
Color 64 BBS Manual Version 8.1A March 2026

Color 64 BBS version 8.1A Manual

Forward by Michael "Nuke" Newkirk

Sysop of Itchy Butt BBS in King George VA (Color 64 v8.1a)

Itchy Butt BBS [itchybutt.org](https://www.itchybutt.org):6502

<https://www.itchybutt.org>

Color 64 Version 8.1a modifications written by Michael Newkirk

For the most up-to-date inclusions of this manual, go to <https://color64wiki.itchybutt.org>

Content was derived from original documentation for Version 8.0.

Credits:

Color 64 Version 8.0 and 8.1 written by Anthony Tolle and Fred Ogle

Color 64 v7.37 was written by Greg Pfountz

The Network 64 v1.26/1.26a modification for Color 64 was written by Adam Fanello

Color 64 BBS Manual Version 8.1A March 2026

Revision Date	Notes
October 2025	<ul style="list-style-type: none">• Updated to 8.1a technical content• Edited flow of content• Minor corrections
March 2026	<ul style="list-style-type: none">• Added research findings to ML Variables, Commands and Functions• Reformatted

Contents

Foreward	8
Software & Installation	11
Overview	13
Hardware Support.....	15
The Program Overlays	19
Disk Speed Enhancers	19
File Storage Drives.....	20
Different System Configurations	23
8.10a Updates from 8.1	25
Technical Support.....	26
Specific System Requirements	26
LT Kernal HD	27
ICT HD.....	29
CMD HD.....	31
CMD RAMLink	32
Fastload Cartridge	33
Skyles Flash!	33
Avatex 1200.....	34
2400 Baud Modems	35
High Speed Modems	36
Ram Expansion Unit (REU)	36
Preconfigured Builds	41
Chapter 2, Installation.....	44
Copying the BBS Files	44
Creating the Boot Programs.....	46
Boot File ("+") Program Details	49
Boot File Descriptions	50
The ML Shell.....	51
SETUP Parameters.....	51
The "vbbs.parms" File	51
SETUP - Main Parameters	52
SETUP - Disk Drive Assignments.....	56

Color 64 BBS Manual Version 8.1A March 2026

SETUP - Upload/Download Directories	57
SETUP - User's Time Limits	58
SETUP - Caller Purge	59
SETUP - Message Categories	59
SETUP - BBS Commands	59
SETUP - Color Code Setup	60
SETUP - Carrier Status	60
SETUP - Saving the Parameters	60
SETUP - Editing/Changing the Parameters	61
SETUP - Printing the Parameters	61
Prior to Running the BBS	62
The System Messages	63
System Files for BBS Operation and Customization	63
Special Provisions for ASCII callers	69
The Help Files and Text Files	70
Email Notification Option	72
BBS Operation	73
Loading The BBS	73
The SYSOP Menu	76
Password Record Information	78
Sysop In/Out Flag	79
Signing On	80
Graphics Mode	83
The User Application	85
Local Console Commands	87
The Caller Log System	88
MCI Commands	89
The Message Output MCI	90
The Variable MCI Command.	91
Text Editing Features	93
Customizable Message Headers	95
User Settings	97
Using Mod Menu 2.0	99

Color 64 BBS Manual Version 8.1A March 2026

The Mod Stat Editor	100
Customization	104
Fast Garbage Collect Routine	107
Routine Maintenance	107
The Crash Routine	108
Network 64 Instructions	109
Incoming Node Accounts	120
Programming with Color 64	125
The New Load Command	126
New Output Commands	127
Enhanced If/Then Statement	129
The ML Command Set	131
ML Variables	135
The ML Functions	139
Generic Routines	143
Color 64 BASIC Variables	145
The Generic Overlay Files	150
The Overlay Loader Lines	153
Individual Overlay Branching	153
The “OV” Settings for Overlays	154
Branching Tables	156
Merging in Modules	161
Adding Games	163
Using Disk Drives	166
Drive Initialization Commands	170
Miscellaneous Programming Notes	175
Undocumented Commands Research Project	192
Tools	203
Directory Tool	203
Password File Tool	203
PlusTerm Program	204
Automated Menu Maker	206
Appendix A Military Time Conversion Chart	208

Tables:

Table 1 - Vice Settings Example	17
Table 2 - Color 64 v8.1a Overlays	19
Table 3 - Network-Related Overlays	19
Table 4 – Drive Group with Drive Assignment Examples	20
Table 5 – Storage Space of Files (at install)	21
Table 6- vsys.remove Copy Routines.....	37
Table 7 - Script Merge Command Listing	40
Table 8 - List of Script Merge Functions	40
Table 9 - Color 64 Files to Copy	44
Table 10 - Machine Language File Selection Based on Configuration.....	45
Table 11 - Additional Boot Files to Copy for REU Use	46
Table 12 - Minimum Boot Files Required (Boot Files / Program Files combined on disk)	46
Table 13 – SD2IEC Directory Layout Example.....	47
Table 14 - Boot Files Functional Description	50
Table 15 – Main Parameters Descriptions.....	52
Table 16 – Disk Drive Assignment Descriptions	57
Table 17 - Directory Information Query Fields.....	58
Table 18 - Summary of System Commands.....	59
Table 19 – System Message Files	64
Table 20 - Disk Commands in DOS Wedge	77
Table 21 - Password Record Field Descriptions.....	78
Table 22 – ASCII Conversion Example	83
Table 23 – ANSI Conversion Example	84
Table 24 - Standard MCI Commands.....	89
Table 25 - Message MCI Commands	90
Table 26 – Basic Token Reference Chart.....	91
Table 27 – Text Editor Miscellaneous Features	93
Table 28 – Custom and Default Message Headers.....	96
Table 29 - Module Menu Main Parameter Definitions	100
Table 30 - Stat Descriptions for Mod Menu	100
Table 31 - Mod Stat Editor Command Definitions	101
Table 32 – Mod Stat Editor Sub Menu Command Descriptions.....	102
Table 33 - Mod Menu Files and their function.....	102
Table 34 – Summary of Required Files for Network 64.....	110
Table 35 – Summary of Required Files for Network 64.....	113

Color 64 BBS Manual Version 8.1A March 2026

Table 36 – Network Menu Command Descriptions	116
Table 37 – Network Maintenance Functions & Descriptions	118
Table 38 - Network Troubleshooting FAQ	124
Table 39 – Text Output Commands	127
Table 40 – Enhanced IF/THEN Statements	129
Table 41 - ML Command Set	131
Table 42 - ML Variable Summary	135
Table 43 - ML Function Descriptions	139
Table 44 - Callable Routines by Line Number for Overlays	143
Table 45 - File Group Assignments	144
Table 46 - List of Color 64 Defined Variables.....	145
Table 47 - OV Settings Cross Reference.....	152
Table 48 - Standard Main Overlay Loader Lines.....	153
Table 49 – Path for settings of OV - VBBS.INIT Overlay	154
Table 50 - Spare Command Pre-Assignments	156
Table 51 - Menu Branch Destinations	159
Table 52- Midnight Maintenance Routine Paths.....	159
Table 53 - Spare Command Pre-Assignments	161
Table 54 – SD2IEC Folder Setup Example	168
Table 55 – LT Kernal Setup Examples	173
Table 56 - Spare Command Reference Chart	178
Table 57 – DA Derived Variables	179
Table 58 – Date Text Derived Variables	179
Table 59 – ADN Derived Variables.....	180
Table 60- Redefined SYSC(X) Calls to Special Character Command Reference	182
Table 61 – BASIC Syntax Changes in 8.1 - Research	192
Table 62 – ML Command Changes in 8.1 - Research	194
Table 63 – ML Variable Changes in 8.1 - Research	198
Table 64 – ML Function Changes in 8.1 - Research	201
Table 65 - Military Time Cross-Reference	208

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 expand unimaginably.

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 was a real byte in the rear. 1200 Baud was only 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

Color 64 BBS Manual Version 8.1A March 2026

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 plane, the exposure of these systems to the younger generations is 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 here we are 20 years into the future and it never materialized. 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 were not captured in the original 2005 manual content. I've incorporated all information I have gathered while working with the software.

*Peace & Love. ---Nuke---
Michael Newkirk – 2025*

Color 64 BBS Manual Version 8.1A March 2026



Version 8.1a

Technical Support: [Color 64 Discourse](#)

[Online Sysop Manual](#)

Software & Installation

[Installation Instructions](#)

Install disks:

Latest Build Date: 11 FEB 26

Available Downloads

File	Description	Link	File Size	Updated
Color 64 v8.1a D64 Install	D64 images of Install disks (5 in total). Includes a sixth D64 file containing Kaleidoscope V4. These files should be copied (not run) from the install disks.	Download	449K	11FEB26
Color 64 v8.1a D81 Install	D81 image of all files. Does not include Kaleidoscope V4. These files should be copied (not run) from the install disk.	Download	332K	11FEB26
Color 64 v8.1a Traditional Build	Contains media formats D64 and D81 of files, documentation, and useful tools. These files should be copied (not run) from the install disks.	Download	87MB	11FEB26
Color 64 v8.1a Prebuilt- Windows LTK (Vice)	This is a prebuilt version of Color 64 v8.1a using a modified version of Vice Emulator to support high-baud rates. Some configuration steps are still required. Refer to the Preconfigured Builds section for more information.	Download	45MB	11FEB26
Color 64 v8.1a Prebuilt- Linux LTK (Vice)	This is a prebuilt version of Color 64 v8.1a using a modified version of Vice Emulator to support high-baud rates. Some configuration steps are still required. Includes scripts for installation of Vice Emulator and BBS Operation. Refer to the Preconfigured Builds section for more information.	Download	10MB	11FEB26
Color 64 v8.1a Prebuilt- SD2IEC Build for C64 hardware	This is a prebuilt version of Color 64 v8.1a for C64/128 hardware configured with a SD2IEC drive. It was pretested using a WiFi modem. Real modem operation untested but assumed to be working. This is an image of a 4GB SD card that can be flashed to your SD card with tools like	Download	273MB	11FEB26

Color 64 BBS Manual Version 8.1A March 2026

	Etcher. Refer to the Preconfigured Builds section for more information.			
Color 64 v8.1a LTK Images only	These are ready to run images to insert in a Vice Emulator running with the LTK Cartridge image. Can be used with Windows or Linux versions. Some configuration steps are still required. Refer to the Preconfigured Builds section for more information.	Download	3MB	11FEB26

Games for 8.1 / 8.1a

Games below were set up for use with the 8.1a Games Module. Slight modifications in the individual games would be required to use with Mod Menu, specifically drive location settings. Refer to [Game Ports for Color 64 on Discourse](#). Contact me on Discourse if you have any problems with this!

- [Big Trouble, Little China](#)
- [Star Trek](#)
- [Wrestle](#)
- [Dragonslayer](#)
- [Asylum](#)
- [Nuke'em V6](#)
- [Empire](#)
- [Trade Wars 2112 \(Beta\)](#)
- [8.1a Games Module \(Compatible with 8.1\)](#)

Overview

Color 64 BBS is one of the most established systems in the Commodore 64 bulletin board world, with decades of refinement behind it. It combines structural depth with practical simplicity, offering powerful system capabilities while remaining accessible and customizable for individual Sysops.

This wiki supports Color 64 versions 8.0 and later. For earlier releases such as 7.37, Oasis BBS (oasisbbs.com) is recommended as a reference resource.

Color 64 is comprised of two primary components:

- The BASIC overlays — These programs manage the operational structure of the BBS, including user interaction, messaging, file handling, and system control.
- The machine language core — This layer extends BASIC functionality, enabling modem control and performance optimizations that allow the system to operate efficiently within the constraints of Commodore hardware.

The overlays were intentionally written in BASIC to encourage modification and customization by Sysops. This design philosophy made continued development possible, including version 8.1.0a. With even modest BASIC knowledge, a Sysop can tailor the system to create a distinctive board. However, programming experience is not required. The stock configuration provides sufficient functionality to operate a full-featured BBS immediately.

Before beginning installation, review all documentation carefully. Planning your system in advance will save considerable time later. Important considerations include:

- System theme and purpose
- Storage allocation for message and file categories
- Hardware configuration and expansion devices

Additional guidance is available in the Installation section.

Features of Color 64 BBS 8.1A include:

- Supports ANSI, ASCII, and PETSCII (CBM) graphics
- Public and Private Messaging
 - Automatic reply capability for public and private messages
 - MCI support (based on user permission level)
 - User signatures and banners (introduced in v8.1a)
- Text file and on-line help repository
- Nine configurable permission levels
 - Allows granular control of:
 - Command access
 - Message base access
 - File base access
 - Time limits per level
- Y2K compliant
- Automatic recovery from system or program errors
- Multiple system operating modes for Sysop use
 - BBS Wait for Call (normal operation)

Color 64 BBS Manual Version 8.1A March 2026

- Local Mode (Sysop logs in at console)
 - Terminal Mode (built-in terminal with upload and download support)
 - DOS Mode (integrated DOS wedge for maintenance)
- Modem support
 - Native 300–2400 baud
 - Up to 38,400 baud with SwiftLink (hardware or VICE emulation)
- File Transfer System
 - Xmodem and Punter protocols (including multi-receive mode)
 - Permission-based upload and download visibility
 - Multiple configurable categories across single or multiple disk systems
 - Chronological download directory with automatic date tracking
- Automated Maintenance Features
 - Automatic deletion of old unread mail
 - Automatic purging of inactive members after a configurable period
- Caller Log history
- Real-time Sysop/User chat
- Supports RAM Expansion Units (REU) up to 2MB

Hardware Support

Color 64 operates reliably on original Commodore hardware using a wide range of disk configurations, including SD2IEC devices. It also functions properly in emulated environments such as VICE, as well as on modern hardware implementations like the Commodore Ultimate Elite II with IEC support.

1581 disk drives and D81 images are fully usable for most areas of the system. However, they are not recommended for components that require the creation of REL files, such as the system PASSWORD file or certain games that generate REL data. If necessary, this limitation can be addressed by creating the REL file on a 1541-compatible drive and then transferring it to the 1581 or D81 image using a reliable REL file copy utility such as COPYALL.

Special configurations this system will support include:

- Commodore Ram Expansion Units (REU):
 - Requires a minimum of 256K capacity.
 - Utilizes the REU as a high-speed virtual disk through the included RAMDOS utility and automated file transfer routines.
 - The bundled RAMDOS version supports REUs expanded up to 2MB, providing near-instant load and save performance.
- CMD RamLink:
 - Compatible with the CMD RamLink device from Creative Micro Designs.
 - May be used with a CMD HD to take advantage of the parallel interface.
 - If equipped with a RamCard and installed RAM, the automatic RAM-disk functionality may also be utilized.
- Skyles 1541 Flash!:
 - Supported on device number 8 only.
 - Significantly improves 1541 performance.
 - Once installed, the system automatically detects device 8 and adjusts operation accordingly.
- Traditional Modems:
 - Supports fully Hayes-compatible modems using the standard AT command set.
 - Some models may require configuration adjustments.
 - Example requiring modification: Avatex 1200 Low-Cost modem (not the HC or 2400 variants).
- Wifi Modems:

Color 64 BBS Manual Version 8.1A March 2026

- Compatible with Hayes AT-based WiFi modems such as WiModem.
- Example initialization string: `ate0x1s0=0s10=30v1`

- **Emulated Modem:**

- Requires TCPSER or BBS Server.
- SwiftLink emulation supports speeds up to 38,400 baud.
- Example TCPSER initialization string: `ate0v1h0x1m0b1`

The best modem to use is one that can be programmed not to auto-answer (ATS0=0) and will accept the command ATA to answer the phone when the phone rings. For 1200 baud operation a Hayes Smartmodem 1200, Volkmodem 12, the 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 1670, switch 3 up and all the 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:**

- Required for communication speeds above 2400 BPS.
- The SwiftLink RS-232 interface from Creative Micro Designs connects via the cartridge port.
- Supports communication rates up to 38,400 BPS with improved reliability.
- Enables use of modem compression protocols such as MNP (Microcom Networking Protocol).
- Compatible with Hayes-compatible high-speed modems.

- **Schnedler TurboMaster CPU:**

- Supports operation at 4.09 MHz, increasing overall system speed significantly.
- Connects through the cartridge port.
- May be combined with SwiftLink for maximum performance.
- Requires the Master Adaptor configured to GEORAM mode when used with SwiftLink.
- SwiftLink I/O address should remain at the factory default of \$DE00 unless hardware compatibility has been verified.
- Although no longer produced, used units may still be available.

- **SD2IEC Drives:**

- Supports both native directory mode and D64/D81 disk images.
- Native mode offers substantial storage capacity.
- REL file creation in native mode may present issues.

Color 64 BBS Manual Version 8.1A March 2026

- Recommended workaround: create REL files on a 1541 disk or within VICE on a D64 image, then transfer using a reliable REL file copier.
 - Directory navigation examples:
 - !CD// — change to root directory
 - !CD//AUX3 — change to AUX3 directory
 - In DOS Wedge mode, use:
 - CD//
 - CD//AUX3
- Ultimate II Elite:
 - The Ultimate II Elite has wifi capability with the ability to answer calls. While it has not yet been formally tested with 8.10a, it should work.
 - The IEC function closely resembles that of SD2IEC commands mentioned above, except you don't use a double-slash ("//") in your directory selection. For example: selecting "Uploads" directory would be: Drive 8, Device Init: 0:!cd/uploads.
 - Vice:
 - Fully compatible when configured with TCPSER or BBS Server.
 - Supports SwiftLink emulation up to 38,400 baud.
 - Storage capacity depends on configuration, including LT Kernal (LTK) or CMD drive environments.
 - LTK environments provide near-instant module loading similar to REU performance.
 - REU emulation is also supported within VICE.

Table 1 - Vice Settings Example

File	Settings
config.ini	[Version] ConfigVersion=3.9 [C64SC] LogToFile=0 SaveResourcesOnExit=1 SoundBufferSize=100 KeymapIndex=1 Window0Width=703 Window0Height=594 Window0Xpos=377 Window0Ypos=424 MachineVideoStandard=2 MachinePowerFrequency=60 VICIIIGLFilter=1 VICIIModel=4 RsDevice3ip232=1 RsDevice3="127.0.0.1:25239" RsDevice3Baud=38400 AutostartPrgDiskImage="" Acia1Dev=2

Color 64 BBS Manual Version 8.1A March 2026

File	Settings
	Acia1Enable=1 Acia1Mode=2 LTKimage0="/home/Color64_81a_servers/production/DiskImages/kernal-disk0.dlk" LTKimage1="/home/Color64_81a_servers/production/DiskImages/kernal-disk1.dlk" LTKimage2="/home/Color64_81a_servers/production/DiskImages/kernal-disk2.dlk" DriveSoundEmulation=1 DriveSoundEmulationVolume=2007 Drive8FixedSize="0" Drive9FixedSize="0" Drive10FixedSize="0" Drive11FixedSize="0" DosName4000="901468-14-16.bin"
Vice execution Command	x64sc -config /mydir/Config/config.ini -cartcrt /mydir/Bins/ltk.crt

(in the above table – change “mydir” to your installation directory)

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 of an alphanumeric. For example:

- \bbs.ovl 8000 = Main Overlay, Version 8.0
- \bbs.ovl 8001 = Main Overlay, Version 8.0 revision 1
- \bbs.ovl 8100 = Main Overlay, Version 8.1
- \bbs.ovl 810a = Main Overlay, Version 8.1.0a

For the purposes of this manual, suffixes will not be referred to when discussing specific overlays, but note that version 8.1.0a does have some unchanged files from previous iterations of 8.0/8.1, so not all files will be shown with suffix "810a".

Table 2 - Color 64 v8.1a Overlays

Overlay	General Description
\bbs.init	Responsible for initial variable setup, caller logon, and password maintenance.
\bbs.msgs	Responsible for all public and private messages, composing and reading.
\bbs.xfer	Responsible for single and multi-file uploads/downloads, and directory maintenance.
\bbs.ovl	Responsible for multi-uploads, help and text files.
\bbs.ov2	This handles loading your online games and modules. Use of the module loader in "\bbs.ov2" is optional to the Sysop. Version 8.1.0a now includes an alternate games module, \bbs.games.
\bbs.ov3	This is used for a system guestbook / "wall" and should be included as part of your system.
\bbs.term	A full-featured terminal program. Optional for install.
\bbs.games	A new game module as an alternative to \bbs.ov2. This module is set to use the AUX3 designated drive and the \bbs.games file itself should be resident there. It is the only overlay placed outside of the designated "programs" drive.
\bbs.profile	User Profile customization area, permits users to create a customized signature for messages as well as a greeting banner to display to other users when they are messaging that person.

If you are going to use Network, then there are two other overlays that Color 64 will use.

Table 3 - Network-Related Overlays

Overlay	General Description
\bbs.nw1	This contains the Network message editor, and handles calling and receiving calls.
\bbs.nw2	This handles the Network maintenance and the distribution of messages. If you do not wish to run Network, then these files are not required.
	Note: File prscrn52750 is also required as part of this.

Disk Speed Enhancers

The program overlays average approximately 90 blocks each. On an unmodified disk drive, load times can be excessive, making the use of a disk speed enhancement system strongly recommended. JiffyDOS is highly recommended, though many other fastloader systems compatible with Color 64 may also be used. The Xetec Lt. Kernal hard drive with its Host Adapter is another supported option; when using Lt. Kernal, ensure the NMI trap is set to OFF (00) in the Lt. Kernal configuration. As a general guideline, if load times exceed 15 seconds, callers are likely to notice and may become dissatisfied.

Color 64 BBS Manual Version 8.1A March 2026

An effective alternative to slow disk access is the addition of a Commodore 1764, 1750, or compatible RAM Expansion Unit (REU) expanded to at least 256K. These devices provide additional RAM, eliminating most overlay load delays during BBS operation. Color 64 supports up to 2MB of external RAM.

Dedicated boot programs are included to install RAMDOS and execute a BASIC script that copies required system files into REU memory. Even if you intend to operate primarily from an REU, it is advisable to first install and run the system using a standard disk device. Once you are familiar with system operation, transitioning to REU-based operation is straightforward.

You may notice that each program overlay filename begins with the “V” character. On a Commodore 64 or 128, this character is entered by holding either SHIFT key and pressing the @ key. The prefix is used to conceal system files from normal directory listings, preventing callers from viewing or downloading them.

File Storage Drives

Once your hardware is assembled, the next step is determining how Color 64 will organize and store its files.

The BBS divides external storage into logical “drives.” In this documentation, a drive refers to any isolated storage area, regardless of hardware type. A drive may be:

- A 1541 or 1571 floppy disk drive
- A 1581 disk drive
- A CMD HD partition
- An REU configured as a RAM disk
- Any other dedicated storage area

Each drive contains one or more file groups. A file group is a collection of related files serving a common function. Every group must reside together on a designated drive area.

For example, the Public Messages group contains all public posts made by users and should reside on a single drive. If sufficient space is available, multiple groups may share the same drive.

Table 4 – Drive Group with Drive Assignment Examples

Drive	Group Assignment (Example only!)
8 (1541)	Boot, Program
9 (1581)	System, Help, Text, Aux1, Aux3
10 (1581)	Private & Public Messages
11 (1571)	Password, Caller Log

All file groups must remain accessible while the BBS is running. Disk swapping is not supported once the system is active.

When referring to a “Program Files drive,” this means the drive on which the Program group resides. The same terminology applies to all file groups. Multiple groups may share a drive if space allows.

Color 64 BBS Manual Version 8.1A March 2026

Below is a description of the file groups used by Color 64. Recommended minimum storage values are provided as general guidance.

Table 5 – Storage Space of Files (at install)

Group	Description	Files Associated	Size (Blocks)
Boot	<ul style="list-style-type: none"> Contains startup files and utilities such as SETUP, password tools, and system loaders. Must reside on drive 0 or drive 1 (LU 0 or LU 1 on Lt. Kernal). Must not be placed in a subdirectory. 	Bootmaker vsys.loadml vsys.mlinit vsys.mlnorm vsys.setup +bbs addt'l "+" files dir.rename ¹ dir tools ¹ pswd tools ¹ menu maker ¹	Approx 200 blocks Additional 70 blocks
Program	<ul style="list-style-type: none"> Contains main overlays, vbbs.parms, and required ML files. Must reside on drive 0 or drive 1. Must not be stored in subdirectories. <p>¹vbbs.ov2 is not currently used and could be omitted until you desire some sort of expansion in functionality.</p> <p>²vsys.edit would only be required if you desire to use the stand-alone editor (+editor)</p> <p>³vbbs.term and vbbs.trmml would only be required if you desire to use the built-in terminal for outbound calls or easy modem communications.</p>	vbbs.init vbbs.msgs vbbs.xfer vbbs.ovl vbbs.ov3 vbbs.profile vbbs.ansi vbbs.ascii vbbs.punt vbbs.xmoc vbbs.parms ----- Optional files: vbbs.ov2 ¹ vsys.edit ² vbbs.term ³ vbbs.trmml ³	Approx 450 blocks Additional 190 blocks
System	<ul style="list-style-type: none"> Contains menus, data files, and relative files used by the BBS. The drive storing System files must support REL files. 	Menu files Data files REL files	Minimum 300 blocks recommended
Help	Contains all the files which a caller can read to get help using the BBS system.	Help files (Sysop defined)	
Text	<ul style="list-style-type: none"> Additional readable files such as graphics or informational documents. May be combined with Help if desired. 	Text files (Sysop defined)	
Password	<ul style="list-style-type: none"> Stores user account records. Approximately 1 block allocated per user. The drive must support REL files. 	vpassword file	Dependent on maximum # of users: ~1 block per user
Private Msgs	Stores private email between callers.		Minimum 300 blocks recommended
Public Msgs	<ul style="list-style-type: none"> Contains the public messages that are posted by callers. Public messages networked to your system are also stored here. 	Private message files	Minimum 500 blocks recommended
Caller Log	<ul style="list-style-type: none"> Stores caller activity logs. Must reside on drive 0 (or LU 0 on Lt. Kernal). 	vcaller log	Sysop Determined

Color 64 BBS Manual Version 8.1A March 2026

Group	Description	Files Associated	Size (Blocks)
Aux1	<ul style="list-style-type: none"> User Profile area (v8.1.0a). Stores signatures and welcome banners. May be disabled in SETUP (Spare 2). 	User signature files User banner files vbbs.profile	Varies by # of users and use of the function.
Aux3	<ul style="list-style-type: none"> Games area (v8.1.0a). May be disabled in SETUP (Spare 1). 	vbbs.games vbbs.menu ... games	25 blocks + games
Network	<ul style="list-style-type: none"> Not needed unless you plan to participate in a Color 64 network. Includes network overlays and generated data files. 	Programs Drive: vsys.net vbbs.nw1 vbbs.nw2 Generated files	Minimum 1000 blocks

Different System Configurations

The most basic Color 64 configuration can operate with two 1541 disk drives and three disks: a Program disk, a Boot disk, and a System disk (containing all remaining file groups). In this setup, the first drive contains the Program Files and is used to boot the system. The second drive contains the System disk and stores the remaining file groups, such as System Files, Public Messages, and related data.

As additional storage becomes available, consider the following guidelines:

- Program Files should remain on a dedicated drive. The Password file may be stored with them if space permits. Traditionally, Program Files are placed on device 8, drive 0.
- On a new system, System Files typically require minimal space and may be combined with Help Files, Text Files, or another smaller group.
- The Caller Log may be stored with the System Files if you do not require separate daily logs. However, the Caller Log must reside on drive 0 of a device.

Beyond these general recommendations, storage allocation is flexible and ultimately determined by your available hardware and preferences. If you are uncertain about initial placement, do not be concerned. The SETUP program allows you to change file group locations at any time, making it easy to redistribute files if a drive becomes too full. You can rebalance storage simply by updating the group assignments and moving the files accordingly.

Before beginning installation, it is recommended that you sketch out your intended drive layout. A simple plan on paper will make the installation process more straightforward.

Throughout the documentation, references may be made to swapping disks during certain boot procedures. If your system does not require disk swapping—such as when using a hard drive—ensure that all required Program and Boot files are accessible when starting Color 64. This can be accomplished by placing them in the same storage area or in separate partitions on the same hard drive.

Color 64 BBS Manual Version 8.1A March 2026

SYSOP Notes – Drive Assignments				
Group	Drive	Dir (if appl)	Init command	Notes
Password File				
System Files				
Help Files				
Public Messages				
Private Messages				
Text Files				
Caller Log				
Program Files				
Network Files				
Aux 1 (User Profile)				
Aux 2				
Aux 3 (Games)				
UD area 1				
UD area 2				
UD area 3				
UD area 4				
UD area 5				
UD area 6				
UD area 7				
UD area 8				
UD area 9				
UD area 10				

8.10a Updates from 8.1

Version 8.1a represents the result of extended hands-on work with the 8.0 and 8.1 codebase. While the internal structure of 8.1 is complex and not fully documented, two years of active testing, modification, and live system operation have produced a number of refinements, feature enhancements, and bug corrections.

All development and testing were performed on a Commodore 128 system, both in stock configuration and with an implemented REU (RAM Expansion Unit). Modifications were limited to the BASIC overlays; the machine language (ML) components remain unchanged.

The ML portion of 8.1 remains largely undocumented. Attempts to fully reverse engineer or expand the ML layer have proven difficult due to limited reference material and lack of official documentation. Some insight can be inferred from earlier 7.37 releases, and ongoing study continues to improve understanding of its structure and capabilities.

One of the more challenging aspects of 8.1 development is the presence of undocumented ML commands, ML variables, and BASIC shortcuts embedded within the overlays. These additions complicate modification efforts because their functionality is not clearly defined. Continued analysis and testing are expected to clarify these behaviors over time, potentially allowing for future refinement in subsequent updates.

Three build variants of 8.1a are available for download at www.itchybutt.org:

- SD2IEC Build
- Traditional Build
- VICE LTK Build (Windows & Linux, includes TCPSER script)

If you are currently running an existing 8.1 system, refer to the “Conversion from 8.0 or 8.1” section for upgrade instructions to 8.1.0a.

Many of the implemented modifications were either enhancements requested by active Sysops or features inspired by earlier 7.37-era releases preserved in the Tri-State Color 64 Sysop archives.

Technical Support

If you need help getting your system operational, troubleshooting configuration issues, or understanding the modifications included in 8.1.0a, you are welcome to reach out. Contact information and additional resources can be found at www.itchybutt.org.

An online version of this manual is available at color64wiki.itchybutt.org and will hold the latest updates.

For questions specifically related to version 7.37, OasisBBS.org is generally the best resource. For version 8.1 or 8.1.0a systems, I am happy to assist where possible.

Specific System Requirements

A wide range of hardware products have been developed for the Commodore 64 and 128, many of which introduce unique features or configuration requirements. Color 64 can accommodate most of these systems when properly configured. The sections that follow describe recommended setup procedures for various hardware configurations.

Important Notes

- Certain hardware configurations require special routines to be merged into the main overlays or supporting programs. These routines are provided as separate merge files on the distribution disks.
- Each hardware section explains whether a merge is required and how to apply it.
- All provided merge files are designed to be compatible with one another and may be combined without conflict.
- A BASIC merge utility such as BAID is required to apply these merges. Lt. Kernal users may use the built-in utilities provided with that system.
- Complete all mandatory configuration changes before starting your BBS system.
- Apply required modifications to the vbbs.xxx and corresponding “xxx small” overlays before placing them into active use.
- Do not overwrite the original Color 64 distribution disks with modified overlays. Always work from backup copies to preserve the original files in case future configuration changes are needed.

LT Kernal HD

The Xetec Lt. Kernal Hard Drive system is well suited for running Color 64. It supports up to nine Logical Units (LUs), each containing fifteen “users” that function similarly to subdirectories. This structure allows for as many as 135 distinct storage areas.

Color 64 cannot use Lt. Kernal and RAMDOS simultaneously if an REU is installed alongside the hard drive.

Hardware Requirements

When operating on a Commodore 64, the HIRAM connector must be properly installed as described in the Lt. Kernal documentation. Without the HIRAM connection, the system will not function correctly.

If you are using a Commodore 128 in 64 mode, the ribbon cable connecting the computer to the Host Adapter automatically handles the HIRAM signal.

Run the CONFIGURE utility and verify that the NMI TRAP setting is disabled (set to 0) for Commodore 64 mode. If NMI TRAP is not set to 0, callers may experience line noise during modem communication.

In the BOOTMAKER program, ensure that you answer “Y” to the Lt. Kernal configuration prompt. Failure to do so may result in garbled modem communication.

Accessing Logical Units

Color 64 accesses Lt. Kernal storage using the “ldlu” and “i” drive commands.

The syntax for the ldlu command is:

`<device><LU><USER>`

Example: To access LU 2, USER 5 on device 8: `l825`

For users 10 through 15, hexadecimal values A through F are used.

Example: LU 3, USER 11 on device 8: `l83b`

For LUs 2 through 9, you must also include an initialization command in SETUP.

Example: To initialize LU 3: `i3`

When combined: `l83b!i3`

Separate individual disk commands with an exclamation point (!).

Faster Disk Access

If your Lt. Kernal uses DOS version 7.1 or later, faster disk access can be achieved by merging optional enhancement routines into specific overlays. These merges are not required for normal operation.

Merge the following files:

- `lkf.init` → `vbbs.init`
- `lkf.msgs` → `vbbs.msgs`
- `lkf.xfer` → `vbbs.xfer`
- `lkf.ovl` → `vbbs.ovl`
- `lkf.nw1` → `vbbs.nw1`
- `lkf.nw2` → `vbbs.nw2`

Color 64 BBS Manual Version 8.1A March 2026

These routines replace the standard free blocks routine with a custom Lt. Kernal command. They assume the Lt. Kernal is configured as device 8.

The “lkf” merges are intended only for systems running Lt. Kernal DOS 7.1 or later.

ICT HD

Color 64 fully supports the InConTrol Data Chief HFD20 hard disk drive system.

Drive assignments are made using the ICT command structure. For example:

- hm4 11 22 — assigns a chain beginning at partition 11 and ending at partition 22
- h10 — assigns files to individual partition 10

Only certain file groups may be stored in chained partitions:

- Public Messages
- Help Files
- Text Files
- Uploads
- Downloads

The following file groups must be assigned to individual partitions:

- System Files
- Private Mail
- Caller Log
- Password File
- Boot Files
- Program Files

If you are using the Epyx Fastload cartridge, Program Files must reside on H0 (the built-in floppy drive). Placing Program Files elsewhere may result in intermittent system lockups.

The ICT system has been successfully tested with the Burst Mode ROM from Chip Level Designs. This ROM enables burst-mode access and can increase disk performance by approximately two to ten times.

If you are using a 1764 or 1750 RAM Expansion Unit, additional Color 64 modules allow ICT maintenance operations to be performed directly from within the BBS. One such module is included with this package (see ICT Utilities Module below).

Download Directory Recommendation

When configuring SETUP, it is recommended that you answer “N” to the question “Multiple Directories Per Drive.” From the perspective of Color 64, each chain behaves as a separate drive. Using one directory per drive or chain eliminates the need to rename files before downloads. While both methods function, this approach simplifies administration.

Note that scanning for new downloads may be slightly slower on the ICT compared to standard Commodore disk drives. This is due to the need to read multiple files across separate partitions. Despite this, performance remains acceptable, particularly when using the Burst Mode ROM.

ICT System Merges

If you are using chained partitions on an ICT DataChief or MiniChief system, the following merges are required. Without these merges, upload and download directories will not function correctly on chained partitions:

Color 64 BBS Manual Version 8.1A March 2026

- `ict.xfer` → `vbbs.xfer`
- `ict.ovl` → `vbbs.ovl`

ICT Utilities Module

Included with the Color 64 package is the `vbbs.ict` module for ICT users. This module provides:

- Multi-chain compression routines
- Automatic return to the “Waiting for Caller” screen
- Busy modem control
- Upload space scanning across chained partitions

The upload space scan routine reports total free blocks and maximum allowed upload size, allowing you to monitor when chains require compression.

The `Vbbs.ict` module is intended for systems using:

- ICT hard drive
- 17xx-series RAM Expansion Unit
- RAMDOS

It allows pattern-based file copying between partitions, chains, and drives. It also performs chain compression and modem control from within Color 64. During copy operations, screen activity will appear at the top display area due to the underlying machine language routines.

Installing the ICT Module

To load `vbbs.ict` into your RAM module, modify `Vsys.remove` to copy the module into memory. If `Vbbs.ict` resides on the Program Files drive, add the following line anywhere between lines 7100 and 7499:

```
7XXX &(8)="vbbs.ict":>310
```

If `vbbs.ict` is stored elsewhere, follow the instructions in the `Vsys.remove` documentation for adding modules to the script.

Next, modify `vbbs.xfer` so that typing “+” at the DOS prompt loads `vbbs.ict`. This can be accomplished by merging `ict.load` into `Vbbs.xfer`.

Once complete, typing “+” and pressing RETURN at the DOS prompt will launch the ICT module.

Important Note

The busy modem feature does not function with the Commodore 1670 modem, as it does not support the ATH1 command or equivalent modem-busy commands.

The `Vbbs.ict` module is intended specifically for ICT hard drive systems that also use RAM Expansion Units.

CMD HD

The Creative Micro Designs (CMD) HD series hard drives are well suited for operating a Color 64 BBS. Disk space may be allocated flexibly by dividing storage into partitions. In Native Mode, CMD partitions also support subdirectories, allowing dynamic organization of file groups without requiring fixed drive assignments.

If you plan to store your Program Files on the CMD HD, improved disk performance is strongly recommended. This can be achieved by:

- Installing JiffyDOS or another compatible fast disk system
- Using a Commodore 17XX-series REU with RAMDOS
- Adding a CMD RamLink device for parallel operation

The CMD HD should not be used in conjunction with fastloader cartridges unless the cartridge manufacturer explicitly confirms compatibility. Certain fastloader cartridges have been known to cause data corruption on CMD HD systems.

For details on selecting partitions and subdirectories, refer to the section titled “Drive Initialization Commands.”

The Real Time Clock

Sysops using a CMD HD or CMD RamLink equipped with a Real Time Clock may enable automatic time synchronization by merging the file `cmd.init` into `vbbs.init`.

Before using this merge, ensure the CMD device clock is set accurately. Once merged, `vbbs.init` will read the current date and time directly from the CMD device during system startup.

After merging `cmd.init`, the following lines may be removed from `vbbs.init`:

```
24030 24190 24210 24220 24230 24250
```

Line 24007 contains a GOSUB 481 call that selects the System Files drive in order to read the time. If your System Files are not located on the CMD device, this call may need to be adjusted.

Note: In `cmd.init`, line 24020 sets the year prefix. If the value is set to 19, the system will interpret the year 2024 as 1924. For current systems, this value should be changed to 20 unless operating in the 2100s.

CMD RAMLink

Color 64 operates very effectively with the CMD RAMLink from Creative Micro Designs. When used in conjunction with a CMD HD and connected via the parallel cable, overlay load times can be significantly reduced.

If you have a Commodore 17XX-series REU and prefer not to configure it as expanded memory for the RAMLink, you may instead operate it in direct mode and use it with RAMDOS. Follow the RAMLink manual instructions for accessing the REU in direct mode, then proceed with the standard Color 64 installation steps for configuring RAMDOS.

The CMD SwiftLink RS-232 interface is compatible with RAMLink and should be connected to the Pass-Thru port. However, SwiftLink will not function if the RAMLink is operating in direct mode, as the Pass-Thru port is disabled in that configuration.

If your RAMLink includes a RamCard with installed RAM, you may run Color 64 overlays directly from it. Be sure to maintain disk backups of any files stored on the RAMLink. Since `\bbs.parms` is read from the Program Files drive, verify that the file resides in the correct location for your configuration.

Additional file groups may also be stored on a RAMLink equipped with RamCard memory, but this is recommended only if the unit has battery backup. Without battery backup, a power interruption will result in complete data loss. Even with battery backup, it is advisable to store primarily static file groups—such as Help Files or Text Files—on the RAMLink.

If your RAMLink includes the Real Time Clock option, refer to the CMD HD section for instructions on enabling automatic date and time synchronization through the appropriate initialization merge.

Fastload Cartridge

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 -> vbbs.init
- fst.xfer -> vbbs.xfer
- fst.ovl -> vbbs.ovl
- fst.nw1 -> vbbs.nw1

Merge "fst.ovxx" into the following programs:

- vbbs.msgs
- vbbs.ov2
- vbbs.ov3
- vbbs.nw2
- vbbs.xxx
- xxx small

Skyles Flash!

Color 64 automatically detects the presence of the Skyles Flash! 1541 interface and will utilize it when available. The interface is supported only on device 8, so the 1541 drive connected to the Flash! must be configured as device 8.

Due to the storage limitations of a single 1541 disk, not all overlays may fit simultaneously. To accommodate the Program Files on a 1541 disk, you may need to omit certain overlays. This typically means excluding either:

- vbbs.ov2 and vbbs.ov3
or
- vbbs.nw1 and vbbs.nw2

Adjust overlay selection according to your system requirements and available disk space.

Avatex 1200

If you are using the original Avatex 1200 modem (not the Avatex 1200 HC or Avatex 2400), overlay modifications are required. The Avatex 1200 is not fully Hayes-compatible, and certain communication routines must be adjusted for proper operation.

Once these changes are applied, the modified overlays will no longer function correctly with a true Hayes-compatible modem. For this reason, create backups of all overlays before performing any merges.

Merge the following files into the specified overlays:

- `ava.init` → `vbbs.init`
- `ava.nw1` → `vbbs.nw1`

In addition, merge `ava.ovxx` into the following overlays:

- `vbbs.msgs`
- `vbbs.xfer`
- `vbbs.ovl`
- `vbbs.ov2`
- `vbbs.ov3`
- `vbbs.nw2`
- `vbbs.xxx`
- `xxx small`

After completing these merges, the system will be configured specifically for the Avatex 1200 modem.

2400 Baud Modems

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 Smart modems 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.

High Speed Modems

If you have a SwiftLink RS-232 cartridge or you are emulating it in VICE, 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 the 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.

Ram Expansion Unit (REU)

The Vsys.remove 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 "Vsys.remove" 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 "Vsys.remove" program.

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 "Vbbs.ovl" the source name would end up "Vbbs.ovl*" and the destination name would end up

Color 64 BBS Manual Version 8.1A March 2026

"vbbs.ovl." Thus, no matter what the version number of vbbs.ovl was, it would still be pried. You will notice that at line 2100, the "copy with version number" routine is used to transfer the vsys.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 vbbs.init, vbbs.msgs, vbbs.xfer, and vbbs.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.

The Built-in Copy Routines

Below is a summary of all the copy routines available in the vsys.remove program:

Table 6- vsys.remove 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.

Color 64 BBS Manual Version 8.1A March 2026

As you can see, almost any possible type of transfer need is taken care of. The script program as-will transfer just the essential files to the REU which are necessary for BBS operation.

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)="Voverlay":>310 This example would copy the file "Voverlay*" from the Program Files and store it on the REU as "Voverlay.". 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)="vgame.emp1":>310
3220 &(8)="vgame.emp2":>310
3230 &(8)="vgame.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

```

Color 64 BBS Manual Version 8.1A March 2026

```
3410 &(8)="vemp.menu1":>400
```

```
3420 &(8)="vemp.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.

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)="vfile":print"hello":&(8)="vfile2"**, 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 remainder 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.

Script-Merge Commands

Below is the list of the special Script-Merge commands:

Table 7 - 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	

Script-Merge Functions

The list below is the extra functions added by Script-Merge. They can be used in any expression if Script-Merge is active:

Table 8 - 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

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.

Preconfigured Builds

Three pre-configured builds of Color 64 v8.1a are available in the Downloads section. Each build includes customized overlays tailored to its intended configuration. The original, unmodified overlays are also included in the directory “D64 Masters” for reference or restoration purposes.

The 8.10a SD2IEC Build

This build is designed for use on a Commodore 64 or Commodore 128 operating in 64 mode with an SD2IEC device assigned as drive 8. The configuration was developed and tested using an SD2IEC device in combination with a WiModem (cbmstuff.com).

An REU (RAM Expansion Unit) is recommended but not required. While the system functions without one, an REU improves performance when loading larger overlays. Systems using JiffyDOS or similar acceleration solutions may also experience improved load times.

The download is provided as a 4GB SD card image. It may be written to an SD card using tools such as Balena Etcher or Raspberry Pi Imager.

Once written to a freshly formatted SD card, the system is immediately usable. No additional installation steps are required. Running +SETUP after first boot is strongly encouraged to review and customize configuration settings.

If you are using TurboMaster, SwiftLink, an REU, or Lt. Kernal HD, re-run the BOOTMAKER 8100 program to properly configure the system for your hardware. Otherwise, the system is ready to operate as imaged.

Included folders in this build are:

- Aux1: Auxiliary 1 drive; User profile area supporting the BBS.PROFILE module
- Aux2: Auxiliary 2 drive
- Aux3: Auxiliary / Games drive
- Uploads: Drive for uploads
- Downloads: Drive for downloads
- Dracopy: Copy program to help you on your way
- Help: Drive for help and text files
- Network: holds Network 1.26a supporting files
- Primsgs: Private messages drive
- Pubmsgsgs: Public messages drive
- Pswd: Password folder
- BBS Documents: for PC use – this manual.

The LTK Build

The LTK builds are designed for use with VICE and TCPSER on either Linux (tested on Kubuntu, Mint and Fedora 43) or Windows 11 systems.

These builds are pre-configured and are in a near ready-to-run state. Three Logical Units (LUs) are defined with the following assignments:

Logical Unit Assignments

Color 64 BBS Manual Version 8.1A March 2026

- LU 0 / User 0 – Programs, System files, Help files, Text files, Caller Log, Network files
- LU 1 / User 0 – Public and Private messages
- LU 1 / User 1–5 – Download categories 1 through 5
- LU 1 / User 15 – Uploads
- LU 2 / User 0 – Auxiliary 1 files (includes user signature information)
- LU 2 / User 1 – Auxiliary 2 files (not utilized in this build)
- LU 2 / User 3 – Auxiliary 3 files (Game area)

Additional Details

- LTK-specific performance modifications from the Color 64 v8.0 disks have been incorporated into the overlays as recommended by the original developers. See [LTK Specific System Requirements](#) for details.
- BOOTMAKER has already been completed.
- Modification of Main Parameters within +SETUP is not required but is strongly recommended prior to deployment.
- The MODEM INIT string in +SETUP is preconfigured for TCP SER operation at 38400 baud.
- Two games are pre-installed. Additional games are included on Disks 4 and 5 (D64 files) in this distribution.
- After adding or modifying games, update the file “Vgames menu” located on LU 2 / User 3. See [Adding Games](#) for instructions.

Additional details on this build:

- LTK-specific performance modifications from the Color 64 v8.0 disks have been incorporated into the overlays as recommended by the original developers. See [LTK Specific System Requirements](#) for details.
- BOOTMAKER has already been completed.
- Modification of Main Parameters within +SETUP is not required but is strongly recommended prior to deployment.
- The MODEM INIT string in +SETUP is preconfigured for TCP SER operation at 38400 baud.
- Two games are pre-installed. Additional games are included on Disks 4 and 5 (D64 files) in this distribution.
- After adding or modifying games, update the file “Vgames menu” located on LU 2 / User 3. See [Adding Games](#) for instructions.

LTK Installation:

Linux users:

- Extract the build into your home directory.
- Open a terminal in the extracted directory.
- Run: `./vicebuild.sh`

Windows users:

- Extract the build into a local user directory such as Documents.
- No build script is required.

Running the LTK Build

Linux:

- Open a terminal in the color64_v81a_ltk directory.
- Run: ./mybbs.sh

Windows:

- Open Windows Explorer.
- Double-click mybbs.bat.

This launches both VICE and TCPSER. Default port is 8502.

Operating the System

From the READY prompt:

- +BBS – Load and run the BBS
- +SETUP – Enter system configuration
- +EDITOR – Load the screen editor

Optional: Email Notification Feature

To enable email notifications of BBS activity:

- Install Python 3 if not already installed.
- Edit notify.py to configure SMTP server information and your chosen BBS login credentials.
- Modify vBBS.INIT at lines 35705–35706:
 - Ensure the username and password match those defined in notify.py.
 - Verify your PC's IP address is correctly defined.
- At PC, edit mybbs.sh and remove the remark from the python3 execution line.
- While the BBS is running, at the call screen press the Up Arrow key (mapped to "\" on a PC keyboard) to enable or disable notifications.

See Email Notification Option for complete configuration details.

Chapter 2, Installation

The installation process consists of three primary stages:

1. Copying the required program files
2. Running the SETUP program
3. Installing the system files

Detailed explanations of each SETUP option are provided in the following sections, along with descriptions of all required system files.

Important

Never use your original Color 64 distribution disks for anything other than the master source of the stock BBS files. Always work from backup copies. It is strongly recommended that you create backups before beginning installation to prevent accidental loss or corruption of the original files.

Before starting, ensure that you have sufficient uninterrupted time to complete the process. A careful and focused installation will reduce the likelihood of configuration errors.

Copying the BBS Files

Note: this step is not required for the pre-built versions (LTK/SD2IEC).

Once you have decided how to divide up your storage, it is time to copy all the files necessary to boot the BBS system. Below are the needed files for your Program drive as well as the Boot files for the boot disk. For larger disk systems, you can combine these two into one singular drive/area. For 1541 floppy disk users, the Boot Drive files will be your Boot disk. For 1571 users, if your program files disk has 700 or more blocks available, you can include the boot 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.

Table 9 - Color 64 Files to Copy

Program/Parms drive	Boot Drive
vbbs.init	bootmaker
vbbs.msgs	+bbs
vbbs.xfer	vsys.loadml
vbbs.ovl	vsys.mlinit
vbbs.ov2	vsys.setup
vbbs.ov3	vsys.edit
vbbs.term * <i>optional, if space avail</i>	pswd tools
vbbs.punt	dir tools
vbbs.xmo/c	menu maker
vbbs.ansi	bbs convert
vbbs.ascii	vsys.mlnorm * <i>or designated ML file per Table 2</i>
vbbs.profile	vsys.net * <i>if using BBS network</i>
vbbs.games * <i>Place in Aux 3 designated drive</i>	prscrn52750 * <i>if using BBS network</i>

Color 64 BBS Manual Version 8.1A March 2026

vbbs.nw1 * if using BBS network	
vbbs.nw2 * if using BBS network	

1. 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 files hold line changes that you will have to incorporate into the overlay the filename is identifying:

```

1  "afr.ovxx 8000"      prg
1  "ava.init 8000"     prg
1  "ava.nw1 8000"     prg
1  "ava.ovxx 8000"     prg
2  "bbu.nw2 8000"     prg
2  "cmd.init 8000"     prg
3  "enm.msgs 8000"     prg
3  "enm.nw1 8000"     prg
3  "enm.nw2 8000"     prg
1  "fst.init 8000"     prg
1  "fst.xfer 8000"     prg
1  "fst.ov1 8000"     prg
1  "fst.nw1 8000"     prg
1  "fst.ovxx 8000"     prg
3  "ict.xfer 8000"     prg
1  "ict.ov1 8000"     prg
1  "ict.load 8000"     prg
1  "lkf.init 8000"     prg
1  "lkf.msgs 8000"     prg
1  "lkf.xfer 8000"     prg
1  "lkf.ov1 8000"     prg
1  "lkf.nw1 8000"     prg
1  "lkf.nw2 8000"     prg
1  "ltk.dos 8000"     prg
3  "ndm.xfer 8000"     prg
3  "ndm.ov1 8000"     prg
3  "ndm.setup 8000"    prg
3  "ndm.net 8000"     prg
3  "ndm.dir 8000"     prg
3  "vnm.nw1 8000"     prg
3  "vnm.nw2 8000"     prg

```

In addition, you need to copy the specific BBS ML file to your Boot Files, depending on your setup identified in Table 6:

Table 10 - Machine Language File Selection Based on Configuration

Swiftlink	TurboMaster CPU	ML File to Copy
No	No	vsys.mlnorm
Yes	No	vsys.mlswft
No	Yes	vsys.mltmno
Yes	Yes	vsys.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.

2. 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:

Color 64 BBS Manual Version 8.1A March 2026

Table 11 - Additional Boot Files to Copy for REU Use

vsys.ramove vsys.ramcpy vsys.rdrein vsys.smerge vsys.ramdos

3. A Shortcut for Booting

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

- As mentioned above, 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
- Copy just the files necessary to boot the BBS system, identified in the table below, to your Program Files. This is only intended if you are not using the 17XX series REU:

Table 12 - Minimum Boot Files Required (Boot Files / Program Files combined on disk)

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

Creating the Boot Programs

Note: this step is not required for the pre-built versions (LTK/SD2IEC).

The next step in the installation process is to use the included *bootmaker* utility to generate the boot programs required to start Color 64. All boot programs begin with a “+” symbol. The +bbs file you previously copied to the Boot Files is an example.

These boot programs are essential because they store important configuration parameters, including whether you are using hardware such as the SwiftLink RS-232 cartridge or a TurboMaster CPU.

The bootmaker utility works by loading the +bbs program into memory, modifying it based on your selections, and then saving updated boot files back to disk. For this reason, the +bbs file must already exist on your Boot disk before running bootmaker.

Running the Bootmaker Program

Insert your Boot disk, then LOAD and RUN the program named:

bootmaker

During execution, you will be prompted for device numbers, drive numbers, and initialization commands.

- Device numbers must be between 8 and 30.
- Drive numbers must be 0 or 1.

Color 64 BBS Manual Version 8.1A March 2026

- Initialization commands may be up to 16 characters.

Refer to the section [Drive Initialization Commands](#) for proper syntax.

When entering drive initialization commands:

- Do not remove or add quotation marks.
- Multiple commands may be separated with “!”
- The boot programs can access only drive 0 or drive 1 of a device.

You may press RETURN to accept defaults where appropriate.

Bootmaker Questions Explained:

Device, Drive, and Init Command for Boot Programs

Enter the parameters for the drive containing your Boot Files. This drive must contain the +bbs program.

For SD2IEC users operating in native directory mode (not disk images), ensure your folder structure is finalized before running bootmaker.

Table 13 – SD2IEC Directory Layout Example

Folder	For	Drive/Initialization Command
ROOT	Program/System Files	0:!cd//
PUBMSGSGS	Public Message Area	0:!cd//pubmsgsgs
PRIVMSGSGS	Private Message Area	0:!cd//privmsgsgs
UD64UPLOADS	Upload Area	0:!cd//ud64uploads
UD64GAMES	Download Area – 64 Games	0:!cd//ud64games
UD64UTIL	Download Area – 64 Utilities	0:!cd//ud64util
HELP	Help Files / Text files	0:!cd//help

Running System with Ram Expansion Unit

You will be asked whether you are using a Commodore 17XX-series REU.

If this is your first installation, it is recommended to complete setup without the REU first. Once the system is stable, you may rerun bootmaker and SETUP to enable REU support.

Answer “Y” only if you intend to run overlays from the REU.

Boot Drive

Enter the device, drive, and initialization parameters for your Boot drive. This is the drive from which you will load +SETUP and other boot utilities.

This may be the same as your Program drive if sufficient space is available.

Program Drive

Enter the parameters for the drive containing your overlays.

If you are using a single disk drive and must swap disks during boot, use the same parameters as the Boot drive.

If using an REU:

- This refers to the device from which overlays will be loaded before transfer to RAMDOS.
- Drive will always be 0.
- ICT users should note that the included ICT utility assumes the REU is device 15.

Need to Swap Disks

If Boot and Program drives are identical and you have only one physical drive, answer "Y".

If using the smaller "bm small" utility, this question will not appear.

Using TurboMaster CPU

Answer "Y" if you are running the Schnedler Systems 4.09 MHz TurboMaster CPU.

Using SwiftLink Cartridge

Answer "Y" if you are using a CMD SwiftLink cartridge (or SwiftLink CRT image in VICE).

Using Lt. Kernal HD

If you answered "N" to SwiftLink, you will be asked whether you are using a Lt. Kernal hard drive. Answer "Y" if applicable.

SwiftLink Address

If SwiftLink is enabled, you must specify its memory address.

Factory default: \$DE00

Optional hardware modifications may set it to:

- \$DF00
- \$D700

If unsure, use \$DE00. If modem communication fails, verify the correct address.

Completion

After all questions are answered, bootmaker will create the necessary boot programs on the disk containing +bbs.

Standard boot files created include:

- +reboot
- +setup
- +net setup
- +editor
- +shell

If REU support was enabled, the following will also be created:

- +ram.start
- +ram.restart
- +ram.reinit
- +ram.bbs
- +ram.reboot

These files are approximately three disk blocks each and will be used regularly during system operation.

Boot File (“+”) Program Details

All boot programs begin with a plus symbol (“+”). The primary entry point during installation is +setup, which is used to configure your BBS. You will return to +setup frequently as your system evolves.

When a boot program is loaded and run, the following actions occur:

- The system performs an initial reset. The screen clears and default colors are restored to ensure a clean operating state.
- Required support files are loaded into memory. The specific files depend on the boot program selected.

Boot programs are the proper and supported entry point into the system. They contain essential configuration information such as device numbers and initialization parameters for your Boot and Program drives. Always start Color 64 using the appropriate boot file.

Boot File Descriptions

The table below describes what each of the boot programs does and the additional files it will load:

Table 14 - Boot Files Functional Description

Filename	Functional Description	Load Sequence
+bbs	Boots the main BBS and proceeds to the date and time prompts.	1) vsys.loadml 2) vsys.mlinit 3) v)sys.mlnorm 4) vbbs.init
+reboot	Same as +bbs, but automatically accepts default responses to the date/time and regenerate index prompts.	
+setup	Boots the SETUP utility.	1) vsys.loadml 2) vsys.mlinit 3) v)sys.mlnorm 4) vsys.setup
+editor	Boots the stand-alone message editor.	1) vsys.loadml 2) vsys.mlinit 3) v)sys.mlnorm 4) vsys.edit
+shell	Loads the ML shell into memory and returns to the BASIC READY prompt. Used when running stand-alone utilities that require ML to be resident.	1) vsys.loadml 2) vsys.mlinit 3) v)sys.mlnorm
REU Boot Files (if REU option was set)		
+ram.start	Copies overlays to the REU, installs RAMDOS, and boots the BBS.	1) vsys.ramove 2) vsys.ramdos 3) vsys.smerge 4) vsys.ramcpy 5) +ram.bbs
+ram.restart	Same as +reboot, but reloads overlays into the REU before restarting.	Refer to +reboot
+ram.reinit	Reinstalls RAMDOS after a system reset so REU contents can be accessed.	vsys.rdrein
+ram.bbs	Boots the BBS using overlays stored in the REU instead of disk. RAMDOS must be active.	Refer to +bbs
+ram.reboot	Same as +ram.bbs, but automatically accepts reboot prompts.	Refer to +reboot
Network Setup Boot File (if Network option was set)		
+net setup	Boots the Network Setup Utility	1) vsys.loadml 2) vsys.mlinit 3) v)sys.mlnorm 4) vsys.net

The ML Shell

Most boot programs begin by loading `\sys.loadml`. This installs the machine language (ML) core into memory and displays the Color 64 title screen.

Once installed, the ML shell remains resident until the computer is powered off or another program overwrites critical memory areas.

Several stand-alone utilities require the ML shell but do not automatically load it. To use them, run `+shell` first if the ML is not already resident. These utilities include:

- `pswd tools`
- `dir tools`
- `menu maker`
- `bbs convert`

SETUP Parameters

Defining system parameters is one of the most important steps in configuring your BBS.

Before running SETUP:

- Ensure the drive designated for the password file is online.
- Confirm that the disk or partition is formatted and empty.
- Verify that sufficient free space exists.

SETUP creates a relative file large enough to support the maximum number of callers you specify.

To begin:

1. Insert your Boot disk (if applicable). 2. LOAD and RUN `" +setup"`.

It is recommended that fastloader cartridges be disabled during the initial SETUP run. JiffyDOS may remain enabled. Some systems may produce a "NO CHANNEL" error when creating new relative files if fastload is active. After the password file exists, fastload is generally safe for normal operations.

You may be prompted to insert your Program disk during SETUP.

The "`\bbs.parms`" File

All BBS parameters are stored in a sequential file named `\bbs.parms`. This file is always read from the Program Files drive so that it remains accessible to overlays during operation and recovery from crashes.

For REU users, `\bbs.parms` is never copied into RAM. It is always read from disk.

If you prefer to store `\bbs.parms` on a disk separate from your overlays:

- Answer "Y" to the disk swap question during bootmaker configuration.
- Insert the desired disk when prompted during SETUP.
- Ensure the correct disk is present whenever the BBS needs to read parameters.

If `\bbs.parms` cannot be found at startup, the system will prompt for the correct disk.

Once the proper disk is inserted and accessed, the Main Parameters screen will appear:



At each SETUP prompt:

- The current or default value appears in brackets.
- Press RETURN to accept the displayed value.
- Enter “-” and press RETURN to return to the previous question.

SETUP - Main Parameters

Note: for pre-built versions (LTK/SD2IEC) of this software, parameters are already configured and your \bbs.parms file has already been created. However, you will want to modify these parameters to your liking. The modem initialization string is already configured for use with TCPSER.

SETUP is where you define how Color 64 will operate on your system: message limits, credit rules, modem behavior, disk assignments, directories, time limits, categories, and command permissions. If you are unsure about a specific prompt, it is usually safe to accept the default value shown in brackets and fine-tune later. The notes below explain how each option affects day-to-day operation.

Table 15 – Main Parameters Descriptions

Question	Description and Settings
Maximum lines per message	Limits the number of lines permitted in a single message. Default: 100 Range: 20–200
Maximum columns per line	Sets the default word-wrap width used by local-mode tools (such as the editor). Default: 38 Suggested: 38 for 40-column systems; 78 for 80-column output.
Maximum # of messages	Maximum number of public messages retained online. When the limit is reached, older messages are deleted as new ones are posted. Default: 50 Range: 25–232 Considerations: 1541 disks allow only 144 directory entries per disk. SFD-1001 allows 224. Hard drive style systems may allow far more, but the BBS itself tops out at 232.
Maximum password number	Sets the maximum number of member records in the vpassword file. Default: 100 Range: 25–9999

Color 64 BBS Manual Version 8.1A March 2026

Question	Description and Settings
	<p>Considerations: 1541 limit is roughly 720 records. If exceeded, new callers will receive the "Vmembership full" message when attempting to apply. Roughly 1 disk block is used per user record.</p> <p>This value can be increased later, but expanding repeatedly over time may fragment the file and slow access. See "Password File Fragmentation" hints below for a clean rebuild method.</p>
Minimum blocks-cycle msgs	<p>Minimum free blocks required on the Public Messages drive before messages begin cycling.</p> <p>Default: 50 Recommended: 25–75</p> <p>If free blocks drop below this value, each new public post will delete the oldest public message even if the maximum message count has not been reached.</p>
Minimum blocks-allow uploads	<p>Minimum free blocks required to permit uploads.</p> <p>Default: 75 Recommended: 10–75</p> <p>If private messages share this same disk, do not set below 25. The BBS will not allow private mail to be sent if free blocks fall below 25.</p> <p>Never set below 10. Space is required for directory processing and temporary files after uploads. The "free upload space" shown to callers is automatically reduced by this value.</p>
Maximum downloads per call	Maximum number of downloads allowed per caller session. Enter 99 for a practical "no limit." Callers who reach the limit must log off and back on to continue downloading. Exemptions can be granted via "No DL file limit level."
Download credits per call	Sets how many download-credit blocks a caller earns per block uploaded.
New mbr download credits	Free download credits granted to each new caller. Once used, callers must upload to earn additional credits (unless exempt).
Credit system exempt level	Access level at which callers become exempt from the credit system. If you are not using credits, set to 1 so all callers are exempt.
Max files on public msgs drive	<p>Maximum directory entry count permitted on the Public Messages drive. The system checks after each user logoff.</p> <p>Recommendations: 1541: ~135 messages SFD-1001: ~210 SD2IEC / LTK: ~200–250, or enter 0 to avoid file counting</p> <p>Higher limits increase message indexing time at boot and may reduce available memory.</p>
Number of days to hold mail	<p>Maximum days private mail is retained before automatic deletion.</p> <p>When mail is purged, an entry is written to the caller log at midnight indicating the member number affected. A safety buffer prevents mass deletion if the system clock is accidentally wrong: if calculated mail age exceeds the purge value by more than 7 days, that mail will not be deleted.</p>
New user access level	<p>Sets the access level assigned to new callers. Some sysops prefer Level 1 until validation; others start users higher. Default level meanings are summarized below.</p> <p>Typical use of levels (defaults):</p> <p>1 - Recommended for unverified new users</p> <ul style="list-style-type: none"> • Read System Messages • Create Profile/Signature • Very limited system time

Color 64 BBS Manual Version 8.1A March 2026

Question	Description and Settings
	<p>2 - "Browse" level</p> <ul style="list-style-type: none"> Includes Level 1 Adds read access to public messages and text files <p>3 - Standard contributor level</p> <ul style="list-style-type: none"> Includes Level 2 Adds posting, uploads, and downloads <p>4-5 - Full normal access</p> <ul style="list-style-type: none"> Same abilities; use categories, time limits, and directory levels to differentiate Common network approach: Level 4 non-network; Level 5 network-approved <p>6 - Privileged callers</p> <ul style="list-style-type: none"> Exempt from per-call time limit No download limit per call No time-between-calls restriction <p>7 - Helper sysops/moderators</p> <ul style="list-style-type: none"> Scratch public messages or downloads Edit download descriptions Release uploads <p>8 - High-level sysop (co-sysop)</p> <ul style="list-style-type: none"> Set clock/date Change user access levels (cannot elevate to sysop) Read caller log Warning: Level 8 can view other users' passwords <p>9 - Full sysop</p> <ul style="list-style-type: none"> Password maintenance Full DOS functions Should be limited to you (and possibly one trusted backup)
Mbrs expired access level	Level assigned automatically when a caller's membership expiration date is reached. The expiration date is stored per user record and is checked at midnight. Useful for trial memberships or time-limited access.
Upload auto-release level	Determines what access level is required for an upload to be automatically released as a public download. Lower-level uploads can be held for sysop review.
System Baud Rate	<p>Sets the maximum communications rate between the computer and modem/interface. The system can step down based on the caller's capabilities (and modem requirements).</p> <p>If you are not using SwiftLink, the practical maximum is 2400. With SwiftLink, 38,400 BPS is supported in many configurations, including TCPSER, where the computer-to-modem rate can remain high even if the modem-to-modem connection is lower, depending on the "Adjust BPS to connect rate" setting.</p>
DD\$ MCI on	Controls whether MCI commands may print the DD\$ field from a caller's password record. DD\$ is an 8-character sysop-defined note field (for example SYSOP, GUEST, REMOTE). Answer N to keep DD\$ private from MCI output.
Using Upload Descriptions	Enables download description files (stored as @filename). Most sysops should answer Y. If directory-entry limits are a concern (notably on SFD-1001), descriptions reduce the maximum number of downloads because each file consumes an additional directory entry. Manual description creation remains possible via the Edit Download Description command.
Mutiple dirs per drive	Enables multiple logical directories on the same physical drive. When enabled, the BBS prefixes uploaded filenames internally with a letter (A-Z) for grouping. Callers do not see the prefix. If you manually add files, you must rename them with the correct leading letter. If you answer N, no renaming is required and everything resides in a single directory.
Daily Log Backup	Enables nightly caller log backup and rotation at midnight. If N, the caller log remains a running recent-history file. If Y, the log is dated and archived daily, then cleared. Log size trimming is governed by

Color 64 BBS Manual Version 8.1A March 2026

Question	Description and Settings
	"Caller log max size blocks" and "Caller log trim blocks," as well as the "Minimum blocks-allow uploads" threshold.
Rerun on Errors	Restarts the BBS automatically after BASIC errors. Normally this should be enabled. Disable only while actively modifying/testing overlays. When enabled, STOP is disabled; breaking requires SHIFT+COMMODORE+CTRL.
Screen Blanking	If enabled, blanks the local sysop screen when callers read or write private mail (excluding feedback to sysop and console-local use). Use this based on your privacy philosophy and moderation needs.
Does your modem support DTR	Indicates whether your modem/interface supports DTR drop for disconnect. Most Hayes-compatible modems and TCPSER support DTR. Some hardware behaves better with DTR disabled (notably certain 1670 setups). If unsure, answer N.
Adjust BPS to connect rate	Controls whether the computer should step down its BPS rate to match the modem connect rate. Many 1200/2400 modems expect this; some high-speed setups (including many TCPSER configurations) can run 38,400 computer-to-modem regardless of connect rate. This may require experimentation depending on your modem.
Run Network v1.26a	Enables Color 64 Network features (sharing messages/files between participating systems). New sysops should normally answer N until the system is stable and coordination with another network sysop is complete. If enabled, you must run +net setup afterward.
Modem Init Command	Initialization command sent to the modem before accepting a caller. Must start with lowercase "at". Recommended strings vary by modem type. Verbose mode (v1) is essential. If the BBS shows OK at init, the string is accepted; if it shows ERROR, remove unsupported parts and test again in a terminal. Examples commonly used: <ul style="list-style-type: none"> • WiModem: ate0x1s0=0s10=30v1 • TCPSER: ate0v1h0x1m0b1
Network Modem Init Command	Sent before each outgoing network call. Some sysops disable error correction for network calls if their modem's MNP interferes. Default (ate0) has worked well for many v8.1a network setups.
Scratch any msg level	Level required to delete any public message (not just the caller's own). Default: 7
Category/Link level	Level required to change message category or thread link. Default: 7
Merge seq file level	Level required to merge a SEQ file into a message via the editor "*" command. Default: 7
Caller log delete level	Level required to delete the caller log. Default: 8
Message MCI level	Level required to use Message MCI commands. Default: 2
Variable MCI level	Level required to use Variable MCI commands. This should be restricted to trusted sysops due to crash potential from misuse. Default: 9
No DL file limit level	Level exempt from "Maximum downloads per call." Default: 6
Min msg memory bytes	Minimum free memory permitted while editing before the system stops accepting new lines. Higher values reduce garbage-collection pauses; lower values allow larger messages. Typical values: 300–700.
Caller log max size blocks	Maximum size of the caller log file (blocks). If exceeded, the log is trimmed using "Caller log trim blocks." Default: 50 Range: 8–200
Caller log trim blocks	Blocks removed when trimming is triggered (by size or low disk space). Should not exceed half of "Caller log max size blocks." Default: 8 Range: 4–100
Validate via email level	Level required to validate a caller through e-mail workflows. Default: 8
Edit any message level	Level required to edit any public message (not only the caller's own). Default: 9

Color 64 BBS Manual Version 8.1A March 2026

Question	Description and Settings
Max chars/40 column header	Maximum header width when output is set to 40 columns. Default: 30
Max Chars/80 column header	Maximum header width when output is set to 80 columns. Default: 70
No time restriction level	Level exempt from per-day time restrictions set in Time Limits. Default: 6
Edit DL Description	Level required to edit download descriptions after viewing them. Default: 7
Use fast garbage collect	Optional performance behavior. Leave at default unless you are tuning for stability/speed on your system. Default: N

Additional hints:

Password File Fragmentation

If you expand the password database repeatedly and suspect fragmentation (slow reads), you can rebuild cleanly:

1. Use pswd tools to back up the vpassword file.
2. Create a new blank disk/partition.
3. Rerun SETUP to create a new password file at the new desired maximum size.
4. Use pswd tools to restore password records.
5. Copy remaining files from the old disk to the new disk.

This restores contiguity and can significantly improve performance.

Credit Exemptions

If a caller uploads files you do not want to count toward credits, adjust their “blocks uploaded” total in Password Maintenance to remove the undesired credit.

A utility program (pswd tools) can reset upload/download counters for all callers or a single caller if you want to restart the credit system.

Credits are calculated as:

$(\text{free credits}) + ((\text{uploads}) \times (\text{credits per upload})) - \text{downloads}$

If credits become ≤ 0 , the caller must upload more before downloading again.

SETUP - Disk Drive Assignments

Note: This is already configured for the pre-built versions (LTK/SD2IEC).

This section assigns each file group to a drive. It is where you balance storage across devices for performance and capacity. If you later change your mind, you can rerun SETUP, adjust assignments, and move the affected files.



Color 64 BBS Manual Version 8.1A March 2026

Each assignment prompt requests four values (device, drive, and init):

```

Drive to edit (A..L)? [A]

Password File
-----
Device (8..30)
[8]
>

Drive!Init
[0:!1800!i0]
>

Force free blocks=30000 (Y/N)
[N]
>

Non-standard dir filter (Y/N)
[N]
>

```

The device number range is 8–30. The drive number is typically 0 or 1. The init string defines how the system selects the correct partition/LU/subdirectory for that file group. The example above shows an Lt. Kernal selection of LU 0/User 0 (“!l800”) plus initialization (“!i0”). These commands are sent to the command channel whenever the associated group is accessed. See “Drive Initialization Commands” for system-specific guidance.

Table 16 – Disk Drive Assignment Descriptions

Question	Description and Settings
Password File	Location of the vpassword file. The assigned drive must support REL files. SFD-1001 supports REL well; ICT chained partitions do not. 1581 and SD2IEC may require REL creation on a 1541/1571 first, then copying with a REL-aware copier.
System Files	Welcome/logoff messages, menus, membership list, membership-full file, and other system text/data. Drive should support REL files.
Help Files	All caller-readable help content.
Public Messages	Public message base and supporting files.
Private Messages	Private mail plus vquestxx new-user application files.
Text Files	Caller-readable text files outside the help system.
Caller Log	Current caller log, “most recent call,” and optionally daily backups.
Program Files	All BBS overlays plus required ML files. If using REU, this must match the REU device number defined by RAMDOS.
Network Files	Network data and files (if Network enabled). Must support REL files.
AUX 1 Files, AUX 2 Files, AUX 3 Files	AUX 1: User Profile area (signatures and banners). AUX 3: Games area (if used). AUX 2: Reserved (unused) for future expansion.

IMPORTANT NOTE: If you are using the Epyx Fastload cartridge, you must use “ui” as the drive command for Program Files or you may experience intermittent lockups.

After completing the assignments, SETUP will ask “Is this correct?” Answer N to revise, or Y to continue to Upload/Download Directories.

SETUP - Upload/Download Directories

This section defines your upload/download directories (A–Z). Each directory has a name, upload/download status, access level, and its own device/drive/init settings.

Color 64 BBS Manual Version 8.1A March 2026

You must create at least one upload and one download directory, even if you intend to restrict access. To effectively disable transfers for callers, set their access level to 9 (sysop) or 10.

```

Number of Directories (1..26)
[6]
>

A Uploads
B category 1 at lu 1 user 1
C category 2 at lu 1 user 2
D category 3 at lu 1 user 3
E category 4 at lu 1 user 4
F category 5 at lu 1 user 5

Directory to edit (A..F)? ☐

```

You will be asked to enter the following information described in the below table for each directory:

Table 17 - Directory Information Query Fields

Information Asked	Details
Description	Display name for the directory (up to 30 characters). Quotes are not permitted.
Allow Downloads	Enables downloads from this directory. The first directory marked Y becomes the default download directory.
Allow Uploads	Enables uploads to this directory. The first directory marked Y becomes the default upload directory.
Access Level	Minimum level required to use the directory (unless it is the default upload/download directory). Use 10 to effectively disable.
Device, Drive, Init	Same structure as Disk Drive Assignments.

When finished entering directories, you will return to the edit prompt. Press RETURN to continue. If SETUP loops back instead of continuing, you likely did not mark at least one upload and one download directory as enabled.

SETUP - User's Time Limits

Time limits apply by access level:

- Per-call limits for Levels 1–5 (AM and PM), with higher levels exempt by default (controlled by “No time restriction level”).
- Time-between-calls delay for Levels 1–5.
- Daily time limits for Levels 1–9.

If a caller exceeds their time while downloading or composing a message, the system will not disconnect them mid-action. Instead, it “borrows” the extra time from the next day by storing a negative balance and restoring time nightly at midnight.

At the end, SETUP will ask “Is this correct?” Answer N to revise, or Y to continue.

SETUP - Caller Purge

Caller purge automatically removes inactive accounts based on access-level thresholds you define. Deleted accounts are permanently removed unless you restore from a vbackup password file.

The purge runs nightly at midnight. A safety check prevents mass deletion if the system clock is incorrect: if the calculated inactivity exceeds your purge value by more than 7 days, the record is not deleted.

At the end, SETUP will ask “Is this correct?” Answer N to revise, or Y to continue.

SETUP - Message Categories

Defines message categories (2–18). Each category has a required access level. Callers below that level will not see or access the category.

Avoid quotation marks in category names; quotes can cause issues when reading vbbs.parms.

At the end, SETUP will ask “Is this correct?” Answer N to revise, or Y to continue.

SETUP - BBS Commands

This section lets you define each command key and the minimum access level required. You can change command letters to match your style, reserve features for higher levels, and disable unused “spare” commands by setting them to level 10.

Table 18 - 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
↑	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 available text files for viewing

Color 64 BBS Manual Version 8.1A March 2026

Key	Name	Level	Description
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 (8.0/8.1 default)
1	Games (8.10a default) - note that "" is still available, if desired	4	Takes user to games menu
2	User Profile	1	Allows user to customize email signature/banner
%	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	

* Post Office includes: Post a Message, Read/Send E-mail, Feedback, and Membership List. Options not permitted by the caller's access level will not be shown.

** User Settings include: page-pauser lines, character delay, and 40/80 column selection.

*** Network commands are disabled if Network is not enabled. See the Network documentation for details.

SETUP - Color Code Setup

Defines the eight color values used by the system. To change a bar, type its number and then the color control code (example: 1 then COMMODORE/7 for medium blue). Adjust until satisfied, then press RETURN to continue.

SETUP - Carrier Status

Determines how the system detects modem carrier. This is critical for stable operation. Ensure the modem is connected, powered on, and not actively connected to a caller. If necessary, disconnect the phone line to avoid false detection.

If you change modem types or relevant modem switches later, rerun SETUP and repeat this section.

SETUP - Saving the Parameters

After completing all sections, return to the SETUP main menu and select option 10 to save parameters.

Two key files are created:

- \bbs.parms (stored on the Program Files drive). This includes drive assignments and system parameters required for normal operation and recovery.
- \password file (created on the drive assigned in Drive Assignments). This stores caller records including names, passwords, access, time remaining, last message read, etc.

After the initial SETUP, the password file includes a SYSOP account (member number 2) with name SYSOP and password SYSOP. Member #2 must remain reserved for SYSOP because feedback messages are delivered to this mailbox.

Color 64 BBS Manual Version 8.1A March 2026

Before opening the board, change your sysop name/password using Password Maintenance (F6 at the waiting-for-call screen or "<" while online).

The "√" (shifted @) prefix on system files hides them from callers and prevents download access.

SETUP – Editing/Changing the Parameters

To change any parameters later, rerun +setup. The existing √bbs.parms file is loaded, and previously saved values appear as defaults. Press RETURN to keep values you do not want to change, and edit only the items you do.

When finished, select option 10 again to write an updated √bbs.parms file.

If you increase "Maximum password number," SETUP will ask whether to expand the password file. Expanding can be done immediately (Y) or record-by-record later (N). Record-by-record expansion can be slower during live operation.

SETUP - Printing the Parameters

Option 12 prints the parameters. A printer must be available as device 4 and be online. You may print a single section or all parameters.

Prior to Running the BBS

Items to Review Before Going Live

- In +SETUP:
 - Main Parameters
 - Message Categories
 - Upload/Download Directories
- Update SYSOP account (#2).

Default credentials: User: SYSOP Password: SYSOP

- If using the Email Notification feature:
 - Edit `\bbs.init`:
 - Line 35705 – Enter your IP and port and replace "username"
 - Line 35706 – Replace "password"
 - Copy `notify.py` to your computer and edit:
 - Username
 - Password
 - SMTP server
 - SMTP server password
 - See [Email Notification Option](#) for details
- Customize SEQ files as needed:
 - `\sayings1`–`\sayings6` (optional logon sayings)
 - To change the number of sayings, edit `\bbs.ov3` line 170:
 - `170 i=int(rnd(1)*6)+1`
 - Change the "6" to match the number of sayings you plan to use.
 - `\sysopin` and `\sysopout`
 - `\welcome1` and `\welcome2`
 - `\systemstart` and `\systemstart2`
 - `\information`
 - `\level1`–`\level9` messages (optional)
 - `\member list msg`
 - `\membership full`
 - `\menu1`–`\menu9`
 - `\new user msg1` and `\new user msg2`
 - `\no mail`
 - `\po menu`
 - `\sysop not here` and `\still not here`
 - `\sysop news` (optional)
 - `\upload message`
 - `\chat enter` and `\chat exit`
- AUX3 (Games Area):
 - Modify `\games menu`
 - Edit `\bbs.games` starting at line 36200 for command handling and game loading

The System Messages

With your parameters defined and your Program and Boot files properly installed, it is time to create and customize your System Messages. This is where your BBS begins to take on its personality.

The install disk includes sample system message files. Many contain placeholder content explaining when the file is displayed or what it is used for. Use a file copier to transfer all desired sample "system messages" onto the drive assigned for your System Files.

As noted earlier, all BBS system filenames must include the check mark character (shifted @) as the first character of the filename.

System Files for BBS Operation and Customization

There are several ways to create or modify system messages:

- A word processor that saves SEQ text files (such as Easy Script)
- The stand-alone BBS message editor
- The built-in message editor within the BBS DOS section
- Kaleidoscope (recommended for menu-style screens)

To use the stand-alone message editor, load and RUN the program "+editor" from your Boot disk (for floppy-based systems).

After loading, you will see a menu similar to this:



The "Read Newsletter" option is a legacy item from the original 8.0 release when a Color 64 newsletter was planned. It remains as an artifact of that era.

Editing a Message File

To edit a message on any drive:

- Press F1.
- Enter the device number of the drive that contains (or will contain) the message file. If the number shown in brackets is correct, press RETURN.
 - Enter the drive number (0 or 1). Again, press RETURN to accept the default.
 - Enter any drive initialization command if required.

Normally, you will press RETURN for the init command. However, special configurations may require commands such as:

Color 64 BBS Manual Version 8.1A March 2026

- `u0>h1` for the back side of a 1571
- `u0>m1` to place a 1571 into 1571 mode
- Hard drive partition commands as needed

Finally, enter the filename — remembering to include the required prefix character:

- “√” for protected system files
- “@” when appropriate for hidden description files

If the file exists, it will load into memory for editing. If it does not exist, it will be created when saved.

Example screen:

```
Device # (0-30) [0] >
Drive # (0-1) [0] >
Drive command >
Filename > √systemstart

Opening 0:√systemstart, 8

√system start
this is the first screen displayed to
the user at connection. you should
keep everything lower case in case the
caller is not a commodore.

(L)ist (E)dit (C)ont. (D)elele
(I)nsert (S)end (N)ew (Q)uit
(H)elp (*)merge (!)column (R)eplace
(F)ile (M)ci:On >
```

Editor Capabilities

The stand-alone editor functions the same way as the online message editor used for public and private messages.

Key differences:


- The stand-alone editor allows messages up to approximately 500 lines.
- It supports full color control codes.
- The online editor is limited to the “Maximum lines per message” value defined in the “√bbs.parms” file.

This makes the stand-alone editor ideal for creating longer system files such as welcome screens, information files, menus, and help documentation.

System Messages Definition List

The table below provides the list of system message files that Color 64 uses:

Table 19 – System Message Files

File	Example	Description
All files are located in the Systems drive unless otherwise noted!		
√systemstart		The very first file displayed when a user connects. Keep this in lower case and graphics-free in case the caller is using ASCII (it will appear upper case to

Color 64 BBS Manual Version 8.1A March 2026

File	Example	Description
		them). Common uses include BBS name, date, time, or an initial greeting.
√systemstart2	<pre>commodore graphics check: please press your del or backspace key</pre>	Immediately follows √systemstart. Edit it and include verbiage to the user to press DEL or Backspace for graphics detection. Should remain lower case and non-graphical.
√banner √abanner	<pre>Commodore Graphics Enabled! Westwood BBS Proud member of the Color 64 Network Itchy Butt * Arrakis * Darkstar</pre>	Displayed after graphics detection. √banner = Commodore (PETSCII) terminals √abanner = ANSI terminals Graphics are fully permitted here.
√welcome1	<pre>WESTWOOD A COLOR 64 8.1A BBS PRESS ANY KEY TO CONTINUE Please enter your user information:</pre>	Main greeting screen shown after graphics check and before credential entry.
√welcome2	<pre>Login Complete! Welcome!</pre>	Optional. Displayed after successful login. Can be graphical or minimal.
√logon stats √logon stats80	<pre>The Cabin BBS User Name: Nuke The SysOp Welcomes Nuke Your Access Level : 8 Times Called Here : 65 You Last Called on: 08/26/2025 Time Allowed/Call : 300 Minutes Time Allowed/Day : 300 Minutes ----- System Calls Today: 3 Msgs Posted Today : 0 New Msgs To Read : 3 press any key...</pre>	Shows current BBS and account status. √logon stats = 40 column √logon stats80 = 80 column Uses extensive Variable MCI commands. Edit cautiously.
√sysopin √sysopout	<pre>The Sysop is out doing errands...</pre>	Displayed depending on whether the SYSOP flag is set to "In" or "Out." Graphics permitted.
√level # msg	<pre>Sysop Access Identified May the force be with you.</pre>	Level-specific message displayed after login. Create files such as "√level 3 msg" or "√level 7 msg." If added or removed while running, press F4 at WFC to reset flags.
√wall	<pre>Artist: Nuke on 3/21/2025 I'm on the wall! W00t! Backintime on 08/02/2025 checkin out the new new westwood! ----- Sign Guestbook? [N]</pre>	Optional user wall / guestbook. Remove command "W" from BBS Commands in +SETUP if deleting.
√sysop news	<pre>08/01/2025 Westwood has transitioned over to Color 64 v8.1a software! Please excuse the mess while construction is occurring. 08/11/2025 Transition is pretty much completed to the new BBS engine. If anyone sees anything awry, please leave me feedback. Thanks for your patience!</pre>	Displays dated SYSOP news entries. Each user only sees unread entries. Keep concise. New entries should have a date immediately following the last line of previous entry. Keep in mind that you don't want this file to get too long. New users will see every single news entry when they go through their logon process. This file is optional.

Color 64 BBS Manual Version 8.1A March 2026

File	Example	Description
vmenu#	<pre> Menu Commands - Level 4 Message Base: File System: R Read Messages D Download Files P Post Message U Upload File O Post Office/email \$ List Files E Edit a Message # Change Drive F Feedback to Sysop + Multi-Upload S Scratch a Message Z View D/L Desc N New Downloads % Sel Protocol C Chat with Sysop 1 Enter Games General Items: Admin: W Mail/Guestbook G Graphics Y/N H Help 1 User Settings M Member List 2 User Profile I System Information A Change Passwd T System Text Files 0 Logoff </pre>	<p>Main menu screen per access level (e.g., vmenu4).</p> <p>Can be graphical.</p> <p>Multi-page menus are possible but may slow navigation.</p>
vchat enter	<pre> The Sysop has wandered in.... </pre>	Displayed when chat session begins.
vchat exit	<pre> The Sysop has headed out..... </pre>	Displayed when chat session ends.
vnew user msg1	<pre> Please be sure to record your user number and password! </pre>	Displayed after new user account assignment. Keep short and avoid screen clears.
vnew user msg2	<pre> Your information has been saved! Thanks for joining in and we look forward to your participation! Enjoy!! </pre>	Displayed after application completion. Include instructions for validation if needed.
vapplication	<pre> Optional:Please enter your full name: #(rn\$)My full name is Please enter your email or "N/A" #(a1\$)My email is Enter your City & State or "N/A" #(a2\$) City, State </pre>	<p>This is displayed to the user, but the display will be different than what you see here as it is being run as a script.</p> <p>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 "vquestXX" where XX is the number of the month (e.g. vquest07 for July). The "vquestXX" 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 "vquestXX" file.</p> <p>These days, people are uncomfortable with providing their "home address" or even their birthday – not to mention both together. Think</p>

Color 64 BBS Manual Version 8.1A March 2026

File	Example	Description
		about what information is pertinent when it comes to your user and the use of your BBS. On my BBS, it's a 50/50 shot that I will even get their full name, let alone their first. That said, if you feel you don't have enough to trust the user with a C64 BBS system, then you can always delete them or not approve access. See the section on "Application" for more information.
vbbs closed		Create this file to prevent new users from registering. If this file exists, the BBS will not allow new user registration. Lower case recommended for ASCII compatibility.
vmembership full	<pre>Sorry; Membership is full at the moment.</pre>	Displayed when maximum user limit has been reached.
vmember list msg	<pre>You may not be listed yet if the sysop has not reviewed your application.</pre>	Displayed before membership list.
vmembership list		Automatically generated user list.
vinformation	<pre>Westwood BBS King George VA USA Running on Color 64 v8.1a with Vice / Lt. Kernal build</pre>	System information file describing hardware, configuration, or other details.
vno mail	<pre>Sorry!! No private mail today!</pre>	Displayed when no private mail is waiting. Optional.
vsysop not here	<pre>Sorry! The sysop appears to be tied up at the moment. Maybe try again later or use Feedback.</pre>	Displayed when chat request receives no response.
vstill not here	<pre>Sorry! The sysop still isn't answering! Please use feedback!</pre>	Displayed on subsequent unanswered chat requests.
vsayings#	<pre>Nature quotes from The Cabin ----- "I felt my lungs inflate with the onrush of scenery - air, mountains, trees, people. I thought, 'this is what it is to be happy.'" -Sylvia Plath [Time Left=299] [?=Menu] COMMAND? []</pre>	Random sayings displayed before first command prompt. Example: Vsayings1 through Vsayings6. Optional.
vgames menu	<pre> GAMES MENU ----- 1 *** Asylum 2 *** BTLC 3 *** DragonSlayer 4 *** Empire II 5 *** Nuke Em 6 *** Star Trek 7 *** Wrestling 8 *** War Games ** Beta Q *** Quit to Main Menu</pre>	Required if running games from AUX3. Disable spare command "1" if using Mod Menu instead.
vlogoff	<pre>Thanks for calling Westwood BBS! Check out these other snazzy setups: Commodore 4 Ever c4everbbs.ddnsgeek.com:20098 Arrakis atl.ddns.net:6403 Eaglewing dantheman.freedomdns.org:6400 Genetic Point g-point.tunk.org:500 Itchy Butt itchybutt.org:6502 Oasis oasisbbs.hopto.org:6400 The Cabin thecabinbbs.com:6400</pre>	Displayed prior to disconnect. Optional.

Color 64 BBS Manual Version 8.1A March 2026

File	Example	Description
vupload msg	Upload: Please be mindful on content being uploaded. Not all of us are old farts....	Displayed before upload begins.
vupload held	This upload will not be released to the public downloads until a SysOp reviews it. Thank you!	Displayed after upload when below auto-release level.
Other notable screens		
vdoshelp	<pre> Q: Status/Read Error ch single cmd S: Display Directory allows patrn H: Change Dev # #<num> N: FORMAT DISK N0:name,id C: Copy a file C0:new=old R: Rename a file R0:new=old F: Read a file F0:name P: Print a file P0:name Z: Regenerate Directory single cmd I: Message editor single cmd X: Two drive copier single cmd SYSOP: Sysop in/out single cmd </pre>	Displayed at DOS Wedge when "?" is entered.
vmsg menu	<pre> ** Message Menu Commands :L:ist :S:end :I:nsert :N:ew :A:bort :H:elp </pre>	Message menu help display.
vwfc	<pre> Idle : 11:09:58 Time: 10:55 pm Status: [] <1> Date: 08/30/2025 Init: ate0v1h0x1m0b1 Calls Today :1 Invalid Calls:0 New Users :0 Msgs Posted :0 Mets Holding:0 Mets Due Out :0 Uploads :0 Downloads :0 High Message:19 Low Message :2 Free Bytes :4550 Total Calls :98 Last Caller :pasiu BBS Name :Westwood BBS Color 64 v8.1a LTK W*E*S*T*W*O*D B*B*S Press ^Z TWICE to LOGON. Sysop: Out Email Notify: No </pre>	<p>Waiting For Call screen (SYSOP console).</p> <p>Uses heavy Variable MCI and ML-populated modem status field. Edit carefully.</p> <p>Also note the "BBS Name" just below "Last Caller" in this screenshot is only populated if you have Network 1.26a running.</p>
vmod edit menu	<pre> EpSuper Mod Menu Mod Stat Editor Commands L list modules U view module stats D disable module stats R re-enable module stats A add module stats E edit module stats X scratch module stats Z zero game plays S editor sub-menu </pre>	Help screen for Mod Menu.
vmod sub menu	<pre> EpSuper Mod Menu Editor Sub-Menu Commands R regenerate module index S set main parameters U view parms and etc. V tally game plays I create level menus C reset module file </pre>	Sub-menu help for Mod Menu.

Special Provisions for ASCII callers

Normally when a BBS uses a large amount of PETSCII graphics, non-graphic (ASCII) callers can have difficulty understanding what is displayed on their screen. Fortunately, Color 64 v8.1 and above handle most screen conversions automatically, making the system usable for both PETSCII and ASCII users.

That said, certain screens — particularly menus with heavy graphics — may not translate cleanly. For these situations, Color 64 provides an optional method for creating alternate system files specifically for ASCII callers.

Included with the system is a merge file called **afr.ovxx** (AFR = ASCII File Read). This merge can be applied to all overlays except for the Network overlays.

How the ASCII Alternate File System Works

- Create your standard system file as usual.
- If you want an ASCII-friendly version, create a second file:
 - Use the same filename.
 - Remove or simplify any troublesome PETSCII graphics.
 - Add a plus sign (+) to the end of the filename.

Example:

- √menu1 (standard PETSCII version)
- √menu1+ (ASCII alternate version)

When an ASCII caller accesses the system, the BBS will automatically display the “+” version of the file if it exists.

Important Limitations

System files that function as scripts rather than simple display files cannot use this method. Examples include:

- √application
- √sysop news

These files are executed as part of program flow rather than being read directly to the modem, and therefore cannot be converted using the AFR method.

The Help Files and Text Files

A complete set of Help Files is included with the Color 64 package, but no Text Files are provided by default. Help Files and Text Files function identically, so the included Help Files serve as a model when creating your own Text Files.

There must be a file on the Help Files disk called "@help files". Notice that this filename begins with an "@" symbol instead of the check mark (✓). The "@" prevents the file from appearing in download directories, while still allowing remote SYSOPs to download, scratch, or re-upload it as needed. In practice, any user can download these help files if you provide instructions. This is particularly useful for callers whose terminal software does not support ASCII capture.

```

Filename
>@help files

Opening 0:@help files, 8

      GENERAL HELP FILES

(1) General
(2) Reading Messages
(3) The Post Office
(4) Posting Messages
(5) Uploads/Downloads
(6) Miscellaneous
(7) Graphics Modes
(8) Using MCI Commands
(9) Using the Network
(10) BBS Profile Editor

(L)ist      (E)dit      (C)ont      (D)delete
(R)ew       (I)nsert    (S)end      (A)abort
(H)elp      (*)merge    (!)col:40   (M)ci:0n
>

```

Both Help Files and Text Files are optional features. However, it is strongly recommended that at least the Help Files be available, especially for users new to the BBS scene.

The "@help files" file acts as a menu. When displayed, it should instruct callers which number to enter to read a specific help file. The format of this menu is entirely up to you. It may be simple text or full graphics, and the numbering does not need to be sequential if you prefer a custom layout.

Each help file on disk must follow this naming convention:

@help1 @help2 @help3

```

0  "help1"      seq
6  "help2"      seq
2  "help3"      seq
4  "help4"      seq
8  "help5"      seq
32 "help6"      seq
16 "help7"      seq
16 "help8"      seq
20 "help9"      seq
8  "help10"     seq
4  "help files" seq
2

```

When a caller enters "1", the BBS will load and display "@help1".

Text Files operate in the same way, except the menu file must be named "@text files", and the individual files follow this format:

@text1 @text2 @text3

If the "@text files" menu file does not exist, callers selecting Text Files will see a message stating that none are available. The same behavior applies if "@help files" is missing.

There is no internal limit to the number of Help or Text files you may create beyond available disk space. If you anticipate exceeding the standard BBS message line limits, use the stand-alone message editor (+EDITOR), which allows messages up to approximately 500 lines.

Be creative when designing Text Files. They can help define the personality of your BBS. Examples might include current news, Color 64 BBS lists, recipes, movie reviews, technical notes, or hobby content relevant to your audience.

After editing your system messages, use a file copier to copy all sample Help Files to the drive designated in the SETUP Drive Assignments section. Then use the Message Editor to read and, if desired, modify each file. These Help Files serve as both documentation for your users and a guide to understanding the system's features.

Email Notification Option

Requirements: Home computer running Python, Wifi Modem or TCPSER

This is a fun feature I put in, but it requires the use of modern technology to get us over the hump. With the power of another computer on your home network and Python3, your BBS will generate emails of BBS events that you can define through the various overlays. I'm using a Kubuntu Linux home computer, where the script listens on a specified port to receive BBS notifications (calls) and then emails me.

Upon BBS startup, you will see at the bottom of the screen "Email Notify: No". This toggles between "Yes" or "No" by using the up arrow on your keyboard (↑). When set to "Yes", the BBS will call out to the networked computer after the user logs off. This takes an additional 8-10 seconds of the total time it takes for the system to reset back to the wait-for-caller screen.

NOTE: This should be done with a computer that is within the same network as your Commodore 64. Your ATDT command that must be set to dial your home computer should be an internal network IP; otherwise, your username and password will be sent unencrypted across the internet.

To setup:

1. Create your python script on your computer. You will find with this build the file "**no-tify.py**". This one is my script for Kubuntu/linux.
2. Next, find out the IP of your computer that will be running the script.
3. Edit the script with your desired username and password for the BBS to login as, your SMTP information, and your appropriate "FROM" and "REPLY TO" email addresses.
4. Script and the code in \bbs.init are set to port 8502. If you want to change the port, make sure you have modified both (see below for \bbs.init).
5. Run the Script
6. In your \bbs.init file, edit the following lines to add your IP address to call, username and password:

```
35705 "ATDT your-internal-ip-address-here:8502":gosub35725:"your-username":gosub35725
```

```
35706 "your-password":gosub35725:gosub1110:'da$+"
"+t$+""+cr$+em$+cr$+"end"+cr$:return
```

The script uses the variable em\$ to record events. The BBS records when a user logs on, enters messages or games, and successfully logs off. It also sends out a notification at 6am/pm and 12am/pm stating that it is still alive and well.

You can add events to this by placing em\$ in other areas (example: "**em\$=em\$+cr\$+"entered area x"**"), but keep in mind that the variable can only hold a total of 255 characters. It's also important to note that if you add something to em\$, it should be done before invoking the **.18** command (save variable state) as the consequent **.19** (restore variable state) command will wipe out whatever you had saved; particularly important in the games area. After the user logs off, the BBS will call your computer, transmit username/password, then the events recorded. The last transmission is "end" which tells the script it is finished. The script will then send the information to your SMTP server.

Since the transmission is going from Commodore to a modern-day PC, only lower case should be used as the python script will see this all in capitals (and capitalized characters from the Commodore will be rejected).

BBS Operation

With your SETUP and organization of your BBS complete, you are ready to start up your BBS operation!

Below are steps on how to load your BBS and understand the various operations of it.

Loading The BBS

At this stage you should already have:

- Created your parameters file (\bbs.parms)
- Created your password file
- Built most of your core system messages and Help files
- Placed the correct files on each drive as defined in SETUP

Each drive defined in your parameters must contain the proper disk or disk image with the required files. If you plan to begin with files already available in the download directory, place them on the appropriate drive now.

There are two separate startup paths:

1. [Loading without a RAM Expansion Unit \(REU\)](#)
2. [Loading with a REU](#)

Loading the System without a REU

If you are not using a RAM Expansion Unit, insert your Boot disk and load:

+bbs

Type RUN and press RETURN.

JiffyDOS users may auto-RUN with:

£+bbs

Lt. Kernal HD users may simply type:

+bbs

and press RETURN.

If you enabled "Swap Disks" in SETUP, you will be prompted to insert your Program disk after the initial boot stage. The system will then load \bbs.parms and all overlays from the Program drive.

If successful, you will see:

Working...

Proceed to [Entering the Date & Time](#).

If the system fails to boot, verify that all required files are present on the Program and System drives.

Loading the System with a REU

Confirm you answered "Y" to the REU option in bootmaker and enabled disk swapping if necessary.

Insert the Boot disk and load:

Color 64 BBS Manual Version 8.1A March 2026

+ram.start

RUN it.

JiffyDOS users may auto-RUN with:

↑+ram.start

The script will copy overlays and ML files into the REU. Scrolling disk commands are normal.

If you see a FILE NOT FOUND ERROR, reset immediately and confirm these files exist:

- √sys.loadml
- √sys.mlinit
- +ram.bbs
- +ram.reboot
- Required ML file

You may be prompted to insert your Program disk during the process.

After copying completes, the system will appear to reset. This is normal. You should briefly see the Color 64 title screen followed by:

Working...

Entering the Date & Time

Enter the date in MM/DD/YYYY format.

If the displayed date (from √variables) is correct, press RETURN.

If you change the date, you will be asked whether to reset time limits. On first startup, answer "N".

Next, set the time:

1. Enter hour in 24-hour format.
2. Enter minute (0–59).

Examples:

- Midnight = 0
- 2 PM = 14
- 8 PM = 20

The system calculates the day of week automatically.

Regenerating the Message Index

You will be asked whether to regenerate the message index.

The index speeds startup and is normally saved at shutdown. If the system was not shut down properly, the index will automatically rebuild.

Regenerate if:

- Messages were scratched offline
- Disk errors occurred
- Maximum message settings were changed

If improper shutdown is detected, the system will rebuild the index regardless of your answer.

After disk activity completes, the BBS will come online.

Rebooting the BBS

All boot prompts have defaults. Most of the time, pressing RETURN four times will restart the system.

To bypass prompts:

- Use +reboot (non-REU)
- Use +ram.restart (REU systems)

The Wait-For-Call (WFC) Screen

When initialization completes, the `\wfc` file is displayed.

You will see modem initialization in the upper left. If it ends with "OK", the modem initialized correctly.

When a connection occurs:

- The modem answers
- Carrier is detected
- Login sequence begins

Manual controls:

- Hold Commodore to force answer
- Press Commodore after answer to hang up

After five invalid sign-on attempts within 24 hours, the system pauses answering for approximately three minutes. Press Commodore to cancel the delay if needed.

The system is now ready for callers.

The SYSOP Menu

While the BBS is waiting for a caller, pressing any function key will display the SYSOP menu.

Important (New System Setup)

Before allowing other callers to log on, you **MUST** post at least one public message yourself.

There are two critical reasons for this:

1. It increments the “last message read” variable so new users can properly access their mailboxes.
2. When posting the very first message on a new system, you **MUST** answer **Y** to: “Start a new subject (Y/N)?”

After logging off with (O), the system saves your message pointers and statistics. For this reason, your first action after installation should be a local-mode login and posting a message.

The SYSOP menu consists of the following:

- **F1: Local Mode Logon**
 - Logs into the BBS locally.
 - You may choose Fast Logon as User #2 (direct to command prompt) or perform a normal login.
 - Graphics mode is automatically enabled and display scrolls at maximum speed.
- **F2: Term Mode**
 - Loads the TERMINAL overlay (\bbs.term) if present.
 - Provides Terminal mode, Graphics/Baud settings, Autodial, DOS Wedge, Upload/Download, Protocol selection, Buffer capture, and Return to BBS.
 - Press F1 while in terminal to return to the Terminal menu.
- **F3: Display Caller Log**
 - Displays or prints (device #4) the \caller log file (HOME pauses, SPACE aborts).
 - Shows caller names, time of call, and activity trail.
- **F4: Network Menu / Set Time & Date**
 - If Network is enabled, opens the Network menu.
 - Otherwise allows you to reset the date/time and reinitialize \level ? msg flags.
 - Use this after adding or removing any \level # msg files.
- **F5: List Feedback (Mailbox)**
 - Displays or prints your mailbox (HOME pauses. SPACE aborts).
 - After reading mail, you may reply and clear the mailbox.
 - Replies are sent from Member #2.
- **F6: Password Maintenance**
 - Displays, prints, and edits password records (HOME pauses. SPACE aborts).
 - You may edit name, password, level, upload/download counts, time remaining, etc.
 - To delete a member, change their access level to 0.

Password record display format:

Color 64 BBS Manual Version 8.1A March 2026

record#: membership name, password
 level, expiration date, membership info
 DL blocks, UL blocks, times called
 access group, last date called, block size
 last message read, time left, posts
 real name, phone number, birth date
 address line 1
 address line 2

Creating a new membership list:

After listing/editing passwords, you will be prompted to create a new membership list.
 On a new system, edit record #2 (SYSOP) to change your name and password.
 After saving, answer **Y** to create a new membership list.

○ F7: DOS Wedge

- Loads the DOS Wedge (from \bbs.xfer).
- Provides disk utilities.

Table 20 - 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
SYSOP	Sysop In/out flag toggle	Single Command

You can use Directory listings to count matching files.

Example: \$0:?private*

○ F8: System Shutdown

- Proper shutdown procedure.
- Saves message index, caller log, and system variables.
- Exits cleanly to BASIC.

Stopping without F8 or GOTO9991 can leave files open and ML vectors pointing to BBS routines.

If STOP was pressed accidentally, type: “CONT”

If necessary, type: “GOTO9991”

Password Record Information

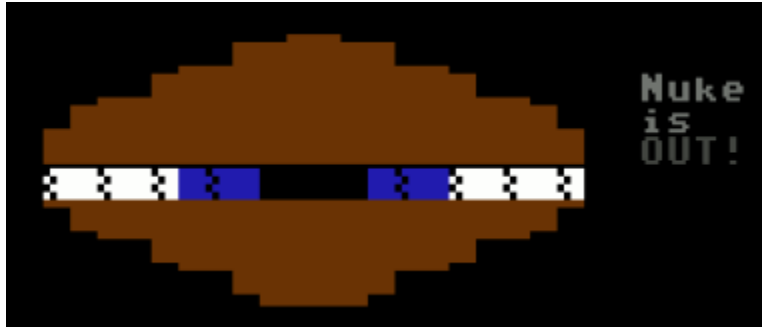
When using the F6 Password Maintenance function, you will notice that each member record contains a substantial amount of stored information. Below is a description of each field contained within a password record.

Table 21 - Password Record Field Descriptions

Field	Description
Access Level	Member's access level (1–9). Default behaviors for each level are defined in the "SETUP – Main Parameters" section.
Membership Expiration Date	If your BBS uses time-based memberships, this field determines when a user's access expires. When the date is reached, the user's level is automatically changed to the "Member expired level" defined in SETUP.
Membership Name (Handle)	The user's selected handle (up to 20 characters). Standard ASCII only — no Commodore graphics. Must contain at least one alphabetic character, as searches match alphabetic characters only.
Password	User-defined password (3–9 characters). Not case-sensitive. Stored internally as uppercase letters and numerals.
Membership Information (DD\$)	Reserved for SYSOP use. Not directly used by the system. This value is stored in the DD\$ variable while the caller is online and may be displayed using the £a6 MCI command (if enabled).
Blocks Uploaded	Total number of blocks uploaded by the caller. This is a raw count and does not directly represent download credit.
Blocks Downloaded	Total number of blocks downloaded. Combined with uploads and the "credits per upload block" setting to determine download credit. This value can result in negative credit.
Time Remaining	Minutes remaining for the caller today. Reset nightly according to the limits defined in SETUP.
Number of Posts	Total number of public messages posted by the caller.
Access Group	Defined for future use.
Last Date Called	Most recent login date. Used during midnight routines for expiration and purge checks.
Last Message Read	The last public message number the caller has read.
Real Name	User's real name (up to 30 characters).
Phone Number	User's phone number (up to 15 characters).
Birth Date	User's birth date
Email or street address	User's contact information. Modern systems typically use email rather than street address.
City/State/Zip	User's city, state, and zip code (up to 30 characters)..
Block Size	Indicates which protocol was used during the last file transfer. If 0 → XModem. If greater than 0 → Punter (value represents block size used).

Sysop In/Out Flag

Much like the Email notification flag at the Wait-for-Caller screen, there is a Sysop: In/Out status at the bottom of the screen. This is toggled by using the left arrow key on your keyboard (←). When set to "In", the BBS will call up the file `√sysopin` to display to the user your customized screen showing that you are in. Conversely, `√sysopout` will be called showing you are out when this flag is set.



Signing On

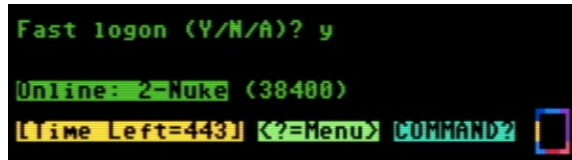
There are two primary ways to initiate a sign-on: a remote caller can connect using their modem, or you can perform a local sign-on by pressing F1 from the SYSOP menu.

Local Sign-On

This is usually the first action taken on a newly running BBS. The Local Sign-On allows the SYSOP to enter the system directly from the console. While you are logged in locally, the modem is taken off-hook, meaning remote callers will receive a busy signal until you log off.

When F1 is pressed, the system asks whether you want to perform a Fast Logon. Fast Logon bypasses all startup screens and places you immediately at the main command prompt as user #2 (the SYSOP account).

If you answer "N", the system performs a normal login. You must enter your user name and password, though the `√systemstart` and `√systemstart2` screens are skipped. The system automatically assumes graphics capability during local login, so the first screen displayed will be `√welcome1` followed by the User ID prompt.



```
Fast logon (Y/N/A)? y
Online: 2-Nuke (38400)
Time Left=443  <2=Menu>  COMMAND?
```

When your local session is complete, log off normally. The system will return to the Waiting for Call state.

Normal Sign-On

When a remote caller connects, the BBS answers the line and negotiates baud rate with the caller's modem. If the connection succeeds, the system displays `√systemstart` and `√systemstart2`, followed by the graphics mode check.

The Graphics Mode Check (and `√systemstart2`)

After the initial startup screens, the BBS performs a graphics capability test. The caller must press the DELETE or BACKSPACE key. You must instruct users to do this within your `√systemstart2` file.

Do not use graphics or color codes in `√systemstart` or `√systemstart2`, as the caller's display capability is not yet known.

The DELETE/BACKSPACE test determines terminal type:

- Commodore graphics terminals send ASCII 20.
- ASCII/ANSI terminals send ASCII 8 or 127.

If ASCII/ANSI is detected, the system asks whether the caller wants ANSI color and graphics. If plain ASCII is selected, the caller is also asked whether line feeds should be added to carriage returns.

After graphics mode is established, the file `√welcome1` is displayed. Test this file in Commodore, ASCII, and ANSI environments to ensure it appears correctly.

Color 64 BBS Manual Version 8.1A March 2026

Note: Earlier versions of Color 64 hardcoded the graphics prompt. In version 8.1a, you must explicitly include instructions in `\systemstart2`. This allows greater customization.

The User Number Prompt

The system prompts for the caller's user number. This corresponds to their record number in the password file.

A new user types "NEW" to apply for membership. An existing user enters their assigned user number. Pressing RETURN displays the membership list.

On a fresh installation, only one user exists: the SYSOP as record #2 with password "SYSOP". You should immediately edit this record and change your password.

Record #1 is reserved internally to store the total number of password records.

If an invalid user number is entered, the system reports the error. Otherwise, the caller is prompted for their password.

The Password Prompt

Passwords are 3–9 characters long and may contain uppercase letters, numbers, and certain non-graphics characters. As the caller types, asterisks are displayed instead of the actual characters.

After three incorrect attempts, the system disconnects and logs an invalid sign-on. After five invalid attempts within 24 hours, the BBS enters a 3-minute lockout period.

If login is successful, the following files display (if present):

- `\welcome2`
- `\logon stats (40-column)` or `\logon stats80 (80-column)`
- `\sysopin` or `\sysopout`
- `\level # msg` (matching the caller's access level)

Next, `\sysop news` is processed (see Installation → System Messages for formatting details).

The system then checks for private mail. If mail exists, headers are shown and the caller may choose to read it. After mail handling, the caller may be prompted to sign the guestbook before reaching the main BBS prompt.

New Users

If a caller enters "NEW" at the user number prompt, the system performs the following steps:

1. The system checks for the file "`\bbs closed`". If it exists, it is displayed and the caller is disconnected.
2. The caller is prompted for a handle (membership name).
 - Must contain at least one alphabetic character
 - Maximum 20 characters
 - No graphics or color codes permitted
3. The system checks the password file:
 - If the name already exists, the caller must choose another.
 - If the password file is full, the file "`\membership full`" is displayed and the caller is disconnected.

Color 64 BBS Manual Version 8.1A March 2026

4. If accepted, the file "\password msg" is displayed and the caller enters a password (3–9 characters). The system then displays the entered information for confirmation.

\new user msg1 is displayed before the application routine. The caller completes the application (see User Application section). After completion, \new user msg2 is displayed. The caller is then taken to the main BBS prompt.

The Main BBS Prompt

All callers eventually reach the main prompt. Commands are executed by pressing a single key corresponding to the defined BBS command. Access depends on the caller's level as configured in SETUP.

Pressing "?" displays the appropriate \menu# file (based on access level). You may use the included menu maker utility to generate level-based menus automatically.

The system guides callers through complex operations with prompts, and help files remain accessible at all times.

Graphics Mode

When a caller first logs on, the system determines the type of terminal being used through the Backspace/DEL test. Commodore graphics terminals send a CHR\$(20) code when the DEL key is pressed, while ANSI and ASCII terminals typically send CHR\$(8) or CHR\$(127).

If the caller is using ANSI or ASCII, the system will ask whether ANSI graphics are supported. If the caller answers "Y", full ANSI color and graphic translation will be enabled. If "N" is selected, the caller will then be asked whether their terminal requires line feeds appended to carriage returns. This accommodates ASCII terminals that automatically add line feeds.

Commodore Graphics Mode

Commodore graphics terminals experience the system as originally designed, with full PETSCII graphics and color control. This capability is the reason the system was named Color 64, as it was among the first BBS systems to fully support advanced Commodore terminal programs such as CCGMS.

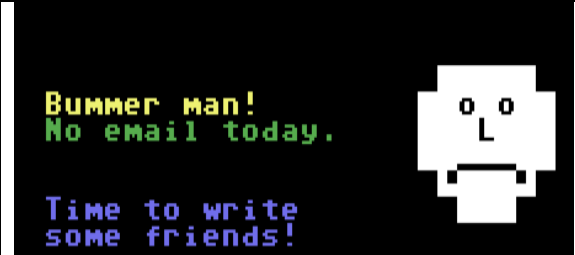
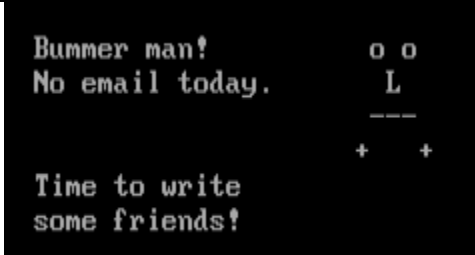
Background color changes are supported using the CTRL/B sequence. When a color key is pressed immediately after CTRL/B, the background color changes accordingly. On Commodore 128 80-column terminals, CTRL/B enables underline mode instead, so background changes do not function the same way. Color 64 properly passes through 80-column control codes such as underline and flashing attributes when appropriate.

The ASCII and ANSI translation routines are handled in part by machine language routines located beginning at memory area \$C044+ (via calls to \$FFD5).

ASCII Mode

When a caller connects using a plain ASCII terminal, Color 64 performs the best possible character translation. PETSCII graphic characters are converted to readable ASCII equivalents, and the system respects the current uppercase/lowercase state of the display.

Table 22 – ASCII Conversion Example

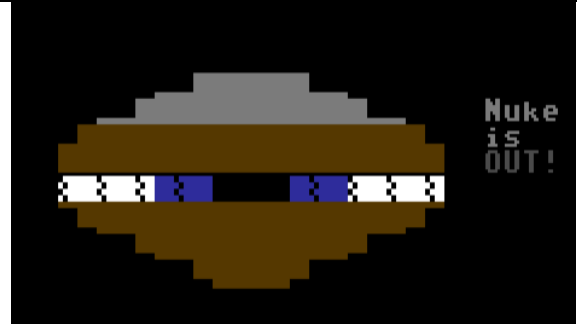
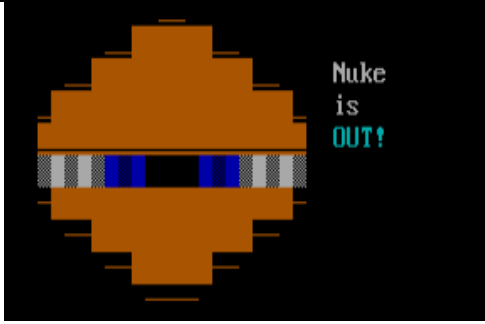
ASCII Conversion Example	
Commodore Graphics (PETSCII)	ASCII Translation by Color 64
 <p>Bummer man! No email today.</p> <p>Time to write some friends!</p>	 <p>Bummer man! o o No email today. L --- + +</p> <p>Time to write some friends!</p>

ANSI Graphics Mode

ANSI callers receive a high-quality conversion of Commodore graphics and color. Color 64 supports 15 ANSI foreground colors. Dark grey may not function properly on some ANSI terminals and is therefore translated to medium grey when necessary.

Color 64 BBS Manual Version 8.1A March 2026

Table 23 – ANSI Conversion Example

ANSI Conversion Example	
Commodore Graphics (PETSCII)	ANSI Translation by Color 64
	

ANSI users should configure their terminal with a default black background for best compatibility. Additionally, the terminal should not automatically append a carriage return after a line feed, since the line feed character is used by the system as a cursor-down control in ANSI mode.

Because many ANSI terminals support only eight background colors, bold color variants may not display correctly in reverse mode. As a result, certain graphic translations may appear slightly different than their Commodore equivalents.

ANSI callers may manually change the active cursor color by pressing the ESC key followed by two digits. The first digit controls bold mode (0 = normal, 1 = bright), and the second digit selects the ANSI color (0–7). This provides access to the full range of color combinations available to Commodore graphics users.

The User Application

The application routine gathers information about new callers and collects membership details such as real name, phone number, and related information. This portion of the BBS is fully customizable.

To enable the feature, the file named "\application" must be present on your System Files drive. The file must follow a specific format, so it is recommended that you begin with the included sample and modify it to suit your needs.

User Input Prompts

When the application routine begins, it prints text from the \application file until it encounters a line where the first character is a "#". When such a line is found, the routine pauses and waits for user input. After the caller enters a response, the routine continues printing text until another "#" line or the end of the file is reached.

Each response entered by the caller is stored as part of their application.

You may include short remarks for each prompt. These remarks are displayed when the caller reviews their completed application and are also included in your mailbox copy. This ensures that both you and the caller can clearly identify which answer corresponds to which question.

To include a remark, place it immediately after the "#" character on the prompt line. The remark should be brief but descriptive.

```

£p£1Please enter your full name (Ex: Fred
d 0. Ogle).
#(rn$)My Full Name Is
Enter your VOICE phone number (Ex: 410/2
85-0428).
#(ph$)Phone Number
Enter your birthdate (MM/DD/YYYY).
#(bd)Birth Date
Enter you street address only (Ex: 235
Pinewood Road).
#(a1$)My Address Is
Enter your city, state/province, and you
r zip-code (Ex: Baltimore MD 21222)
#(a2$)City, State/Province, Zip Code
What kind of computer will you be callin
g with most often?
#Computer I'll be calling with:

```

Anything following the "#" will precede the caller's response in both their review screen and your mailbox copy. For example, if the remark is:

1. My Full Name Is:

and the caller enters "John Smith", it will appear in your mailbox as:

My Full Name Is: John Smith

Keep remarks concise to maintain readability.

Getting Membership Information

The application routine can automatically capture certain fields that are stored directly in the caller's password record. Five special codes are recognized when placed in a prompt line:

- rn\$ – Real Name
- ph\$ – Phone Number

Color 64 BBS Manual Version 8.1A March 2026

- bd – Birth Date
- a1\$ – Address Line 1
- a2\$ – Address Line 2

These are BASIC variable names used internally by the system. When displayed to the caller and stored in your mailbox, the variable codes themselves are not shown. Only the readable text portion of the prompt appears.

The information entered by the caller is saved into their permanent password record. By default, callers cannot leave these special prompts blank. If you want to allow blank responses, delete line 18851 of `\bbs.init`.

You may include as many or as few of these special prompts as desired. This is the only automated method for collecting this specific membership data.

Once the Caller is Finished

When a new user completes the application, the results are sent to your mailbox and also written to a monthly archive file stored on your PRIVATE MESSAGES drive.

The archive file is automatically created each month using the format:

`"\questXX"`

Where "XX" represents the month number. For example, July applications are stored in `\quest07`.

This archive is maintained automatically and may be copied, backed up, or scratched at your discretion.

If you would like application results delivered to a different mailbox, examine the code beginning around line 18700 in `\bbs.init` and locate the references to `"\private 2"`. This appears in two locations. Change both occurrences to redirect application mail to a different user ID, such as a co-sysop responsible for validations.

Local Console Commands

The Sysop's keyboard is always active, even when a user is logged in and using the system. This allows you to enter a command for a new user, log a caller off, invoke chat, etc.

Here are some primary features:

Chat Mode (F1)

Pressing F1 when the user is at a command prompt will enable CHAT mode. 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. Other than the user terminating the call by disconnecting, they cannot terminate a chat session. Only the Sysop can, by pressing F1 again.

Quick Caller Editor (F5)

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 can 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.

The Caller Log System

The Caller Log is stored on disk as a sequential file named “\caller log”. Its behavior and size limits are controlled by the parameters you defined in SETUP.

Due to BASIC memory limitations, the Caller Log must always reside on drive 0. This restriction applies only to this file type and should not present any operational issues for your BBS.

As the BBS runs, the variable LG\$ is automatically updated with log information whenever a program performs a GOSUB to line 8003 (for i\$) or 8004 (for a\$). The LG\$ variable can hold up to 250 characters before it is written to a special memory buffer. This buffer can store two such entries before the data must be transferred to disk.

When the buffer becomes full, or when the BBS explicitly forces the operation, its contents are written to a temporary file named “\l” on the Caller Log drive. If multiple buffer dumps occur, additional information is appended to the same “\l” file.

When the BBS returns to the Wait-For-Call screen, the contents of “\l” are appended to “\caller log”. At this point, the message “Appending to Caller Log” appears on the screen. This same message will also appear during a proper shutdown or after a crash recovery. This mechanism ensures that caller activity is preserved and written safely to disk.

During the append process, a duplicate file named “\l.tmp” is created. This file is an exact copy of “\l” and is not scratched. It provides a snapshot of the most recent call. When you view the Caller Log, you will be asked “View Last Call?”. If you answer “Y”, the system displays the contents of “\l.tmp”. If you answer “N”, the full Caller Log is displayed instead.

After viewing the complete Caller Log, you will be prompted to scratch it. If you answer “Y”, the log will be erased and restarted. If you answer “N”, it will remain intact. The ability to scratch the Caller Log is access-level definable in SETUP. If you intend to maintain historical records, simply avoid scratching the file.

Caller Log maintenance is handled by lines 28500–28770 in “\bbs.init”. This routine automatically trims the Caller Log when it exceeds the maximum size defined in SETUP. The file is also governed by the “minimum UL space” setting. If either the maximum log size is exceeded or the minimum free space threshold is reached, the system trims the Caller Log by the number of blocks specified in SETUP.

By default, the maximum Caller Log size is 50 blocks and the default trim size is 8 blocks. The trim size does not mean that exactly 8 blocks are always removed. Instead, the routine reduces the file to the required size and then removes an additional 8 blocks to provide working space for future entries.

The entire process is automatic and designed to prevent disk-full conditions. If, in a worst-case scenario, trimming cannot free sufficient space, the system will scratch the Caller Log entirely to avoid a Disk Full Error.

MCI Commands

Color 64 includes support for MCI commands. MCI stands for Message Command Interpreter, a feature common to many BBS systems. An MCI command is a special character sequence embedded in a message to control formatting or trigger functions. For example, one command can pause output until a key is pressed, while another enables rainbow text effects.

MCI commands are interpreted only when the message is displayed, either in the editor or when read by a user. While typing, you will simply see their character representation. MCI commands may also be used with BASIC output routines (see section 3.11.1). They function anywhere ML-based output occurs, except within BASIC's PRINT statement, which does not use the ML output handler.

While in the message editor, every MCI command begins with the British Pound symbol (£). The character immediately following determines the command and must be an unshifted letter. Shifted letters will not be recognized. For example, typing £c tells the system to wait for a keypress. In Uppercase/Graphics mode, unshifted characters appear as uppercase, but they are still interpreted correctly.

Some commands require additional characters. Rainbow modes, for example, must be followed by color selections entered using CTRL or C= key combinations, just as when changing cursor colors. Most rainbow modes accept 2 to 8 colors. If fewer than 2 colors are entered, the current mode remains unchanged. If more than 8 are entered, only the first 8 are used. Caps/Punctuation mode is limited to 2 or 3 colors.

Standard MCI Commands

The table below lists the standard MCI commands:

Table 24 - Standard MCI Commands

Cmd	Mode Name	Function
£r	Character Rainbow	Color changes with each character
£w	Word Rainbow	Color changes with each word
£s	Sentence Rainbow	Color changes with each sentence that ends with a period (.), question mark (?), or exclamation point (!)
£l	Line Rainbow	Color changes with each line that ends with a carriage return
£g	Paragraph Rainbow	Color changes with each paragraph that ends with two carriage returns
£p	Caps/Punctuation	Uses 2 or 3 colors. With 3 colors: separate colors for uppercase, lowercase, and punctuation/graphics. With 2 colors: alphabetic vs punctuation. Produces a visually distinctive effect.
£n	Normal	Cancels current mode and returns to normal output
£v	Character Velocity	Follow with digit 0–9. 0 is fastest (no delay), 9 is slowest. Cancel with £v0 or RETURN.
£c	Wait	Pauses until a key is pressed
£t	Tab	Follow with position 0–79. If current position is less, spaces are added. If greater, cursor-left characters are printed. Tabs on the same line should move in one direction only. 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.

Two additional commands may be restricted by security level in SETUP: Message Output MCI and Variable MCI.

The Message Output MCI

£a outputs system or user-specific information. The digit following the command selects the output:

Table 25 - Message MCI Commands

Cmd	Function
£a0	Print text entered from last £i command
£a1	Current caller's name/alias (NA\$)
£a2	Caller's last calling date (LD\$)
£a3	Current time (T\$)
£a4	Current date (DA\$)
£a5	Last caller's name (LC\$). If last call was network exchange, remote BBS name is shown.
£a6	Membership information field (DD\$), SYSOP-defined
£a7	User's real name (RN\$)
£a8	User's birth date (BD\$)
£a9	User's phone number (PH\$)
£aa	Email or street address (A1\$)
£ab	City, state, and ZIP (A2\$)

Note: Because **DD\$** may contain sensitive membership information, **SETUP** allows disabling of **£a6** for security.

The Variable MCI Command.

£[allows output of any BASIC variable or expression. An expression may be numeric or string-based, such as:

5*4+3/16 a\$+mid\$(str\$(lv),2)

Type £[followed by the expression and terminate with a closing bracket]. The system evaluates the expression and prints the result.

BASIC keywords are stored internally as single-character tokens. Normally this tokenization happens automatically when entering program lines. However, within £[expressions, you must either manually enter tokens or use the automatic keyword cruncher.

Typing BASIC Tokens

The table below lists common numeric and string tokens and their key combinations:

Table 26 – Basic Token Reference Chart

Token	Hex	Key Combination	CHR\$()
+	AA	C= + J	CHR\$(170)
-	AB	C= + K	CHR\$(171)
*	AC	C= + L	CHR\$(172)
/	AD	C= + Z	CHR\$(173)
↑	AE	C= + X	CHR\$(174)
AND	AF	C= + C	CHR\$(175)
OR	B0	C= + V	CHR\$(176)
>	B1	C= + B	CHR\$(177)
=	B2	C= + N	CHR\$(178)
<	B3	C= + M	CHR\$(179)
SGN	B4	C= + H	CHR\$(180)
INT	B5	C= + J	CHR\$(181)
ABS	B6	C= + L	CHR\$(182)
USR	B7	C= + Y	CHR\$(183)
FRE	B8	C= + U	CHR\$(184)
POS	B9	C= + O	CHR\$(185)
SQR	BA	SHIFT-@	CHR\$(186)
RND	BB	C= + F	CHR\$(187)
LOG	BC	C= + C	CHR\$(188)
EXP	BD	C= + X	CHR\$(189)
COS	BE	C= + V	CHR\$(190)
SIN	BF	C= + B	CHR\$(191)
TAN	C0	SHIFT-*	CHR\$(192)
ATN	C1	SHIFT-A	CHR\$(193)
PEEK	C2	SHIFT-B	CHR\$(194)
LEN	C3	SHIFT-C	CHR\$(195)

Example: To output INT(RND(0)*16+5), convert keywords and mathematical operators to their token equivalents. The completed command appears visually as shown below:

£[((0) * 16 + 5)]

Color 64 BBS Manual Version 8.1A March 2026



Close the expression with] to execute it. Any syntax error within the expression will generate the same error BASIC normally produces. For this reason, access to £[should be restricted to trusted users.

Automatic Keyword Cruncher

To avoid manually entering tokens, place a greater-than symbol (>) immediately after the opening bracket:

£[>int(rnd(0)*16+5)]

This version is easier to type and read, though slightly slower in execution. Comparison:

Automatic Keyword Cruncher Example	
"int(rnd(0)*16+5)"	
With Automatic Keyword Cruncher	Without Automatic Keyword Cruncher
	
Key Sequence: £ [<C=J> (<C=F> (0) <C=D> 16 <C=N> 5)]	

MCI Command Security

Access levels for £a and £[are configurable in SETUP. If a caller lacks permission, the system prevents entry of the command. The message loader also filters unauthorized usage.

Internally, allowed commands are converted to special characters. For example, the "a" in £a is stored as F2. The "[" in £[is stored as F4. These function-key characters cannot be entered directly by online callers, preventing unauthorized command injection.

Using MCI Commands in BASIC

When programming in BASIC, you must enter £a and £[using their internal representations. Instead of typing "£a", enter "£" followed by F2. For Variable MCI, enter "£" followed by F4. These appear visually as £a and £[, but internally use protected characters.

Text Editing Features

The Text Editor for Color 64 (both stand-alone and internal BBS versions) have helpful shortcuts to make your life easy in your day-to-day work with the files and mail content. The table below lists the miscellaneous features available while editing or entering text:

Table 27 – Text Editor Miscellaneous Features

Function	Key	Details
Lowercase	CTRL-L	Places an internal code in the text forcing display to transition to lower-case
Uppercase	CTRL-U	Same as above, but forces uppercase instead
Delete	←	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	This can be used when entering a line of input, it will delete the word you are currently on, as well as any trailing spaces after it.
Line Delete	CTRL-X	This will delete all the text you have typed so far on a single line.
Line Reprint	CTRL-V	This will re-print everything you have typed so far after printing a carriage return. This is not really something you would use in local mode but is advantageous for remote users who are experiencing any line noise resulting in garbage characters on their screen. This will verify what the BBS captured in keystrokes.
Center	CTRL-C	Centers the text you have just typed (before hitting the RETURN key). Centering the text will strip out all inserts and recorded deletes. You will still need to add a carriage return after centering a line. Centering will not function if input does not begin at the far-left column of the screen.
Dynamic Delete	DEL	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	N/A	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	Editor Menu - !	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	/@	This is a helpful command that you can use while editing a mail message. It 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 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.

Color 64 BBS Manual Version 8.1A March 2026

There is a special carbon copy feature built into our private mailbox editor. After reading any incoming mail, at the (D)elele, (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).

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 (only printed for network messages).

Line 6: The CITY information line (only printed for network messages and only if city was supplied)

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 (you will still need to do a new line in the message editor for the next information line). 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.

40-Column Example:

(CTRL-Y) means to press CTRL and Y Key for a carriage return.

(RET) means to press the return key

```
1: ----- (CTRL-Y) (RET)
2: From User £[i$] (CTRL-Y) (RET)
3: on £[i$] (CTRL-Y) (RET)
4: Subj: £[i$] (CTRL-Y) (RET)
5: From Node: £[i$] (CTRL-Y) (RET)
6: in £[i$] (CTRL-Y) (RET)
7: ----- (CTRL-Y) (RET)
```

Save.....

80-Column Example:

```
1: ----- (CTRL-Y) (RET)
2: From User £[i$] on (RET)
3: £[i$] (CTRL-Y) (RET)
4: Subj: £[i$] (CTRL-Y) (RET)
5: From Node: £[i$] in (RET)
6: £[i$] (CTRL-Y) (RET)
7: ----- (CTRL-Y) (RET)
```

Save.....

Color 64 BBS Manual Version 8.1A March 2026

In the 80-column example above, lines 2-3 and 5-6 are merged when displayed by omitting the "(CTRL-Y)" key combination in lines 2 and 5.

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 table below).

Table 28 – Custom and Default Message Headers

Customized (40 Column)	Default
<pre>#362 - General From Patient Rick-T137 (#3) on August 17, 2025 - 9:38 am Topic: (R)New user</pre> <p>Header used for above:</p> <pre>Opening 0:\headers, 8 From Patient £[i\$]£t37 on £[i\$]£t37 Topic: £[i\$]£t37 Mode: £[i\$]£t37 City: £[i\$]£t37 </pre>	<pre>#9 - Westwood Front Desk From Nuke (#2) on August 11, 2025 - 9:35 am Subj: Network Messaging Help</pre>

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).

User Settings

Users have the option to modify their profile both for BBS performance as well as customization of their profile. There are two areas set up for Color 64 v8.10a:

- User Settings
- User Profile Editor

Edit User Stats – Modifying User 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 will be at a default of 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.

Edit User Profile – User Self-Service Profile Editor

A new module has been added for Color 64 8.10a, the BBS Profile Editor, which is loaded by `\bbs.profile`. It permits the user to edit some aspects of their user record, create an auto-signature for the message editor, and make a customizable screen that is shown during their login process and displayed to other users when sending them a private email.

The feature is called by the Spare Command "2" which must be set up for permissions in the +SETUP program. It is up to the SYSOP to decide who can be allowed to use this function.

See picture below as an example. This is a unique feature of the BBS; Users can have their own personalized greeting to other users when the user sends them an email! This started with me having a unique personalized banner for each user showing their username in a cool graphic format when they login and it just kind of grew from there. I figured users might like to make their own and

Color 64 BBS Manual Version 8.1A March 2026

be able to have it displayed when being emailed. Users can create their own or edit whatever is there.



Both the user signature and their profile banner are stored in the AUX1 designated area. Note that any files generated here in the user space (AUX1) will be deleted for a specific user if their account is deleted by the Sysop.

For their user record, the users are permitted to change their real name, email address, phone number, and city/state. This is stored back in the password file of the system.

The user signature that was added in the Message area is recalled when the user types "**=sig**" on a blank line in the editor.

Using Mod Menu 2.0

Color 64 v8.10a is defaulted to not use the Mod Menu functionality that came with versions 8.0 and 8.1; instead, it uses the AUX3 area for associated Games the Sysop chooses to use. However, you are not locked into this! You can still instead use the Mod Menu 2.0, particularly if you are moving your system from an existing 8.0/8.1 system where this was already in play.

Just remember that if you are using the games supplied on version 8.1a install disks, you will have to dig in the code and change your drive designation commands (like `H=14:gosub460`) to the appropriate area to support your use of Mod Menu.

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, 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.

Note there is 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 when adding new modules.

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.

Color 64 BBS Manual Version 8.1A March 2026

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 allotted 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 the table below:

Table 29 - 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 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. <u>Note:</u> 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.

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 'v' 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 the table below:

Table 30 - 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.

Color 64 BBS Manual Version 8.1A March 2026

Stat	Description
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.

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.

The table below defines the commands and their functions:

Table 31 - 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.

The table below lists the Editor's Sub Menu functions below:

Color 64 BBS Manual Version 8.1A March 2026

Table 32 – 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 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	<p>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:</p> <ul style="list-style-type: none"> • $ii < x$ - 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) <p>Following these comparisons 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).</p>
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.

Mod Menu Files

The table below 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 33 - Mod Menu Files and their function

Command	Definition
vmod 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.
vmod index	Module index used in the stat editor.
vmod parms	The main parms for the module menu.
vmod menu <#>	Menu file for access level <#>; example: <i>vmod index 1</i> for level 1 users
vmod index.bak	A temporary file created during the use of the editor.
vmod 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 Parm's
- Reset Module File

Also remember that if you use the ON OV GOSUB in a module, the very next line must be the one that re-loads vbbs.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:

"vmod 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.
"vmod edit menu"	This is a list of commands available in the mod stat editor and is displayed when '?' is entered from the editor prompt.
"vmod sub menu"	This is a list of commands available from the editor sub-menu.

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 "\vbbs.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.

Color 64 BBS Manual Version 8.1A March 2026

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).

Customization

The Mod Menu program is already installed and ready to use in "vbbs.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 "vmod " (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 it 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.

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:

Color 64 BBS Manual Version 8.1A March 2026

- 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\$+"vbbs.ov2*",dv

Use the line number that you wrote down in place of XXXXX. This line will GOSUB the game/module, then load vbbs.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\$+"vbbs.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.

Color 64 BBS Manual Version 8.1A March 2026

- 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.

Fast Garbage Collect Routine

Color 64 includes an enhanced machine language routine designed to speed up garbage collection. Garbage collection is the process by which BASIC reclaims memory occupied by string data that is no longer in use. On a standard Commodore 64 running BASIC alone, this process can introduce noticeable delays, sometimes long enough to interrupt the flow of a program.

The Color 64 fast garbage collection routine significantly reduces that delay. Instead of waiting while BASIC slowly reorganizes memory, the ML routine streamlines the cleanup process, resulting in faster overall system performance.

Several Color 64 BASIC routines rely heavily on this optimization, including:

- Regenerate Message Index
- Directory Regenerate

During these operations, you may notice the screen going blank. This is intentional. Disabling screen output allows the computer to complete memory operations more quickly. The temporary blank display is simply an indicator that the fast routine is actively working behind the scenes.

SYSOPs using a Commodore 128 gain an additional advantage. When running in 2 MHz mode, the garbage collection routine executes at double speed, further improving performance during intensive operations.

By default, the fast garbage collection routine is not always enabled. If you prefer to keep it active at all times for maximum system responsiveness, edit line 10241 of `Vbbs.init`. Change: `M7=.` to: `M7=1`

With the routine permanently enabled, you will observe more frequent screen blanking during memory-intensive tasks. This behavior is normal and confirms that the accelerated garbage collection is in use.

For additional technical details, refer to the descriptions of the !48 and !53 variables in [ML Variables](#).

Routine Maintenance

Several areas of maintenance will be required on your part to maintain smooth BBS operations and a pleasant environment for the end user.

These include:

Upload/Download directory upkeep

- Keeping the Upload drive managed as users upload files
- Regenerating directories as needed (Sysop has option to do this at DOS Wedge with the "%" command OR use "DIR Tools")

Password File Upkeep / Membership List

- Assigning New Users Access Levels
- Removal of Users, if Required
- Regeneration of Membership List
- May be accessed at Sysop Menu online with "<" command, Sysop Menu Screen (F6), or Sysop can use "PSWD Tools"

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 vbbs.init, the message would appear as **"syntax12000:vbbs.ini*"**. Note that the STOP key is disabled when "Rerun on errors" is enabled to prevent an accidental stopping of the BASIC program.

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 problem, locking up your computer.

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. Another point to note is that saving an overlay after the BBS is in a stopped condition (without typing the NEW statement) will cause a saved file to become a size equivalent of vbbs.init and will lead to LOAD OVERFLOW errors the next time you try to run the BBS.

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 vbbs.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.

Network 64 Instructions

Before enabling Network, read this entire chapter carefully.

Important Notice to New Color 64 BBS Owners

If you have just begun running Color 64 BBS, or have not yet started, do NOT enable Network at this time.

You should first become thoroughly familiar with the operation and structure of Color 64 BBS before introducing networking. Network configuration and troubleshooting require a solid understanding of the system's internal behavior. Attempting to use Network prematurely can result in confusion and difficult-to-diagnose problems.

Additionally, do not enable Network unless you already know which system(s) you intend to network with. At least one node must be entered through the NET SETUP program before the system can start when Network is enabled. Without a configured node, the system will not initialize properly.

8.1a Networking Updates & Improvements

Version 8.1a includes several updates and refinements to networking functionality:

- More user-friendly node editor within NET SETUP
- Telnet-friendly operation
- Removal of billing setup and display (Billing files and records still exist)
 - User balances are no longer checked when sending messages
 - Charge rates and user balances are not displayed

If billing functionality is required, a version with billing enabled can be provided separately.

Hardware Requirements

Network operation requires a fast method of loading overlays within the BBS environment. Slow disk access can cause timing-related failures.

The following hardware configurations are known to meet the required performance level:

- Commodore REU series
- Xetec Lt. Kernel hard drive system
- CMD hard drive system with either JiffyDOS or RAMLink
- VICE emulation configured to emulate one of the above

A potential candidate not yet fully tested is the Commodore Ultimate Elite II. With JiffyDOS implementation and higher clock speeds, it may meet the necessary performance requirements.

Fast loading is required not only for local performance, but also to maintain proper timing between networked nodes. The system contains precise timing routines that expect overlays and processes to complete within specific intervals. If those expectations are not met, the network may not function correctly.

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

Summary of Required Files

The table below provides a summary of the required and optional network-related files included with your Color 64 system.

Table 34 – Summary of Required Files for Network 64

File	Description
*** Required Files ***	
vbbs.nw1, vbbs.nw2	These are the two main network program modules and should be in the Program Files.
vsys.net	This is the NET SETUP program, similar to BBS SETUP. It allows you to define drives and configure individual node information.
prscrn52750	This file must be located on the Boot Files drive with Net Setup. It allows screen dumps to the printer while inside the program. (**)
vrlog, vslog	These are the receive and send logs for Network file transfers. (*)
bck to bill	Stand-alone program used to create a new billing file from a backup. It should be located with your Boot Files.
bbu.nw2	Optional merge file that installs an automatic billing backup feature into the midnight routine.
vnode app	Sequential file that you edit. It is used the first time you call a new node to introduce your system. The name of your BBS is automatically inserted at the beginning of this message when it applies to a new node. (*)
vconditions	Sequential file that you create. It is used when a node calls you for the first time and describes your validation requirements for new nodes. (*)
vtemp xref	Optional file used to assign nodes new ID numbers without requiring them to change their existing 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	
vnode x users	Membership list of users on your BBS who have access to Network. Created nightly (midnight) and provided to remote nodes as your system membership list.
vnode [#] users	[#] = Node Number. Membership list for an individual node. This file is received when you send data and request a new listing.
v+node [#]	Packet of public and private messages posted to the node specified by [#].
v+file [#]	File containing information about file transfers to be sent to the node specified by [#].
vnode ledger	Shows node account transactions. Remains on the drive until moved or scratched. Present but not used in this version.
vpublic storage	Holds messages you have released from received Node public messages.
vnode list	List provided when sending to a node. Created by the Net Setup program.
vntwrk.parms	Contains setup details for how Network is to run, including data for each node. Created by the Net Setup program.
vnode accounts	Relative file containing name, password, level, and last date called for each incoming node.
vnode billing	Relative file containing dollar amounts for each user on the system. Present but not used in this version.

Installation

The following steps prepare your system to run 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 the correct drive is selected for the Network Files.
- 5) Go to *BBS Commands* (option 7) and verify Network-specific command levels are set as desired:

- **Post Network Message:** Used by you or users to create network messages. Typically safe for most users except new users.
- **Net Maint Menu:** Network maintenance submenu for online/offline maintenance. Includes options related to node status and selecting files to transmit. This should be restricted to top staff.
- **Release Publics:** Allows public messages received from the network to be released. This is well-suited to a co-sysop role and is covered in detail later.
- **Restrict Posts:** Not a command. Defines a per-call limit on the number of network posts a user can make. The first value is a level exemption (users at or above this level are not limited). The second value is the maximum number of posts per call for users below the exemption. This should be accessible only by the Sysop.

Network Setup

Locate the file called "+net setup" in your Boot Files. Ensure the file "prscrn52750" is located on the same drive. The "+net setup" loader works like other boot files, except it loads the "vsys.net" (NET SETUP) program.

Once the program loads, you may be prompted to insert your Program disk.

You will then be asked a series of questions:

1) Number of Nodes

You should already have at least one system in mind. The minimum is "1", and you can add more later. You will need information about each node in a later step, so if you do not have the required information, abort setup and return when ready.

You may define 1 to 99 nodes, but higher values require additional memory.

2) Public Message Category

Network supports both private E-Mail and public messages. This setting determines how incoming public messages are handled.

You have two choices:

* Automatically place public messages directly into the message base by entering the destination category letter.

* Hold public messages in a special holding file for manual release by entering 0 (zero).

You may type "?" at this prompt to display a list of categories.

3) Open and Close Times

Defines the time window when network calls and message transfers are allowed. This feature originated to avoid higher phone-rate windows, but may still be useful.

Restrictions:

* The open time cannot be later than the close time.

* The open/close window cannot span midnight (for example, 23:00 to 03:00 is not allowed). You may open at 00:00, but you cannot remain open through midnight.

Times are entered in 24-hour (military) format. These values can be adjusted later.

4) Days Request Membership List

Each night at midnight, your BBS creates a network membership list (vnode x users). When you call other nodes, they may request a fresh copy of this list.

Color 64 BBS Manual Version 8.1A March 2026

This value determines how many days to wait before requesting a new membership list from a node you call. Lower values increase request frequency.

5) Your BBS Name

Enter your BBS name as it should appear in outgoing network message headers. Maximum length is 25 characters.

6) Does Modem Support BUSY and NO DIALTONE?

If your modem returns BUSY or NO DIALTONE response codes, you may choose to answer "Y". Check your modem documentation for the ATX command details.

This setting affects how the network lockout mechanism interprets failed call attempts.

If enabled, a node may be locked when the modem returns NO CARRIER (indicating repeated ringing with no answer). With BUSY/NO DIALTONE support, NO CARRIER is more likely to indicate the remote system is down, not picking up, or otherwise unavailable.

If you answer "N", the system will not lock nodes based on NO CARRIER/BUSY/NO DIALTONE responses.

If your modem does not support BUSY and NO DIALTONE, you must answer "N".

7) File Release Directory

Determines where files received through the network are placed. You may choose a U/D directory or the Network drive.

If a U/D directory is selected, the directory is automatically updated as files are received (similar to multi-upload).

8) Hold Files for Release?

Determines whether received files are held for manual release or made available immediately, similar to standard file transfers and the auto-release level setting in SETUP.

If you have already chosen to place received files on the Network drive, this option has no effect.

Net Setup - The Node Editor

After completing the main Network questions, you will be placed at the node editor prompt:

```
Select (1-1) [0] (?=List) or <CR>?
```

At this prompt, you can edit any node. The value in double brackets [] indicates the last node number you edited. Entering "?" displays a list of node names and their status, which helps locate open slots or specific entries.

Before editing nodes, keep these definitions in mind:

- **Node Number:** Assigned by you in +NET SETUP. Used for selecting a destination when users post a network message. The remote system does not see or use this number. (Outgoing calls)
- **Node ID Number:** Assigned by you (or automatically assigned during electronic application) and displayed in the NETWORK MAINTENANCE menu. This is effectively the remote node's "user id" when it calls your system. (Incoming calls)
-

Color 64 BBS Manual Version 8.1A March 2026

INCOMING and **OUTGOING** are separate. NET SETUP configures OUTGOING calls. The Node Editor in NETWORK MAINTENANCE configures INCOMING node access.

To edit the first node, enter 1 (Node Number One) and press RETURN. You will see a screen similar to the following:

```

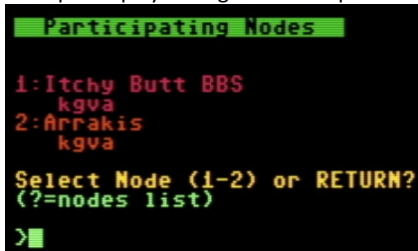
Node Editor
1. Node Name: Westwood
2. BBS ID Node assigned to you: 1
3. City/State: kgva
4. Telnet/Phone: westwoodbbs.org:64738
5. Password Assigned to you: hopper1
6. Baud Rate: 38400

S: Save Entry
A: Abort

Command:

```

Table 35 – Summary of Required Files for Network 64

Field	Description
1. Node Name	<p>Name of the remote node (maximum 25 characters). This name is displayed to users when selecting a destination for network posts. If you want to replace this node with another system, enter a new node name.</p> <p>Example display during a network post:</p> 
2. Node Membership ID Number	<p>At this prompt, you can edit any node. The value in double brackets [] indicates the last node number you edited. Entering "?" displays a list of node names and their status, which helps locate open slots or specific entries.</p> <p>Before editing nodes, keep these definitions in mind:</p> <ul style="list-style-type: none"> Node Number: Assigned by you in +NET SETUP. Used for selecting a destination when users post a network message. The remote system does not see or use this number. (Outgoing calls) Node ID Number: Assigned by you (or automatically assigned during electronic application) and displayed in the NETWORK MAINTENANCE menu. This is effectively the remote node's "user id" when it calls your system. (Incoming calls) INCOMING and OUTGOING are separate. NET SETUP configures OUTGOING calls. The Node Editor in NETWORK MAINTENANCE configures INCOMING node access. <p>To edit the first node, enter 1 (Node Number One) and press RETURN. You will see a screen similar to the following:</p>
3. City and State	City and State of the remote node. Use a slash to separate the city and state. Do not use commas.
4. Telnet/Phone	<p>Telnet address or phone number used to call the remote node. You may include special dialing prefixes if needed (for example, "t" to force tone dialing).</p> <p>Telnet format: domain:port Example: westwoodbbs.org:64738</p>

Color 64 BBS Manual Version 8.1A March 2026

Field	Description
5. Password	<p>Password for your account on the remote node.</p> <p>If you were assigned a password, enter it here. If you entered "1" for the membership ID to apply as a new node, enter the password you wish to request.</p> <p>Requirements:</p> <ul style="list-style-type: none"> • 3–9 characters • Avoid special characters • Lowercase will be converted to uppercase <p>The remote Sysop may accept your requested password or assign a different one.</p>
6. Baud Rate	<p>Baud rate for the remote node. If unknown, enter the highest rate supported by your system. The modem will detect the remote rate and automatically adjust.</p>
Removed questions from 8.0/8.1	<p>Billing related questions and functions were removed for 8.10a. Some files are still generated for it, but it is unused.</p>

After saving your entry, one of two things may occur:

- If you entered a Node Membership ID of 1, the system copies your Vnode app file into v+node x (where x is the node number). Your BBS name is automatically inserted at the beginning because the first line of the node application is used as your system name on the remote node.
 - To override this and present a different name, place the desired name at the beginning of Vnode app preceded by two CTRL/N characters.
- If you replaced an existing node name with a different one, you may be asked to confirm that you are replacing one node with another. If confirmed, the old node's files are scratched.

When all nodes have been entered, press RETURN at the select node prompt. The required files will be created as needed:

- vntwrk.parms
- vnode list
- vnode accounts
- vnode billing

Note that the last two files are created only if they do not already exist.

You may alter node information later, but use care. If you change drive assignments, remember to move existing network files from the old drive to the new one. You must shut down the BBS to make changes, so it is recommended to plan changes and apply them together.

One important caution: Do not edit vntwrk.parms or vnode.list with a message editor. These files are not designed to be modified with standard text editing tools and doing so can cause network instability and system errors.

Booting the BBS with Network

Now that you have finished setup, it is time to boot the BBS. If you are running your system using a REU, the "Vsys.remove" program will automatically copy your Network overlays from the Program Files.

Regardless of how you boot the system, ensure the two main modules (vbbs.nw1 & vbbs.nw2) are located on the Program Files drive. If you use RAMDOS, verify that these files are copied into the REU when it is loaded. Forgetting to include these two modules in a RAMDOS system is the most common cause of boot failures. The BBS will not complete the boot process without them present in RAM.

REU users should also confirm that sufficient memory space is available before loading. Insufficient REU space is another common issue.

The Wait-For-Call Screen

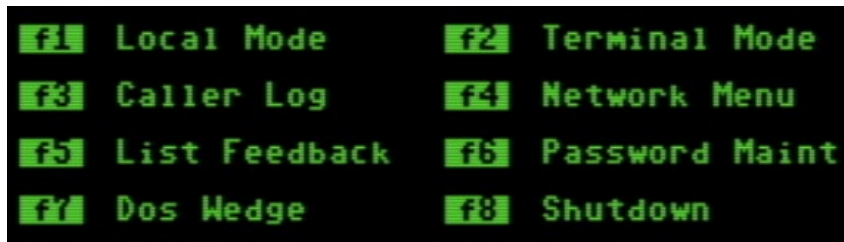
Boot the BBS. When you reach the call waiting screen, note the following items:

- Two status line values are now displayed: **Nets Holding** and **Nets Due Out**.

Nets Holding:0 Nets Due Out :0

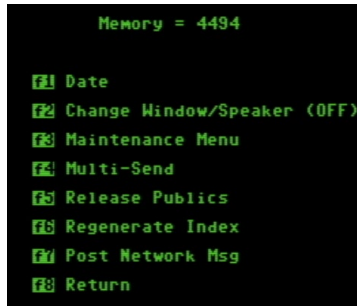
- Nets Holding** shows the number of public network messages waiting to be released. If you are not using the holding feature described in section 5, this value will always remain 0.
- Nets Due Out** shows the number of nodes scheduled to be called. It does not indicate the number of messages pending, only the number of nodes due for contact. If a node is locked, it will continue to appear as due until a call attempt is made, at which point the status will adjust accordingly.

Press any function key (F1–F8) and review the menu.



You will notice that the CHANGE TIME/DATE function previously assigned to F4 is no longer shown. It has been replaced with **NETWORK MENU**. Press F4 to enter the Network Menu.

The Network Menu



At the top of the Network Menu screen, a free memory check displays the number of bytes currently available. The exact value will vary depending on your system configuration. If available memory approaches 3000 bytes, your BBS is heavily loaded and may be at risk of an out-of-memory condition. Systems with numerous modifications, games, or a large number of configured nodes consume additional memory. If memory becomes too low, reduce the number of nodes through the NET SETUP program.

When selecting any of the eight menu options, you may notice a brief screen flash. This simply confirms that your keypress was received. Some options may require a short delay to access.

Table 36 – Network Menu Command Descriptions

Menu Command	Command Description
F1: Change Date/Time	Performs the same function previously available on the Sysop Menu.
F2: Change Window	Temporarily overrides the Network open and close window times defined in NET SETUP. The override remains active until you change it again, restart the BBS, or the system resets.
F3: Maintenance	Opens the Network Maintenance submenu. This section is used frequently and is covered in detail later in the documentation.
F4: Multi-Send	<p>Allows the Sysop (offline only) to send the same message or file to multiple nodes without recreating it for each destination.</p> <p>To use this feature:</p> <ul style="list-style-type: none"> First create a NET DEFAULT MESSAGE in the POST NETWORK MSG section, or a DEFAULT FILE in the ATTACH A FILE section. These defaults are separate from normal BBS default messages. Enter the node numbers separated by commas (similar to multi-download), or enter "all" to send to every configured node. <p>After confirming your selection, individual copies are created automatically. When finished, you may choose to retain the default message or file for future use (until replaced). If the mail verification modification is installed, it is not available when using Multi-Send. Refer to the Mail Verification section in "Miscellaneous Options/Features" for details.</p>
F5: Release Public	Used to manage held public network messages. You may re-read, hold, delete, or release messages into your message base. All pointers are updated automatically. This option is also available online.
F6: Regenerate Network Index	<p>Rebuilds the network index. Use this if the Nets Holding or Nets Due Out values appear incorrect.</p> <p>This option is also useful after manually deleting node messages or files at the DOS level (node messages begin with V+node, node files begin with V+file).</p> <p>The network index is automatically regenerated at BBS startup and at midnight.</p>
F7: Post Network Msg	Used to create and send a message to a remote node. This option is also available online. When first accessed, free space is checked on the HOLDING drive. The reserved free space is the value defined in BBS SETUP under the upload free space allowance.

Color 64 BBS Manual Version 8.1A March 2026

Menu Command	Command Description
	<p>You will then be prompted to select a node:</p> <ul style="list-style-type: none">• Enter "?" to display a list of participating nodes.• Enter the node number and confirm the displayed BBS name by answering "Y".• Next, choose whether the message is public or private. <p>Public messages:</p> <ul style="list-style-type: none">• Enter a subject.• Enter the standard message editor. <p>Private messages:</p> <ul style="list-style-type: none">• Enter "?" to view the node's membership list (available after the first exchange).• Enter the user's name or user number. Entering a number is faster.• Enter "SYSOP" to send directly to user #2 on the remote system. <p>The message editor functions exactly like standard BBS message entry. File merging using the C= key is supported.</p> <p>If E-mail verification is installed, you will be asked whether you want confirmation when the message is transmitted. If enabled, a notice will be placed in your mailbox showing the date and time the message was sent.</p>
F8: Return to BBS	Returns to the Waiting-for-Call screen


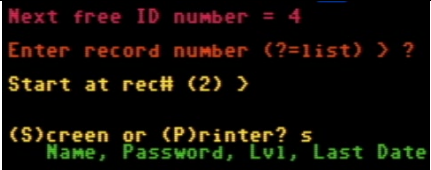
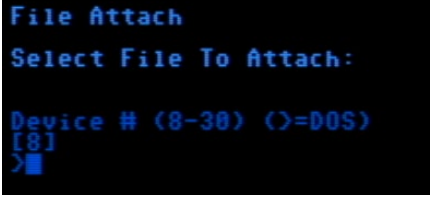
Network Maintenance

This is your Network Maintenance menu and below that are the options and their descriptions.

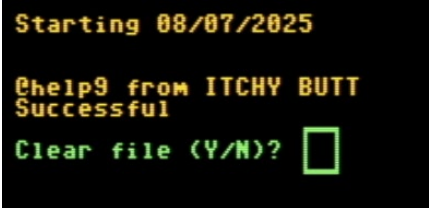
```

Network Maintenance
1) Node Status Report
2) Node Account File Edit
3) Attach File
4) Read Send Log
5) Read Receive Log
Select (1-5) or RETURN?
  
```

Table 37 – Network Maintenance Functions & Descriptions

Menu Command	Screenshot	Command Description
1: Node Status Report		<p>Displays the current status of all nodes and allows you to LOCK or UNLOCK individual nodes.</p> <p>If a node is locked:</p> <ul style="list-style-type: none"> No outgoing messages will be sent to that node. You will not be allowed to post new messages to it. <p>Upon entering this option, all nodes and their statistics are displayed. Press RETURN to go back to the menu. Enter a node number to toggle the lock status of one or more nodes.</p> <p>Nodes may also become locked automatically under the following conditions:</p> <ul style="list-style-type: none"> ACCESS – The system attempted to call the node but was denied access. CARRIER – Enabled through Net Setup special response handling. This occurs when a call attempt results in a NO CARRIER response (no answer or modem did not connect).
2: Node Account File Edit		<p>Used to edit incoming node account records. This section manages the credentials and access information for remote systems calling your BBS.</p> <p>Refer to section, Incoming Node Accounts, for detailed information about the fields and procedures used in this editor.</p>
3: Attach a File to Send		<p>Allows you to send a PRG or SEQ file to another node. The receiving node must be running Network 1.26 or 1.26a. If a remote system is running Network 1.24, the transfer will be aborted.</p> <p>Steps:</p> <ul style="list-style-type: none"> Enter the device number where the file resides. You may enter the ">" character instead to access a Mini-DOS. The Mini-DOS allows drive switching, directory viewing, and reading SEQ files. Press RETURN to exit Mini-DOS. Enter the drive number, drive command, and file name exactly as it appears on disk.

Color 64 BBS Manual Version 8.1A March 2026

Menu Command	Screenshot	Command Description
		<ul style="list-style-type: none"> • Enter the destination node number, or enter "?" to list nodes. You may also enter "x" to use the default file send feature. You will be asked whether the file should be deleted after transmission. • Next, enter the filename to be used on the receiving node. Press RETURN to use the same name as on your disk. If a file description exists, you will be prompted to include it. <p>A file named "V+file x" (where x is the node number) is created on your Network drive. This file stores the transmission information.</p> <p>Transfer guidelines:</p> <ul style="list-style-type: none"> • Do not exceed 20 files per call. • Network v1.26 nodes enforce an 18-minute transfer limit. • At 2400 baud, approximately 450 blocks can be transferred in 18 minutes. • At 1200 baud, limit file size to approximately 250 blocks. <p>If both systems are running v1.26a (Color 64 v8 includes v1.26a), there is no enforced file size limit. However, practical limits of 1000 blocks or less are recommended.</p> <p>You are not required to send a message with a file, but you may do so. Messages and files are transmitted during the same call.</p>
4/5: Read Receive and Send Logs		<p>Displays the contents of the "Vslog" (send log) and "Vrlog" (receive log) files. These logs provide detailed information about network transfers.</p> <p>After viewing a log, you will be prompted to clear it. It is recommended to clear these logs periodically to prevent them from growing excessively.</p>

Incoming Node Accounts

When a remote node calls your BBS, it logs in using an ID number and password, just like a normal user. If the password does not match the stored password in your node account file, access will be denied.

New Node Requests

When a remote system connects as a new node, your network assigns it a new ID number. The remote system sends its requested password during this process.

After the initial call:

- The remote system stores the ID number assigned by your network in its VNTWRK.PARMS file.
- Your system stores the remote system's chosen password in your Vnode accounts file.

At this stage, the new node exists in your records but is:

- Locked / UNVALIDATED
- Assigned an access level of 0

Future calls from that node will be denied until you manually validate it.

To validate a new node:

- 1) Using Menu Option 1 (Node Status Report), unlock the node by selecting it and pressing "U".
- 2) Using Menu Option 2 (Node Account File Edit), select the corresponding record and assign it a Level of 1, then save.
 - a) The node is now Validated but Unreplyable. This means the remote system can send messages and files to you, but your system cannot send replies.
- 3) To make the node Replyable (allowing your BBS to send messages back):
 - a) Ensure the remote BBS exists in your VNTWRK.PARMS file (entered through +NET SETUP).
 - b) If not present, add it using +NET SETUP and assign it a Node ID of 1 so your system can request access from the remote Sysop.
 - c) Return to the Node Account File Edit and assign the node an access level equal to 1 plus the node number as it appears in your outgoing node list.
 1. The system will display "Link set with <node name>" (verify carefully). The node will now be replyable.

Refer to the "Node Access Levels" section below for detailed explanation.

Editing Node Accounts

Node accounts can only be edited through the Network Maintenance menu. All accounts are stored in a relative file named Vnode accounts.

Each record contains four fields:

- Field #1 – NODE NAME
 - Limited to 25 characters. This name is for internal reference only and is not visible to remote systems.
- Field #2 – PASSWORD

Color 64 BBS Manual Version 8.1A March 2026

- Limited to 9 characters.
- Field #3 – LEVEL
 - Accepts values from 0 to 101. Access level behavior is explained in the next section.
- Field #4 – LAST CALL DATE
 - Displays the last date the node called your system. This is informational and may help determine whether to retain or remove inactive nodes.

To delete a node account, type "delete" at the NAME input prompt.

Node Access Levels

Access levels operate in three functional modes.

Level 0 – Unvalidated

- Incoming calls are denied.
- Newly created node accounts default to level 0.

Level 1 – Validated, Unreplyable

- The node may send messages and files to your system.
- Your system cannot send replies to that node.

Replyable Nodes – Linked Access To allow replies, the access level must correspond to the outgoing node number.

Example:

Suppose your outgoing node list (from "?" in Post Network Msg) shows:

```
1: THE ABC BBS
   Brooklyn/NY
2: THE DEF BBS
   Chicago/IL
3: THE GHI BBS
   Los Angeles/CA
```

In this example, THE DEF BBS is node number 2.

To make an incoming account replyable to THE DEF BBS, assign it an access level of:

1 + node number

So for node number 2:

Access level = 3

General Rule:

TO MAKE A NODE REPLYABLE, SET THE ACCESS LEVEL TO 1 PLUS THE NODE NUMBER AS IT APPEARS IN YOUR OUTGOING NODE LIST.

Examples:

Color 64 BBS Manual Version 8.1A March 2026

- Node #24 → Access Level 25
- If you do NOT want replies enabled → Access Level 1

If an incorrect level is entered, replies will be routed to the wrong node.

Note: Network 1.26 simplifies this process. At the access level prompt, enter “?” to display your outgoing node list. Enter the node number from the list and the system will automatically link the incoming node and assign the correct access level. After saving, the BBS confirms which outgoing node is linked.

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 vbbs.nw1 and vbbs.nw2, respectively.

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 for a user, 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.

Troubleshooting

This troubleshooting section is not exhaustive. The table below addresses the most common problems encountered by new Network Sysops.

Table 38 - Network Troubleshooting FAQ

Question	Answer
After booting and answering the "regenerate message index" prompt, I receive a FILE NOT FOUND error and the BBS attempts to reboot.	<p>This usually indicates that a required Network module is missing. Verify that the two primary modules, vbbs.nw1 and vbbs.nw2, are located on your PROGRAMS drive. If your PROGRAMS drive is loaded into RAM:</p> <ul style="list-style-type: none"> • Confirm that both modules were successfully copied into RAM. • For RAMDOS users, the file "vsys.remove" is responsible for loading these modules. Refer to the "Bootting the BBS with Network" section for details. • Ensure sufficient free RAM space is available. A 1764 REU (256K RAM) may be limited in available memory. <p>Also confirm that the filenames are correct. The first eight characters must be exactly vbbs.nw1 and vbbs.nw2. If renamed or truncated incorrectly, the BBS will not locate them.</p>
When replying to a public or private network message, the reply attempts to post locally instead of going through the network.	<p>This is typically caused by incorrect node access level configuration.</p> <p>Review the "Node Access Levels" section. Incoming nodes must be configured as replyable in order for replies to route back through the network. If the access level is set incorrectly, replies will default to local posting.</p>
During BBS startup, the system crashes with "?file data error".	<p>This usually indicates corruption of either the Vntwrk.parms file or the Vnode list file. Review the Net Setup section to verify configuration. If the file is corrupted and no backup exists, recovery may not be possible. However, you may attempt to extract usable information by:</p> <ul style="list-style-type: none"> • Using the BBS DOS f: command to read the file. • Recording any recognizable data. • Scratching the damaged file. • Rebuilding the configuration using Net Setup. <p>The Vntwrk.parms file format (no blank lines) is structured as follows:</p> <p>Number of Nodes Category Assignments Open Hour Close Hour BBS Name Days Between Membership List Requests Node 1 – Node Membership ID Number Node 1 – Node Name Node 1 – Baud Rate Node 1 – Phone or Telnet Address Node 1 – Assigned Password Node 2 – (same sequence repeats per node)</p>

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.

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 provides an enhanced LOAD command that extends the functionality of the standard BASIC LOAD. The syntax is:

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

Parameters:

<filename> – The name of the file to load. In Color 64 it is common to use `dr$+"filename"`.

<device> – The device number.

<load address> – Optional. Specifies the memory address where the file will be loaded.

Load behavior depends on whether a load address is supplied:

- If no load address is specified, the system assumes a BASIC program. The file loads and automatically runs.
- If a load address less than 256 is specified, the file loads into memory at the address defined internally within the file (typical for ML files or memory tables). The BASIC program in memory will not auto-run.
- If a load address greater than 255 is specified, the file loads directly into memory at the address you provide. This also does not auto-run BASIC.

In summary:

- No load address → Loads and runs BASIC programs.
- Load address specified → Loads directly into memory without auto-running BASIC (unless loading directly into BASIC program or variable memory areas).

Examples:

↑dr\$+"vbbs.ini*",8	Loads and runs "vbbs.ini*" from device 8, drive dr\$
↑dr\$+"vbbs.ans*",8,0	Loads into memory "vbbs.ans*" from device 8, drive dr\$ into the location specified in the file
↑dr\$+"vsome file",8,58000	Loads the program file "vsome file" into memory from device 8, drive dr\$ into memory at address 58000

Overflow Protection

The enhanced LOAD command includes protection against BASIC memory overflow. If a BASIC overlay being loaded exceeds the allocated program space and overwrites variable memory, the BBS automatically shuts down and displays a LOAD overflow error.

When this occurs:

- All variables are lost.
- The message index must be regenerated.

This situation typically happens when an overlay file is larger than `vbbs.init`. Allowing it to load would overwrite variable memory and destabilize the system.

If you encounter a LOAD overflow error, reduce the size of the overlay so that it is smaller than `vbbs.init` before attempting to load it again.

New Output Commands

The Color 64 output commands differ from the standard BASIC PRINT statement. These commands send output both to the local screen and to the remote user. A regular PRINT statement only displays text on the Sysop's screen when a caller is online.

Additionally, the Color 64 output commands automatically convert output to ASCII or ANSI when the remote user is operating in those text modes.

Table 39 – Text Output Commands

Cmd	Details
#	<p>Format: #[<expression list>]</p> <p>Example: </p> <p>Output: </p> <p><expression list>: <expression list> may be a single expression or multiple expressions separated by semicolons (similar to PRINT).</p> <ul style="list-style-type: none"> If nothing follows the "#" command, the contents of A\$ will be printed. This shortcut is commonly used when A\$ contains data read from disk or keyboard input and must also be recorded in the caller log. The "#" command automatically appends a carriage return unless a trailing semicolon is included.
\$	<p>Same usage rules as "#", except no carriage return is appended. This makes it suitable for prompts.</p> <p>Example: </p>
% and &	<p>"%" and "&" function like "#" and "\$" respectively, but automatically adjust alphabetic characters (A–Z) to remain readable if Uppercase/Graphics mode is enabled.</p> <p>If the system is in Uppercase/Graphics mode and shifted letters are present, these commands automatically convert them back to readable letters instead of Commodore graphic symbols.</p> <p>These commands are commonly used in text editor routines where uppercase and lowercase modes may be switched dynamically.</p>
' and (<p>These commands operate similarly to "#" and "\$", but output is sent only to the modem and not displayed on the Sysop's local screen. They are primarily used when sending AT modem commands or when transmitting data that must not be processed by the normal output formatting routines. The CTRL/Y character does not function with these commands.</p> <p>The apostrophe (') command:</p> <ul style="list-style-type: none"> Uses the same format as "#". Sends output directly to the modem. Performs no ASCII/ANSI conversion. Appends a carriage return unless a trailing semicolon is included. <p>The open parenthesis "(" command:</p> <ul style="list-style-type: none"> Functions like the apostrophe command. Never appends a carriage return.

Important Note on Logging

When output must also be recorded in the caller log, you must use the format:

A\$=<text>:#

If you do not assign the text to A\$ first, incorrect information may be written to the caller log because these output commands do not modify A\$.

This caution applies to any situation where printed data is later reused elsewhere in the program.

Control-Y Command in Quotes

A special control character, CTRL/Y, has been added to the output routines. It functions like a carriage return but may be embedded directly inside quoted text.

In BASIC, CTRL/Y can be typed within quotation marks. On the BBS, it may be entered without terminating the current input line.

Key characteristics:

- Behaves like a carriage return.
- Can be recorded within text.
- Does not cancel the velocity MCI command (£v), allowing long formatted output without interruption.

Restrictions:

- CTRL/Y cannot replace CR\$ when writing structured data to disk (such as message headers or system parameter files).
- It should only be used when output is intended strictly for screen display and not later read by BASIC's INPUT routine.
- BASIC's INPUT command does not recognize CTRL/Y as a valid end-of-line character.

An exception exists when CTRL/Y is intended to be stored as part of the data itself. In that case, it must still be followed by a standard carriage return.

See the customizable message headers section for an example of proper usage.

Enhanced If/Then Statement

The Color 64 ML introduces enhanced decision-making commands that expand upon the standard BASIC IF/THEN structure.

Table 40 – Enhanced IF/THEN Statements

Cmd	Details
IF/THEN/ELSE	<p>Format: IF <expression> THEN <statements> : £ <statements></p> <p>The British Pound character "£" functions as ELSE in Color 64 BASIC.</p> <p>Example:</p> <pre>IF A=5 THEN PRINT "Hello" : £ PRINT "Goodbye"</pre> <p>Result: If A equals 5, "Hello" is printed. Otherwise, "Goodbye" is printed.</p>
AND-ELSE	<p>Represented by the characters "¡£".</p> <p>This behaves similarly to ELSE, but with one important difference:</p> <ul style="list-style-type: none"> • If the expression is TRUE, both the THEN section and the ¡£ section execute. • If the expression is FALSE, only the ¡£ section executes. <p>Example:</p> <pre>IF A=5 THEN PRINT "Hello" : ¡£ PRINT "Goodbye"</pre> <p>Result: If A equals 5, both "Hello" and "Goodbye" are printed. If A does not equal 5, only "Goodbye" is printed.</p> <p>Whatever follows "¡£" will always execute.</p>
REDECIDE	<p>Format: !+ <statement> or !- <statement></p> <p>These commands reference the result of the most recent IF/THEN evaluation.</p> <ul style="list-style-type: none"> • !+ executes only if the previous IF/THEN evaluated TRUE. • !- executes only if the previous IF/THEN evaluated FALSE. <p>Example:</p> <pre>10 IF A=5 THEN PRINT "Hello, "; 20 !+ PRINT "How Are You?" 30 !- PRINT "What is Your Name?"</pre> <p>Result: If A equals 5: "Hello, How Are You?" is printed.</p> <p>If A does not equal 5: Only "What is Your Name?" is printed.</p> <p>ELSE may also be combined with REDECIDE statements. In this case, the £ portion executes when the opposite condition of the REDECIDE is true.</p> <p>Example:</p> <pre>10 IF A=5 THEN PRINT "Joe" : £ PRINT "Jim" 20 !+ PRINT "Jenna" : £ PRINT "Jessi" 30 !- PRINT "Jerry" : £ PRINT "John"</pre>

Color 64 BBS Manual Version 8.1A March 2026

Cmd	Details
	<pre>40 !+ PRINT "Jack" : £ PRINT "Frank"</pre> <p>Result: If A equals 5: Joe, Jenna, John, and Jack are printed.</p> <p>If A does not equal 5: Jim, Jessi, Jerry, and Frank are printed.</p> <p><u>Important:</u> Executing a REDECIDE statement does not change the TRUE or FALSE status of the most recent IF/THEN. Modifying variables used in the original IF/THEN condition also does not retroactively alter its result.</p>

Multiple IF/THEN/ELSE Statements on One Line

You may place more than one IF/THEN/ELSE statement on a single line (or any combination of the enhanced decision commands described above). However, it is important to understand how the interpreter evaluates FALSE conditions.

If an IF/THEN expression evaluates FALSE, the interpreter searches forward on the same line for the first “£” (ELSE) symbol that follows that expression.

Example:

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

Behavior:

If the expression IF A=1 evaluates FALSE, execution jumps to the first “£” symbol encountered after that expression. In this example, that results in:

```
C=3
```

If A=1 evaluates TRUE, then execution continues normally and the second IF/THEN expression is evaluated.

Evaluation results:

```
A=0, B=0 → C=3
```

```
A=0, B=1 → C=3
```

```
A=1, B=0 → C=3
```

```
A=1, B=1 → C=2
```

When chaining multiple conditional statements on a single line, carefully consider how ELSE resolution will occur. Misplacing a “£” symbol can cause logic to execute differently than intended.

The ML Command Set

The ML commands provide BASIC with direct access to the faster and more powerful machine language routines built into Color 64. Some of these commands perform complex and frequently used operations (such as input handling), while others support lower-level system control.

All ML commands begin with a period (".") followed by two digits. Some commands require additional parameters, which follow the command after a comma.

Example:

1000 .01:IF LEFT\$(TX\$,1)<>"*" THEN 1000

In this example, the .01 command reads a line from disk. If the line does not begin with an asterisk, the program loops and reads the next line. This example also demonstrates how certain input commands use TX\$ as the input buffer.

The following table summarizes the ML commands available in Color 64 v8.1.0a overlays.

Table 41 - ML Command Set

Cmd	Details
.00	Get a typed character. Non-blocking character input. Checks for a keypress from either the local keyboard or the remote user. <ul style="list-style-type: none"> Returns: ASCII value in !02 (0 if no key pressed) Status in !01 Alphabetic characters are automatically converted to uppercase. Example: .00:P=!01:A\$=CHR\$(!02)
.01	Input a line of data from disk. Uses file number in !14. Input ends when: <ul style="list-style-type: none"> Maximum length of TX\$ is reached, or Character in !04 is encountered (unless !15 ≠ 0) Data is stored in TX\$. Character count stored in !40. The terminating character (such as CR) is not included. Data may also be accessed via @0 or assigned directly with @5. Example: .01:SR=ST:A\$=@0
.02	Input a line of text from the user. Reads from keyboard or remote user. Input ends with CR, CTRL/X, or CTRL/P. Returns: <ul style="list-style-type: none"> Status in !01 Character count in !40 Text stored in TX\$. May also use @0 or skip with @8. Example: .02:I\$=@0
.03	Equivalent of the "\$" command... Equivalent to "\$" output command, but requires expression after comma.
.04 / .05	Activate protected mode. When in this condition, the program can only be stopped by holding down SHIFT/COMMODORE/CONTROL on the local keyboard. In the event of a system crash, a GOTO9991 which contains the code to reinitialize. The .05 command disables protected mode.
.06	Equivalent of the "#" command... Expression must follow comma.
.07	Activate terminal mode. Invokes a basic terminal program (Plusterm) on the local system. 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.

Color 64 BBS Manual Version 8.1A March 2026

Cmd	Details															
	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	Session/Input Poll. Polls for typed input (non-blocking, local or remote) See Undocumented features/commands section for more information.															
.09	Single Key Input Poll. Polls keyboard for a single keypress and returns value to A\$ (with high-order bit). See Undocumented features/commands section for more information.															
.10 / .11	Activate carrier-detect checking interrupt. .10 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. .11 deactivates the interrupt.															
.12	Output a sequential file. The file number is stored in variable !14 and the file must already be opened. <ul style="list-style-type: none">Set !16 to 1, to prevent user from aborting output; If set to zero, the output can be aborted by pressing the space bar or by typing CTRL/P.Set !17 to a non-zero value to activate page-pausing, where the value of !17 is the number of lines.															
.13	Dump caller log buffer to disk. Dumps the temporary caller log buffer to disk file number 98 and clears the buffer. Intended to be used only by the normal BBS caller log routines.															
.14	Set level parameters for MCI commands. Format: .14,<msg level>,<variable level> <ul style="list-style-type: none">Uses the value of LV (user access level) for the current user's level, which is why LV must be one of the first variables defined.<msg level> = level for the message MCI's (£a0, £a1, etc.)<ul style="list-style-type: none">Add a value of 128 to <msg level> to disable DD\$ message MCI (£a6) printing<variable level> = level for the variable MCI command (£!) use															
.15 / .16	Set the "Chat Begin" / Set the "Chat End" text. Not implemented in 8.1/8.10a.															
.17	Put text into caller log buffer. Format: .17,<string> <string> = Text to be put in the temporary caller log buffer (usually the string is LG\$). Used in conjunction with the variable !20 and the .13 command to handle caller log functions. Intended to be used only by the normal BBS caller log routines.															
.18	Variable Stack / Killer Saves the current BASIC variable memory environment so it can later be restored with the .19 command. This command operates as a stack. Each time .18 is executed, the entire current variable state is pushed onto an internal stack and a new working environment becomes active. The previous environment remains preserved on the stack while the new environment may be freely modified. The .19 command pops the most recently saved environment off the stack and restores all variable values to that state. Up to eight stacked environments may exist simultaneously. Attempting a ninth consecutive .18 will generate an error. <table><tr><th colspan="3">Variable Stack Example</th></tr><tr><th>Stack Level</th><th>Description</th><th>A\$ Value</th></tr><tr><td>1</td><td>Initial BBS environment after startup</td><td>"Dog"</td></tr><tr><td>2</td><td>.18 executed, A\$ is modified</td><td>"Cat"</td></tr><tr><td>3</td><td>.18 executed again, A\$ modified</td><td>"Dead Cat"</td></tr></table>	Variable Stack Example			Stack Level	Description	A\$ Value	1	Initial BBS environment after startup	"Dog"	2	.18 executed, A\$ is modified	"Cat"	3	.18 executed again, A\$ modified	"Dead Cat"
Variable Stack Example																
Stack Level	Description	A\$ Value														
1	Initial BBS environment after startup	"Dog"														
2	.18 executed, A\$ is modified	"Cat"														
3	.18 executed again, A\$ modified	"Dead Cat"														

Color 64 BBS Manual Version 8.1A March 2026

Cmd	Details
	<p>Example Behavior:</p> <ul style="list-style-type: none"> At Stack Level 1, variables are in their normal initialized state Executing .18 saves this state and creates Stack Level 2. Variables may now be changed without affecting Level 1. Executing .18 again creates Stack Level 3. Stack Level 2 is now preserved on the stack. Variables may now be changed without affecting Levels 1 or 2. Executing .19 restores Stack Level 2 (A\$ becomes "Cat"). Executing .19 again restores Stack Level 1 (A\$ becomes "Dog"). <p>Each .18 pushes a complete variable snapshot. Each .19 restores the most recently saved snapshot.</p>
.20	<p>Set BPS rate. Format: .20,<value> Values: 0–2 (non-SwiftLink) or 0–7 (SwiftLink) Maps to 300–38400 baud. Do not change during active session.</p> <p>Notes:</p> <ul style="list-style-type: none"> 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 / .22	<p>Control DTR (Data Terminal Ready) Line. DTR is a line to the modem which indicates the readiness of the computer to send and receive data. .21 = Activate .22 = Deactivate</p>
.23	<p>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. Used by Plusterm.</p>
.24	<p>Wait for end of modem transmit. Ensures output buffer is empty. Used for non-Swiftlink systems.</p>
.25	<p>Cancel modem output. Cancel output and clear output buffer immediately.</p>
.26	<p>Clear input buffer. Will clear all data holding in the modem input buffer.</p>
.27 / .28	<p>Receive / Transmit a file using current protocol. Used by the BBS file transfer routines.</p>
.29 / .30	<p>Receive / Transmit a header block (multi punter). Used by the BBS file transfer routines.</p>
.31	<p>Used but undocumented. See Undocumented features/commands section for more useless information..</p>
.32	<p>Set DATA statement line number. Functions similarly to the BASIC RESTORE command, but allows explicit control over the line number from which DATA statements will be read.</p> <p>If used by itself (.32), the DATA pointer is set to the current line.</p> <p>If used in the form .32,<number>, where <number> is a BASIC line number, the DATA pointer is set to begin reading DATA statements from that specified line. Example: .32,20000</p>
.33	<p>Activate extra variable memory mode. By default, BASIC variable memory begins immediately after the last byte of the "vbbs.init" overlay. This fixed location allows other overlays to be loaded without disturbing variables.</p>

Color 64 BBS Manual Version 8.1A March 2026

Cmd	Details
	<p>The .33 command relocates the start of BASIC variable memory to immediately after the last byte of the currently loaded overlay. This effectively increases available variable space by the size difference between the current overlay and "Vbbs.init".</p> <p>This command must be used carefully. It should be paired with the variable stack commands (.18 / .19). When .34 is later executed, the variable area is moved back to its original location. If insufficient free space exists at that time, system data or string variables may be overwritten.</p> <p>See the .34 command for more information.</p>
.34	<p>Deactivate extra variable memory mode. Restores the normal variable memory location established by "Vbbs.init", reversing the effect of .33.</p> <p>A .18 should precede each .33, and a corresponding .19 should precede each .34. This ensures variables are preserved and memory is safely returned to its original configuration.</p> <p>If .34 is not executed before "Vbbs.init" is reloaded, a LOAD OVERFLOW error will occur.</p> <p>If insufficient free memory exists when .34 is executed, string variables may be overwritten.</p>
.35	<p>Defines the color sequence used by the system's standard rainbow mode (such as the colors cycled by the F1 and F5 keys).</p> <p>Format: .35,<string></p> <p><string> must contain exactly eight color control characters. These eight characters determine the rainbow color rotation order.</p>
.36	<p>Used but undocumented. See Undocumented features/commands section for more useless information..</p>
.38	<p>Bind / Declare / Register operands (Set up an ML-side descriptor for upcoming operations) Used in conjunction with .39-.40 for performing old-to-new message linking.</p> <ul style="list-style-type: none"> Slot 0 points to MN, slot 1 points to MR%. ;OK, ;1K, ;OF, are scratch registers used. <ul style="list-style-type: none"> ;1K = Read flag for message. ;OF = Message number compared. <p>See Undocumented features/commands: .38 for more information.</p>
.39	<p>Run an iterator/scan using the binding Used in conjunction with .38, .40 and .41 for performing old-to-new message linking. This populates the ";" scratch registers (;OF, ;1K, etc.) that the program loops test.</p> <p>See Undocumented features/commands: .38 for more information.</p>
.40	<p>Commit/Update step for the scan Used in conjunction with .38, .39 and .41 for performing old-to-new message linking. Invoked when a record should be skipped (read already or wrong category), "Advance/commit scan state for K"</p> <p>See Undocumented features/commands: .38 for more information.</p>
.41	<p>Finalize/Flush Step Used in conjunction with .38-.40 for performing old-to-new message linking. Wraps up the relink process.</p> <p>See Undocumented features/commands: .38 for more information.</p>

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, an error will result. Some variables are READ ONLY, as will be indicated in the following listing. 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.

Example: **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.

The table below summarizes the ML Variables:

Table 42 - ML Variable Summary

Var	Description
!00	Carrier detect mode. If less than 128, carrier timeout is enabled. If 128 or greater, carrier timeout is disabled. See !11 .
!01	Status variable. Status after input commands (.00, .02, @8). 0 = OK 1 = Aborted (CTRL/P) 3 = Pause (CTRL/S) 4 = Aborted (CTRL/X) 255 = Carrier lost
!02	General input/output character. Holds character returned by .00 or .07. Otherwise contains undefined data.
!03	Maximum column number for word wrap. When input exceeds this column and word wrap mode is enabled, wrapping occurs. Wrapped word (up to 15 characters) is buffered and reinserted at next input.
!04	End of line character. 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. 0 = ASCII translation 1 = Graphics mode (Commodore/ANSI) 2 = Simulated ASCII (graphics input restricted) See also !18 .
!06	Case constraints on input. If non-zero, alphabetic input (.02 or @8) is forced to uppercase.
!07	Word wrap mode. If non-zero, word wrap is enabled. See !03 .
!08	Wrap/Width Threshold See Undocumented features !8 for further information.
!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. Active only in ASCII translation mode. If non-zero, ASCII 10 follows ASCII 13. See !05 .

Color 64 BBS Manual Version 8.1A March 2026

Var	Description
!11	Carrier timeout flag. 0 = Timer suspended 1 = Timer active (counts to 255 if no carrier) Carrier is checked at 1/60 second intervals (~4.25 seconds timeout). Will hold value of 255 if timed out.
!12	Local mode. 1 = disables modem I/O (local mode - no information is outputted to modem) 0 = enables modem I/O.
!13	Inactivity timeout flag. Non-zero value indicates inactivity timeout has occurred (~2 minutes of inactivity).
!14	File number for disk input. Used by .01, @5, .12 and other disk routines.
!15	End of line mode. If non-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. If non-zero, then the next sequential file read (with the .12 command) cannot be aborted by either the spacebar or CTRL/P keys. Resets to zero after completion of file read.
!17	Number of pager pauser lines. Format: !17,X X=0 disables pager. Non-zero sets pause interval (X number of lines) during .12 sequential reads.
!18	ANSI mode. Active only in Graphics translation mode. If non-zero, ANSI translation is enabled. See !05.
!19	MCI disable mode. If non-zero, MCI processing is disabled for one output line. Automatically resets after output.
!20	Caller log buffer status flag (Read-Only). 0 = Empty 1 = Contains data >1 = Full (requires disk write)
!21	Variable Stack number (Read-Only). Current .18 stack level. 0 if no active stacks.
!22	Network file transfer timeout timer. Set to 128 + minutes when network file transfer initiated (starts timer). 0 disables timer. Transfer aborts if timer expires.
!23	Non-SwiftLink BPS timer table. Table of 18 bytes; holds the timer values used by the Non-SwiftLink systems when sending and receiving the individual bits of data. <ul style="list-style-type: none"> There are 6 bytes for each BPS rate (300, 1200, and 2400 respectively), For each of the 6 bytes there are 2 bytes for the output bit time, input bit time, and half-bit time Each of the two bytes is a 16-bit value arranged in low byte, high byte order. <p>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).</p>
!24	Carrier type. Hardware comparison value for carrier detection (set by Setup).
!25	Carrier status (Read-Only). Non-zero if carrier detected.
!26	SwiftLink/Non-SwiftLink flag (Read-Only). <u>0 = Non-SwiftLink 1 = SwiftLink system</u>
!27	Term buffer mode. If this is not zero, then all characters printed to the screen will also be put into the term buffer. <ul style="list-style-type: none"> Parameters of the buffers are established by the !28, !29, and !30 variables.

Color 64 BBS Manual Version 8.1A March 2026

Var	Description
	<ul style="list-style-type: none"> Output is diverted to file number 3 (normally screen output). 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. <ul style="list-style-type: none"> !28(0) is the low byte, and !28(1) is the high byte. 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 (Read-Only). 0=300, 1=1200, 2=2400, 3=4800, 4=9600, 5=14400, 6=19200, 7=38400
!32	Modem Input buffer empty flag. 0 = Empty Non-zero = Characters pending
!33	X-Modem file conversion mode. If non-zero, converts ASCII to PETSCII during transfer.
!34	Number of tries for X-Modem CRC mode. 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. Sets the Block size in bytes for Punter protocol.
!36	File type for transfer. 1 or 3 = PRG 2 = SEQ
!37	Current protocol type (Read-Only). 0 = Punter 1 = X-Modem
!38	File transfer timeout flag. Non-zero after transfer complete indicates carrier or network timeout during transfer.
!39	End of transfer status. Non-zero indicates aborted transfer.
!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.
!41	Alternate output file mode. If non-zero, BBS output is diverted to file #3. <ul style="list-style-type: none"> Affects BBS output only (BASIC PRINT commands unaffected).
!42	Day of the month (Read-Only). Updated by @16 function.
!43	Month of year (Read-Only). Updated by @16 function.
!44	PM/AM Flag. 0 = AM Non-zero = PM
!45	Current time hour (Read-Only). Holds values 1–12, updated by @11 .
!46	Scratch register General-purpose read/write variable. See Undocumented features: !46 for more information.
!47	Character output delay. May be used to slow output of the system for slower systems. 0 = No delay 255 = Maximum delay

Color 64 BBS Manual Version 8.1A March 2026

Var	Description
!48	Fast garbage collect mode. If non-zero, enables fast string garbage collection. Triggered automatically when free string space < 256 bytes. Supports up to 2048 strings; disables if exceeded. Screen blanks during execution. Recommended for string-intensive routines.
!49	Boot device numbers (Read-Only). A 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 (Read-Only). A 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 (Read-Only). Defined by BOOTMAKER program. 0 = No swap required Non-zero = Disk swap required See also !49, !50, and the function @30.
!52	Most recent find location (Read Only). Holds location result of @2 or @25 find functions.
!53	Flag - 2Mhz during Fast Garbage Collect. If 1, C128 attempts 2 MHz mode during fast garbage collect. No effect on C64. Automatically set for C128 users.
!55	Message record delimiter flag See Undocumented features: .31 for more information.
!56	Message editor command state Indicates editor command mode. See Undocumented features: !56 for more information.
!57	Editor prefix trigger Defines key entering editor command mode. See Undocumented features: !57 for more information.
!58	Editor variable (Unknown function) Under investigation. See Undocumented features: !58 for more information.
!59	Text-entry mode flag (Editor) Non-zero indicates text-entry-only state. See Undocumented features: !59 for more information.
!60	Editor menu selection Stores selected editor menu item. See Undocumented features: !60 for more information.

The ML Functions

The ML functions provide operations that are more flexible than ML commands or ML variables. In practice, they behave like built-in BASIC functions: some execute an ML routine and then return a computed value. Functions can return either numeric or string results (including string values that cannot be returned through ML variables).

All ML functions begin with an "@" followed by one or two digits (the function number). Functions are either STRING or NUMERIC, as indicated in the table. Some functions accept required and/or optional parameters, which are placed in parentheses after the function number.

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

In this example:

- @5 reads a line from disk (like .01) and returns it as a string.
- VAL() is used to test whether the returned line is numeric; if so, @1 prints the number without the leading space added by STR\$ for positive values.
- The loop continues until end-of-file (as indicated by ST).

All ML functions (@##) invoke address \$4E21 for ML processing.

The table below lists the ML Functions and descriptions.

Table 43 - ML Function Descriptions

ML FUNC	Format & Description
@0	Input Buffer Slice (TX/!40) Format: @0 : STRING Equivalent to LEFT\$(TX\$,!40). Commonly used after disk input (.01) or keyed input (.02) when the input buffer is TX\$ and the character count is in !40. Example: .01:A\$=@0
@1	STR\$ Without Leading Space Format: @1(<number>) : STRING Returns the equivalent of STR\$(<number>) without the leading blank for positive numbers. Negative values retain the minus sign. Examples: <ul style="list-style-type: none"> • @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	Find Substring (Forward) Format: @2(<string1> , <string2> [, <number>]) : NUMERIC Searches for <string1> within <string2> from left to right and returns the 1-based position. Returns 0 if not found. Optional <number> behavior: <ul style="list-style-type: none"> • If omitted: return position of first occurrence • If <number> > 0: return position of the <number>th occurrence • If <number> = 0: return the count of occurrences of <string1> in <string2> Examples: <ul style="list-style-type: none"> • @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

Color 64 BBS Manual Version 8.1A March 2026

ML FUNC	Format & Description
	The !52 variable stores the most recent find position (0 if the count form was used).
@3	Repeat First Character Format: @3(<string> , <number>) : STRING Returns a string made of the first character of <string> repeated <number> times. Example: @3("A",5) returns "AAAAA" <number> range: 0–255. If <string> is null, returns null.
@4	Modem GET Character Format: @4 : STRING Modem input character (GET-style). Returns one character if available; otherwise returns a null string. Example: A\$=@4.
@5	Read Disk Line (Returns String) Format: @5 : STRING Performs a disk line read (.01) and returns the resulting line as a string. Similar to @0, but executes the read first. Example: A\$=@5:SR=ST:PRINT#9,A\$;
@6	Extended ASC / 16-Bit Extract Format: @6(<string1> [, <string2>]) : NUMERIC Extended ASC for 8-bit and 16-bit extraction. Behavior depends on argument form: <ul style="list-style-type: none"> One string, length 0–1: returns PETSCII of first character; returns 0 if null string (no error). Example: GET#8,A\$:I=@6(A\$) One string, length 2: returns 16-bit value from the first two characters (low byte = 1st char, high byte = 2nd char). Example: IFLEN(I\$)=2THENL=@6(I\$) Two strings: treats first character of each as low/high bytes and returns a 16-bit value. Null strings are treated as 0. Example: GET#8,A\$,B\$:L=@6(A\$,B\$)
@7	Extended CHR\$ (16-bit to 2 Bytes) Format: @7(<number>) : STRING Extended CHR\$. Returns a 2-character string containing the low byte then high byte of <number>. Useful for relative file positioning. Example: PRINT#15,"p"CHR\$(104)@7(RN)CHR\$(1)
@8	Read Keyed Line (Returns String) Format: @8 : STRING Performs keyed line input (.02) and returns the resulting line as a string. Example: I\$=@8:P=!01:IFP=0THENRETURN
@9	Enhanced FRE() Format: @9 : NUMERIC Enhanced FRE(). Returns free memory as a positive value and does not force garbage collection; uses an internal calculation routine. Example: I=@9:IFI>3000THENRETURN
@10	Strip Control/Graphic Codes Format: @10(<string> [, <number>]) : STRING Returns <string> with graphics/control characters removed. Optional <number> returns only the first <number> characters (LEFT\$-style). <number> range: 0–255. Example: A\$=@10(@0,15)
@11	Get Time String Format: @11 : STRING Returns current time as "HH:MM am" or "HH:MM pm". Updates !45 (hour) and !44 (AM/PM flag: 0=AM, non-zero=PM).
@12	Overlay String / Replace at Position Format: @12(<string1> , <string2> , <number>) : STRING Overlays <string1> onto <string2> starting at position <number> (1–255), replacing existing characters. Pads with spaces if needed. Examples:

Color 64 BBS Manual Version 8.1A March 2026

ML FUNC	Format & Description
	<ul style="list-style-type: none"> • @12("there","Hello",7) will return the string "Hello there" • @12("","+++++",4) will return "+++*++++" <p>Special replace mode: if <number>=0, uses the most recent find position from @2 or @25. If that value is 0 (or if @2/@25 was used in count mode), an ILLEGAL QUANTITY error will occur.</p> <p>Example (replace all "." with "/"):</p> <p>1000 IF @2(".",I\$)>0 THEN I\$=@12("/",I\$,0) : GOTO 1000</p>
@13	Capture Crash Error Format: @13 : STRING Crash routine helper: returns the BASIC error message text associated with the crash.
@14	Capture Crash Line Number Format: @14 : STRING Crash routine helper: returns the BASIC error line number as a string.
@15	Convert Date to ADN Format: @15(<year> , <month> , <day>) : NUMERIC Returns an ADN (absolute day number) from year/month/day. Returns -1 if any parameter is invalid.
@16	Convert ADN to Year Format: @16(<number>) : NUMERIC Converts ADN in <number> to a date and returns the year. Updates !43 (month) and !42 (day). Invalid ADN yields undefined results.
@17	Convert ADN to Day of Week Format: @17(<number>) : NUMERIC Returns day-of-week from ADN in <number>: 0–6 for Sunday–Saturday.
@18	Convert ADN to Date String Format: @18(<number>) : STRING Returns date string "MM/DD/YYYY" from ADN in <number>. If ADN is invalid, returns "--/--/----".
@19	Convert Date String to ADN Format: @19(<string>) : NUMERIC Converts date string "MM/DD/YYYY" to ADN. Returns -1 if invalid.
@20	Current Overlay Filename Format: @20 : STRING Returns the filename of the BASIC overlay currently in memory (used by crash routines).
@21	Alpha-Only, Lowercase Filter Format: @21(<string>) : STRING Returns <string> with all non-alphabetic characters removed and converts letters to lowercase. Example: @21("This Is A Test") would return "thisisatest".
@22	Compress Number for Disks Format: @22(<number>) : STRING Returns a compressed numeric string for disk storage by packing two digits per byte (approximately half-length of STR\$ output). Uses a special encoding to avoid producing a carriage return character. Example: PRINT#8,@22(I)
@23	Uncompress Number from Disk Format: @23(<string>) : NUMERIC Inverse of @22: uncompresses a number stored in <string>. Invalid strings yield undefined results. Example: INPUT#8,I\$:=@23(I\$)
@24	Pad/Terminate Relative Record Format: @24(<string> , <number>) : STRING Pads/truncates a record to length <number> for relative file use. <ul style="list-style-type: none"> • If <string> is shorter than <number>, pads with carriage returns to reach <number>. • If <string> is >= <number>, truncates to <number>-1 and appends a carriage return. Null strings or <number><2 may yield undefined results. Example: F\$=F\$+@24("name",15)
@25	Find Substring (Reverse) Format: @25(<string1> , <string2> [, <number>]) : NUMERIC Same as @2, but searches from right to left (end of <string2> toward the beginning).
@26	Get Delimited Fields

Color 64 BBS Manual Version 8.1A March 2026

ML FUNC	Format & Description
	<p>Format: @26(<string1> , <string2> , <number>) : STRING</p> <p>Returns the <number>th section of <string2> separated by the delimiter in <string1>.</p> <p>Example: @26("!", "cp6!i6!cd//directory", 2) returns "i6" because the "!" is the separator character, and "i6" is the second section divided off by the "!" character.</p> <p><number> must be non-zero.</p>
@27	<p>Convert Calendar Age from ADN</p> <p>Format: @27(<number1> , <number2>) : NUMERIC</p> <p>Returns calendar age in years between two ADNs: from <number2> to <number1>.</p> <p>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.</p>
@28	<p>Find String in Array</p> <p>Format: @28(<string> , <array>) : NUMERIC</p> <p>Searches a one-dimensional string array for an exact match to <string> and returns the index.</p> <p>Returns -1 if not found.</p> <p><Array> is the array name without "\$" or parentheses (for example, pass A for A\$()).</p> <p>Errors if array is missing or not 1-D.</p> <p>Example: I=@28(A\$,A) : IFI<0THEN#"not found!"</p>
@29	<p>Read N Characters</p> <p>Format: @29(<number>) : STRING</p> <p>This function is like @5, except that <number> is how many characters you want read in.</p> <ul style="list-style-type: none"> • It is a viable alternative to the GET# command in some cases. • It does not check for End of File • It does not stop at the End of Line character stored in !04.
@30	<p>Boot Drive Init Command</p> <p>Format: @30(<number>) : STRING</p> <p>This function is used in conjunction with the variables !49, !50, and !51.</p> <p>It returns the drive init command for one of the three boot drives:</p> <p>0=Boot drive, 1=Program drive, 2=External drive.</p>
@31	<p>Sequential Read (Enhanced)</p> <p>Format: @31</p> <p>Newer function (not documented in the v8.0 manuscript).</p> <p>See Undocumented features: @31 for more information.</p>
@32	<p>Retrieve File Block Count</p> <p>Format: @32</p> <p>Newer function (not documented in the v8.0 manuscript).</p> <p>See Undocumented features: @32 for more information.</p>
@33	<p>Return 1-Byte Null String</p> <p>Format: @33</p> <p>See Undocumented features: @33 for more information.</p>

Generic Routines

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

Callable Routines

Table 44 - 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
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 vbbs.init

Color 64 BBS Manual Version 8.1A March 2026

Line	Function
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 response from modem for 2 seconds
15915	Print B\$ init string to modem, and wait 3 seconds for OK response

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. The table below provides the list of all the groups and their respective assignments:

Table 45 - 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 * (User Profile)
5	Public Messages	13	Auxiliary Files 2 *
6	Private Files	14	Auxiliary Files 3 * (Games)
7	Text Files	15	Reserved for future use
<p>* 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 H=12 for AUX1 H=13 for AUX2 H=14 for AUX3 perform a GOSUB460</p>			

Color 64 BBS Manual Version 8.1A March 2026

Color 64 BASIC Variables

The table below defines all variables used by Color 64 BBS:

Table 46 - 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 in other routines that use the array A\$().
A\$	This variable has several uses. It is used to return the character read by the routine at line 110, and it is also frequently used when information is read from disk. The general output commands (#, \$, %, & and ') will print the contents of A\$ if no string expression is specified. This is a 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 message lines. It is also used by many other routines that handle larger amounts of string data. It is always dimensioned to 232 when the system is first initialized.
A1\$	First line of the current caller's address as stored in the password file.
A2\$	Second line of the current caller's address as stored in the password file.
A3	Used in the main overlays to flag that an operation with AUX3 (Games) is about to be performed. It is immediately cleared after use at line 104. Applies to version 8.1a only.
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 are released into, and which UD directory Network file transfers are released into. Defined in Net Setup.
AT	Stores whether the modem supports DTR disconnect.
AZ\$	The MODEM INIT string for Hayes-type modems as defined in SETUP (example: ATE1X1S0=1S2=43F1Q0V1M0).
B	General use.
B\$	General use.
BA	Used only in the Network overlays. 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 vvariables.
BN%	Used only in Network overlays. If this variable is set, the system will 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 0, the caller last used Xmodem; otherwise, they last used Punter.
C	General-use variable often used to temporarily compute the current caller's absolute download credit.
C\$	General-use variable often used to temporarily hold the contents of TX\$, so the previous value of TX\$ can be preserved in cases where TX\$ needs to be changed.
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 given to each caller as defined in SETUP. See also C1, PD, and PU.
C2\$	This variable is two CR\$'s added together. It saves memory and time to print C2\$ instead of CR\$CR\$ or CHR\$(13)CHR\$(13).
C3	Credit system exemption level as defined in SETUP.
C4	Maximum number of files allowed on the public messages drive as defined in SETUP.
CA	Stores the current message category. It is used only in vbbs.msgs.
CA\$()	Category names as defined in SETUP.
CA%()	Category of each message stored in public messages.
CA()	Level for each category as defined in SETUP.
CC	Number of message categories as defined in SETUP.
CD	Modem carrier-detect status value as defined in SETUP.
CL%	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 displayed at the ONLINE: prompt after a caller requests chat mode.
D()	This array stores the drive assignments for file groups as defined in SETUP. It is not wise to change these values unless you understand how the device number and drive number have 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\$()	Array (1 to 12) of month names, January through December.
D2	The current day of the week (0 to 6, for Sunday through Saturday) as calculated by the timer routine at 1110-1190.
D2\$()	Array (0 to 6) of day names, Sunday through 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.

Color 64 BBS Manual Version 8.1A March 2026

Var	Definition & Function
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 for 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 (for example, 0, 1, etc.).
DT	Number of downloads made by the caller on this call.
DV	Device number of the currently selected drive.
DX	Last selected device/drive. The system uses this variable to save time so it 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 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.
EM\$	Stores user actions and summary information for the email report.
ER	Error number read from the disk error channel by the routine at 510.
ER\$	Error description read from the disk error channel.
ES	Error sector read from the disk error channel.
ET	Error track read from the 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 password file records. This allows easier modifications to the password file during system upgrades by simply changing the values in SETUP instead of throughout the other program overlays.
F(0)	The BBS's RERUN ON ERRORS status as defined in SETUP.
F1\$	Sometimes used for filename storage prior to calling a file-open subroutine.
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 the BBS is using 1541 Flash! from Skyles Electric Works.
FM	Status of the current caller's word-wrap condition, used only in the message editor. If it is 1, word wrap is on. If it is 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 vbbs.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 the mod menu is entered and removed from memory when the mod menu is exited. Used only in vbbs.ov2.
G()	Mod menu array dimensioned each time the mod menu is entered and removed from memory when the mod menu is exited. Used only in vbbs.ov2.
GA%	Number of game plays for use by Mod Menu. It is first defined in vbbs.init so that the variable-killer routines in vbbs.ov2 do not remove it from memory.
GM	Used only in vbbs.games; flag indicating a valid game choice has been selected.
H	Last selected drive number. H can be a number between 0 and 15 depending on what the last selected group of files was (for example, Password File, System Files, Public Messages, etc.).
H\$	Last used drive init command.
H\$()	Drive commands for drive assignments as defined in SETUP (for example, i0, ui, hm4 20 28).
HD\$()	Custom message header information as read in when a user selects [R]ead Public Msgs.
HM	Highest message number on the system as stored in vvariables.
I	Probably the most-used numeric variable that does not have a specific function. It is used mostly as a general index variable for FOR/NEXT loops.
I\$	General-use variable, although it is specifically designed as the variable in which typed information is stored when the input routines at lines 300-310 are used.
I\$()	General-use array that should never be permanently dimensioned. The multi-scratch, multi-release, and multi-view functions in vbbs.xfer dimension this variable, then execute a variable kill to remove it from memory when finished using it.
I%()	General-use array that should never be permanently dimensioned in memory. The self-contained directory regenerate routine in vbbs.xfer dimensions this array for temporary use, then discards it with the variable killer when the routine is over.

Color 64 BBS Manual Version 8.1A March 2026

Var	Definition & Function
IC	This variable is set to 1 when the 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. It should never be used to store important information because the caller log routine will destroy any information stored in II.
J	General-use variable 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 expiration date (ED\$) 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 to their password file. Nuke's note: This is a system security risk if L2 is ever used for another purpose. DO NOT USE!
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 ADNs for more information.
LD\$	Current caller's last-called date in the form "MM/DD/YYYY". It is calculated from the variable LD.
LG	Temporary variable used by the caller log routines. It can also be used to store temporary information, but the data in LG will be destroyed if the caller log routines are used.
LG\$	Temporary holding place for information to be later stored in the caller log. When this variable becomes longer than about 200 characters, it is automatically stored into an area of RAM that contains the caller log, and the variable is reset to "".
LK%()	This array stores the message links in subject chains.
LM	The highest message number that a caller has read on this and previous calls. When a caller logs off, this variable is 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 later call in which they had not read any messages. On a caller's first call, the BBS does not want to scan for mail, etc.
LM%()	This array stores the status indicating whether a level message exists for each level. These variables are assigned when the BBS is initially loaded or when the current time and date are adjusted.
LV	The current caller's access level, or the access level of the user being edited in password maintenance. Altering this variable will affect only the current call.
M	Used in vbbs.msgs to keep track of the last message read.
M1	System BPS rate as defined in SETUP.
M2	Used in vbbs.msgs to keep track of the next message to be read.
M3	Indicates whether Network is being used, as defined in SETUP.
M4	Indicates whether the system will adjust BPS rate to connect rate, as defined in SETUP.
M6	Default column (word-wrap) setting as defined in SETUP.
M7	Default Fast-Garbage-Collect mode as defined in line 10241 of vbbs.init. See the 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 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 public message.
MP	Maximum password number on the BBS as defined in SETUP.
MR	Number of messages read so far when reading E-mail.
MR%()	This array keeps track of which messages have already been read.
MS	Number of messages to skip to read the next piece of E-mail.
MT	Used in the multi-download 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 vbbs.msgs to keep track of the next message to be read.
N	General use.
N%()	Array holding the 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 messages.
ND	Set when the date changes and reset after the end-of-day routines are run.
NF\$	Used to hold the name and ID# of a remote node.

Color 64 BBS Manual Version 8.1A March 2026

Var	Definition & Function
NI%	Used only in Network overlays. It is the next open space in the incoming node accounts file.
NN\$	BBS system name, defined in Net Setup.
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 vvariables. 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 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 'ON OV GOTO XX,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 inputs to make sure the BBS aborts if carrier is lost or the caller types ↑P.
P\$	General use.
PD	Current caller's number of blocks downloaded. See also C1, C2, and PU.
PH\$	Current caller's phone number as stored in the password file.
PM	<ul style="list-style-type: none"> If this variable is 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 the 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 number of blocks uploaded. The variables C1, C2, PD, and PU are used to determine a caller's 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 also as a temporary holding variable.
R1	Temporary holding area for the sysop's record number when using 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 is saved and then reassigned to the value of the caller number being edited.
RN\$	Current caller's real name as stored in the password file.
RU	System auto-release level as defined in SETUP.
RX%	See variable FU.
S	Used only in Network overlays. Holds the duration of a Network call.
S1	Number of new callers for the current day as stored in vvariables.
S2	Number of calls today as stored in vvariables.
S3	Number of uploads today as stored in vvariables.
S4	Number of downloads today as stored in vvariables.
S5	Number of messages posted today as stored in vvariables.
S6	Number of public net messages waiting as stored in vvariables.
S7	Number of outgoing messages to send as stored in vvariables.
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 is set equal to this variable and then stored in the password file.
SR	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 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 vvariables.
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	Total time, in jiffies, that the 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 the caller has for this call. TM-INT(TI/3600) represents the number of minutes the caller has left for this call.
TS	Number of times the current caller has called. It is saved in the password file.

Color 64 BBS Manual Version 8.1A March 2026

Var	Definition & Function
TT	This variable represents the time remaining for today less the value in TM. The total of TM and TT represents 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 from keyboard or disk. It should not be changed except by 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 the BBS as defined in SETUP. If it has a value of 1, the system is defined as using upload descriptions. If it has a value of 0, the system is not using upload descriptions.
UX\$	This variable is used to store the first few lines of the upload description while in the message editor in vbbs.xfer.
VE	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	Several routines use the same message editor, and this variable tells the BBS where to go after it finishes editing the message (for example, back to Read Feedback, Read Mail, Read Messages, or Posting a Message).
WD%()	This array stores the purge parameters for the different caller access levels as defined in SETUP.
X	General use.
General variables used without specific definition: A, A\$, B, B\$, C, C\$, I, I\$, II, J, N, P\$, T, X, Z\$	

The Generic Overlay Files

For programmers, Color 64 also includes generic skeleton overlays that can be used to write online games and modules.

- **The vbbs.xxx Module**

The program "vbbs.xxx" includes all of the routines found in "vbbs.ov3", except that it does not contain any "spare" command routines. It can be used for modules that normally have a spare command of their own, but that you may want to install as a non-system overlay

- **The XXX Small Module**

The file called "xxx small" is intended for games and modules that are loaded by the Mod Menu program. This program is stripped down to only 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 or modules that start at line 28000 or above:

- First, delete the old overlay portion of the game (lines 0-27999), if necessary.
- Next, merge in "xxx small". Remember to change line 1 to reflect the filename for re-saving purposes.
- Next, enter line 5 as follows:
`5 gosubXXXXX:gosub489:.dr$+"vbbs.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.

The game or module is then ready to load from the Mod Menu. Line 5 ensures that control returns to the Mod Menu when the game or module ends. Any game or module accessed through the Mod Menu MUST return to the Mod Menu so that a "variable kill" can be activated.

Many modules add extra variables and arrays to memory that will not be used again until the module is loaded another time. Mod Menu handles this by removing those 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 to store other "permanent" information. These variables will be lost unless they are created before the game is played. If desired, you can assign these variables a dummy value in the vbbs.init initialization routine so they will be preserved.

- **The XXX Off-Line Module**

The included file "xxx ol" is a special module that allows you to 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 full BBS system.

Color 64 BBS Manual Version 8.1A March 2026

Lines 10500 to 10550 of the overlay are where the user parameters are set. Set the different variables in this line range to values that will be useful during testing of your program. Consult the "Basic Variables" section for information on what each variable is used for.

The first line of your routine should be at line 11000, because the program drops through to this line after the vbbs.parms file has been read in. If your program does not normally start at this line, for example if the game starts at line 30000, then you can simply place a GOTO command at line 11000 to transfer control to the appropriate line number.

The ML shell must be installed before you can run a program with this module. This means you will need to use an off-line program after shutting down the BBS system, or 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 need to reboot the BBS in order to test the program again. Simply RUN it and it will restart.

▪ **The XXX Message Editor Module**

Some games or 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 "vbbs.xxx". The routines and their entry points are as follows:

- Line 1205 - Displays the message "Message Maker" and asks for a device number, drive, command, and filename for editing. This is usually intended for Sysop or Co-Sysop modules only.
- Line 1215 - Prints "opening <filename>...". The filename must be placed in the variable FI\$ before calling this routine. The currently selected drive is used. The program then proceeds into the next routine. Use this routine when the user needs to see which file is being opened.
- Line 1220 - Attempts to open a file, using 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, the routine drops into the message editor so a new message can be created. Use this routine for modules that edit existing files, such as a Trivia module that allows the Trivia Sysop to edit questions.
- Line 1270 - Jumps directly into the message editor. The message is stored using the filename in FI\$ and is written to 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 includes all of the features of the editors built into the other modules, including the Help command and the Merge command, for those who have access.

Overlay Cross-Reference

The actual command routines are usually located in various places throughout the overlays. Sometimes an entire command is not fully contained in one overlay. For example, the [A]lter Password command must eventually branch to Vbbs.init to change the password. In the table below, the overlay listed is the first overlay branched to. For the Alter Password command, that first destination is Vbbs.ovl. The line numbers for the true underlying routines are listed under **Actual**.

Commands not present in the current overlay are sent to one of the overlay loader lines, which exist at lines 88 to 103. These lines load the overlay in which the command actually resides. The loader line is shown under **Load**, and the overlay abbreviation is shown under **Overlay**.

Before loading the new overlay, the variable **OV** is set so that the newly loaded overlay knows where to branch once loaded. Usually this is set to **1** automatically when the standard loader lines are used, because most overlays reserve **OV=1** for a command relayed from another overlay.

Once the new overlay is loaded, the value in **I** is used again by the branch tables beginning at line 13160. In this way, the command number is preserved even though the command has been handed off to another overlay.

Sometimes the **OV** setting is not 1. This usually occurs in overlays that do not have a normal main prompt of their own, such as Vbbs.init. In those cases, the loader must set a different **OV** value so the receiving overlay enters the correct routine.

Table 47 - OV Settings Cross Reference

Key	No	MINI	Actual	Overlay	Load	OV	Function
R	0	13202	3005	MSGS	90	1	Read Public Messages
@	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
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
+	26	13400	35020	OVL	96	1	Multi-Upload
Z	27	13410	13710	XFER	92	1	Edit Download Description

Color 64 BBS Manual Version 8.1A March 2026

Key	No	MINI	Actual	Overlay	Load	OV	Function
Y	28	13420	14010	XFER	92	1	Release a Download
1	30	13430	14310	MSG5	90	1	Spare Command 1 (8.1a Games)
2	31	13440	13440	XFER	92	1	Spare Command 2 (8.1a User Profile)
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
?	--		13010	All			Main BBS Menu

The Overlay Loader Lines

The table below defines the standard main overlay loader lines. Some of these lines are not present in every overlay because they may not be needed.

Table 48 - Standard Main Overlay Loader Lines

Line	Overlay	Line	Overlay
89	vbbbs.nw2	97	vbbbs.ovl
91	vbbbs.msgs	99	vbbbs.ov2
93	vbbbs.xfer	101	vbbbs.ov3
95	vbbbs.init	103	vbbbs.nw1
35651	vbbbs.games	35652	vbbbs.profile

In overlays that have a main prompt, there is usually a line immediately before each loader line that sets **OV=1**. For example, to set **OV=1** and load the vbbbs.msgs overlay, use **GOTO 90**, which is one line before 91.

Individual Overlay Branching

The following sections describe how each overlay behaves depending on the setting of **OV** when the overlay is loaded. The path of GOTOs, GOSUBs, and overlay jumps is listed in a shorthand format.

Example path from vbbbs.nw1: **OV=3 : 50, (12010), (12220), [6/INIT]**

This regenerates the Net index from the Net Menu

In this notation:

- Plain line numbers, such as **50**, indicate a **GOTO**.
- Line numbers in parentheses indicate a **GOSUB**.
- A jump to another overlay is shown as **[N/XXXX]**, where **N** is the new value of **OV** and **XXXX** is the destination overlay.

Using the example above, the actual flow would be:

Color 64 BBS Manual Version 8.1A March 2026

5 goto50

Line 5 is always the first branch point, where the overlay performs its **ON OV GOTO** dispatch.

50 gosub12010:gosub12220:ov=6:goto95

Line 95 is standard INIT loader

In some cases this shorthand is slightly simplified from the exact code, but it provides a reliable map of the actual control flow.

The “OV” Settings for Overlays

The table below shows the paths for all defined settings of **OV** in the various overlays.

Table 49 – Path for settings of OV - vBBS.INIT Overlay

Line Command	Setting area
vbbs.init	
OV=1 : 40, (19044), 30	Post-logout 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 vbbs.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
OV=9 : 78, (13735), [3,MSGs]	Set new password (relayed from vbbs.ovl)
OV=10 : 65, (12092), [2/MSGs]	End of midnight routine (back to WFC)
OV=11 : 35, (11905), 30	Reset vlevel 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
OV=14 : 79, (13735)	Save User Profile from vbbs.profile
vbbs.msgs	
OV=1 : 60, (13160), 57	Command relayed from other overlay
OV=2 : 50, <read vwelcome2>, (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]	Logout
---- : 74, [3/INIT]	Back to Answer SYSOP Feedback prompt
vbbs.xfer	
OV=1 : 60, (13160), [3/OVL]	Command relayed from other overlay
OV=2 : 65, (22010), [4/INIT]	DOS command (from SYSOP Menu)
vbbs.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	Purge old mail Validate disks update download accesses

Color 64 BBS Manual Version 8.1A March 2026

Line Command	Setting area
IF M3=1 THEN [7/NW1]	-- Continue midnight routine in INIT -- Continue with Network midnight routines
OV=3 : 80, (19010), [1/INIT]	Logoff
vbbs.ov2 * and vbbs.ov3	
OV=1 : 60, (13160), [3/OVL]	Command relayed from other overlay
* (Note that the included vbbs.ov2 overlay jumps directly to the Mod Menu without going through 13160 first)	
vbbs.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
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
vbbs.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

Branching Tables

Command Branches

The table below cross-references the main prompt command keys, their internal command numbers, and where they branch.

When a user presses a key at the main prompt, Color 64 does not immediately jump to a routine based on the key itself. Instead, it performs a lookup:

- The pressed key is first converted to its ASCII value.
- The program searches the array **CM%(I,1)**, which stores the ASCII value for each defined command key.
- When a match is found, the corresponding array position becomes the command number in **I**.
- The caller's access level is then checked against **CM%(I,2)**, which stores the required level for that command.
- If the key is valid and the caller has access, the value in **I** is then used by the branch tables beginning at line 13160.

In the table below, the value shown under **No.** is the command number ultimately stored in **I**.

Table 50 - Spare Command Pre-Assignments

Command	Value of "I"		Command	Value of "I"
@	1		N	22
P	2		X	23
S	3		T	24
\$	4		L	25
D	5		+	26
#	6		Z	27
U	7		<blank>	29
!	8		1	30
F	9		2	31
C	10		3	32
A	11		*	33
O	12		5	34
G	13		6	35
H	14		7	36
W	15		8	37
M	16		9	38
I	17		%	39
E	18		=	40
↑	19		&	41
>	20		-	42
<	21			

For commands that are in the currently loaded overlay, there are ON GOTO statements to load a special "mini routine" specific to that command which exists in the line ranges 13200 to 13480 of the overlays. That

mini-routine normally performs a **GOSUB** to the actual command routine, then does a **GOTO 13100** to return to the main prompt.

This extra step exists because some command routines are also called from other places, such as the SYSOP menu. In those cases, the routine must perform its work without always forcing a return to the main prompt. The mini-routine provides a standard entry path for main-prompt commands while still allowing the underlying routine to be reused elsewhere.

The line numbers for these mini-routines are listed in the table below under the heading **Mini**.

How the ON GOTO branch tables work

The series of **ON GOTO** statements beginning at line 13160 is easier to understand if you think of it as a two-step dispatcher.

First, line 13160 chooses which group of commands is being handled. The commands are grouped into blocks of ten:

- 0 to 9
- 10 to 19
- 20 to 28
- 30 to 39
- 40 to 42

Line 13160 does not jump directly to the final command routine. Instead, it jumps to the correct *branch table* for that group.

For example:

ON I/10+1 GOTO 13170,13180,13190,13195,13197

If **I=7**, then **I/10+1** evaluates to 1, so BASIC branches to line **13170**, which handles commands 0 to 9.

If **I=24**, then **I/10+1** evaluates to 3, so BASIC branches to line **13190**, which handles commands 20 to 28.

Once inside the proper group, a second **ON GOTO** chooses the exact destination within that group. Example: **13170 ON I+1 GOTO 13202,13205,13210,13220,92,92,92,92,96,13260**

This line handles command numbers 0 through 9. BASIC uses **I+1** because **ON GOTO** counts from 1, while the command numbers begin at 0.

So, if **I=0**, BASIC takes item 1 in the list and branches to **13202**. If **I=1**, it takes item 2 and branches to **13205**. If **I=4**, it takes item 5 and branches to **92**.

The same idea is used for the later ranges, except the program subtracts an offset so the numbering again starts at 1 for that range:

- **ON I-9 GOTO ...** handles 10 to 19
- **ON I-19 GOTO ...** handles 20 to 28
- **ON I-29 GOTO ...** handles 30 to 39
- **ON I-39 GOTO ...** handles 40 to 42

This is why the branch tables can look confusing at first glance. They are not testing the key directly. They are using the command number in **I**, first to select the correct table, then to select the correct destination within that table.

Color 64 BBS Manual Version 8.1A March 2026

Example - using the code below:

```
13140 FOR I=. TO 42:IF ASC(A$)<>CM%(I,1) NEXT:GOSUB995:@
13145 @I=29 OR LV<CM%(I,2) OR I>39 AND M3=.
13150 #:GOSUB8004
13160 ON I/10+1 GOTO13170,13180,13190,13195,13197
13170 ON I+1 GOTO13202,13205,13210,13220,92,92,92,92,96,13260
13180 ON I-9 GOTO96,96,13290,96,96,96,13330,96,13360,13370
13190 ON I-19 GOTO92,13375,96,92,96,96,96,92,92
13195 ON I-29 GOTO35651,35652,96,98,100,96,96,96,96,96
13197 ON I-39 GOTO102,88,88
```

If the user presses **P**:

- the search finds **P** in the command array
- **I** becomes **2**
- line 13160 branches to **13170** because command 2 is in the 0 to 9 range
- line 13170 evaluates **ON I+1**, which becomes **ON 3**
- BASIC takes the 3rd item in that list, which is **13210**

That is the mini-routine for the **[P]ost a Public Message** command.

If the user presses **3**:

- **I=32**
- line 13160 branches to **13195**
- line 13195 evaluates **ON I-29**, which becomes **ON 3**
- BASIC takes the 3rd item in the list, which is **96**

Line 96 is an overlay loader, so command 32 is not handled locally in that overlay and must be relayed elsewhere.

Color 64 BBS Manual Version 8.1A March 2026

SYSOP Menu Branching

The table below shows where each menu command branches, by overlay.

Table 51 - Menu Branch Destinations

Key	Line (+function, if applicable)	Destination
SYSOP MENU - vbbs.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
F8	12640, 9999	Shut BBS down
NETWORK MENU - vbbs.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 - vbbs.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

Midnight Routines

The midnight maintenance routines begin in the vbbs.init overlay and continue through several other overlays. The table below maps this process.

Table 52- Midnight Maintenance Routine Paths

Line Command	Branch Functional Area
vbbs.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 <ul style="list-style-type: none"> • 28005 purges old mail • 25005 validates disks • 25100 updates download access count
7/NW1 : 85, (12010),(12220), [8/NW2]	Calls to

Color 64 BBS Manual Version 8.1A March 2026

Line Command	Branch Functional Area
	<ul style="list-style-type: none">12010 regenerate Network Index12220 count public net msgs holding
8/NW2 : 22, (21120), [10/INIT]	Call to 21120 create vnode x users
10/INIT : 65, (12092)	Call to 12092 end of midnight routine; return to WFC

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. Good options are BAID64, since it is in the public domain, and it does a true merge (not just an append), and Solidus SYSRES.
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. The table below summarizes the defined overlays and lines executed for spare commands 1-9:

Table 53 - Spare Command Pre-Assignments

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

Color 64 BBS Manual Version 8.1A March 2026

As you may notice, there is only one spare command pointing to vbbs.ov2 and vbbs.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.

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 vbbs.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.

Adding Games

An easy-to-customize Games module (**vbbs.games**) has been added that resides in the AUX3 area. Also included with this build are several games that were ported from older 7.37 systems and updated for use with Color 64 v8.1. All games were coded to use the AUX3 area (H=14).

The Mod Menu was intentionally avoided so the overall look and feel of the BBS could be customized more easily. During testing, occasional errors were encountered when using the Mod Menu in an SD2IEC environment. These issues were not investigated further because the AUX3 approach worked reliably. The issue is suspected to be related to REL file handling.

All files required by the games are located in the AUX3 area, including the executable program files as well as their supporting SEQ and REL files. In several cases, the original game code was modified to support this arrangement.

Games included on the 8.10a Install Disks are:

- Asylum
- DragonSlayer
- Star Trek
- Empire II (updated)
- BWF Wrestling
- Nuke'em v6
- Big Trouble in Little China

For version 8.10a, games are accessed using **Spare Command 1** from the main BBS menu. This command must be enabled in the **+SETUP** program in order for it to function.

The SYSOP can determine which access levels are permitted to use this command. The command assignment itself can also be changed through **+SETUP** if desired.

The game assignments are intentionally disabled in the **vbbs.games** overlay by default. This allows the SYSOP to control which games are active. To enable a game, edit the overlay and remove the corresponding **REM** statement (or add a new line) and add the option to the menu displayed in **vgames menu**.

See the example below:

```

36198 rem remove rem statements from the
games you want activated
36199 rem be sure to update your menu!
change # selection to match!
36200 gm=:
36201 ifa$="1"then#:$="Entering Empire
II":i$="/game.emp1x":gm=1
36202 rem ifa$="2"then#:$="Entering the
Asylum":i$="/bbs.asylum*":gm=1
36203 rem ifa$="3"then#:$="Hitting the
mat":i$="/bbs.wrestle*":gm=1
36204 rem ifa$="4"then#:$="Going
nuclear":i$="/bbs.nuke em*":gm=1
36205 rem ifa$="5"then#:$="Big Trouble
there":i$="/bbs.bt1c*":gm=1
36206 rem ifa$="6"then#:$="Greetings
Dragonslayer!":i$="/bbs.dragon*":gm=1
36207 rem ifa$="7"then#:$="Time to beam
up!":i$="/star trek":gm=1
36298
ifgm=1thenz$=:i$:gosub35950:a$=c2$+a$:goto36300

```

Color 64 BBS Manual Version 8.1A March 2026

You may either remove the corresponding `REM` statement to activate a command, or create a new program line with the appropriate menu option.

The selection logic operates as follows:

- The user enters a menu selection which is stored in `A$`.
- Line **36200** clears the flag that indicates whether a valid selection was made (`GM=.`).
- Lines **36201–36297** contain the command checks for valid selections.

For each valid selection:

- The program checks if the user's entry matches the option.
- The selection is echoed back to the user.
- `A$` is reused to print a confirmation message (for example: "Loading Empire!").
- The game filename is stored in `I$`.
- The valid-selection flag is set (`GM=1`).

Line **36298** checks the value of `GM`. If a valid selection was made, the program branches to `AUX3` to locate and load the requested file.

Note that the file `vgames menu` must contain the menu text shown to users so they know which options are available.

If you are using a REU and want the games to reside in REU memory, the game PRG files can be moved to the main PROGRAMS drive. In this case, the utility `vsys.remove` must be modified so that it copies these files into the REU.

If you choose to do this, you will need to verify that each game's code references the correct drive location (PROGRAMS or AUX3). Some games reference additional files internally. **Empire II** should be examined carefully because it loads multiple program and sequential files during execution.

If you are not comfortable modifying these programs, it is recommended that you simply leave them in the AUX3 area and do not attempt to relocate them to the REU.

One game requires an initial setup procedure before it can be used:

Big Trouble in Little China

Run the program `BTLC SETUP`.

For SD2IEC users, when prompted with:

```
Insert SYSTEM disk in drive 8
```

you may simply press `Y` when asked "OK?". The program will create the required REL file in the AUX3 directory unless you have changed directories beforehand.

Before placing the BBS into operation, it is recommended that the SYSOP run each game at least once as a player to ensure that all required files are created properly.

This is particularly important for **Empire II**. The first time the program runs it will create all required files and then return to the command prompt. Running the program a second time will start the game normally.

Nuke'em requires initialization on first run. When prompted to reset the game, answer **Y**. The program will scroll while creating the required records and then return to the command prompt.

After initialization, run the program again. If it asks whether you are a new ruler, the initialization was successful and the game is ready to play.

For **BWF Wrestling**, press **8** the first time the program runs to create the required REL file. Note that this command is not displayed on the screen.

Using Disk Drives

Using Legacy Commodore Disk Drives

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

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

The BASIC **OPEN** command uses these parameters. Its syntax is:

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

The drive number is included in the file name. Example: `OPEN 9,8,2,"0:TEST"`

In this example:

- **9** = logical file number
- **8** = device number
- **2** = secondary address
- **0** = drive number
- **TEST** = file name

Each parameter is described below.

Device Number

Any device connected to the Commodore serial port has a **device number**. This number allows the computer to distinguish between multiple devices such as printers, plotters, disk drives, or other peripherals.

Each device connected to the system must have a unique device number. If two devices share the same number, the computer cannot determine which device should respond.

Traditionally, disk drives use device numbers from **8 to 30**. For example:

- The Commodore 1541 drive typically supports device numbers **8–15**.
- Devices such as the CMD HD can support device numbers up to **30**.

Example: `LOAD "0:FILE",8`

In this command, **8** is the device number.

Drive Number

In addition to the device number, many disk devices also support a **drive number**. This allows a single device to access multiple disk mechanisms.

Examples:

- The Commodore 1541 has only one drive and it is permanently **drive 0**.

Color 64 BBS Manual Version 8.1A March 2026

- Dual-drive units such as the MSD-SD2 allow selection of multiple drives.
- On Lt. Kernal hard drives, the drive number is commonly referred to as the **LU (Logical Unit)**.

Example: `LOAD "0:FILE",8`

Here the **0** indicates drive 0, which appears before the filename and is separated by a colon.

Secondary Address

To communicate with a disk drive, the computer must open a communications *channel*. When using the **OPEN** command, this channel number is called the **secondary address**.

Important channel assignments include:

- **0–1** — Reserved for program loading and saving
- **15** — Command channel
- **2–127** — Available for normal file access

When multiple files are opened on the same device, each file must use a unique secondary address. This allows the disk drive to determine which file the computer is referencing.

The **command channel (15)** is used to send commands to the disk drive, such as:

- Initialize
- Validate
- Scratch
- Copy

Some devices also support additional commands. For example, the Lt. Kernal drive supports the **LG** command for retrieving hard drive information.

When opening the command channel, the “filename” parameter is interpreted as a command string rather than a file name.

Logical File Number

The **logical file number** is used internally by BASIC to track open files. After opening a file, this number is used with the **PRINT#** and **INPUT#** statements.

This simplifies program code because it avoids repeatedly specifying the device and drive numbers when performing file operations.

The logical file number is strictly for BASIC’s internal tracking and does not affect the device itself. Because of this, the logical file number and secondary address are technically independent.

However, programmers often use the same value for both. For example: `OPEN 8,DV,8,DR$+FI$`

File Name

The final parameter is the **file name**. Each file on a disk has a unique name, up to **16 characters** long.

In the **OPEN** command, the drive number is included in the filename string: `"0:FILENAME"`

The format is: `<drive number>:<file name>`

Color 64 BBS Manual Version 8.1A March 2026

Hard drive systems such as CMD HD or Lt. Kernal provide additional functionality. Many hard drives divide storage into **partitions**. In some systems, partitions are accessed through drive numbers; in others, special drive commands must be used to select the active partition.

Using SD2IEC

Color 64 BBS v8.1 and v8.10a operate reliably using only an **SD2IEC** drive or in combination with traditional Commodore disk drives.

Using SD2IEC offers several advantages:

- Significantly more available disk space
- Larger storage for messages and uploads
- Removal of traditional directory file limits

Successful SD2IEC operation depends on using the correct **device initialization commands** when assigning drives in the **+SETUP** program.

While discussions on the internet sometimes mention REL file problems with SD2IEC, no issues were encountered during testing. This may be related to running the system directly from the SD card's folder structure rather than using disk image files such as `.D64` or `.D81`.

An example SD2IEC folder configuration is shown below.

Table 54 – SD2IEC Folder Setup Example

Folder name on SD card	Functional area (or "Drive selection")	Settings in +SETUP for Drive Assignment
Nothing (root folder)	Programs & Systems drive	Drive: 8 Device Init: 0:!cd//
HELP	Help & Text files	Drive: 8 Device Init: 0:!cd//help
PRIMSGS	Private Messages Drive	Drive: 8 Device Init: 0:!cd//primsgs
PUBMSGs	Public Messages Drive	Drive: 8 Device Init: 0:!cd//pubmsgs
AUX1	Drive for AUX 1	Drive: 8 Device Init: 0:!cd//aux1
AUX2	Drive for AUX 2	Drive: 8 Device Init: 0:!cd//aux2
AUX3	Drive for AUX3	Drive: 8 Device Init: 0:!cd//aux3
UPLOADS	Uploads drive	Drive: 8 Device Init: 0:!cd//uploads
GAMES	Games Download folder	Drive: 8 Device Init: 0:!cd//games

Color 64 BBS Manual Version 8.1A March 2026

Folder name on SD card	Functional area (or “Drive selection”)	Settings in +SETUP for Drive Assignment
USERS	User Profile area (see section G.3.7 for information on this – applies to version 8.1a only)	Drive: 8 Device Init: 0:lcd//users

Additional directories may be used depending on your system configuration, but the same approach applies.

When using SD2IEC, the same drive selection methods work with other Color 64 features such as:

- DOS Wedge
- Message Editor
- Any routine that prompts for device and drive numbers

One minor issue to be aware of: SD2IEC does not support the Commodore **VALIDATE** command used during the BBS midnight maintenance routine. As a result, two syntax errors may appear in the system log at midnight. These errors do not affect BBS operation and can safely be ignored.

LT Kernal or CMD HD

No special configuration is required specifically for Lt. Kernal or CMD hard drives beyond normal drive assignments. However, these systems make extensive use of initialization commands and partition selection.

For this reason, the next section (*Drive Initialization Commands*) is particularly important when configuring these devices.

Drive Initialization Commands

This section summarizes drive initialization commands that work well with a variety of Commodore-compatible storage devices.

A **drive initialization command** is a command string sent to the disk device that prepares it for access. These commands are also commonly used to select partitions, logical units (LU), users, or subdirectories depending on the device being used.

Some devices require more than one command. Multiple commands can be chained together using the exclamation point (!) character.

Since the **initialize** command is supported by virtually all Commodore-compatible drives, it is commonly included in initialization strings. The **i** command may also be used to change the drive number used internally by Color 64.

For example, to select drive number 13 (such as on a CMD HD), the following command could be used: **i13**

This updates the variable **DR\$** so that Color 64 knows the current drive number in use.

Note that if an **i** command is used in a drive initialization string, it will override the drive number (0 or 1) specified in **+SETUP**.

Also note that the **i** command does *not* change the drive number for initialization commands used in the Bootmaker utilities (**bootmaker**, **bm ram**, and **bm small**).

Common initialization commands:

- **1541 or compatible drives**

```
Use: i0
```

- **SFD-1001, CBM 2031, CBM 9060, CBM 9090, or other single-drive devices**

```
Use: i0
```

- **MSD-SD2, CBM 4040, CBM 8050, CBM 8250, or other dual-drive devices**

```
Use i0 for drive 0 or i1 for drive 1.
```

1571 or Compatible Disk Drives

The Commodore 1571 can operate in three modes:

- 1541 single-sided compatibility mode
- 1571 double-capacity mode
- 1571 dual-side mode

If operating the 1571 in standard 1541 compatibility mode (single-sided, ~664 blocks), the initialization command: **i0** works the same as with a 1541 drive.

Color 64 BBS Manual Version 8.1A March 2026

However, because the 1571 is a double-sided drive, it can store:

- **1328 blocks** in a single directory in full 1571 mode, or
- **664 blocks per side** using two separate directories.

The second configuration is generally recommended because it provides two directories of 144 files each and has historically proven to be more reliable.

Initialization commands:

- Single directory (1328 blocks): `u0>m1!i0`
- Side 0 (664 blocks): `u0>h0!i0`
- Side 1 (664 blocks): `u0>h1!i0`

When using these modes, ensure the diskette is formatted properly.

You can format disks using the DOS wedge in JiffyDOS, the Message Editor, or from within the BBS.

Examples:

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 partition** on a CMD HD automatically behaves as the double-sided 1328-block format and will not accept these mode commands.

If you need a single-side layout, use a **1541 emulation partition** instead.

1581 Disk Drives

The Commodore 1581 is a double-sided 3.5" disk drive, so both disk sides are always used automatically.

The drive also supports **partitions**, although they are optional.

If you are not using partitions, a simple initialization command is sufficient: `i0`

If partitions are used, they must first be created using either:

- the 1581 drive utilities, or
- a partition management utility program.

To select a partition: `i0!/0:PARTITION NAME`

Where:

- `i0` initializes the drive
- `/` ensures the root directory is selected
- `/0:PARTITION NAME` selects the partition

The standalone `/` command can often be omitted because `i0` usually returns to the root directory automatically. If directory errors occur, restoring the extra `/` command may resolve them.

Nested partitions can be selected by adding additional commands: `!/0:PARTITION NAME`

Selecting the root directory again can be done with either: `i0!/` or `i0`

Color 64 BBS Manual Version 8.1A March 2026

The stock Commodore 1581 firmware contains several bugs that can cause reliability problems in BBS environments. For this reason, it is recommended that critical system files (such as overlays) not be stored on a stock 1581 drive.

If JiffyDOS is installed in the 1581, these firmware issues are corrected and the drive can be used more safely.

A **1581 emulation partition** on a CMD HD behaves exactly like a 1581 drive but without the firmware bugs.

CMD Hard Drive and CMD RamLink

When using CMD storage devices, the first step is selecting the appropriate partition.

This is done using the `cp` and `i` commands.

Example: `cp2!i2`

This selects **partition 2** and updates the active drive number accordingly.

This procedure works for all partition types.

For **1541 or 1571 emulation partitions**, no additional commands are normally required.

For **1581 emulation partitions**, you may also need to select the directory using:

`/O:PARTITION NAME`

For **native mode partitions**, subdirectories can be used. These are selected with the `cd` command.

Example: `cp5!i5!cd//games`

This selects:

- partition 5
- drive number 5
- the `games` subdirectory

The `i5` portion ensures that Color 64 correctly updates its internal drive number.

CMD FD series drives

At the time of writing, the CMD FD series drives had not been fully tested with Color 64. However, because these drives are designed to be compatible with standard 1581 disks, they should accept the same initialization and partition commands used for the 1581.

Consult your drive manual for additional device-specific commands.

Lt. Kernal Hard Drive

The Lt. Kernal hard drive organizes storage into:

- **LU (Logical Units)** — comparable to partitions
- **USER areas** — comparable to subdirectories within an LU

LU numbers range from **0–9**. USER numbers range from **0–15** and are expressed in **hexadecimal**:

- A = 10
- B = 11
- C = 12

Color 64 BBS Manual Version 8.1A March 2026

- D = 13
- E = 14
- F = 15

The Lt. Kernal command used to select LU and USER is: `l<device><LU><USER>`

Example:

To select LU 2, USER 11: `l82bli2`

To select LU 0, USER 5: `l805li0`

The additional `i` command is required so Color 64 updates the drive number internally.

Testing with Color 64 v8.1 / v8.1a confirmed that proper LU selection is essential for reliable disk access.

When configuring the drive in **+SETUP**, choose device **8** and enter the initialization string for the desired LU and USER.

Example configurations:

Table 55 – LT Kernal Setup Examples

LU 0 / User 0	LU 1 / User 0
<pre>Device (8..30) [8] > Drive!Init [0:!!1800!i0] >■</pre>	<pre>Device (8..30) [8] > Drive!Init [1:!!1810!i1] >■</pre>
LU 2 / User 3	LU 2 / User 10
<pre>Device (8..30) [8] > Drive!Init [2:!!1823!i2] >■</pre>	<pre>F:Device (8..15) [8] > F:Drive!Init [2:!!181a!i2] >■</pre>

ICT Hard Drive

The ICT hard drive system contains both:

- a built-in floppy disk drive
- a hard drive partition system

Do **not** use the `i` command in ICT initialization strings, as this can confuse the device.

Instead, configure the drive number as **0** in **+SETUP**.

Common commands:

- `h0` — Select built-in floppy drive
- `hN` — Select hard drive partition N

Example: `h2` selects partition 2.

Color 64 BBS Manual Version 8.1A March 2026

The command: `hm4` enables partition chaining.

Example: `hm4 5 7` chains partitions 5 through 7 together.

If partition chaining is used, special ICT merges must be installed in the BBS. See the section covering ICT HD integration with Color 64 for additional details.

Ram Expansion Unit

Although the Commodore 17xx series REU is not technically a disk device, it can emulate disk-like behavior for storage purposes.

A simple initialization command is usually sufficient: `i0`

Miscellaneous Programming Notes

Rainbow Mode

If an **F1** character is placed inside a quoted text string that is being sent to the user, Rainbow Mode will be enabled. In this mode, characters are displayed using the system color sequence defined in SETUP.

To disable Rainbow Mode, place an **F3** character inside the quoted text.

Example:

```
"{F1}Rainbow text here{F3}"
```

Color Sequencing

If an **F5** character is placed inside a quoted text string, the cursor will advance to the *next color* in the color sequence defined in SETUP. This allows color cycling within printed output without explicitly specifying each color.

Border Color

If an **F7** character is placed inside a quoted text string, the border and screen background will switch to black.

To restore the normal background color and uppercase mode, execute:

```
GOSUB 13680
```

Sequential File Display

To display a sequential file located in the *System Files* area:

```
F$="filename":GOSUB 205
```

This routine opens the file, sends its contents to both the screen and modem, and then closes the file.

Data Input from File

Sequential files used for data storage can be read using standard BASIC file input commands.

Example:

```
INPUT#
```

Although machine language routines exist to perform similar functions, they are not required for most applications. Standard BASIC file I/O generally performs adequately for typical BBS modules.

Capture User Input

Color 64 BBS Manual Version 8.1A March 2026

Color 64 provides several routines for capturing user input.

- To input a full line typed by the user:

```
GOSUB 310
```

The result is stored in the variable **I\$**.

- To read a single character from the modem or console:

```
GOSUB 110
```

If a character was typed, it will be stored in **A\$**.

- To prompt for a Yes/No response:

```
GOSUB 1010
```

The result (**Y** or **N**) will be returned in **A\$**.

- To display the standard "Are you sure?" confirmation prompt:

```
GOSUB 1005
```

Caller Log

To store information in the caller log, use one of the following methods:

```
A$="info to save":GOSUB 8004
```

or

```
I$="info to save":GOSUB 8003
```

These routines append the specified information to the caller log maintained by the BBS.

Carrier Lost Check

After user input or sequential file reads, you can test the variable **P** to determine if the connection has been interrupted.

Possible values of **P** include:

- **P=255** – Carrier lost or timeout occurred
- **P=1** – Caller pressed **CTRL/P** to abort

It is recommended that programs include the following line after input operations:

```
IF P THEN RETURN
```

This ensures the routine exits cleanly if the caller disconnects or aborts the operation.

Disk Error Check

To check the disk error channel after accessing a disk drive:

```
GOSUB 510
```

If an error occurred, the message will automatically be printed to both the local screen and the caller's terminal.

The following variables will contain the error information:

- **ER** – Error number
- **ER\$** – Error description
- **ET** – Error track
- **ES** – Error sector

Some disk status codes are not automatically displayed and must be handled manually if needed:

- 62
- 63
- 64
- 73

If a *disk full* condition occurs, the routine will automatically attempt to validate the disk.

Select System Files Drive

To select the System Files drive:

```
GOSUB 481
```

Convert Input (I\$) to Numeric Format

To convert the string stored in **I\$** into a numeric value:

```
GOSUB 610
```

The resulting numeric value will be stored in variable **I**.

If **I\$** contains non-numeric characters, the resulting value of **I** will be 0.

Spare Commands

Color 64 BBS Manual Version 8.1A March 2026

The Color 64 command system includes several "spare" command slots that can be assigned to custom programs or modules.

Refer to the following table to determine which program overlay and entry line are used by each spare command.

Table 56 - Spare Command Reference Chart

Command	Loads Prg	Executes Line		Command	Loads Prg	Executes Line
<i>Spare1</i> (8.1a – Games)	<i>Msgs</i>	13430		Spare6	Ov1	13465
<i>Spare2</i> (8.1a – User Profile)	<i>Xfer</i>	13440		Spare7	Ov1	13470
Spare3	Ov1	13450		Spare8	Ov1	13475
Spare4	Ov2	13455		Spare9	Ov1	13480
Spare5	Ov3	13460				

As shown above, each overlay has at least one spare command entry point. Spare commands **6 through 9** all branch to **vbbs.ov1**, which makes it convenient to place several smaller utilities in that overlay.

This design allows:

- vbbs.ov1 – multiple small utilities
- vbbs.ov2 / vbbs.ov3 – larger programs or menus

In Color 64 **v8.1a**, Spare Commands 1 and 2 are already assigned (can be modified, if desired):

- Spare 1 – Games Menu
- Spare 2 – User Profile Editor

If you wish to run multiple programs from a single spare command, a common approach is to implement a small menu program. For example, Spare 4 or Spare 5 can load a menu which then branches to several internal subroutines depending on the user's selection.

ADNs (Absolute Day Numbers)

The BBS system stores the current date in the numeric variable **DA** using a format known as an *Absolute Day Number* (ADN).

An ADN represents the total number of days that have elapsed since **January 1, 1800 (01/01/1800)**. While this representation may seem unusual at first, it greatly simplifies date calculations. The Color 64 machine language routines perform all required conversions automatically, so most programs never need to manipulate the ADN value directly.

Under normal BBS operation, users never see ADN values. However, sysops who write programs or modules should understand how this system works.

Machine language functions **@15 through @19** are used internally to perform ADN calculations and conversions. For detailed information about these functions, refer to the section "*ML Function Summary*".

Date Handling in the Time Routine

The *current time* routine located at line **1110** of each overlay also manages the system date. When midnight occurs, this routine increments the variable **DA** by one day. The updated value of **DA** is then used to calculate the other date-related variables.

Anytime you want to use one of the information variables listed below, you must perform the GOSUB to 1110.

The following variables are derived from **DA**:

Table 57 – DA Derived Variables

Value	Description
DA\$	Current date in the format MM/DD/YYYY
D1	Current month (1–12)
D2	Day of week (0–6, where 0 = Sunday and 6 = Saturday)

The formatted date string **DA\$** is generated automatically from the ADN value.

Month and Day Name Arrays

Two string arrays are provided for converting dates into readable text:

Table 58 – Date Text Derived Variables

Array	Contents
D1\$()	Month names, where D1\$(1) through D1\$(12) correspond to January through December
D2\$()	Day names, where D2\$(0) through D2\$(6) correspond to Sunday through Saturday

Color 64 BBS Manual Version 8.1A March 2026

These arrays allow programs to display dates in plain English format.

Other ADN-Based Variables

Several additional variables store dates internally using ADN format:

Table 59 – ADN Derived Variables

Variable	Description
ED	Caller's membership expiration date
LD	Caller's last call date
BD	Caller's date of birth

When user information is read from the password file, the corresponding formatted string variables are also generated automatically:

- **ED\$** – Membership expiration date (formatted)
- **LD\$** – Last call date (formatted)
- **BD\$** – Date of birth (formatted)

Calculating Differences Between Dates

Because dates are stored as Absolute Day Numbers, calculating the number of days between two dates is simple.

To determine the number of days between two dates:

```
difference = higher ADN – lower ADN
```

The result is the exact number of days between the two dates.

Calculating Future Dates

To determine the date after a specific number of days, simply add the desired number of days to the current ADN value:

```
future_date = DA + number_of_days
```

The resulting ADN can then be converted back into formatted date variables using the standard system routines.

Calculating Age or Year Differences

Machine language function **@27** calculates the number of calendar years between two ADN values.

This function is commonly used to determine a caller's age from their stored date of birth:

```
@27(DA,BD)
```

In this example:

Color 64 BBS Manual Version 8.1A March 2026

- **DA** – Current date
- **BD** – Caller's date of birth

The function returns the number of full calendar years between the two dates.

Converting from 7.37

This section is intended for SYSOPs who want to convert older version 7.37 and Super ML merge-in code for use with version 8.0 and above. It is also relevant to SYSOPs who previously ran separate overlay modules through custom menus.

Beginning with Color 64 version 8.0, the traditional SYSC(x) and PEEK(C(x)) methods of interfacing with the ML were replaced. Those methods worked well when the system was simpler, but they were not flexible enough to support newer features such as SwiftLink compatibility. For this reason, BASIC was modified to interpret a new ML interface while the ML is active.

In the 8.x versions, the ML interface is divided into **commands** and **variables**.

The ML commands begin with a period (.). For example, the equivalent of the old SYSC(5) command is .01, which gets a line of disk input.

The ML variables can be used in expressions much like normal BASIC variables, but they are referenced differently. All ML variables begin with an exclamation point (!) followed by two digits. For example, the equivalent of PEEK(C(22)) is the ML variable !05 (graphics mode).

To assign values to ML variables, use a format similar to a POKE statement. For example, the equivalent of POKE(C(35)),9 would be: !14,9 which sets the file device number to 9.

The table below redefines the old C(X) references using their newer replacements. Note that several references are now obsolete and therefore have no direct substitution listed.

Also note that many of the old output-related calls such as SYSC(0) have been replaced by special one-character commands such as #.

Table 60- 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)	

Converting BASIC code Steps

When you load a 7.37 module or program, you will find old SYSC and PEEKC calls that must be converted for the program to work properly under 8.0 / 8.1. In most cases, this is a fairly straightforward process.

- **Step 1:** Determine the core source code for the module being converted. This is usually indicated by a GOSUB in line 5, such as GOSUB 30000. If the file is only a small modification to an existing overlay, this step, and possibly Step 2, may not apply.

```
remopen1,8,15,"!0":print#1,"s0:v2112.*
ain":close1:save"0:v2112.main",8
ifpeek(817)=223thenprint#15,"ui
gosub30000
```

In the example above, the line 5 GOSUB points to line 30000, so the "core code" most likely begins there. Occasionally, a few lines immediately before that point may also need to be copied over.

- **Step 2:** Delete all code except the core code. In this example, delete everything up to line 29999 and keep line 30000 and above.

SYSRES users: delete0-29999

LTK users: del 0-29999

- **Step 3:** Begin replacing SYSC and PEEKC statements using the conversion chart above.

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)	

SYSRES commands:

1. **changeC@sysC(0)@@#@**
2. **changeC@sysC(1)@@\$@**
3. **changeC@sysC(5)@@.01@**
4. **find@peekC(@**
(find instances and change accordingly)
example:
pokeC(23),0 would become...
!06,0
5. Make sure you tackled all SYSC statements by searching for SYSC(:**find@sysC(@**

While there may be others, SYS(0), SYS(1), SYS(5), and PEEKC(23) are the most common.

- **Step 4:** Save a temporary version of this core file

Step 5: Merge in Overlay 2 or the "XXX SMALL" overlay from 8.0 or 8.1 (whichever version applies)

SYSRES users: use the MERGE command `merge "xxx small"` or merge Overlay 2 as appropriate.

- **Step 6:** Update line 5 of the new file so that it jumps to the module's core code, as described in the section on merging modules.
- **Step 7:** Save the file and then groom or clean up any remaining issues.

From the example shown in Step 1, after the merge was completed, the first few lines looked like this:

```
rem save"0:v2112.main",8
goto30000
.19:ov=. :h=14:gosub460:↑dr$+"bbs.game
x",dv
```

Color 64 BBS Manual Version 8.1A March 2026

In that example, `GOSUB30000` was changed to `GOTO30000` because the game did not appear to exit cleanly and was leaving the GOSUB on the stack.

Once the program is functioning, you can often further refine the old 7.37 statements into cleaner 8.x syntax. In many cases:

`a$="something":#` becomes `#"something"`

`a$="something"+cr$+str$(ba)+na$+i$` becomes `#"something"cr$bana$i$`

`a$="Do you want something?":$` becomes `$"Do you want something?"`

However, if the value in `A$` is also being used for the caller log, such as when a nearby `GOSUB8004` is present, then that particular instance should probably not be refined in this way.

Also look for opportunities to simplify repetitive code. This is especially common in older games, where the same command sequences are repeated over and over.

Example:

```
30050 #"Okay! Let's do that":gosub31000:gosub32000:goto30000
30XXX .... other lines of stuff
30600 #"No? All right then":gosub31000:gosub32000:goto30000
```

For SYSRES users, you can find repeated sequences using:

```
find@gosub31000:gosub32000:goto30000@
```

You can then consolidate them into a subroutine:

```
30500 #"Okay! Let's do that":goto50000
30XXX .... other lines of stuff
30600 #"No? All right then":goto50000
50000 gosub31000:gosub32000:goto30000
```

For SYSRES users, this can be made even easier with:

```
change@gosub31000:gosub32000:goto30000@@goto50000@
```

**** Be aware that this replacement would also affect line 50000 itself. One workaround is to temporarily alter that line so it will not qualify during the global change. For example, you might change:**

```
50000 gosubb31000:gosub32000....
```

and then change it back after the replacements are completed.

Another important point during 7.37 conversions is to make sure all file calls are going to the correct locations. Games that use REL files, SEQ files, or other support files often expect them to be in the original Systems file area. If you place them elsewhere, the program must be updated accordingly.

Examples of these kind of data calls are shown below:

Color 64 BBS Manual Version 8.1A March 2026

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

And....

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)

For example, if your games are stored in AUX 3, but the old 7.37 game expects the Systems drive, then in the **core code only** you might change:

- all instances of `GOSUB202` to `GOSUB203`
- all instances of `GOSUB481` to `H=14:GOSUB460`

This redirects access to AUX3.

Be careful to make these changes only in the module's core code. Do *not* alter the standard overlay support lines unless you fully understand the consequences.

Changes from 7.37 to 8.X to note

New Carrier Detect Test: One significant change is the way the BBS detects whether carrier is still present. The older method looked like this:

```
ifcd=(peek(56577)and16)thenprint"carrier lost!"
ifcd<>(peek(56577)and16)thenprint"carrier present"
```

The new method replaces these direct hardware checks with ML variable `I25`, which is updated with carrier status every 1/60th of a second. This was necessary because the standard RS-232 routines and the SwiftLink RS-232 routines detect carrier differently, so a simple `PEEK` no longer works reliably across both methods.

The new method is:

```
if!25=0thenprint"carrier lost!"
if!25<>0thenprint"carrier present"
```

This new method is both shorter and easier to remember.

- New Modem Output Command**

Color 64 BBS Manual Version 8.1A March 2026

Another change affects the way output is sent directly to the modem. In older versions, you would use:

```
print#5, "atdt";p$
```

In the newer method, this becomes: `"atdt";p$`

This new command is shorter and is required because file number 5 is no longer opened during BBS operation. This change is part of maintaining compatibility with the SwiftLink routines.

The New Modem Input Function: The method for reading characters directly from the modem has also changed. The old format was: `get#5, a$`

The new method uses one of the built-in Color 64 ML functions: `a$=@4`

The `@4` function reads 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.

The Automatic Module Converter: Included with the Color 64 package is a program called "**m-con 50000**", an ML program that can be used to convert BASIC code from Color 64 version 7.37 and Super ML. It was designed primarily to convert small optional merge-ins, although with some programming skill it can also help convert games and utility modules.

To use it, first load it into memory with: `LOAD"m-con*", 8, 1`

Then issue a `NEW` command to ensure BASIC memory is clear.

Once **m-con** is resident, load the BASIC program you wish to convert and type: `SYS50000`

to begin the conversion. You do not need to reload **m-con** as long as the computer remains on and no program overwrites the memory area where it resides.

M-con will convert:

- `SYSC (x)` references
- `POKEC (x)` references
- `PEEK (C (x))` references
- `PRINT#5` modem-output statements
- Super ML MCI commands
- CTRL/O characters into version 8.0 format

Things that **m-con** cannot do include:

- Converting references to the old carrier-detect test. These must be changed manually.
- Converting `GET#5` statements that read directly from the modem. These must also be changed manually.

This conversion program should *not* be used to convert a full overlay, especially one of the main overlays. Those must be replaced entirely.

If you are converting separate overlay-based modules or games, there are additional requirements. The routines in lines 0 to 27000 are the standard core subroutines of a Color 64 overlay, and many of them were revised for the newer versions, especially the caller log routine.

To convert older overlays that still contain these lines, strip those lines out carefully, making sure not to remove any of the actual module code. Then merge in the appropriate "**xxx**" skeleton overlay after running **m-con** on the remaining code.

For most games and small modules, the "**xxx small**" overlay is ideal because it is compact while still providing the required support routines.

You should also consult the programming section for more information about these skeleton overlays and how they are intended to be used.

SYSOPs who used earlier versions of the add-drive mod should note that the **H** setting for the three auxiliary file groups is now **12, 13, and 14**.

This means that to select AUX 1, the new method is: `H=12:GOSUB460`

If you were using the old add-drive mod, you may need to update overlays that selected AUX 1 with `H=11`, because Network now uses that value to select the Network drive.

One practical way to convert to the new system is:

- Change your SETUP definitions so that what was formerly AUX 2 and AUX 3 now become AUX 1 and AUX 2
- Change all module references of `H=11` to `H=14`
- Configure AUX 3 in SETUP to hold what was formerly stored in AUX 1

Note: The **m-con** program converts all references to CTRL/A (the Super ML MCI character) inside quoted strings into the English pound sign (the version 8.0 MCI character, £), *except* when CTRL/A appears inside a `PRINT#15` statement.

This exception exists because programs that use relative files often use CTRL/A as part of the relative-file position command sent to file number 15.

If you are converting a program that uses relative files, it is wise to first inspect all CTRL/A references to determine whether any of them are part of a relative-file command. If so, change those references to `CHR$(1)` before running the converter so they will be protected during conversion.

7.37 BBS Convert

Note: The documentation below comes from the original 8.0 manual. It's important to note that there is no formal testing on converting 7.37 to 8.10a, though I have proven converting an 8.0 to 8.1 (strict file replacement) and 8.1 to 8.10a.

It may be prudent to:

- 1) follow the below instructions to get to 8.0
- 2) THEN follow 8.10a instructions for upgrade.

Once you are finished converting all the necessary files, read the "MF Convert" section on converting your vmod file if you are planning to use it for 8.10a. Otherwise, you can proceed to the Converting 8.0 or 8.1 to 8.10a section.

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 vbbs.parms file present for you to be able to use this conversion utility. If you are going to convert your original vbbs.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 vbbs.parms file.

The program will first check to see if your vbbs.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

Color 64 BBS Manual Version 8.1A March 2026

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 "V*" 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 Vvariables 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.

7.37 MF Convert

Note: The documentation below comes from the original 8.0 manual. It's important to note that there is no formal testing on converting 7.37 to 8.10a, though I have proven converting an 8.0 to 8.1 (strict file replacement) and 8.1 to 8.10a.

It may be prudent to:

- 1) follow the below instructions to get to 8.0
- 2) THEN follow 8.10a instructions for upgrade.

Once you are finished converting all the necessary files, read the "MF Convert" section on converting your vmod file if you are planning to use it for 8.10a. Otherwise, you can proceed to the Converting 8.0 or 8.1 to 8.10a section.

The included program called "mf convert" can be used to convert earlier version "Vmod 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.

Before running the program, you should make sure that you have plenty of room on the drive where your "Vmod 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.

Converting from 8.0 or 8.1

Version 8.1a will operate correctly on a functioning v8.1 system, since no changes were made to the **vbbs.parms** or **vpassword** files.

Back up your system before performing any of the following steps.

The following actions must be completed:

- **Replace the 8100 files with the new 810a files.** **vbbs.init 810a**

Be sure the original 8100 files (or 8000 files, where noted) are removed. If older versions remain in place, normal BBS operations may still load them. All of the following files belong in the PROGRAMS area:

- **vbbs.msgs 810a**
- **vbbs.xfer 810a**
- **vbbs.nw1 810a**
- **vbbs.nw2 810a**
- **vbbs.ovl 810a**
- **vbbs.ov2 810a** (*optional*)
- **vbbs.ov3 810a**
- **vsys.edit 810a** (*replaces vsys.edit 8000*)
- **vsys.net 810a**
- **dir tools 810a** (*replaces dir tools 8000*)
- **pswd tools 810a** (*replaces pswd tools 8000*)
- **xxx msg 810a** (*v8.1 systems will not already have this file, so if present, replace xxx msg 8000 instead*)

Add the following files:

- **vbbs.profile** (*place in the PROGRAMS drive*)
- **vbbs.games 810a** (*place in the AUX3 drive unless you plan to continue using the Mod Menu for games*)

Modify the following file as needed:

- **vsys.remove** (*REU users only*)

Verify that +SETUP defines the following:

- AUX1 disk parameters needed to support User Profile operations
- AUX3 disk parameters needed to support Games, **if** you are not continuing to use a pre-existing Mod Menu setup from an earlier BBS configuration

Finally, review the "Prior to Running" section to make sure all remaining changes required for 8.10a have been completed.

Undocumented Commands Research Project

At the time of this revision for Color 64 v8.1a, several ML-related commands and variables remain undocumented in the original 8.0 manuscript. In addition, a number of BASIC shortcuts appear to have been introduced or expanded as part of the 8.1 upgrade. Much of the information below was derived through reverse-engineering, primarily by comparing version 8.1 code to earlier 8.0 and 7.37 implementations.

The notes below summarize observations of how the code behaved in 8.0 or 7.37, along with the shortcut behavior and changes observed in 8.1.

Please note that this is a work in progress, and the online wiki (color64wiki.itchybutt.org) is updated frequently. Some observations may still be incomplete or tentative, as they remain under active investigation. Even so, understanding some of the modified BASIC can be relatively straightforward.

If the ML is loaded into your system (for example, by loading and running **+SHELL**), you can write a small test program of your own by calling the ML entry point in the first line:

```
10 SYS49923
20 #"Test of program"
30 ETC....
```

General Programming

Table 61 – BASIC Syntax Changes in 8.1 - Research

Topic	Observation / Conclusion	
Simple Addition	Color 64 will sometimes allow a shorthand form of numeric addition without requiring the full: var=var+value For example: A=A+12 can sometimes be written as: A+12 This appears to work only with numeric variables, not strings, and use has not been completely consistent in testing. Use with caution.	
Branch Shortcuts	8.0 Example 1: <pre>10100 print"ggInsert Params disk and press RETURNgg 10110 geta\$:ifa\$<>cr\$then10110</pre> Example 2: <pre>8220 geta\$:ifa\$=""ora\$="g"then8220 8230 return</pre> Example 3: <pre>11250 getH8,a\$,a\$,b\$,a\$:ifstthen11300 11260 a\$=05:ifasc(b\$+nu\$)>9thena\$=" "+a\$ 11265 ifmid\$(a\$,5,2)<>"✓"then11250 11270 mn(.)=mn(.)+1:mn(mn(.))=val(mid\$(a\$,7,6)) 11272 mf%(mn(.))=val(mid\$(a\$,15,5)) 11275 ca%(mn(.))=asc(mid\$(a\$,20))-192 11280 ifmn(.)>mmthenclose8-gosub11400:go to11210 11290 goto11250</pre> Example 4: <pre>11585 j=028(i\$,a):ifj>-1thena\$(j)=a\$:go to11585:z1\$="" 11590 next:48,m7:return</pre>	8.1 Example 1: <pre>10100 print"ggInsert Params disk and press RETURNgg 10110 [geta\$:@a\$<>cr\$</pre> Example 2: <pre>8220 [geta\$:@a\$=""ora\$="g":return</pre> Example 3: <pre>11250 [a=06(029(2,2)):ifstthen11300 11260 a\$=05:ifa>9thena\$=" "+a\$ 11265 ifmid\$(a\$,5,2)<>"✓" "e 11270 mn(.)+1:mn(mn(.))=val(mid\$(a\$,7,6)) 11272 mf%(mn(.))=val(mid\$(a\$,15,5)) 11275 ca%(mn(.))=asc(mid\$(a\$,20))-192 11280 cmn(.)<=mm:close8:a=mn(1):fori=2to mn(.):ifmn(i)<athena=mn(i)</pre> Example 4: <pre>11585 [j=028(i\$,a):ifj<.1@a\$(j)=a\$:0 11590 next:48,cm%(16,.):return</pre>
<p><u>Conclusion:</u> [and @ are used as loop shortcuts, with [marking the start point and @ providing the condition that causes execution to return to the most recent loop anchor.</p> <p>Also note, in line 8220 (Example 2), how the RETURN statement moved from line 8230 into the 8220 loop in version 8.1, indicating that the return executes when A\$<>"" and A\$<>HOME.</p> <ul style="list-style-type: none"> [starts a loop (the anchor) 		



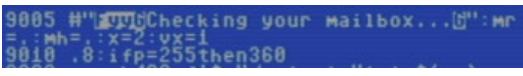
Color 64 BBS Manual Version 8.1A March 2026

Topic	Observation / Conclusion	
	<ul style="list-style-type: none"> @ is the loop condition and jumps back to the last anchor @[x] jumps back if x is non-zero] is the Color 64 BASIC conditional action delimiter; it separates the IF test from the statement list executed when the test is true <p>This is heavily used with status handling:</p> <ul style="list-style-type: none"> @[SR] means: if SR<>0, jump back to the last [anchor. If SR=0, execution falls through and the loop ends naturally. <p>So you will see numerous patterns like this:</p> <p>perform read / set SR=ST then use @[SR] to continue while status says "keep going"</p>	
"THEN" not required for "IF" stmt	8.0	8.1
	<pre>8160 gosub1005:#:ifa\$<"Y"thenreturn</pre> <p>Conclusion: THEN is not required in many cases. A form like: IF <condition> <action> is valid in most observed uses.</p>	<pre>8160 gosub1005:#:ifa\$<"Y"return</pre>
IF stmt and "(else) alternate use	8.0	8.1
	<pre>3341 iflv=>cm%(2,2)andlv=>cm%(29,2)andlv<=>m)mandfr<>1thena\$=a\$+"@ (P)ivate "</pre> <p>Observation: The 8.1 authors often reversed the comparison order in variable checks while preserving the same end-result.</p> <p>Conclusion: This appears to serve as a not function.</p> <p>Above Example meaning: "if LV is not less than CM%(2,2), and LV is not less than CM%(29,2), and FR does not equal 1, then print (P)ivate."</p> <p>Another example:</p> <pre>10 sys49923 20 a=20 30 ifa>19&print"not greater than 19":got o40 35 print"greater than 19" 40 end 30: d#00 1u00 user00 port#00 0000 run greater than 19</pre>	<pre>3341 iflv<cm%(2,2)&iflv<cm%(29,2)&iffr=1 &)"@ (P)ivate "</pre>
@	Example (bbs.msgs)	
	8.0	8.1
	<pre>3730 iflv<cm%(29,2)then3715:z#:#:fr\$=""su s="private":goto12824 3735 iflv<cm%(.,2)then3715:z#:#:goto9805</pre> <p>Conclusion: This seems to act somewhat like an IF keyword in certain contexts, although testing this idea in a traditional standalone sense resulted in a syntax error.</p>	<pre>3730 @lv<cm%(29,2):#:#:su\$="private":goto 12824 3735 @lv<cm%(.,2):#:#:goto9805</pre>
[var\$]	Example (bbs.msgs)	
	8.0	8.1
	<pre>4525 \$"TitleSubject > ":gosub310:ifi\$=""th enreturn</pre> <p>Conclusion: This can serve as an empty-variable or null-string check.</p>	<pre>4525 \$"TitleSubject > ":gosub310:if[i\$]ret urn</pre>

Color 64 BBS Manual Version 8.1A March 2026

ML Commands

Table 62 – ML Command Changes in 8.1 - Research

Topic	Observation / Conclusion							
.8	.8 - Session/Input Poll Previously documented in 8.0 as the “equivalent of the % command,” this appears to have been rewritten in version 8.1. It is especially noticeable in <code>bbs.msgs</code> , where comparison shows that <code>.8</code> has replaced the <code>GOSUB110</code> routine, which in 7.37 called <code>SYSC(4)</code> or, in 8.0, <code>.00</code> .							
	7.37  Note that the above section calls routine at line 110, shown below: 	8.1 At same line location in 8.1, you can see there is no call to the routine at 110... 						
	Polls for typed input in a non-blocking manner, whether local or remote, normalizes transient input-status states, and refreshes session status. It is typically used inside scrolling output or file-display loops and before slow operations. Observed side effects include updating <code>P</code> : <ul style="list-style-type: none"><code>0</code> = OK<code>1</code> = user abort<code>255</code> = carrier lost / timeout It may also update the same underlying status and character registers used by <code>.00</code> . While further testing is still required, this appears to be the practical replacement for <code>GOSUB110</code> .							
.9	Previously documented in 8.0 as a “do not use” command that replaced the <code>&</code> function, this appears to have been repurposed in 8.1 for single-key command input, replacing the traditional <code>GOSUB110</code> routine. .9 – Single-Key Command Input Poll Overview The <code>.9</code> command polls for a single keypress and stores the result in the BASIC string variable <code>A\$</code> . It is used at the command prompt to capture one-character command input from either the local keyboard or the remote modem connection. Previous versions of Color 64 used <code>GOSUB110</code> in the same areas. When a key is pressed: <ul style="list-style-type: none"><code>A\$</code> is set to the key pressedThe character is stored in high-bit PETSCII format Example: <ul style="list-style-type: none">Pressing "X" gives <code>ASC(A\$)=216</code>Pressing "O" gives <code>ASC(A\$)=207</code>(standard ASCII value + 128) When no key is pressed: <ul style="list-style-type: none"><code>A\$=""</code>The command does not block execution Examples (<code>bbs.msgs</code>) <table><tr><th>Usage</th><th>Example</th></tr><tr><td>Bare Poll</td><td>110 .9:RETURN 995 .9:IF A\$<>NU\$ OR P=1 THEN 995:RETURN (loop until a key appears or abort)</td></tr><tr><td>Filtered Keys</td><td>1020 .9,"YN":RETURN</td></tr></table>		Usage	Example	Bare Poll	110 .9:RETURN 995 .9:IF A\$<>NU\$ OR P=1 THEN 995:RETURN (loop until a key appears or abort)	Filtered Keys	1020 .9,"YN":RETURN
Usage	Example							
Bare Poll	110 .9:RETURN 995 .9:IF A\$<>NU\$ OR P=1 THEN 995:RETURN (loop until a key appears or abort)							
Filtered Keys	1020 .9,"YN":RETURN							

Topic	Observation / Conclusion				
	<div> <div>3715 .9,"12345"+CR\$:IF P ... (menu choices plus Enter)</div> <div>Space-as-Abort Convention 9132 .9:IF A\$=" " THEN P=1 (and several more spots do the same)</div> </div> <p>So .9 is essentially the modern ML command that replaced the old SYSC(4) “get typed character” behavior, with an added feature: it can be instructed to accept only a defined set of characters.</p>				
.31 (& !55)	<div> <div>Example: (bbs.msgs)</div> <div> <div>8.0</div> <div> <pre> 9041 H"Scanning mail... 9042 .01:sp=st:ifxthen#0:x=x-1 9043 if#0=" " thena=a+1:x=2:H" 9044 ifsr=" " then9042 9045 close# print#15,"s"+dr\$+"temp":ifl v<cw%(29,2)then9048 </pre> </div> <div>8.1</div> <div> <pre> 9041 H"Scanning mail... 9042 [.31:sr=st:ifxthen#0:x- 9043 if!55thena+ :x=2:H" 9044 e[sr] </pre> </div> </div> <p>Conclusion: .31 – Message-Aware Disk Input (Version 8.1)</p> <p>Earlier documentation listed ML command .31 as “not used.” Analysis of version 8.1 confirms this is incorrect. In Color 64 v8.1, the .31 command is actively used within message-handling overlays, especially BBS.MSGS, and functions as a specialized disk-input routine. It replaces the earlier .01 disk-input command used in version 8.0 when scanning and reading private messages.</p> <p>Comparison: Version 8.0 vs Version 8.1</p> <table border="1"> <thead> <tr> <th>8.0 Details</th><th>8.1 Details</th></tr> </thead> <tbody> <tr> <td> <p>Message scanning used: .01.</p> <p>After each read, BASIC manually checked for a message delimiter: IF @0 = CHR\$(14) THEN ...</p> <p>The byte {0E} (CHR\$(14)) acted as the message record separator. When this character was encountered, the system incremented or decremented message counters, control flow moved to the next message record, and the loop continued scanning. Delimiter detection was handled explicitly in BASIC.</p> </td><td> <p>In version 8.1, .01 is replaced in message routines by .31.</p> <p>The .31 command performs the disk input operation and internally detects the message delimiter. Instead of BASIC checking @0 = CHR\$(14), version 8.1 uses the ML variable IF !55 THEN When !55 becomes non-zero, the system has reached the end of the current message record, message counters are updated, and control moves to the next message or exits the loop. Delimiter detection is now handled inside the ML routine rather than in BASIC.</p> <p><u>!55 – Message Record Delimiter Flag</u> !55 is an ML variable introduced in version 8.1 that serves as a message record boundary flag. It is set automatically by .31 when the message delimiter byte, previously tested as CHR\$(14) in version 8.0, is encountered.</p> <p>!55 = 0 → No delimiter encountered !55 ≠ 0 → End of message record reached When !55 is detected:</p> <ul style="list-style-type: none"> the current message-scan loop updates counters the system transitions to the next message record display or processing routines advance accordingly </td></tr> </tbody> </table> </div>	8.0 Details	8.1 Details	<p>Message scanning used: .01.</p> <p>After each read, BASIC manually checked for a message delimiter: IF @0 = CHR\$(14) THEN ...</p> <p>The byte {0E} (CHR\$(14)) acted as the message record separator. When this character was encountered, the system incremented or decremented message counters, control flow moved to the next message record, and the loop continued scanning. Delimiter detection was handled explicitly in BASIC.</p>	<p>In version 8.1, .01 is replaced in message routines by .31.</p> <p>The .31 command performs the disk input operation and internally detects the message delimiter. Instead of BASIC checking @0 = CHR\$(14), version 8.1 uses the ML variable IF !55 THEN When !55 becomes non-zero, the system has reached the end of the current message record, message counters are updated, and control moves to the next message or exits the loop. Delimiter detection is now handled inside the ML routine rather than in BASIC.</p> <p><u>!55 – Message Record Delimiter Flag</u> !55 is an ML variable introduced in version 8.1 that serves as a message record boundary flag. It is set automatically by .31 when the message delimiter byte, previously tested as CHR\$(14) in version 8.0, is encountered.</p> <p>!55 = 0 → No delimiter encountered !55 ≠ 0 → End of message record reached When !55 is detected:</p> <ul style="list-style-type: none"> the current message-scan loop updates counters the system transitions to the next message record display or processing routines advance accordingly
8.0 Details	8.1 Details				
<p>Message scanning used: .01.</p> <p>After each read, BASIC manually checked for a message delimiter: IF @0 = CHR\$(14) THEN ...</p> <p>The byte {0E} (CHR\$(14)) acted as the message record separator. When this character was encountered, the system incremented or decremented message counters, control flow moved to the next message record, and the loop continued scanning. Delimiter detection was handled explicitly in BASIC.</p>	<p>In version 8.1, .01 is replaced in message routines by .31.</p> <p>The .31 command performs the disk input operation and internally detects the message delimiter. Instead of BASIC checking @0 = CHR\$(14), version 8.1 uses the ML variable IF !55 THEN When !55 becomes non-zero, the system has reached the end of the current message record, message counters are updated, and control moves to the next message or exits the loop. Delimiter detection is now handled inside the ML routine rather than in BASIC.</p> <p><u>!55 – Message Record Delimiter Flag</u> !55 is an ML variable introduced in version 8.1 that serves as a message record boundary flag. It is set automatically by .31 when the message delimiter byte, previously tested as CHR\$(14) in version 8.0, is encountered.</p> <p>!55 = 0 → No delimiter encountered !55 ≠ 0 → End of message record reached When !55 is detected:</p> <ul style="list-style-type: none"> the current message-scan loop updates counters the system transitions to the next message record display or processing routines advance accordingly 				
.36	<div> <div>Example (bbs.init)</div> <div> <div>8.0</div> <div> <pre> 13640 if!05then\$" return 13645 return 13650 Z" 24 </pre> </div> <div>8.1</div> <div> <pre> 13640 if!55" return 13645 if!55" 13650 Z" 24 </pre> </div> </div> <p>.36 – Prompt-Cycle Housekeeping (Behavior Unconfirmed)</p> <p>Overview The .36 command is called immediately after printing the command prompt and just before entering the main command-input polling loop (.9).</p> <p>In testing within the live BBS environment, removing .36 did not alter normal command-prompt behavior, local keyboard input, or remote input handling. This suggests .36 is either redundant in v8.1, reserved for edge cases, or included for compatibility with earlier ML builds.</p> <p>Observed Call Pattern</p> <ul style="list-style-type: none"> Prompt is printed (PETSCII-dependent) .36 executes unconditionally Control returns into the .9 command polling loop </div>				

Color 64 BBS Manual Version 8.1A March 2026

Topic	Observation / Conclusion										
	Testing Notes With .36 present: <ul style="list-style-type: none">Pressing a command key results in A\$ being set to high-bit PETSCIIP functions normally and remains 0 during command-prompt operations With .36 removed: <ul style="list-style-type: none">No visible change under normal BBS operation.9 continued to function normally Working Hypothesis .36 likely performs a defensive reset of internal prompt or input state, such as clearing a latch, normalizing editor mode, or handling rare typeahead or carrier-transition conditions. No definitive visible effect has been confirmed under standard test conditions.										
	Summary .36 is executed as part of the prompt cycle in v8.1 but does not appear necessary for normal command entry. Further testing under stress conditions, such as typeahead flooding, carrier transitions, or mode switching, may reveal its purpose.										
	Observed in bbs.msgs										
	8.0	8.1									
	(Current message vs high message)										
<pre>3550 ifm2>mn(.)thenreturn 3555 form2=m2tomn(.):ifmr%(m2)orca%(m2)< >ca thennext:return 3560 gosub485:fs="/" +str\$(mn(m2))+" *":g oto203</pre>	<pre>3550 ifm2>mn(.)return 3555 .38,0mr%,ica%:.39,km2:form2=m2tomn(.):if;0kor;1k<>ca then.40k:next:return 3560 gosub485:fs="/" +str\$(mn(m2))+" *":g oto203</pre>										
(Reading message)											
<pre>3130 gosub610:ifi<lori>mn(mn(.))thenretu rn 3140 form=mn(.):toistep-1:mr%(m)=.:ifmn(m)>i thennext:m=1 3200 lm=hm:mx=m:ifra thenra=m</pre>	<pre>3130 gosub610:ifi<lori>mn(mn(.))return 3140 m=mn(.):.38,0mn,lmr%:.39,fm,km:1;k =.:if;0f<:il&.41fk:m-:0m:m=1 3200 lm=hm:mx=m:ifra thenra=m</pre>										
(Storing message)											
<pre>1524 fori=1tomn(.):mn(i)=mn(i+1):1k%(i)= 1k%(i+1):1k%(i)=1k%(i)-1:mr%(i)=mr%(i+1) 1525 ca%(i)=ca%(i+1):mr%(i)=mr%(i+1):nex t:m=m-1:m2=m2-1:mx=mx-1 1526 ifra>1 thenra=ra-1</pre>	<pre>1524 fori=1tomn(.):if;11=i then;1k=ii&:1 k=;11-1 1525 ;8f=;0g;2k=;21;3k=;31;4k=;41:.40 fgk1:next 1526 ifra>1 thenra=</pre>										
Message Relink Commands in v8.1											
<table><tr><th>Command</th><th>Purpose</th></tr><tr><td>.38</td><td>Initialize / bind relink iterator operands</td></tr><tr><td>.39</td><td>Perform iterator scan step</td></tr><tr><td>.40</td><td>Advance iterator (skip/continue)</td></tr><tr><td>.41</td><td>Finalize relink process</td></tr></table>		Command	Purpose	.38	Initialize / bind relink iterator operands	.39	Perform iterator scan step	.40	Advance iterator (skip/continue)	.41	Finalize relink process
Command	Purpose										
.38	Initialize / bind relink iterator operands										
.39	Perform iterator scan step										
.40	Advance iterator (skip/continue)										
.41	Finalize relink process										
.38 Relink Iterator Initialization The .38 command was introduced in Color 64 v8.1 as part of the internal message relinking system and serves as the entry point for that process. It does not perform file I/O directly and does not read or write messages itself. Instead, it initializes and binds internal machine-language structures used by the relinking iterator.											
It is designed to be used as the first step in a command sequence: .38 → .39 → .40 / .41 This sequence replaces the more BASIC-driven record-scanning logic found in v8.0.											
What .38 Does <ul style="list-style-type: none">Binds specific BASIC variables or arrays to ML-side descriptorsRegisters operands for the relink iteratorInitializes scratch registers used by subsequent .39 operationsPrepares the ML environment for scanning and linking old message records In BBS.MSGS, this binding typically associates: Slot 0 → MN (message number) Slot 1 → MR% (message-read flags array)											
The ML iterator then uses internal scratch registers such as: ;OK, ;1K, and ;0F These are later examined by BASIC to determine loop control and branching.											
In v8.0, message scanning and linking logic was performed directly in BASIC, including character comparisons and explicit loop control.											

Color 64 BBS Manual Version 8.1A March 2026

Topic	Observation / Conclusion
	<p>In v8.1, that logic was moved into machine language for speed and structural cleanliness. BASIC now acts primarily as a decision layer while ML handles the iterator mechanics.</p> <p>Important Notes:</p> <ul style="list-style-type: none">• .38 must precede .39 in relink routines• .38 does not itself advance or scan records• It prepares the ML environment used by the iterator• It is not a general-purpose file command• It appears specific to the message relinking system

Color 64 BBS Manual Version 8.1A March 2026

ML Variables

Table 63 – ML Variable Changes in 8.1 - Research

Topic	Observation / Conclusion
!8	<p>Scratch Register</p> <p>!46 was originally documented in 8.0 as unused. However, the 8.1 version uses it in disk operations across all overlays at the disk routine lines 710-740 as well as the log routine lines 8003-8040.</p> <pre> 710 !46,!14:!14,88:!4,..:iff\$then730:endif 8thena=3e4:goto740 720 open88,dv,..,"\$"+dr\$+ml\$:gosub770:gos ub780:goto740 730 open88,dv,..,"\$"+dr\$+f\$+"*":gosub770: ifer\$gosub790 740 !4,13:!14,!46:close88:return </pre> <p>In those routines, !46 holds a temporary value. The lines around 710 use it to preserve a disk channel number, while the lines around 8000 use it as temporary variable storage.</p> <p>It can hold values in the range 0-255. Any other value results in an “illegal quantity” error.</p> <pre> !8 sys49923 20 !46,255 30 print !46 40 !46,256 50 print !46 run 255 ?illegal quantity .error in 40 </pre>
!55	<p>Message Record Delimiter Flag</p> <p>Used in conjunction with !57 and !60. When true, this appears to monitor for !57 in order to invoke command-state behavior. See !57 and !60 for more information.</p>
!56	<p>Message Editor Command State (True/False) (Editor)</p> <p>Used in conjunction with !57 and !60. When true, this appears to monitor for !57 in order to invoke command-state behavior. See !57 and !60 for more information.</p>
!57	<p>Prefix Trigger Assignment (Editor)</p> <p>This works in concert with !56 and !60 in the message-editor command logic. In the unmodified 8.1 code, the prefix trigger assignment is 47, which corresponds to /. This makes / the command prefix used in the message editor when !56 is true. The command value itself is stored in !60. See !56 and !60 for more information.</p>
!58	<p>Unknown – (Editor)</p> <p>Present in editor functions, specifically during text entry after pressing RETURN on a line, but its overall function remains unknown. It appears only in bbs.msgs in a subroutine beginning at line 310.</p>
!59	<p>Text-Entry Mode Flag (Editor)</p> <p>Present in editor functions. This appears to be set high by ML routines while in text-entry mode so code that deals with the command menu can be bypassed.</p>
!60	<p>Menu Selection Holder (Editor)</p> <p>Overview</p>

Color 64 BBS Manual Version 8.1A March 2026

Topic	Observation / Conclusion																					
	<p>In Color 64 v8.1, !60 acts as a state variable for the message editor command interface. It is used to control, and dispatch, the editor's slash-command / CMD> menu behavior, and appears to be treated as a small integer mode or selection value. This logic does not exist in the v8.0 editor flow, indicating it was introduced or substantially expanded in v8.1.</p> <p>Observed behavior in bbs.msgs (v8.1)</p> <p>The editor command-prompt section sets up a list of command keywords, reads user input, and then stores the selected command number into !60. Later code uses ON !60 GOTO to jump to the appropriate command handler.</p> <p>Key excerpt (v8.1, around 2930–2939):</p> <pre>2931 !60,.:II\$="divider,rap,elete,dit,end,merge":&"...CMD> ..." 2932 [.9,I\$: ... :IF!60THEN&@3(DE\$,LG) 2933 ...]2935:\!60,II: ... (stores selection into !60) 2936 IFII<11 !60,.:!56,-(II=9):!57,47 2937 I\$="":II\$="":RETURN</pre> <p>Important notes:</p> <ul style="list-style-type: none">!60,. clears or resets the state (sets it to 0)!60,II stores the selected menu item index (II) into !60The prompt offers 6 commands: divider, wrap, delete, edit, end, merge <p>Dispatcher usage</p> <p>Version 8.1 uses !60 as the selector in a computed jump: 1302 ON!60 GOTO 1314,1305,1306,1365,1310,1312:!60,1:@ This strongly indicates:</p> <ul style="list-style-type: none">!60 holds a value in the range 1–6 representing the selected commandAfter a handler runs, code commonly resets !60 back to 1 as the default state <p>Based on the command list order from line 2931, the implied mapping is:</p> <table><tr><th>!60 Value</th><th>Command</th><th>Notes</th></tr><tr><td>1</td><td>Divider</td><td>Insert or manage message divider</td></tr><tr><td>2</td><td>Wrap</td><td>Wordwrap toggle / behavior</td></tr><tr><td>3</td><td>Delete</td><td>Delete Operation (line / section)</td></tr><tr><td>4</td><td>Edit</td><td>Edit Operation</td></tr><tr><td>5</td><td>End</td><td>End editor / finish</td></tr><tr><td>6</td><td>Merge</td><td>Merge behavior (message / text)</td></tr></table> <p>Slash – Command prefix interaction (" / ")</p> <p>!60 is also used as a boolean-style flag to enable slash-style command entry behavior. One observed usage is: 311 II=GO(!60,47,.)</p> <p>Where: 47 is ASCII "/"</p> <p>This suggests that when !60 is active (non-zero), / is used as the command-prefix character by the editor logic</p>	!60 Value	Command	Notes	1	Divider	Insert or manage message divider	2	Wrap	Wordwrap toggle / behavior	3	Delete	Delete Operation (line / section)	4	Edit	Edit Operation	5	End	End editor / finish	6	Merge	Merge behavior (message / text)
!60 Value	Command	Notes																				
1	Divider	Insert or manage message divider																				
2	Wrap	Wordwrap toggle / behavior																				
3	Delete	Delete Operation (line / section)																				
4	Edit	Edit Operation																				
5	End	End editor / finish																				
6	Merge	Merge behavior (message / text)																				

Color 64 BBS Manual Version 8.1A March 2026

Topic	Observation / Conclusion
	<p>This supports the interpretation that !60 is not merely a one-shot selection value, but the editor command-mode state that can be:</p> <ul style="list-style-type: none"> • cleared (0) while collecting or editing input, or performing screen cleanup • set (1..6) when a command has been selected and needs to be dispatched <p>Additional references in the editor flow</p> <p>Line 1300 shows !60 being initialized prior to the editor loop and used with other state variables: 1300 !7,FM:A=I:!60,1:[GOSUB311:IFP]...</p> <p>Lines 2885 / 2888 show the same “set !60, dispatch, then reset” pattern: 2885 ... :!60,1:[GOSUB311:IFP]... 2888 ON!60 GOTO 2890,1305,2889,2895:!60,1:@</p> <p>Working conclusion</p> <ul style="list-style-type: none"> • In Color 64 v8.1, !60 is a message-editor command state / selector variable: • It enables slash-command style operation (/ prefix) when active • It holds the selected editor-command index (1..6) • It is used in ON-GOTO dispatch tables to jump to the correct handler • It is frequently reset back to 1 after a handler executes <p>Open questions</p> <ul style="list-style-type: none"> • Whether !60 is also modified by the .9 input routine, or by other ML routines, to signal screen-edit cleanup, such as controlling the delete-echo behavior at line 2932 • Whether !60 has additional meanings beyond editor-related command selection <p>This is also used in conjunction with !56 and !57.</p>

Color 64 BBS Manual Version 8.1A March 2026

ML Functions

Table 64 – ML Function Changes in 8.1 - Research

Topic	Observation / Conclusion				
@31	<p>Sequential Read (Enhanced)</p> <p>Overview: @31 is a machine-language function introduced in later Color 64 versions that replaces the older @5 + !40 sequential line-read pattern used in version 8.0. It retrieves the next logical record from an open sequential file and updates ST as a side effect.</p> <p>Higher-Level Replacement for @5</p> <ul style="list-style-type: none"> Encapsulates record retrieval and length handling internally Reduces BASIC-level string churn Improves performance and garbage-collection stability <p>It is used prominently in merge-message routines within BBS.MSGS.</p> <p>Behavior:</p> <ul style="list-style-type: none"> Returns a string, typically assigned to I\$ Updates ST (disk / file status) Designed for use inside ST-controlled loops Commonly paired with @9 (enhanced FRE) for memory safety <p>Example Usage (v8.1a)</p> <pre>2330 T=@9:[I\$=@31:II=ST:GOSUB2400 2335 T=T-LEN(I\$):A\$(A)=I\$:A+=@[II] AND A<ML+4 AND T>=CM%(7,.):GOTO505</pre> <p>Functional Characteristics:</p> <p>1) Sequential File Read</p> <ul style="list-style-type: none"> Retrieves the next record or line from the currently open sequential file Replaces the older @5 + !40 logic ST must be checked after each call <p>2) Status-Driven Looping</p> <p>Typical pattern:</p> <pre>I\$=@31 II=ST @[II]</pre> <p>Loop continues while ST indicates a valid read</p> <p>This is frequently paired with the enhanced FRE function @9 and the maximum-message-lines value CM%(7,.), as seen in the merge-message function of bbs.msgs starting at line 2330.</p> <p>In comparison:</p> <table> <tr> <th>8.0</th><th>8.1</th></tr> <tr> <td>2330 I\$=@5:IF !40<LEN(TX\$) THEN I\$=I\$+CR\$</td><td>2330 T=@9:[I\$=@31:II=ST:GOSUB2400</td></tr> </table> <p>Notes:</p> <ul style="list-style-type: none"> @31 is not a pure function <ul style="list-style-type: none"> It requires an open file context It updates ST It should be used within the normal BBS runtime environment <p>Conclusion: @31 is an enhanced sequential-record read function used primarily in message handling and merge routines. It streamlines record retrieval, updates ST for loop control, and is typically combined with @9-based memory-budget safeguards.</p>	8.0	8.1	2330 I\$=@5:IF !40<LEN(TX\$) THEN I\$=I\$+CR\$	2330 T=@9:[I\$=@31:II=ST:GOSUB2400
8.0	8.1				
2330 I\$=@5:IF !40<LEN(TX\$) THEN I\$=I\$+CR\$	2330 T=@9:[I\$=@31:II=ST:GOSUB2400				

Color 64 BBS Manual Version 8.1A March 2026

@32**@32(2,2) Return Block Count**

The @32 ML function is used with an open directory listing, specifically the "\$" directory stream, to read and parse a directory entry. It returns a numeric value and advances the internal "current directory entry" pointer so that subsequent calls read the next entry.

In Color 64 v8.1, @32 is commonly used as a fast replacement for the older BASIC parsing method in which a program would GET# fields from a "\$" listing and use @6(low\$,high\$) to build a 16-bit value. In practice, @32 is used to obtain the directory entry's block count, or file size in blocks, as an integer.

Syntax: A=@32(2,2)

Return Value: Returns the current or next directory entry's block count as an integer.

Status / End-of-List Behavior:

@32 works as an iterator over directory entries. When there are no more matching entries, or the end of the directory stream is reached, BASIC's ST is set. Code typically tests ST immediately after calling @32.

Typical pattern:

```
A=@32(2,2)
IF ST THEN ... :rem end of directory (or no match)
```

Common Usage Patterns:

1) Check if a file exists, or get its size

- Open a filtered directory listing and attempt to read the first match
 - If ST is set after the call, the file was not found
- ```
OPEN 8,DV,8,"$"+DR$+F$+"*"
GOSUB 770 :rem read/skip directory header (implementation-specific)
A=@32(2,2)
IF ST THEN ER=62:A$="NOT IN DIRECTORY: "+F$:GOTO <error handler>
```

2) Display a directory-style listing (blocks + filename)

- Call @32 repeatedly until ST is set
  - The returned value is typically printed as the blocks column
  - The filename is commonly retrieved via @5 after each call, since @32 updates the current-entry context
- ```
B=-3
B=B+1
C=@32(2,2)
IF ST THEN PRINT B;" FILES SELECTED":RETURN
PRINT C;TAB(6);@5
```

3) Maintenance checks based on file size (example: caller-log trimming)

- Compare @32's returned block count against a configured maximum
 - Also used in free-space safety checks to avoid dropping below MU, the minimum number of blocks required for uploads
- ```
F$="/CALLER LOG":GOSUB 710
P=A-CM%(8,.)
P=-P*(P>.)
:rem P is positive only when the file exceeds the configured limit
```

**Notes**

- @32(2,2) is always used in conjunction with an open "\$" directory stream
- The parameters (2,2) are used in v8.1 code for directory-entry parsing; other parameter combinations, if any, are currently undocumented
- @32 is generally paired with other ML functions that return fields for the current directory entry, commonly @5 for the filename. @32 updates that current-entry context each time it is called

**8.0 Comparison:**

| Version | Typical Technique                                                                 | Result                                                          |
|---------|-----------------------------------------------------------------------------------|-----------------------------------------------------------------|
| 8.0     | Parse the "\$" listing using GET# and build a 16-bit value using @6(low\$,high\$) | File block count as an integer                                  |
| 8.1     | Read and parse the directory entry directly using @32(2,2)                        | File block count as an integer (faster, no manual GET# parsing) |

**@33****Return CHR\$(0) Null**

@33 is used exclusively in the editor portion of message composition and appears to be involved when the user is utilizing the / command within the editor, where @33 is part of removing the prompt from the editing screen. It returns a string of length 1 containing CHR\$(0).

## Tools

### Directory Tool

The "**dir tools**" program is a utility that can be used to quickly create a new **vdirectory** file for download directories.

Although Color 64 will automatically create a **vdirectory** file the first time a caller requests a directory listing, that regeneration process can take quite a while, especially if the disk contains a large number of files.

The "**dir tools**" program is a small BASIC utility that leaves plenty of free memory available, allowing it to run much faster. The program provides two main functions:

- Create a directory from scratch
- Regenerate an existing directory listing

To use "**dir tools**", boot the ML shell using the "**+shell**" program, then load and run "**dir tools**". You may also load and run it after using any other program that already uses the ML, such as **SETUP**.

The tools program reads in the system **PARMS** file.

Once the program is running, use the **Create Directory** option when creating a **vdirectory** file for the first time. This routine will scratch any existing **vdirectory** file already present on the disk and then build a new one.

This method is much faster than allowing the full regeneration to occur while the BBS is running. If you intend to start the system with hundreds of files already present in your download areas, it is recommended that you use this create utility on all relevant disks before bringing the BBS online.

You can also use the **Regenerate Directory** option to scan the disk directory and update the **vdirectory** file with any new entries. The program will also remove entries that are no longer valid.

It is recommended that this regeneration function be run periodically as part of normal maintenance to keep the directory listing clean and accurate.

As with the create function, using the regeneration routine from this stand-alone utility is significantly faster than doing the same work from within the live BBS system.

### Password File Tool

The "**pswd tools**" utility allows you to copy data from the relative password file into an easier-to-manage sequential file for backup purposes. It can also restore the password file, either in full or one record at a time, from that backup file when needed.

With this utility, you can safely store the password file on any device you choose, such as an SFD-1001 or a hard drive, if device supports relative files.

To use "**pswd tools**", boot the ML shell using the "**+shell**" program, then load and run "**pswd tools**". You may also load and run it after using another ML-based program, such as **SETUP**. The utility reads in the system **PARMS** file.

#### Backing Up the Password File

To back up the password file, choose the "**Backup Password File**" option from the "**pswd tools**" menu.

## Color 64 BBS Manual Version 8.1A March 2026

The program will ask for confirmation and then begin creating the backup file in the same file section as the password file. As the process runs, you should see a counter increment, showing the number of records backed up.

When the backup is complete, a new sequential file named "**vpassword backup**" will exist on the disk. This file contains all information currently stored in the password file, but in a sequential format that is easier to handle.

Once the backup is finished, you should copy the "**vpassword backup**" file to another disk or storage device for safekeeping using your preferred file copier.

### Restoring the Password File

To restore the password file, first make sure that a relative password file already exists in the target file section.

The restore routine does *not* create a new password file. It only writes backed-up data into an existing one. If a password file does not already exist in the desired location, run the **SETUP** program first so that it can create one for you.

The password file does not need to be blank. The restore process will write over any data already present.

You must also make sure the "**vpassword backup**" sequential file is present in the same file section. If necessary, use a file copier to place the backup file on the correct disk before beginning the restore.

To start the restore, choose the "**Restore Password File**" option from the "**pswd tools**" menu.

The program will ask for the record number to restore, or **0** to restore all records.

Once started, a counter should begin incrementing as the backup file is written into the password file. This process is much slower than the backup process, which is normal when writing into a relative file.

When the restore is finished, the password file should be ready for use.

### Fix Password

The "**pswd tools**" menu also includes an option called "**Fix Password**".

This utility can be useful if a caller's record becomes unreadable due to corrupted data. It does *not* repair disk read errors. Instead, it attempts to straighten out records that contain damaged or inconsistent data.

Under normal circumstances, restoring the damaged record from "**vpassword backup**" is the preferred solution. However, if no backup file is available, "**Fix Password**" may be the only remaining option for salvaging what is left of the password file.

## PlusTerm Program

The **Plusterm** program was written by Color 64 enthusiast **Sam Lewit**, who also created the Color 64 Network system and many other utilities. It was originally developed as an optional replacement for the built-in terminal program included with Color 64 v7.37, but beginning with Color 64 v8 it became the standard terminal program.

The program file is named "**vbbs.term**". It is automatically loaded from the **Program Files** area when the SYSOP selects **F2** from the SYSOP Menu.

Having Plusterm resident is optional. If disk space permits, add the files associated with the program:

---

**Color 64 BBS Manual Version 8.1A March 2026**

- **vbbs.term**
- **vsys.mltmno** or **vsys.mltmsw**

If the terminal program is not present in the Program Files area, the system will simply return to the wait-for-call screen when **F2** is selected.

**Some features of Plusterm include:**

- Uploads and downloads supporting **Xmodem** and **Punter** protocols
- **Multi-Punter** upload and download transfers
- A phonebook with auto-dial capability, storing up to **20 numbers**
- **2400 BPS adjustment** controls for certain modems
- A built-in communications **buffer** capable of handling thousands of bytes
- Adjustable **re-dial rate** for the auto-dialer
- **Programmable function keys**, individually defined for each phonebook entry

**Notes on Using Plusterm**

The **2400 BPS adjustment** feature applies only to systems that do **not** use SwiftLink. Editing these values will not affect SwiftLink communications. The adjustment should only be used if garbled communications appear to be caused by timing differences between the computer and the modem. This condition can occur when a modem runs slightly “fast” or “slow.” In most situations these settings will not need to be changed, but experienced users can adjust them if necessary.

The **buffer** in Plusterm is dynamic. The memory used by the buffer occupies the space between the end of the terminal program and the memory location marking the end of **vbbs.init**. On many systems this allows more than **10,000 bytes** of buffer space.

The amount of available buffer memory depends on program sizes:

- The **smaller** the terminal program, the more buffer space is available.
- The **larger** the **vbbs.init** program, the more buffer space is available.

For this reason, it is important not to add modifications unrelated to the terminal program into the terminal program itself.

When leaving the terminal to return to the BBS system, the buffer will be overwritten and its contents lost. If data remains in the buffer, Plusterm will display a warning before exiting in case the contents need to be saved.

If it becomes necessary to load the buffer with a sequential file, perform the following steps:

1. OPEN the buffer
2. Enter the DOS menu (**F4**)
3. Use the command `f:filename`
4. CLOSE the buffer

The buffer can be sent to one of three destinations:

- **SCREEN**
- **DISK**
- **MODEM**

This is done by selecting **(P)rint** from the **BUFFER** menu.

## Function Keys and the Phonebook

There are **7 programmable function keys** available for each phonebook entry. These definitions are created by editing the phonebook.

There are also **7 default definitions** that are loaded when the terminal program starts. However, once any phonebook entry is used, these default definitions are replaced by the definitions associated with that entry and remain changed until the terminal program is exited and re-entered.

To edit the default function-key definitions, select **0** from the phonebook editor menu.

To include a carriage return as part of a stored command sequence (for example, sending a password followed by RETURN), insert a **CTRL-Y** in the command string:

1. Type the password (do not press RETURN)
2. Hold the **CONTROL** key and press **Y**
3. Observe the cursor move down one line
4. Press the **RETURN** key

The **CTRL-Y** character will automatically be converted to a carriage return when the data is transmitted to the modem.

## Automated Menu Maker

The included program "**menu maker**" is a self-contained utility used to automatically create command menus for each access level on the system.

To use "**menu maker**", first boot the "**+shell**" program to install the Color 64 ML. Then load and run the "**menu maker**" program. If you have just shut down your BBS, you may also load and run "**menu maker**" without first running "**+shell**", since the ML will still be resident in memory.

When the program runs, it loads the "**Vbbs.parms**" file in order to determine the access level associated with each command.

The DATA statements located at lines **40000–40043** contain the text descriptions for each command number. Any entries marked "**\*\*UNUSED\*\***" are ignored when the menus are created, since they correspond to parameters that are not actual commands.

You may change these descriptions as desired; however, only the first **15 characters** of each description are used. This limitation ensures that the menu formatting remains consistent.

Lines **20035–20050** contain the header used for the generated menus. The current header text is generic and can be modified to whatever you prefer.

The menus are designed to display correctly in both **40-column** and **80-column** modes. Each command entry consists of the command character and description totaling **20 characters**, which allows two entries per line in 40-column mode.

Example of the menu display in 40 columns (an **F5** color code precedes each description, causing it to cycle through the 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 later change a command character in **SETUP** (without changing the access level), the menu files will automatically reflect the new command character. In that case, it is not necessary to run the menu maker program again.

The generated menu files also include MCIs that print the current **time** and **date**, which helps reduce overall program space requirements.

## Appendix A Military Time Conversion Chart

Table 65 - Military Time Cross-Reference

| Hour<br>Standard | Hour<br>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               |