64HDD – The CBM Drive Emulator

{This manual is being developed in conjunction with the program. The general sections of the manual will discuss items as if the functions and capabilities have been verified and proven, however the "general" section may get ahead of the current technical capability of the software. For what the program actually does in its present form please refer to the Configuration, Command Reference and Specification Pages}.

Overview

64HDD is software which provides CBM Drive emulation without the need of special drivers or kernal patches on the CBM computer. It does this through a X1541 or XE1541 cable connecting the LPT port of a PC based controller and the CBM computer. Real CBM drives can still be daisy chained into the system. A parallel transfer cable can also be added to dramatically reduce program load times. 64HDD supports all MSDOS based mass storage hardware and various CBM-emulator disk images. It also emulates most CBM drive commands, omitting only those which require the 65xx hardware (for example M-E, memory execute).

Introduction

64HDD is a program for a PC based "system" which emulates the serial bus protocol of Commodore equipment and additionally provides emulated devices on this bus. Primarily, the vision for 64HDD is to allow a PC based controller to be built up and to serve the CBM computers as a mass storage device.

Though there have been several hard disk drives developed over time for the Commodore 8-bit machines, most have been expensive, rare and… expensive (I intentionally said that twice). The mass storage products by CMD (Creative Micro Designs) are today the most readily available, and indeed CMD is one of the few companies who still support us Commodore users and collectors.

The older 1541, 1571, 1581 devices were based around micro-systems using the 65xx hardware. These days 386/486 hardware is getting cheap enough to dedicate these systems as micro-controllers to 64HDD CBM mass storage device. Also many 386/486 boards exist with all I/O on-board simplifying construction, and there is a wealth of applications which have proven the use of X1541 and XE1541 cables as a way of controlling /reading /writing to a CBM drive. The 64HDD emulator does the reverse and acts as a server (intelligent drive controller) for the CBM computer (any CBM computer which uses the standard CBM serial bus protocol: Vic20, C64, C128, Plus/4, C16 etc).

Additionally, the 64HDD program can support any media device that can be supported by the 386/486, for which an MSDOS compatible driver /system is required. These include: FDD, HDD, CDROM, RamDrive etc. In addition to supporting the hardware, various CBM image formats can be supported including disk (D64, D71, D81), Tape (T64) and archive (LNX). Raw binary and text files saved in the MSDOS filing system can also be "loaded". New formats are supported such as MSDOS and H64 (a large scale version of a D81 which is capable of supporting hard drive size applications with a CBM structure of up to 255 tracks and 255 sectors). The MSDOS file system can be configured in a way which allows 16character filenames to be used (these use the Windows95 LongFileName (LFN) format). Also, 1581 emulation with a 1.44Mb 3.5" floppy drive is possible with the appropriate program add-ins.

Emulation of the standard CBM drive command set has been extended with a number of new commands added to provide Real-Time-Clock information (from the PC controller's RTC) and to manage the new hardware and devices. Mouse support for the CBM computer is provided by command channel functions when a PC mouse is attached and configured to the 64HDD host.

Very few standard commands, namely those requiring the specific 65xx hardware such as memory execute are not supported, though reads /writes to drive RAM /ROM are generally supported whilst accessing CBM disk images.

Version One of 64HDD will only support the standard CBM serial protocol, and hence without the ability to support memory execute commands, most fast-loaders will not function correctly (or at all). It is expected Version Two will support CBM Burst mode (may require a modified cable as the burst line is not on the standard X1541 cable, and Version Three is envisaged to support JiffyDOS (with JiffyDOS equipped computers). Versions Two and Three may not eventuate if their protocols cannot be emulated with a humble 386/486.

It is not intended to emulate the 1541 on a cycle basis and the execute capabilities. This would be a task better left to those who have written C64 emulators. But alas no such product has been spawned from their fine work. There are of course down sides associated with to writing for a 386/486 CPU in mind. These primarily are code execution speed related and another reason for not even contemplating an implementation of CPU emulation.

The 386/486 hardware platform can be a stand-alone PC device in a standard PC case, or can be integrated in one of the following ways as is author's the intended implementation:

1) Small (Baby-AT form factor) all-in-one 486 board fitted within the case of a C128DCR; or

2) Small (Sub-Baby-AT form factor) all-in-one 486 board fitted in the case of a 1571 or 1541 drive (for transportability).

The hardware specification level of the PC controller is reasonably minimal, and the design of 64HDD is for user-free operation once configured (as such a keyboard and /or video card is not required and can be flagged "off" in the

PC BIOS). Because of this, no “pretty” user screen is planned for the software, but special commands are provided to manage the PC hardware via the CBM computer.

Background

The concept of 64HDD was inspired by the number of PC based CBM emulators being released and the relative drop in the price of PC hardware comparative to that of CBM compatible specialist hardware. The concept was that someone out there must have this “product” which would do what emulators do, especially as there were programs readily available to create disk images from a LPT port controlled 1541 (my recommendation is Trans64).

An extensive search on the Internet identified many programs which provided PC-comms to the C64, but required either drivers or kernal patches. These also often necessitated specialist cables and didn’t support “standard” commands and direct disk access. Only two programs where identified that attempted what I was hoping was possible:

VC1541	This program works well (particularly v0.3pl3) on a 486, but had no write or command channel support. Unfortunately only the newer versions introduced the save and command functions, but required Pentium processors (even the 486 version of the program worked unreliably on my 486DX2/66). Pentiums were not yet the commodity I could dedicate to this project. Additionally, the direction of the program was going down the “user interface path” rather than the “black box” system controller which was my objective.
SERVER64	This is another good program and has both read and write capabilities, unfortunately it also had no command channel support and could only work with one image or directory per initialisation. It appears work on this stopped long ago.

Consequently, this project was born with the purpose of implementing some of the features the above two programs did, plus a range of other functions. Ultimately, the aim is to make this a viable option for those not able to get their hands on a CMD hard disk, and further more, allow those with many emulator disk images to use them easily with their original CBM equipment.

There were various sources of information regarding the protocol documented on the Internet, and even some code in various programming languages. The implementation of the protocol supported by the aforementioned programs was unfortunately not adequate for detailed emulation, the reason being that they employed “streaming” of data rather than true control of data transmission with ATN signal acknowledgment. Basically, what this meant was that the simple streaming techniques worked fine for streaming a PRG file, but could not be used to control SEQ data or byte-by-byte transmissions as required for command control. This was new ground for the 64HDD project, and implementation was made all the more difficult because of the CPU constraint imposed.

CBM Computer Compatibility

64HDD is intended to be compatible with all CBM models which had a CBM serial port such as that used by drives such as the 1541. This port is also referred to as an IEC serial port. CBM models include:

> C64, C64c, SX64, C128, C128D, and C128DCR C16, Plus/4 (and presumably also C116, etc) and VIC20 (and VC20)

Of course, not every software release will be fully tested with each of these machines. Generally, the 64HDD emulator is being developed with a C128DCR with testing in both C64 and C128 modes. The development PC hosting the emulator is a 486SX33. Please report any incompatibilities.

Please note, as the 64HDD emulator does not support cycle emulation and CPU execution, schemes which are NTSC /PAL sensitive cannot be expected to work.

64HDD has been tested with standard CBM Kernals and the JiffyDOS Kernal for the C64. It is not intended to support alternate turbo Kernals that do not attempt to identify the device before initiating turbo transactions.

> Special Note for troublesome communications: for the IEC communications to work reliably with some installations of 64HDD you may need to have a real CBM disk drive (eg 1541) connected as part of the daisy chain, and it must be switched "on". This is generally observed to be a reflection of the PC controller not being fast enough.

The only two characters which may give rise to unexpected results when used in a filename, these being the "/" and "$" characters which now are used to control directory access. It is also worth noting that some CBM characters displayed are not true ASCII as the PETASCII map contains several duplications. For example "~" is CHR$(126), and not CHR$(255) as would be sent if contained in filename quotes. Remember though that JiffyDOS and other DOS wedges to not accept constructed filenames, eg `/"abc"+chr$(126)+"xyz"` will not work with JD which will issue `/"abc"` only.

64HDD will co-exist with GEOS, though at present GEOS can not be run from a 64HDD device. The configuration I use with GEOS is 64HDD devices #10, #11 and #15, with real CBM disks as #8 and #9. There is a GEOS RTC driver provided with the 64HDD file distribution, and assistance in developing additional utilities is encouraged.

64HDD will work with the Handic "Vic-Switch" and will presumable work with other IEC multiplexing /networking devices. For the Handic configuration a real CBM drive was found to be definitely required in the daisy-chain.

If a particular disk drive command is not support then one of the following errors will be reported:

31,SYNTAX ERROR
This means commands is not understood or syntax is incorrectly given.

8x,COMMAND NOT SUPPORTED (xxxxx)
This means that the command is understood, but that its function has not been implemented yet or is unavailable with the selected file system. Error 89 means that 64HDD does not yet support the function and the "beeps" will also be sounded each time such a command is given (for example with M-E). In C128 mode, 64HDD will also beep as the burst load command is tested for, but loading will continue as normal after this.

Some additional error codes have also been added such as error 78 for when an MSDOS device is "Not Ready"

Protocol Speed

The transfer rate for 64HDD can be configured by the user using command-line switches. The default setting uses timings that are essentially to Commodore specifications so any improvement in transfer rate is in the reduction of "seek" and decode times.

However, optimisation of the timings is possible via the use of the `–faster` and the `–fastest` command line switches. These switches reduce the duration of the high and low CLOCK transitions to the minimum levels, thus maximising data throughput. These options however might not work with your particular PC hardware configuration as they are dependent on the speed of the PC running 64HDD. Higher clock rate PC controllers (>40MHz) should use the `–faster` setting, slower clock rate PCs (<33MHz) should use the `–fastest` setting. An optimised system should be capable of attaining load /save performance at least comparable to a 1581 drive, and generally similar to the CMD HDD with the best settings. If you are using with a VIC20 the `–vic` option will set timings to the 1540 standard.

A parallel cable connection has been introduced to 64HDD. This, in combination with the PwrLoaders should speed communications up by 30-50 times over a standard CBM1541. See separate documentation is with the add-in package.

Configuration Guide

Basic Requirements:

- 386/486 PC controller (or motherboard with CPU). Minimum speed is believed to be a 386SX25 (for basic LOAD type operations). You can use a Pentium or higher system (having said that, I have not tested 64HDD with one, and as a point of note a Pentium of equal speed will run the 16bit code slower than a 486)
- 1x LPT port mapped to a standard location (LPT1, LPT2, LPT3). The LPT port needs to be an old SPP type if an X1541 cable/adaptor is used. A PS2 type bi-directional port might work. On-board LPT ports on Pentium and newer machines may not operate as SPP correctly even if set to do so. For these systems you will need to install and "old" 8- or 16-bit I/O card or alternatively use a XE1541 cable/adaptor.
- X1541 or XE1541 cable (refer to LPT port requirements). Alternatively, a X1541 or XE1514 adaptor can be used in conjunction with a standard CBM serial cable (my preference for shielding reasons).
- 1x mass storage device capable of booting (FDD or HDD)
- 512kB RAM (generally the minimum for this spec of machine, rather than a 64HDD requirement)
- MSDOS (or compatible). MSDOS 7 (Win95) appears to be incompatible and if the CPU is not fast (have had success reported with a P200 system).
- A real CBM disk drive such as a 1541, 1571, or 1581 may be required as part of the daisy-chain.

Optional Requirements:

- 1.44MB 3.5" Floppy Disk Drive for file transfer to the machine and for 1581 support
- PC Speaker for "start-up" beep, and "clicking" to signal activity after each block is transferred.
- keyboard (for maintenance), not scroll-lock light will flicker to represent 64HDD "disk activity"
- display card /monitor (for maintenance, personally during program development I use a CGA card as its output drives a 1901 or 1084 Commodore monitor, so at a "flick of a switch" you can see either the PC or CBM screen – of course you have to change cables to view the C128 80column screen)
- CDROM drive and DOS driver (for mega mass storage)
- PC Mouse and DOS driver (for MOUSE: command channel support)

Building the PC:

Information on constructing a PC from boards, etc is readily available, so rather than dwell on the detail only the highlights will be given. Construction of a PC in a CBM case is trickier. No instructions are yet available.

- Mount main board (often a mother board to the case)

- Add “cards” for video, etc and RAM SIMMs
- Change any jumpers on the main board necessary to disable functions duplicated by the add-on cards. For example, if you add a CGA card and the main board has “on-board” video, you will need to disable it. You may also need to set a jumper for selecting video mode (normally determine whether bios uses colour or not)
- Install your mass storage device. If you will not be installing a floppy drive make sure the HDD has been pre-formatted and an operating system installed, otherwise install also a floppy drive. Install any other “drives” at this point.
- Connect power cables to the main board and drives; connect keyboard and monitor.
- Start machine, enter BIOS set up (usually hold DEL-key) configure options to make your system run (do not ask me what these are). If you will ultimately be using the PC without keyboard or display either set these options as “not installed” or disable “halt on errors”.
- Re-boot machine, if starting with an unformatted mass media device format this and install the MSDOS system
- Check that the machine works reliably as a PC before going further…

Configuring 64HDD:

- 64HDD is distributed in either a ZIP or EXE compressed file. Expand the file (typically using the `-d` option) and follow included installation instructions in the README.TXT file. The documentation below is comparatively too general.
- Know beforehand the LPT number to which the X1541 or XE1541 cable is connected. You might need to manually set the LPT number should auto detection not work. You will need to specify `-xe` if you are using the XE1541 cable, refer to README.TXT.
- Decide which “paths” you want, and how you wish to configure your device numbers. Real CBM peripherals and the 64HDD emulator cannot have the same device number (they will clash and lock the system up). You can set the device /path for each number used on the command line calling 64HDD (or in the batch file). You may need to modify /install one of the example batch files to do what you want. See README.TXT for more information. Remember that the MSDOS command-line is limited in length and that these limits should not be exceeded. Connect to default paths if you want to start many devices simultaneously.
- Connect the PC to the CBM computer using the cable. You can daisy chain other CBM peripherals as normal. Obviously, you’ll need either a double X1541 adaptor or an external disk drive (which has double ports) if you want several external devices attached.
- Generally speaking, a PC LPT port will pull lines “low” when not “running”. This has the effect of pulling the ATN line low which holds all real CBM products at “attention” – ie waiting for a command. It also conflicts with some CBM computers (eg the C128) which have an auto-booting mode (hence, the machine will not get to the cursor until the PC has released the ATN line).

Note: This has not been found to be dangerous to the CBM hardware, but the practice of booting the PC before switching on CBM equipment may be recommended if you are worried.

- Start the batch file controlling 64HDD. As 64HDD boots check to see that the devices and paths are as you require and that the X1541 LPT port is found.
- When 64HDD is up and running (signalled by a long "beep" if your PC has a speaker) switch "on" the CBM machine.

  Note: If your X1541 cable connects the RESET line from the IEC serial port, the 64HDD emulator will reset when the CBM computer resets. This occurs when you switch on the CBM computer, but by using the batch file supplied with 64HDD, the emulator will restart. 64HDD can be reset by resetting the CBM computer. The 64HDD emulator will not be reset by pressing the "drive reset" on a C128D or DCR.
- After the emulator has started issue the `LOAD"$",device` command. If a PC directory is listed then you are in business. If not, see trouble shooting.
- Attach one of the disk images supplied with 64HDD utilities and work through the examples to test function and capability.
- If you wish to have LongFileName support in the MSDOS environment refer to details in that section of the manual and follow the instructions carefully.

Using a RAMDISK to speed things up (especially LFN and Power-Loader Operation):

The use of a RAMDISK can improve the performance of 64HDD in several ways. Firstly, each time 64HDD is subsequently loaded, it is done so from RAM which is faster. Secondly, the DOS ROM images which are stored in the system directory will also be faster to access. Thirdly, the directory scratch files will be created faster. And fourthly, there will be significant improvements to the speed of LFN and Pwr-Load support functions as these also will be available in RAM and any temporary files can be created in RAM also.

- Install RAMDRIVE.SYS (or RAMDISK.SYS) as per the instructions that come with your version of MSDOS. Depending upon your system is it recommended to use extended memory, and as much of it as is available since 64HDD make no other use of it.
- Replicate the 64HDD directory structure on the RAMDRIVE. COPY or LCOPY all EXE and BAT files to their corresponding locations and also all files in the 64HDD\SYSTEM directory
- Modify the PATH statement to include the EXE location on the RAMDRIVE as the first item
- If you wish to always use the RAMDISK configuration, make the above changes part of your CONFIG.SYS and AUTOEXEC.BAT files.
- Modify GO64HDD.BAT such that it starts the copy of 64HDD.EXE residing on the RAMDRIVE. This is important as it will force the LFN tools to also be loaded from the RAMDRIVE.

- Modify the GO64HDD.BAT file so that it copies back the CMDPRTN.TBL file to a permanent location (otherwise the updates will be lost when power is switched off)
- Modify the GO64HDD.BAT file such that calls to 64HDD include the +sysdir option to set the system directory as the RAMDRIVE location. The path to the system directory is limited to 32characters.
- You may also modify the GO64HDD.BAT file such that calls to 64HDD include the +prtndir option to set the CMDPRTN.TBL save directory to a location that is not the same as +sysdir (such as the HDD). This change will ensure that all changes to the partition table are saved to non-volitile memory, but will slow down operation. The path to the partition directory is limited to 32characters, and the option must be set after the +sysdir option.

Notes:

- A “SmartDrive” style disk cache system may give similar benefits in disk operating speed.
- The MSDOS command prompt is limited in size (limited by Microsoft, not 64HDD) and so be mindful about the number of options used. Remember, you can enable drives to the default path with –9, -10, etc.

Trouble Shooting 64HDD:

- Trans64 Compatibility: Note TR64 will not work when the X1541 cable is connected to a CBM Computer instead of, or in addition to, CBM disk drives (even if the CBM computer is “off”). The following fix was provided by Jochen Adler:

  After a reset the C64 sets clock to active (0V) and thus TRANS64 cannot use this line. The poke56576,7 sets all serial lines to inactive (5V) and TRANS64 then works.
- X1541 LPT port not known: use MSD (Microsoft Diagnostics) or another PC utility to identify the printer ports you have available and their addresses.
- X1541 connection not working:
  - Have you used a X1541 cable before (eg with TRANS64, etc)?
  - Have you got the X1541 cable attached to the correct LPT port?
  - Is the LPT port you are using the correct type?
  - Is the connector securely fastened (I often find this is the problem – a bumped cable)?
- Intermittent operation: check that idle indicator (twirling bar) works when 64HDD started and after loading
- Suspended operation: if data transmission is lost soon after the transfer begins it is likely that the PC controller is not running the emulator fast enough. Keep in mind the program’s minimum specification, and avoid unnecessary TSRs which take away from processing performance (eg if you do not need a mouse driver, do not install it).
- Searching for….: If you are using the XE1541 cable, chances are that you have not specified `-xe` option when starting 64HDD.
- Device Not Present Error: check that device is activated when 64HDD is initialised. If you get this message during serial and /or GET# data

transfers it may be a serial bus timing error – please notify the author. Some problems are known with the C16/Plus4/Vic20 during GET# data transfers. This indicates that the emulator is not running fast enough or is installed on a PC controller which is not able to run the program fast enough. Solutions are to use a faster PC or to run a real CBM drive in the daisy chain. There should not be any problems with operation on a C64/C128 as the signals are slightly different (slower) but similar solutions may also be possible if your configuration has this problem.

- Load Error: this may be the same problem as above, and can be resolved by adding a real CBM drive to the daisy chain.
- 64HDD beep-beep-beep: the emulator has been given a command that is not currently support. For example the M-E command is not processed. Some fastloader schemes rely on drive programming and will probably not work with 64HDD. Note: when using 64HDD with a C128, there will be three beeps, but the load operation proceeds. These beeps are because the C128 checks whether 64HDD is cable of burst mode before reverting to normal load routines.
- HDD is not there when I boot from a floppy disk: are you using Win98 or higher with an incompatible FAT system? If so there is nothing that can be done as MSDOS does not have access to the disk.

Command Reference Guide

Directory and Disk Image Access /Loading

Command: Select 64HDD directory
Applicability:

■ All ☐ MSDOS ☐ D64 ☐ D71 ☐ D81 ☐ H64 (Native) ☐ T64 ☐ LNX

Syntax:
```
LOAD"$drive:/path=type",device
OPEN fileno,device,15,"$drive:/path=type":CLOSE
```

Where types allowed include:

P	for PRG files in MSDOS mode (PRG can also be used)
S	for SEQ files in MSDOS mode (SEQ can also be used)
U	for USR files in MSDOS mode (USR can also be used)
R	for REL files in MSDOS mode (REL can also be used)
C	for CBM files in disk image file systems
B	for directories in MSDOS and disk image file systems
DIR	for MSDOS directories
D64	for D64 images
D71	for D71 images
D81	for D81 images
D2M	for D2M images (from FD2000 made by 1581COPY)
D*	for all supported disk images
T64	for T64 images
LNX	for LNX images
MSD	for all other files

Wildcards ? and * permitted for filtering the directory listing

Examples:

Load current directory assigned to this device for viewing (use LIST as per CBM practice to see it)
```
LOAD"$",11
```

Load only directory header and blocks free information
```
LOAD"$$",11
```

Load current directory but only files matching this type

`LOAD"$,D64",11` *(only works in MSDOS mode)*

`LOAD"$,D64,P",11` *(only in MSDOS mode, multiple types)*

`LOAD"$T?ST*,P",11`

Set C:\ as current directory for future commands
```
LOAD"$C:/",11
```

Set C:\ as current directory for future commands
```
OPEN1,11,15,"$C:/"
INPUT#1, EN,EM$,T,S
CLOSE1
```

```
PRINT “NEW PATH:”;EM$
```

Change to D:\ as current directory for future commands
```
LOAD”$D:/”,11
```

Set “this-dir” as directory for future commands
```
LOAD”$THIS-DIR”,11
```

Attach “this D64 image” as an emulated disk for future commands
```
LOAD”$THIS.D64”,11
```

Dettach “this D64 image” and set root on current drive
```
LOAD”$/”,11
```

Errors:

- With the `OPEN` format of the command the error message returns the new path attached, but error code will be set to [0].
- If a partition change is requested, but not available, then error [77] may result
- If a partition change or drive is requested that is not available or would normally result in a “Drive Not Ready” message in MSDOS then error [78] may result

Notes:

- Image files will be listed with the file extension in both the name and the type. The image can only be attached if the file extension is given in the name (else a file of that name is assumed).
- The MSDOS directory listing allocates an extra digit for block size information (this is because files and disk images can have 4digit values for their size – not possible on a 1541!)
- The MSDOS directory separator “\” is exchanged for “/” on the CBM machine as the CBM keyboard has no “\” key.
- When an image or MSDOS directory is attached the directory is sent if a LOAD was issued. It is not sent if the command channel was used.
- An image can be detached by setting the directory to a known MSDOS location, eg. `LOAD”$/”,11`. Be sure to set to a known /existing directory. `LOAD”$..”,11` can also be used to return to the MSDOS directory the disk image was attached from.
- In MSDOS modes, relative paths may be used (eg. `LOAD”$..”,11`) but remember that path length is limited to 255 characters by some versions of MSDOS. It is recommended to frequently set directory to root when using relative paths.
- In MSDOS mode, a new drive may be logged by giving the absolute path to a directory, for example the root directory: “D:\”
- Pressing RUN/STOP will abort the directory load.
- With the OPEN command format the directory is set for the device, but no listing is sent on this channel. The device’s directory can be retrieved

using the normal CBM directory read from secondary address 0 for the device with file name `$`, eg `OPEN fileno,device,0,”$”` and use `GET#`
- In MSDOS mode, “blocks free” is number of blocks available on the drive, limited to 63999 because of BASIC line number limitations
- In LNX and T64 modes, “blocks free” will always be zero as write support is not provided for these image types.
- Incompatibility: because $name is now used to change to the directory or disk image, if “name” is not a directory a file not found message or default directory load will be performed. This will have some incompatibility with programs that try to identify “blocks free” by reading the information after they execute a `OPEN1,11,0,”$!#$%”` command. To patch such instances, change the last character to a wildcard, eg `“$!#$*”`

Command: **Fast 64HDD Directory Reading**
Applicability:

■ All ☐ MSDOS ☐ D64 ☐ D71 ☐ D81 ☐ H64 (Native) ☐ T64 ☐ LNX
Syntax: `OPEN fileno,device,3,”$drive:/path”`

Examples:
Get current directory using fast read
```
1 OPEN 1,11,3,”$”
2 INPUT#1,D$
5 INPUT#1,BL,FL$,TY$
6 IF TY$=”FRE” THEN 10
7 PRINT BL,FL$,TY$
8 GOTO 5
10 PRINT BL;FL$
15 CLOSE 1
```

Errors:
- See previous

Notes:
- Directory access with secondary address of 3 will initial fast directory read format (this is not supported by real CBM DOS).
- The fast format means that directory information only is sent and not the normal BASIC format detail as per a read of $ with secondary channel of 0
- First INPUT# gets the directory or disk header. This contains information such as disk id, disk name, path or file system depending upon what is attached. The DOS version in the header can be used to identify which file system is active according to the following:

1T	T64 image attached
1L	LNX image attached
PC	MSDOS directory
LF	MSDOS directory, but with LongFileName support
2A	D64 image (unless disk image has corrupt DOS Version)
2A	D71 image (unless disk image has corrupt DOS Version)
3D	D81 image (unless disk image has corrupt DOS Version)

1H D2M image (unless disk image has corrupt DOS Version)

- Second and subsequent INPUT# fetch three items in the format:
  Blocks, Filename$, Type$
- Filetype FRE is reserved for use as a flag that the details just received is block free information and there is no further data to be sent
- The notes for normal directory access should also be reviewed, but the 63999 block free limitation does not apply.

Cartridge / DOS Wedge Compatibility

The 64HDD emulator has been tested with a number of Cartridge and DOS Wedge utilities to assess compatibility as outlined in the table below. Incompatibility is normally a result of using non-standard communications or unimplemented functions, for example trying to read the directory using M-R (not yet supported) or fast-loader routines. The former type of incompatibility will be resolved with time; the latter may be outside the scope of 64HDD.

Cartridge /DOS Aid	**Works (directory /status)**	**Remarks**
DOS Wedge 5.1 (C64)	Yes/Yes	
BASIC 7.0 (C128)	Yes/Yes	DIRECTORY, DS$ is valid, DLOAD works
JiffyDOS 6.01 Wedge	Yes/Yes	@ and @$, on slow PC set-ups, use a real 1541 in series
Final Cart III	Yes/Yes	
KCS Power Cart	Yes/Yes	
Cockroach TurboROM	Yes/Yes	
Freeze Machine	No/No	Hangs, not sure why
Warp Speed (64/128)	Yes/Yes	
Epyx Fast Loader	No/No	Hangs, not sure why
Action Replay V6.0	Yes/Yes	Use OFF to turn turbo load off.

Device Independent Commands

Command: Get /Set 64HDD Time

Applicability:

■ All ☐ MSDOS ☐ D64 ☐ D71 ☐ D81 ☐ H64 (Native) ☐ T64 ☐ LNX

Syntax:

```
T:hhmmss
T0:hhmmss
TIME:hhmmss
TIME0:hhmmss
```

Examples:

Get time from 64HDD and set CBM time string

```
OPEN 1,11,15,”T:”
INPUT#1, EN,EM$,ET,ES: TI$=LEFT$(EM$,6)
CLOSE 1:
```

Set 64HDD RTC time to 13:20:30 (hhmmss, 24hr clock)

```
OPEN 1,11,15,”T:132030”:CLOSE 1:
```

Errors:

- Current time returned as the error text. EN, ET, ES always 0
- Syntax error [30] returned if format is too long /short

Notes:

- “hhmmss” must be an ASCII string
- works regardless of disk image /directory selected
- no checking that time setting was successful or that time was valid

---

Command: Get /Set 64HDD Date

Applicability:

■ All ☐ MSDOS ☐ D64 ☐ D71 ☐ D81 ☐ H64 (Native) ☐ T64 ☐ LNX

Syntax:

```
D:dd/mm/yyyy
D0:dd/mm/yyyy
DATE:dd/mm/yyyy
DATE0:dd/mm/yyyy
```

Examples:

Get date from 64HDD and print string

```
OPEN 1,11,15,”D:”
INPUT#1, EN,EM$,ET,ES: PRINT “DATE: “;EM$
CLOSE 1:
```

Set 64HDD RTC date to 30/01/1999 (dd/mm/yyyy)

```
OPEN 1,11,15,”D:30/01/1999”
CLOSE 1:
```

Errors:

- Current date returned as the error text. EN, ET, ES always 0
- Syntax error [30] returned if format is too long /short

Notes:

- "dd/mm/yyyy" must be an ASCII string
- though "/" are returned other delimiters may be used for setting, eg: space and "-"
- works regardless of disk image /directory selected
- no checking that date setting was successful or that date was valid
- Y2K compliance is a function of the RTC chip used in the PC, and not the 64HDD program

---

Command: **Get 64HDD Day**
Applicability:

■ All ☐ MSDOS ☐ D64 ☐ D71 ☐ D81 ☐ H64 (Native) ☐ T64 ☐ LNX

Syntax:
```
DAY:
DAY0:
```
Examples:

Get day-of-week from 64HDD and print string
```
OPEN 1,11,15,"DAY:"
INPUT#1, EN,EM$,ET,ES: PRINT "TODAY IS: ";EM$
CLOSE 1:
```

Errors:

- Current day-of-week returned as the error text. EN, ET, ES always 0

Notes:

- works regardless of disk image /directory selected
- full name of day returned, eg: MONDAY
- Y2K compliance is a function of the RTC chip used in the PC, and not the 64HDD program
- Availability of day-of-week is a function of the RTC type and MSDOS version being used as the emulator O/S

Command: Get 64HDD Device Space Information
Applicability:

■ All ☐ MSDOS ☐ D64 ☐ D71 ☐ D81 ☐ H64 (Native) ☐ T64 ☐ LNX

Syntax:
```
F:drive_letter
F0:drive_letter
FREE:drive_letter
FREE0:drive_letter
```
Examples:

Get blocks free on drive D
```
OPEN 1,11,15,"F:D"
INPUT#1, EN,EM$,ET,ES: PRINT EM$
BL=VAL(EM$)
CLOSE 1:
```

Errors:

- Current device space (in CBM blocks) is returned. EN, ET, ES always 0
- Syntax error [30] returned if format is too short or drive_letter < A
- If drive not responding or then error text indicates this as "ERROR CHECKING DRIVE D", for example.
- If drive information is available, VAL can be used to strip number of blocks from error message

Notes:

- works regardless of disk image /directory selected
- only reports what space MSDOS "sees" for the device
- when used whilst a disk image is "logged", it reports space for the device not the image

---

Command: Exit 64HDD and set error_level upon exit
Applicability:

■ All ☐ MSDOS ☐ D64 ☐ D71 ☐ D81 ☐ H64 (Native) ☐ T64 ☐ LNX

Syntax:
```
E:error_level
E0:error_level
EXIT:error_level
EXIT0:error_level
```
Examples:

Exit with error_level = 8
```
OPEN 1,11,15,"EXIT:8"
INPUT#1, EN,EM$,ET,ES: PRINT EM$
CLOSE 1
```

Errors:

- Return is Ok or equivalent EN, ET, ES always 0
- Syntax error [30] returned if format is too short or error_level < 0

Notes:

- works regardless of disk image /directory selected
- 64HDD will exit when the next file closure is made, and the emulator returns to idle mode (hence all other CBM files /channels should already be closed before executing this command)
- The error_level is used to control execution of a BATCH file on the PC controller. The batch file can be configured to restart 64HDD, or to restart it with different device assignments (see example batch files).

Command: Read PC Mouse Position and button status (ASCII)
Applicability:

■ All ☐ MSDOS ☐ D64 ☐ D71 ☐ D81 ☐ H64 (Native) ☐ T64 ☐ LNX

Syntax:
```
M:
M0:
```

```
MOUSE:
MOUSE0:
```

Examples:

Read PC mouse position and button status

```
OPEN 1,11,15,"M:"
INPUT#1, EN,EM$,MX,MY
B=VAL(EM$)
CLOSE 1
```

Errors:

- Return is Button information with EN=0, MX=X mouse position (0-511), MY = Y mouse position (0-255)
- B=1 is left Button, B=2 is right Button, B=3 is both Buttons

Notes:

- works regardless of disk image /directory selected

Command: Read PC Mouse Position and button status (Binary)
Applicability:

■ All ☐ MSDOS ☐ D64 ☐ D71 ☐ D81 ☐ H64 (Native) ☐ T64 ☐ LNX

Syntax:

```
M1:
MOUSE1:
```

Examples:

Read PC mouse position and button status

```
OPEN 1,11,15,"M1:"
GET#1,X1: GET#1,XL :GET#1,XH :GET#1,Y :GET#1,B
X=XL+256*XH
CLOSE 1
```

Errors:

- X mouse position (0-319), MY = Y mouse position (0-199)
- B=1 is left Button, B=2 is right Button, B=3 is both Buttons

Notes:

- works regardless of disk image /directory selected

Command: Run an external MSDOS application
Applicability:

■ All ☐ MSDOS ☐ D64 ☐ D71 ☐ D81 ☐ H64 (Native) ☐ T64 ☐ LNX

Syntax:

```
MSD:application_name
MSD0:application_name
```

Examples:

Shell to application

```
OPEN 1,11,15,"MSD:application"
```

```
CLOSE 1
```

Errors:

- Return is Ok or equivalent EN, ET, ES always 0

Notes:

- works regardless of disk image /directory selected
- 64HDD shells to the *application*, which must be on the path or a path must be given
- an *application* which writes to the screen may corrupt the 64HDD display, this may not be a problem if you do not need to refer to the display (for example in a dedicated system)
- the screen output of many programs is diverted rather than displayed with >MSD.SEQ with the file being written to the +sysdir\MSD.SEQ
- This command may be used for example to run PC based cross compilers

---

Command: Report protocol speed mode and Pentium CPU speed
Applicability:

■ All ☐ MSDOS ☐ D64 ☐ D71 ☐ D81 ☐ H64 (Native) ☐ T64 ☐ LNX

Syntax:

```
SPEED:
SPEED0:
```

Examples:

Report

```
OPEN 1,11,15,"SPEED:"
INPUT#1, EN,EM$,ES,ET
CLOSE 1
```

Errors:

- Return is Ok, with ET defining the mode, 1=std, 2=faster, 3=fastest, +10 if `UI-` has been previously issued. ES defines CPU MHz if Pentium TSC is being used for microsecond timings. ES=0 if Programmable Interval Timer (PIT) is used for timings.

Notes:

- works regardless of disk image /directory selected

Command: Switch to ShortFileName Mode
Applicability:

■ All ☐ MSDOS ☐ D64 ☐ D71 ☐ D81 ☐ H64 (Native) ☐ T64 ☐ LNX

Syntax:

```
SFN:
SFN0:
```

Examples:

Switch to SFN mode for future 64HDD access

```
OPEN 1,11,15,"SFN:"
CLOSE 1
```

Errors:
- Return is Ok, with ES=0 and ET defining the mode, 0=SFN, 1=LFN

Notes:
- works regardless of disk image /directory selected
- filename functions will now be in 8.3 MSDOS format

Command: Switch to LongFileName Mode
Applicability:

■ All ☐ MSDOS ☐ D64 ☐ D71 ☐ D81 ☐ H64 (Native) ☐ T64 ☐ LNX
Syntax:
```
LFN:
LFN0:
```

Examples:
Switch to LFN mode for future 64HDD access
```
OPEN 1,11,15,"LFN:"
INPUT#1, EN,EM$,ET,ES
CLOSE 1
```

Errors:
- Return is Ok if LFN mode was available at 64HDD boot-up, with ES=0 and ET defining the mode, 0=SFN, 1=LFN
- Error [160] returned if LFN was not successfully booted on 64HDD start-up

Notes:
- works regardless of disk image /directory selected
- filename functions will now be in 16.3 MSDOS format
- with JiffyDOS on you CBM computer you will need to enclose the command in quotes as follows so as to avoid the "lock" command which begins with L also: `@"LFN:`

Command: Activate another 64HDD Device
Applicability:

■ All ☐ MSDOS ☐ D64 ☐ D71 ☐ D81 ☐ H64 (Native) ☐ T64 ☐ LNX
Syntax: U0>+chr$(device_number)

Examples:
Activate device 10
```
OPEN 1,11,15,"U0>"+CHR$(10)
INPUT#1, EN,EM$,ET,ES
CLOSE 1
```

Errors:
- Return is Ok if device_number within allowable range
- Error [89] if DEVICE_NUMBER is not supported by 64HDD

Notes:

- works regardless of disk image /directory selected
- command will not work when sent by @ in JiffyDOS, use syntax above instead.
- device will be activated to the default path, c:\

File Handling Commands

Command: Open File for Read/Write
Applicability:

☐ All ■ MSDOS ☐ D64 ☐ D71 ☐ D81 ☐ H64 (Native) ☐ T64 ☐ LNX
Syntax: `“filename,type,mode”`
Examples:
Open file to write ASCII data to it
```
OPEN 1,11,1,”TESTFILE,S,W”
FOR X=1TO10:PRINT#1,”NUMBER:”;X:NEXT
CLOSE 1
```

Open file to read ASCII data from it
```
OPEN 1,11,1,”TESTFILE,S,R”
FOR X=1TO10:INPUT#1,A$:PRINT A$:NEXT
CLOSE 1
```

Save currently loaded basic program
```
SAVE”TESTFILE”,11
```

Load program file
```
LOAD”TESTFILE”,11
```

Errors:
- errors returned for file not able to be open

Notes:
- In the MSDOS file system, existing file is overwritten
- In the MSDOS file system, the name is limited to 8characters unless LFN support is selected and installed, in which case the limit is 16characters. If a type is specified, then this is converted to a 3character extension. If no file type is specified, PRG is assumed.
- PRG file load supported in all file systems.
- SEQ file reads supported in MSDOS and disk image file systems.
- RUN/STOP key press supported for PRG files.
- FILE NOT FOUND supported for PRG and SEQ formats

MSDOS ASCII Compatibility

A special ASCII exchange mode is supported by 64HDD. If the file type is specified with a “shifted” type specifier, basic alphabetic ASCII translation is performed. This means for example, by writing a file with the “shift-S” filetype, a sequential file will be written, but upper and lower case will be correctly viewable in an MSDOS editor. Likewise reading an MSDOS text file by changing filetype to USR and then opening it with a “shift-U” filetype will result in the MSDOS ASCII codes being exchanged for PETASCII equivalents.

Error Channel Reading

Command: Reading the error channel
Applicability:

■ All ☐ MSDOS ☐ D64 ☐ D71 ☐ D81 ☐ H64 (Native) ☐ T64 ☐ LNX
Syntax: See example,
Return format is:
err_number, err_string, err_track, err_sector
Examples:
Read error information on this "device" (using INPUT#1)
```
OPEN 1,11,15
INPUT#1, EN, EM$,ET,ES
PRINT EN, EM$,ET;ES
CLOSE 1
```

Read error information on this "device" (using GET#1)
```
10 OPEN 1,11,15
20 GET#1, A$:PRINT A$;
30 IF ST<>64 THEN 20
40 CLOSE 1
```

Errors:

Notes:
- Error information returned regardless of mode in which the emulator is operating in
- Only last error flagged is return
- Error may not be a fault, but informational. Error number will be set to 0 in this case, or error 99 if emulator interrogated before error flag set.
- Errors 8x are new and indicate if error was a result of the 64HDD emulator not supporting the previous command channel request.
- Dependent Devices only respond with errors compatible with error messages to the real CBM device plus the appropriate error 8x and 99 codes.

Command Channel Commands

Command: Scratch file(s) from current device /directory
Applicability:

☐ All ■ MSDOS ☐ D64 ☐ D71 ☐ D81 ☐ H64 (Native) ☐ T64 ☐ LNX

Syntax:
```
“S:filename[;M]”
“S0:filename[;M]”
“SCRATCH:filename[;M]”
“SCRATCH0:filename[;M]”
```

```
    Where:
         ;M   specified that filenames should be
              treated as explicitly give, as is the
              case when working with MSDOS files.
              This option is only valid in the MSDOS
              file system.
```

Examples:
Scratch this file only
```
OPEN 1,11,15,”S:TESTFILE”
CLOSE 1
```

Scratch this MSDOS file only
```
OPEN 1,11,15,”S:TESTFILE.JPG;M”
CLOSE 1
```

Scratch files matching this spec
```
OPEN 1,11,15,”S:TE?TFIL*.PRG”
CLOSE 1:
```

Scratch files matching this spec (all extensions /types)
```
OPEN 1,11,15,”S:TE?TFIL*”
CLOSE 1:
```

Errors:
- Error [62] returned for file not able to be found or can not be scratched
- File delete count may not be valid if using LFN mode (this is a limitation of the external LDEL program)

Notes:
- In the MSDOS file system, 8.3 filename validity is omitted an error 62 returned if file not found
- In the MSDOS file system, if “;M” has not been specified attempts will be made to delete the given filename with extension “.PRG”, “.SEQ” and “.USR”. This should not disturb existing CBM programs as they would only expect one file of a given name in each as type checking is not done by CBM DOS.

Command: **Rename file on /in current device /directory**
Applicability:

☐ All ■ MSDOS ☐ D64 ☐ D71 ☐ D81 ☐ H64 (Native) ☐ T64 ☐ LNX

Syntax:
```
“R:newfilename=oldname[;M]”
“R0:newname=oldname[;M]”
“RENAME:newname=oldname[;M]”
“RENAME0:newname=oldname[;M]”
```

```
    Where:
         ;M   specified that filenames should be
              treated as explicitly give, as is the
              case when working with MSDOS files.
              This option is only valid in the MSDOS
              file system.
```

Examples:
Rename this file
```
OPEN 1,11,15,”R:NEWFILE=OLDFILE”
CLOSE 1
```

Rename this MSDOS file
```
OPEN 1,11,15,”R:NEWFILE.JPG=OLDFILE.JPG;M”
CLOSE 1
```

Errors:
- Errors [62] and [60] returned as appropriate
- If using LFNs Ok is always reported (limitation of the external LREN program)

Notes:
- In the MSDOS file system, if “;M” has not been specified, three attempts are made to rename the file using extensions PRG, SEQ and USR. Each rename attempt is to the same “file type”, eg PRG will be renamed to PRG.
- In the MSDOS file system, 8.3 filename validity is omitted for both oldfile and newfile as a robust “check and approve” strategy still needs development, hence correct renaming is up to the user and presently requires filename extensions to be given.
- In the MSDOS file system, do not put space characters on either side of the “=”

Command: Copy file on /in current device /directory to new name
Merge multiple sequential files to one combined file
Applicability:

☐ All ■ MSDOS ☐ D64 ☐ D71 ☐ D81 ☐ H64 (Native) ☐ T64 ☐ LNX

Syntax (copy):
```
“C:backup=original[;M]”
“C0:backup=original[;M]”
“COPY:backup=original[;M]”
```

```
                    “COPY0:backup=original[;M]”

     Where:
           ;M   specified that filenames should be
                treated as explicitly give, as is the
                case when working with MSDOS files.
                This option is only valid in the MSDOS
                file system.
```

Syntax (merge):

```
          “C:combined=file1,file2,file3,file4”
          “C0:combined=file1,file2,file3,file4”

          “COPY:combined=file1,file2,file3,file4”
          “COPY0:combined=file1,file2,file3,file4”
```

Examples:

Copy this file

```
OPEN 1,11,15,”C:BACKUP=ORIGINAL”
CLOSE 1
```

Copy this file

```
OPEN 1,11,15,”C:BACKUP.TAS=ORIGINAL.TAS;M”
CLOSE 1
```

Copy this disk image from current location to a:

```
OPEN 1,11,15,”C:A:/GAMES.D64=GAMES.D64”
CLOSE 1
```

Merge these files

```
OPEN 1,11,15,”C:All=FILE1,FILE2,FILE3”
CLOSE 1
```

Merge these files

```
OPEN 1,11,15,”C:All.SEQ=FILE1.SEQ,FILE2.SEQ,FILE3.SEQ”
CLOSE 1
```

Errors:

- Errors [62] and [25] returned as appropriate

Notes:

- In the MSDOS file system, if “;M” has not been specified, three attempts are made to copy the file using extensions PRG, SEQ and USR. Each copy attempt is to the same “file type”, eg PRG will be renamed to PRG.
- In the MSDOS file system, 8.3 or 16.3 filename validity is checked for merge and SEQ file type is forced (ie only SEQ files can be merged).
- In the MSDOS file system, 8.3 or 16.3 filename validity is omitted for copy as a robust “check and approve” strategy still needs development, hence correct naming is up to the user and presently requires filename extensions to be given.

- In the MSDOS file system, 8.3 or 16.3 filename name may be the name of a disk image and if a full pathname given, a disk image may be copied between drives
- In the MSDOS file system, do not put space characters on either side of the "=" or "," unless it is truly part of the LongFileName.
- In the MSDOS file system, there is no limitation on the number of files which may be merged only the standard CBM limitation on the length of the command string.

Command: Make MSDOS directory
Applicability:

☐ All ■ MSDOS ☐ D64 ☐ D71 ☐ D81 ☐ H64 (Native) ☐ T64 ☐ LNX

Syntax:
```
"MD:new_dirname"
"MD0:new_dirname"
```
Examples:

Create this new directory
```
OPEN 1,11,15,"MD:NEW-DIR"
CLOSE 1
```

Errors:
- Error [77] returned if failure to create

Notes:
- In the MSDOS file system, 8.0 filename validity is enforced. The new directory is created branching from the current path. Wildcards are not permitted.

Command: Change MSDOS directory
Applicability:

☐ All ■ MSDOS ☐ D64 ☐ D71 ☐ D81 ☐ H64 (Native) ☐ T64 ☐ LNX

Syntax:
```
"CD:dirname"
"CD0:dirname"
```
Examples:

Change to this directory
```
OPEN 1,11,15,"CD:NEW-DIR"
CLOSE 1
```

Change to the root
```
OPEN 1,11,15,"CD:/"
CLOSE 1
```

Change to the root of drive D:
```
OPEN 1,11,15,"CD:D:/"
CLOSE 1
```

Errors:
- Error [77] returned if failure to create

- Error [2] returned if successful

Notes:

- In the MSDOS file system, 8.3 filename validity checking is omitted so as to be compatible with directory structures created before 64HDD. Therefore it is up to the user to correctly enter valid directory names. Wildcards are not permitted.
- To log a new drive use absolute naming for the directory, eg “d:/”

Command: Remove MSDOS directory
Applicability:

☐ All ■ MSDOS ☐ D64 ☐ D71 ☐ D81 ☐ H64 (Native) ☐ T64 ☐ LNX

Syntax:

```
“RD: dirname”
“RD0: dirname”
```

Examples:

Remove this directory

```
OPEN 1,11,15,”RD:TEMP”
CLOSE 1
```

Errors:

- Error [77] returned if failure to delete

Notes:

- In the MSDOS file system, 8.3 filename validity checking is omitted so as to be compatible with directory structures created before 64HDD. Therefore it is up to the user to correctly enter valid directory names. Wildcards are not permitted.

Command: Initialize device
Applicability:

☐ All ■ MSDOS ☐ D64 ☐ D71 ☐ D81 ☐ H64 (Native) ☐ T64 ☐ LNX

Syntax:

```
“I”
“I0”
“INITIALIZE”
“INITIALIZE0”
```

Examples:

Initialize this device

```
OPEN 1,11,15,”I”
CLOSE 1
```

Errors:

- Always error [0] returned

Notes:

- In the MSDOS file system, the command is acknowledged, but not acted upon.

Command: Validate device /disk
Applicability:

☐ All ■ MSDOS ☐ D64 ☐ D71 ☐ D81 ☐ H64 (Native) ☐ T64 ☐ LNX

Syntax:
```
“V”
“V0”
“VALIDATE”
“VALIDATE0”
```

Examples:

Validate this device
```
OPEN 1,11,15,”V”
CLOSE 1
```

Errors:
- Always error [0] returned

Notes:
- In the MSDOS file system, the command is acknowledged, but not acted upon.

Command: Warm boot / Cold boot
Applicability:

☐ All ■ MSDOS ☐ D64 ☐ D71 ☐ D81 ☐ H64 (Native) ☐ T64 ☐ LNX

Syntax:
```
“UI”
“UJ”
“U9”
```

Examples:

Warm boot this device
```
OPEN 1,11,15,”UI”
INPUT#1, EN,EM$,T,S
CLOSE 1
```

Errors:
- Error [73] returned with EM$ containing 64HDD version message. The last four letters of EM$ are “EMUL” can be used to detect that this is an emulated drive and as such advanced 64HDD features can be implemented.

Notes:
- Currently the error message is identical regardless of file system being used
- Active devices and partitions remain unchanged
- `UI-` will set protocol timings to 1540 (ie VIC20 or blanked screen required on C64/C128). With this timing the data valid pulse width is shorter and does not take into account the C64’s bad-line requirements.
- `UI+` will return timings to 1541 protocol

1581COPY Commands

The 1581COPY program by Wolfgang Mosser (Womo) is supported by 64HDD and allows the emulator to treat the 1.44FDD as a 1581 drive indirectly. It should be noted that you must use DD disks as these are most compatible with the real 1581 disk drive. 64HDD will work with disk images transferred using this utility - it cannot directly read a 1581 formatted disk.

Before attempting to use 1581COPY from 64HDD make sure that your system is indeed compatible and capable of producing 1581 readable disks. I have in the past had problems with both incompatible FDD controllers and poor quality floppy disks. As 1581COPY is used as an external resource error information and control is limited. Control of 1581COPY is via two batch files, T1581.BAT and F1581.BAT, and these files together with 1581COPY.EXE need to be on the “path” searched by MSDOS.

The extended error information available is controlled within the two BAT files. If the 64HDD system directory is in a different location to that shown in these files, these files must be modified to reflect the new path. For example, the default is c:\64hdd\system, but if a RAMdisk is used the path may be e:\64hdd\system. Be sure to change all occurrences in both T1581 and F1581 files.

Command: Transfer D81 image to 1.44FDD using 1581COPY
Applicability:

☐ All ■ MSDOS ☐ D64 ☐ D71 ☐ D81 ☐ H64 (Native) ☐ T64 ☐ LNX
Syntax:
```
“T1581:filename.D81”
“T1581:”
```

Examples:
Transfer D81 image in current directory to 1.44FDD
```
OPEN 1,11,15,”T1581:DUCKS.D81”
CLOSE 1
```

Format 1.44FDD using 1581COPY but with no transfer
```
OPEN 1,11,15,”T1581:”
CLOSE 1
```

Errors:
- Error [180] returned for any system error
- Error [35] and appropriate message returned for other errors

Notes:
- D81 Disk image has to be in current directory (else path must be specified)
- The transfer is controlled by shelling to the T1581.BAT file. Modify this file as needed. You require Womo’s 1581COPY program (version greater than 0.52) to run this function. The emulator will not respond to other

commands whilst the transfer takes place (this is because 1581COPY is a stand alone program and not integrated into 64HDD)
- Support is only offered in MSDOS file system as a precaution against transferring a disk image which is currently active.

Command: Transfer 1581 disk in 1.44FDD to D81 with 1581COPY
Applicability:

☐ All ■ MSDOS ☐ D64 ☐ D71 ☐ D81 ☐ H64 (Native) ☐ T64 ☐ LNX
Syntax: `"F1581:filename.D81"`

Examples:
Transfer 1581 disk from 1.44FDD to D81 image in current directory with given name
```
OPEN 1,11,15,"F1581:TEST.D81"
CLOSE 1
```

Errors:
- Error [181] returned for any error detected
- Error [35] and appropriate message returned for other errors

Notes:
- D81 Disk image is written in current directory (else path must be specified)
- The transfer is controlled by shelling to the F1581.BAT file. Modify this file as needed. You require Womo's 1581COPY program (version greater than 0.52) to run this function.
- The emulator will not respond to other commands whilst the transfer takes place (this is because 1581COPY is a stand alone program and not integrated into 64HDD)
- Support is only offered in MSDOS file system as a precaution against transferring a disk image which is currently active.

Partitions and Unit Support (Shortcuts)

Both the Lt.Kernal and CMD Hard Drives support the concept of multi-unit support. Originally Commodore used the unit number to differentiate the two drives in a dual disk system, for example `LOAD"0:TEST",11` for drive0 and `LOAD"1:TEST",11` for drive1. These two third-party hard drives however implemented the unit numbering system to call up a predefined number of "partitions", or what were effectively images of floppy disk software. Both had disk units 0 through 255 available, with 0 being the current image. CMD uses unit 255 as the system partition.

64HDD also supports partitioning in this regard, but extends the concept further in a couple of ways:

1) units 0 to 999 are available (you should reserve units 900-999 for future usage if you wish them to be compatible)
2) units can point to disk images or MSDOS directories or paths, for example unit 10 can be defined to be your CDROM at d:\, whilst unit 5 can be you favourite game.
3) units 0 to 999 are mapped across all 8 possible device numbers supported by 64HDD, thus unit 8 can be available from device 8 and device 11 at the same time.

Unit numbers can be thought of as "shortcuts" and can be set up to alleviate a lot of keyboard typing.

Unit definitions are saved in the file CMDPARTN.TBL in the 64HDD "System Directory". This file is in ASCII format and may be edited using a text editor. The first five characters of each line are reserved [abc] for the unit number label. After this is the path definition. The units must be sequential listed and are left bank if undefined. For example:

```
-----
[009]
[010]c:\64hdd\utils
[011]
[012]c:\test.d64
[013]
-----
```

The partition file can also be modified using the appropriate 64HDD commands. Remember, that if the 64HDD "system directory" is stored on a RAMDisk for speed it should be copied back to a real disk by the GO64HDD batch file if changes are to be kept for future use. The following conventions should be adhered to:

Unit 0	Default "partition", re-definitions ignored
Unit 255	System "partition" for CMD compatibility, set to 64HDD "System Directory" location
Units 900-999	Reserved for future 64HDD functions

Command: Change partition to path defined by unit number
Applicability:

■ All ☐ MSDOS ☐ D64 ☐ D71 ☐ D81 ☐ H64 (Native) ☐ T64 ☐ LNX

Syntax: `“CPx”`
`“cPn”`

Examples:
Change to partition defined as unit 10
```
OPEN 1,11,15,”CP10”
CLOSE 1
```

Change to partition defined as unit 10
```
open 1,11,15,”cP”+chr$(10)
close 1
```

Errors:
- error [77] set if partition not found
- error [0] or error [2] set if partition was found and change successful

Notes:
- x is ASCII partition number and allows 0-999 values to be used
- n is binary partition number and allows only 0-255 values to be used

---

Command: Add current path /image to partition table
Applicability:

■ All ☐ MSDOS ☐ D64 ☐ D71 ☐ D81 ☐ H64 (Native) ☐ T64 ☐ LNX

Syntax: `“APx”`
`“aPn”`

Examples:
Set current path /image to partition table unit 20
```
OPEN 1,11,15,”AP20”
CLOSE 1
```

Set current path /image to partition table unit 20
```
open 1,11,15,”aP”+chr$(20)
close 1
```

Errors:
- error [77] set if partition not found
- error [0] or error [2] set if partition was found and change successful

Notes:

Command: Delete current partition table definition
Applicability:

■ All ☐ MSDOS ☐ D64 ☐ D71 ☐ D81 ☐ H64 (Native) ☐ T64 ☐ LNX
Syntax:
```
“DPx”
“dPn”
```

Examples:
Delete definition of unit 7
```
OPEN 1,11,15,”DP7”
CLOSE 1
```

Delete definition of unit 7
```
open 1,11,15,”dP”+chr$(7)
close 1
```

Errors:
- error [77] set if partition not found
- error [0] or error [2] set if partition was found and change successful

Notes:

Command: Information about partition
Applicability:

■ All ☐ MSDOS ☐ D64 ☐ D71 ☐ D81 ☐ H64 (Native) ☐ T64 ☐ LNX
Syntax:
```
“IPx”
“iPn”
```

Examples:
Get definition of unit 7 and assign to EM$
```
OPEN 1,11,15,”IP7”
INPUT #1, EN, EM$, ET, ES
CLOSE 1
```

Get definition of unit 7 and assign to EM$
```
open 1,11,15,”iP”+chr$(7)
INPUT #1, EN, EM$, ET, ES
close 1
```

Errors:
- error [0]

Notes:
- undefined partitions are returned as “.” signifying that no partition change would be made if called
- partitions 990-999 return “RESERVED”
- only partitions 0-255 can be requested using the binary method

Command: Load /save /open from /to unit path
Applicability:

■ All ☐ MSDOS ☐ D64 ☐ D71 ☐ D81 ☐ H64 (Native) ☐ T64 ☐ LNX

Syntax:
```
LOAD“pn:$dirspec”,device
LOAD“pn:filename”,device
LOAD“pn//dir1/dir2:filename”,device
SAVE“xpnfilename”,device
OPEN lfn,dev,sec,“x:filename,type,mode”
```

Examples:

Load default program from disk image associated with unit200
```
LOAD”200:*”,11
```

Load program “test” from path associated with unit205
```
LOAD”205:test”,11
```

Load program from path building on definition in unit4, note full progname required!!!
```
LOAD”4//64hdd/utils:errorchk.prg”,11
```

Load directory from path associated with unit100
```
LOAD”100:$”,11
```

Errors:
- error [77] set if partition not found
- error [62] set if file not found
- error [0] or error [2] set if partition was found and change successful

Notes:
- if unit definition is not a found the change is ignored and an attempt is made to load from the current path (ie unit 0)
- with the syntax “pn/dir1/dir2:” the last directory name terminates with a “:” and not “/” and the full progname (including type) is required

---

Command: Load partition table
Applicability:

☐ All ■ MSDOS ☐ D64 ☐ D71 ☐ D81 ☐ H64 (Native) ☐ T64 ☐ LNX

Syntax:
```
LOAD“$=P”,device
LOAD“$=P:,type”,device
```

```
    Type can include:  4 for 1541 (D64)
                           7 for 1571 (D71)
                           8 for 1581 (D81)
                           D64, D71, D81, T64
                           D* for all disk images
                           L for LNX
```

Examples:

Load complete partition table

```
LOAD”$=P”,11
```

Load partition table only with definitions to disk images

```
LOAD”$=P:,D*”,11
```

Errors:

-

Notes:

- partition table can currently only be called from a device attached to an MSDOS directory (ie not an image)
- partition information is displayed in wide format (not exactly CMD compatible)
- *type* only allows restriction of the types to be controlled. No naming pattern match is considered.
- partition definitions longer than the display name are abbreviated and are show with only the start and end portions of the definition

CMD Compatible Time and Date Commands

CMD FD and HD series devices support an option RTC device. 64HDD supports compatible commands, though the preference is to use the native commands as these are simpler to convert (for example to CBM native TI$) and have no Y2K related issues.

The CMD commands are provided only to ensure compatibility with programs written for CMD RTC equipped devices (not including SmartMouse). It must be pointed out that the CMD operating system has Y2K compatibility issues as it uses only a 2-digit year format. As a consequence setting the date using the CMD commands may set the PC's RTC to the wrong date. 64HDD makes the assumption that the year will be 2000+set_year, but this precludes initialising the system at an earlier date. If you wish to do this use the native commands. If you are still using 64HDD in the year 2100 you will also have problems, use the 64HDD native command to avoid these issues.

Command: **Read /set RTC in CMD ASCII format**
Applicability:

■ All ☐ MSDOS ☐ D64 ☐ D71 ☐ D81 ☐ H64 (Native) ☐ T64 ☐ LNX

Syntax:
```
"T-RA"
"T-WAdow. mo/da/yr/ hr:mi:se xM"
```

Examples:
See Notes.
```
10 OPEN15,11,15,"T-RA"
20 GET#15,A$:T$=T$+A$:IF ST<>64 THEN 20
30 CLOSE 15

     t$ contains the time in this order:
          "dow. mo/da/yr hr:mi:se xM"+chr$(13)

          day-of-week: 4 letters + one space
          SUN. MON. TUES WED. THUR FRI. SAT.
          mo  month: 01-12
          da  day
          yr  year
          hr  hour: 01-12
          mi  minute: 00-59
          se  second: 00-59
          x   A or P : AM/PM
```

Errors:
- Error [30] returned if time/date setting string the wrong length
- Error is [0] if length Ok

Notes:
- See examples bundled with the 64HDD software, or refer to the CMD user manuals.

Command: **Read /set RTC in CMD decimal format**
Applicability:

■ All ☐ MSDOS ☐ D64 ☐ D71 ☐ D81 ☐ H64 (Native) ☐ T64 ☐ LNX

Syntax:
```
"T-RD"
"T-WD"+9bytes
```

Examples:
```
See Notes, decimal format means byte is 0-99.
     byte 0:  day-of-week 0=sun, 1=mon...
     byte 1:  year
     byte 2:  month
     byte 3:  day
     byte 4:  hour (0-12)
     byte 5:  minute
     byte 6:  second
     byte 7:  AM/PM flap 0=AM  <>0 = PM
     byte 8:  chr$(13)
```

Errors:
- Always returns error [0] regardless

Notes:
- See examples bundled with the 64HDD software, or refer to the CMD user manuals.

Command: **Read /set RTC in CMD BCD format**
Applicability:

■ All ☐ MSDOS ☐ D64 ☐ D71 ☐ D81 ☐ H64 (Native) ☐ T64 ☐ LNX

Syntax:
```
"T-RB"
"T-WB"+9bytes
```

Examples:
```
See Notes, BCD format means high/low nibble is 0-9.
     byte 0:  day-of-week 0=sun, 1=mon...
     byte 1:  year
     byte 2:  month
     byte 3:  day
     byte 4:  hour (0-12)
     byte 5:  minute
     byte 6:  second
     byte 7:  AM/PM flap 0=AM  <>0 = PM
     byte 8:  chr$(13)
```

Errors:
- Always returns error [0] regardless

Notes:

- See examples bundled with the 64HDD software, or refer to the CMD user manuals.

Command: Load Directory with TimeStamp
Applicability:

■ All ☐ MSDOS ☐ D64 ☐ D71 ☐ D81 ☐ H64 (Native) ☐ T64 ☐ LNX
Syntax: `LOAD“$=T:dirspec,type”,device`

Examples:
Load directory from default path with CMD timestamp format
```
LOAD”$=T”,11
```

Load directory from default path with CMD Long timestamp format
```
LOAD”$=T,L”,11
```

*Load timestamped directory from default path with name matching files**
```
LOAD”$=Tfiles*”,11
```

Errors:
-

Notes:
- only MSDOS file system has timestamp information currently
- CMD date selection fields are not supported

Text Exchange (Cross-Platform Compatibility)

Let's face it; from time to time there is the need to exchange text data between the Commodore machine and the rest of the world. 64HDD is capable of performing on-the-fly ASCII text translations by simply opening the file with a {shift-type} rather than the normal {type} specifier. Alpha-translation is only performed.

The possibilities are endless, and combined with the other cross-platform functions within 64HDD, some level of co-operative processing is possible. Consider the following examples:

- Using a Cross-platform Compiler: write a batch file on your 64HDD machine that compiles your source code (saved on-the-fly to true-ASCII) into a PRG file. The batch file is called by 64HDD's `MSD:` command. Output from the compiler is redirected to the MSD.SEQ file in the system directory, and can be read using the on-the-fly ASCII translator.

- Use a PC finder tool to locate files on a CD: use one of the popular *whereis* utilities available on the PC by calling it with the MSD: command. The results can be viewed in the MSD.SEQ file in the system directory.

Notes:
- on-the-fly translations are modal, meaning that the last file open determines if it is on or off. Therefore the last file open will determine whether the ASCII translation occurs for all other files that are also open.
- 64HDD waits on the drive that has been set to have the system directory. As a result, MSD programs are most easily controlled by running a batch file that can specify the full path name of the utility and any associated input output files. Additionally the batch file can set any local environmental variables.
- Remember, the location of the MSD.SEQ file is in partition 255. If 64HDD users have configured their system properly, the partition number can be used as a universal way of locating the file.

"Multi-Disk" Compatibility

Some C64 programs were distributed on multiple 1541 disks. With some of these programs it is possible to copy the contents to a 1581 image and use all levels of a game for example without "flipping" the disk. However, many programs will not transfer to formats other than that of the original disk. Nor, do some programs allow you to enter disk commands so that you can re-assign which disk image is attached to the current disk. The problem also occurs when you have "program" and "data" disks, for example with a compiler, word-processor and GEOS. This has been a problem for CMD hard drive users for quite some time now.

To help in these situations, 64HDD now supports disk "flipping" at the push of a button. There are two ways to use this feature:

- You can add a switch to you X1541 or XE1514 connection. This suits users who do have a standalone 64HDD system. The modification involves connecting a switch to pin15 of the LPT plug and to ground (pins18-25). The X1541 cable may already use pin15 for the "detection" loop, but this link can be removed and the cable will still work. The XE1541 cable has no such link in place. This switch is the "disk-flip" button.
- You use the F12 key on systems which have a keyboard.

In order to use the "multiple-disk" feature, disk images or directories need to be assigned as partitions, or short-cuts. The "disk-flip" function is only available when device #8 is activated on 64HDD.

Partition/short-cut [001] through [010] take on a special meaning when the "disk-flip" button is pressed. Each press of the button causes 64HDD to search for the next valid definition. This definition is then attached to device #8 and becomes the default for subsequent accesses. If a partition/short-cut in the [001]-[010] range is undefined, 64HDD skips to the next. After partition/short-cut [010], 64HDD jumps loops back to [001].

Notes:

- Each press of the "disk swap" button when 64HDD is "idling" (that is twirling bar is rotating), will increment the "disk number" between 1 and 10. If a valid partition is assigned to this number, the partition is assigned to device #8. The partition can be a disk /tape /LNX image or native MSDOS partition
- Blank definitions are skipped. Therefore if a 2-disk game is used only partitions 1 and 2 should be defined. If 3 double-sided disks make up the game only partitions 1,2,3,4,5,6 should be defined. Hence a maximum "game" size is 5 double-sided 1541 disks or ten 1581 disks.
- When a valid partition (that is, "disk") is changed to, the corresponding number of beeps is given, for example 1 for disk1, 2 for disk2, etc. The current disk number should also be shown on the PC display (top line).

- The button should have no effect during loads, but may upset SEQ transmissions because 64HDD may idle between blocks. 64HDD does put the C64 into hold-off mode though during the swap, and so may not be a problem (except for transmission is now from another source)
- The 64HDD distribution includes a sample program to demonstrate clearing and assigning partition definitions (this has long been a feature of 64HDD). This could be modified to suit the particular program, and you could create one per game and include the drive number swap code.

Specifications Guide

Dependent Devices:

File system devices are termed dependent devices. The behaviour of the emulator depends upon the mode the file system is in.

For example, when a D64 image is logged the emulator will respond and emulate a 1541 drive (unless 1571 mode is specified by the appropriate CBM command). Because the mode is dependent, only applicable commands will be available and ROM/RAM maps will reflect the mode specified. A D64 image can of course be "seen" with both a 1541 and 1571 as in real life. A D81 image can only be accessed via a 1581 emulated interface.

Files within the MSDOS file structure are treated in a mode which is neither 1541/1571/1581. Here the file extension defines the type of file with the following conventions used:

PRG	Default /program file
SEQ	Sequential file
USR	User file (only PRG /SEQ format though)

Logging of a file with a disk, tape or archive image extension will switch modes to the appropriate device interface.

Files in the MSDOS structure other than the above types are treated as binary and can also be loaded. In the CBM directory listing they are identified as type "MSD" (abbreviation for MSDOS, though they may really be CBM binary and text files). For these to load the complete name (including file extension) must be used. The full name will be shown in the CBM directory listing.

64HDD has up to 8 dependent devices (devices 8, 9, 10, 11, 12, 13, 14 and 15) and not all need to be activated. Optionally a "boot floppy" in drive A: can be used to re-configure the system as required, or preferably 64HDD is set up to work in conjunction with batch file which respond to error_level commands sent by the emulator's EXIT command. The numbering of these devices cannot however conflict with real devices on the CBM serial bus. This conflict can easily occur if you have a SX64 or a C128D with internal drive(s). You will have to soft/hard change the number in these drives before starting (or restarting) the emulated drive. Each "initialised" dependent device needs to have a path assigned at 64HDD start-up, else a default "c:\" location is assumed. More than one device can be logged to a given path, but only one device should be logged to disk image (this is because a disk image may get corrupted if one device modifies the image whist the other is not watching).

64HDD File Systems Modes:

<u>CBM Disk Images (D64, D71, D81)</u>

- Read only of PRG files for (streamed transmission)
    - 35 track D64 format images
    - 70 track D71 format images
    - 80 track D81 format images
- Error setting for T&S error encoded disk images
- Refer to project specification sheet for intended capabilities.

<u>CBM Tape Images (T64)</u>

- Read only of directory "track" and PRG files for T64 format images
- No command channel support is planned (Error 8x will always be returned)

<u>Archive (LNX)</u>

- Read only of directory track and PRG files for LNX format images
- No command channel support is planned (Error 8x will always be returned)

<u>Native Format</u>

- Intended to be a large T&S compatible image replicating much of the specification of a 1581 drive. This format is intended to give "hard disk" size capabilities in a CBM compatible drive.
- Not yet supported refer to project specification sheet for intended capabilities.

<u>MSDOS File System</u>

- Read and write of PRG, SEQ and USR files (OPEN 1,11,2,"filename,type" where type is "P", "S" or "U")
- Name is searched as requested, but if not found an attempt will be made to append the appropriate extension (default is PRG)
- All other non-disk files are streamed from start to finish as if SEQ
- Command channel support for working with directories, scratching, copying and modifying files.
- Refer to project specification sheet for intended capabilities.
- By default, 64HDD will only create files with valid 8.3 names, and will substitute legal file extensions where not provided. Generally upon reads (files or directories) will be as requested, and only upon failure will substitution be tried. For example, if a file `"TEST FOR ERROR"` is saved, 64HDD will save it as `"TESTFORE.PRG"` in order to obey MSDOS rules. An attempt to load it by the name `"TEST FOR ERROR"` file be attempted, but as this will fail a retry with the name `"TESTFORE.PRG"` will be made without an error being flagged to the CBM user. For file or directory deletion or copying the complete MSDOS filename (with extension) or a wildcard is required.
- If the LongFileName (LFN) system has been correctly installed, 64HDD will allow the creation of valid 16.3 filenames. The LFN system uses a naming /file format which is Windows95 compatible and as such, files can be transferred to /from a Windows95 system.

MSDOS LongFileName Support:

With the installation of Odi's LongFileName (LFN) utilities, 64HDD can support most CBM 16character filenames. This improves CBM software compatibility. Version 1.55 of the LFN tools is required with v0.6a3 and higher of 64HDD. These upgraded LFN tools now allow long names to be read on ISO9660 format CD-ROMs.

In LFN mode, 64HDD uses 16.3 format names instead of the 8.3 format dictated by MSDOS. The 16.3 format is subset of the LFN format introduced by Windows95. However, Windows95 or MSDOS7x did not provide tools that allow the LFN files to be created or copied in any environment except for when Windows95 is actually running in its Graphical Mode.

There are several tools available for free which allow the Windows95 LFN format to be used in MSDOS, each having advantages and disadvantages. 64HDD has chosen to use Odi's LFN Utilities. The LFN utilities included in the Zip distributed with 64HDD are the only LFN utilities that will be verified for correct operation with 64HDD.

MSDOS LFN support has many benefits and so it is likely to be important to the 64HDD user, but before it will work correctly the LFN utilities need to be installed correctly. If the utilities are not correctly installed, then there is the risk that 64HDD or the tools will corrupt the directory structure. Before you proceed it is recommended to read the LFN.TXT file distributed in the LFNFILES Zip.

There are three important steps to follow:
1) Install the LFN utilities into the correct directory
2) Unzip and/or LCOPY the `LFN$$$16CHAR$LFN.$$$` file to the correct directory
3) Check that the CONFIG.SYS file has sufficient BUFFERS and FILES available

64HDD performs several tests before it will allow LFN mode to be selected. These tests should catch 95% of possible installation errors, and should errors be found a double "beep" and prompt will be displayed.

LFN mode is selected <u>only</u> when the `–lfn` command line switch is used when starting 64HDD, and should be added to the relevant line in the GO64HDD.BAT file.

<u>Detailed LFN Set-up:</u>

Step 1: The LFN utilities are distributed in a Zip file called LFNFILES.ZIP. This Zip contains one LongFileName file which is used by 64HDD as the name template for creating all new LFNs. To create this LFN file a Windows95 version of WinZip is required (ie versions 6+). Unfortunately there is no way around this as I have not found an LFN Unzip tool for MSDOS. The

LFN template file is called `LFN$$$16CHAR$LFN.$$$` and is the only file which must be created under the Windows95 LFN system. The rest of the Zip can be expanded with DOS "PKUNZIP –d" for example.

Only a subset of Odi's LFN utilities are included and these must be located in the directory from which 64HDD.EXE is executed. Additionally, the directory must be on the MSDOS PATH as 64HDD will need to find them. The **must have** utilities are: LDIR.EXE, LREN.EXE, LCOPY.EXE and LDEL.EXE

64HDD checks to see if these files are correctly located.

Step 2: The LFN template file `LFN$$$16CHAR$LFN.$$$` must be moved to the 64HDD system directory as defined by `+sysdir` or the default `c:\64hdd\system` location if not specified. The file can only be correctly copied using the LCOPY utility or Windows95. If necessary you will LCOPY it to a floppy disk, and then LCOPY it to the location. If you are using a RAMDISK for the 64HDD system directory, you will also need to LCOPY the file to the RAMDISK. You can use LDIR to see that the LFN is as shown above. The MSDOS name should be LFN$$$~1.$$$

If you used an MSDOS unzip tool it is likely that there will be a LFN$$$16.$$$ file in the 64HDD system directory. This must be deleted. Use LDEL in preference to DEL as it will correctly delete any LFN files. You may need to use DEL if LDEL says the file was not found (that is no LFNs were found).

Step 3: Check the setting of the BUFFERS and FILES in the CONFIG.SYS file. LFNs need lots of channels open for assembling the filename. Be generous and set FILES=50 and BUFFERS=17,0

You will need to re-boot the PC for these new settings to take effect.

Notes, Tips and Suggestions:

- You can create LFNs in the root directory of a disk, however should errors result the disk structure may be damaged. It is perhaps best to limit these files to floppy disks which can easily be re-formated.
- If you've decided to use LFNs with 64HDD use LCOPY, LREN and LDEL in preference to DEL, REN and COPY when doing you disk house-keeping. This will ensure directory entries do not get corrupted.
- A corrupted directory entry can only be cleared by using ScanDisk for Windows95 or by removing the entire directory. For this reason it may be recommended practice that LFNs be created in subdirectories and not the root directory of your Hard Disk.
- The LFN tools only work on drives which use the FAT system or ISO9660 CD file system (LFN tools v1.55). This generally includes Floppy and Hard

Disks, and some CD-ROMs. Only CD-ROMs written using the ISO9660 standard will work. {note: directory names need to follow the 8.3 format as per the following note}

- 64HDD only implements LFNs for filenames. Directories are limited to 8character names and LFN named directories are only accessible by their shortname.
- Disk /tape /LNX image names are also assume to be in 8.3 format. If not the name may be truncated and the extension incorrectly displayed. The image may still be attached to, but only if you know the complete name.
- Automatic recognition of files by their extension type is only possible if the three character extension is capitalised in the LFN format when viewed with LDIR (ie "test file.PRG" is recognised as a PRG, "test file.prg" is identified as a MSD file). 64HDD creates capitalised PRG, SEQ and USR extensions. Files with non-capitalised extensions must have their whole name specified or a wildcard used. Capitalisation refers to how the names appear when viewed with LDIR and not how they appear on the CBM machine.
- Some characters are still invalid in the Windows95 LFN system and 64HDD checks and removes these. This includes / and \ which are directory specifiers and other punctuation used for redirection. However, spaces and periods are allowed as part of the 16.3 name. Upper and lower characters are accepted and used in the name. 64HDD, when LFN is enabled, does some translation for PETASCII characters such that they are mapped to the true ASCII.
- When LFN mode is used, directories may take a little longer to generate before they are sent to the CBM computer. This is normal and can be improved by the use of a RAMDISK or SMARTDRIVE cache.

Independent Devices:

These devices are not file system devices. They are for example devices that receive and send data, but are not used for storing file data. A good example is the Real-Time-Clock (RTC) in the PC controller. The time /date for this device can be set via the command channel regardless of whether in 1541 /1571 /1581 /etc mode (none of which support RTC commands). The concept of "independent" devices is used also for interrogating the functioning of the controller and emulator without having to work from a keyboard and monitor attached to the PC controller.

A PC compatible mouse may be attached to a 64HDD serial or PS2 port. With the appropriate MSDOS mouse driver installed, the position of the mouse and button status can be read by the CBM computer and the information used in the program you write.

Other device independent functions will be added as concepts are defined.

X1541 Cable Specification:

The program requires the following adaptor cable to be connected between your parallel printer port (LPT1/LPT2/LPT3) and a commodore disk drive. This cable is also known as X1541 cable because it was used the first time by a program with this name. Nowadays, it is used by many programs. At present only the original X1541 and the XE1541 cable schemes are supported. If you have a newer machine with on-the-mainboard LPT ports chances, are it will not work with the X1541. To overcome this limitation you need an older LPT card installed in a slot as a secondary LPT port or an XE1541 interface. (X1541 schematic below also shows optional tape connector for use on the C64S emulator. Alternate details are in the README.TXT file. X1541 and XE1541 schematics can also be found in the Star Commander documentation or on their website.)