs” and “+reboot” files will be created in your Program Files.

2.3 The Boot Process

Now that you have created the boot programs, it would be a good time for me to explain their function. Don't use any of these programs yet because we will be running SETUP in just a bit; I just want to describe what each boot program does.

When you use any of the boot programs that start with the plus sign symbol ("+"), several things occur. The first thing that you will notice is that it appears that the computer does a reset (the screen clears, and the colors go back to default). The reset is done to make sure the system is ready to load the necessary programs. Next, another program will be loaded into memory by the boot program. The function of this next program varies depending on the individual boot program; it may load other programs, or it may perform a function of its own. The only important thing to

remember is that before you can use any function of the BBS system, one of the boot programs must be used at least once. The reason for this is that the boot programs themselves contain the essential parameters of your system (such as the device numbers of your Program Files and Boot Files).

Table 9 below describes what each of the boot programs does:

Table 9 - Boot Files Functional Description

Filename	Functional Description	Load Sequence
+bbs	Boots the main BBS system to the point of the date and time setting prompts.	1) √sys.loadml 2) √sys.mlinit 3) √)sys.mlnorm 4) √bbs.init
+reboot	Like +bbs above, except +reboot will cause the default settings to be accepted by the “date and time” and “regenerate message index” questions, as if you had hit RETURN at these prompts yourself.	
+setup	Boots the SETUP utility.	1) √sys.loadml 2) √sys.mlinit 3) √)sys.mlnorm 4) √sys.setup
+editor	Boots the stand-alone message editor.	1) √sys.loadml 2) √sys.mlinit 3) √)sys.mlnorm 4) √sys.edit

Filename	Functional Description	Load Sequence
+net setup	Boots the Network Setup Utility	1) √sys.loadml 2) √sys.mlinit 3) √)sys.mlnorm 4) √sys.net
+shell	Loads the ML shell (see section 2.4 below) into memory and returns to the BASIC "ready." prompt. This is used for the included stand-alone utilities if the ML is not already in memory. It first loads the √sys.loadml program, which in turn loads √sys.mlinit, and finally the specific system ML program.	1) √sys.loadml 2) √sys.mlinit 3) √)sys.mlnorm
+ram.start	Main BBS system boot program for REU users. This runs the BASIC script program necessary to copy your overlays to the REU and then will boot the system just like the +bbs program.	1) √sys.ramove 2) √sys.ramdos 3) √sys.smerge 4) √sys.ramcpy 5) +ram.bbs
+ram.restart	Like +reboot program, but for REU users as it copies the overlays before rebooting the system. It will also load +ram.reboot instead of +ram√bbs	
+ram.reinit	In case of a system reset, this program can be used to reinstall the RAMDOS program necessary to access the files stored on the REU	√sys.rdrein
+ram.bbs	Used only if your overlays have already been copied to the REU and RAMDOS is active. This behaves like the standard +bbs program but loads all files from the REU instead of the Boot and Program disks.	
+ram.reboot	Used only if your overlays have already been copied to the REU and RAMDOS is active. This behaves like the standard +reboot program but loads all files from the REU instead of the Boot and Program disks.	

2.4 The ML Shell

When using the +bbs, +reboot, +setup, +net setup, +editor, and +shell programs (and the +ram.bbs and +ram.reboot files for REU users), the "√sys.loadml" file is loaded as the first step in the boot process. This file then loads the necessary ML code into memory and installs the ML, at which point the Color 64 title screen is displayed. Once this ML "shell" is installed in memory, it will stay there until you either shut the computer off or run another non-Color 64 program that alters essential memory (the bootmaker programs do this). If you are using the "+shell" program, then the boot process ends once the ML has been installed in memory.

There are several support programs that come with Color 64 that need the ML installed to run properly. These are "pswd tools", "dir tools", "menu maker", and "bbs convert". You can start these

up after shutting down your BBS system because the ML will still be installed, or you can run the +shell boot program to install the ML.

Now that I have described the boot process for you, it is time to use the SETUP utility.

2.5 SETUP Parameters

One of the most important things you will need to do when setting up your BBS is to define a few SYSOP selectable parameters. To do this you must run the SETUP program included with Color 64. Now is a good time to make sure the drive that you assigned to the password file is online and has enough space on it. It is necessary to make sure that the disk in the drive or the partition on the hard drive is formatted and blank. The SETUP program will create a relative file on this drive large enough to hold the number of callers you designated.

If you have a Boot disk, then insert the disk into your disk drive now. Otherwise, just make sure you are accessing your Boot Files. Next, LOAD and RUN the program called "+setup". You may want to run SETUP the first time with Fastload disabled (You can leave JiffyDos enabled). On some systems, you will get a "NO CHANNEL" error when creating REL files (like the password file) while Fastload is enabled. If later you rerun SETUP to just change a parameter, it is ok to leave Fastload enabled. It seems to be safe to read and write REL records with Fastload enabled, just not when creating a new file. When you boot SETUP, you may be asked to insert your Program disk into the drive.

2.5.1 The "√bbs.parms" File

Your BBS parameters are stored in a file called "√bbs.parms" and will ALWAYS be stored on your Program Files drive. This is necessary because the parms must always be accessible to the program overlays, not just during the boot process, but also in case the system crashes. For REU users, your "√bbs.parms" file is NOT copied to RAM--it is always read in from the Program drive. It is possible, however, to keep your parms on a disk that is not the Program disk. You just need to make sure that the proper disk is in the drive when the BBS program gets to the point where it needs to be read in the parms. If the BBS program cannot find the parms file when it is first loaded, it will ask you to insert a Parms disk. Thus, if you want to use this method, then you must answer "Y" to the "swap disks" question in the "bm ram" program, then when SETUP asks for the Program disk, insert the disk which you want to store your "√bbs.parms" file on.

Insert your Program/Parms disk now and press RETURN if necessary. After some disk access, the program should go right to the Main Parameters (2.6) screen. SETUP will do this only if a parms file does not already exist in your Program Files. Since you are just starting your system, this is OK as the SETUP program will need you to answer a few questions about how your BBS is to run.

At most of the questions in setup, you will also see the current or default value displayed in square brackets ("[" and "]"). If you wish to use the current value, just press RETURN to accept it. Also, at most of the prompts you can type " **-** " (minus sign) and press RETURN to go back to the previous question. This can be helpful if you accidentally "miss" the question you wanted to change.

2.6 SETUP - Main Parameters

The Main Parameters section is where you get to define how some of the critical functions of your BBS will be performed. If you are not sure about how to answer some of the questions, it should be OK to accept the default value given to you. You should, however, read the description of each of the items so you can understand how the BBS will use the information.

1. Maximum lines per message
   Sets the maximum number of lines to limit each message. The default is 100 with an allowable range of 20 to 200. The smaller messages will take up less disk space, leaving more room for more messages or other files, while longer messages can save a lot of "continued" messages.

2. Maximum columns per line
   Sets the default word wrap length you would like when you use local-mode features like the text editor. You should set it to 38 for a 40-column formatted message, or 78 for an 80-column formatted message. The default is 38.

3. Maximum # of messages
   Sets the maximum number of public messages the system will keep online at any one time. The program will automatically remove the oldest public message when a new message is posted, and the total number of public messages online exceeds this variable. The default is 50 with an allowable range of 25 to 232. The Commodore 1541 and 1571 disk drives will allow up to 144 files to be stored on one diskette, while an SFD 1001 allows 224 files and the 9060 and 9090 hard drives can have an unlimited number of files in the directory. The number of files on Lt.Kernal HD systems and CMD HD Native mode partitions is virtually unlimited also. 232 is the maximum number of public messages that Color 64 can handle at any one time.

4. Maximum password number
   Sets the maximum number of passwords you want to set the system up for. It defaults to 100 with an allowable range of 25 to 9999. Please note that some floppy drives (like the 1541) cannot handle relative file sizes (the type of file used for the password file) as high as 9999 records. The maximum number of records for the 1541 is 720 records, so it should be safe to assume that any other disk drive can support at least 720 records. Check your disk drive reference manual to see what the limit is for your disk drive. If a new member applies for a password causing this value to be exceeded, then the BBS will send a message ("√membership full") telling the caller that the membership is temporarily full and to call

back later. You will want to make sure you give this value some real consideration as it will determine the size of the password file to initially create (about 1 block is used for each member). If you change this value when you re-run setup, you will be asked if you want to expand the file, although doing so could create a "fragmented" or "scattered" file on the disk, slowing the access time. If you are going to have enough disk space, then I recommend you set up the password file for 250 to 500 members (depending on the size of the city you are in). This will consume about 250 to 500 blocks of disk space, but in the long run the system will run faster. If you did create this file too small and later decide to increase its size and if you do find that the password file does get scattered, you could use “pswd tools" program to make a backup of your password file. Then on a new blank disk, rerun SETUP to make a new larger password file and use "pswd tools" to restore all your passwords into the new password file. Then use a file copier to copy all other files from the old disk onto the new one. This will make all files contiguous again, greatly speeding up disk access time. In the Disk Drive Assignments, you will be deciding which drive to put your password file on.

5. Minimum blocks-cycle msgs
   Sets the minimum number of blocks free allowed on the Public Messages drive before the messages start cycling. You do not want to let this disk completely fill up, especially if it will also be storing private messages on it. A value of 50 is defaulted with a range of 25 to 75 recommended. If blocks free on this disk ever fall below this variable, every time any new message is posted, the oldest public message will be deleted even if the maximum number of messages is not exceeded.

6. Minimum blocks-allow uploads
   Sets the minimum number of blocks free to allow uploads. A value of 75 is defaulted with a range of 10 to 75 recommended. If you are going to be storing private messages on the same disk, then you will not want to set this value less than 25. The BBS will not allow private messages to be sent if the directory falls below 25 blocks free. You will never want to set the value below 10. This is to allow space for the “√directory" file to be processed after an upload. Note that the free space message given at the end of the directory list and before an upload will automatically be decremented by this value (i.e. free upload space is the total free minus the minimum blocks free). Also, the caller log routines use this value in determining when the caller log should be trimmed if it is taking up too much space.

7. Maximum downloads per call
   Sets the maximum number of downloads you will allow per call. If you enter "99", then for all practical purposes there will not be a limit to the number of downloads per call. When a caller reaches their maximum downloads per call, they must logoff and logon again before they can download more. You can set the level at which callers are exempt from this function when you enter the level for "No DL file limit level" asked later in this section.

8. Download credits per call
   Sets the number of blocks of download credit you will give per block uploaded.

9. New mbr download credits
   Sets how many free credit blocks for each new caller. After these free credits are used up, your callers will have to upload to earn more download credits.

10. Credit system exempt level
    Sets the access level at which your callers become exempt from the credit system. If you do not plan to use this credit system, just set this variable to 1 (making all your callers exempt).

    While we are talking about the credit system, I want to go over a few more things. If one of your callers uploads a program that you do not want to count as a credit, you will need to use Password Maintenance to change their number of blocks uploaded variable to less than what is currently in that field (depending on the number of blocks uploaded). Say for example they already had 100 blocks uploaded and they uploaded another 50 that you do not want them to get credit for. Go into Password Maintenance and change the blocks uploaded from 150 to 100. If they had already downloaded blocks based on that upload credit, then when they call next time, they will have to upload many more blocks before they can download again. This is important, as some callers will try to take advantage of the system by uploading the same files over and over (with different names) or by padding normal files to get more credit for the upload (often you can see the padding in your transfer window during an upload/download).

    Included with this package is a program called "pswd tools" that can be used to reset all or your caller's upload and download counters, or optionally just one caller's counters. This would be useful if you decide to start the credit system over after your BBS has been in operation for a while.

    The way this BBS program tracks credits is by a mathematical formula based on the caller's number of blocks uploaded and downloaded. The formula used to calculate download credits is:

    (free credits) + ((uploads)x(credits per upload)) - downloads

    This formula can calculate a negative credit if you implement this credit system on an operational BBS without first clearing the upload and download counters, or if you remove any upload credits from a caller who had already downloaded blocks based on that upload's credit, or if later you change the number of credits per upload or number of free credits variables. If the number is less than or equal to 0, then that caller will not be able to download until they upload enough blocks to make their credit status positive again.

11. Max files on public msgs drive
    Sets the maximum number of files that you want to be on the public messages drive. If you are using a 1541, I suggest 135. If you are using an SFD 1001, I suggest 210. The way this

works is that after a caller logs off, the system will go to the public messages drive and count the total number of files on that drive. This routine is handy if you are storing both public and private messages on the same drive, or if you are allowing uploads to go to the public messages drive. Since there is no way of predicting just how many uploads or private messages will be on this disk, we can program the system to scan that directory and if the directory is getting close to filling up, scratch some public messages. This allows us to run our small 1541s near capacity (making the most use of the drive all the time). A 1541 will handle 144 files, so a setting of 135 leaves space for 9 new messages/uploads to be stored on this disk during any one call. One more thing, if you are only storing public messages on the drive (nothing else) or using a 9060, 9090, or any other hard disk drive that has an unlimited directory, there is no need for the system to spend all that time counting files. If this is the case, entering a 0 here will tell the system to not count files after each logoff.

12. Number of days to hold mail
    Sets the maximum number of days that you want the BBS to hold private mail before deleting it. If some of your callers do not call for a long period of time, there could become a significant amount of mail online that really does not need to be there. Most likely if a caller does not read their mail for more than 30 days, by the time they do read it, it is no longer important. When mail is deleted, an entry is made in the caller log at midnight indicating the member's number who had their mail deleted. This could be useful in determining if this value is set too low and deleting too much mail. There is a safety check built in to prevent everyone's mail from being deleted if you accidentally type in the wrong year or month when loading the BBS. When the system calculates how many days a piece of mail has been online, if that number exceeds the purge mail variable by 7 days, that mail will not be deleted. You should remember this if in the future you decide to change this purge mail variable.

13. New user access level
    Sets the access level you want to give to your new users. There are nine access levels allowed. Each level will have a certain amount of access and a certain amount of time as defined by you the SYSOP. I like to start all my new users at access level 1 and increase their access only after validation. But you may feel differently, so this is where you would set the new user access variable.

    Now would be a good time for me to describe the various access levels as they are defaulted. You will be able to redefine much of this by changing the information on appropriate SETUP screens. But if you don't, Table 10 lists what the different access levels will look like.

Table 10 - User Level Capabilities

Level	Capabilities
1	Recommended for all new users until verified ✓ Read System Messages, Help files ✓ Read Public Messages

Level	Capabilities
	✓ Hold private mail / Send private messages ✓ Perform download/uploads ✓ Read text files
2	This access level allows new users to look around and decide about joining. This is also not a bad place to start all new users. ✓ Read System Messages, Help files, public messages, text files ✓ Hold private mail / Send private messages ✓ Perform download/uploads
3	Good level to give callers if you are trying to get voluntary contributions. You could also set the credit system exemptionlevel to 4, so that level 3 callers have something to gain by sending in a financial contribution ✓ Read System Messages, Help files, Public Messages, Text Files ✓ Post Public Messages ✓ Uploads / Downloads × Hold private mail / Send private messages
4-5	✓ Full use of system Since both levels have the same abilities, you will be able to setup the different message categories, download directories, time limites, etc and maybe give some of your most contributing callers more access.
6	Good option / access for long distance callers or personal friends so they will not be disconnected before their daily time limit has been exceeded. If your “Per Call” time limit is the same as the daily time limit, then this leel can be considered one more variation to levels 4 & 5 ✓ Adds exemption for “per call” time limit ✓ Removes maximum download limit per call ✓ Removes time between calls limitation
7	Access for regular sysops. They can be of help in maintaining the download directories and by removing offensive or no-longer-needed public messages. In addition to level 6 permissions, this level provides: ✓ Scratch any public message or download ✓ Edit download descriptions ✓ Release uploads for public access
8	High-level Sysop. Includes all features of Level 7 with addition of: ✓ Set Clock & Date ✓ Change another user's Access Level (cannot make a user SYSOP or change a SYSOP’s level) ✓ Read caller log This level gives enough access to allow someone to run the BBS for a weekend or so while you may be away. YOU DO NEED TO BE CAREFUL WHO YOU GIVE THIS ACCESS LEVEL. Since a level 8 will be able to see anyone else's passwords, you owe it to your callers to make sure the people you select for level 8 are trustworthy!
9	Adds to level 8 the abilities to:

Level	Capabilities
	✓ Password Maintenance ✓ Full DOS capabilities This access level should only be given to yourself and possibly one other caller if you are going to be gone for a long while. There really is not much you cannot do with this access level, so you should be very cautious about who you give level 9 access.

14. Mbrs expired access level
    Sets the access level you want the BBS to assign to a caller when their membership has expired. In each caller's password record, there is a field that can contain a membership expiration date. You would use the Password Maintenance or the (V)alidate option to enter/edit this date. At midnight, a compare of this date to the current date will be made and if there is a match, that caller's access level will automatically be changed to the level you define here. This allows us to give a caller a certain number of days on the system and not have to try to remember to lower that caller's access level on any certain day. One use of this feature would be to give callers a 30-day trial membership, for example. And since the system automatically lowers your caller's access level, there may be less hard feelings towards you afterwards.

15. Upload auto-release level
    This question needs some explanation. When a caller uploads a file to the system, do you want that file to automatically be released as a public download, or would you prefer it to be held for a SYSOP to review before it is released? With Color 64 BBS, you can have it so that lower-level caller's uploads are held for "sysop release", while higher level caller's uploads will "auto release". Make sure you never set this command higher than the [Y]Release A Download command or your SYSOPs will not be able to release any uploads. You may want to setup your system so that just your SYSOP's uploads will auto-release. If so, enter 7 here. Or you may want all uploads to auto-release. If so, enter a 1 here.

16. System Baud Rate
    Sets the BPS (Bits Per Second) rate that the system will set itself to when it is waiting for a caller, so it should be set to the maximum possible rate for your modem. This is because Hayes-compatible modems should automatically step-down to the correct BPS rate when someone calls using a lower BPS rate. If you are not using the SwiftLink RS232 interface, then the highest BPS rate for your system is 2400 BPS. For SwiftLink users, the use of high-speed modems becomes a little complex. Most high-speed modems that support MNP and rates above 2400 BPS allow the computer to communicate with the modem at rates much greater than even the maximum connect rate of the modem. In our experimentation with the Supra Fax 14.4K BPS modem, we have found you can set the rate at 38,400 BPS with no problems at all. This allows the computer to send data to the modem at the fastest possible speed and allows the modem to take care of the necessary data buffering. The computer and modem will be communicating with each other at a rate of 38,400 BPS even if the host modem and the remote modem are communicating at only 1200 BPS, but only if the computer doesn't have to adjust its BPS rate to the connection rate (as some modems

require). Thus, this setting should be used in conjunction with the "Adjust BPS to connect rate" question asked later in this section.

17. DD$ MCI on
    This question deals with the built in MCI (Message Command Interpreter) commands, which allow you to print messages and do special formatting inside of text files. One of the messages that the MCI commands can print is the current contents of the variable DD$ (see section 3.8, on MCI commands, for more information). DD$ is one of the items that is stored in each caller's password record, and it is up to the SYSOP on how this item is used. It is limited to 8 characters, so this field can hold a short 8-character note about the caller that will be printed on the Password Maintenance report. It can be used to note callers that have been given free access; you can enter SYSOP for callers who are SYSOPs of other systems, or GUEST for members on your system as a guest, or REMOTE for remote SYSOPs. If you want to keep this information private (i.e. the MCI commands won't print this information), then answer "N" to this question.

18. Using upload descriptions
    Enables or disables the "upload descriptions" feature of Color 64 BBS. Most of you will want to answer "Y" to this question. The only time you may want to say "N" is if you do not have enough disk space and all the extra file descriptions will fill up the directory or disk. The popular SFD 1001 disk drive has a limitation of only 224 files in its directory. This sounds like a lot until you realize that each download requires a second file for the description. So, if you are using an SFD 1001 and you want to use upload descriptions, you will be limited to 111 files in the downloads (allowing for the. directory and √dir.tmp files). This gets even worse if you are also putting system files, messages, help files, and/or text files on the same SFD. Anyway, that is why I have it set up so you can run either way. Also, if you answer "n" to download descriptions, it is possible for you or your SYSOPs to manually enter a download description on any specific file. Just use the [E]dit Dnld Desc command to create the description file. Also, all download descriptions will be stored on the disk as "@dnldname". Example, the description file for "ccgms termz4.0+" is "@ccgms termz4.0+". Before starting an upload, a check is always made to make sure a download under that name and a description under that name does not exist. If you ever see a "file exists" error and cannot figure out why, remember about the download descriptions (even if you are not using them).

19. Multiple dirs per drive
    This question is asking if you will have multiple directories on one disk drive. This option is mainly for large disk drives (like 5, 10 or 20-meg hard disk drives) and will allow you to divide that large area of disk space up into more manageable, smaller areas. You could use this option on any disk drive, but it may not be worth the effort unless you have a lot of files on one disk drive. If you do answer Y to this question, every file uploaded to the system will automatically have a letter (A-Z) added to the beginning of its filename. The callers do not see this letter, it is just used by the BBS to group the files into different directories. If you manually copy a file onto your download disk, you will need to rename the file, so the proper

capital letter is the first character of the filename. There is a utility included in this package to help you rename your current downloads called "dir rename". When you use it, the delete key deletes the first character of the filename, any other key pressed becomes the first character of the filename. Remember, if you answer N to this multiple directory question, you will not need to rename any files. They will be displayed to your callers exactly as they are on the disk, with all the files on one drive in the same directory.

20. Daily Log Backup
    Enables / Disables the daily backup of your caller log during the midnight reset routine. If you answer "N", then the caller log file will just contain a running list of the most recent calls. If you answer "Y", then each night the caller log will be backed up and dated, and the caller log will be cleared. In either case, the size of the current caller log is limited by several factors. First, the minimum space on the caller log drive is determined by the "Minimum blocks-allow uploads" question. Also, two later questions in this section, “Caller log max size blocks" and "Caller log trim blocks", determine the maximum size of the caller log and the number of blocks that will be trimmed from the caller log if the maximum size is exceeded, or the minimum space is available. See the section in this documentation on the Caller Log System for more information on how the caller log is maintained.

21. Rerun on Errors
    Enables / disables program restart on BASIC errors. If a caller does find a way to crash the BBS causing a BASIC error or if you have made a change to the BASIC coding and accidentally left an error in the program, the system will automatically rerun itself. Normally you DO want this option enabled. We all sleep better at night, knowing that the BBS will still be running when we get up in the morning. The only time you may want to disable this option is if you are making modifications to the BBS programs and are still testing the changes. Note that when rerun is enabled, the STOP key will be disabled. Once the STOP key is disabled, then only by pressing a combination of the SHIFT, COMMODORE, and CONTROL keys will the program "break". See the section titled "Additional Commands Console Mode Only" for more information on this feature.

22. Screen Blanking
    Enables / disables screen blanking on SYSOP side when a caller is reading or writing a private message. There are some that feel that a private message should be totally private, not even read by the SYSOP. While others feel the SYSOP is totally responsible for everything on the BBS, even the private messages. This option allows you to set up your system the way you feel is best. If you do answer "Y" to this question, the screen will be blanked when a caller is reading their mail or sending mail to anyone else on the system. The screen will NOT be blanked when a caller is sending feedback or private mail to your mailbox or when you are accessing the system from the console.

23. Does your modem support DTR
    Identifies if your modem and interface both support DTR (Data Terminal Ready) and if you have all switches set such the phone line will disconnect if DTR is false. Most Hayes-

compatible modems and interfaces do support DTR, The Volksmodem 12 when used with cable J does not support DTR, nor does the 1670 (the 1670 does support DTR but seems to work better if we disable this feature). If you are unsure, answer N to this question.

24. Adjust BPS to connect rate
    This question is for those SYSOPs who will be using high speed modems (2400 MNP and higher BPS rates). Once a caller connects to the system, most 1200 and 2400 modems will automatically step down the BPS rate to the connection rate and expect the computer to do so. If you are using a 1200 BPS modem or a 2400 BPS modem, then most of the time you should answer yes to this question. Otherwise, you may need to do some experimentation to find out if your high-speed modem requires the computer to also step down its BPS rate. From our own experience we know that the Supra Fax 14.4K BPS modem will let the computer communicate with the modem at a rate up to 38,400 BPS even if the modem-to-modem rate is only 1200 BPS. Thus, for the Supra Fax, you would answer "N" to this question.

25. Run Network v1.26a
    This is a very important question because it determines if your system is going to run Network 64 along with the regular BBS system. Network 64 is an addition to the regular Color 64 system that allows your BBS to exchange messages and files with other Color 64 BBS systems either locally or long distance. If you are setting up your system for the first time, it is best if you don't run the Network right away. Run Color 64 as is for a while just to get the hang of using your BBS system. You won't even need to copy the Network files onto your Program Files drive. If you do decide to run Network, then you should read the Network section before answering "Y" to this question.

26. Modem Init Command
    This is where you will define the initialize command to be sent to your modem before accepting a caller. By allowing you to enter the initialize command here (instead of having it hard coded into the program), Color 64 BBS can support many more types of modems. But this also means you will need to know a little about your modem before you can answer this question. I am not going to attempt to explain each setting in the init string, please look in your modem manual if you wonder what they do. I do want to say a few things though:

    - First, the init string must begin with “at" (lower case). This stands for attention and needs to be entered only once even though many commands are in this one string.
    - Second, if your modem will accept it, it is best to follow the attention with a reset (z). This reset is required for many types of modems to keep them from getting locked up. But on the Hayes Smartmodem 1200 and both Commodore 1670 modems, a reset will cause the remainder of the init command to be cleared. So don’t use the "z" reset with these modems. In summary, use “ate1x1s0=0s10=30f1q0v1m0" for the Hayes Smartmodem 1200 and most compatibles. If you are using a Volksmodem 12 or some other Hayes-compatible that does not clear the command buffer when it executes an "atz”, your modem may work better if you add a "z" just after the "at" and just before the

"e1". Change the modem init command to “ate1x1s0=1s10=30f1q0v1m0" for the new (4 switch) version 1670 or “ate1x1s0=2f1q0v1m0" for the original 1670 (the version with only 3 switches). If you have the 17XX ram expansion module with an old 1670, it is safe to change the "s0=2" above to "s0=1".

- If your modem does not support any of the commands in the above init string (such as the s10=30), just take that part of the command out.
- If you are having a problem finding which part of the init string is causing you a problem, you can load up a terminal program and try typing in the command.
- If you get an "ok", then you are all set. If it says "error", you need to find which part of the command is not supported. Also, if after loading the BBS the waiting for caller screen is printed, but the time and date are not printed, then there is something in this modem init string that is not set correctly. If you have any problems getting your modem setup right, let me know. I have had pretty good success helping others get their modems set up correctly.

27. Network Modem Init Command
    This is just like the previous parameter, except that the Network modem init command is sent to the modem before each outgoing Network call. SYSOPs with high-speed modems can set this command so that MNP or other error correction is turned off during Network calls (error correction can interfere with Network calls).

28. Scratch any msg level
    Sets the access level to scratch any public message. Normally, callers are allowed to scratch only their own messages if they wish. Anyone who has the level for this option (default is 7) will be able to scratch any public message on the system, even if it is not their own. This should be reserved for remote SYSOPs who take care of message maintenance.

29. Category/link level
    Sets the access level to change the category or the threading link of messages. Someone with access to this option (default is level 7) will be allowed to move a message into another category or link a message to another message so that it will be part of the same message “thread". This should be reserved for remote SYSOPs.

30. Merge seq file level
    Sets the access level to merge a sequential file into a message. When someone is editing a message, and they have access to this option (default is level 7), then they are allowed to use the "*" command from the editor prompt to merge in a sequential file of their choice. This should be reserved for remote SYSOPs who do message maintenance.

31. Caller log delete level
    Sets the access level to delete the caller log. Someone who can do this (default is level 8) will be able to scratch the caller log and start it over after viewing it. This is for remote SYSOP use.

32. Message MCI level
    Sets the access level needed to use the special Message MCI commands anywhere on the system. This defaults to level 2, although you may wish to limit the use of the Message MCI commands (sometimes they can start quite a controversy) to higher level individuals. The Message MCIs were mainly provided as a shortcut to print certain information such as the time or date, although many people find it entertaining to put them to fun uses.

33. Variable MCI level
    Sets the access level to use the Variable MCI command. Most of the time, use of this advanced MCI should be restricted to the highest-level SYSOPs. An error in a Variable MCI command can cause the system to crash, and if the error occurs while a sequential file is printing it may cause the reboot routine to fail. So, make sure that whoever has access to use the Variable MCI command (default is level 9) is VERY familiar with the way BASIC works and the way that the Variable MCI command works.

34. No DL file limit level
    Sets the access level at which a caller is exempt from the the "Maximum downloads per call" setting. The default is level 6.

35. Min msg memory bytes
    Sets the minimum number of free characters of memory when in the text editor. When you edit a text file, the computer has only so much free memory in which to store the message, and as you type each line the amount of free memory decreases. This value determines the minimum amount of memory that can be available before the computer will not let you add any more lines. If you set it to a high value (like 700), then you will avoid lengthy "garbage collection" delays while the computer tries to compensate for low memory. If you set it to a lower number (like 300), you will have more memory available for editing the message.

36. Caller log max size blocks
    Sets the maximum size (in blocks) of the caller log file. The caller log is constantly being added to, so it is necessary to put a limit on the size of the file to avoid excessively large files. The default is 50, with an allowable range of 8 to 200 blocks. If the maximum size is exceeded, then the caller log will be trimmed, using the value set by “Caller log trim blocks".

37. Caller log trim blocks
    Sets the number of blocks that will be trimmed off the caller log when the maximum size is exceeded, or when the minimum upload blocks is reached. The default is 8, with an allowable range of 4 to 100 blocks. It should also never exceed half the maximum size of the caller log.

38. Validate via e-mail level
    Sets the access level to validate another caller through e-mail. If you have a remote SYSOP who does validations, they can validate a caller by (A)uto-replying to the caller's mail, then

typing (V)alidate at the editor command prompt. If they have access to this command, they will be able to set the caller's level to validate them. The default for this command is 8.

39. Edit any message level
    Sets the access level to be able to edit any public message. Normally callers may only edit their own messages, but a caller who has access to this option will be able to edit any message on the system using the [E]Edit message command. Only that caller's messages will be displayed, but he/she can type any message number at the prompt. This should be reserved for remote SYSOPs who do message maintenance.

40. Max chars/40 column header
    Sets the maximum width of the header information as printed in the customizable header routine (see section 3.13, on the customizable message headers), when the system is currently set to 40 columns output. If you wish to use the included custom header file, then this parameter must be set to 30 (the default).

41. Max Chars/80 column header
    Same as "Max chars/40 column header", except for when the system is set for 80 columns output. The default is 70, which will work with the included customized header files.

42. No time restriction level
    Sets the access level when a caller will not be restricted by the daily time limit as set in the caller's Time Limits section. The default is 6.

43. Edit DL Description
    Sets the access level for callers who can edit download descriptions, after using the [Z]View DL Description command to view the description first.

2.7 SETUP - Disk Drive Assignments

This section will be used to assign which groups of files will be stored on which disk drives. It is here that you will be able to balance the load between your drives to make the most efficient use of your hardware. If at first you are not sure where to put which files, don't worry too much. Once the system starts running and if you see one drive has too many files, you can rerun SETUP, change the appropriate variables, then move the affected files.

Each of these questions asks for three parameters, separated by commas. You will see "8,0,i0" printed as the default when you first run SETUP. The first number is the device number of the desired drive and can range from 8 to 30. The second number is the drive number and can range from 0 to 1 (for dual drives or LU0 or LU1 for the Lt.Kernal hard disk drive). The third entry (the i0) is the command to be sent to the command channel whenever this group of files are requested. You should now read the section "Drive Initialization Commands" for information on which init commands should be used for your system.

For each question, you can either press RETURN to accept the current value listed in brackets or you can edit the value. To edit the value, you may retype the entire entry, separating each parameter with a comma, or you can leave one or more parameters unchanged by skipping it, but you must still include a comma to separate the parameters. For example, if the current value is "8,0,i0" and you just want to change the init command to "cp3!i3", then you would type in ",,cp3!i3" and press RETURN. If you just want to change the device number to 9, then you would type "9,," and press RETURN. Notice that you must always have two commas in the input line.

1. Password File
   Sets the device, drive number, and disk initialization command for the disk drive where the "√password file" will be stored. You can "point" this to any drive you want, but if you point it to a drive other than a 1541, make sure the drive is capable of handling relative files. For example, the SFD 1001 does a fine job with relative files, but you cannot store relative files in a chain of ICT HD partitions.

2. System Files
   Stored on this drive are the welcome messages, logoff message, menus, information file, membership list, membership full message, or any message that does not fall into one of the other categories. The System Files drive should be able to store relative files.

3. Help Files
   Stored on this drive are all the Help Files that your callers can access. See the section on the Help Files for more information on this subject.

4. Public Messages
   Stored on this drive are all the public messages which your callers will post. Many of the questions in the Main Parameters (2.6) section relate to the Public Messages drive.

5. Private Messages
   Stored on this drive are all the private e-mail that callers can send to one another. Also stored on this drive will be the ".questxx" files for the new member applications. See the section on the new member Application routine for more information.

6. Text Files
   Stored on this drive are all the Text Files your callers can access. See the section on Text Files for more information on this subject.

7. Caller Log
   Stored on this drive is the current caller log, a log of the most recent call, and any dated backups of the caller log. Note that you can only set the drive number to 0 for this drive. This is because of BASIC programming space limitations, but it shouldn't cause any major inconvenience, as most drives use drive 0 to access the current drive (or partition for some

HDs). Lt. Kernal users will have to store the caller log on LU0. See the section on the caller log for more information.

8. Program Files
   Stored on this drive are all the BASIC BBS overlays, plus any ML files which will be loaded during BBS operation. If you are using an REU, then you must enter the device number of the REU you chose in the "bm ram" program.

9. Network Files
   Stored on this drive are all the files related to Network if you decide to run Network. This drive must be able to store relative files.

10. AUX 1 Files, AUX 2 Files, AUX 3 Files
    These three drives are for when you expand your system to include more games and modules. These are general purpose drives and can be used for any application you wish. See the Programming section for information on using the Auxiliary drives.

As you can see, the Drive Assignments section allows you to move your files around any way you need to best balance the load between your available drives.

IMPORTANT NOTE: If you are using the Epyx Fastload cartridge you MUST always use "ui" as the drive command for Program Files or you will experience intermittent system lockups.

Again, you will be asked "Is this correct?" If you need to change any of the previous answers, answer "N" and you will be taken through all the questions again. If you are sure everything is correct, just enter a "Y" and press return. Then you will be be taken to the Upload/Download Directories section.

2.8 SETUP - Upload/Download Directories

In this section of setup, you will go through a series of questions defining each of your upload and download directories. You can set up up to 26 different directories on any number of drives. Each directory will be assigned a description, download status, upload status, access level, device number, drive number and a drive command.

You will want to give each UD Directory a unique name since your callers will reference this name when selecting drives. The first drive that has a “Y" in the download status field will be the system's default download drive and the first drive with a "Y" in the upload status field will be the system’s default upload drive. This can be the same drive (simply by entering "Y" in both download and upload status fields) or totally separate drives. You need to make sure the access level assigned to these drives is for the lowest level callers. You would not want the system to default to one of your restricted download directories!

The first thing you need to do is enter the number of directories you would like to have on your system. You can have up to 26 directories, and each directory is referenced by a letter from A to Z. While you are entering the directory information, you should also keep track of the directories and their ID letter on paper. After you enter the number of directories, if you didn't change the number of directories (if previously set up), then you will be asked which directory you wish to edit. Enter the letter assigned to a directory to edit its information.

- Editing Directory Information
  If you have just started up your system for the first time, increased the number of directories, or you have just decided to edit a directory, then you will be asked to enter the following information described in Table 11 for each directory:

Table 11 - Directory Information Query Fields

Information Asked	Details
Description	The description is the name of the directory as you want it to appear in the list of UD directories. It should not be more than 30 characters in length, and you will not be allowed to include quotation marks in the description.
Allow Downloads	If you want to allow downloads from this directory, then answer "Y" to this question. If this directory is the first directory that has "Y" selected here, it will be the default download directory for all callers, regardless of their access level. This means that if a caller were to use the [D]Download command without first using the [#] Change Directory command, then they will be accessing the default download directory.
Allow Uploads	If you want to allow uploads to this directory, then answer "Y" to this question. If this directory is the first directory that has "Y" selected here, it will be the default upload directory for all callers, regardless of their access level. This means that if a caller were to use the [U] Upload command without first using the [#] Change Directory command, then they will be accessing the default upload directory.
Access Level	This is the access level that will apply to all callers attempting to access the directory unless the directory happens to be the default upload or download directory. You can set the level to 10 to effectively "turn off" the directory.
Device, Drive, Init	This is just like the questions in the Disk Drive Assignments section (2.7).

- Editing the Other Directories
  Once you enter the information for all the directories, you will return to the question asking which directory you wish to edit. If you do not wish to change any more information, then you can just press RETURN. Otherwise, you can enter the letter of the directory to edit, or you can press "-" to go back to the question asking for the number of directories.

  If you decide not to edit any other directories, you will be asked "Is this correct?" If you need to change any of the previous answers, answer "N" and you will be taken through all the

questions again. If you are sure everything is correct, just press "Y". Then you will be taken to the next section of the SETUP program. If after answering "Y", the program still jumps back to the beginning of the Uploads/Downloads section instead of going on to the next screen, make sure you have assigned at least one download directory and at least one upload directory. Even if you don't want to allow uploads or downloads, you must assign at least one of each (just set the access level to a high level if you want).

2.9 SETUP - User's Time Limits

These questions will be used to set the time limits for the different access levels. First you will be asked to set the maximum time allowed "per call" for levels 1-5 during AM hours and again during PM hours. This is different than the maximum allowed per day in that a caller may get 60 minutes per day but might only be allowed 30 minutes per call during PM hours, forcing them to not tie up the BBS for a continuous amount of time during what is normally the busier hours. In the default system setup, this variable applies to all callers with access levels 1 through 5, and access levels 6 and above are exempt. This can be changed through the "No time restriction level" in Main Parameters (section 2.6). You might give your long-distance callers and close friends level 6, so they will not be disconnected prematurely. Any value from 1 to 900 is ok. Note that if you do not want a "per call" time limit, then set either/both values to 900.

The next question is asking for the time required to pass before a level 1-5 caller can get back onto the system. You would use this value to prevent one caller from staying online for the maximum "per call" time limit, then immediately calling back for another session. Any value from 0 to 30 is ok. If you do not care to use this "between calls" feature, just put a 0 in this location. Again, access level 6 callers and above are exempt from this variable, unless you change the "No time restriction level" in Main Parameters.

The remaining questions in this section are for the daily time limits for access levels 1 through 9. Any value from 1 to 900 is ok. Set them any way you see fit. The system will not disconnect a caller if they are in the middle of a download or entering a message and their time expires. But the system will "borrow" the extra time used from the next day. So, if a caller has a 20 minute per daytime limit and spends 60 minutes typing feedback, the system will enter -40 minutes in that caller's membership record. Then every night at midnight, this time will be incremented by 20 minutes. This caller would not be allowed back onto the system until their daily time limit goes positive again (3 days in this example). Note that all callers are subject to these limits, regardless of the setting of "No time restriction level" in Main Parameters.

At the end of this section, once again you will be asked "Is this correct?" If you need to change any of the previous answers, answer "N" and you will be taken through all the questions again. If you are sure everything is correct, just enter a "Y" and press return.

2.10 SETUP - Caller Purge

The next section is where you will assign the number of days that must pass before the BBS's purge routine will automatically delete a member that has not called in a while. When the BBS deletes a member, they are completely deleted; there is no way to bring them back unless you have a "√backup password" file you can use to restore their record. An example might be to set the system to automatically delete any level 1 not calling in 15 days, level 2 not calling in 31 days, or level 3 not calling in 62 days. And set all other levels to 999 days so the system will not automatically delete them (you can use the membership expiration feature to lower their access level to a 2 after a set number of days or when their fees run out).

The way this routine works is every night at midnight, each password record is evaluated and if that caller has not called in the defined number of days, they are deleted. An entry is made into the caller log (their record number followed by an asterisk to differentiate from a membership expired status). As a safety check, when the system calculates the number of days since the last call and compares it to the current date, if the calculation exceeds the purge variable by more than 7 days, the system will not delete that caller. Without this safety check, if you accidentally entered the wrong year while loading the BBS and the system processed its "end of day" routines, ALL your members would have been deleted! I strongly urge you to use the caller purge feature, it will make the long-term maintenance of the BBS a lot easier.

At the end of this section, once again you will be asked "Is this correct?" If you need to change any of the previous answers, answer "N" and you will be taken through all the questions again. If you are sure everything is correct, just enter a "Y" and press RETURN.

2.11 SETUP - Message Categories

This screen will be used to enter the message base categories. You can have anywhere from 2 to 18 categories. When a message is posted, the caller will be asked to select from one of these categories for their message. And when a caller reads the messages, they will be given the option to read just the messages in one category or the messages in all the categories that they have access to. You will not be able to use quotation marks in your message category titles, because quotes in the name could cause errors when the BBS is reading in the "√bbs.parms" file. Each category will have an access level assigned to it. Any callers with a level lower than the category's level will not see any messages in that category. It is recommended that you assign your category levels in order with the lowest levels at the top of the screen and the highest at the bottom. That way when a lower-level caller sees the list of categories, they will not see letters missing from the list. The less they know they are missing, the less they are going to complain.

Again, you will be asked "Is this correct?" If you need to change one of the previous categories, answer "N" and you will be taken through all the questions again. If you are sure everything is correct, just enter a "Y" and press return.

2.12 SETUP - BBS Commands

The next section is where you can really start to customize your system. Every command that your caller's type from the main command prompt and the level required to execute that command are SYSOP defined. For example, if you do not like "O" for logoff, you can change it to "G" (goodbye). But then you need to remember to assign something different for the graphics command. And if you like level 1 callers to be able to read all messages, you will be able to adjust that here too. The commands marked "spare 1" through "spare 9" are for future expansion, although the system comes with Mod Menu pre-installed in spare 4. For now, setting the access level for the unused spare commands to 10 will eliminate them from the system. Currently, if anyone selects an unused space command (and has a high enough access level), the BBS will load in the appropriate module and then just return to the command prompt. These have been left like this so you will be able to easily add your own modules to Color 64 BBS using these commands. If you are not a programmer, don't despair; we already have dozens of modules written that you can just merge into your system. Many of them are available from our BBS as well as instructions for how to add them to your system (in our private text files area). Anyway, the spare commands are something you can make use of later, after you are more familiar with Color 64 BBS.

Table 12 provides a summary of all the commands built into Color 64, with their default character assignments and levels:

Table 12 - Summary of System Commands

Key	Name	Level	Description
R	Read Messages	2	Read public messages
@	Post Office	1	Post Office sub-menu*
P	Post a Message	3	Post a public message
S	Scratch a Message	3	Scratch a public message
$	Show Directory	1	Displays directory of current directory selection
D	Download a File	3	Perform single or multiple download
#	Change Directory Selection	3	Select a different directory
U	Upload File	3	Perform single upload
!	Edit User Stats	1	Change default terminal settings **
F	Send Feedback	1	Send Sysop Feedback
C	Page SYSOP for Chat	1	Invoke chat session with SYSOP
A	Alter Password	1	Allows change of password
O	Logoff System	1	Perform Logoff activities / Hang up
G	Graphics On/Off	1	Toggle Graphics mode
H	Read Help Files	1	Review System Help Files
W	Welcome Message	1	Re-read the system Welcome message
M	Membership List	1	Show Membership List / Search Members
I	Information	1	View BBS Information file
E	Edit a Message	3	Edit a public message

Key	Name	Level	Description
↑	Set Time & Date	9	Change system time and/or date
>	DOS Wedge	9	Sends SYSOP to DOS Wedge for disk commands
<	Password Maintenance	9	Sends SYSOP to User/Password Maintenance
N	New Downloads	3	Scans system for new downloads
X	Scratch a Download	7	Remove a download from system
T	Text Files	2	Lists for selection available text files for viewing
L	Caller Log	8	Displays call activity and actions on the system
+	Multi-Upload	3	Perform a multi-upload using Punter protocol
Z	View Download Description	3	
Y	Release a Download	7	Make a download available to end users
*	Games/Modules	3	Takes user to the Module menu
%	Protocol Select	3	Allows selection of specific protocol for file transfer
=	Post Network Mail	3	Send public/private network message ***
&	Billing Maintenance	9	Billing / Node maintenance ***
-	Release Public	8	Release public nets holding ***
Other notable internal functions (not a command assignment)			
Send Private Mail		4	
Restrict Posts ***		6	

* This is a menu of 5 options: Post a Message, Read / Send E-mail, Feedback, Membership List.
If a caller doesn't have the access level to use an option, it will not be accessible.

** Change customized information such as the number of lines for the page-pauser, character delay, and 40/80 column selection.

*** Network commands will be disabled online if you chose not to run Network.
For information on these commands and the Restrict Posts setting, see the Network documentation.

2.13 SETUP - Color Code Setup

The next screen is where you will define the 8 colors that Color 64 BBS will use. Throughout the program, whenever we want text in a different color, the BBS will select the next color from this sequence. You will see 9 reverse bars of numbers on the screen. To change the color for bar 1, just type **1** then type the control code for the desired color. For example, to make bar 1 medium blue, you would type **1** followed by **Commodore/7**. The bar will change colors right away. This gives you an idea of what these colors will look like next to each other. Keep adjusting the colors until you have this screen looking the way you like, then just press RETURN.

2.14 SETUP - Carrier Status

This section of SETUP is used to determine how the system recognizes if the modem is detecting a carrier signal. This is important because it allows the BBS to know when someone is connected to the system, or if they have disconnected. If the carrier status is not correctly determined, then the system will most likely not work at all. The screen will tell you to make sure your modem is

connected, turned on and ready to receive a call, but to make sure that no one is connected to it. It is recommended that you unplug the phone line from the modem to make sure no one is connected or that a dial tone will not generate a false carrier detect status. The BBS will now be ready to learn the carrier status of your modem. If at any time in the future you change modem types or switches insider your modem, it is important that you rerun SETUP with the modem connected so that the new carrier detect can be determined. Also, if you change your system from a non-SwiftLink system to a SwiftLink system or vice versa, you must run SETUP and select this section, otherwise your system will not run correctly.

When you are ready, press RETURN.

2.15 Saving the Parameters

If you are just setting up your system, you will return to the main menu once all sections have been completed. From the SETUP menu you should now select option 10 to save your parameters to your Program files. You should see some disk access while the SETUP program creates the necessary files. If you selected a large number for the Maximum Password Number, then it may take some time for the password file to be created. When finished, there should be 2 new files on your system. The first file is "√bbs.parms" and must always be on your Program disk or in your Program files. This is the file that stores the drive assignments, and until it is read in, the BBS program has no way of knowing where to find any of your other files. The second file just created is the "√password file" and it will be created on the disk assigned by the Drive Assignments section. This password file is where all your caller's names, passwords, access levels, time remaining today, last message read, etc. are stored. Right now, there is only one password in the password file. It is membership number 2, name SYSOP, and password SYSOP. The BBS has created this password for you. As SYSOP, must always have membership number 2. If you aren't using a Feedback Menu, all feedback will be put into mailbox number #2 and all feedback replies will come from membership number 2. You will want to use the Password Maintenance routine (**F6** from the call waiting screen or "**<**" while online) to change your name and password (see description below) before putting the BBS online for the public to call.

Notice that all these file names have a check mark in front of their name. This same check mark (shifted @) will precede all BBS filenames. This special code prevents anyone from being able to download any of these files and possibly reading your password file, feedback, or someone else's mail.

If at any time you decide that you need to change any of the above parameters, just load the "SETUP" program again. The existing "√bbs.parms" file will be read into memory and you will be able to select any of the sections displayed on the main menu. At most of the prompts you will be able to just hit RETURN to accept the current value, allowing you to quickly move through all the questions you don't wish to change. Just keep hitting RETURN until you get to the parameter you want changed. After you have edited all the sections you wish to change, just select option 10 from the main menu and a new "√bbs.parms" file will be written to disk. If you did not change the

maximum password number, then the password file will not be affected. Reducing the maximum password number will not change the file either, but those records already stored beyond the maximum number will be inaccessible. If you increased the maximum password number, you will be asked if you wish to expand the password file to the new number. If you answer "N", then the system will simply add a new relative file record for each new member. If you answer "Y", the file will be expanded to allow new member records to be stored. The difference between these two is that adding new records one by one (answering "N" to expand the file) can be somewhat slow while a caller is online.

2.16 SETUP - Printing the Parameters

From the SETUP main menu, you can also select option 12 to print a list of the parms you have entered. From the print menu, you can decide to print just one of the sections, or you can print all the parms. Be sure you have a printer hooked up to the computer as device number 4, and that the printer is online.

2.17 The System Messages

Now that you have defined all your parameters and all the Program Files and Boot Files are in their proper place, it is time to create your system messages. This is where you get to let your creative ideas flow. I have included a full set of example messages that you can use as is or edit to suit your own tastes. Use a file copier to copy all the sample "system messages" onto the disk you have assigned to store your System Files. As explained earlier, all these filenames must always have the check mark (shifted @) in front of the filename.

There are several ways for you to write your system messages. You can use any word processor that creates seq text files (like Easy Script), the stand-alone message editor program on the BBS PROGRAM disk, or the message editor routine built into the DOS section of the BBS.

To create/edit messages with the stand-alone BBS message editor, load the program called "+editor" and RUN it (this will be on you Boot disk for floppy systems). In a few seconds, you will see a menu with 4 choices:

1) Message editor
2) DOS wedge routine
3) Newsletter reader
4) Return to BASIC

I have included the newsletter reader subprogram in preparation for our plans for development of a Color 64 newsletter. To edit a message on any drive, press **F1** and you will see an input prompt for the device number of the drive that contains (or will contain) the message file. In brackets is a number; just press RETURN if this number is correct; otherwise, enter the desired device number and press RETURN. Then you will see a prompt asking for the drive number with a zero in brackets.

Again, if this is correct, just press RETURN, otherwise enter a **1** (for dual drive number 1 or LU1 of the Lt. Kernal hard disk drive) and hit RETURN. Next you will see a prompt asking for the drive command to be sent to the drive. Normally you will just hit return, unless you are using an HD system, or you are wanting to access the back side or a 1571 (u0>h1) or put a 1571 in 1571 mode (u0>m1). The last question is for the filename to edit. Enter the desired filename remembering to add the "√" or "@" symbol, as required. If the file already exists, then it will be loaded into memory for you to edit. If it does not already exist, then one will be created after you enter and save the message. This message editor works just like the message editor on the BBS for entering public and private messages. This message editor will allow you to create/edit messages almost as large as 500 lines, in full color. In contrast, the online message editor is limited to the maximum number of lines per message as defined in the "√bbs.parms" file.

Here is a description of the system messages that Color 64 uses:

2.17.1 √welcome1, √welcome2

Let's start by editing the welcome messages. For now, let's use "+editor" to edit the sample messages I sent you. Then later if you decide to totally rewrite the messages, you can use whatever seq file editor you want. The BBS has provisions for 2 different welcome messages ("√welcome1" and "√welcome2"). The first one will be sent before the password prompts and the second one will be sent after the password prompts and before checking your mailbox. The BBS expects both files to be present on the system messages drive. If either one is missing, there will be a short delay, then everything will continue as normal.

2.17.2 √logon stats, √logon stats 80

There is one other file that prints before "√welcome2", however. It is printed just after the caller enters their password, and the file is called "√logon stats". It is simply a display of the caller's current statistics so they can see what their status is on the system. If the caller has chosen 80 column output, then the "√logon stats80" file will be displayed instead. There are examples of both included in your Color 64 package, and both take advantage of the built in MCI messages of Color 64. Until you are familiar with the way the Variable MCI command works you should not try to edit these messages (they are already generic system messages).

2.17.3 √sysop news

In addition to the other welcome messages, there is an optional "√sysop news" file. This file will be displayed to your callers after welcome2 and just before checking their mailbox. If this file does not exist, there will be slight hesitation, then the BBS will continue checking their mailbox. There are 2 ways this file differs from "√welcome2". First the file cannot be aborted by the caller; It must be read to completion. Second, you can set a date in the file and only callers with their last called date

less than this date will read the message. I find this a handy place to post messages that you want all your callers to read. Let me take a few minutes to explain how the date works. First, if you don't enter a date, the whole file will be read to each caller every time they call. If the first line of the file is a date (must be in the format MM/DD/YYYY and be the only characters on that line), then the file will be read only to callers who's last called date is less than this date. If while reading the file, it finds a second date line (or third or fourth), if that date is less than the caller's last date called, the file will stop reading. This allows you to combine several messages in one file and only have your callers read the file one time. If they ever did want to reread the "√sysop news" file, they can do it by re-reading [W]elcome messages. One more important point to make. When reading this file, the cursor will skip down 2 lines only when it sees a date in the file and that date is greater than the caller's last called date. So, if you create a "√sysop news" file that does not have a date in it, you will want to begin the file with a couple blank lines to separate it from the "√welcome2" text. And when adding a second date in the file, you do NOT want to put any blank lines between the first message and the date just before the second message.

2.17.4 √logoff

Also, there is a logoff message named "√logoff" that will be sent just before the caller is disconnected. Again, this message is expected to be there, but no damage is done if it is missing.

2.17.5 √new user msg1, √new user msg2

The next messages you should look at are the new member messages. There are 2 of them again ("√new user msg1" & "√new user msg2"). The first one is just a short note telling the new user to make sure they make a note of their password information and leads them into the application section of the BBS. This message should be short so that the new member's membership number and password already printed on the caller's screen will not scroll off the top before they have had a chance to write down the information. The second message is the first message you will send to all new users just after completing the application. It should tell them how to start and stop messages and tell them what (if anything) else they will need to do to gain full access to your system.

2.17.6 √application

Speaking of application, Color 64 BBS allows you to fully customize this area of the BBS. You may just want the city and state they are calling from, or you may want to ask them everything. I recommend you keep your questions to a minimum. Realize that any information given is quite often bogus. I was on a system the other day and it asked my sex and occupation. What has that got to do with the BBS?? If you are going to ask such questions, maybe make a note in "√new user msg1" that they only need to answer as much as they are comfortable with. Anyway, there is a sample application named "√application" already on the disk. The application is a combination of a regular text and special "prompt lines" that will cause the BBS to stop and wait for the caller's

response to the application question. Any line beginning with a "#" (number sign) will not be printed, but instead will cause the application routine to stop and wait for the caller to type something. The text after the "#" symbol will then be part of the application information that is put in your mailbox. Then the next bit of text will be printed until the next prompt, where the BBS will wait for another response, and this repeats until the last line of the file is used. Then it prints everything back to the caller and asks if this is correct. If they answer "N", then the application routine will begin again. Otherwise, the answers are stored in your mailbox and in a file called "√questXX" where XX is the number of the month (e.g. √quest07 for July). The "√questXX" file is stored in the Private Files section. You will be able to print your mailbox for a hard copy while all your remote SYSOPs will be able to see the same answers in the "√questXX" file. If you want honest answers in your application, tell them at the end of "√new user msg1". See the section on "The Application" for more detailed instructions on using this feature.

2.17.7 √bbs closed

If you want to setup your system to not accept any new users, you can create a file named "√bbs closed". Whenever a caller enters "NEW" to access the new user routines a quick check is made for the presence of this file. If it is present, the file will display to the caller then they will be disconnected. Some people find this handy when going to be out of town for a period and others may just want to run a BBS for a select group of people (eg. a user's group). One exception is when you log on locally (at the console), you can always type "NEW" and get into the new user section. This allows you to easily enter in any new users without having to remove or rename the "√bbs closed" file.

2.17.8 √level ? msg

Another message you need to have for the new users is the "√level 1 msg". It is sent just after "√welcome2" for each caller that is still at an access level of 1. It is to remind them that they have not met your membership requirements and/or exactly what they need to do before they can gain higher access to your system. Of course, if you are not starting new users at a level of one, then you will not need this message.

Also, you can have a "√level X msg" for any level. When the system is loaded up, it looks on the system messages drive for "√level 1 msg", "√level 2 msg", "√level 3 msg", etc., and sets a flag for each one it finds. So, if you want to send a message to all level 3 callers, and all level 7 and 8 sysops, just create the appropriate message files and name them "√level 3 msg", "√level 7 msg", and "√level 8 msg", respectively. If the BBS is already running when you create or remove any of these messages, use (F4) or "." to reset the date and time. This routine also includes the code to reset the level message flags.

2.17.9 √membership full

Another message you might need for new users is "√membership full". This file will be used to notify a caller that membership to your BBS is currently full and ask them to call again later. The reason for having a membership full status instead of automatically cycling old passwords is to eliminate the chance of a "hacker" repeatedly logging on as a new user while you are away on a weekend and cycling everyone's password off the system.

2.17.10 √member list msg, √membership list

Another message you may want to edit is called "√member list msg". This short little message explains to the callers why their name may not be on the membership list even though they may be a new member. We need to have some control over what names are put on the membership list. So rather than have the system automatically list all names from the password file, we maintain a second sequential file called "√membership list". And there is an option in the Password Maintenance routine to make a new membership list. This gives you a chance to remove any offensive words in members names (you should be able to see them in the caller log) before making a public list that your callers will see. I am afraid to say that you may occasionally see some obscene names or offensive words used as members names. Anyway, the "√member list msg" is sent just before the membership list to explain why a new user may not see their name listed until after the system operator has had a chance to review all the new applications.

2.17.11 √information

Another message you will need to edit is the "√information" file. This should tell your callers a little something about your system like where you are located, the purpose of the system, etc.

2.17.12 √sysop not here / √still not here

Another couple messages you will want to personalize is "√sysop not here" and "√still not here". The first one should be a short little message that you send to your callers after they attempt to chat and if you do not respond to the page. The second one is read if a caller attempts to chat with you a second time even though you had not answered the first page. This prevents callers from wasting a lot of time ringing the pager repeatedly. And since the border on your console is highlighted whenever a caller wants to chat, there really is no reason to page the SYSOP more than once.

2.17.13 √upload message / √upload held

In the upload's routines, there are a couple more messages you may want to personalize. The first is √upload msg". This file now simply says "Do not upload any copyright software". This is a constant reminder to your callers that it is illegal to upload any copyright software to any BBS. You may want to mention what will happen if the rules are broken or rephrase it in some way. The second file you may want to look at is "√upload held". This is the file that tells a caller that their upload will be held until a SYSOP releases it. This file is only sent to callers who have an access level below the "auto release" level defined in the UPLOAD/DOWNLOAD section of SETUP.

2.17.14 √wfc

The Wait-For-Call screen prints the sequential file called "√wfc". The included file is a very good example of the effective use of the Variable MCI command, and you can use it as a template to create your own customized Wait-For-Call screen displaying the information important to you.

2.17.15 √msg.menu

The menu for text editor options has been put in the file called "√msg.menu", and is printed only when the caller hits (H) for help at the editor prompt. This file needs to be in your System Files so your callers will be able to get help on using the editor.

2.17.16 √menu?

In addition to the other system files, you will also need to edit each of the menu files. They are 9 menu files named, "√menu1", "√menu2", "√menu3", etc., one for each access level. Example menus are included. You will at least need to edit the line that says, "Your BBS name" and replace it with your BBS name. But hopefully you will be able to come with some different menus that will make your system unique! Also included with the system is a program called "menu maker" which will automatically create menus for your system for all the levels. This can save a lot of work, and if you know BASIC, you can edit the menu maker program to customize the way the menus look. Note that the menu maker program uses an MCI command to print each command key, so that if you change the key used for a command (but not the level), the menus will automatically print the correct key.

2.17.17 √mod edit menu / √mod sub menu

These two files are used by Mod Menu as help files when in the Mod Stat Editor.

2.17.18 Special Provisions for ASCII callers

Before moving onto the other files, I want to cover one more thing about system files. Often when you use a lot of graphics in your system files, non-graphic callers will have a difficult time reading the file (sometimes it is just impossible). This is especially troublesome in the menus and welcome messages. I do have provisions in Color 64 BBS to have alternate system files just for ASCII callers. But this is totally optional. If you skip onto the next paragraph, all callers will read the same messages (which is normally just fine). Anyway, there is a merge included in the system called "afr.ovxx" (AFR for Ascii File Read), which you can merge into all of your overlays except for the Network overlays. Here is how it works: First you create all the files as normal for the system. Then for each of these files that you would like to have a corresponding ASCII file, you create a file with the same name, except add a "+" at the end of the file name (e.g. √menu1 would be √menu1+). Any system file that is not actually “read" to the modem cannot have separate non-graphic versions (ex. √application, √sysop news, etc).

2.18 The Help Files and Text Files

A full set of help files are included with your Color 64 package, but no text files. The Help Files and Text Files work the same, so use the help Files as a guide when setting up your text files. There needs to be a file in the Help Files disk called "@help files". Notice that instead of having a check mark as the first character of the filename, it has an “@” symbol. This symbol will prevent the file from showing up in the download directory, but still allow your remote SYSOPs to download, scratch and re-upload these files as necessary. As a matter of fact, anyone can download these help files (you will just have to tell them how). This is an easy way for your callers to download them if their terminal program does not support an ASCII file capture.

The "@help files" file is a menu file that when read should tell your callers what number to enter to read the desired file. This menu file can be in any format you want, even use graphics if desired and you can use any number of numbers in any order you want. Each help file on disk should have the filename format of "@help1", "@help2", "@help3", etc. When your caller enters "1", they will read "@help1".

Text files work the same only you use "@text files" to describe “@text1", "@text2", etc. If the "@text files" file is not on the disk when a caller asks for the text files, he will see a message saying that there are not any text files online. Same goes for the "@help files" file. There is not any limit to the maximum number of help files or text files allowed other than the amount of free space on your disk drive.

After editing all the system messages, you should use a file copier to copy all the sample help files onto the disk you assigned on the drive assignments screen of SETUP. Then use MSG EDITOR to read (and if necessary, edit) each of these help files. They will tell you and all your new users how to use this BBS to its fullest. I did not repeat the same information here, so make sure you take the time to read them. In the UPLOAD/DOWNLOAD file, there is a sentence saying that you will be glad to make a copy of a Punter color/graphics terminal program for any caller needing one and for them to contact you through feedback if they are interested. I have included CCGMS TERM 4.0+ and

CGTERM 128 v3.3 in the Color 64 package. These are just a couple good color/graphic terminal programs for you to give to your callers. They each support many kinds of modems, variable baud rates up to 2400, punter, ASCII uploads & downloads and much more. If you would rather handle callers needing a Punter terminal program a different way, you will need to edit this help file and make the appropriate changes.

Chapter 3, BBS Operation

Now it is time to get into the actual operation of the Color 64 BBS system. The first thing we will do is boot the BBS system, and then I will cover the different parts of the BBS system as we encounter them.

3.1 Loading The BBS

At this point, you should have already created your parameters file, password file, system messages and help files. Each drive assigned in your parameters file should have a disk in it with the proper files already on it. If you want to start your system with some files in the download directory, make sure you put them in the proper drive now.

- Booting for Systems without Ram Expander
  If you are not using a 17XX series Ram Expander, you are ready to load Color 64 BBS for the first time (the big moment). If you are using a boot disk, or if you have the boot programs on your Program disk, then insert the disk now. Load the program called "+bbs" as if it were a BASIC program, then type RUN and hit RETURN. JiffyDos users can auto-RUN the program by typing “£+bbs" and pressing RETURN. Lt. Kernal HD users can just type "+bbs" and hit RETURN to boot the program. Once you run the program, you may see a message indicating that you should insert your Program disk. This is necessary because the Program disk contains all of the overlays for the BBS, and your "√bbs.parms" file. If you have your system set up with the boot programs and the overlays in the same area, or if you use the "bm small” program, you will not see the prompt to insert the disk. If everything loads normally, the computer will print "Working..." at the top of the screen. This means that the BBS is loading its parameters in from disk. Now skip to "Entering the Time and Date" below.

  If the BBS fails to boot normally, then it probably means that an essential file is missing in the Program Files or the System Files. Check the installation instructions to make sure that the correct files have been copied to the BBS system disks.

- Booting for Systems with Ram Expander
  If you are going to have your overlays load from a 17XX series Ram Expander, you may want to double check that you did answer "Y" to the REU question in the "bootmaker" program. If you did not, then the essential boot programs will not be in your Boot Files. Also, make sure that if you have separate Program and Boot disks that you answered "Y" to the "swap disks" question in the "bm ram" boot-maker program.

  Insert your Boot disk now if necessary. Next, load the program called “+ram.start" as if it were a BASIC program, then type RUN and hit RETURN. JiffyDos users can auto-RUN the

program by typing "£+ram.start" and pressing RETURN. Once you run the program, the BASIC script program that copies your programs to the REU will be loaded. You will see a lot of strange commands scrolling by on the screen, but this is just showing you what the script program is doing now. If at any time you definitely see a "file not found error" you should reset your computer immediately and then double check that all the necessary files are in place; the files copied off of the Boot Files into the REU are √sys.loadml, √sys.mlinit, +ram.bbs, +ram.reboot, and the ML file.

If there are no errors, then at some point in the process (depending on your setup) you may see a message asking you to insert your Program Disk and press RETURN. Just follow the instructions and you will be fine. The whole process can take from under a minute to several minutes, depending on how fast your disk device is and how many files are being copied onto the REU. Once the copying is done it will look like the computer is resetting, but this is just a normal part of the boot process. You should briefly see the Color 64 title screen and then the program will display "Working..." at the top of the screen.

- Entering the Time and Date
  Next, you will be asked to enter the correct date. You must enter it in the format of MM/DD/YYYY. MM is the number of the current month and must be 2 digits in length. DD is the number of the current day and must also be 2 digits in length. YYYY is the number of the current year and must be 4 digits in length. Separate the month, day, and year with a diagonal slash (/). Normally, you will never need to enter this date again. When the system is shut down (using F8) and many times during the day, a file is written to disk ("√variables") that contains the current date. Just above the question asking for the date is a line displaying the current date and time. If the date is already correct, you would just hit RETURN. If the date is changed (for whatever reason), then the BBS figures it was "asleep" when midnight passed and did not get a chance to reset all the time limits. If that is true, then answer Y to the question asking to reset the time limits. This will set a flag and each caller's time limits will be reset after the BBS program has finished loading up. If this is your first time to bring up the system, then just type an "N", since there really aren't any members' time limits to be reset.

  The next question you will be asked is about the current time. Color 64 uses one of the clocks built into the CIA chips, so you should find the clock will keep very accurate time. Once this clock is set and if the computer has not been turned off, the clock WILL be correct (even if you don't set it again for months). If the time is already correct, just press the RETURN key. But if the clock needs to be changed, you need to enter the correct time in two steps. First enter the current hour using 24-hour military format. 12 midnight would be 0, 10 am would be 10, 12 noon would be 12, 2 pm would 14, 8 pm would 20, etc. Next you will be asked to enter the current minute. No surprises here! Just enter 0 to 59, as required. This is a little confusing, but it saves a few bytes of precious memory that can be put to better use when a caller is online.

After you enter the time, the program will display the full time and date. For example, if the date is 12/10/1993 and the time is 4:19 pm, then the computer would display "Friday, December 10, 1993 - 4:19 pm". Note that the computer automatically calculates the correct day of week from the date you enter.

- Regenerating the Message Index
  After a brief pause, you will see a question asking if you want to regenerate the message index. When the system is shut down, a file is saved on the message disk that contains this message index. Having this file on disk can greatly speed up reloading the BBS. If the BBS should go down because of a power failure then, when reloading, the message index will not be present on the message disk and the BBS will automatically regenerate it. But if for any other reason you want to regenerate this index (maybe you scratched a few files while off line, or you had a disk error and lost several messages, or you went into SETUP and changed the number of messages on the system), answer a "Y" to this question and it will scratch the index file, forcing the BBS to recreate a new index a little later.

  After you answer this last question, the disk drive(s) will go to work for a minute or two and then the system will be ready to use. After your system has collected a lot of public messages and if you answer "y" to regenerate a new message index (or a power outage caused the system to shut down without saving the old index) this working time can be a long time (for slow disk drives) or perhaps as short as 5 minutes.

- Rebooting the BBS
  If you didn't notice, all the questions we just answered had defaults. Pressing RETURN would have used the default instead of having to type in the answer. Most of the time, if you take the BBS down to run another program, when you do reload the system, all you need to do is press RETURN 4 times and the program will bring itself back up. One exception is the time, if the BBS was powered off at all while the BBS was offline, the clock will have to be reset. Anyway, if you know you want to accept all defaults, you can load the "+reboot" program instead of "+bbs". This reboot file is also handy if you are running on a C128 and want to use one of the autoboot-64 utilities available in the public domain. If you are using an REU, then you would load the "+ram.restart" program to reboot and accept the defaults.

- The Wait-For-Call Screen
  When you see a screen that displays the current time, date, and the name of the last caller (and maybe other information if you are using the "√wfc" file), the BBS is booted and is at the waiting for caller screen. When the phone rings, the BBS will instruct the modem to answer, then wait for a carrier signal and for the caller to press DEL/BACKSPACE. If this does not happen, then the modem will disconnect and begin waiting for the next call. If you are talking voice to someone and want to connect them directly without them having to call back, just hold down the Commodore key for a couple seconds and the modem will pick up the line just as if it had detected a ring. Likewise, if you press the Commodore key just after the modem answers the line, it will hang up without waiting the full 30 seconds for a carrier detect. Also, after the fifth invalid sign-on within 24 hours, the BBS will go into a wait loop

and not answer the line for about 3 minutes. This should be sufficient to discourage callers from attempting to guess another caller's password. If you are nearby and want to abort this delay, you can press the Commodore key to allow the BBS to return to the Waiting for Caller status to bypass the timer for an invalid sign-on.

3.2 The SYSOP Menu

While the BBS program is waiting for a caller, if you press any of the function keys, you will see a SYSOP menu.

- F1: Local Mode Logon
  F1 on this menu is local mode. This allows you to go online the BBS just as if you called up over the phone line, except "√welcome1" will not be sent, graphics mode will automatically be turned on and text will scroll across the screen at a maximum rate. You will still need to type in your membership number and password and everything else acts just like you are online over a modem. When starting a new system, it is IMPORTANT that you enter at least one public message before letting any other callers log on. There are two reasons for this. First, this will force the last message read variable to be incremented by one (allowing new users to read their mailboxes the next time they log on to your BBS). Second, it is critical that when the first message is posted on a new system that you always answer Y to the question "Start a new subject (Y/N)?". When you type O for logoff, the BBS will save your last message read variable along with several other stats and then begin waiting for the next caller.

- F2: Term Mode
  F2 on the SYSOP menu will take you to the TERMINAL program. If you decided to put the "√bbs.term" overlay with your Program Files, it will be loaded and you will enter the Plusterm program. You will see another menu allowing Terminal mode, Change Graphics/Baud Rate, Autodial, Dos Wedge, Upload/Download, Protocol (Punter or Xmodem), Buffer, and Return to BBS mode. This term program supports full color capabilities and full multi-file transfer capabilities. It also has a built-in buffer function for capturing text information online. Once in terminal mode, you press F1 to exit back to the TERMINAL menu to start an upload or download, use the buffer, or return to the BBS. See the section on Plusterm for more specific instructions on the use of the Plusterm program.

- F3: Display Caller Log
  F3 on the SYSOP menu allows you to print the caller log onto the screen or a printer (Commodore or compatible setup as device #4 only). The caller log is stored on disk as √caller log" and is continuously updating itself. The log is limited to a size selected by you in the SETUP program and will hold information for several calls in the order they were made. When someone calls, their name and the time they called are added to the log. Then a “trail" is made of their activity. This can be helpful in determining what areas of the BBS are most frequently accessed or which downloads are most popular. Also, it can be useful in

tracking down a problem caller. Press the home key to pause (it generates the same code as a CTRL/S) or SPACE BAR to abort.

- F4: Network Menu/Set Time and Date
  F4 on the SYSOP menu is a dual function key. If you are running the Network configuration, then this will take you to the Network menu (see Network documentation for more information). If you are not running Network, then pressing F4 allows you to reset the date and clock and re-initialize the "√level ? msg" variables. There will be little need to change the time or date, but if you ever add or remove a "√level ? msg", make sure you select this option (or use the [↑]Change Time & Date command online) to reset the appropriate variables.

- F5: Display/Answer Mail
  F5 on the SYSOP menu allows you to print your mailbox to the screen or printer. (If you are using an SFD with the CSI Connect 400 interface to store your private messages, you will not be able to print your mailbox to the printer due to a bug in that interface). Use the HOME key to pause or SPACE BAR to abort. After reading your mail, you will be given the option to send a private message in response and clear your mailbox file. All replies from this selection are addressed as sent from membership number 2.

- F6: Password Maintenance
  F6 on the SYSOP menu will take you into the password maintenance routine. This routine is used to display (or print) and edit your password file. Again, if you are storing your password file on an SFD with the Connect 400 interface, you will not be able to print. Use the HOME key to pause and the SPACE BAR to abort. At the end of the listing, you will be told how many passwords are on file. If you see this value approaching the maximum number of passwords that you set your system up for, you will know it is time to either remove some callers that have not called in a long time or to increase the size of the password file. When the listing is completed or aborted (abort with the space bar - you may have to hold it down for a few seconds), you will be given the opportunity to edit any of the password records. Enter the membership number of the member to edit and that member's information will be printed on the screen (access level, name, password, etc.). If you need to add/subtract from the number of blocks uploaded/downloaded, or change the caller's name/handle, password, access level, or time remaining today, you will be able to do that here. One by one, you will be allowed to enter new information or just press RETURN to leave the information unchanged. To delete a member from the membership list, change their access level to "**0**". You will be asked "Are you sure?". This will also automatically delete any old mail that still may be on the system addressed to this member.

  The information that is printed when you are listing the password file is formatted in a special way. Here is a template of the information that is printed:

  record#: membership name, password,
  level, expiration date, membership information,

DL blocks, UL blocks, times called,
access group, last date called, block size,
last message read, time left, number of posts,
real name
phone number, birth date,
address line 1
address line 2

When you are printing the information to a printer, it will be formatted for 80 columns, but will still be in the same order as the above template. Refer to section 3.3 for information on the use of each of these fields.

One function you will need to perform regularly is to create a new "√membership list". This is the list that the callers will see when they select (M)embership list. Having a membership list separate from the password file that is updated only by the Sysop allows us to remove any embarrassing words or phrases from the password file before making a new public membership list. After you have listed and possibly edited the password file, you will be given the opportunity to create a new membership list. If you are just setting up your system for the first time and have not yet made a membership list, then now is a good time to edit your name and password. When you list the password file, you will only see one member. It has the password that the BBS automatically set up for you the SYSOP. Enter "**2**" for the record number to edit. Then press RETURN until you are prompted for your name. Type in your name. Then press RETURN until you are prompted for your password. Now type in your new password. Then type return a few more times (to skip past the rest of the information) and your password information will be saved to disk. Next, press RETURN when asked for the record number to edit and answer "Y" to create a new membership list. When the routine is finished, the program will automatically return to the Waiting for Caller screen.

- F7: DOS Wedge
  F7 on the SYSOP menu will take you to the BBS Dos Wedge routine. This routine is in "√bbs.xfer", so there will be a delay after pressing F7 (unless you are running an REU or HD system). It operates exactly like the one in the stand-alone message editor program. Table 13 below lists the Dos Commands available in the DOS Wedge:

Table 13 - Disk Commands in DOS Wedge

Wedge Command	Function	Format
@	Status/Read Error Channel	Single command
$	Display directory	Can pattern match – "*$0:A**"
#	Change Device Number	#<*num*>
N	Format Disk	N0:<*diskname*>,<*id*>
C	Copy File	C0:<*source_filename*>=<*target_filename*>
R	Rename a File	R0:<*oldname*>=<*newname*>
F	Read a File	F0:<*filename*>
P	Print a File	P0:<*filename*>
%	Regenerate Directory	Single Command
!	Message Editor	Single Command
X	Two Drive Copier	Single Command

An extra added to the directory routine is that it counts the number files selected. For example, if you want to see how many callers have mail waiting for them, you could switch to the drive that holds the private messages and then do a $0:?private* and hit RETURN. All filenames matching that pattern will list on the screen and at the end you will see a line saying how many files were selected. You can find this feature very handy when trying to balance your files between the drives. When reading a directory or sequential file, press the HOME key to pause, or the SPACE BAR to abort.

- F8: System Shutdown
  F8 will save the message index, caller log, system variables and exit to BASIC. This is the ONE AND ONLY proper way to shut the system down. If you hit the stop key by accident with the stop key enabled (rerun on errors turned off), type "cont" and hit return. If that doesn't work, type "**GOTO9999**" (without quotes) and let the system shut down, saving the message index, caller log and variables. Stopping the system without using F8 (or GOTO9999) will leave files open and some vectors pointing to the BBS's ML routines. In any case, using the computer for any other purposes in this condition will most likely cause a system crash. Also, the message index will not be saved to disk, causing the system to have to regenerate that index when the system is restarted.

3.3 Password Record Information

As you will notice if you use the F6 Password Maintenance function, there is a lot of information stored in each member's password record. Table 14 below lists the field descriptions contained in the record:

Table 14 - Password Record Field Descriptions

Field	Description
Access Level	Member's access level, which can range from 1 to 9. See the "SETUP-Main Parameters" section for a discussion of the default access level settings.
Membership Experiation Date	If your system is membership driven, then you may have certain membership periods. If you wish to have a user's membership expire on a certain date, then this is the field you would set. Once a user's membership expires, their level is set to the "Member expired level" in SETUP (See "SETUP-Main Parameters").
Membership Name (Handle)	“Handle” selected by user. This field is limited to 20 characters and can contain any standard ASCII characters (Commodore graphics are not allowed). Also, the membership name MUST contain at least one alphabetic character, because only alphabetic characters are matched during a membership name search.
Password	User’s password (entered by user); 3-9 characters in length. Not case sensitive and stored as uppercase and numerals
Membership Information	Not used by the system. Sysop may use at their discretion. This information is stored in the variable DD$ while the caller is online and can be displayed through the £a6 MCI command (see section 3.8, on MCI commands), but this MCI command can be disabled.
Blocks Uploaded	Number of blocks the user has uploaded. This is a raw figure and does not actually represent the user's download credit.
Blocks Downloaded	Number of blocks the user has downloaded. Download credit is a number depending on this figure, blocks downloaded, and the number of download credits per upload block (as defined in SETUP), so this can end up being a negative number.
Time Remaining	Amount of time caller has remaining for the day. This number is reset each night to the figures entered in SETUP
Number of Posts	Total number of public messages caller has posted.
Access Group	Defined for future use.
Last Date Called	Last date member logged on. Used during the midnight routine to check expiration and purge dates.
Last Message Read	Last public message the caller has read.
Real Name	User’s real name. Up to 30 characters in length.
Phone Number	User’s phone number. Up to 15 characters in length.
Birth Date	User’s birthdate
Street Address	First line of user’s address. Up to 30 characters in length.
City/State/Zip	Second line of address. Up to 30 characters in length.
Block Size	Used to determine if caller used XModem or Punter on their most recent file transfer. If 0, they used XModem. If greater than zero, they used Punter and the value shown is their block size.

3.4 Signing On

There are a couple ways to initiate a sign-on: A user can call your system using their modem, or you can do a local sign-on by pressing F1 from the Sysop Menu. If a user calls the system, then the BBS will answer the phone and attempt to connect with the remote modem. If the connection succeeds, the BBS will find out the baud rate and then display the graphics mode prompt. If you do a local sign-on however, then you will go straight to the user number prompt (section 3.4.2 below).

- The Graphics Mode Prompt
  The first thing that a caller is asked to do, once a modem connection is made, is to press their BACKSPACE or DELETE key. This is used to detect if the caller is using a Commodore graphics terminal program or an ASCII/ANSI terminal. Commodore graphics terminals send a character with an ASCII value of 20 for their DELETE character, while ANSI and ASCII terminals will send a character with an ASCII value of 8 or 127. If they are using ASCII/ANSI, then the BBS program will ask another question to see if the user wants to see full ANSI color and graphics. If the caller selects plain ASCII translation, then they will also be asked if their terminal requires line feeds added to carriage returns.

  Once the graphics mode has been determined, then the file "√welcome1" in the System Files is printed. You should test to see that this file looks OK in both Commodore, ASCII, and ANSI terminal programs, because it is the first welcome message that all callers will see.

- The User Number Prompt
  What occurs at this point is where sign-on begins if you select F1 from the Sysop Menu.

  First, a prompt is displayed asking for the user number of the caller. The user number is the record number in the password file that contains the caller's information; the BBS automatically assigns a number to each new user who applies to the BBS system. A new user would type "NEW" at this prompt to apply for membership on your BBS system. A current user would enter their user number, or a caller may press RETURN to view the membership list.

  When the system is first installed, there will be only one user: you the SYSOP. The SYSOP's user number is always 2, and when the system is first installed the the SYSOP's password is "SYSOP". You should use password maintenance to edit your record and change your password (and other relevant information too). You may be wondering why the SYSOP is user number (record) 2 and not number 1. The reason for this is that record number 1 in the password file is used to store the current number of records in the file. This information must be stored along with the password file because it could get lost or destroyed if stored in a different file.

If a caller enters a number at this user number prompt, then it will be checked to see if it is a valid user number. If it is not, then the BBS will tell the caller that the number is invalid. Otherwise, the caller will be asked for their password.

- The Password Prompt
  Passwords can be from 3 to 9 characters, and they are restricted to uppercase alphabetic characters, numerals, and some other non-graphics characters. As the caller is typing their password, it will not be displayed on your screen; only asterisks will be displayed as each character is typed. This is a small security measure for the case that you might not want someone in the room to see a password as it is being typed in.

  If the caller happens to enter the incorrect password, they will be notified that the password is incorrect and will be sent back to the user number prompt. If after three attempts the user cannot correctly enter a password, then the system will hang up and record the incident as an invalid sign-on. After the fifth invalid call within one 24-hour period the system will go into a waiting mode, where it will not answer any calls for about 3 minutes, to deter pranksters and hackers.

  If all goes well and the caller enters the correct password, then the rest of their information will be loaded in from the password file and the file “√logon stats" will be printed. If the user had set their default terminal setting to 80 columns, then the file called "√logon stats80" will be printed instead. These two files are used to display the user's current information, such as their number of calls, download credit, etc., by using the Variable MCI command feature built into Color 64.

  After this, the system will display the file called "√welcome2" as a second welcome message for the system. This file can be used for anything you want, or you can even omit it from your System Files if you feel that you do not need it.

  After the √welcome2 message has been displayed, then the system will print a “√level ? msg" depending on the level of the caller (i.e. level 1 users will see "√level 1 msg"). This can be used if you want certain messages to be seen only by specific level callers, or you can omit these messages if you do not need them.

  Once this is done then the "√sysop news" file will be printed, but since it has a special format, you may want to check section 2.17.3 in Chapter 2 for information on the SYSOP news file.

  Next, the system will check to see if the caller has any private e-mail waiting for them. If they do, then the system will do a quick scan of the mail headers to show the caller who the messages were from and who posted them. The caller can then decide if they want to read their e-mail or not. If the caller decided to hold their e-mail, or once they are done reading their e-mail, the system will then take the caller to the main BBS prompt.

- New Users
  If a caller entered "NEW" at the user number prompt, then the system would perform the following procedures:
  1) Will check to see if the file called "√bbs closed" exists in your System Files and if it does it will be displayed to the caller, after which they will be disconnected from the system. You should create this file if you want to close the system to any new users.
  2) If the BBS is not closed, then the caller will be asked to enter a name or a handle. This is what will be put into their password record as their membership name, the name that all the other users will see. The membership name must contain at least one alphabetic character and can be no more than 20 characters long. No color or graphics can be included in the membership name, so it can be made up of upper- and lower-case letters, numerals, and non-graphics characters.
  3) Next, will check the password file to see if that name already exists, and to find a spot in the password file for the new user.
     a. If the membership name already exists, then the caller will be notified and will be allowed to enter another membership name.
     b. If the password file is full, then the system will print the file called "√membership full" in the System Files to notify the user that no more space is available, after which the user will be disconnected.
  4) If all goes well, then the file "√password msg" in the System Files will be printed and the caller will be asked for the password that they want to use on your BBS. The password must be from 3 to 9 characters and is limited to a small set of characters to prevent callers from entering passwords with graphics characters (most often which they would not be able to remember easily).

  Once the caller successfully enters a password the BBS system will display the new user's information for review. After that, the file called "√new user msg1" in the System Files will be printed as a prelude to the application routine. This file can be used as information about the system, or specific instructions about the application to follow. Once that is done the user will be asked to fill out an application for membership on your system. I have detailed all the information about the application in section 3.5.

  Once the application is completed, then the BBS system will display the file called "√new user msg2", after which the user will be sent to the main BBS prompt.

- The Main BBS Prompt
  All callers, new or otherwise, will eventually end up at the main BBS prompt, from which they can decide which aspect of the system they want to utilize next. From this prompt the caller only needs to type a single key that corresponds to a main BBS command. Users will be able to use a command only if their level is equal to or higher than the level defined in SETUP.

Also, the user can also press the "?" question mark key to get a menu of commands available to them. The file called "√menu?" will be printed, where the "?" is the level of the user (e.g. a level 1 user will see the √menu1 file). You can use the included menu maker utility to create the level menus if you wish, because it will automatically put the valid commands for each level into the menus.

The function of each command is self-explanatory, and the system will guide a caller through a series of prompts for complex operations. Also, the caller will always have access to the help files, so that they can find out how to use specific features of the system.

3.5 The Application

The application routine is used to find out information about your new users. The application routine is also used to get other membership information such as a real name and phone number. Fortunately for you, this aspect of the BBS program can be fully customized. To use the application feature, you'll need to have the file called "√application" present on your System Files drive. The √application file must be formatted a certain way, so pay attention. I suggest you use the included "√application" file and then modify it to suit your own needs.

- User Input Prompts
  When the application routine first begins, it will spit out any text it finds in this .application file, UNTIL it sees a "#" as the first character in any line. When it finds the "#", it will stop, and await user input, then it will continue until it finds another "#" or the end of the file. The user input at each of these prompts is put into their application.

  The routine gives you the capability of including simple remarks for each question. For example, after a new user fills out the entire application, they are given their responses. The only problem is, they probably forgot what the questions were. This problem is corrected by allowing you to add a small remark which will show up after the caller completes the questions. These remarks will also go into your mailbox so that you will be able to know what the questions were as well.

  To include a remark, you place it after the "#" in the file. The remark should be as short as possible but should still convey the meaning of the information in the answer. Here's a sample of one question with the remark:

```
Please enter your real full name. Do NOT use a handle.
  #Name:
```

  Anything included after the "#" will precede the caller's answer when they review their information, and in the final application. So, if the caller answered: "John Smith", it would end up in your mailbox as: "Name: John Smith". Just remember that the remarks should be as short as possible.

- Getting Membership Information
  The application routine is where you will also get some of the information that goes into the caller's password record, such as the real name, phone number, etc. The application routine will recognize 5 special remarks that will indicate you want to ask for this information. To accomplish this, you use a prompt line in this format:

```
#(XXX)Remark Text
```

  "**XXX**" above would be one of the 5 codes (rn$, ph$, bd, a1$, a2$), which are simply the BASIC variable names for the real name, phone number, birth date, address line 1, and address line 2, respectively. The choice of these as the codes was arbitrary, but I thought it would make things easier in the future if more codes were added. Here is an example of prompts for all 5 codes:

```
#(rn$)Real Name:
  #(ph$)Phone Number:
  #(bd)Birth Date:
  #(a1$)Street Address:
  #(a2$)City State ZIP:
```

  When displayed on the screen to the caller, and when put in the mailbox, the "(**XXX**)" part of the codes will not be included as part of the text (e.g. it would appear as Real Name, not (rn$)Real name). The information that they enter will be put in their permanent password record, and the routine is currently set up so that the caller cannot leave a question blank. If you wish for callers to be able to leave these types of questions blank, then delete line 18851 of √bbs.init. Also, you can ask as many or as few of these special questions as you like but remember that this is the only automatic way to attain this information.

- Once the Caller is Finished
  The application routine will send the results of a new user's application to your mailbox AND to an ongoing monthly self-maintaining archive file. This file will be automatically created on your PRIVATE MESSAGES drive each month with the filename of: "√questXX". "XX" being the month of applications in that file. This file serves no purpose other than to keep a record of new users' info. You can scratch the file or copy the file off at any time you like.

  If you would like to redirect the app results to go into someone else's mailbox, look through the code (18700+) and look for the phrase "√private 2". You will see it in two places. Make sure you change it in both places if you want the mail to go to a different ID (perhaps a co-sysop who validates).

3.6 Additional Commands Console Mode Only

The keyboard is always active, even when a caller is online. This allows you to enter a command for a new user, log a caller off, etc.

3.6.1 Chat Mode

F1 from almost anywhere in the BBS will enable CHAT mode. F3 will exit. When you exit CHAT mode, a CONTROL/P will automatically be placed into the keyboard buffer. So, if your caller is reading a long text file, you will want to avoid breaking in until they reach a good stopping point. During the time you are in CHAT mode, the BBS call timer stops. The caller's time online and time used today will not reflect this time in CHAT mode.

3.6.2 Quick Caller Editor

It is possible to change a caller's access level, time remaining, blocks uploaded, blocks downloaded and online status while the caller is online. To do this, make sure the caller is at the main command prompt and press F5. On your screen only (the caller will not see this), you will see a six-part input prompt asking for level, time1, time2, dnld, upld,1=local.

- Level is the caller's access level for this call. Changing this variable is just a temporary change, the next time this caller calls back, they will be back to their original level. Use Password Maintenance to permanently change a caller's access level.
- Time 1 is the amount of time remaining on this call and time 2 is the amount of time carried forward for future calls today.
- Time 2 will be added to time 1 when the caller logs off, giving the amount of time that caller has remaining today. The current values for each input are already typed on the screen, so just cursor right and left to change the desired values and press RETURN.
- Dnld is the number of blocks this caller has downloaded and Upld is the number of blocks the caller has uploaded. If you want to give a caller some more download credits while they are online, you would either raise the blocks uploaded or lower the blocks downloaded. Local is either a 1 or 0. If you enter a 1 here, the system will go into local mode.

Even when you exit this routine, the caller will not see what is going on. This allows you to put a caller on hold, while you quickly look at a password in the password file or check the caller log. To put the caller back online, press F5 at the command prompt, then change local to 0. The caller on the other end will then see another command prompt.

3.6.3 Carbon Copy

There is a special carbon copy feature built into our private mailbox editor. After reading any incoming mail, at the (D)elete, (R)eread, (A)utoreply prompt, if you press the Commodore key you will see a "carbon copy" prompt. If you answer Y, this piece of mail will be transferred to the default message section of the BBS. I find this handy when a caller leaves a question in feedback that would benefit more people if it were in the public messages area. All you would need to do is select (P)ost a Message, answer N to private, select the appropriate category and then use **/*** to carbon copy the default message into the message editor (more on using Color 64's default message can be read in your help files on the back side of your master disk).

3.7 The Caller Log System

The caller log is stored on disk in sequential file format. The filename is "√caller log" and it is maintained through the parameters you defined in SETUP. The caller log must always be stored on drive 0 because of BASIC programming space limitations. This is the only type of file that has this limitation, so it should pose no threat to a BBS system.

As the BBS system is running, the variable **LG$** is automatically updated with log information whenever a program GOSUB's line 8003 (for **i$**) or 8004 (for **a$**). LG$ can hold a maximum of 250 characters before it is dumped into a special buffer in memory. This buffer can hold two "dumps" before the information has to be transferred to disk.

When the buffer is full, or when the BBS program forces the command, this buffer is transferred to a temporary file called "√l" on the Caller Log drive. If more than one file dump is made, the information is appended to the "√l" file.

When the BBS program returns to the wait-for-call screen, the "√l" file is appended to the "√caller log" file. This is when the message "Appending to Caller Log" appears on the screen. This message also appears when you shut down the system or the system crashes. This ensures that the caller log information is secure and complete.

In the process of appending to the caller log, a file called "√l.tmp" is created, which is an exact duplicate of "√l". This duplicate is not scratched and can be used to look at what occurred during the most recent call. This is taken advantage of with the "View Last Call?" question displayed when viewing the caller log. If you answer "Y", you will view a log of the last call. If you answer "N", then you will see the caller log in its entirety.

After viewing the full-length caller log, you will be asked if you wish to scratch the caller log. If you answer "Y", then the caller log will be scratched and restarted. If you answer "N" then the caller log will be left alone. If you wish to maintain daily records of the caller log, then do not scratch the log. The "Scratch Caller Log" option is level definable in SETUP.

Lines 28500-28770 in "√bbs.init" act as the caller log maintenance routine. This routine trims the caller log when it exceeds a designated size. The maximum size of "√caller log" is defined in SETUP.

The file will also be limited by the "minimum UL space" setting in SETUP. If either the max size is reached, or the minimum amount of space is reached then the "√caller log" file will be trimmed by the number of blocks designated in SETUP.

The default maximum caller log size is 50 and the default trim size is 8. The 8 doesn't mean that only 8 blocks will be trimmed in all cases, but it means that the file will be trimmed to the necessary length, then 8 blocks will be removed to allow for more filling. The process is completely automatic and attempts to compensate for even the worst possible case. If all attempts at reducing the size of the caller log fail, the routine will scratch the caller log to prevent a Disk Full Error.

3.8 MCI Commands

Color 64 supports MCI commands--MCI stands for Message Command Interpreter, a system used on many BBS types. In short, an MCI command is a set of characters that you embed in a message to control the appearance of the text or perform special functions. One command could inform the computer to wait for a key to be pressed, and another command could turn on rainbow mode. MCI commands are not performed until the lines you have typed are viewed by you or someone else (whether listing the lines in the editor or by reading the message itself). Until then, all you see is a representation of them as you type the lines. MCI commands can also be used with the BASIC output commands (see section 3.11.1, on using MCI commands in BASIC). In fact, MCI commands can be used anywhere that output can occur, except for BASIC's PRINT statement, because PRINT is not tied into the ML output routines.

While in the message editor, all MCI commands begin with the British Pound symbol (£). The next character in the MCI command tells the computer which command you wish to use. It must be an unshifted letter; the interpreter will not recognize shifted letter commands. A 'c' for example tells the computer to wait for a key to be pressed. On the screen the command would appear as '**£c'**. Of course, if you are in Uppercase/Graphics mode then unshifted characters will appear as uppercase characters (i.e. '.C').

Some MCI commands require additional characters to supplement the command, such as the rainbow mode MCI commands. If you wish to use standard character rainbow mode, then the sequence would be £r followed by 2 to 8 colors. To type the colors, just use the CTRL and C= keys just as if you were changing the cursor color (e.g. CTRL/2 is white, C=3 is light red, etc.). The colors will be used just like the rainbow mode enacted by the 'F3' character, except they will be your own chosen colors rather than the system colors. If you use less than 2 colors, the current rainbow mode will be left unchanged. If you use more than 8 colors, only the first 8 will be used. All rainbow modes except for Caps/Punctuation Mode accept 2 to 8 colors.

Table 15 below defines the standard MCI commands:

Table 15 - Standard MCI Commands

Cmd	Mode Name	Function
£r	Character Rainbow	Color changes for each character
£w	Word Rainbow	The color changes for each word
£s	Sentence Rainbow	The color changes for each sentence that ends with a period (.), question mark (?), or exclamation point (!)
£l	Line Rainbow	The color changes for each line that ends with a carriage return
£g	Paragraph Rainbow	The color changes for each paragraph that ends with two carriage returns
£p	Caps/Punctuation	This will use from 2 to 3 colors. If 3 colors are used, a separate color is defined for uppercase letters, lowercase letters, and punctuation/graphics characters respectively. If only 2 colors are used, then a separate color is defined for alphabetic and punctuation characters, respectively. Both versions produce a very nice-looking effect
£n	Normal	This command cancels the currently set mode (returning text to normal output)
£v	Character Velocity	This command is followed with a digit from 0 to 9. 0 sets the output speed so there is no delay between characters (which is the fastest). 9 is the slowest speed. This command is cancelled by a **£v0** command or by a RETURN.
£c	Wait	Wait for key to be pressed by user
£t	Tab	A number from 0 to 79 must follow this command which indicates which tab position to go to. If the current position is less than the tab position, then spaces are printed. For example, if you were at position 5 (which would be the equivalent of hitting the spacebar 5 times), then issued a **.t10** command, the computer would automatically add another five spaces to get to the correct position. If the current position is greater than the tab position, then crsr-lefts are printed. Note: If several tabs are to be executed on the same line, they should all be in the same direction (e.g. from left to right, or from right to left). Moving back and forth on the screen may produce undesirable effects.
£i	Wait for Line of Input	Waits for a line of input to be entered. Up to 127 characters can be entered. The text that was entered is stored in memory and can be printed with the £a0 command (refer to Section 3.9).

3.9 The Message Output MCI

There are two MCI commands that have level-restricted access. You will see that their levels of use are definable in SETUP. The first one is the message output MCI:

£a: message output MCI. This command prints a different message depending on the digit following the command as defined in Table 16 below.

Table 16 - Message MCI Commands

Cmd	Function
£a0	Print the input typed when the last **£i** command was issued
£a1	Print the name/alias of the current caller (This uses the current contents of the string NA$)
£a2	Print the last calling date of the user (**LD$**)
£a3	Current time (**T$**)
£a4	Current date (**DA$**)
£a5	Print the name/alias of the last caller (**LC$**). Note: If the last call was an incoming network exchange, then the name of the remote BBS will be stored in this variable.
£a6	Print the contents of the string variable **DD$**. This is the SYSOP definable "Membership Information" field in the caller's password record. See section 3.3, "Password Record Information", for information on its use.
£a7	User's real name (**RN$**)
£a8	User's birth date (**BD$**)
£a9	User's phone number (**PH$**)
£aa	User's street address (**A1$**)
£ab	User's city, state, and ZIP (**A2$**)

One more note about the Message (**£a**) MCI command. The command **£a6** prints the contents of **DD$** (membership information). The information in **DD$** may vary as you decide how it is used. Therefore, the option of disabling the **DD$** commands for security reasons is available in SETUP.

3.10 The Variable MCI Command.

The other level-restricted MCI command is the Variable MCI command:

£[: variable expression output. This MCI command lets you print any type of variable or expression you want.

An expression is just a mathematical formula which you use in BASIC to calculate things. For example, ***'5*4+3/16'*** is an expression, just as '***a$+mid$(str$(lv),2)***' is an expression. Just follow the £[with an expression, and then end the string of characters with a close-bracket symbol (**]**). This command requires some explanation about the inner workings of BASIC.

3.10.1 Keywords and Tokens

The BASIC language is made up of what are called "keywords", which are the very base commands and functions of the BASIC language. PRINT, for example, is a BASIC keyword. Since there really aren't that many keywords, it would be a waste of time and space if the BASIC interpreter stored each letter of each of these keywords individually. Instead, the BASIC interpreter scans each line as you enter it into the program and crunches each keyword into a single character called a "token". When you LIST the program, of course, these tokens are printed as the entire keyword. The keyword crunching process saves space when the program is being stored on disk, and it saves time when the program is running (for it doesn't have to figure out what the command is by searching character by character). When you are editing a BASIC program, the whole process of keyword crunching is invisible, because it is taken care of by the computer from when you enter the line to when you LIST it again.

The **£[** command is different, though, because it expects you to take care of the keyword crunching for it. The reason for this is that the regular BASIC interpreter is busy running a program and doesn't have any need to go about crunching keywords inside of MCI commands. Don't get worried, though, because I have installed a feature into this command which allows you to use it without understanding the BASIC tokens. Just read on though because the process of keyword crunching is very simple.

3.10.2 Typing the BASIC Tokens

It is pure chance, I guess, that all of the necessary keywords that would be needed in the **£[** command are easily typeable from the Commodore keyboard. This fact, though, allows you to access all the functions and symbols that you would have access to when typing a line in BASIC. The only difference, though, is that all of them are represented by a single character. The enclosed chart, labeled "BASIC Keywords and Their Tokens" ** shows all the numeric/string symbols and functions with their token equivalents. The "cmdr" by a character means to press that key while holding the Commodore logo key down, just like when you type Commodore graphics characters. If something is not on the chart, then it doesn't need to be converted (such as the "%" percent sign). Also note that all mathematical symbols such as the plus and minus signs need to be converted to a token. In the following paragraphs, angle brackets "< >" will enclose special characters.

** This documentation cannot be located at the time of the manual rewrite ~~ Nuke

Say you wanted the computer to print the expression ***"INT(RND(0)*16+5)***" with the £[command. The first thing you would do is consult the chart and find the token for "INT". This happens to be the C=J character (which looks like a thick vertical line). Then the parenthesis would be typed as normal. Next would be the keyword "RND". On the chart, it is listed as the C=F character (a small square). Then everything would be the same until the "*" character. So far, the command would be "£ [", then a C=J symbol, then a "(", then a C=F symbol, then "(0)".

Next, we remember that mathematical symbols like "*" are also converted to tokens. Using the chart, it would be converted to a C=D character (a small square). The "16" is fine as it is. The "+"

sign is converted to C=N (a thin vertical line). And the final parenthesis is left alone. The complete command would be (without the spaces):

£ [<C=J> (<C=F> (0) <C=D> 1 6 <C=N> 5)]

A final "**]**" close-bracket symbol finishes the command. This example appears on the keywords sheet as it would appear on screen.

Again, anything that isn't on the chart is usable as it is. Remember that the text between the brackets is under the same rules as any BASIC expression. Therefore, you can create a SYNTAX ERROR, or any other error a bad expression may create. So be careful to make sure the expression has been entered correctly. For this reason, it is also highly recommended that use of this command is restricted to only you, or those who you can absolutely trust with using the command.

3.10.3 Automatic Keyword Cruncher

If you still do not understand tokens, it is perfectly fine. This is a concept that rarely comes up in everyday practical programming. That is why I included the "automatic keyword cruncher". Be forewarned, though, that the automatic keyword cruncher calculates the expression a lot slower than a manually crunched expression. Keep this fact in mind if you notice slight delays while printing a file that uses a lot of Variable MCI commands. To use the automatic keyword cruncher, just include the "**>**“ greater-than symbol right after the "**[**" opening bracket. Here is a comparison of how the £ [command would be with and without the automatic keyword-cruncher, using the expression "**int(rnd(0)*16+5)**":

With automatic keyword cruncher:

£[>int(rnd(0)*16+5)]

Without automatic keyword cruncher (remember that <C=J>, <C=F>, <C=D>, and <C=N> are single Commodore key characters):

£ [<C=J> (<C=F> (0) <C=D> 16 <C=N> 5)]

As you can see, the automatic version is a lot simpler to read, although it takes up more room (in characters). I apologize for the long description of this command, and any confusion you may have now, but it is such a powerful MCI command that it is very difficult to explain.

3.11 MCI Command Security

You can set the levels for the "**£a**" Message and the "**£[**" Variable MCI commands in SETUP. If the caller doesn't have access to a command, then he will not be allowed to type the character after a

"£". Also, the default message loader will sift out any unauthorized use of MCI commands if the levels do not permit.

The way the security measures are accomplished involves substituting special characters in the place of what the caller is typing. For example, if the caller has access to the **£a** command, the input routine will convert the "a" in the command to an F2 (function key two) character. This process is completely invisible to the caller, and it appears that an "a" is being typed because that is what is printed. Internally, though, the command sequence is stored as a "£" character and an F2 character. If the **£[** command is used, then the "[" character is converted to an F4 (function key four).

All of this may seem to be a lot of programming to accomplish nothing, but it is the most effective way to prevent tampering of the MCI commands. Both the F2 character and the F4 character are "impossible keys" when online, meaning that no matter what a caller does, there is no way to directly input a converted MCI command if they do not have access.

3.11.1 Using MCI Commands in BASIC

Because of the security measures mentioned above, it is necessary for you to use the £a and £[commands in their native form when programming in BASIC. This means that instead of typing "£" then "a" for a Message MCI command inside the quotes of a BASIC statement, you will have to type "." then F2 (an uppercase reverse "I"). The same with the Variable MCI, which instead of appearing as "**£[**" like in the message editor, it will appear as a "." then F4 (an uppercase reverse 'J').

3.12 Text Editing Features

Here are some miscellaneous features available while editing or entering text in the BBS text editor.

- Uppercase/Lowercase Toggle
  While in the text editor, you can use CTRL/L and CTRL/U to switch to lowercase and uppercase, respectively. CTRL/N also works the same as CTRL/L.

- Recorded Delete Key
  The left arrow (←) symbol works as a 'recorded delete' in the message editor. Instead of deleting text from memory, this character will print a standard Commodore Graphics delete character. This has the effect of pulling text to the left when the cursor is positioned left of the text.

- Word Delete
  CTRL/W is a word delete and can be used when entering a line of input. Beware, though, this command will also delete all the proceeding spaces positioned after the word. For

example, if you had typed "Hello " with the three spaces after the word, and then hit CTRL/W, then the three spaces and the word would be deleted.

- Line Delete
  CTRL/X is line delete and will delete all the text you have typed so far on a single line. You can use this if you find it necessary to retype your line.

- Line Reprint
  CTRL/V is a line reprint command and will re-print everything you have typed so far after printing a carriage return. This function is not as useful locally as it is remotely. A user who is experiencing line noise can use this feature to verify what they have typed so far if noise is encountered while entering text.

- Centering
  Use CTRL/C to center the text you have just typed (before hitting the RETURN key). Centering the text will strip out all inserts and recorded deletes. A carriage return is not added, so you can still add to the line after centering it. Centering will not function if input does not begin at the far-left column of the screen.

- Dynamic Delete Key
  Characters are deleted "by type" when you use the DEL key. This means, for example, that if you delete a CRSR-UP then the cursor will move down. Or if you were to delete a color character, then the color you were using previously will be activated. This is very helpful for accurately editing your text.

- Message Editor Memory
  The text editor works in a way to account for not only the number of free lines, but it accounts for free memory as well. A function allows the message editor to keep constant track of the free memory available. If the memory runs out, then the same thing occurs as if the number of available lines was used up. This means you do not have to worry about the maximum number of lines that was chosen in SETUP, so you could set it to 200 lines if you wished. You can choose what the minimum amount of free memory is from SETUP.

- MCI Toggle
  An option appears at the line editor prompt to allow you to turn MCI commands on or off when viewing lines of the message being edited. This is convenient for when many commands have been used and you wish to edit or remove them.

- Message Separator Function
  The "**/@**" command is a helpful command that you can use while editing a mail message. This command will put a message divider character (a CTRL/N on a line by itself) at that point in your message. The message divider is used by the private mail and network routines to separate individual messages. Any caller who has access to the "Edit Any Message" parameter in SETUP can use this command. If a caller attempts to type a CTRL/N

on a line all by itself, the ML input routine will automatically change it into two (and therefore will not corrupt a private message or node message). The **/@** command is the only way this character can be added (if you delete a line with the character in it, you will have to use the Insert command and retype the **/@** command). You should only use this command if you are familiar with the format of private and Network messages.

3.13 Customizable Message Headers

The [**R**]Read Public Messages routine allows you to customize the message headers. When the caller activates the Read Messages command, the program will load one of two files into memory. If the caller is currently using 40 columns, then "√headers" is used. If the caller is currently using 80 columns, then "√headers80" is used. Here is the format of the file:

Line 1: The message header top. This may be a line that divides off the message header, or it may be left blank.
Line 2: The FROM information line.
Line 3: The DATE information line.
Line 4: The SUBJ information line.
Line 5: The NODE information line.
Line 6: The CITY information line.
Line 7: The message header bottom. This will be the last thing printed after the header information.

In each of the header lines, you may designate where the actual variable information (such as the subject of the message) will be printed with the following Variable MCI: **£[i$]**. This prints the contents of **i$**, which will hold the variable information for each line as it is printed.

If you wish the line to be followed by a Carriage Return (as usual), then each line must end in a CTRL/Y. CTRL/Y, as explained in the Programming instructions, is a substitute for the RETURN character, but it won't end a line in the message editor. This allows you to add a Carriage Return to the end of each line and have the Public Message routine read it in from the file as part of the line. If you don't include a CTRL/Y character on a line, then the next line of the header will be printed on the same line.

For an example of the format of the files, examine the two "√headers" files included. For the example headers you must set the 40-column header limit (as defined in SETUP) to 30, and the 80-column header limit to 70.

If the program fails to find a file on disk, it will revert to what was previously used, or if you don't use these custom files at all, then the default message headers will be used (see below).

When the header information (the actual public message header) is read in, it is stripped of all graphics control characters. This is used in conjunction with the two 'width' parameters in SETUP

to limit the width of the information. This makes the use of the MCI Tab command effective in creating a border around the message header (again, see the included files for an example of this).

3.13.1 The Default Message Headers

Here is what the default Color 64 message headers would look like if they were in the file. **<rvs-on>, <rvs-off>, <space>,** and **<ctrl/y>** are representative of what to type:

Line 1: blank [would be just a RETURN in the message editor]
Line 2: <rvs-on>From:<rvs-off><space>£[i$]<ctrl/y>
Line 3: <rvs-on>Date:<rvs-off><space>£[i$]<ctrl/y>
Line 4: <rvs-on>Subj:<rvs-off><space>£[i$]<ctrl/y>
Line 5: <rvs-on>Node:<rvs-off><space>£[i$]<ctrl/y>
Line 6: <rvs-on>City:<rvs-off><space>£[i$]<ctrl/y>
Line 7: blank [would be just a RETURN in the message editor]

Below is a scenario using the above setup. In this example, we will assume that the caller is using 40-columns and that the 40-Column Header Limit is set to 20 characters. If the information in the public message was:

From: ANTHONY TOLLE (#3)
Date: 02/04/92 - 3:09 pm
Subj: (R)The reason for the delays

Then the output would be:

```
From: ANTHONY TOLLE (#3)
  Date: 02/04/92 - 3:09 pm
  Subj: (R)The reason for th
```

Not much difference, eh? Well, the included files are lot fancier. This was just a demonstration of how the √headers files will be interpreted, especially how the information is inserted into the header and how the length is controlled (note that only the length of **i$** is checked, not the length of the entire line).

Also note that the NODE and CITY information lines are not printed in the case that the message is not a network message. Also, the CITY line will not be printed if the sending node did not include it in their outgoing messages.

3.14 The Crash Routine

If you answered "Y" to the question "Rerun on errors" question in SETUP, then another routine has also been enabled that will automatically keep track of where errors occur. If the program crashes, then a line will be put in the caller log in the following format:

<error><line no.>:<overlay name>

For example, if a syntax error occurred at line 12000 in the overlay √bbs.init, the message would appear as ***"syntax12000:√bbs.ini***". Note that the STOP key is disabled when "Rerun on errors" is enabled to prevent an accidental stopping of the BASIC program.

3.14.1 Stopping the Program

There are times when it may be desirable to "break" the BBS program, even though the STOP key is disabled. Maybe you have a programming error and need to examine variables, maybe your modem locked up and the program is waiting for it to clear, etc. If you ever get into this situation, just press, and hold the SHIFT, COMMODORE, and CONTROL keys all at once. If this does not seem to work, try hitting the STOP key again (sometimes hitting SHIFT/COMMODORE/CONTROL will re-enable the STOP key). If this does not help then possibly you have experienced some type of power or hardware hardware problem, locking up your computer.

3.14.2 Once the Program is Stopped

If the BBS program is ever stopped for any reason, then there are a few procedures that should be followed. First, you can look at the contents of different BASIC variables using the PRINT command, but you should never under any circumstance edit the BASIC program currently in memory. This would cause all BASIC variables to be lost (including the message index information) and would necessitate a full BBS reboot.

If all variables are intact, then you can type ***"GOTO9991"*** and press RETURN to cause the system to save the message index, save the variables, and update the caller log. Once this shutdown operation is completed then it is safe to load another BASIC program and edit it, or from this point you can type RUN and press RETURN to restart the system because the √bbs.init program will be in memory.

If you use GOTO9991 to shut down, then the error message put in the caller log will always be "BREAK", even if the system crashed on an error. The only way for the exact error message to be included in the caller log is if the "Rerun on errors" option is on and the BBS automatically restarts the system.

3.15 Graphics Modes

When a caller first logs on, the system determines what type of general terminal they are using with the "Backspace/DEL" test. Commodore graphics terminals send a CHR$(20) code when the caller hits the DEL key, while ANSI and ASCII callers will send a CHR$(8) or CHR$(127) code when they hit the backspace or delete keys.

If they are using ANSI or ASCII, the system then asks the caller if their terminal supports ANSI graphics. If they answer "Y", then the computer will make full use of ANSI color and graphics characters. If they answer "N" then they will also be asked if their terminal requires line feeds to work correctly. This is for ASCII terminals that may already attach a linefeed to each carriage return received.

- Commodore Graphics Mode
  Commodore graphics terminal programs can get the full effect of all the various color changes and graphics characters used in the BBS system. This is primarily the reason this BBS system was named Color 64, because it was one of the first to fully support the new line of Commodore graphics term programs such as CCGMS.

  Color 64 supports the background color change that was standardized in the early term programs. A background color change is achieved though the CTRL/B character; if a color is typed after a CTRL/B character, then the background will change to that color. This does not function for Commodore 128 80 column term programs, because CTRL/B is used to turn on underlining. Color 64 will allow you to print files specifically designed for the 80 column C128 screen because it will "pass through" the control characters used for underlining and flashing modes.

- ASCII Mode
  Color 64 will attempt the best translation possible when the caller is calling using a plain ASCII terminal program, although use of these programs is decreasing because of the greater availability of color terminal programs for Commodore and non-Commodore computer callers.

  All the Commodore graphics characters will be converted to a suitable substitution, and Color 64 will even consider the current uppercase/lowercase mode of the text screen.

- ANSI Graphics Mode
  ANSI callers should be quite satisfied with the conversion made by Color 64. Color 64 supports 15 ANSI colors (dark grey does not work on some ANSI terms, so is translated to medium grey), and makes the best possible conversion of Commodore graphics characters.

  ANSI callers should set their terminal so that the default background color is black, to adequately emulate Commodore graphics. Also, the term should be set that a carriage

return will not be appended after a linefeed, because the line feed character is used as the "cursor down" character for ANSI callers.

Since many ANSI terms only support 8 different background colors, the other 8 "bold" colors may not be active in REVERSE mode, thus some graphics translations may not be as satisfactory as possible.

One other input option is available to ANSI callers. If they type the ESCape character and follow it with two digits, then they can change the current cursor color. The first digit is the bold setting, 0 for normal colors or 1 for the brighter "bold" colors. The second digit is the ANSI color from 0 to 7. This way they can access all the possible colors that Commodore graphics callers can use.

3.16 User Terminal Settings

If a caller types the **[!]Edit User Stats** command from the main prompt, they will be asked three questions regarding the way their terminal is set up.

- Page-Pauser Lines
  This is the setting for the screen height of the current caller. If the caller is reading a particularly long message, this setting will automatically pause after the number of lines indicated here.

- 40/80 Column Setting
  This option allows the caller to have Color 64 take advantage of 80-column word wrapping in the text editor, and the special 80 column modifications made to Color 64.

- Character Delay
  Some callers may complain to you about garbled text information; this may be caused by line noise in some situations. In some cases, however, especially if you are using a Lt. Kernal or CMD w/RamLink, it may be caused by the fact that the output speed of Color 64 is too fast for their terminal program to keep track of. In this case, they can increase the character delay number until the output problems disappear. Also, if it is a line noise problem, changing this setting will sometimes alleviate the trouble.

  When a new user signs on this option defaults to 7, which is just right for 2400 bps callers. Callers signing on at higher BPS rates may not find the speed fast enough however, so you may want to display a general note about this character delay feature (perhaps in the user application) so that all users can adjust this setting to suit their own terminal program.

3.17 Fast Garbage Collect Routine

One of the specialized routines in the Color 64 ML is a "fast garbage collection" feature. Garbage collection is the term used to describe what the computer does when it cleans memory of garbage strings that BASIC is no longer using. In standard C64 BASIC the delay during garbage collection can be quite long, which decreases the performance of programs. The Color 64 fast routine decreases the amount of time that this collection takes.

Some of the Color 64 BASIC routines that specifically take advantage of this feature are the regenerate message index and the directory regenerate routines. You will notice that the screen goes blank sometimes while these routines are working. This is because a blank screen will allow the computer to work faster. Also, SYSOPs with Commodore 128 computers will have a bonus: the garbage collection routine will work at 2 megahertz.

Note that on a stock Color 64 system the fast routine is not always active. If you want to have the routine on all the time, allowing faster operation of the BBS system, then you should edit line 10241 of √bbs.init and change the part that says "M7=." to read "M7=1". You will notice the screen blanking more often, but this just shows that the fast routine is active.

For more information on this feature, see the descriptions of the !48 and !53 variables in section 5.1.7.

Chapter 4, Network 64 Instructions

Before you dive into Network, you should read through this entire chapter first.

IMPORTANT NOTICE TO NEW COLOR 64 BBS OWNERS! If you just recently started running Color 64 BBS (or haven't even started yet), do not use Network! You should wait until you've become thoroughly familiar with Color 64 BBS before you get into Network. If you are new to Color 64 BBS and you start using Network, you'll be well over your head in hot water regardless of how well you think you can handle it.

Also, don't use Network unless you already know who you want to network with. This is because you must have at least one node entered in through the NET SETUP program before you can even start the system up if Network is enabled.

4.1 Hardware Requirements

It is required that you have a fast means of loading overlays within the BBS environment. There are only three I know of which can accomplish this speed:

- The Commodore REU series
- The Xetec Lt. Kernel hard drive system, and
- The CMD hard drive system with either JiffyDos or RAMLink

This requirement is not only for your own benefit, but also for those of your nodes. There is also the real possibility that if you are not using one of these methods, the network will not work properly since there are precise timing routines within the system that expect certain things to happen at certain times.

Aside from the above requirements, Network 64 is fully compatible with all hardware which is supported by Color 64 BBS.

4.2 Brief Summary of Required Files

Table 17 below provides a summary of all the required and optional network related files which are included with your Color 64 system. Refer to the documentation for specific details of what each does.

Table 17 - Summary of Required Files for Network 64

Files	Description
** Required Files **	
√bbs.nw1, √bbs.nw2	These are the two main network program modules and should be in the Program Files.
√sys.net	This is the NET SETUP program, like BBS SETUP. It allows you to define drives and specify individual node data.
prscrn52750	This file must be in the Boot Files with Net Setup. It allows you to do screen dumps to your printer while inside the program. (**)
√rlog, √slog	These are the receive and send logs for Network file transfers. (*)
bck to bill	This is a stand-alone program which will create a new billing file from a backup. It would be located with your Boot Files.
bbu.nw2	This is an optional merge file which will install an automatic billing backup feature into your midnight routine.
√node app	This sequential file is a file you edit. It's used when calling a new node for the first time to introduce yourself to the new node. The name of your BBS will automatically be inserted at the beginning of this message when it applies to a new node. (*)
√conditions	This is a sequential file you create. It is used when a node calls you for the first time. This file describes your validation requirements for the new node. (*)
√temp xref	This optional file's purpose is to be able to assign nodes new ID numbers without them having to change their IDs. It is created and edited manually. (*)
* Must ALWAYS remain on your NETWORK drive. ** Must ALWAYS remain on the Boot Files drive.	
** Other Files ** These files may be self-generated and appear on your drive as they are needed	
√node x users	A membership list of all users on your BBS who have access to Network and is created every night (midnight). It is provided to remote nodes so that they have a membership list of your system.
√node *[#]* users	*[#] = Node Number* - This is the corresponding membership list for any one node you may have. This file is received at the time you send data and are ready for a new listing.
√+node *[#]*	A packet of actual messages which you or a user has posted to node number specified by *[#]*.
√+file *[#]*	A file containing information on another file(s) to be sent to node number specified by *[#]*.
√node ledger	Shows account node transactions. This will remain on your drive until moved or scratched.
√public storage	Holds messages you have released from received Node public message.
√node list	List provided when sending to a Node. Created by Net Setup program.
√ntwrk.parms	Contains details from Setup on how network is to be run. Also contains data for each Node. Created by the Net Setup program.
√node accounts	Relative file containing name, password, level, and last date called for each incoming Node you have.
√node billing	Relative file containing dollar amount for each user on system

4.3 Installation

The following actions are what is needed to prepare your system for Network:

1) Boot the normal BBS SETUP program
2) Select “Main Parameters” (option 1) to edit
3) Near the end of the parameter list, answer "Y" to the question asking if you want to run Network
4) Go to Disk Drive Assignments (option 2) and verify you have the proper drive selected for the Network Files
5) Go to BBS Commands (option 7) and verify Network-specific commands are set to desired user level

4.3.1 Post Network Msg

This is where you or your users create the network messages. Usually, it should be safe to allow all but the new users into this section.

4.3.2 Net Maint Menu

This is a sub-menu within the network that will allow you to do certain maintenance either on-line or off. This section has options for editing/viewing the billing file, the node account file, and the node status file. It is also where you select files to be transmitted. Much more on these options later. This should be restricted to only the top staff on your BBS system.

4.3.3 Release Publics

This is an ideal area for a co-sysop. This is the feature that allows you to release public messages which have been stored from the network. This subject will be covered in detail later in the documentation.

4.3.4 Restrict Posts

This is not a command; This feature allows you to define a limit to the number of network posts a user can make. If you choose not to charge for any or all nodes, you might not want people sending out too much in one call. The first number is the level exemption. If a user’s level is equal or greater than this number, they would be allowed to post network messages to their hearts content. The second number is the maximum number of posts a user can make per call (unless that user has a level which exempts him/her).

4.4 Network Setup

Now comes the real thing; we're going to really start setting up the network! Locate the file called "+net setup" in your Boot Files. Make sure the file called "prscrn52750" is located on the same drive. The "+net setup" program works just like the other boot files that you use with your system, except that it loads the "√sys.net" (NET SETUP) program.

Once the program is loaded, you may be asked to insert your Program disk into the drive. Be sure that your parms have been changed as instructed in section 4.3, Installation.

IMPORTANT! NEVER UNDER ANY CIRCUMSTANCES USE YOUR DELETE OR INSERT KEYS WHILE IN NETWORK SETUP OR THE BBS SETUP PROGRAM. IT WILL RUIN YOUR DATA ALMOST EVERY TIME! IF YOU NEED TO EDIT SOMETHING, ALWAYS USE YOUR CURSOR LEFT OR CURSOR RIGHT KEYS AND TYPE OVER WHAT YOU ARE EDITING.

1. Number of Nodes
   After a few seconds you're asked for the number of nodes you would like to have. This is important! Count the number of nodes you have in mind and enter this number. You are being asked this question because you will then be asked to fill in data for each one of these nodes, so if you have no info on any nodes, don't run this program until you do. If you need some nodes, look at the enclosed node listing and choose from this list if you like. You may have from 1 to 99 nodes. The higher number you use, the more memory that is required.

2. Public Message Category
   This question serves two purposes. If you aren't familiar with how the network works, then you should know that with Network you can send private E-Mail or public messages. If someone sends you a public message, this question will determine how that public message is handled. You have two choices: You can have the message stored in a special holding file, and then release it manually, or you can have the message automatically go into the message base. If you want the message to go directly into the message base, enter the category letter you wish the message to go into. You can type a question mark to get a list of your categories at this point. If you would prefer to hold the messages, and release them manually, enter a 0 (zero) now.

3. Open and Close Times
   These two questions require a little explaining. When a user sends a network message on your BBS, the file is stored on your disk. The file doesn’t get sent out until you want it to. The next two questions tell the network what time of day you would like messages to be sent. You will be asked to enter the time in military time (i.e. 0=midnight, 9=9am, 22=10pm – see Table 17 below). This time limit is called a window. If there are messages due out, they will only go out inside this window time (there is an exception to this.... read on). There are two restrictions you need to keep in mind. First, you cannot open the window at a time later than you close the window (this should be obvious). This is the basis of the second restriction: You cannot have the window open through midnight. In other words, you can

open it at midnight, but you couldn't open it at 11pm, and close it at 3am. When you open and close depends largely on what time zone you are in, and when your phone rates are the lowest (most are 11pm-8am). I recommend midnight to 6AM, but it's entirely up to you. The network dials out only after 5 minutes of inactivity. If a caller doesn't call within 5 minutes, and we are in the open window zone, and there is network mail due out, the network will attempt to dial out. If a connection is not made, the cycle will continue after another 5 minutes of inactivity. You can always change the window times, so a little experimentation will determine what's right for you. Also note that the window closes at the exact moment when the current time reaches the close time (i.e. a close time of 6 am means the window will close AT 6:00 am exactly).

Enter your open and closed times now.

Table 18 - Military Time Cross-Reference

Hour Standard	Hour Military
00-11 AM	00-11
12 PM	12
1 PM	13
2 PM	14
3 PM	15
4 PM	16
5 PM	17
6 PM	18
7 PM	19
8 PM	20
9 PM	21
10 PM	22
11 PM	23

4. Days Request Membership List
   This question deals with the special membership list. At midnight every night, your BBS will create a special network membership list (called √node x users on your disk). When you send out net messages to your nodes, they may request a fresh listing from you. This is the file they will get. This value determines how long you would like to wait before requesting a new membership list when you call a node. Mine is set for 20 days. The smaller the number, the more requests your network will make, and the longer you will be on-line.

5. Your BBS Name
   This question is asking you for your BBS name. This name will appear on ALL headers of all messages sent out. The maximum length is 25 characters.

6. Does Modem Support BUSY and NO DIALTONE?
   This question is a tad complicated to explain. If your modem can return the BUSY or NO DIALTONE response codes (almost all 2400 modems and some 1200s do), you MAY want to answer "y" to this question. Check your modem manual under the command ATX; it should tell you there. The reason I said you MAY want to answer yes is because you don't have to even if your modem does support these status responses. Let me try and explain the purpose. Say that when the BBS dials out to a node and the phone just rings and rings and doesn't ever answer, you MAY prefer that from now on that node be locked out from being called. There is a built-in feature in the network which will lock a node when certain conditions arise, and one of the conditions is called a CARRIER LOCK. This means that the BBS attempted to dial out to a node, but your modem returned a NO CARRIER response code. With a modem that supports the BUSY and NO DIALTONE responses, then you should never be getting a NO CARRIER response, UNLESS of course the remote node's modem isn't picking up, the node has crashed, or the node is down. In the case of one of these conditions, the line will ring and ring and the modem will return NO CARRIER. Thus, you could answer "y" to this question and the computer will lock out any node that returns NO CARRIER, because something is probably wrong with that node. But the node will NOT get locked out if your modem returns a BUSY or a NO DIALTONE response code, because either the node is busy or there was a problem dialing. If you answer "n" to this question, then the modem will not get locked by ANY of either the NO CARRIER, BUSY, or NO DIALTONE codes, either because your modem doesn't support BUSY or NO DIALTONE, or because you prefer that nodes do not get locked out because of the NO CARRIER response.

   I answer NO to this question even though my modem does support these response codes. The reason being, if the BBS calls out to a node that is resetting either after a caller, or at midnight, you will get the NO CARRIER, and the node would be locked out. I didn't want that, but it's entirely up to you how you select this. Remember, if your modem does not support these response codes, then in a way you're lucky as you have no difficult choices to make here; You must answer NO.

7. File Release Directory
   This question asks you what it should do with programs received through the Network. You may choose to have files placed in a U/D directory or have them placed on your Network drive. If you choose to have the files in a U/D directory, that directory will automatically be updated when files are received (i.e. it works just like multi-upload).

8. Hold Files for Release?
   Lastly, you will be asked if you want files that are received through the Network to be held for your release, or if you want them to be available right away. This works just like normal file transfers and the 'auto-release level' in SETUP. If you have already decided that files received will go on your Network drive, this option will have no effect.

4.5 NET SETUP - The Node Editor

After you have completed the main Network questions section, the real fun begins! If everything has gone smoothly so far, you should be at the individual node editor prompt. At this prompt, you're given the option to edit any of the individual nodes. Also, the number you see within the double brackets [], tells you the last NODE NUMBER you edited. It may be hard to understand now why you'd need this, but believe me, it helps to determine where you are in the editing process. Also, at this prompt, if you hit a question mark (?), you will get a listing of each NODE NAME, and its status. This is especially handy for finding a blank spot, or when trying to locate a certain node to edit.

Before we start editing individual nodes, let's clarify one thing: When you assign a node a certain number, this will be called your NODE NUMBER, and is used to identify a node when users send outgoing messages to that node. The remote node will never see this number, because it is for your own (or your user's) use only, to choose a node to send messages to. When you set a node's ID number, this is the NODE'S ID NUMBER, and is somewhat like the node's own password number on your system. In other words, it is used by your system to recognize when the remote BBS is calling. This has been a little confusion in the past, so I thought I'd clarify it before we went on. Also remember that INCOMING and OUTGOING are two separate things. When you use NET SETUP, you're setting up for OUTGOING calls. When you use the Node Editor (explained later), you're setting up for INCOMING calls.

Now it's time to edit our first node. Enter the number 1 (NODE NUMBER ONE), and hit RETURN.

- Node Name
  The first prompt asks you for the name of the node. You are limited to 25 characters. The name you enter here will appear to all users when they go to post a network message. If you wanted to replace this node with another, simply enter the new node's name.

- Node Membership ID Number
  Now you need to enter the membership ID number for the node which you are working on. There are two ways to get this ID number. One way is by requesting that the Sysop of this node set you up on his or her system. He or she will then give you an ID number and password. An easier way is to let the network do an equivalent of a NEW USER process. If you know your ID number, enter it now, otherwise enter a 1 at this point. The number 1 means that you will be calling this node for the very first time. After you call the node, it will automatically issue you an ID number, and the network will enter it into this slot automatically.

- City and State
  Next is the city and state of the node. Note that you MUST have a slash SOMEWHERE in this line. Don't use a comma (i.e. use Berkeley/CA and NOT Berkely,CA). This question is 100% cosmetics.

- Phone Number

Now for the phone number. Enter any special access codes or whatever you'll need. For example, put a "t" in front of the phone number if you want the modem to use TONE dial instead of PULSE dial.

- Password
  Password time. If you were assigned a password, enter it now. If you answer the ID prompt with a 1, then make up the password you would like. Remember, 3-9 characters only. Don't use any exotic characters either as they will be stripped out. You can enter in lower or uppercase. Any lowercase characters will automatically be changed to uppercase for you.

- Baud Rate
  Now for the baud rate. You are allowed to enter the BPS rates listed. If you are not sure what rate this node is using, you should enter the highest rate supported by your system. Your modem will recognize the baud rate of the node, and the network will automatically step it down to the proper baud rate.

The following two questions determine how much (if any) you plan on charging to allow messages to be sent. You can set up a different rate for each node. They can all be totally free if you choose. If you are not going to charge for this node, enter 0.00 at the next two prompts.

- First 1000 Bytes
  The first charge prompt determines how much you will charge for the first 1000 bytes or less. Say you wanted to charge 25 cents for the first 1000 bytes (most will be well under 1000 bytes). Make sure that you enter it as 0.25 and not 25. or 25.00.

- Each Additional 100 Bytes
  The second charge prompt is for each additional 100 bytes after the initial 1000 bytes. You can leave either of these two prompts zero if you like. Say you wanted to allow the first 1000 bytes free, then only charge for any message after 1000 bytes. You could do that by leaving the 1000-byte question 0.00. With a little trial and error, you can set this up to break even when your phone bill comes in.

  Speaking of phone charges, in case you're not familiar with AT&T's Reach Out America plan, you might want to inquire about it. You sign up once, and thereafter, at certain times of the day, you are allowed to call anywhere in the USA for something like $7.50 an hour. This is something like 60 average node calls for that amount (most node calls are under 60 seconds!). For more info, call AT&T at 800-222-0300.

Now answer "y" if the screen is properly filled out. At this point, one of two things could happen. If you set this node up with an ID number of 1, you will have a little bit of disk activity. What is happening, exactly, is that the program is copying your √node app file into the √+node x file (x being the node number). The name of your BBS will be automatically inserted at the beginning of the file, because the first line in the node application is used as the name of your BBS on the remote node. If you wish to override this feature and tell the remote node a different name than what you have

entered in the Net Setup, then you can put the name at the beginning of the √node app file, preceding it with two CTRL/N characters.

The other thing that might happen is if you entered a BBS name other than the one which was there before (replacing one with another). You will be asked if you are indeed replacing one node with another. If you answer yes to this, the old node's files will be scratched.

You have set up your first node! It's tricky, yes, but you'll get the hang of it. When you've set up all your nodes, simply hit RETURN at the select node number prompt, and all the required files will be created. Just for reference, these files are: √ntwrk.parms, √node list, √node accounts, √node billing. Note that the last two files are only created if they do not exist already.

You can always alter any of the info in this file, but obviously, you want to use a little judgement. If you make changes to the drive setup, don't forget to move the files which may exist on the old drive! Things like that. Unfortunately, you will always have to take the BBS down to make any changes, so do what I do and write down all the changes you need to make over the course of a few days, so that you can tackle a bunch of stuff at once instead of shutting down and booting up repeatedly.

One very important word of caution: Never ever edit either the √ntwrk.parms or √node.list file with a message editor! If you do, your computer, your drives, and your modem will explode! Just kidding, but still... Don't do it. Your BBS will probably crash left and right, you'll be sending net mail to the pizza parlor down the block, and generally creating havoc for yourself and quite possibly your nodes.

4.6 Booting the BBS with Network

Now that you have finished with the setup (it wasn't that bad, was it?), it's time to boot up the BBS. If you are running your system using an REU, then your "√sys.ramove" program will automatically copy your Network overlays from the Program Files

Regardless of which way you boot up, make certain that the two main modules (√bbs.nw1 & √bbs.nw2) are on the Program Files drive. If you use RAMDOS, make sure that these files are getting stashed into RAM at the time RAM is getting loaded. Leaving out these two main modules in a ramdos system is by far the number one thing people forget to do! These modules must get loaded in at the time all the other files are getting loaded in. You can't get the bbs booted all the way without these files being in RAM.

RAM users please make sure you have sufficient space available! This is the second most common problem.

4.6.1 The Wait-For-Call Screen

Boot up your BBS now. When you get to the familiar call waiting screen, you might want to take note of a couple items:

- There are two "status line" values which were not previously updated. They are labeled Nets Holding and Nets Due Out.
    - Nets Holding shows you how many (if any) public messages which have come in from the network, are waiting to be released. If you are not using this holding feature as described in section 5, then this number will always be 0.
    - Nets Due Out shows you how many calls are due to be made. It does NOT show you how many messages are due out; it only shows you the number of nodes which are due to be called. If a node is locked for some reason (read on for info on locking nodes), it will show this node as due to go out until it tries to go out, in which case it will adjust itself. More on this later...

Ok, now hit any one of your function keys (F1-F8) and review the menu again. The CHANGE TIME/DATE function is gone! Don't worry, it's still around. Instead, at F4, you'll see what's called NETWORK MENU - Hit F4 now. Now you are at the Network menu. Everything on this menu will be explained in detail.

4.7 The Network Menu

The first thing you see is a free memory check. This displays how many bytes you have free in memory. It's impossible to say where this should be, but if it gets down to 3000 or so, I'd say you're running a loaded BBS (lots of modifications and unregulated games) and run the risk of an out of memory error. You may need to reduce the number of nodes through the net setup program if this gets too low, because the more nodes you have the more memory is used.

Whenever you choose any of these 8 options, you should see a quick flash on your screen. This is just to confirm that your keypress was acknowledged since some options may take a second or two to access.

- F1: Change Date/Time
  There it is! This does the same exact thing it did when it was on the Sysop Menu.

- F2: Change Window
  This is an option to change your window times (read the Network SETUP section). This option will remain in effect till either you change it again here, your BBS crashes, or you re-start it. It is a temporary change only. It's easy to use, and pretty much self-explanatory. From this option, you may also turn the volume on on your modem while dialing out. This can be helpful if you are getting a lot of NO CARRIERs.

- F3: Maintenance

This accesses yet another menu. The difference between the maintenance menu and the one you are looking at now is that Maintenance is accessible via this menu and accessible on-line as well. We'll go into this a bit later.

- F4: Multi-Send
  This is a feature which allows you (only yourself since it is not available on-line) to send the same message or file to more than one node at a time without having to visit the message maker each time. The first thing you need to do is to create a NET DEFAULT MESSAGE or a DEFAULT FILE. This is different than your normal BBS default message. It can only be created in the POST NETWORK MSG section. A DEFAULT FILE is created in the ATTACH A FILE section. After you have created this default net msg or file, you can then use this feature. To use, simply enter the NODE NUMBERS you wish to send to separated by commas (just like in a multi-download). If you want to send to every node you have, enter the word "all".

  After you make your selection, individual copies will be made. When you're done, you have the option of saving this default (indefinitely or until you create a new one). Note: If installed, the mail verification option is not available when sending out multi-messages. See the paragraph on the Mail Verification mod in the "Miscellaneous Options/Features" section for more information.

- F5: Release publics
  This is where you go to release any public messages which may be held. Once again, you may not need this feature. This is another easy to use, self-explanatory option. You can re-read a message, hold it, delete it, or release it into your message base (all pointers will be set properly). This option is also available on-line.

- F6: Regenerate Network Index
  This can be used anytime you suspect something is screwy. If your Nets Holding or Nets Due Out status values are incorrect, select this option, and the index will be regenerated quickly. This is also used in case you wanted to delete a node message or file directly in the BBS DOS (node messages begin with the prefix √+node, node files begin with the prefix √+file). The network index is automatically regenerated when you boot up the BBS, and every midnight, so things should pretty much stay in good shape.

- F7: Post Network Msg
  This is where you go to enter the message to any node. An awful lot goes on here, so this will be a lengthy description. This option is available online as well.

  When any user first accesses this section, their account balance will be shown. If you are at level 9, this process is bypassed. Also currently, free space is checked on your HOLDING drive. The "reserve" free space used is the value you defined in the BBS SETUP under the free space allowance for uploads.

Next, you are at the prompt which asks you which node you want to select. If you are at a level 9, you will see an additional message reminding you how to send a default net message. If you enter "***?***", you will get a listing of all participating nodes along with their rates (if any). When you enter the number of the node, you will be given the name of the BBS. Answer "***Y***" if it is correct.

Now you are asked whether the message is private or public. If you choose public, then you will be asked for the subject, and will be dumped into the message maker. If you answer "private", then you have a few options:

- You can at this point look at this nodes membership list by entering "**?**". You will of course not have a list until you send your first message.
- You can enter the user's name or number. Depending upon how far down the list the user is, this can take some time as each line needs to be scanned sequentially. Entering a number is slightly faster than using a name. Entering "SYSOP" will bypass the search, and the message will go to user #2 (the sysop).

By now you should be in the message maker. Everything is exactly as it is when you are in the BBS message maker. You can use the C= key to merge files just as before. When you go to save the message, it will not be saved right away. You will first be shown the length of the message (in bytes). If the message costs more than you have in your account, you will be told so, and will not be allowed to send it unless either you shorten it, or abort.

After you can send the message, your account will be adjusted (unless the node was free or you are at level 9), and you will be shown your new balance. If you decide to use the E-mail verification mode (see the paragraph in "Miscellaneous Options/Features" for information on merging it in), you will also be asked if you would like E-Mail verification of when the message is sent. If you answer "**Y**" to this question, when the message is sent out to the node, a note will be put in your mailbox showing you the time and date the message went out.

At this time, a file called √node ledger is appended to (or created). This "mini-log" contains all transactions. This file can be used to reconstruct a billing file if needed. It will get larger and larger as time goes on (unless all your nodes are free). If a node is free, nothing gets put into this file.

- F8: Return to BBS
  This brings you back to the call waiting screen.

4.8 Network Maintenance

Now for the Maintenance section. The first few options are easy to figure out. They all deal with the billing file.

- Option 1: Billing List/Print
  It does just what it says. It allows you to list or print out the any part of the billing file.

- Option 2: Billing Edit
  This allows you to edit any account.

- Option 3: Report Generator
  This requires a little explanation. This feature lets you look at the billing file in a few flexible ways. If you want to see any records which have something in them, you will enter either "**>0**" or "**<>0**". If you want to see records which have less than $1 in them, you can enter "**<1.00**" or "**<1**".

- Option 4: Total Accounts
  This does just what it says. It will go through the entire billing file and tally up the grand total amount.

- Option 5: Node Status
  This option allows you to LOCK or UNLOCK a node and you will use this option often. When locking a node, if there are any messages, they will not go out under any circumstances. Additionally, you will not be allowed to post a message to that node. LOCKing a node is done manually. You will first be shown all the nodes with their stats, and after the listing if you hit **RETURN**, you'll go back to the menu. If you enter a node number, you'll be given a chance to Lock or Unlock one or more nodes.

  There are two other ways a file can become locked: ACCESS and CARRIER, which are done automatically.
  - ACCESS means that the node was called, and you were denied access for some reason.
  - CARRIER is when you have selected the special response code setting in Net Setup. This means that you called a node, and your modem returned a NO CARRIER (indicating that the node did not pick up, or no answer, or voice).

- Option 6: Node Editor
  This is where you go to edit your incoming node accounts. Everything here is straight forward. See section 4.10, Incoming Node Accounts, for more information on exactly what you'll need to do at this editor.

- Option 7: Attach a File to Send
  This option allows you to send a PRG or SEQ file to another node. The node must be running Network 1.26 or 1.26a for this to work. If you try to send a file to a Network 1.24 node, your BBS will discover this fact and hang up.

  Steps to utilize this option are as follows:

- Enter the device number of the file you want to send. From here, you may choose to enter the '**>**' character instead, taking you to a Mini-DOS. This Mini-DOS only lets you switch drives, read directories, and view SEQ files. Enter **RETURN** to exit.
- Enter the drive number, drive command, and file name as it appears on your disk.
- Enter the node number you wish to send the file to. Enter the number or enter '**?**' for a list of nodes. You may also choose to enter '**x**' for a default file send.
- The BBS will query if you would like the file to be deleted after it is sent.
- You are then prompted for the file name for the receiving node. If you press only **RETURN,** it will use the same name as the file is on your disk.
- If a file description is found for the file you are sending, you will be asked if you would like to have it sent also.

A file named "√+file x", where 'x' is the node number, will appear on your Network drive. This file contains all the information that you just entered. The BBS does not limit the number of files that you may send; however, you should not exceed 20 files at a time. For v1.26 nodes, there is an 18-minute limit on transfer time. At 2400 baud, about 450 blocks can be transferred in 18 minutes. A v1.26 node will not be able to receive files larger than 450 blocks. If you or the v1.26 node only runs 1200 baud, limit the file size to about 250 blocks. If both you and the remote node are running v1.26a (Color 64 version 8 includes v1.26a), then there is no limit placed on the size of the files, although you should limit file sizes to 1000 blocks or less for practical reasons. It is not required that you send a message with the file, but on the other hand, you may send one. Messages and files will be sent during the same call.

- Options 8 and 9: Read Receive and Send Logs
  These options will display the "√slog" and "√rlog" files. These files provide a more detailed description of file transfers. After viewing one of these logs, you will be asked if you want to clear it. It's best to clear these files after you read them or they will just continue to grow.

4.9 Incoming Node Accounts

When a node calls in to your BBS, it will send an ID number and a password, just like the normal BBS login. If the password doesn't match the assigned password, no access will be provided. Additionally, if the node calling in is new, your network will send an ID number to the node, and the node will send you its desired password. After the call is made, the node will store the ID which was issued to it by your network, and on your end, you will store the node's password in your account file. All subsequent calls from this node will go through normal channels.

Like the BBS, there is an access level for nodes. When a new node calls in for the first time, it is given an access level of 0. This means that any node with an access level of 0 will not even be allowed access. If this new node called you again and you did not yet validate it, it would have wasted a call because it wouldn't be allowed in. The access number must be greather than 0 to gain access. There is a lot involved with node access levels. More on this very shortly.

4.9.1 Editing Node Accounts

Currently there is only one way to edit an account, from the Network Maintenance menu. All accounts are stored in a relative file called "√node accounts". In the "√node accounts" file, there is room for four separate fields.

- Field #1 is the NODE NAME. The name is limited to 25 characters. Don't worry about counting the length, as the editor will only take the first 25 characters. Incidentally, this name is for your own use only. Nobody else will ever see it but you (or a co-sysop who has access to the maintenance section).
- Field #2 is the password. This is limited to 9 characters.
- Field #3 is the level field. The only valid numbers allowed here are "0" to "101". More on the levels in a second.
- Field #4 is the last date the node called in. This field's only purpose is for you to see how often nodes are calling, so you could perhaps decide when it might be time to delete them. I guess you could call this a manual purge.

You can delete a node by typing "**delete**" at the NAME input.

4.9.2 Node Access Levels

The reason we're saving the access level for last, is because admittedly, it's somewhat confusing. It's not complicated, just confusing. There are three different modes of access levels. Mode one would be unvalidated. This is indicated by having a 0 (zero) in the level field. With a 0, an incoming node would be denied access. When a node calls in as new, their level will automatically be set to zero. Mode two would be a simple unreplyable, validated node. Here is the key. You may have a node set up to call you, but for some reason, you may not want to be calling them. Or, to put it another way, if you don't want any incoming messages from this node to be replyable, give this incoming node an access level of 1. Got that so far? Level 0 is totally unvalidated, level 1 is validated, but replies are not allowed to anything they send to you or your users.

Now, remember that discussion we had about NODE NUMBERS? If not, go back to paragraph in the "Net Setup" section and re-read it, because it's important. Let's say you have 3 nodes, ok? When you go to post a message to a node, and you hit "**?**" for a listing of the available nodes, let's say you get a list that looks like this:

```
1: THE ABC BBS $0.00 $0.00
     Brooklyn/NY

2: THE DEF BBS $0.00 $0.00
     Chicago/IL

3: THE GHI BBS $0.00 $0.00
     Los Angeles/CA
```

Now according to this list, "THE DEF BBS" is node number 2. Got it? It doesn't matter where in your node accounts file this BBS is located, and we're not talking about NODE ID's. We're talking about the NODE NUMBER.

Ok, back to levels. Let's say you're in the node editor and the node account number 1 was "THE GHI BBS", because it may have been the first one that applied to your system. You want to be able to reply to any messages this BBS sends to you. Here's the important part: At the access level, you would enter 3 PLUS 1 (4) as the access level.

TO MAKE ANY NODE REPLYABLE, GIVE THEM AN ACCESS LEVEL OF 1 PLUS THE NODE NUMBER AS IT APPEARS ON YOUR NODE LISTING. IF YOU WANT NODE NUMBER 24 TO BE REPLYABLE, YOU WOULD ENTER 25 AS THE ACCESS LEVEL IN THE NODE ACCOUNT; IF YOU DID NOT WANT TO BE ABLE TO REPLY TO A NODE ACCOUNT, YOU WOULD ENTER A 1.

I realize this is a little awkward, but I did not want to add an extra field just for this purpose. If you inadvertently entered an incorrect access level, the reply would go to the wrong node.

Network 1.26 has added some abilities to help you along here. Entering '**?**' at the access level prompt will show you a list of outgoing nodes. Enter the node number from the list and it will link the incoming node to it and make it replyable. After entering the level, the BBS will tell you what outgoing node it is linked to.

4.10 Miscellaneous Options/Features

Here are some other optional features that you might want to take advantage of if you are running Network:

- Network Activity
  One thing you should know is that most of the Network activity appears in the caller log in blue. This should allow you to easily distinguish Network activity from regular BBS activity.

- Network Mail Verification

Included with the system are two merges you can use to install the Network Mail Verification mod. This mod will allow users to have a message put in their mailbox verifying when their outgoing net mail was sent. The user will be asked if they want their net mail verified after they type their message. The two files are called "vnm.nw1" and "vnm.nw2" and should be merged into √bbs.nw1 and √bbs.nw2, respectively.

- Billing Backup
  If you are not running all your nodes as free, you may want the added comfort of knowing that you will not lose track of anyone's money, by installing an automatic midnight backup feature of the billing file. It's very simple. Just merge in the file called "bbu.nw1" into √bbs.nw1. The program called "bck to bill" will restore the backup file if you need to do so.

- Force Outgoing Send
  There is an option at the call waiting screen which allows you to manually force a call out to a node. This will only work if you have messages to be sent out. If a node is locked, or there are no messages due out, nothing will happen. All you need to do is to hold down the CTRL key for about two seconds. This will allow you to send out messages OUTSIDE of the normal window time.

- Undeliverable Network Mail
  When a node calls in, and sends E-Mail, before that E-Mail is delivered, the name on the header of the E-Mail will first be checked against your password file to be sure that the user still exists. If not, or if for any reason the mail is undeliverable as written, it will end up in YOUR mailbox with a note attached to it showing you where it was supposed to go. You can then direct it properly.

- Network Membership Lists
  When setting up a new node, you won't have its membership list till after you send the first message. Until you get the membership list, you will only be allowed to send either public messages or E-Mail to Sysop only. So even if you have already pre-arranged access on the remote system, you will still have to send at least one public message, or a private message to SYSOP.

- The Network Transfer Timeout
  There is an IRQ timer (an IRQ program operates in the "background" monitoring computer activity) in Color 64 which is active whenever the network is on-line. This keeps track of the on-line time. If after 5 minutes it still detects a carrier, it will force the modem to disconnect. This will ensure that nodes will not and cannot ever become "locked" together for any reason. For file transfers, the timer setting is calculated based on the size of the file.

- The Off-Hook Mod
  The OFF-HOOK mod has been incorporated into Network 1.26. This way, users or other nodes won't be trying to call in when the BBS is distributing messages or updating a U/D directory. To use Off-Hook, you need to have at least a 2400 BPS Hayes-compatible modem

and have DTR disabled. To do this, answer NO to DTR in √bbs.setup, and add an 'h0' to the end of your modem init command. Next, boot up a term program, set it to your maximum baud rate, and send the following command to your modem: **ATZ&D0&W**

If you leave DTR enabled, the Off-Hook commands will have no effect on your modem.

4.11 Troubleshooting

This troubleshooting section is far from complete. Table 19 below covers the most common problems that new Network SYSOPs have.

Table 19 - Network Troubleshooting FAQ

Question	Answer
After I boot up and answer the "regenerate message index" question, I get a FILE NOT FOUND error and the BBS attempts to reboot.	This means that a main module is missing or cannot be found. Make sure that the two main modules, √bbs.nw1 and √bbs.nw2 are located on your PROGRAMS drive. If your Program drive is in RAM, there are two things to watch for. One, make sure that these two modules did in fact get loaded into RAM properly. For RAMDOS users, the file called "√sys.ramove" is the means of storing these modules into RAM. Refer to "Booting the BBS with Network" for more info. The second problem may be free memory in RAM. A 1764 (with 256K ram) might be pressed for free space. The two modules may be named improperly. The first eight characters must be named √bbs.nw1 and √bbs.nw2 for the BBS to find these modules.
When someone sends a public or private message and I reply to it, instead of replying through the network, it tries to reply locally on the BBS.	Don't feel bad as this happens a lot. You need to brush up on the section of setting access levels for node accounts. Remember that you can have an incoming node be replyable or NON-replyable.
When booting up my BBS, half way into the start of it, I get a crash with "?file data error".	What this most likely means is that either your √ntwrk.parms file or your √node list file got trashed. The number one reason for this happening, is because you used your delete key while editing in the Net Setup program. Review the section on using the Net Setup program. If you determine that your file is indeed trashed, and you never made a backup, you probably won't be able to salvage the file, but you may be able to get some valuable information from it, by reading the file with the BBS DOS f: feature, and jotting down any recognizable data, then scratching the file and starting anew.

Chapter 5, Programming with Color 64

Color 64 was written to be easily modified. Several things have been designed into the system to make it easier for the programmer to modify the BBS system:

- The Color 64 ML adds many powerful commands and functions to BASIC's vocabulary, so that code can be designed as efficiently as possible.
- The overlays were designed uniformly, so that all overlays have the same basic "skeleton" of essential routines.
- The overlays were left "open ended", which means that you can easily install one of the many pre-written modules into your overlays.

5.1 The Enhanced BASIC Language

Color 64 offers a set of powerful new additions to the BASIC programming language. These additions only work while the BBS program is running, as it is a modification of BASIC made by the ML associated with the BBS program.

The descriptions of the new commands are presented in the following format: Items enclosed in angle brackets, "**<**" and "**>**", are descriptions of data items that you must provide. Items enclosed in square brackets, "**[**" and "**]**", are optional items.

- The New Load Command
  The Color 64 ML adds a new LOAD command that is very much like the original BASIC LOAD command, except it adds a couple new features. The format for the new command is:

  ↑<filename>, <device> [, <load address>]

  <filename>: the name of the file to be loaded (in Color 64, its common to use dr$+"filename").
  <device>: the device number.
  <load address>: the load address at which you wish the program to load. If you do not include <load address>, then the computer assumes that you are loading a BASIC program. This means that the program will auto-run when it loads. If you use a load address less than 256, then the computer assumes you are loading a file directly into memory at the address specified in the file (either an ML file or memory table). This type of load will not cause the BASIC program in memory to auto-run. If the load address is greater than 255 then the file is loaded directly into memory at the specified address.

  Again, loading files without a loading address will load and runs BASIC files. Loading files with a loading address does not affect BASIC (unless of course a program is loading directly into the program or variable memory sections). Here are a few examples of the command:

.dr$+"√bbs.ini*",8	**|** This will load "√bbs.ini*" from device 8, drive dr$ and RUN it:
.dr$+".tab.ans*",8,0	**|** This will load the memory table "√bbs.ans*" from device 8, drive dr$ into memory at the location specified in the file
.dr$+".some file",8,58000	**|** This will load the program file ".some file" into memory from device 8, drive dr$ into memory at address 58000

Another feature of the new LOAD command is that it checks if a BASIC program has overflowed memory and overwritten variables. If this does occur, then the BBS program automatically shuts down and displays an error message indicating a LOAD overflow. All variables will be lost, and the message index will have to be regenerated. This error occurs when a BASIC overlay is bigger than √bbs.init, and it would overwrite variable memory -- often wreaking havoc on the BBS program when gone unchecked.

- The Output Commands

The Color 64 output commands differ from the BASIC PRINT statement because they not only print the information to the screen, but they also print the output to the user that is online. If you use a regular PRINT statement while a user is online, then the text will only be printed on your own screen. Another thing that the output commands do is automatically convert the output to ASCII or ANSI, if the user online is using one of those two text modes.

The "#" command is the one that is closest to BASIC's own PRINT statement.
Its format is:

 # [<expression list>]

<expression list>: A single or list of expressions that may be separated by semicolons (just like the PRINT statement). If you do not include anything after the "#" command, then the contents of A$ will be printed. This shortcut is often used in routines where A$ is read in from disk or from the keyboard and must be used if you wish to put the information in the caller log as well. The "#" command will print a carriage return at the end of the line unless you put a ";" semicolon at the end of the command.

1000 #"Hello, "na$", you have level"lv	**|** Example of the # command Note that numeric variables will always have a space before them if they are positive, or a negative sign if they are negative.

The "$" command works much the same as the "#" command, but it will never print a carriage return at the end of the line. This can be used to save a byte because you won't have to include a semicolon as in the following example:

1000 $"Enter your name >"

The "%" and "&" commands are substitutes for the "#" and "$" commands, respectively. Their difference is that they will automatically adjust alphabetic text (A through Z) so that it will always be readable in the case that Uppercase/Graphics mode has been turned on. If the computer is in Uppercase/Graphics mode and the text line to be printed contains shifted letters, then the line will "unshift" them so that they will be letters (and not Commodore graphics characters) when printed. These commands are used in the text editor routines, where the uppercase and lowercase modes can be changed at will.

A warning about using the single character commands: If the text to be printed is also intended to be put in the caller log, then you must make sure that you use the A$=<text>:# version. Otherwise, the wrong information may be put in the caller log because A$ will never be changed to the correct string. The above commands do not alter A$. This same warning also applies to any other situation where the information to be printed may be used elsewhere.

- The CTRL/Y Character
  A special control character has been added to the output routines that allows you to print carriage returns without using the CR$ and C2$ variables. CTRL/Y behaves exactly like a carriage return, except it can be “recorded". This means that in BASIC you can type it inside quotes, and on the BBS, you type it without ending the current input line. CTRL/Y does not cancel the velocity MCI command (£v) either, which enables you to type a long message without having the carriage return cancel the velocity command. Please note that CTRL/Y CANNOT replace CR$ when data is being printed to disk files (data such as a message header or system parms). The only times CTRL/Y should be used is if you know that the output is going to be printed only to the screen, and not read in from a file as data. The reason for this is that CTRL/Y is not recognized by BASIC's INPUT routine as a valid end-of-line character. An exception to this restriction is if the CTRL/Y is intended to be read in as part of the data, in which case it would still have to be followed by a normal carriage return. See the section on customizable message headers for an example of this.

- 5.1.4 Direct Modem Output
  Two other commands are intended to print data directly to the modem and are used when "AT" commands are to be sent, and when information needs to be transmitted "unformatted" by the regular output commands. The CTRL/Y character does NOT work with these two commands:

  The " **'** " apostrophe command follows the same format as the '#' command. This command outputs the text directly to the modem, without doing any conversion and without printing anything to the screen. A carriage return will be printed at the end of the output line, unless a ";" semicolon is included at the end of the command.

  The " **(** " open parenthesis command works much the same as the " **'** " command, except that a carriage return will never be printed at the end of the output line.

- The Enhanced IF/THEN Statement
  The Color 64 ML adds a powerful set of decision-making commands that supplement the IF/THEN statement:

  a. The ELSE Statement
     The first of the added commands is the ELSE statement. Used in conjunction with IF/THEN, ELSE is used to test if the expression is FALSE rather than TRUE. The ELSE statement is represented by the British Pound symbol '£'. Here is the format of the '£' statement:

     IF <expression> THEN <statements>: £ <statements>

     | If the <expression> is TRUE, then the statements following the THEN are executed and everything after the '£' symbol is ignored. If the <expression> is false, then the THEN section is skipped and the statements following the '£' are executed. See next example:

     IF A=5 THEN PRINT "Hello": £ PRINT "Goodbye"

     | If A were to equal 5, then "Hello" would be printed. If A were to equal 5, then "Hello" would be printed. If A were not 5 then "Goodbye" would be printed.

  b. The AND-ELSE Statement

     A variation of the ELSE statement is the AND-ELSE statement. This is represented by the characters '**!£**'. It behaves the same as the ELSE statement, except that if the expression was TRUE, then the statements after the '**!£**' are executed ALSO (hence both the first AND the second sections are executed, or ELSE just the second section is executed). Here is an example:

     IF A=5 THEN PRINT "Hello":!£ PRINT "Goodbye"

     | If A were to equal 5 then BOTH "Hello" AND "Goodbye" would be printed, but if the statement were false then only "Goodbye" would be printed. This would be the same as having another line of programming on the same line, because whatever is after the '**!£**' is going to be executed regardless of the outcome of the decision.

  c. 5.1.5.3 The REDECIDE Statement

     Another addition to BASIC is the REDECIDE statement. It is represented by either the characters '**!+**' or '**!-**'. This statement uses the result of the most recent IF/THEN

statement to decide. The !+ statement will execute only if the last IF/THEN expression was TRUE. The **!-** statement will execute only if the last IF/THEN expression was FALSE (hence '+' for TRUE and '-' for FALSE). Here is an example:

```
10 IF A=5 THEN PRINT "Hello, ";
20 !+PRINT "How Are You?"
30 !- PRINT "What is Your Name?"
```

| If A was to equal 5, then both the first and second statements would be executed, which would print "Hello, How Are You?"
If A was not 5 then only the REDECIDE-FALSE (the !-) statement would be executed, which would print "What is Your Name?"

You can also use ELSE with the REDECIDE statements. ELSE will execute if the opposite of the REDECIDE statement is true. Here is an example:

```
10 IF A=5 THEN PRINT"4 ";:. PRINT"1 ";
20 !+ PRINT"3 ";:. PRINT"2 ";
30 !- PRINT"3 ";:. PRINT"2 ";
40 !+ PRINT"1":. PRINT"4"
```

| If A was to equal 5, then "4 3 2 1" would be printed. If A was not 5 then "1 2 3 4" would be printed. Note that the execution of a REDECIDE statement, or the alteration of a variable that was in the original IF/THEN statement, does NOT change the true or false nature of the most recent IF/THEN.

d. Combination IF/THEN Statements

You can use more than one IF/THEN/ELSE statement on a line (or any combination of the commands listed above, for that matter), but note that if an IF/THEN expression is false, the BASIC interpreter with look for the first '£' symbol that occurs after the expression. Thus, unless IF/THEN statements on a line should execute the same ELSE statement, the IF/THEN's should be matched up to the appropriate ELSE's. Here is an example:

```
IF A=1 THEN C=1: IF B=1 THEN C=2 : £ C=3
```

| The results from this line would be as follows:

A=0 , B=0 -> C=3
A=0 , B=1 -> C=3
A=1 , B=0 -> C=3
A=1 , B=1 -> C=2

Compare the following line and its respective results to the one above:

IF A=1 THEN C=1 : £ IF B=1 THEN C=2 : £ C=3

| Results for this line would be:

A=0 , B=0 -> C=3
A=0 , B=1 -> C=2
A=1 , B=0 -> C=1
A=1 , B=1 -> C=1

As you can see, the presence or absence of just one '£' can make a big difference to the outcome of an IF/THEN/ELSE statement! Plan out the use of the IF/THEN/ELSE statement carefully and consider all the possible outcomes. By doing this, you will ensure that the proper decisions are being made.

These commands are advanced, and you don't really need to use them unless you wish to trim the size of the code or make a more efficient routine. Do not use them unless you are familiar with their operation.

- The ML Command Set

The ML commands are a way for BASIC to access the faster and more powerful subroutines built into the Color 64 ML. Some of these commands do complex things and are very useful (such as getting a line of typed input), while others aren't used very often (enabling and disabling interrupts).

All ML commands begin with " **.** " period character, followed by two digits. Some commands require additional parameters, which follow the command after a " **,** " comma. Here is an example of using an ML Command:

1000 .01:ifleft$(tx$,1)<>"*"then1000

| In this example, the **.01** command reads a line of input in from disk, then if the input line did not begin with an asterisk, it would loop back and read another line. This is also an example of how some of the input commands use **TX$** as a buffer for input.

Table 20 provides a summarized listing of all the ML commands:

Table 20 - ML Command Set

Cmd	Details
.00	**Get a typed character.** This command does not wait for a key to be pressed but looks to see if a character has been typed. Input is accepted from the BBS system keyboard or from the user currently online. The ASCII value of the character is returned in **!02**, and the status is returned in **!01**. The ASCII value will be 0 (nu$) if no character has been typed. All alphabetic characters are converted to uppercase (A through Z). **Example: .00:P=!01:A$=CHR$(!02)**

Cmd	Details
.01	**Input a line of data from disk.** The file number in **!14** used. End of line (EOL) is reached if: • Maximum number of characters (length of **TX$**) is read in or • The character stored in **!04** is reached (usually set to a carriage return), unless **!15** is set to non-zero, then **!04** will not end input. Data is stored in **TX$** starting at the first character. The number of characters input is stored in **!40**. If EOL was reached at the **!04** character, such character will not be included in **TX$** (e.g. a carriage return will not be part of the input line). If **!40** is less than the length of **TX$**, then the line was terminated with the **!04** character or the end of file was reached before another **!04** character. You can access the information by using **TX$**, the **@0** function, or you can use the **@5** function if you want to assign disk data directly to a variable (and skip the **.01** command). **Example: .01:SR=ST:A$=@0**
.02	**Input a line of text from the user.** Input is accepted from the BBS system keyboard or from the user currently online. The maximum number of characters to input is the length of **TX$**. Input is terminated with a carriage return, or by pressing CTRL/X or CTRL/P. The status is returned in **!01**. The number of characters typed is returned in **!40**. The information is stored in **TX$** starting at the first character. If **!40** is less than the length of **TX$**, then the line was terminated with a carriage return. Otherwise, input was ended because the maximum number of characters was reached. You can access the text by using **TX$**, the **@0** function, or you can use the **@8** function to skip the **.02** command. **Example: .02:I$=@0**
.03	**Equivalent of the "$" command**... ... but the expression to be printed must follow the command after a " **,** " comma. Do not use this command in any programming, as its redundancy will probably earn its removal from the next version (i.e. its function will be replaced with another command).
.04	**Activate protected mode.** RUN/STOP will be disabled, and the program can only be stopped by holding down SHIFT/COMMODORE/CONTROL. If the system crashes, a **GOTO9991** will be initiated. Line 9991 should contain the code to execute after a system crash. The **.05** command disables protected mode.
.05	**Deactivates protected mode.** See the .04 command for more information.
.06	**Equivalent of the "#" command**... ... but the expression to be printed must follow the command after a " **,** " comma. Do not use this command for programming; see the description of the **.03** command for more information.
.07	**Activate terminal mode.** Terminal mode (or Term mode) is a basic terminal program which allows the Sysop to dial out to other systems. Term mode will end if the Sysop presses any of the function keys (**F1 through F8**). The ASCII value of the function key pressed is return in **!02**. The Plusterm program makes use of the built-in buffer functions (see **.23, !27, !28, !29,** and **!30**). Local mode (**!12**) must not be on, and DTR must be enabled.
.08	**Equivalent of the "%" command**...

Cmd	Details
	... but the expression to be printed must follow the command after a " **,** " comma. Do not use this command for programming; see the description of the **.03** command for more information.
.09	**Equivalent of the "&" command**... ... but the expression to be printed must follow the command after a " **,** " comma. Do not use this command for programming; see the description of the **.03** command for more information.
.10	**Activate carrier-detect checking interrupt.** This activates an interrupt (a program which is executed every 1/60 of a second), which performs the functions necessary to check the status of the modem's carrier detect line. It also handles the Network file transfer timeout, the carrier detect timeout, and the inactivity timeout. Related variables are **!13**, **!22**, **!00**, **!11**, **!24**, and **!25**. The **.11** command disables the interrupt.
.11	**Deactivate carrier interrupt.** See the **.10** command for more information.
.12	**Output a sequential file.** The file number is stored in **!14** and the file must already be opened. If **!16** is set to 1, the file cannot be aborted. If **!17** is set to non-zero, then the page-pauser will be active and **!17** is the number of lines. The file can be aborted by pressing the space bar or by typing CTRL/P.
.13	**Dump caller log buffer to disk.** This will dump the temporary caller log buffer to disk file number 98 and clear the buffer. This command is intended to be used only by the normal BBS caller log routines, and the system could crash if this command is used outside of this routine.
.14	**Set level parameters for MCI commands.** This command sets the levels for the use of the MCI commands. The system uses the value of **LV** for the current user's level, which is why **LV** must be one of the first variables defined. The command is in the format **.14,<*number1*>,<*number2*>** where <*number1*> is the level for the message MCI's (£a0, £a1, etc.), and <*number2*> is the level for the variable MCI command (£[). If <*number1*> has 128 added to it, then the **DD$** message MCI (£a6) is disabled (i.e. will not print anything).
.15	**Set the "Chat Begin" text.** Allows you to customize the text printed when chat mode begins. The format of the command is **.15,<*string*>** where <*string*> is a character string containing your custom message. The chat mode routine does not print any clear-screens or do any preliminary output before printing this text, so you will have to include these control characters yourself. The length of the string is limited to 40 characters, and any extra will be trimmed off the end. **Example: .15,"<clear>-Chat Begin-"+c2$**
.16	**Set the "Chat End" text.** Like the **.15** command, but it sets the message for when chat mode ends. After this text is printed, a CTRL/P is issued to abort any current output.
.17	**Put text into caller log buffer.** This command is in the form **.17,<*string*>**, where <*string*> is the text to be put in the temporary caller log buffer (usually the string is **LG$**). This command is used in conjunction with the **!20** variable and the **.13** command to handle caller log functions.

Cmd	Details
	This command is intended to be used only by the normal BBS caller log routines, and the system could crash if this command is used outside of these routines.
.18	**Set up variable-killer.** This command causes the current variable memory configuration to be "memorized" by the system. Later, when you use the **.19** (Activate variable-killer) command, this information will be used to kill all new variables added to memory since the **.18** command was used. If you use more than one **.18** before another **.19**, then the memory configurations will be "stacked" so that an **.18**/**.19** pair can be "nested" inside another **.18**/**.19** pair. This allows a subroutine to use temporary variables and discard them without disrupting another routine's temporary variables. Up to eight **.18** commands can be stacked, and an error will result if a 9th consecutive **.18** command is issued.
.19	**Activate variable-killer.** Kills all the variables added since an **.18** command. See the **.18** command description for information on how the variable killer works.
.20	Set BPS rate for computer to modem communications. This command is in the format ***.20,<number>*** where <*number*> is a value from 0 to 2 for non-SwiftLink systems or 0 to 7 for SwiftLink systems. The values correspond to the BPS rates 300, 1200, 2400, 4800, 9600, 14400, 19200, 38400 respectively. For non-SwiftLink systems, the BPS rate will be set to 2400 if any value above 2 is used. For SwiftLink systems, the BPS rate will be set to 38400 if a value above 7 is used. Once a BPS rate is set when a user is online, it should not be changed until the user disconnects.
.21	**Activate DTR (Data Terminal Ready).** DTR is a line to the modem which indicates the readiness of the computer to send and receive data. If DTR is active, then all input/output functions are normal. This is used in conjunction with **.22** to control the DTR line.
.22	**Deactivate DTR.** Turning off DTR can have varying effects. The most favorable effect is to cause the modem to hang up if on-line and return to command state (ready for AT commands). In this case, it is best to have the DTR question in Setup set to YES. Otherwise, the modem may just ignore the DTR signal, reset, or even something else depending on how the modem is set up. 99% of the time, though, the **.22** command has a beneficial effect (there have not been any major problems). Since disabling DTR on a SwiftLink system also disables all input and output, the DTR line is just switched off momentarily.
.23	**Dump buffer contents.** This function dumps the contents of the Term mode buffer to file number 3. The Sysop can press the space bar or CTRL/P to abort if output is directed to the screen. See for more information. This command is used by the Plusterm program.
.24	**Wait for end of modem transmit.** On non-SwiftLink systems, there is an output buffer which stores characters waiting to be sent over the modem. This command will wait for the buffer to be completely emptied, thus ensuring that no data will be lost if some other type of modem command is to be issued. SwiftLink systems don't use an output buffer.
.25	**Cancel modem output.** Unlike the above command, which waits for the output buffer to clear, this will cancel the output immediately and clear the buffer.
.26	**Clear input buffer.**

Cmd	Details
	Will clear all data holding in the modem input buffer. Ensures a clean slate for another input command.
.27	**Receive a file using current protocol.** Used by the BBS file transfer routines.
.28	**Transmit a file using current protocol.** Used by the BBS file transfer routines.
.29	**Receive a header block (multi punter).** Used by the BBS file transfer routines.
.30	**Transmit a header block (multi punter).** Used by the BBS file transfer routines.
.31	**Does nothing** Used by the module converter to fill in for commands that are no longer used.
.32	**Set DATA statement line number.** Acts as a RESTORE command but allows you to set the line from which DATA statements will be read. If used by itself, the DATA pointer will be set to the current line. If used in the form ***.32,<number>*** where <*number*> is a line number, then the DATA pointer will be set to that line. **Example: .32,20000**
.33	**Activate extra variable memory mode.** Most of the time, system memory is set up so that BASIC variables always begin at the same place while the BBS program is running. This place is located after the end of the last byte of the "√bbs.init" overlay, which allows other overlays to be loaded without overwriting variable memory. The **.33** command will cause the variables section to be moved to after the last byte of the current overlay in memory. This means that if the current program is 10000 bytes smaller than "√bbs.init", there will be 10000 more bytes of free variable memory opened up after the **.33** command. This command should be used in conjunction with the variable-killer commands, because the **.34** command will close this extra space and if there is not enough space free, a lot of system information could be overwritten. See the **.34** command for more information.
.34	**Deactivate extra variable memory mode.** This closes the extra memory opened by the **.33** command. A **.18** should precede every **.33** command, and a **.19** should precede every **.34** command. This will ensure that memory is returned to its original state. If the **.34** command is not issued before the "√bbs.init" overlay is loaded, then a LOAD OVERFLOW error will occur. If there is not enough space free before the **.34** command is issued, then string variables will be overwritten.
.35	Set system rainbow mode color list. This command sets the colors that the system will use for the standard rainbow colors (i.e. the ones used by the F1 and F5 characters). The format of the command is ***.35,<string>*** where <*string*> is a string of 8 color control characters.

- The ML Variables
  The ML Variables are a way for BASIC to get certain information about the BBS environment, as well as to define the way the BBS operates. Thus, you can read the information in these variables, as well as assign values to them.

  All ML variables begin with an "**!**" exclamation point, followed by two digits. They can be used in expressions just like BASIC variables. Assigning a value to an ML variable follows the same format as the POKE command; the format is "**!XX,value**". The range of possible values for all ML variables is 0 to 255. If the program tries to set an ML variable to a value outside of this range, then an error will result. Some variables are READ ONLY, as will be indicated in the following listing. If an attempt is made to write to one of these variables, then an error will result. And finally, some variables have multiple elements and are addressed like BASIC arrays, and an error will result if an index outside of the allowable range is used. Here is an example of using an ML variable:

  1000 !04,0:.01:!04,13:if!40<5then1000

  What this does is first set **!04** to 0, where **!04** is the end-of-line character used when reading in from the disk. Then the **.01** command reads in a disk line. The second **!04** then sets the End of Line character back to the normal carriage return character. Then the line tests !40, which is the number of characters read in during the last disk input, to see if at least 5 characters were read in. If not, then the loop repeats.

  Table 21 below summarizes the ML Variables:

Table 21 - ML Variable Summary

Var	Description
!00	**Carrier detect mode.** If this is less than 128, then the regular carrier detect timeout is enabled (i.e. the timeout timer will run). Otherwise, the carrier detect timeout is disabled. See the **!11** variable for information on the carrier timeout.
!01	**Status variable.** Contains the status after input commands like **.00** and **.02** (also the **@8** function). The values are as follows: 0 = OK, 1 = aborted by CTRL/P, 3 = pause by CTRL/S, 4 = aborted by CTRL/X, and 255 = carrier lost.
!02	**General input/output character.** Contains the character returned by commands like the **.00** and **.07** commands. Any other time, this will contain random data.
!03	**Maximum column number for word wrap.** When the user is typing in a line and the cursor goes beyond this column number, a word wrap will engage if word wrap mode is on. A word that is wrapped (up to 15 characters) is placed in a special word wrap buffer and will be "typed in" by the computer at the next input.
!04	**End of line character.**

Var	Description
	The **.01** command or the **@5** function will read in a line of data until this character is reached, unless the maximum line length is reached, or **!15** is set. See also **.01** and **!15**.
!05	**Translation mode.** This is used to set the translation mode of the BBS. If it is set to 0, then the current mode is ASCII translation. When in ASCII mode, the system uses the most basic set of characters for input and output and performs the necessary conversion from ASCII to PET ASCII and vice versa. Setting this variable to 1 means Graphics mode (either Commodore or ANSI) will be used. This mode takes advantage of all color and graphics on the system. Setting this variable to 2 means simulated ASCII mode (i.e. if in Graphics mode, any input will be limited to alphabetic, numeric, and common ASCII symbol characters). Setting this variable to 2 will allow you to keep Graphics users from entering color control characters and special graphics. See the **!18** variable for information on ANSI mode.
!06	**Case constraints on input.** If this is set to a non-zero value, then all alphabetic characters entered (when **.02** or **@8** is used) will be converted to uppercase.
!07	**Word wrap mode.** If this is not zero, word wrap mode is engaged for line input. See **!03**
!08	Reserved for future use.
!09	**Character output mask.** If not 0, this character will be printed instead of what is typed when inputting a line (the **.02** command or the **@8** function). This is used when the password is entered (asterisks are printed).
!10	**Line feed mode.** This is active only for ASCII translation mode. If this is not zero, then a line feed (ASCII 10) will be printed after each carriage return (ASCII 13). See the **!05** variable for information on the translation mode.
!11	**Carrier timeout flag.** If this is set to zero, then the carrier detect timeout timer will be on hold. If set to 1, then the timer will count to 255 if carrier is not detected. Since the carrier is checked every 1/60 of a second, a timeout will occur after approximately 4.25 seconds of no carrier. This variable will contain 255 if a timeout has occurred.
!12	**Local mode.** Setting this to 1 will disable all modem input and output. This allows the BBS to be used in local mode without characters being sent to the modem. Setting this to 0 allows modem input/output to continue.
!13	**Inactivity timeout flag.** If this is not zero, then it means that a timeout occurred because of inactivity on the part of the user. A user gets approximately 2 minutes before a timeout will occur.
!14	**File number for disk input.** This used for many disk commands and functions including **.01**, **@5**, and **.12**.
!15	**End of line mode.** If this variable is not zero then the disk input routine will ignore the **!04** end of line character and will fill the input buffer until the end of the file is reached, or until the maximum number of characters (length of **TX$**) has been reached.
!16	**Seq file abort disable.**

Var	Description
	If this is not zero, then the next sequential file read (with the .12 command) will not be abortable by either the spacebar or CTRL/P keys. After the file is read, this will be reset to zero.
!17	**Number of pager pauser lines.** If this is zero, then the page-pauser will not be active. If not zero, then this value is used during a sequential file read (the **.12** command) as the number of lines for the page-pauser. For example, if **!17** is set to 15, then the BBS will pause and wait for a character after every 15 lines printed when the **.12** command is used.
!18	**ANSI mode.** This is only active for Graphics translation mode. If this is set to a non-zero value, then ANSI translation mode is enabled. See the **!05** variable for information on the translation mode. When in ANSI mode, the system acts just as if the system is running in Commodore Graphics mode, except that all output is converted to the appropriate ANSI codes.
!19	**MCI disable mode.** If this is not zero, then MCI's will be disabled for one line of output. This is always reset to 0 after every output command.
!20	**Caller log buffer status flag.** This read-only variable indicates the status of the caller log temporary buffer. If it is zero, then the buffer is empty. If it is 1, then the buffer has some contents in it. If it is greater than 1, then the buffer is full and needs to be dumped to disk. This is for use only be the regular caller log routines.
!21	**Variable-killer stack number.** This read-only variable indicates the current level of "nested" variable kills. If no **.18** commands are currently in effect, then this will be set to 0. As each **.18** command is issued, then this will increase by one.
!22	**Network file transfer timeout timer.** When a network file transfer is initiated, this is to be set to 128 plus the number of minutes for the timeout timer. The timer will start as soon as the variable is set. Setting this variable to zero turns off the timer. If the timer runs for the full length of time, then the current file transfer will be aborted.
!23	**Non-SwiftLink BPS timer table.** This table of 18 bytes holds the timer values used by the Non-SwiftLink systems when sending and receiving the individual bits of data. There are 6 bytes for each BPS rate (300, 1200, and 2400 respectively), and of the 6 bytes there are 2 bytes each for the output bit time, input bit time, and half-bit time, respectively. Each of the two bytes is a 16-bit value arranged in low byte, high byte order. Thus, the table is as follows: 300 BPS output bit time, 300 BPS input bit time, 300 BPS half-bit time, 1200 BPS output bit time, etc. To access any individual byte of the table, just index **!23** like an array (i.e. ***!23(0)*** would be byte 0 of the table).
!24	**Carrier type.** Holds the value which is compared to the computer's hardware register to see if the carrier is detected. This value is determined by Setup when you set the carrier type.
!25	**Carrier status.** This read-only variable will be non-zero if carrier is detected, and zero if there is no carrier.
!26	**SwiftLink/Non-SwiftLink flag.**

Var	Description
	Read-only variable; If this is 0, then the system is a Non-SwiftLink system. If it is 1, then the system is using the SwiftLink cartridge
!27	**Term buffer mode.** If this is not zero, then all characters printed to the screen will also be put into the term buffer. The parameters of the buffers are established by the **!28**, **!29**, and **!30** variables. Output is diverted to file number 3 (normally screen output). This is used by the Plusterm program.
!28	**Term buffer pointer.** This is a 16-bit address in low byte, high byte format which points to the next open character in the term buffer. **!28(0)** is the low byte, and **!28(1)** is the high byte. This is used by the Plusterm program.
!29	**Term buffer bottom pointer.** This is a 16-bit address in low byte, high byte format which is the address of the first byte of the term buffer. This is used by the Plusterm program.
!30	**Term buffer top pointer.** This is a 16-bit address in low byte, high byte format which is the address of the last possible byte of the term buffer (i.e. the buffer pointer cannot go beyond this point). This is used by the Plusterm program.
!31	**Current BPS rate.** This read-only value in the range 0 to 7 indicates the current BPS rate of computer to modem communication. Possible values are: 0 = 300, 1 = 1200, 2 = 2400, 3 = 4800, 4 = 9600, 5 = 14400, 6 = 19200, 7 = 38400.
!32	**Input buffer empty flag.** If this read-only variable is 0, then the modem input buffer is currently empty. Otherwise, there are still characters waiting in the modem input buffer.
!33	**X-Modem file conversion mode.** If this is not 0, then the file being transferred will be converted from standard ASCII to PET ASCII.
!34	**Number of tries for X-Modem CRC mode.** This sets the number of times that the system will attempt to engage CRC (Cyclical Redundancy Check) at the beginning of an X-Modem file transfer.
!35	**Punter block size.** This sets the number of bytes for the Punter file transfer blocks.
!36	**File type for transfer.** This value is the file type of the current file transfer. If it is 1 or 3, then the file is a PRG file. If it is 2, then the file is a SEQ file.
!37	**Current protocol type.** The current protocols set this read-only value to 0 for Punter, and 1 for X-Modem.
!38	**File transfer timeout flag.** If this is not zero at the end of a transfer, then it means a carrier-detect timeout occurred during the transfer. Also, it could mean a Network file transfer timeout.
!39	**End of transfer status.** If this is not zero at the end of the transfer, it means the transfer was aborted.
!40	**Number of input characters.** For keyboard input (**.02** or **@8**) or disk input (**.01** or **@5**), this value is the number of characters read in during input.

Var	Description
!41	**Alternate output file mode.** If this is not zero, then screen output for the BBS is diverted to file number 3 (usually the screen or the printer). Only the regular BBS output commands are affected, so the BASIC PRINT command will not be affected.
!42	**Day of the month.** This read-only variable will contain the day of the month calculated from the most recent **@16** function. See the **@16** function for more information.
!43	**Month of year.** This read-only variable will contain the month of the year calculated from the most recent **@16** function. See the **@16** function for more information.
!44	**PM/AM Flag.** When the **@11** function is used to get the time, this read-only variable will be 0 if the time is currently AM, otherwise the time is PM.
!45	**Current time hour.** When the **@11** function is used to get the time, this read-only variable will contain the hour (1 to 12). See the **@11** function for more information.
!46	Not used
!47	**Character output delay.** This sets the delay between characters to allow slower systems to keep up with the output of Color 64. A value of 0 means no delay, while 255 means maximum delay.
!48	**Fast garbage collect mode.** Garbage collection is the term used for the computer's method of managing the memory used by string variables. As each new string is added to memory, it is simply plopped into memory at the next open space. As strings are removed from memory, though, the space they occupied just sits there unused. Because of this the amount of open space for new strings slowly decreases, and the computer will eventually have to "collect" all the garbage strings just sitting in memory. This process can take up to a couple minutes on a system with lots of variables, so this mode was designed to allow an alternative (and faster) routine to be used. Fast garbage collect mode is turned on by setting **!48** to a non-zero value. Before executing each command, the computer will automatically check to see how much memory is open for strings. If it drops below 256 bytes, the routine will execute a fast version of the garbage collection routine used by the computer. Garbage collection delays which could have lasted minutes will last only seconds. The fast garbage collect can handle a maximum of 2048 strings, and if there are more the **!48** variable will be set back to 0. This is to avoid lengthy delays by having two conflicting garbage collection routines working at the same time. You should have **!48** on when using routines that read in a lot of string data (the directory regenerate routine uses this variable).
!49	**Boot device numbers.** A read-only table of 3 values, **!49(0) to !49(2)**, which holds the device numbers for the Boot drive, Program drive, and External drive, respectively. See also **!50**, **!51**, and the function **@30.**
!50	**Boot drive numbers.** This is a read-only table of 3 values, **!50(0) to !50(2)**, which holds the drive numbers for the Boot drive, Program drive, and External drive, respectively. See also **!49**, **!51**, and the function **@30**.
!51	**Disk swapping flag.**

Var	Description
	This is a read-only flag defined through the boot maker programs when you answer the question for if you need to swap the Program and Boot disks for floppy drive systems. If this is 0, then no disk swapping is necessary, otherwise this will be non-zero. See also **!49**, **!50**, and the function **@30**.
!52	**Most recent find location.** This read-only variable will contain the value of the most recent **@2** or **@25** find function executed. This can be used to avoid having to assign the value of the function to a temporary variable. This will be 0 if you use the find functions to find the actual number of finds, rather than the location of a find.
!53	**Flag - 2Mhz during Fast Garbage Collect.** The setting of this variable determines whether the system will attempt to enter 2-megahertz mode on 128 computers when doing the Fast Garbage Collect (FGC) routine. This variable is set in line 150 of the √sys.loadml program when the ML is installed. If it is 1, then the computer will try to use 2Mhz mode when in the FGC routine. This has no effect for C64 users or systems using the TurboMaster CPU, but this feature will automatically be used for C128 users. If you notice that this setting is causing a problem when the system is running, then you may need to change the √sys.loadml program to set this variable to 0. This way, the system will never attempt to enter 2Mhz mode. Also, regardless of the setting of this variable the screen will always be blanked during a FGC. See also the variable !48 for more information on Fast Garbage Collect.

- 5.1.8 ML Function Summary

  The ML functions are used where the ML commands and ML variables will not be able to perform more complex operations. Simply put, a function is like a variable that can perform a subroutine before returning a computed value. Some functions also have parameters, which are used to perform the function's specific operation. Since the ML variables cannot return string values, the functions also take care of this need as well.

  All functions begin with an "**@**" at sign, followed by one or two digits (the function number). Functions are either string or numeric functions, as will be indicated. Some functions have parameters that are required and/or optional; parameters are included in parenthesis after the function number. Here is an example of using a function:

  1000 a$=@5:sr=st:a=val(a$):ifa<>0thenprint@1(a)
  1010 ifsr=.then1000

  | What this routine would do is first use the function **@5**, which reads in a line of disk data like the **.01** command, and then returns the line of data as its value. Then the routine tests to see if what was read in was numeric characters. If it was then it will use the **@1** function to print the value of the input without a leading space. The routine continues reading in data until the end of the file.

  Table 22 below lists the ML Functions and descriptions.

Table 22 - ML Function Descriptions

ML FUNC	Format & Description
@0	**Format: @0 : STRING** This is the equivalent of ***LEFT$(TX$,!40***). This is used for when lines are returned from disk input (**.01**) or from key input (**.02**). Example: **.01:A$=@0**
@1	**Format: @1(<*number*>) : STRING** This returns the equivalent of ***STR$(<number>)***, but it will not include the leading blank that accompanies positive numbers. Examples: • **@1(500)** will return the string "500" • **@1(-10)** will return the string "-10" This is better than ***MID$(STR$(<number>),2)*** because it will not strip a negative sign.
@2	**Format: @2(<*string1*> , <*string2*> [, <*number*>]) : NUMERIC** This will search for and return the position of <*string1*> inside <*string2*>, searching from the first character to the end of <string2>. If <*string1*> cannot be found in <*string2*>, then 0 is returned. If you include <*number*>, then the **@2** function will return the position of the <*number*> occurrence of <*string1*> in <*string2*>. If <number> is 0, then the **@2** function will return the number of occurences of <*string1*> in <*string2*>. Examples: • **@2("b","abc")** would return 2, the position of b in abc • **@2("b","def")** would return 0, because b is not in def • **@2("b","abcdabc",2)** would return 6, the position of the second b • **@2("b","bobby",0)** would return 3, the number of b's in bobby The **!52** variable will contain the value of the most recent find but will be 0 if the find command was used to find the number of occurances of <*string1*>.
@3	**Format: @3(<*string*> , <*number*>) : STRING** This function returns a string with <*number*> characters of the first character of <*string*>. Example: **@3("A",5)** would return the string "AAAAA". <*number*> can range from 0 to 255. If the string is a null string, then a null will be returned.
@4	**Format: @4 : STRING** This function is for input from the modem and returns a character as if you had used a GET statement. Example: **A$=@4**. If there are no characters in the modem input buffer, then a null (**nu$**) will be returned.
@5	**Format: @5 : STRING** This is a the same as the **@0** function, except it executes a **.01** (get line from disk) command before returning the string. Example: **A$=@5:SR=ST:PRINT#9,A$;**
@6	**Format: @6(<*string1*> [, <*string2*>]) : NUMERIC** This is an extended ASC function. Its operation varies depending on the number characters you use in each string (regardless of if you use two strings or not). Here is a table of the possible outcomes:

ML FUNC	Format & Description
	• One string, length of 0 or 1: This will return the PET ASCII value of <*string1*>, and will return 0 if it is a null string (not an error like ASC). **Example: GET#8,A$:I=@6(A$)** • One string, length of 2: This will return the 16-bit value of the string. The formula is 256*<2nd char> + <first char> using the characters' appropriate PET ASCII values. **Example: IFLEN(I$)=2THENL=@6(I$)** This version is not as useful as the next one: • Two strings, any lengths: This is useful for when a GET# command reads from disk two characters which represent the low and high byte of a 16-bit value. <*string1*> represents the low-byte character, and <*string2*> represents the high-byte character. **Example: GET#8,A$,B$:L=@6(A$,B$)** **L** would contain the 16-bit value of the 2-byte input. Only the first character of each of these strings will be used in the calculation. If a string is a null string, then its value is considered zero.
@7	**Format: @7(<*number*>) : STRING** This is an extended CHR$ function. It will return a two-character string representing the low and high bytes of <*number*>, respectively. This is useful for the relative file positioning command. Example: **PRINT#15,"p"CHR$(104)@7(RN)CHR$(1)**
@8	**Format: @8 : STRING** This is the same as **@0**, except it executes an **.02** (Get keyed line of input) command before returning the string. Example: **I$=@8:P=!01:IFP=0THENRETURN**
@9	**Format: @9 : NUMERIC** An enhanced FRE function. This returns the amount of free memory as a positive number (unlike how FRE sometimes returns a negative number). This routine does not perform a garbage collection (the slow delay often accompanying using the FRE function), but instead uses a routine that calculates free memory accurately. Example: I=@9:IFI>3000THENRETURN
@10	**Format: @10(<*string*> [, <*number*>]) : STRING** This function returns <string>, but with all graphics control characters stripped from it. This is useful for determining the exact width of the set of normal characters in the string (for centering, etc.). If you include <number>, then only the first <number> characters will be returned, just like the LEFT$ function. <*number*> can range from 0 to 255. Example: A$=@10(@0,15)
@11	**Format: @11 : STRING** This function returns the current time in the format "HH:MM am" or "HH:MM pm", where HH is the current hour and MM is the current minute. The **!45** and **!44** variables are also changed. **!45** will contain the hour, and **!44** will be 0 if the time is AM, or non-zero if the time is PM.
@12	**Format: @12(<*string1*> , <*string2*> , <*number*>) : STRING**

ML FUNC	Format & Description
	This function returns a string that is <*string2*> with <*string1*> overlayed (replacing existing characters) at position <*number*>. <*number*> can range from 1 to 255. If the strings do not overlap, then the space in between them will be padded with spaces. Examples: • **@12("there","Hello",7)** will return the string "Hello there" • **@12("*","++++++++",4)** will return "+++*++++" The **@12** function can also be used in conjunction with the **@2** and **@25** search-string functions as a "search and replace" function. If 0 is used for <*number*> in the **@12** function, then the value of the MOST RECENT **@2** or **@25** function will be used. If that value is 0, then an ILLEGAL QUANTITY error will occur. Also, if the **@2** or **@25** function was used to find the number of occurrences of a string rather than the position, then an error will occur. Example of the replace function: **1000 IF @2(".",I$)>0 THEN I$=@12("/",I$,0) : GOTO 1000** The above routine will keep replacing all the "." characters in **I$** with a "/" character until no more are found. See lines 2400-2403 in √bbs.msgs (the routine that sifts merged messages for unauthorized MCI usage) for an example of the use of the **@12** function.
@13	**Format: @13 : STRING** This function is used in the crash routine to determine what the BASIC error message was at the crash.
@14	**Format: @14 : STRING** This function is also used in the crash routine. It returns the error line number as a string.
@15	**Format: @15(<*year*> , <*month*> , <*day*>) : NUMERIC** This function returns an ADN (absolute day number) using the year, month, and day parameters. If either the year, month, or day is invalid, then negative one is returned.
@16	**Format: @16(<*number*>) : NUMERIC** This function converts the ADN in <*number*> to a date and returns the YEAR of the date. The month and day are returned in the **!43** and **!42** variables, respectively. If the ADN is invalid, then the results will be unpredictable.
@17	**Format: @17(<*number*>) : NUMERIC** This function returns the day of the week using the ADN in <*number*>. The value will be 0 to 6 for Sunday through Saturday.
@18	**Format: @18(<number>) : STRING** This function will return a string in the format "MM/DD/YYYY" using the AND in <*number*>. If the ADN is invalid, then the string "--/--/----" will be returned.
@19	**Format: @19(<*string*>) : NUMERIC** <*string*> is a date in the format "MM/DD/YYYY". This function will return the ADN calculated from <*string*>. If the date is invalid, then the function will return negative 1.
@20	**Format: @20 : STRING** This function returns the file name of the BASIC overlay currently in memory. It is used by the crash routine to get the name of the current overlay.
@21	**Format: @21(<*string*>) : STRING** This function returns a string that is <*string*> stripped of all characters that are not alphabetic. It will also convert any uppercase letters to lowercase.

ML FUNC	Format & Description
	Example: **@21("This Is A Test")** would return "thisisatest". This function is used for the password file to enable a search for users who have uppercase characters and graphics in their username.
@22	**Format: @22(<*number*>) : STRING** This function will return a special compressed version of <*number*> for writing to a disk file. It is like a STR$ function, but packs two digits into each byte, cutting the length of the string in half. Programmer's note: This does NOT convert the number to BCD (Binary Coded Decimal). Rather, it uses a special method to avoid ever creating a carriage return character in the string (because such a character would not enable the compressed number to be read in from disk). Example: **PRINT#8,@22(I)**
@23	**Format: @23(<*string*>) : NUMERIC** This function performs the opposite of the **@22** function and uncompresses the number stored in <*string*>. An invalid string will cause unpredictable results. Example: **INPUT#8,I$:I=@23(I$)**
@24	**Format: @24(<*string*> , <*number*>) : STRING** This function is used to pad relative file records to get them ready to print to disk. • If <*string*> is less than <number> characters in length, then <*string*> will be returned with enough carriage returns padded on the end to make it <*number*> characters in length. • If <*string*> is the same as or more than <*number*> characters in length, then <*string*> will be truncated to <*number*> minus one character, and a carriage return will be added, so the resulting string will be <*number*> characters in length. • If <*string*> is a null, or <*number*> is less than 2 characters, then there will be unpredictable results: Example: **F$=F$+@24("name",15)**
@25	**Format: @25(<*string1*> , <*string2*> [, <*number*>]) : NUMERIC** This works the same as the **@2** function, except the search for the string proceeds from the END of <*string2*> to the beginning (backwards).
@26	**Format: @26(<*string1*> , <*string2*> , <*number*>) : STRING** This function returns the <number> section of <*string2*> separated by the character in <*string1*>. Example: **@26("!","cp6!i6!cd//directory",2)** would return "i6", because the "!" is the separator character, and "i6" is the second section divided off by the "!" character. This function is used by the drive command routine for dividing up the drive command into its sections. If <*number*> is 0, then an error will result.
@27	**Format: @27(<*number1*> , <*number2*>) : NUMERIC** This function calculates the calendar age from two ADN's. The age is the number of calendar years from the ADN in <*number2*> to the ADN in <*number1*> (like subtracting <*number2*> from <*number1*>).

ML FUNC	Format & Description
	For example, to calculate someone's age on the BBS system, the formula would be: **age=@27(DA,BD)** where DA is the system date ADN, and BD is the birth date ADN.
@28	**Format: @28(<*string*> , <*array*>) : NUMERIC** This function is a search function which will search through an array to find <*string*>. <*array*> is the name of a one-dimensional string array, but the name should not include either the $ character or the array parenthesis (e.g. the string array "a$()" would simply be "a"). The function will return the index of <array> where the string was found. If the string is not found, will return a negative one. The function will search through the entire array, and the string must match the array element exactly. If <*array*> is not found in memory or is more than one dimension, then an error will result. Example: **I=@28(A$,A) : IFI<0THEN#"not found!"**
@29	**Format: @29(<*number*>) : STRING** This function is like **@5**, except that <*number*> is how many characters you want read in. It is a viable alternative to the GET# command in some cases. It does not check for End of File and it does not stop at the End of Line character stored in **!04**.
@30	**Format: @30(<*number*>) : STRING** This function is used in conjunction with the variables **!49**, **!50**, and **!51**. It returns the drive init command for one of the three boot drives: 0=Boot drive, 1=Program drive, 2=External drive.

5.2 The Generic Overlay Files

For the programmer, Color 64 also includes generic skeleton overlays which can be used to write online games and modules:

- The √bbs.xxx Module
  The program "√bbs.xxx" has all the routines that "√bbs.ov3" has in it, except it doesn't have any "spare" command routines. You can use this for modules that would regularly have a spare command of their own, but that you might want to install as a non-system overlay.

- The XXX Small Module
  The file called "xxx small" is for use with games and modules that are to be loaded by the Mod Menu program. This program is stripped of everything but the essential routines that most games will need. The procedure for setting up a new game for the Mod Menu is as follows (for games/modules that start at line 28000 or above):

  First, delete the old overlay file out of the game (lines 0-27999) if necessary.

  Next, merge in "xxx small". Remember to change line 1 to reflect the filename for resaving purposes.

Next, enter line 5 as follows:

5 gosubXXXXX:gosub489:.dr$+"√bbs.ov2*",dv

XXXXX is the line number that needs to be GOSUB'ed to start the game or module.

If you want the overlay to use one of the AUX file groups rather than the system files, then you need to make one other change. In line 481 change the "H=1" to H=12, H=13, or H=14 to use AUX 1, AUX 2, or AUX 3, respectively.

Now the game/module is ready to load from the Mod Menu. Line 5 ensures that control will return to the Mod Menu when the game/module ends. Any game or module accessed through the Mod Menu MUST return to the Mod Menu so that a "variable kill" to be activated. Many modules add extra variables and arrays to memory that won't even be used until the module is loaded again. Mod Menu takes care of this by removing these extra variables from memory when control is returned to the menu. This process is affectionately referred to as the "variable killer".

Note: Some games use variables to indicate how many times a user has played the game while online, or other "permanent" information. These variables will be lost unless they are created before the game is played. If you wish, you can set these variables to a dummy value in the √bbs.init initialization routine so they will be preserved.

- The XXX Off-Line Module
  The included file "xxx ol" is a special module with which you can write your own off-line programs, separate from the BBS system. This makes it easier to test games or modifications without having to start the BBS system.

  Line 10500 to 10550 of the overlay is where the user parameters are set. Set the different variables in this line range to values that would be helpful during the testing of your program. Consult section 5.8 for information on what each variable is used for.

  The first line of your routine should be at line 11000, because the program just drops through to this line after the √bbs.parms file has been read in. If your program does not regularly start at this line (i.e. the game starts at line 30000) then you can simply put a GOTO command at line 11000 to go to the appropriate line number.

  The ML shell will need to be installed before you can run a program with this module, so you will need to use an off-line program after shutting down the BBS system or you need to run the "+shell" boot program.

  While the program is running, you will be able to stop it just as you would the BBS program, but you will not have to reboot the BBS to test the program again. Simply RUN it and it will restart.

- The XXX Message Editor Module
  Some games/modules may use the text editor. The included file "xxx msg" is a self-contained message editor that can be merged into "xxx small" or "√bbs.xxx". Here are all the routines and their entry points:

  a. Line 1205 - This displays the message "Message Maker" and asks for a device number, drive, command, and filename for editing. This is usually for Sysop/CoSysop modules only.
  b. Line 1215 - This routine prints "opening <filename>...". The filename should be put in the variable FI$ before calling this routine. The currently selected drive is used. The program then proceeds with the next routine. Use this routine when the user needs to see which file is being opened.
  c. Line 1220 - This attempts to open a file (the filename stored in **FI$)** for editing. No message indicating which file is being opened is printed. The currently selected drive is used. If the file does not exist, then the routine drops into the message editor to edit a new message. Use this routine for modules which edit existing files (such as a Trivia module that allows the Trivia sysop to edit questions, etc.).
  d. Line 1270 - This jumps right into the message editor. The message is stored with the filename FI$ and is stored on the currently selected drive. Any existing file is scratched before the new file is saved. (Use this routine when the user must create a new file from scratch).

  This message editor has all the features of the editors built into the other modules, including the Help command and the Merge command (for those who have access).

5.4 Various Programming Notes

- If you place an F1 character inside the quotes of text to be sent, rainbow mode will be turned on, using the system colors defined in SETUP. Turn off by placing an F3 character inside the quotes.
- If you place an F5 character inside the quotes, the cursor will change to the next color from the sequence defined in SETUP.
- If you place an F7 character inside the quotes, the border/screen will change to black.
- To read a sequential file from the System Files, use the following:
  f$="filename":gosub205
- To input data from a sequential file you may be using for data storage, you can use normal INPUT commands as always. I have some ML routines to do the same thing, but it is not necessary that you use it; INPUT# should work fine in most cases.
- To input a typed line from the user, use **GOSUB 310**. Your caller's input will be stored in **I$**.
- To get one character from the modem/console, use **GOSUB 110**. If a character had been typed, it will be stored in **A$**.
- To wait for a Y/N response, use **GOSUB 1010**. Their response will be in **A$**.
- To ask "Are you sure?", use **GOSUB 1005**

- To store something in the caller log, do one of the following:
  a) **a$="info to save":gosub8004** or
  b) **i$="info to save":gosub8003**
- After inputting from the user or reading a seq file, if you check the value of **P**, you will be able to determine if there has been a carrier lost. If **P=255**, then either carrier was lost or there has been a timeout. If **P=1** then the caller typed CTRL/P and you should assume they want to abort the current function. Generally, after every input, it is recommended to have the following line of code: **IF P THEN RETURN**
- To reset the background color or uppercase mode, do the following: **GOSUB 13680**
- To check the error channel after accessing the disk drive, use **GOSUB 510**. If there was a disk error, it will automatically print to your screen and the caller's. The variables **er, er$, et, es** will contain the disk status. Some types of disk errors will not print and should be handled by your routine if necessary. They are #62, #63, #64 and #73. Also, if there is a disk full error, **GOSUB 510** will automatically validate the disk.
- To select the system files drive: **GOSUB 481**
- To convert **I$** to a numeric format, use **GOSUB 610**. The numeric value will be held in variable **I**. And if **I$** contains non-numeric information, the variable **I** will be equal to **0**.
- To use one of the "spare" commands in SETUP, refer to Table 23 to determine which command to use.

Table 23 - Spare Command Reference Chart

Command	Loads Prg	Executes Line		Command	Loads Prg	Executes Line
Spare1	Msgs	13430		Spare6	Ov1	13465
Spare2	Xfer	13440		Spare7	Ov1	13470
Spare3	Ov1	13450		Spare8	Ov1	13475
Spare4	Ov2	13455		Spare9	Ov1	13480
Spare5	Ov3	13460				

As you can see, there is one spare command pointed to each program file with spare 6 through 9 all pointing to √bbs.ovl. This allows you to put all you smaller files in √bbs.ovl and save √bbs.ov2 and √bbs.ov3 for larger files (the included Mod Menu program uses √bbs.ov2). If you want to have lots of little programs in ov2 or ov3, I recommend you design a menu that is called spare 4 or spare 5, then have this command select the desired subroutine.

5.4.1 Generic Routines

The following is a table of generic routines that are present in all overlays unless otherwise specified.

Table 24 - Callable Routines by Line Number for Overlays

Line	Function
1	REM line -- This is the save and replace overlay routine
105	Print "." to screen (no carriage return), drop to next line
110	Get input character and put in A$
155	Print two carriage returns (C2$)
160	Print A$ with carriage return
202	Open file (F$) on SYSTEMS drive
203	Open file (F$) on default (current) drive
205	Open and read file (F$) on SYSTEMS drive (not in Network overlays)
210	Open and read file (F$) on default (current) drive
220	Read open file (F$) - Will not print PRESS ANY KEY if needed
265	Read open file (F$) - Will print PRESS ANY KEY if needed
280	PRESS ANY KEY routine
300	Plain ASCII string input to I$, no prompt
302	Uppercase only string input to I$, with prompt
303	Plain ASCII string input to I$, with prompt
305	Commodore graphics allowed string input to I$, with prompt
310	Commodore graphics allowed string input to I$, no prompt
360	Timeout/Carrier/Abort/Time Limit Exceeded routine
390	Removes leading and trailing spaces from I$
460	Select files group. See "File Group Numbers"
480	Select Password File drive (not in all overlays)
481	Select System Files drive (not in all overlays)
482	Select Help Files drive (not in all overlays)
483	Select default download directory (not in all overlays)
484	Select default upload directory (not in all overlays)
485	Select Public Messages drive (not in all overlays)
486	Select Private Messages drive (not in all overlays)
487	Select Text Files drive (not in all overlays)
488	Select Caller Log drive (not in all overlays)
489	Select Program Files drive
490	Select group 10, used for misc. drive group (not in all overlays)
505	Read error channel and close logical file 8
510	Read error channel into ER, ER$, ES, ET
605	Get line of input and convert to numeric (not in all overlays)
610	Convert I$ to numeric
995	Clear keyboard input buffer
1005	Ask ARE YOU SURE and get "Y" or "N" response
1010	Print box prompt and wait for "Y" or "N" response
1020	Wait for "Y" or "N" response from user
1110	Get current time in T$
1530	Clear the array A$ from elements 0 to A-1
8003	Put contents of I$ into caller log
8004	Put contents of A$ into caller log

Line	Function
8010	Clear contents of LG$ by dumping it to caller log buffer
9991	Crash routine
9999	Part of crash routine. Close files, then OV=5 and goto √bbs.init
13600	Print "Online" information about current user
13630	Time Limit Exceeded message
13640	Print box prompt if using Commodore or ANSI graphics
13652	Disconnect current user
13680	Reset screen if background not black or if in uppercase mode
13695	Wait for a any type of response from modem for 2 seconds
15915	Print B$ init string to modem, and wait 3 seconds for OK response

5.4.2 File Group Numbers

When you use the **GOSUB460** routine to select a file group, the variable **H** must first be set to the group number. Table 25 provides the list of all the groups and their respective assignments:

Table 25 - File Group Assignments

Group ID	Group Name		Group ID	Group Name
0	Password File		8	Caller Log
1	System Files		9	Program Files
2	Help Files		10	Used for temporary storage of drive information
3	Default DL Directory		11	Network Files
4	Default UL Directory		12	Auxiliary Files 1 *
5	Public Messages		13	Auxiliary Files 2 *
6	Private Files		14	Auxiliary Files 3 *
7	Text Files		15	Reserved for future use

* There is no built-in line number to select the AUX file groups, so you must select them manually.
To do this, set H to the following value then perform a GOSUB460 | H=12 for AUX1 | H=13 for AUX2 | H=14 for AUX3

5.5 Merging in Modules

The following notes are intended to help you when merging an optional "spare command" module into your Color 64 overlays.

1. Make sure that the module was written for your BBS program version. If it is for a previous version, then you will need to find the correct version of the module or use the included module converter program (see the section on the module converter in the upgrade documentation).

2. Load a programmer's utility. I recommend BAID64, since it is in the public domain, and it does a true merge (not just an append). It is included with the Color 64 package.

3. Make sure the module that you are going to merge will not overwrite any lines already existing in the intended overlay program. Just load the overlay program and list the line range used by the module and if you don't see anything... it is ok. If not, consider placing the module in one of the other overlays. The exception to this case is if the module was intended to replace some of the standard BBS routines.

   Once you have determined that the module will not overwrite any existing code, just enter the following command (this is for BAID64): **merge "module/name"**

   Make sure that the BBS overlay program is already in memory. In just a few moments, the lines from the module will be merged with your system overlay file.

4. The next thing you need to do is to modify the code so that one of the spare commands will execute the module. There are 9 spare commands available in Color 64 BBS. Each of these spare commands will load a pre-defined overlay file and execute a pre-defined line number. Table X summarizes the defined overlays and lines executed for spare commands 1-9:

Table 26 - Spare Command Pre-Assignments

Spare Assignment	Overlay Assignment	Line Execution
Spare 1	√bbs.msgs	13430
Spare 2	√bbs.xfer	13440
Spare 3	√bbs.ovl	13450
Spare 4	√bbs.ov2	13455
Spare 5	√bbs.ov3	13460
Spare 6	√bbs.ovl	13465
Spare 7	√bbs.ovl	13470
Spare 8	√bbs.ovl	13475
Spare 9	√bbs.ovl	13480

As you may notice, there is only one spare command pointing to √bbs.ov2 and √bbs.ov3. I intended that these overlays be used by modules that would require the whole overlay (like the included Mod Menu program), or that you use a menu program to access several subprograms in the same overlay.

Anyway, you need to modify the spare line at the appropriate line number based on the overlay program you are merging with. When you list 13430-13480, you will see the available space command lines for the overlay program loaded into memory. They look like:

```
134XX goto13100:rem spare X
```

You will not see all the lines in the above table, just the lines that apply for the overlay program you have loaded. You will want to use the insert key to push the "goto13100" over and enter a gosub pointing to your newly merged module. Example:

```
134XX gosubXXXXX:goto13100:rem spare X
```

5. Save this modified version of your overlay program back onto the disk. If you list line 1 of the program, then delete "1 rem" and hit return, the line will automatically scratch the old overlay program and then save your modified version on your disk. After it is saved, do a directory to see the size of the new file. This file must always be smaller than the √bbs.init overlay. If the file is too large, you will need to use a different overlay or take something else out of this overlay.

6. Run the SETUP program and edit the BBS Commands section. In this section, you will want to set the access level for the intended spare command to the desired value. You can also change the command key required to execute the module. If you do change the command, make sure you don't select a command that is already used.

Well, that will do it. You may want to write down the line numbers occupied by this module so that if you decide to remove it to make space for another one you will know exactly which lines to delete (using your programming utility - like BAID64) and to change the spare command line back to normal.

5.6 Color 64 Command Branching Table

This table represents the overlays and line numbers at which the main commands reside. The default command keys are listed here under the heading KEY.

What happens when a user presses a key at the main prompt is that the command is searched for in the array **CM%(I,1)**, which stores the ASCII value of each of the commands. Once found, the user's level is compared to the command level stored in **CM%(I,2)**. If all is well, then the command number ends up in the variable I. The command number is listed under "No." in the table.

Next, the variable I is used in a series of ON GOTO commands that start at line 13160. Since there are far too many commands to do the ON GOTO in one line, the ON GOTO statement at line 13160 jumps to one of the following 4 lines which handle 10 commands each. Line 13170 handles commands 0 to 9, line 13180 handles commands 10 to 19, line 13190 handles commands 20 to 28

(29 is not an actual command), line 13195 handles commands 30 to 39, and line 13197 handles commands 40 to 42.

For commands that are in the currently loaded overlay, the ON GOTO will be to a special "mini routine", which exists in the line range 13200 to 13480. This mini routine usually does another GOSUB to the actual routine, then does a GOTO13100 to return to the main prompt. This is done because some of these main routines are used locally from areas like the SYSOP menu, in which case you don't want to return to the main prompt. The line numbers for these mini routines are listed in the table under the heading MINI.

The actual command routines are usually located in various places in the overlays. Sometime the entire command function does not exist totally in one overlay, such as the [A]Alter Password command which must go to √bbs.init to change the password. In this table the overlays are listed which are first branched to (in the case of the Alter Password command this means the √bbs.ovl overlay). The line numbers for the actual routines are listed in the table under ACTUAL.

Commands that are not in the current overlay are sent to one of the overlay loader lines which exist at lines 88 to 103. These lines will load the appropriate overlay in which the command exists. The loader line is listed in this table under LOAD, and the abbreviation for each overlay is listed in the table under OVRL.

Before loading the appropriate overlay, the variable **OV** is set so that the new overlay will know where to branch to once loaded. Usually this is set to 1 automatically when the loader lines are used, because most overlays reserve **OV** number 1 to handle a command that has been relayed to it. Once in the overlay the variable I is used to perform the ON GOTO statements at line 13160. Sometimes, however, the **OV** number is not 1 as in the case of the Set Date & Time command. Usually this is in the case of overlays which do not have a menu of their own, such as √bbs.init. The appropriate settings for **OV** are listed in Table 27 below:

Table 27 - OV Settings Cross Reference

Key	No.	MINI	Actual	Overlay	Load	**OV**	Function
R	0	13202	3005	MSGS	90	1	Read Public Messages
Q	1	13205	3710	MSGS	90	1	Post Office
P	2	13210	1260	MSGS	90	1	Post a Public Message
S	3	13220	4005	MSGS	90	1	Scratch a Public Message
$	4	13230	405	XFER	92	1	View Download Directory
D	5	13240	16505	XFER	92	1	Download File(s)
#	6	13245	16110	XFER	92	1	Choose a Download Directory
U	7	13250	16300	XFER	92	1	Upload a File
!	8	13255	27005	OVL	96	1	Edit User Stats
F	9	13260	7010	MSGS	90	1	Feedback to Sysop
C	10	13270	15002	OVL	96	1	Chat with the Sysop
A	11	13280	13735	OVL	96	1	Alter Password
O	12	13290	19010	OVL	96	1	Log off System

Key	No.	MINI	Actual	Overlay	Load	OV	Function
G	13	13295	13910	OVL	96	1	Select Graphics/Ascii/Ansi
H	14	13300	17015	OVL	96	1	View Help Files
W	15	13310	13310	OVL	96	1	View Welcome Files
M	16	13330	19510	MSGS	90	1	View Membership List
I	17	13350	13350	OVL	96	1	View Information File
E	18	13360	5010	MSGS	90	1	Edit a Public Message
.	19	13370	24005	INIT	95	2	Set Date & Time
>	20	13372	22010	XFER	92	1	Dos Wedge
<	21	13375	20010	INIT	95	7	Password Maintenance
N	22	13377	15505	OVL	96	1	New Downloads
X	23	13380	13810	XFER	92	1	Scratch a Download
T	24	13385	17010	XFER	96	1	Text Files Area
L	25	13380	8110	OVL	96	1	View Caller Log
+	25	13400	35020	OVL	96	1	Multi-Upload
Z	27	13410	13710	XFER	92	1	Edit Download Description
Y	28	13420	14010	XFER	92	1	Release a Download
	29						Not a command
1	30	13430	14310	MSGS	90	1	Spare Command 1
2	31	13440	13440	XFER	92	1	Spare Command 2
3	32	13450	13450	OVL	96	1	Spare Command 3
*	33	13455	2000	OV2	98	1	Spare Command 4 (Mod Menu)
5	34		13460	OV3	100	1	Spare Command 5
6	35	13465	13465	OVL	96	1	Spare Command 6
7	36	13470	13470	OVL	96	1	Spare Command 7
8	37	13475	13475	OVL	96	1	Spare Command 8
9	38	13480	13480	OVL	96	1	Spare Command 9
%	39	13425	16050	OVL	96	1	Select Transfer Protocol
=	40		14005	NW1	102	1	Network Post
&	41		34010	NW2	88	1	Network Maintenance Menu
-	42		9010	NW2	88	1	Release Network Messages
	43						Not a command
?	--		13010	All			Main BBS Menu

5.6.1 The Overlay Loader Lines

Table 28 defines the standard main overlay loader lines. Some of these lines are not in all the overlays because they may not be needed:

Table 28 - Standard Main Overlay Loader Lines

Line	Overlay		Line	Overlay
89	√bbs.nw2		97	√bbs.ovl
91	√bbs.msgs		99	√bbs.ov2
93	√bbs.xfer		101	√bbs.ov3

95	√bbs.init		103	√bbs.nw1

Also, in the overlays that have a main prompt there is a line before each of the above lines that will set OV=1. For example, to set OV=1 and load the √bbs.msgs overlay, you would **GOTO 90** (one line before 91).

5.7 Individual Overlay Branching

The following sections serve as descriptions of how each overlay behaves depending on the setting of OV when the overlay is loaded. The path of GOTOs, GOSUBs, and other information is listed in a special format. Here is an example of the path of OV=3 in √bbs.nw1:

OV=3 : 50, (12010), (12220), [6/INIT] | *Regen Net index (from Net Menu)*

Plain line numbers (as in the 50) are the equivalent of a GOTO command. Line numbers in parenthesis are the equivalent of a GOSUB command. A branch to another overlay is listed in the format [N/XXXX] where N the setting of OV and XXXX is the overlay to be loaded. Finally, the description of the function follows the path. In the above example then, the path of the command routine would be as follows:

5 goto50 *(line 5 is always first, where it does an ON OV GOTO)*
50 gosub12010:gosub12220:ov=6:goto95 *(line 95 is standard INIT loader)*

Sometimes this path is somewhat simplified from the actual code used in the overlays, but it gives a good idea of what functions are performed where.

5.7.1 The √bbs.INIT Overlay

Table 29 depicts the paths for all settings of OV in the various overlays:

Table 29 – Path for settings of OV - √BBS.INIT Overlay

Line Command	Setting area
√bbs.init	
OV=1 : 40, (19044), 30	Post-logoff procedures
OV=2 : 50, (12865), [3/MSGS]	Set Date & Time (from main BBS prompt)
OV=3 : 60, (12760), 30	Back to Answer Feedback prompt (from √bbs.msgs)
OV=4 : 30, (12010), [2/MSGS]	Wait-For-Call screen
OV=5 : 9999	Save msg index, variables, end program
OV=6 : 70, (12800), 30	Network Menu
OV=7 : 75, (20010), [3/MSGS]	Password Maint (from main BBS prompt)
OV=8 : 80, (2710), [4/MSGS]	Validate by E-mail

Line Command	Setting area
OV=9 : 78, (13735), [3,MSGS]	Set new password (relayed from √bbs.ovl)
OV=10 : 65, (12092), [2/MSGS]	End of midnight routine (back to WFC)
OV=11 : 35, (11905), 30	Reset √level x files
OV=12 : 85, (18240), [2/MSGS]	Return to application routine (from NW1)
OV=13 : 33, (29010), 30	Return to WFC after new node applied
√bbs.msgs	
OV=1 : 60, (13160), 57	Command relayed from other overlay
OV=2 : 50, <read √welcome2>, (13523), (8510), (9005), 55	Welcome msgs
OV=3 : 55, (13100), 57	Main BBS prompt
OV=4 : 70, (1410), <ON VX GOTO 71,72,55,74>	General msg editor prompt
OV=5 : 65, (12822), [3/INIT]	Answer SYSOP feedback
OV=6 : 80, (7020), 55	Feedback to SYSOP
OV=7 : 71, (9048), 55	Continue reading mail (after replying Net mail)
OV=8 : 72, (3675), (3320), 55	Continue reading msgs (after Net reply)
---- : 57, [3/OVL]	Logoff
---- : 74, [3/INIT]	Back to Answer SYSOP Feedback prompt
√bbs.xfer	
OV=1 : 60, (13160), [3/OVL]	Command relayed from other overlay
OV=2 : 65, (22010), [4/INIT]	DOS command (from SYSOP Menu)
√bbs.ovl	
OV=1 : 60, (13160), 80	Command relayed from other overlay
OV=2 : 70, (28005), *or* **(25005),** *or* **(25100),** *AND* **IF M3=0 THEN [10/INIT]** *or* **IF M3=1 THEN [7/NW1]**	Purge old mail Validate disks update download accesses -- Continue midnight routine in INIT -- Continue with Network midnight routines
OV=3 : 80, (19010), [1/INIT]	Logoff
√bbs.ov2 * and √bbs.ov3	
OV=1 : 60, (13160), [3/OVL]	Command relayed from other overlay
* (Note that the included √bbs.ov2 overlay jumps directly to the Mod Menu without going through 13160 first)	
√bbs.nw1	
OV=1 : 60, (14005), [3/MSGS]	Post Network Msg (from main BBS prompt)
OV=2 : 40, (14005), [6/INIT]	Post Net Msg (from Net Menu)
OV=3 : 50, (12010), (12220), [6/INIT]	Regen. Net index (from Net Menu)
OV=4 : 35	Illegal OV number. No longer used in version 8.0
OV=5 : 67, (1110), (8010), (40000) *if there was a carrier lock then* **[7/NW2]** *if OK then* **(8010),(7520), [13/INIT]**	Outgoing Net Send -- Update LOCK status -- Post-outgoing routines
OV=6 : 13110	Node login
OV=7 : 85, (12010), (12220), [8/NW2]	Regen Net index (midnight routine)
OV=8 : 35	See OV=4
OV=9 : 35	See OV=4
OV=10 : 35	See OV=4

Line Command	Setting area
OV=11 : 30, (14490), [12/INIT]	New billing entry (during application)
OV=12 : 35	See OV=4
OV=13 : 15, (7010), [7/MSGS]	Private net reply (during e-mail reading)
OV=14 : 20, (7010), [8/MSGS]	Private net reply (during public msgs)
OV=15 : 66, (12010), (12220), [11/INIT]	Regen Net index (start up)
OV=16 : 74, (8010), [13/INIT]	Continue outgoing after CARRIER LOCK
OV=17 : 35	See OV=4
OV=18 : 77, (40000), 72	Continue outgoing after Verify Net Mail
√bbs.nw2	
OV=1 : 70, <ON I-40 GOSUB 3010,4010>, [3/MSGS]	Relay from other overlay
OV=2 : 25, (3010), [6/INIT]	Net Maintenance Menu
OV=3 : 30	Illegal OV number. No longer used in version 8.0
OV=4 : 35, (7010), (7500), [6/INIT]	Set window/speaker (from Net Menu)
OV=5 : 40, (11750), (18520), (18600), [15/NW1]	System start up routine
OV=6 : 50, (5001), [6/INIT]	Send default net message (from Net Menu)
OV=7 : 55, (6050), (6220), [16/NW1]	Update after CARRIER LOCK
OV=8 : 22, (21120), [10/INIT]	Midnight routine; create .node x users
OV=9 : 27, (6290), (6600), [13/INIT]	New node application
OV=10 : 23, (4010), [6/INIT]	Release public net msgs (from Net Menu)
OV=11 : 30	See OV=3
OV=12 : 37, (6300), (6800), [18/NW1]	Post-application to remote node
OV=13 : 47, (1110), [18/NW1]	For use by Verify Net Mail mod
OV=14 : 65, (35440), (8100), [13/INIT]	Distribute incoming Net data

5.7.2 Branching

Table 30 depicts where each of the menu commands branch to, dependent on overlay.

Table 30 - Menu Branch Destinations

Key	Line (+function, if applicable)	Destination
SYSOP MENU - √bbs.init overlay	**Routines begin at line 12510**	
F1	12100	Logon (Local Mode)
F2	12615	Load BBS term; if not found, return to 12010
F3	12635, (8100), 12010	View Caller Log
F4	12610 if M3=0 then (12865) if M3=1 then (12800), 12010	 Set Date & time Go to Net Menu
F5	12630, (12710), 12010	Read Feedback to Sysop
F6	12620, (20010), 12010	Password Maintenance
F7	12625, [3/XFER]	Dos Wedge

Key	Line (+function, if applicable)	Destination
F8	12640, 9999	Shut BBS down
NETWORK MENU - √bbs.init overlay Routines begin at line 12800		
F1	12865, (24005), 11920	Set Date & Time
F2	12850, [4/NW2]	Set Net window/speaker
F3	12835, [2/NW2]	Net Maintenance Menu
F4	12855, [6/NW2]	Default net message multi-send
F5	12840, [10/NW2]	Release public net messages holding
F6	12860, [3/NW1]	Regenerate Network index
F7	12845, [2/NW1]	Post Network Message
F8		Return to WFC
NETWORK MAINTENANCE MENU - √bbs.nw2 overlay Routines begin at line 3010		
1	3080	Billing List/Print
2	3160	Billing Edit
3	3300	Billing Report Generator
4	3450	Billing Total Accounts
5	6010	Node Status Report
6	7110	Node Account File Edit
7	7600	Attach File
8	3072	Read Send Log
9	3073	Read Receive Log

5.7.2 Midnight Routines

The midnight maintenance routines begin in the √bbs.init overlay and span several other overlays. Table 31 depicts these routines:

Table 31- Midnight Maintenance Routine Paths

Line Command	Branch Functional Area
√bbs.init	
INIT : (26210), [2/OVL]	Call to 26210 resets time limits
2/OVL : 70, (28005), (25005), (25100), if M3=0 then [10/INIT] else if M3=1 then [7/NW1]	Calls to • 28005 purges old mail • 25005 validates disks • 25100 updates download access count
7/NW1 : 85, (12010),(12220), [8/NW2]	Calls to • 12010 regenerate Network Index • 12220 count public net msgs holding
8/NW2 : 22, (21120), [10/INIT]	Call to 21120 create √node x users
10/INIT : 65, (12092)	Call to 12092 end of midnight routine; return to WFC

5.8 Color 64 BASIC Variables

Table 32 defines all variables used by Color 64 BBS:

Table 32 - List of Color 64 Defined Variables

Var	Definition & Function
A	General use numeric variable. It is used to store the current number of lines in the message editor and other routines that use the array A$().
A$	This variable has several uses: It is used to return the character read in in the routine at line 110. It is used very often for when information is read in from disk also. Also, the general output commands (#, $, %, & and ') will print the contents of A$ if no string expression is specified. This is a very useful shortcut in many cases and saves programming space.
A$()	This is the most heavily used string array in the Color 64 system. It is used in the message editor to hold the message lines. It is also used by many other routines which handle larger amounts of string information. It is always dimensioned to 232 when the system is first initialized.
A1$	First line of current caller's address as stored in the password file.
A2$	Second line of current caller's address as stored in the password file.
AG	Current caller's access group as stored in the password file.
AR%	Used only in Network overlays, to determine which message category public messages get released into, and which UD directory that Network file transfers are released into. Defined in Net Setup.
AT	Stores whether your modem supports DTR disconnect or not.
AZ$	The MODEM INIT string for Hayes type modems as defined in SETUP (Ex. ATE1X1S0=1S2=43F1Q0V1M0).
B	General Use
B$	General Use
BA	Used only in the Network overlays, this holds the current balance of the caller.
BD	Current caller's date of birth in ADN format as stored in the password file.
BD$	Current caller's date of birth in the form "MM/DD/YYYY", calculated from the variable BD.
BM	Lowest message number on the system as stored in √variables.
BN%	Used only in Network overlays. If this variable is set the system is to lock out nodes when, upon attempting to call a remote node, a NO CARRIER status is returned from the modem. Defined in Net Setup.
BR	Current caller's baud rate.
BS	Last used block size of the current caller. If it is a 0, the caller last used Xmodem, otherwise they last used Punter.
C	General use variable which is often used to temporarily compute the current caller's absolute download credit.
C$	General use variable which is used many times to temporarily hold the contents of TX$, so that the previous value of TX$ can be preserved in cases where TX$ needs to be changed.

Var	Definition & Function
C1	Number of download blocks given for each block uploaded as defined in SETUP. See also C2, PD, and PU.
C2	Number of free upload blocks you give each caller as defined in SETUP. See also C1, PD, and PU.
C2$	This variable is two cr$'s added together. Again, it saves memory and time to print c2$ instead of crcr or chr$(13)chr$(13).
C3	Credit system exemption level as defined in SETUP.
C4	Maximum files you want to allow on your public messages drive as defined in SETUP.
CA	Stores the current message category. It is only used in √bbs.msgs.
CA$()	Category names as defined in SETUP.
CA%()	The category of each message stored in public messages.
CA()	The level for each category as defined in SETUP.
CC	Number of message categories as defined in SETUP.
CD	Your modem's carrier detect status value as defined by SETUP.
CL%	The Network window closing hour as defined in Net Setup.
CM%()	This integer array stores the BBS COMMANDS as defined in SETUP. This array is also used to store some miscellaneous SETUP parameters.
CR$	This variable stores the value of a carriage return. It saves memory and time to print cr$ instead of chr$(13).
CS$	Chat subject that is displayed at the ONLINE: prompt after a caller requests chat mode.
D()	This variable array stores the drive assignments for your file groups as defined in SETUP. It is not wise to change these values unless you understand how the device number and drive number has been encoded in the variable; use GOSUB481-489 to select the desired drives.
D1	The current month as computed by the timer routine at 1110-1190.
D1$()	The array (1 to 12) of month names, January to December.
D2	The current day of the week (0 to 6, for Sunday to Saturday) as calculated by the timer routine at 1110-1190.
D2$()	The array (0 to 6) of names of the days of the week, Sunday to Saturday.
D3	The current day of the month as calculated by the timer routine at 1110-1190.
D4	The current year as calculated by the timer routine at 1110-1190.
DA	Current system ADN (Absolute Day Number). See the section on Absolute Day Numbers for more information.
DA$	Current system date in the format "MM/DD/YYYY". It is calculated from the variable DA.
DC	Maximum number of downloads per call as defined in SETUP.
DD	Default download directory. This is always the first directory defined in SETUP with a Y to download status. See also UD.
DD$	Current caller's membership information.
DE$	This variable stores the value of a DELETE character (CBM ASCII chr$(20)).
DH$()	Drive init commands for the UD directories as defined in SETUP.
DN$()	Names of the UD directories as defined in SETUP.
DN%()	Miscellaneous UD directory parameters as defined in SETUP.
DR$	Drive number of the currently selected drive (e.g. 0:, 1:, etc.).
DT	Number of downloads made made by the caller on this call.
DV	Device number of the currently selected drive.

Var	Definition & Function
DX	Last selected device/drive. The system uses this variable to save time so that the system will not reselect a drive that was already selected. Also, setting this variable to 0 followed by a GOSUB460 will force a reselect of the current drive.
E1$	This variable stores the first 3 or 4 disk errors that are displayed on the waiting for caller screen.
ED	Current caller's membership expiration date as stored in the password file as an ADN.
ED$	Current caller's membership expiration date in the format "MM/DD/YYYY". Calculated from ED.
ER	Error number read in from disk error channel by the routine at 510.
ER$	Error description read in from disk error channel.
ES	Error sector read in from disk error channel.
ET	Error track read in from disk error channel.
F$	This variable stores the last filename accessed. Use **F$="filename":GOSUB210** to read a seq file to the screen and modem. In some cases, this variable is used to store temporary string data.
F%()	This array variable, along with F(), stores the field pointers for password file records.
F()	This array variable, along with F%(), stores the field pointers for the password file records. This allows easier modifications to the password file (for system upgrades) by simply changing the values in SETUP instead of through all of the other program overlays.
F(0)	The BBS's RERUN ON ERRORS status as defined in SETUP.
FI$	This variable stores the filename being edited by the online message editor. It is also used in the directory maintenance routines.
FL	Set to 1 if your BBS is using 1541 Flash! from Skyles Electric Works.
FM	Status of the current caller's word wrap condition (use only in the message editor). If it is a 1, word wrap is on. If it is a 0, word wrap is off.
FR	Flag for the FROM information of the last read message.
FR$	FROM information of the last read message.
FU	This variable, along with FU$, RX%, TE$, TX%, TY, and Z3, is used only in the √bbs.term program and is removed from memory when the term program automatically uses the variable killer.
FU$	See variable FU.
G$()	Mod menu array dimensioned each time mod menu is entered, and removed from memory when the mod menu is exited. Used only in √bbs.ov2.
G()	Mod menu array dimensioned each time mod menu is entered, and removed from memory when the mod menu is exited. Used only in √bbs.ov2.
GA%	Number of game plays for use by Mod Menu. It is first defined in √bbs.init so that the variable killer routines in √bbs.ov2 do not remove it from memory.
H	Last selected drive number. H can be a number between 0 and 15 depending on what the last selected group of files was (e.g. Password File, System Files, Public Messages, etc).
H$	The last used drive init command.
H$()	The drive commands for your drive assignments as defined in SETUP (e.g. i0, ui, hm4 20 28).
HD$()	The custom message header information as read in when a user selects [R]Read Public Msgs

Var	Definition & Function
HM	Highest message number on the system as stored in √variables.
I	This is probably the most used numeric variable that doesn't have a specific function. It is used mostly as a general index variable for FOR/NEXT loops.
I$	This variable is a general use variable, although it is specifically designed as the variable in which typed information is stored when the input routines at line 300-310 are used.
I$()	General use array that should never be permanently dimensioned. The multi-scratch, multi-release, and multi-view functions in √bbs.xfer dimension this variable then execute a variable kill to remove it from memory when finished using it.
I%()	General use array which should never be permanently dimensioned in memory. The self-contained directory regenerate routine in √bbs.xfer dimensions this array for temporary use then discards it with the variable killer when the routine is over.
IC	This variable is set to 1 when your BBS is addressing an ICT hard disk system in chain mode. It is set to 0 when addressing all other drives or the ICT drive in non-chain mode.
II	Another general use numeric variable, which should never be used to store important information because the caller log routine will destroy any information stored in II.
J	General use variable which is often used like the variable I.
KK	If this variable is set, the caller online has created a DEFAULT MESSAGE.
L1	The access level that the BBS will automatically give a caller when their membership expires date (DD$) equals the system date (DA$) as defined in SETUP.
L2	Current caller's permanent access level. If you alter a caller's access level while they are online, LV is altered and will have an effect for the remainder of the call, but it is the value of L2 that is written back in their password file.
LB	This variable is used to store miscellaneous information about the user online. It is stored in the password file and keeps track of: page-pauser lines, 40/80 column setting, and character speed.
LC$	Last caller's membership name. If the last caller had an access level of 1, this variable will be "NEW USER". If the last caller was a Network node, then this variable will be the name of the node.
LD	Current caller's last called date as an ADN (Absolute Day Number). See the section on ADN's for more information.
LD$	Current caller's last called date in the form "MM/DD/YYYY". It is calculated from the variable LD.
LG	This is a temporary variable used by the caller log routines. You can also use this variable to store temporary information, but the data in LG will be destroyed if the caller log routines are used.
LG$	This variable is a temporary holding place for the information to be later stored in the caller log. When this variable is longer than about 200 characters, it will automatically be stored into an area of RAM that contains the caller log and the variable will be reset to "".
LK%()	This array stores the message links in the subject chains.
LM	The highest message number that a caller has read on this and previous calls. When a caller logs off, this variable will be assigned at least equal to the lowest message on the system. This allows the BBS to know if this is a caller's first call or a second (or more) call but they had not read any messages. On a callers first call, we don't want to scan for mail, etc.

Var	Definition & Function
LM%()	This variable array stores the status indicating if a level message exists for each level. These variables are assigned when the BBS in initially loaded or when the current time and date are adjusted.
LV	The current caller's (or the user's being edited in password maintenance) access level. Altering this variable will only have effect on the current call.
M	Used in √bbs.msgs to keep track of last message read.
M1	The system BPS rate as defined in SETUP.
M2	Used in √bbs.msgs to keep track of next message to be read.
M3	Indicates if are using Network, as defined in SETUP.
M4	Indicates if system shall adjust BPS rate to connect rate, as defined in SETUP.
M6	The default column (word wrap) setting defined in SETUP
M7	The default Fast-Garbage-Collect mode as defined in line 10241 of √bbs.init. See description of the !48 variable for more information.
MB	Minimum number of blocks allowed on the system before messages automatically start to cycle, as defined in SETUP.
MC	Maximum number of columns (word wrap setting) for the current user. Because it is used for the word wrap routines, it is set to the screen width minus 2. Thus, it is set to 38 for 40-column users and 78 for 80-column users.
MD	Maximum number of days that a caller's mail will be held before it is automatically purged, as defined in SETUP.
MF%()	This array stores the ID number of the user who posted each message.
MH	Keeps track of the number of pieces of mail held when a caller is reading their mailbox.
ML	Maximum number of lines per message as defined in SETUP.
MM	Maximum number of messages on the BBS as defined in SETUP.
MN()	This array stores the message number of each of the public messages.
MP	Maximum password number on the BBS as defined in SETUP.
MR	Number of message read so far when reading E-mail.
MR%()	This variable array keeps track of which messages have already been read or not.
MS	Number of messages to skip to read next piece of E-mail.
MT	Used in the multidownload routines to keep track of how many files have been sent so far.
MT%	Current speaker on/off setting in Network.
MU	Minimum number of blocks needed to allow uploads as defined in SETUP. The BLOCKS FREE message after the directory display or before an upload will automatically be adjusted by this value.
MX	Used in √bbs.msgs to keep track of next message to be read.
N	General Use.
N%()	Array holding status of each outgoing node.
NA$	Current caller's membership name.
NC%()	This array is used to calculate the number of new messages in each category when reading public msgs.
ND	Set when the date changes and is reset after the end of day routines are run.
NF$	Used to hold name and ID# of remote node.
NI%	Used only in Network overlays, it's the next open space in incoming node accounts file.
NN$	Your BBS system's name, defined in Net Setup.

Var	Definition & Function
NN%	Number of nodes, defined in Net Setup.
NN()	Prices for outgoing nodes: (x,0) = first 1000 bytes, (x,1) = each additional 100 bytes.
NU	Total number of callers who have called the BBS, as stored in √variables. NU is incremented each time a different caller calls the BBS (it is not incremented if the same caller calls twice in a row).
NU$	This variable stores a chr$(0). It saves memory to use NU$ instead of chr$(0).
OP%	Opening hour for the window of outgoing Network messages.
OV	This variable stores a pointer number that will be used after loading another overlay to determine what line number to execute. Line 5 of every program contains an 'onovgotoXX,XX,XX' command to make sure the proper routines are run based on the value of OV.
P	General status variable. If carrier is lost, p=255. If the caller types ↑p, p=1. Use "IF P THEN RETURN" after all your inputs to make sure the BBS will abort if carrier is lost or the caller types ↑p.
P$	General use.
PD	Caller's current number of blocks downloaded. See also C1, C2, and PU.
PH$	Current caller's phone number as stored in the password file.
PM	If this variable is a 0, the clock is in the AM range. If this variable is 1, the clock is in the PM range.
PN	Used only in Network overlays, holds number of public messages received when distributing incoming messages.
PR	This variable is set if you answer P to the "(S)creen or (P)rinter" question.
PS	Current caller's number of posts as stored in the password file.
PU	Current caller's current number of blocks uploaded. The variables C1, C2, PD, and PU are used to determine a callers download credit status with the following formula: • c2+pu*c1-pd
PW$	Current caller's password. It is saved in each caller's membership record in the password file.
QZ%	used as a flag when editing a message for when MCIs are turned off, and as a temporary holding variable.
R1	This variable is a temporary holding area for the sysops record number when using the Validate or Password Maintenance.
RA	Set when the user is reading ALL message categories.
RN	Current caller's record number (membership number). Also, during password maintenance this variable will be saved and will be reassigned to the value of the caller number being edited.
RN$	Current caller's real name as stored in the password file.
RU	The system's auto-release level as defined in SETUP.
RX%	See variable FU.
S	Used only in Network overlays, holds duration of Network call.
S1	Number of new callers for current day as stored in √variables.
S2	Number of calls today as stored in √variables.
S3	Number of uploads today as stored in √variables.
S4	Number of downloads today as stored in √variables.
S5	Number of messages posted today as stored in √variables.
S6	Number of public net messages holding as stored in √variables.

Var	Definition & Function
S7	Number of outgoing messages to send as stored in √variables.
SB	Set if you answered Y to screen blanking in SETUP.
SC	Set if you answered Y to daily caller log backup in SETUP.
SD$	Current download directory prefix. If you are not using multiple directories per drive, this variable will always equal "". Otherwise, it will equal "A" for directory A, "B" for directory B, etc.
SM	Highest message number read on this call. When a caller logs on, this variable is assigned the same value as LM, then as public messages are read, this variable is assigned the highest message number read. When a caller logs off, LM will be set equal to this variable, then stored in the password file.
SR	This variable is used to represent the status byte of the current I/O function. ST is automatically updated by the C64 after every I/O and SR is used to save the value of ST when needed.
SU$	This variable stores the subject of the last read message.
T	General use.
T$	Current time. You must **GOSUB1110** to update this variable.
T()	This variable array stores the daily time limits for each access level and the per call limits for AM and PM hours, as defined in SETUP.
T1	Number of invalid sign-ons since midnight last night as stored in √variables.
TA	This variable, along with TB, is used by the multi-download routines to keep track of where the BBS is in the list of files to send.
TB	See TA
TE$	See variable FU.
TI	The total time (in jiffies) that this caller has been online.
TI$	This variable is set to "000000" when a caller logs on and is automatically incremented every second. It is used to keep track of how long a caller has been online. It is also used to keep track of how long the BBS has been waiting between calls.
TM	Total minutes that this caller has for this call. TM-INT(TI*3600) represents the number of minutes that this caller has left for this call.
TS	Number of times the current caller has called. It is saved in the password file.
TT	This variable represents the time remaining for today less the value in TM. The total of TM and TT represent the total time the caller has for the rest of the day.
TX$	This is a special variable that Color 64 reserves for disk reads and typed input. The length of TX$ determines the maximum number of characters that can be read in from keyboard or disk. It should not be changed, except by the special system routines that need to lengthen TX$ to copy larger amounts of disk data.
TX%	See variable FU.
TY	See variable FU.
UD	Default upload directory number. This is always the first upload directory in SETUP with a Y in the upload status.
UX	This variable stores the UPLOAD DESCRIPTION status of your BBS as defined in SETUP. If it has a value of 1, your system is defined to be using upload descriptions. If it has a value of 0, your system is not using upload descriptions.
UX$	This variable is used to store the first few lines of the upload description when in the message editor in √bbs.xfer.

Var	Definition & Function
VE	This is used by Network to store the Network version number of the remote BBS node during a Network call.
VS	This variable is set if the variables have been saved and is cleared when another caller calls.
VX	There are several routines all using the same message editor and this variable tells the BBS where to go after it finishes editing the message (e.g. back to Read Feedback, Read Mail, Read Messages, or Posting a Message).
WD%()	This variable array stores the different caller access level's purge parameters as defined in SETUP.
X	General use.
X$	Only used in Network overlays, general use.
X%	Used only in Network overlays, holds current node ID#.
Z	Used only in Network overlays, general use.
Z$	General use.
Z3	See variable FU.
General variables used without specific definition: A, A$, B, B$, C, C$, I, I$, II, J, N, P$, T, X, Z$	

5.9 ADNs (Absolute Day Numbers)

The BBS system keeps track of the current date in the numeric variable DA as a value called an Absolute Day Number (ADN). An ADN is the number of days that have passed since January 1, 1800 (01/01/1800). This may seem inconvenient at first, but the Color 64 ML routines take care of all the necessary calculations, and you really don't even have to know that this is going on. You will never see an ADN when your BBS system is online, but you should know about this if you plan to program on your BBS system. ML functions @15 to @19 are all used to facilitate ADN calculations and conversions. For more information on what these functions do, consult section 5.1.8, "ML Function Summary".

The "current time" routine at line 1110 of an overlay is also used to calculate the current date when midnight has come. All it does is increment the variable **DA** by one and then use **DA** to get the value of **DA$** and other information as well. The variable **DA$** will hold the current date in the format "MM/DD/YYYY" where MM is the month (00 to 12), DD is the day (00 to 31), and YYYY is the year (1800 to 9999). The variable **D1** will be the current month (1 to 12). The variable **D2** will be the day of the week in the range 0 to 6 for Sunday to Saturday, respectively. The variable **D3** will be the day of the month (0 to 31). The variable D4 will be the year (1800 to 9999).

The string array variables **D1$** and **D2$** can be used to convert a date into plain English format. The strings **D1$(1)** to **D1$(12)** hold the names of the twelve months of the year January to December. The strings **D2$(0)** to **D2$(6)** hold the names of the days of the week from Sunday to Saturday.

Other variables which are stored in ADN format are **ED** (caller's membership expiration date), **LD** (callers's last call date), and **BD** (caller's date of birth). Also, when the user's information is read in from their password record, the variables **ED$**, **LD$**, and **BD$** will be set accordingly.

To find the difference in days between two dates, all you need to do is subtract the lower ADN from the higher ADN. This will be the exact number of days between the two dates. Also, to find out what the the current date would be after a certain number of days all you would need to do is add the number of days to the variable DA and you would have the new date.

There is also another function, @27, which can be used to find the number of calendar years between two dates. Using this function you can quickly find the age of someone using the caller's date of birth, by using the format **@27(DA,BD)**.

5.10 Using Commodore Disk Drives

To access a file on a Commodore compatible disk drive, there are five parameters that define how the computer communicates with a disk device:

- Logical File Number
- Device Number
- Secondary Address
- Drive Number
- Filename

The OPEN command uses all these parameters and here is its syntax:

OPEN <logical file #>, <device #>, <secondary address>, <file name>

The drive number is included in the file name. Here is an example of the use of all of these elements:

OPEN 9, 8, 2, "0:TEST"

Nine (9) is the logical file number, Eight (8) is the device number, Two (2) is the secondary address, zero (0) is the drive number, and "TEST" is the file name. I will discuss the use of each of these elements in the order of importance.

- Device Number
  Any device that plugs into the Commodore's serial port has a DEVICE NUMBER assigned to it, which allows the computer to distinguish between several different devices. A device can be a printer, plotter, disk drive, or any other device that can interface through the serial port. All the devices connected to the computer must have their own individual device number, or the computer will not be able to figure out which device is which.

Traditionally, disk drives may have any device number from 8 to 30. The Commodore 1541 disk drive, for example, can be customized to a device number from 8 to 15, but other devices such as the CMD HD can be set to be any device number from 8 to 30. In the example command LOAD"0:FILE",8 the device number is 8.

- Drive Number
  Along with the device number, disk devices are also accessed by a DRIVE NUMBER. The drive number was designed to allow one device to access more than one disk (as with the MSD-SD2 dual disk drive). The 1541 disk drive has only one drive permanently set to drive 0. In the example command LOAD"0:FILE",8 the drive number is 0, because the drive number precedes the file name separated by a colon. For Lt. Kernal hard drives, the drive number is also referred to as the LU (logical unit).

- Secondary Address
  To communicate with a disk drive, the computer must open up a communications "channel". When the OPEN command is used, this channel is referred to as the SECONDARY ADDRESS. Channel numbers 0 and 1 are reserved for loading and saving programs. Channel number 15 is used as the "command channel". All the other channel numbers (up to 127) are free for use with common communications. Another thing to note is that when opening more than one file on a single device, each file must have its own individual channel number, because this is how the disk drive knows which file you want to access.

  The command channel is used to send special commands to the disk drive. Some of the more common commands are the initialize, validate, scratch, and copy commands. Other commands are specific to certain devices. For example, the Lt. Kernal drive has the "LG" command which is used to get information about the hard drive. When opening the command channel, a file name is not used. Instead, the text you use as the file name is sent to the channel as a drive command.

- Logical File Number
  The computer also uses a LOGICAL FILE NUMBER to keep track of all open files. Once you open a file, the file number is what you use with the PRINT# and INPUT# statements. This sure makes things a lot simpler than having to refer to the device number, drive number, etc. every time you want to send or receive information to or from a file. The file number is for the computer's own use and has nothing to do with actual device communications. This means that the file number and secondary address are completely independent of each other, although programmers traditionally make the secondary address the same as the logical file number (i.e. OPEN 8, DV, 8, DR$+FI$).

- File Name
  Finally, the last parameter needed is the FILE NAME. Each of the files on a drive has a unique name, which makes things easier to keep track of, especially since a file name can be up to 16 characters in length. Note that the drive number is included in the file name in

the OPEN command. The drive number is first, followed by a colon, then the actual file name.

Hard drive devices, such as the CMD and Lt. Kernal hard drives, have many more features than the traditional floppy devices. Most hard drives divide up their storage into units called "partitions". Sometimes, the computer can access these partitions by using the drive number. Otherwise, special commands must be sent to the HD to select a partition for use.

Chapter 6, Mod Menu 2.0

Mod Menu was designed before the days of our "Super ML" upgrade. We noticed that it was very difficult to keep track of all the modules and games that we had running on the Sonic Temple, so we came up with the premise of the Mod Menu. Basically, this program does everything that all the menu programs do, except that the device numbers, drive numbers, file names, and etc. are all stored in one file which can be edited online. Plus, features like counting game plays and setting individual module levels were added. And all of this with a sophisticated editor allowing you to easily keep track of up to 250 games and modules.

I am making a distinction between a "game" and a "module" in this documentation. A game is a program overlay specifically used as a game for recreational purpose. A module is any program overlay (even a game). In other words, the word "module" can be used as a generic term for any program overlay, while "game" refers specifically to a game overlay. Mod Menu handles game and non-game modules differently, as you will see in the documentation. Also, I have chosen the generic term "stat" to refer to the record of a module's statistics stored in the Mod Menu file (e.g. a "game stat" or a "module stat"). Here is a brief run-down of some of the more interesting features of Mod Menu:

- Can handle up to 250 modules. From the Mod Menu prompt, a user just types the number of the module and presses RETURN.
- Mod Menu makes excellent use of the Super Variable Killer feature of Color 64. This prevents memory waste.
- Non-game Modules can have a maximum level as well as a minimum level (to prevent higher access users from using a program such as an auto-validator).
- The variable OV can be set for each individual module, allowing you to run multiple games/utilities out of single programs.

You can also customize the very programming of Mod Menu to make your system unique:

- The prefix for all the mod menu support files (i.e. the "√mod" prefix) can be set by changing just one line, allowing you to run multiple Mod Menus. Also, the color used in the caller log for Mod Menu activity can be customized also.
- The default device, drive, and init command can be altered for when adding new modules.

6.1 Installing the Mod Menu Files

The first thing you should do is make sure that the files "√mod edit menu" and "√mod sub menu" are already copied on to your System Files drive. These are the two help menus used in the Mod Stat Editor.

The first person to use Mod Menu must be the SYSOP (user number 2). All other users will get an error message indicating the parms are missing and the program will quit to the main prompt.

If the SYSOP does access Mod Menu first there will be an error message, but the program will ask if the SYSOP wants to set the main parameters. When the program is run for the first time, the module file will be created also. Here is the sequence of events: First, you will be asked if you wish to set the main parameters. If you answer "N", you will exit the mod menu. If you answer "Y", then the program will notify you that the module file is missing. When asked if you wish to create the module file, you can answer "N" to quit the mod menu, or "Y" to continue. The program will then ask you what you wish the maximum possible module number is to be. The file will be permanently alloted that number of slots to be used for module stats. Make sure you allow yourself enough room for expansion, for the program does not include a file expander utility.

If all goes well, the module file will be created (it may take some time if you chose a high maximum stat number). After this takes place, you will be allowed to set the main parameters. These are: The GAMES access level, the maximum number of plays per call, the credit cost per play, and the restrictions exempt level, defined in Table 33 below:

Table 33 - Module Menu Main Parameter Definitions

Main Parameters	Definition
Games Access Level	This is the minimum access level a user must have to play ANY game.
Maximum Number of Plays Per Call	This is the number of times a user can play a game. This does not mean each individual game can be played this many times, only that if any game is played, the total number of plays left is decreased.
Credit Cost Per Play	The credit cost per play is the number of credits that will be subtracted from the user's download credits when he/she plays a game. If the user has zero credits or a negative number of credits, then the person will not be allowed to play any games. Note: Setting either the maximum number of plays to zero or setting the credit cost per play to zero will disable that restriction for users.
Restriction Exempt Level	This is the minimum level at which a user is exempt from the above two restrictions. Setting this level to 10 means that no one is exempt from the restrictions.

6.2 The Mod Stat Editor

After the main Mod Menu parameters are set, they will be saved, and you will be sent to the module menu prompt. From the main prompt, you can type '√' and press RETURN to enter the Mod Stat Editor. When you enter the editor for the first time, it must regenerate the module index, an index that is used to allow faster editing of the module stats.

The module stat editor has been greatly enhanced since the original module menu program. The modules are now stored by number rather than by key. This allows a greater number of stats to be accessed and increases the speed of the program. A stat's number can range from 1 to whatever the maximum number stored in the parms is (up to 250). Each stat can be either a normal module or a game module, and the information stored in a stat will vary depending on its type.

All stats contain the following information defined in Table 34 below:

Table 34 - Stat Descriptions for Mod Menu

Stat	Description
Description	A string of 1 to 20 characters that should be a good enough description of the module to be contained in a menu.
Module Device Number	The device number of the storage unit that the module will be loaded off of.
Module drive number	The logical drive number of the module's storage unit.
Drive Init Command	The initialization command of the module's storage unit.
OV number	The number that the variable **OV** will be set to before the module is loaded. At the beginning of the module, an ON OV GOSUB statement can access the appropriate routine.
In non-game modules the following information is also stored:	
Minimum Access Level	The minimum level a user must have to access this module.
Maximum Access Level	The maximum access a user may have to access this module. This can be set to 10 to allow all users above the minimum to access the module. A user with an access level higher than this level will not be able to use the module. In game modules there is no minimum or maximum level, but rather the number of plays for that game is stored instead. When you add a new game stat to the Mod Menu, the number of plays will be set to zero.

6.3 Mod Stat Editor Commands

At most of the sub-prompts (such as the "add stat" command), you are allowed to enter "*****" (the asterisk character) to select the next qualified stat for that function. Thus, for example, entering "*****" at the "add stat" prompt would select the next empty stat. If you have set a high value for the

maximum stat (such as 250), then there may be delays while the index is being searched for the next available stat. If there are no more qualified stats, then a message will be issued, and you will be returned to the editor prompt.

Table 35 below defines the commands and their functions:

Table 35 - Mod Stat Editor Command Definitions

Command	Definition
L: **List Modules**	Lists all the occupied stats, including the number, description, type, status, and (if applicable) levels. The type is indicated by the presence or absence of an asterisk "*****". If one is present, the module is a game. The status is indicated by the presence or absence of an exclamation point "**!**". If one is present, then the module has been disabled. If the module is not a game, then the access level range is displayed.
V: **View Stats**	Allows you to view the information stored in the individual module's stats. (note that at most prompts you can also enter '**?**' to get a listing of qualified stats)
D: **Disable Module**	This does not erase a stat, but just prevents any user from accessing the module. This can be used to shut a module down for repairs.
R: **Re-enable Module**	The opposite of the above function, allowing users to access the module again.
A: **Add New Module Stats**	Allows you to add a new module to the file.
E: **Edit Module Stats**	Allows you to edit an individual module's information. It also allows you to move stats to a different slot.
X: **Scratch Module Stats**	Permanently erases a module stat. If the data in a stat is corrupted, then you can choose not to view the information before you scratch the stat. After the stat is erased, its slot can be used for a new stat.
Z: **Zero Game Plays**	Allows you to set the number of plays to zero for individual games or for all of them.
S: **Enter the Editor Sub-Menu**	The sub menu is a collection of routines that are not used as often and can be more potentially dangerous than the standard functions.

Table 36 lists the Editor's Sub Menu functions below:

Table 36 – Mod Stat Editor Sub Menu Command Descriptions

Command	Definition
R: **Regenerate module index**	The module index is used for quick access of the individual stats by containing the following information for each stat: whether it is a game, if it is disabled or not, and if it contains anything or not. The information is stored in compressed binary form in the string **g$(0)**. If you ever suspect that the index is not correct or get an INDEX INCORRECT error, then use

Command	Definition
	the regenerate command. The index is stored on disk for use during editing.
S: **Set Main Parameters**	Allows you to alter the main parameters described in the installation section.
V: **View Parameters**	Gives you an overview of the module menu. It lists the main parms, then tallies up the different types of stats. It then asks you if you wish the number of total game plays to be tallied (as explained next)
T: **Tally Game Plays**	Allows you to see the total number of game plays for all games stored in the module file. If you use this command before [V]iewing game stats, then the percent ranking and comparison is printed below the game information.
C: **Create Level Menus**	This command allows you to create menus for all the access levels. This considers the minimum and maximum levels and the game level. If no one has any access for a level, then the menu file is just scratched. Program lines 5450 and 5460 allow you to subdivide the modules into categories -- the headers of which can appear in the menu. In each line you will see three comparisons of the variables i and ii: • ii<x - x is the lower boundary of the division • i>=x - here x is also the lower boundary (as in 1 of a 1 to 50 range) • i<=y - here y is the UPPER boundary of the division (as in 50 of a 1 to 50 range) Following these comparisions is a print#9 statement that will print the header of the division if any modules lie in that range. The program is currently set to print "Online Games:" as a header for modules 1 to 99. The program will print "Modules:" as a header for modules 100 to 199. Note that if no modules exist in these ranges, then the appropriate header will not be printed. Study these lines and you will be able to alter them to use your own headers or add new ones (just pay attention to the difference between the variables II and I).
R: **Reset Module File**	Allows you to erase all the stats and start from scratch. A backup file is made before the actual file is created though.

6.5 Mod Menu Files

Table 37 lists the Mod Menu file names by their default names, so if you change the file name prefix variable **g$(2)** in line 2020 of the Mod Menu overlay, the file names will be different:

Table 37 - Mod Menu Files and their function

Command	Definition
√mod file	A relative file storing all the information and can range from about 5 to 125 blocks in size depending on the maximum stat number.

Command	Definition
√mod index	Module index used in the stat editor.
√mod parms	The main parms for the module menu.
√mod menu <#>	Menu file for access level <#>; example: *√mod index 1* for level 1 users
√mod index.bak	A temporary file created during the use of the editor.
√mod file.bak	Backup of the module file created when resetting the module file.

Keeping these filenames in mind, you should not set the length of the filename prefix to more than 7 characters. Otherwise, you may get disk errors.

The following menu options can only be accessed by user number 2 (the SYSOP):

- Scratch Stats
- Zero All Game Plays
- Set Main Parms
- Reset Module File

Also remember that if you use the ON OV GOSUB in a module, that the very next line must be the one that re-loads √bbs.ov2. Also, remember that a multi-overlay game usually requires **OV** to be set to 0 when loading the main overlay.

One note before continuing: There are some files with names that do not use the prefix set in line 2020 and are used in accordance with the mod stat editor. The fact that they do not use the prefix allows multiple Mod Menus to use the same help files. They are as follows:

"√mod ed sysops"	This file, if you wish to create it, should contain a list of user numbers for those you wish to have access to the mod stat editor.
"√mod edit menu"	This is a list of commands available in the mod stat editor and is displayed when '**?**' is entered from the editor prompt.
"√mod sub menu"	This is a list of commands available from the editor sub-menu.

6.6 Mod Menu Operation

Once a user selects the [*]Mod Menu command from the main prompt, the BBS program will display "Entering Mod Menu..." and they will be taken to the Mod Menu main prompt.

At the prompt will be displayed some information. Depending on the level of the user, the program may display the number of game plays left and/or the cost in credits per play. On the SYSOP's side, the name of the current caller and the current baud rate is displayed instead.

From the Mod Menu prompt, the user can do two things: The user can press "**?**" to see a menu of all modules they can use, or the user can type the number of the module to select it.

After a user selects a module, several things occur. First, if a module is a game, then the number of plays is updated. Then, regardless of the module type, the message "Loading <module name>..." will be displayed and the module will be loaded into memory using the stats entered in the module file.

Before a module is loaded, the BBS program sets up for a Variable-Kill. This means that when the module returns control to the "√bbs.ov2" program, which it must do, then all new variables added in the module program will be removed from memory. This ensures proper memory management of your BBS system.

Note that if you are running multiple Mod Menus, you should make sure that the module returns to the Mod Menu from which it was loaded. If this is not done, then users will get confused if they return to a different Mod Menu (with a different set of modules).

6.7 Customization

The Mod Menu program is already installed and ready to use in "√bbs.ov2" in the Color 64 package, but first you may wish to customize the program.

- Mod Menu Drive
  Line 2000 contains a GOSUB 481 which is the only drive selection GOSUB. This can be altered to choose a different drive to store the module menu files (e.g. h=12:gosub460 to use AUX 1).

- Mod Menu Files Prefix
  In line 2020, the variable **g$(2)** is set to the file prefix that is used for the entire program. Currently it is set as "√mod " (note the space included after the 'd'. This can be altered to allow you to run multiple Mod Menus on the same drive.

- Mod Menu Caller Log Activity
  Also in line 2020, the variable **g$(6)** is set as the color used in all the module menu caller-log activity. Currently is is set as light grey (C=8). This can be used to differentiate between multiple Mod Menu programs.

- Default Modules Drive Parameters

  Also, you can change the default module parameters used when adding a new module to the system:

- You can set the default module device number by altering lines 3250 and 3260.
- You can set the default drive number by altering lines 3280 and 3290.
- You can set the default drive command by altering lines 3220 and 3330.
- You can set the default OV number (explained later) by altering lines 3270 and 3280.

6.8 Creating Modules

Included in the Color 64 package is the program "xxx small". This small program is a skeleton overlay with just enough of the core BASIC code in it to allow modules to run. You can merge one of the many existing mods into this overlay to create a new module. Many of the existing modules may already have lines in the range 0 to 27999, which is the range of lines used by the xxx small overlay. If they do, then it may be necessary to remove the old routines from the module before merging in the new xxx small overlay. Because of this, if the module has any of its important main code in the line range of 0 to 27999, then it is possible you won't be able to use the following method to make it into a module that can be loaded from the Mod Menu. If you use just modules which have their code at lines 30000 or above, then you should not have any problems.

As you create each module, be sure to write down its description, device number, drive number, drive command, and filename for future use in filling the stat file. To do these modules you will need a good BASIC programming utility with a MERGE command, such as BAID 64. Here is how you alter each module:

- Merging in the XXX Overlay
    - Load the program into memory and locate the section of code that GOSUBs the module program. This code is usually located at line 5 and has a GOSUB that executes the module program. It sometimes has a remark telling what the program does. Here is an example:

      5 gosub32050:rem cardsharks!

      Write down this line number because you will be deleting this line. Now you need to ensure that the old XXX module is removed from the program. To do this, use a delete command to remove lines 0-27999 (this is why you cannot convert modules that have important lines in this range). Next, use a true-merge command to merge in the "xxx small" module.

- Editing the Module
    - Next, you must modify or add two lines: lines 1 and 5. Line 1 is the line that you use for re-saving the overlay. Just edit it so that it uses the correct filename and device number. If you forget to edit this line, then you run the risk of resaving the modified program over your xxx small module. Be sure that you do not remove the REM from line 1. Line 5 is the line that will GOSUB the module routine. Type it in like this:

5 gosubXXXXX:gosub489:.dr$+"√bbs.ov2*",dv

Use the line number that you wrote down in place of XXXXX. This line will GOSUB the game/module, then load √bbs.ov2 (the Mod Menu) so that a variable kill can take place.

- You can also modify the program to run several mods out of one program. To do this there are two methods:
  - You can use one of the many small menu programs available to add a menu to the module. Using this method, you would have only one stat in the mod file, which users could select to access your custom menu.
  - Another method involves taking advantage of setting the **OV** variable through the mod menu program. Line 5 in the module would be an ON/GOSUB command in this format:

    5 onovgosubXXXXX,XXXXX,XXXXX | Where *XXXXX* are the beginning line Line numbers for each of the mods in the overlay.

    Then you would add line 6 to look like this:

    6 gosub489:.dr$+"√bbs.ov2*",dv | This is necessary so that the Mod Menu program will be reloaded when any of the small mods has ended.
  - Next, you would add a separate mod stat to the mod file for each of the individual mods in the overlay. For each one, you would set the **OV** setting to 1, 2, 3, or whatever number corresponds to the position of the line number in the ON/GOSUB command in the module overlay. For example, if line 5 in the module looked like "5 **onovgosub30000,31000** ", then you would set **OV** to 1 in the stat editor to use the mod at line 30000, and you would set **OV** to 2 to use the mod at line 31000. This method makes it look as if all the mods are separate although they are running out of a single overlay. For an example of this, see the sports picks module included with the Color 64 system.
- Once the changes are made, save the module program to disk. Again, be sure that you write down all the information for the module, including the **OV** setting if necessary. Writing the information down will greatly reduce the amount of time it takes to add the mod stats to the Mod Menu file.

Chapter 7, Specific System Requirements

There have been many products produced for the Commodore 64 and 128 computers, many of which have special features and/or requirements that can be accounted for in Color 64. In the following sections, I will try to describe how to best set up your system with various hardware configurations.

Things to remember:

- Some systems require that special routines be merged into the main overlays and other programs; these routines have been included as separate merges on the included diskettes. Each of the following sections will describe which systems need merges and how to use them. All merges have been designed so that they can be combined with each other without interfering with any of the other merges. You must have a utility program (such as BAID) that allows you to merge BASIC programs before you can use these files. Lt. Kernal users can use the built-in utilities for this purpose.
- Remember to do all the mandatory changes BEFORE you start your system up and remember to make the changes to the "√bbs.xxx" and "xxx small" overlays before using them in modules and overlays.
- Do not use the original Color 64 disks to save the changed overlays. Otherwise, if you change your system configuration later, you will not have the original files anymore. Make a backup of you Color 64 disks to reduce the possibility of losing everything.

7.1 Lt. Kernal Hard Drive Systems

Xetec's Lt. Kernal Hard Drive system is an excellent choice to run your BBS system on. It supports up to 9 different partitions of disk space called Logical Units, and each of these units has 15 "users" which are like subdirectories. This can mean you can have up to 135 seperate file directories. Color 64 will NOT be able to use the Lt. Kernal and RAMDOS at the same time, if you have an REU in configuration with the HD.

To use the Lt. Kernal HD on the 64, you must have the HIRAM connector hooked up, as described in the Lt. Kernal manual. If you do not have the HIRAM connection, then the system will not work correctly. If you are using a Commodore 128 in 64 mode, then the ribbon cable connecting the computer and the Host Adaptor takes care of the HIRAM signal.

Another thing you need to do is run your CONFIGURE program and make sure that the NMI TRAP setting is off (set to 0) for the Commodore 64 mode of the Lt. Kernal. If this is not set to 0, then your users may experience "line noise" when calling your system.

And finally, you need to make sure that you answer "Y" to the Lt. Kernal question in your boot-maker program. If this is not set to "Y", then modem communications will be garbled.

Color 64 can access any configuration that you may have in mind through the "ldlu" and "i" drive commands. The "ldlu" command's syntax is **l<*device*><*LU*><*USER*>**. For example, to access LU 2, USER 5 the command would be "l825", assuming you want to use device 8. For users 10 through 15, you use the hexadecimal equivalent of A through F. For example, LU 3, USER 11 would be "l83b".

For LU's 2 through 9, you need to add an "**i**" command to the drive initialization command in setup. If you wanted to use LU 3, then the command would be "i3". When both commands are put together, the whole command would look like "l83b!i3", if you wanted to use LU 3, USER 11. Notice that you separate individual disk commands with an "**!**" exclamation point.

7.1.1 Faster Disk Access

If your Lt.Kernal HD uses DOS version 7.1 or greater, then you can achieve faster disk access in some of the overlays by merging the following files into the specified overlays. These merges are optional, so they are not absolutely required for your system to function normally:

```
lkf.init  -> √bbs.init
lkf.msgs -> √bbs.msgs
lkf.xfer  -> √bbs.xfer
lkf.ovl   -> √bbs.ovl
lkf.nw1   -> √bbs.nw1
lkf.nw2   -> √bbs.nw2
```

These routines replace the standard free blocks routine with a custom Lt. Kernal command. The routines assume that your Lt. Kernal is being used as device 8. Remember: The "lkf" merges are for systems that use DOS version 7.1 or greater!

7.2 ICT Series Hard Drive Systems

Color 64 BBS fully supports the InConTrol Data Chief HFD20 hard disk drive system. All you need to do is use "hm4 11 22" as the drive command for whatever files you want stored in a chain defined as starting at partition 11 and ending at partition 22 or h10 for files you want stored in an individual partition 10. The only files that you can put in chain mode are Public Messages Help Files, Text Files, Uploads and Downloads. System Files, Private Mail, Caller Logs, Password File, Boot Files and Program Files must be assigned individual partitions. And if you are using the Epyx Fastload cartridge, you must put your program files on H0 (the built-in floppy) or you will experience intermittent system lockups.

The ICT has been tested with the Burst Mode Rom from Chip Level Designs and it works great! This ROM allows your C64 or C128 to access your hard drive in its burst mode of operation, speeding up all disk access by 2 to 10 times. Also, if you have a 1764 or 1750 Ram Expansion module, there are

Color 64 modules available that will let you do all your ICT maintenance from within Color 64. One such module is included in this package (see below after merge information).

As far as download directories, I recommend you answer "N" to the question "Multiple Directories Per Drive" in SETUP. As far as Color 64 BBS is concerned, each different chain looks like a different drive. Running one directory per drive/chain has the advantage of not having to rename files before they can be downloaded too. It will work either way, this is just my recommendation.

One last comment. The scan for new downloads is a little slower on the ICT drive than on other drives. That is because it IS necessary to read two different files in 2 different partitions. So, we are having to open and close a lot more files than on the CBM drives. But all in all, it does work well and is still fast enough in my opinion (especially if you have the burst mode ROM in your computer).

7.2.1 ICT System Merges

If you run a system with an ICT (In ConTrol) DataChief or MiniChief hard drive, and you plan to use chained partitions, then you need to merge the following files into the specified programs. If these merges aren't made, then your upload/download directories will not function properly on chained partitions:

ict.xfer -> √bbs.xfer
ict.ovl -> √bbs.ovl

7.2.2 The ICT Utilities Module

Also included in your Color 64 package is a version of the popular "√bbs.ict" module for ICT Hard Drive users. It adds a multiple-chain compress routine, auto-return to the waiting for caller screen, busy modem and a scan upload space routine that will scan all upload directories that are chains on an ICT and show you the total blocks free and the max blocks allowed on an upload. This last routine can be used to keep an eye on your upload space on the ICT and to determine when you need to compress your chains.

For those that aren't already familiar with this program, it is a special program just for those persons using an ICT hard disk drive, a 17XX series Ram Expander, and RAMDOS on your Color 64 BBS. This routine allows you to copy files using a pattern match from any one partition or chain to any other partition or chain or drive. It also allows you to compress chains, busy the modem, scan all upload space, etc. all from within Color 64 BBS. Note that when the program is doing a copy operation, you will see data flash across the top of the screen as the copy is proceeding, due to the ML routine.

All you need to do is to modify your "√sys.ramove" file to copy "√bbs.ict" into your ram module. If you put your √bbs.ict module with your Program Files, this can be accomplished by adding the following line anywhere in the line range 7100 to 7499: **7XXX &(8)="√bbs.ict":>310**

Otherwise, if your √bbs.ict module is located elsewhere, you will need to follow the instructions for adding to the script program in the section on the "√sys.ramove" program.

Next, you will need to modify your √bbs.xfer overlay so that it will load √bbs.ict when you type "+" from the DOS prompt. Just merge "ict.load" into √bbs.xfer and this will be taken care of for you. Once this is done, typing "+" and hitting return at the DOS prompt will send you to the ICT module.

Note: The busy modem feature does not work on the Commodore 1670 modem - it does not support the ATH1 command (or any other command that can be used to busy the modem).

Remember, the √bbs.ict mod is for ICT hard drive systems with ram expanders only.

7.3 CMD Hard Drive Systems

The HD series of hard drives from Creative Micro Designs is another excellent choice for running a BBS system. You can allocate your disk space in almost any configuration you can think of, dividing the storage up into partitions. The CMD HD's Native Mode partitions also support subdirectories, which allow you to dynamically allocate space to individual file directories.

The only "requirement" for using the CMD HD might be this: If you plan to have your Program Files on the CMD, then I strongly suggest you invest in the JiffyDos fast disk system. Or, since the CMD HD is fully compatible with any of the Commodore 17XX series REU's, you could also configure your BBS system to use RAMDOS for faster loading. Also, you could also invest in CMD's RamLink device, which would give you the benefit of fast parallel operation.

The CMD HD should NEVER be used along with any type of fastloader cartridge, unless the designers of the fastloader specifically say it is compatible with the CMD HD. It has been found that the use of some fastloader cartridges can corrupt the data stored on your CMD HD.

See the section "Drive Initialization Commands" for information on how to select partitions and subdirectories on your CMD.

7.3.1 The Real Time Clock

SYSOPs using a CMD HD (or a CMD RamLink with Real Time Clock) should merge "cmd.init" into "√bbs.init" to take advantage of the built in clock on the CMD device. MAKE SURE the clock is set to the correct time before starting the system up with this merge in place. The √bbs.init program will now read the time off of the CMD device upon booting up. After making this merge you may delete the following lines from √bbs.init: 24030, 24190, 24210, 24220, 24230, 24250.

Also, there is a GOSUB481 located in line 24007. This is meant to select the CMD device to input the time. Since GOSUB481 selects the system files drive, this may need to be changed if your system files are not stored on your CMD device.

7.4 CMD RamLink Systems

Color 64 works great with this super device from Creative Micro Designs. If you also have a CMD HD, you can use the parallel cable and have your overlays load in a flash.

If you have a 17XX series REU from Commodore and you do not wish to use it as extra RAM on the RamLink, you can also use it with RAMDOS. Just follow the instructions in your RamLink manual for accessing the REU "as is" (i.e. "direct" mode), then follow the regular Color 64 installation instructions for using RAMDOS.

The SwiftLink RS-232 interface from CMD can be used with RamLink and should be plugged into the Pass-Thru port of the RamLink. However, the SwiftLink cartridge will NOT function if the RamLink is in "direct" mode, because the Pass-Thru port would be disabled.

If you also have RamCard and RAM installed in your RamLink, you can also run your Color 64 programs on it if you wish. You just need to make sure that you have disk backups of all files that you put onto the RamLink. Because the √bbs.parms are read in from the Program Files drive, you need to make sure that your √bbs.parms file is in the proper place.

You can also run any other file groups on a RamLink with RamCard, but you should do so only if you have the battery backup option. If you do not have the battery backup and power fails, all files on your RamLink will be lost. Even if you do have the battery backup, it is suggested that you only store "static" file groups like Help Files or Text Files on RamLink.

If you have the Real Time Clock option on your RamLink, you should also read the above CMD HD section for information on the "ict.init" merge which will allow your BBS to automatically read the time and date from the CMD RamLink.

7.5 Fastloader Cartridge Systems

Some fastloader cartridges may require some special merges for the BBS program to work properly during LOAD operations. To check if you need the following merges, turn your computer on with the fastloader in place (and active), and type the following line and press RETURN: **PRINTPEEK(817)**

If the number that is printed is 223 (two hundred twenty-three), then you need these fastloader merges, which need to be merged into the specified overlays:

fst.init -> √bbs.init

fst.xfer -> √bbs.xfer
fst.ovl -> √bbs.ovl
fst.nw1 -> √bbs.nw1

Merge "fst.ovxx" into the following programs:

- √bbs.msgs
- √bbs.ov2
- √bbs.ov3
- √bbs.nw2
- √bbs.xxx
- xxx small

7.6 The Skyles Flash! 1541 Interface

The Color 64 system will automatically detect if you are using the Flash! interface. The system will only take advantage of the interface on device 8, however. As such, you will need to make sure the 1541 to be used with the interface is device 8. To fit all of your program files on the 1541 disk, you will have to leave out a couple overlays, either the "√bbs.ov2" and "√bbs.ov3" overlays, or the "√bbs.nw1" or "√bbs.nw2" overlays.

7.7 The Avatex 1200 Low-Cost Modem

If you use the Avatex 1200 modem (not the Avatex 1200 HC or the Avatex 2400) on your system, then your overlays need to be changed because the Avatex 1200 is not completely Hayes-compatible. Once your system has been modified, you will not be able to use the overlays with a true Hayes-compatible modem, so MAKE A BACKUP of all the overlays to be changed. Merge the following files into the specified overlays:

ava.init -> √bbs.init
ava.nw1 -> √bbs.nw1

Merge "ava.ovxx" into the following files:

- √bbs.msgs
- √bbs.xfer
- √bbs.ovl
- √bbs.ov2
- √bbs.ov3
- √bbs.nw2
- √bbs.xxx
- xxx small

7.8 Using Color 64 BBS With A 2400 Baud Modem

Color 64 BBS supports the Hayes Smartmodem 2400 and many good compatible modems. Without using the SwiftLink interface, you should find that your C64 and Color 64 can handle 2400 baud file transfers with very few errors.

Here is how I recommend you configure your modem to set up your system for 2400 baud.

- Hayes Smartmodems come from the factory with DTR and CARRIER DETECT forced true. The 1200 baud modems had switches while the 2400 modems don't. Instead, they have a series of commands that are entered from a terminal program then stored in permanent memory. The following commands are what I recommend you type to set up your modem. Just load a simple ASCII terminal program, and at 300 baud type the following:
    - **AT &D3** (some compatibles prefer AT&D2). Press RETURN and then enter the next command:
    - **AT &F &C1 &S0 X1 S0=0 M0 E0**
    - Now switch your terminal to 2400 baud and type: **AT&W** (this will write this to your modem's permanent memory)
    - AFTER you have performed the above, you need to run SETUP and, in the MODEM INIT command, enter: **ats7=15s10=30**

  The first step above presets your modem to:
    - Disconnect and reset to power on configuration when DTR is dropped
    - Monitor carrier lost and disconnect if false
    - Force DSR true
    - Not auto-answer the phone
    - Turn off the speaker
    - Turn echo off; and
    - Save this configuration in permanent memory within the modem.

  After typing the ATE0 command, the modem will quit echoing to the screen. This is a more reliable way to send "at" commands at 2400 baud because in a few cases echo mode can cause garbled communications. Anyway, the BBS does not need echo and when you want to auto-dial another BBS, just use the autodialer built into the term.

  The modem init command in SETUP will set the modem to wait for a carrier for 15 seconds. I found that 30 seconds was too long and often the telephone companies off-hook attention signal would confuse the modem to think it had carrier. Besides, 15 seconds is plenty long to wait for a carrier. This s7 register is not saved in permanent memory when using the AT&W command, so we need to enter it here in SETUP. The second part of the modem init set the modem to wait the maximum amount of time before disconnecting when carrier is lost. This is to help somewhat if there is line noise or if the caller is using some special service like call-waiting.

7.9 Using Color 64 with High-Speed Modems

If you have a SwiftLink RS-232 cartridge, you can take advantage of a high-speed modem, communicating at rates of up to 38,400 BPS. There are many different types of high-speed modems, some of which may not be compatible with others because of diverse "standards" set down by different companies. If you plan to use a high-speed modem, you should invest in one that can handle many different communications protocols. Some of the more common protocols are V.32bis and V.42bis.

Also, if you plan to take advantage of the error correction and data compression features of a modem, you need to check the modem to see if you must adjust the computer-modem BPS rate to be the same as the modem-modem BPS rate. In many cases it is better if you can set the computer to modem communications to the highest rate, no matter what the modem-to-modem communication rate is. The "Adjust BPS" question in setup determines how the computer will react when a call is received. If you answered "Y" to the question, then the computer will always match its own BPS rate to that of the "CONNECT" message. If you answer "N", then the computer to modem rate will remain constant and the modem will handle the necessary conversion.

Chapter 8, The √sys.RAMOVE Script Program

If you want to have your program overlays run off a RAM expander, then the quickest way to transfer the programs would be to LOAD them into memory and then SAVE them to the REU RAM-disk. This is exactly what the "√sys.ramove" program does, because it is a BASIC "script" program that uses the Script-Merge utility designed by myself. A script is simply a set of instructions that are executed as if you typed them in from the keyboard. Since a script program is stored in a protected area of memory, it can use LOAD and SAVE operations just as you would from BASIC's READY prompt. Script-Merge also has a set of specialized commands that allow it to function like a program. It has its own variables, branching commands, and a merge command built in. I will cover the information which is important to changing your "√sys.ramove" program.

8.1 The File Transfers

The file transfers in the script are divided into sections. Each section is responsible for copying a specific set of files to the RAM expander.

- The BASIC Boot Files
  Lines 2100 to 2999 are dedicated to copying the necessary Boot Files to the REU. Of this range, lines 2100-2499 are for those programs which can be loaded into memory with the BASIC LOAD command. This means that the programs will be loaded from disk and then saved to the RAM-disk, using the BASIC LOAD and SAVE commands. The script commands in this range are in the following format: **&(8)="program name":>XXX**

  The "**&**" character represents one of the built-in variables in Script-Merge and has elements like a BASIC array. Thus the first part of this command assigns the name of the program to the variable **&(8)**, which will be used as the source file name in the transfer. The "**>XXX**" command is the equivalent of BASIC's GOSUB command, where *XXX* is the line number to branch to.

  The routine starting at line 310 of the script is one version of the copy routine. First, it sets the destination name to the overlay name plus a "**.**" period character. Next, the routine tacks an "*****" asterisk onto the end of the source name. The "**.**" and "*****" characters are used so that the script will not have to know the actual overlay revision number of a program to copy it. For example, if **&(8)** was set to "√bbs.ovl" the source name would end up "√bbs.ovl*" and the destination name would end up "√bbs.ovl." Thus, no matter what the version number of √bbs.ovl was, it would still be pied. You will notice that at line 2100, the "copy with version number" routine is used to transfer the √sys.loadml program.

  Line 300 marks the beginning of another version of the load routine. This routine does not change the source name, and sets the destination name the same as the source name.

Thus, this routine is used for programs that do not have a version number suffix. This is used for the +ram.bbs and +ram.reboot programs in script lines 2110 and 2120.

- The Non-BASIC Boot Files
  Lines 2500 to 2999 are reserved for copying the boot files which cannot be transferred via the BASIC LOAD and SAVE commands. This range uses another copy routine located at line 510. This routine differs from the previous ones in that it is used for non-BASIC programs and sequential files. Before calling the routine, **&(7)** must be set with the one character file type. The standard types are "p" for PRG and "s" for SEQ. Relative files cannot be copied by this script program. This routine also works like the one at line 310 in that it copies files with a version number. Line 2500 sets the type to "p" for PRG and then lines 2510 and 2510 copy the appropriate ML file to the REU.

- The BASIC Program Files
  Lines 7100 to 7999 are used to transfer the necessary Program Files to the REU. Of this range, lines 7100 to 7499 are to be used for programs that can be transferred via LOADing and SAVEing them from BASIC. The copy routine at line 310 (copy with version number) is used for the √bbs.init, √bbs.msgs, √bbs.xfer, and √bbs.ovl overlays.

  For the remaining overlays the routine at line 210 is used. This routine works much the same as the one at 310, except it checks to see if the file exists before attempting to copy it. If the file does not exist, then the program will not attempt to copy it. If the file does exist, then the transfer will be completed normally. This is used for the overlays which the SYSOP may or may not have chosen to use on the BBS system.

- The Non-BASIC Program Files
  Lines 7500 to 7999 are to be used for copying non-BASIC programs or sequential files. The type is set to "p" PRG at line 7500 and will be used for all following transfers until it is changed again (the stock script does not copy any sequential files). The routine at line 510 (copy file with version number) is used for the remaining files.

8.2 The Built-in Copy Routines

Table 38 lists a summary of all the copy routines available in the √sys.ramove program:

Table 38 - √sys.ramove Copy Routines

Line	Description	Which Variables to Set
Checks if file exists first:		
200	BASIC file, no version number	&(8)=source & dest. name
210	BASIC file, with version number	&(8)=source & dest. name, w/o suffix
220	BASIC file, no version number	&(8)=source name, &(9)=dest. name
400	Non-BASIC, no version number	&(7)=type, &(8)=source & dest. name
410	Non-BASIC, with version number	&(7)=type, &(8)=src & dst, w/o suffix

420	Non-BASIC, no version number	&(7)=type, &(8)=source, &(9)=dest.
No check if file exists:		
300	BASIC file, no version number	&(8)=source & dest. name
310	BASIC file, with version number	&(8)=source & dest. name, w/o suffix
320	BASIC file, no version number	&(8)=source name, &(9)=dest. name
500	Non-BASIC, no version number	&(7)=type, &(8)=source & dest. name
510	Non-BASIC, with version number	&(7)=type, &(8)=src & dst, w/o suffix
520	Non-BASIC, no version number	&(7)=type, &(8)=source, &(9)=dest.

As you can see, almost any possible type of transfer need is taken care of. The script program as-is will transfer just the essential files to the REU which are necessary for BBS operation.

8.3 Adding to the Script

Adding to the script to copy additional files is as easy as using the same methods already used in the script program. Just look at the lines where the regular files are copied and use the same procedures. Here are some general guidelines though:

- Additional Overlays
  If you have extra program overlays that are stored with your Program Files, then you should have no problem including them in the script. Just add another line in the range 7100 to 7499 in the following format: **7XXX &(8)=".overlay":>310**

  This example would copy the file "√overlay*" from the Program Files and store it on the REU as "√overlay.". If you wish to set the exact overlay name, you will use ">300" instead of ">310". Then nothing would be added to the file name in **&(8)**, the source and destination name. If you wish to set different source and destination names, then you would use ">320" after setting **&(8)** with the source name and **&(9)** with the destination name.

- Overlays Not in Program Files
  If your extra games or overlays are not stored with your Program Files, then you will need to add extra sections of code in the range 3000 to 6999. The first thing that would need to do is to set the device and drive number parameters. The current device number is stored in the variable **%(1)**, and the drive number prefix (e.g. "0:") is stored in the variable **&(1)**. Then you would open the error channel and send the appropriate drive initialization command to the disk device with your extra overlays. After that you would use the >300, >310, or >320 copy routines like normal. Here is an example:

```
3000 :set device and drive
3010 %(1)=8:&(1)="0:"
3020 :
3100 :open error channel
3110 close15:open15,%(1),15,"i0"
```

```
3120 :
3200 :copy files
3210 &(8)=".game.emp1":>310
3220 &(8)=".game.emp2":>310
3230 &(8)=".game.emp3":>310
```

Note that you can precede a line with a ":" colon and it will not be executed in the script.

- Sequential Files
  If you need to copy sequential files, then you would follow the same basic procedure, but you would need to use the >400, >410, or >420 routines to copy the file. Here is an example (continuing the above example):

```
3300 :set file type
3310 &(7)="s"
3320 :
3400 :copy files
3410 &(8)=".emp.menu1":>400
3420 &(8)=".emp.menu2":>400
```

Just remember when adding to the script to put the additions in the 3000 to 6999 range, so that the last thing that is done is always the Program Files (lines 7000 to 7999). This will ensure that the script will be ready to boot the BBS.

One more thing: As you will see in the following rules, the script must not exceed 30 disk blocks in size, or the script will not work.

8.4 Script-Merge Rules

The first rule that you should remember is that once a normal BASIC command is executed in a script line, then the rest of the line will be interpreted without the aid of the additional script commands. For example, if this line were in a script: **1000 &(8)=".file":print"hello":&(8)=".file2"**, an error would result when the second **&(8)** were reached. This happens because Script-Merge recognizes that you want to do a regular BASIC command, so it copies the rest of the line to a special location and just jumps into the regular BASIC execution routines. Thus, you can begin a line with script-merge commands and then switch to normal BASIC, but once performed, the remainer of the line will be "blind" to script-merge command content. The next line in the script will be able to use the special commands again. One exception to this is the variables, which are always accessible from anywhere.

8.4.1 Script-Merge Commands

Table 39 provides the list of the special Script-Merge commands:

Table 39 - Script Merge Command Listing

Char	Command	Syntax/Description
@	Print Text	Just like the BASIC PRINT command
*****	Wait for Key	Works just like PRINT, but then waits for key
.	Input Text	Works just like PRINT, but inputs line afterward
,	Delete Lines	,line# [-line#]
/	Merge File	/<file name>
#	GOTO	#line#
>	GOSUB	>line#
_	RETURN	
]	Enter Line	]line# <basic text>
.	IF	.expression:<commands if true>
%	Set %() var	%(0..9)=<numeric expression, -32768 to 32767>
&	Set &() var	&(0..9)=<string expression, max 16 characters>
‘	Set ‘() var	'(0..9)=<string expression, max 16 characters>
-	END	

8.4.2 Script-Merge Functions

Table 40 lists the extra functions added by Script-Merge. They can be used in any expression if Script-Merge is active:

Table 40 - List of Script Merge Functions

Chars	Description
%(x)	Numeric vars %(0) to %(9). Can store numbers from -32768 to 32767.
&(x)	String variables &(0) to &(9). Max string length is 16 characters.
'(x)	String variables '(0) to '(9). Max string length is 16 characters.
$	Returns input from last "*" or "." command
!(x)	Returns 0 if line x doesn't exist, or non-zero if line x exists

8.4.3 Script Size Limits

Script-Merge scripts have a size limit. The BASIC program containing the script can be a maximum of 8000 bytes large. A good idea would be to make sure that the size never exceeds 30 disk blocks, which is well within the maximum.

Chapter 9, Color 64 Update Information

SYSOPs who are now running Color 64 version 7.37 or the Super ML upgrade may be interested in how much has changed in the new version. Since Color 64 version 8.0 is an upgrade from Super ML, I have divided the update information into two sections so that SYSOPS of both types of systems can see what is different from their BBS system.

9.1 Changes from Version 7.37 to Super ML

Here are the changes that were part of the Super ML upgrade (the precursor to version 8.0), that are included in version 8.0:

- Some of the added features include: a faster sequential file reader, customizable message headers, faster word wrapping, word wrap in chat mode, built in page-pauser, and added text editor features such as centering, word delete, better uppercase/lowercase switching, etc.
- The system now supports ANSI color and graphics conversion built into the system. Also, an improved ASCII conversion was installed.
- MCI commands have been added that allow you to put special commands right inside of sequential file messages. Some of the commands included are rainbow mode, slow mode, and a Variable MCI command.
- The caller log is now stored in a sequential file, which is maintained automatically. Options in SETUP allow you to define caller log parameters such as the maximum size, and whether you want daily backups.
- Several new parameters have been added to SETUP which allow you to customize more of your system. Included is the ability to set levels for features which previously had access levels hard coded into the system.
- The drive initialization command routines were revised to allow you to send multiple drive commands to a device by separating commands with an "!" exclamation point character.
- Programming features were added to both enhance BBS operation and to allow overlays to be reduced in size. Features include an IF/THEN/ELSE statement, extra commands, and a new set of functions that increase the power and speed of BASIC.
- The Wait-For-Call (WFC) screen has been moved into a sequential file named "√wfc". The included file is a very good example of the use of the Variable MCI command built into version 8.0. Also, the user statistics displayed at logon have been moved into a sequential file named "√logon stats".
- The "Graphics On/Off" function is no longer located in every overlay. This function has been moved entirely to √bbs.ovl, where it has also been modified to accommodate Commodore, ASCII, and ANSI users.

- The "Change Baud Rate" function has been replaced with the "Edit User Stats" function. The command now asks the user for information regarding their terminal setup, such as screen width and screen length. This function resides only in the "√bbs.ovl" overlay.
- The Application Plus mod has been updated and built into the BBS system. This modification allows the SYSOP to fully customize the look of the application and the information acquired through it.
- Mod Menu has been included with the system. This program allows online games and modules to be tracked through a single menu, storing all the information (such as the drive and filename) in a relative file. An editor is built in that allows modules and games to be easily added to and removed from the menu.
- The system now comes with Network 64. The BBS program has been upgraded to take full advantage of Network.

9.2 Changes Since Super ML

Here are the changes made since the release of Super ML:

- The system is now compatible with both the SwiftLink RS-232 interface and the Schnedler Systems TurboMaster CPU.
- The records stored in the password file have been changed and expanded to store more information. Along with the regular information, the records now store a real name, phone number, birth date, and address. Also, the membership name can now include lowercase letters and some non-alphabetic characters.
- The date system has been changed so that it is accurate down to a single day. The date is now in the general format "MM/DD/YYYY", so that the year is now a four-digit number. Also, the system is now capable of converting the date to the long english text format (e.g. "Thursday, December 23, 1993").
- The latest version of Mod Menu, version 2.0, has been included with the system and built into √bbs.ov2. This new version of the program can support up to 250 modules and adds new features such as charging credits for game plays. Those who do not want to take advantage of Mod Menu's advanced features can use the included "√stock.ov2" overlay.
- The drive init command routine has been revised yet again. Now you can send as many commands as you wish (Super ML was limited to two) by separating them with an "!" exclamation point character. Also, the "i" command can now be used to set the drive number stored in DR$, which means that Lt.Kernal users can now make use of LU's 2 through 9.
- The SYSC(x) commands have been replaced with commands that are automatically installed by the ML. The commands are now in the format ".XX" where XX is two digits. Also, the PEEK(C(x)) references have been replaced with a set of "variables" in the form "!XX" where XX is two digits.
- The USR(x) functions built into Super ML have been replaced by the set of "@X" functions, where X is the number of the function. There are now a total of 31 different functions.

- The system can now be configured to use any device as the boot device, and included are utilities which will create custom boot programs based on the configuration of your system. Disk swapping prompts will now be displayed only if your system is configured for floppy drive use.
- A new system for transferring your overlays to a Ram Expansion Unit has been created: a fully editable script program that allows you to customize every aspect of the REU transfer.
- An upgraded version of Plusterm has been included with the system. This version allows you to take advantage of higher BPS rates available through SwiftLink, and fixes or modifies other features of the original Plusterm programs.
- The system now includes Network version 1.26a. This is an upgrade from the original Network v1.24 add-on, and fixes some of the features of version 1.26. Also, Network can be completely switched in or out through SETUP; the Network overlays don't even need to be in your Program Files if Network is switched out!
- The Application Plus mod has been revised yet again and is the standard application routine on the system. It now allows you to acquire the new password record information such as the real name and address.
- The ANSI conversion routines have been modified to support 15 ANSI colors and are more compatible with the various ANSI terminal programs.
- The directory regenerate function in √bbs.xfer has been replaced with a faster routine that can handle any number of files, even large 400+ file directories. Plus, the directory update functions will automatically be able to handle file sizes over 999 blocks.
- The "add-drive" mod is now built in, adding 3 auxiliary file groups to the drive setup.
- The "Edit DL Description" command has been changed to "View DL Description" and allows callers to view multiple descriptions.
- The Scratch DL and Release DL commands now support multiple scratching and releasing.

9.3 Upgrading Previous Color 64 Versions

If you are running Color 64 version 7.37 or the Super ML upgrade, then you will be able to convert some of the more important items of your system to the new version 8.0 format. With the included conversion utility called "bbs convert", you will be able to convert your √bbs.parms file, password file, UD directories, Network parms, Super ML sequential files, and your √variables file. Also included is another stand-alone utility called "mf convert" that will allow you to convert the "√mod file" of previous Mod Menu versions to version 2.0.

Unfortunately, you will not be able to convert your current main overlays to work with version 8. This applies to both V7.37 and Super ML systems; there have been too many changes for the older overlays to work properly. However, you WILL be able to convert some of the optional merge-in modifications that you may have installed in the past. There is a utility included called "m-con 50000" that will automatically convert these merge-ins to be compatible with the version 8.0 system. Notice that I said "some" of the optional merge-ins; many of the merges created for the original V7.37 system will not be compatible even if the converter utility is used on them.

To begin converting your system, the first thing you should do is read the installation documentation for version 8.0. Make a backup of your current system and then install the Program Files and Boot Files as instructed, but don't run the SETUP program yet because you should be able to convert your original √bbs.parms file. However, the included utility program won't be able to convert your parms file if you have installed any special modifications in the file (e.g. the add-drive mod). In this case you will have to run the version 8.0 SETUP program after scratching your original √bbs.parms file, making sure to print your current parms first so that you can re-enter them in the new SETUP program. If you haven't made any modifications to the parms file format, then you should copy your original √bbs.parms file to the Program Files drive that you have chosen.

Other than your Program and Boot files, all the original files groups on your system can remain exactly where they are. One exception is your help files; you should erase the old standard help files and copy in the new ones provided. You should make sure that your password file is still present if you must run the version 8.0 setup program, otherwise it will create a whole new password file.

9.4 The "BBS CONVERT" Utility

The included program called "bbs convert" is used to convert some of your important V7.37 or Super ML files to version 8.0 format. To use this utility, you must boot the "+shell" program to install ML in memory. Then LOAD and RUN the "bbs convert" program as you would a regular BASIC program and insert your Program disk if required.

There MUST be a √bbs.parms file present for you to be able to use this conversion utility. If you are going to convert your original √bbs.parms file, then make sure that the file is located in the Program Files. Otherwise, make sure you have used the version 8.0 SETUP program to create a √bbs.parms file.

The program will first check to see if your √bbs.parms file is version 8.0 format. If it is not, you will be asked some questions to convert your parms file. Otherwise, since you must have re-entered your parms, you should skip the next few paragraphs and continue reading at "The Conversion Menu".

- The first question you will be asked is if you were running Super ML. Answer "Y" if you did install the Super ML upgrade.
- The next question will ask if you were running Network 64. Answer "Y" to this question if you were running any version of Network.

The program will then read in the current parms and re-write them to disk. If your parms file is not the exact same stock format for V7.37 or Super ML (with or without Network), then the conversion will be faulty.

Once your parms are saved the main menu will be displayed.

- The Conversion Menu
  From this menu you have 5 options, although some of them are not intended for all systems. Once again, you should make sure that you have a backup of any of the affected areas before proceeding with the conversion. Otherwise, you may run the risk of irretrievably losing an important file. Here is a description of each menu option:

- Convert Password File
  Use this option to convert your V7.37 or Super ML password file to version 8.0 format. Each user record will occupy exactly one block of disk space, so you will need to make sure that have enough room on your Password File drive to store both the old file and the new one. You will be informed of this fact when you choose this option, and you will be asked if you wish to proceed with the conversion. The actual conversion process may take a while, depending on how many users you have on your system.

  The password file converter considers several possibilities. If you are one of those SYSOPs who has decided to use the Expiration Date field as other information such as "GUEST" or "SYSOP" flags, then it will be preserved in the transfer. Expiration dates will be converted appropriately also. Finally, if you installed one of the mods that makes use of this field as a phone number, then it will automatically be stored in the new dedicated phone number field of the password record.

- Convert UD Directories
  You must use this option to convert your UD directories to use the new date format of version 8.0. Using this is mandatory if you wish to keep your current UD directories.

- Convert Network v1.24 Parms
  Color 64 version 8.0 includes version 1.26a of Network 64. If you were previously running version Network version 1.24, then you need to convert your Network parms with this option. Those who were using version 1.26 don't need to use this option; your Network parms file is already compatible.

- Convert Super ML Messages
  If you were using the Super ML upgrade, then you will need to use this option to convert your system messages to version 8.0 format. What is done is that the MCI commands and CTRL/O characters are updated to the new format, because version 8.0 does not use CTRL/A as the MCI prefix, and uses CTRL/Y instead of CTRL/O. You will be asked for the device, drive, and init parameters of the files you wish to convert.

  Next you will be asked for the pattern of files to convert. If you are doing your System Files drive, then you should just enter "**.***" as the pattern.

  The program will then automatically read in the directory of the drive using the pattern it was given and will convert all the sequential files matching that pattern. Even if some of the files

are not Super ML messages, it should be safe to convert them, because CTRL/A and CTRL/O are not standard Commodore 64 control characters and thus non-Super ML files should be unaffected. If you are concerned about the integrity of other sequential files, you should copy them to another disk before using this conversion option. Note that if you have Commodore 128-specific sequential files on your system, then the CTRL/O character (which enables flashing characters) will be incorrectly converted to a CTRL/Y, so make sure that any files that are 128-specific are moved to a safe area.

You should use this option on your System Files, and Public Messages, and any other drive which may contain Super ML format messages. Note that the entire conversion process will be aborted if a disk error occurs while the files are being processed. You should make sure that there are no locked files in the directory because they will cause an error if the utility attempts to convert them.

- Convert √variables File
  This option will convert your V7.37 or Super ML variables file to use the new version 8.0 date format. This is mandatory if you wish to preserve your current variables.

- After Using "BBS CONVERT"
  Once you are finished converting all the necessary files, you should now start up the version 8.0 SETUP program and fill in any parameters that are new to version 8.0. If you were not using Mod Menu, then your system is now ready to use. Otherwise, read this next section on converting your √mod file.

9.5 The "MF CONVERT" Program

The included program called "mf convert" can be used to convert earlier version "√mod file" files to the Mod Menu 2.0 format. Just LOAD and RUN "mf convert" just like any other BASIC program and follow the directions on screen.

efore running the program, you should make sure that you have plenty of room on the drive where your "√mod file" is. You may want to read the instructions on Mod Menu 2.0 before doing the conversion, because you will need information on why you need to enter information such as the maximum module number.

9.6 Converting BASIC Code

This section is intended for SYSOPs who want to convert old version 7.37 and Super ML merge-in code to version 8.0. Also, those who were running separate overlay modules through a menu will want to read this section.

Color 64 version 8.0 has completely replaced the traditional SYSC(x) and PEEK(C(x)) methods of interfacing with the ML. These methods were fine for when the system was simpler but would not be able to adequately handle more complex features like SwiftLink compatibility. For this reason, BASIC has been modified to interpret new commands when the ML is active.

In the new system, the ML interface is divided into "commands" and "variables". The ML commands all begin with a "." period character. For example, the equivalent of the SYSC(5) command is .01 (get line of disk input).

The ML variables can be used in expressions just like normal BASIC variables but are defined differently. All ML variables begin with the "!" exclamation point character and are followed by two digits. For example, the equivalent of PEEK(C(22)) is the ML variable !05 (graphics mode). To assign values to these ML variables, a format like the POKE command is used. For example, the equivalent of POKE(C(35)),9 would be !14,9 (set file device number to 9).

Table 41 redefines the old C(X) references to their new substitutions. Note that more than a few references are now obsolete, and no substitution is listed. Also, the old output commands like SYSC(0) have been replaced by special one character commands like the "#" command. For more information on the ML commands and ML variables, consult the programming section.

Table 41- Redefined SYSC(X) Calls to Special Character Command Reference

Old	New		Old	New		Old	New		Old	New		Old	New
C(0)	#		C(11)			C(22)	!05		C(33)	.11		C(44)	.14
C(1)	$		C(12)	!00		C(23)	!06		C(34)	!13		C(45)	.15
C(2)	!01		C(13)			C(24)	.07		C(35)	!14		C(46)	.16
C(3)	!02		C(14)			C(25)	%		C(36)	!15		C(47)	.17
C(4)	.00		C(15)			C(26)	&		C(37)	.12		C(48)	.18
C(5)	.01		C(16)	!07		C(27)	!10		C(38)	!16		C(49)	.19
C(6)	!04		C(17)			C(28)	!11		C(39)	!17		C(50)	!21
C(7)	!03		C(18)			C(29)			C(40)	!18		C(51)	!22
C(8)	.02		C(19)	.04		C(30)			C(41)	!19		C(52)	
C(9)			C(20)	.05		C(31)	!12		C(42)	!20		C(53)	
C(10)			C(21)	!09		C(32)	.10		C(43)	.13		C(54)	

9.6.1 The New Carrier Detect Test

Some other references have been changed also. One such change is the way that the BBS program detects if the carrier is still present. The old method looked like this:

```
ifcd=(peek(56577)and16)thenprint"carrier lost!"
ifcd<>(peek(56577)and16)thenprint"carrier present"
```

The new method replaces these with an ML variable (!25) that is updated with the status of the carrier every 1/60th of a second. The reason this is done because the standard RS-232 routines and the SwiftLink RS-232 routines detect the carrier in much different ways; a simple PEEK will no longer work with both methods. Here is what the new method looks like:

```
if!25=0thenprint"carrier lost!"
if!25<>0thenprint"carrier present"
```

As you can see the new method is much easier to remember (and type!) than the old method.

9.6.2 The New Modem Output Command

Another change involves the way that output is sent straight to the modem, and how characters are read in from the modem. In the old method, to print something straight to the modem, you would use the following format:

```
print#5,"atdt";p$
```

With the new method the command looks like this:

```
'"atdt";p$
```

In effect the new command is shorter and must be used because file number 5 is no longer opened when the BBS is in operation. This is to maintain compatibility with the new SwiftLink routines.

9.6.3 The New Modem Input Function

Finally, the way characters are read in from the modem has changed also. Here is the old format:

```
get#5,a$
```

The new method uses one of the built-in functions of the Color 64 ML. Here is an example of the new method:

```
a$=@4
```

The @4 function reads in a character from the modem and returns it as a string. For more information on the """ apostrophe command and the @4 function, consult the programming documentation.

9.7 The Automatic Module Converter

Included with the Color 64 package is a program called "m-con 50000", which is an ML program that you can use to convert BASIC code from Color 64 version 7.37 and Super ML. This program was intended only to convert the small optional merge-ins for version 7.37 and Super ML, although you can use it to convert game and utility modules if you have a little programming skill. To use this program, you must load it into memory by using LOAD"m-con*",8,1. Then enter a NEW command to make sure that BASIC memory is cleared.

Once this program is in memory, all you need to do is load the BASIC program that you wish to convert, then type SYS50000 to start the conversion process. You don't need to load m-con again if the computer is turned on and you haven't used any programs that alter the memory where m-con resides.

M-con will convert all the SYSC(x), POKEC(x), and PEEK(C(x)) references, as well as the PRINT#5 statements that print output to the modem. M-con also converts the Super ML MCI commands and CTRL/O characters to the version 8.0 format.

- Things that m-con cannot do is:
  1. It cannot convert references to the old carrier detect test; this must be done manually (see above section on the new carrier detect test)
  2. It cannot convert GET#5 statements used to read information straight in from the modem; this must be done manually (see above section on the new modem input function).

- Converting Large Modules
  This conversion program cannot be used to convert an entire overlay, especially any of the main overlays--these must be totally replaced. However, if you have modules and games that are separate overlays, then there are a few requirements to convert them. The subroutines in lines 0 to 27000 are the core subroutines of any Color 64 overlay, and most of them have been revised with the new version, especially the caller log routine. To convert overlays that still have these lines, then you need to strip the lines out (being careful not to take any of the module with it), and then re-merge an appropriate "xxx" skeleton overlay into the module after using m-con to convert the older module. For most games and small modules, the "xxx small" overlay (weighing in at a small 13 blocks) is perfect for installing the necessary core subroutines. You should consult the programming section regarding these overlays for more information on their use.

- The New Add-Drive Mod

SYSOPs who were using previous versions of the add-drive mod should also note that the "H" setting for the three auxiliary file groups is now 12, 13, and 14. This means that to select AUX 1, you would need to use H=12:GOSUB460. If you were using the old add-drive mode, you may need to change some of your overlays that selected AUX 1 with "H=11", because Network now uses that number to select the Network drive. Any easy way to convert to the new system would be to do this: For AUX 2 and AUX 3, all you will need to do is change the drive settings in SETUP so that they are now AUX 1 and AUX 2 (because they are already using H=12 and H=13), then change all references to "H=11" in modules to "H=14", and then configure AUX 3 in setup to what was once your AUX 1 files.

Note:
The m-con program will convert all references to CTRL/A (the Super ML MCI character) inside of quotes to an english pound sign (the version 8.0 MCI character), EXCEPT if the CTRL/A is part of a PRINT#15 statement. This is because programs that use relative files often use the CTRL/A character in the relative file position command sent to file number 15. If you are converting a program that uses relative files, then you may want to check all references to CTRL/A first to see if they will be used as part of a relative file command. If so, then just change these references to use CHR$(1) instead, so they will be protected during the conversion.

Appendix A, UD Directory Maintenance

The "dir tools" program is a utility you can use to quickly create a new √directory file for your download directories. It is true that Color 64 would automatically create a √directory file the first time a caller asked for a directory, but that directory regenerate can take quite a while (especially if you have a a lot of files on the disk). The "dir tools" is a small BASIC program, leaving plenty of free memory so that the program is very fast. The program has two utilities: one to create a directory from scratch, and one to regenerate the directory listing.

To use the "dir tools" program, you can boot up the ML shell (the "+shell" program) and then load and run the "dir tools" program. You can also load and run this program after using any other program that also uses the ML (such as SETUP). The tools program reads in your PARMS file.

Once the program is booted, you can use the Create Directory option for creating your √directory file the first time (it will scratch an existing √directory file if you have one on your disk). This routine will be a lot quicker than doing a full regeneration while the BBS system is running. So if you are wanting to start your system with hundreds of files already in your download areas, I recommend you use this directory create utility on all your disks before you load up your BBS.

You can also use the Regenerate Directory option to scan through the disk directory and update the √directory file with any new entries. Also, the program will eliminate any entries which are no longer valid. It is suggested that you perform this operation routinely, just to clean up the directory. Again, using the regenerate function in this stand-alone program is faster than doing it from the actual BBS system.

Appendix B, Password File Maintenance

The "pswd tools" utility will allow you to copy the data out of the REL password file into an easier to handle SEQ file for backup purposes and restore the password file (all or just one record) from that backup file when needed. With this utility, you can safely store the password file anywhere you choose (like on an SFD 1001 or hard disk drive), if that drive is capable of handling relative files. To use the "pswd tools" program, you can boot up the ML shell (the "+shell" program) and then load and run the "pswd tools" program. You can also load and run this program after using any other program that also uses the ML (such as SETUP). The tools program reads in your PARMS file.

To BACKUP the password file, choose the "Backup Password File" option from the menu in "pswd tools". The program will ask if you are sure, and then will start creating the backup file in the same files section as the password file. You should start seeing a counter increment, indicating the current number of records backed up. When the program is finished, you will find a new sequential file on your disk called "√password backup". This file contains all the information currently stored in your password file, but in an easier to handle SEQ format. Then you NEED to copy this file onto another disk for safe keeping using any file copier.

To RESTORE the password file, you first need to make sure a relative password file already exists. The restore program will not create a password file, it only fills the password file with data from the password file backup. So, if there is not already a password file in the desired file section, run the SETUP program and it will make a new one for you. The password file does not have to be blank, RESTORE will write over any data currently in the file. Also, you need to make sure the "√password backup" sequential file is also in that file section. Use a file copier if necessary to copy the password backup file onto the correct disk. To begin the RESTORE, choose the "Restore Password File" option from the menu in "pswd tools". It will ask for the record number to restore or 0 to restore all records. That is all you need to do. You should now see a counter increment as the backup file is restored into the password file. This restore is much slower than the backup, that is normal when writing into a relative file. Hopefully, we will not be doing too many restores anyway. When the restore is finished, the password file will be ready to use.

There is also an option in "pswd tools" called "Fix Password". This utility can be useful if for some reason you find a caller's record is unreadable. This program will NOT fix disk read errors, it just straightens out records that have corrupted data. Normally you would use "Restore Password File" to restore a bad record, but if you do not have a "√backup password" file, this may be the only other way to salvage what is left of your password file.

Appendix C, The Plusterm Overlay

The Plusterm program was written by Color 64 enthusiast Sam Lewit, who also created Network and many other fine programs. It was written originally as an optional substitute for the built-in term program that came with Color 64 v7.37, but is now included as the standard terminal program with Color 64 v8. The file name is "√bbs.term" and will be automatically loaded from your Program Files when you choose F2 from the Sysop Menu. If the file is not located in your Program Files, then the program will return to the wait-for-call screen.

Some of the features of Plusterm are:

- Uploads and downloads, supporting Xmodem and Punter protocols
- Multi-Punter upload and download transfers
- Phonebook with auto-dial, storing up to 20 numbers
- 2400 BPS Adjustments for "quirky" modems.
- Built in buffer, capable of handling thousands of bytes
- Adjustable Re-Dial Rate for the auto-dialer
- Programmable Function Keys, individually set for each phone book entry

All these features are very simple to use, so I'm not going to bore you with endless documentation on it.

There ARE a few things you need to know though. The 2400 BPS adjustment feature is for non-SwiftLink systems ONLY. It will not affect SwiftLink communications if it is edited. The 2400 BPS adjustment should be used if you know that garbled communications are being caused by computer to modem communications. Most often this may result if the modem is slightly "fast" or "slow". Most of the time, you should not even have to change these numbers, but if you have experience with this type of problem you should be able to adjust the figures with ease.

The buffer is a dynamic feature of Plusterm. The space that is used by the buffer is in memory between the end of the terminal program and the byte that marks the end of √bbs.init. On many systems you can achieve over 10000 bytes of buffer space. Note that the SMALLER the terminal prg and the LARGER the bbs.init prg is, the more buffer space you'll have. This is why it's most important not to add any mods unrelated to the terminal into the term prg. Also, when you're done with the terminal, and want to go back into the BBS, the buffer will be overwritten and lost. If there is ANYTHING in the buffer, you will get a warning prompt before leaving just in case you forgot to save the buffer contents.

If you need to load the buffer with a sequential file for any reason, simply OPEN the buffer, go into the DOS (F4), and do a: f:filename. Then CLOSE the buffer.

You can send the buffer to any of 3 places: The SCREEN, DISK, or MODEM. You do this by selecting (P)rint from the BUFFER menu.

There are 7 function keys available for EACH number stored in your phonebook. These are defined by editing the phonebook. There is also 7 DEFAULT definitions which are loaded in at the time you enter the term. The only problem with this, is that as soon as you use any function from the phonebook, these defaults are gone till you enter the terminal again. To edit the default functions, choose 0 from the phone book editor menu. In order to include a carriage return as part of the command (like <*your password*> + *carriage return*), include a "CTRL-Y" in your string:

1. Type password (do not hit return)
2. Hold the CONTROL key and press "Y"; observe cursor go down one line
3. Hit the RETURN key

The CTRL-Y will automatically be converted to a carriage return when the data is sent to the modem.

Appendix D, The Menu Maker Program

The included program "menu maker" is a self-contained menu-making program that will create a menu for each access level. To use menu maker, first boot the "+shell" program to install the ML. Then load and run the menu maker program. Also, if you have just shut your BBS down, you can load and run "menu maker" without loading the "+shell" program, because the ML will still be installed.

When you run the program, it will load your "√bbs.parms" file to get the level for each command. The data statements from line 40000-40043 contain the description for each command number. Those descriptions that say "**UNUSED**" are not used in creating the menus, because they are associated with parameters that aren't commands.

Change the descriptions to whatever you wish, but only the first 15 characters of each description will be used (for formatting purposes).

Lines 20035-20050 contain the header of the menus. Currently it has some generic text in it, but change it to whatever you wish. The menus are created so that they work in both 40 and 80 columns because each command character and description totals out to 20 characters each. Here is a demonstration of what the descriptions will look like in 40 columns (an F5 character precedes each description, so it cycles through your system colors):

[R] Read Msgs [@] Post Office

[L] Caller Log [M] Membership List

The menu maker uses a Variable MCI for each command character inside the brackets. This means that if you change a command character (but not a level for one) in SETUP, the menus will automatically change to reflect the new command character (and thus you won't have to re-run the menu maker).

MCIs to print the time and date are also included in the files, to save program space.

Appendix E, Drive Initialization Commands

This section is a summary of different drive init commands that seem to work best for a variety of disk drive devices. A drive init command is simply a command that tells the disk drive to prepare for access, and a method by which you can specify which partition (LU and USER for Lt. Kernal users) or subdirectory to use on 1581, RamLink, or Hard Drive systems.

Since some devices may require multiple commands, you can separate individual commands with the exclamation point "!" character. Also, since (as far as I know) the initialize command is universal to all Commodore drives, you can also change the drive number used by Color 64 by following an "i" command with the drive number of your choice. Thus, to select drive number 13 (perhaps on a CMD HD), you would include an "i13" command somewhere in the init command. This will change the variable DR$ (drive number) to reflect the current drive number in use. Note that if you DO use the "i" command in a drive init string, this WILL override the drive number (0 or 1) you used in SETUP. Note that the "i" command does NOT change the drive number for init commands entered in the bootmaker programs (bootmaker, bm ram, and bm small).

Here are the different init commands:

- 1541 or compatible disk drive: a simple "i0" command should suffice.
- SFD 1001, CBM 2031, CBM 9060, CBM 9090, or other single drive devices: use the "i0" command.
- MSD-SD2, CBM 4040, CBM 8050, CBM 8250, or other dual drive devices: you can use "i0" to select drive 0, or "i1" to select drive 1.

E.1 1571 or Compatible Disk Drives

This disk drive can behave in 3 different ways: 1541 single drive mode, 1571 double capacity mode, or 1571 dual side mode. If you want to use the 1571 in 1541 mode, single sided, 664 blocks per disk, then you can just use "i0" just like a 1541. But since the 1571 is a double sided drive, it is capable of storing 1328 blocks in one directory in 1571 mode, or 664 blocks in separate directories on each side of the disk. I recommend the latter arrangement, since you will have twice the space in the directory (144 files on each side) and it has been found that 1571s run more reliably in this mode. Anyway, if you want to have one directory of 144 files/1328 blocks free, use "u0>m1!i0". If you want to have two directories of 144 files/664 blocks each, use "u0>h0!i0" for the normal side, or "u0>h1!i0" for the second side. Also, when using a 1571 in one of these to modes, make sure you have a diskette properly formatted. If you don't already have a diskette formatted, you can use the DOS wedge in the JiffyDos, MSG EDITOR, or the BBS and enter in the same drive command followed by the command to format the disk. Example:

1571 mode - u0>m1 then n0:diskname,id
1541 mode side 0 - u0>h0 then n0:diskname,id

1541 mode side 1 - u0>h1 then n0:diskname,id

Note that a 1571 emulation mode partition on a CMD HD does not accept any of these commands, because it is simply a simulation of the double sided 144 files/1328 blocks mode. Use a 1541 emulation mode partition if you need to use something as a single side 1571 disk, because a 1541 mode partition supports 1571 burst commands.

E.2 1581 Disk Drives

This disk drive has a few special features of its own. This 3.5" disk drive already uses both sides of the diskette, so you do not need to concern yourself with this element. However, the 1581 does support partitions, but it is not necessary to use this feature. If you do not wish to use partitions, a simple "i0" command will suffice.

If you do wish to use partitions on the 1581, you must follow instructions in your disk drive manual to create the partition, or you can use a 1581 utility program to create the partition for you. The command you should use to select the partition is "i0!/!/0:PARTITION NAME". The "i0" is to initialize the drive, the "/" is to make sure the root directory is selected, and the "/0:PARTITION NAME" is to select the partition. You can omit the "/" that is all by itself if you wish, because the "i0" command should automatically select the root directory. If you notice errors, though, you should put the extra command back in to ensure proper partition use. If you need to select a partition within another partition, simply tack on another "!/0:PARTITION NAME" to the end of the init command. And finally, if you need to select the root directory, a "i0!/" command or a simple "i0" command should do the trick.

The stock 1581 drive produced by Commodore has a few bugs in it that make it very unreliable for use on a BBS system. If you must store anything on a 1581 drive, be sure to limit it to public messages or downloads or something that is not vital to BBS operations. You should avoid running your overlays on a 1581 drive. However, if you are using JiffyDos on your 1581, the original Commodore bugs should be fixed, and it should be fine to use a JiffyDos drive. Also, a 1581 emulation mode partition on a CMD HD behaves exactly like a 1581, but without the bugs. See the CMD HD section on how to properly use a 1581 emulation mode partition.

E.3 CMD Hard Drive

The first thing you need to do is select the proper partition. This is accomplished by using the "cp" and "i" commands. For example, to select partition 2 you would use the command "cp2!i2". This applies to ANY partition, regardless of its type. For 1541 or 1571 emulation mode partitions on the CMD HD, no additional commands are necessary. For 1581 emulation mode partitions, you may need to add a "/0:PARTITION NAME" command to select a 1581 style partition.

For native mode partitions on the CMD HD, you can also use subdirectories. This is accomplished through the "cd" command. For example, to select the subdirectory "games" in native mode partition 5, you would use "cp5!i5!cd//games". Note you need to include the "i5" command to ensure that the BBS system knows to use drive number 5.

E.4 CMD RamLink

This uses the same commands as the CMD HD.

E.5 CMD FD series drives

As of this writing, I have not had a chance to use one of these advanced 3.5" floppy drives. However, since the FD drives are supposed to be compatible with standard 1581 disks, it should accept the same commands for initialization and partition selection. See your drive manual for information on other commands.

E.6 Lt. Kernal Hard Drive

The Lt. Kernal HD divides its storage up into LU's (logical units) and USERs. A logical unit can be compared to a separate partition of storage space, independent of the other LU's. A USER can be compared to asubdirectory, sharing the common space of the LU with all the other USERs. LU's can have a number from 0 to 9, while USERs are numbered from 0 to 15.

On the Lt. Kernal HD, the "l" command is used to select the LU and USER, and is in the form l<device><LU><USER>, where device is the device number of the Lt. Kernal HD (usually 8), LU is the logical unit number (0 to 9), and USER is a hexadecimal number from 0 to F (the letters A through F correspond to the numbers 10 through 15, respectively).

You also need to include an "i" command to set the drive number (LU number) on the Lt. Kernal. For example, to select LU 2, USER 11, you would use the command "l82b!i2". To select LU 0, user 5, you would use "l805!i0". Just remember that for the USER number, the letters A, B, C, D, E, and F are used to mean 10, 11, 12, 13, 14, and 15, respectively.

E.7 ICT Hard Drive

The ICT HD has two parts, the built-in floppy disk drive, and a hard drive. You should not use the "i" command in the init commands for an ICT HD, because this could confuse the HD. Rather, set the drive number to 0 in allcases in SETUP. Here is a summary of commands: "h0" selects the built-in floppy drive. "h" followed by a number greater than 0 will select the appropriate HD partition. For

example, "h2" will select partition 2. The "hm4" command is the partition chain command. For example, to chain partitions 5 to 7, you would use "hm4 5 7". If you are going to use partition chains, you must have the special ICT merges installed in your BBS. See the section on the using the ICT HD with Color 64 for more information.

E.8 Ram Expansion Unit

Although the 17XX series REU is not a true disk device, it nevertheless simulates the operation of a disk device. A simple "i0" command should be fine.