
Pro-DOS v2.0

A CP/M 2.2 workalike OS for the SAM Coupé

User Manual

Software License

Pro-DOS v2.0 is freeware, released under the [zlib/libpng license](#) (Zlib)

Pro-DOS v2.0, a CP/M 2.2 workalike OS for the SAM Coupé

Copyright © 1991-2014 Chris Pile

This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

- 1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.*
- 2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.*
- 3. This notice may not be removed or altered from any source distribution.*

*Chris Pile <chris@digital-reality.co.uk>
May 2014*

Acknowledgements

- Wayne Weedon:** Wayne recovered my original v1.9 source disks and has been involved with Pro-DOS since 1991.
- Simon Owen¹:** Simon generously provided me with some crucially needed hardware during v2.0's development. He is also the developer of the superb SimCoupé emulator.
- Steve Parry-Thomas:** Steve has been a supporter of Pro-DOS since v1.9 way back in 1991. He also maintains the Pro-DOS archive² to preserve disks and documentation.
- Andrew Collier:** Andrew wrote PyZ80³, the Z80 assembler I used to build v2.0 and an assembler I would recommend to other developers of SAM Coupé software.
- Colin Piggot⁴:** Colin produces the superb Trinity Ethernet Interface and other SAM Coupé hardware/software. He is also one of the most active supporters of the SAM Coupé.
- Brian Gaff:** BG Services were the original publishers of Pro-DOS and Brian, via Wayne, kindly let me know that I am now free to do as I please with Pro-DOS.

¹ <http://simonowen.com>

² <http://www.samcoupe-pro-dos.co.uk>

³ <http://www.intensity.org.uk/samcoupe/pyz80.html>

⁴ <http://www.samcoupe.com>

Pro-DOS Mk II - The Rebirth!

Pro-DOS v2.0 may share the name of its predecessor, but that is pretty much all it shares. Initially the plan was to take version 1.9, improve the screen routines and add *official*¹ ATOM-Lite support.

As often happens, what began as a trivial exercise turned into a complete rewrite! v2.0 is very much its own entity and shares virtually none of v1.9's original code. This rebirth also included a complete rewrite of the *system utilities* as well as the user manual.

Pro-DOS v2.0 provides superior performance to its older relative and remains backwardly compatible. In fact, Pro-DOS v2.0 is closer to how I should have written v1.9 back in 1991!

This manual focuses on using Pro-DOS rather than programming for it and, as such, omits most of the technical aspects of the system.

Pro-DOS v2.0 - Features

- Better CP/M 2.2 compatibility
- Up to eight simultaneously visible disk drives - Drives **A:** through to **H:**
- CompactFlash support using the ATOM-Lite (right-hand drive bay)
- Dual CF adapter support in the ATOM-Lite - seen as two separate drives
- MMC/SD/SDHC flash support using Quazar's Trinity Ethernet Interface
- 100% compatible with the *AL-Patch* format² used by AL-Pro-DOS v1.9
- Pro-DOS can optionally boot from its own format 720k CP/M disks
- If your SAM has an AL-BOOT ROM Pro-DOS can also boot from CF cards
- Up to 64-Gigabytes (65535 *logical disks*) on CF/MMC/SD/SDHC cards
- Will use any external RAM packs to provide up to four 1MB RAM Drives
- Improved performance, with particular attention paid to screen output
- More accurate *Heath/Zenith H19/Z19/DEC VT52* terminal emulation
- Better command-line editing functions
- Command-line can accept user areas - so **DIR A4:*.COM** is valid
- Can simultaneously change drives and user areas at the command prompt
- MS-DOS-like command history buffer for the eight most recent commands
- Retro *green-screen* look with funky flashing cursor - changeable of course!
- Will save/restore your chosen screen colours to/from the Trinity EEPROM
- System bell sound makes a better "*ding*" and no longer pauses the system
- The **DIR** command now outputs in alphabetically sorted columns
- Better memory use to allow v2.0 to retain v1.9's internal RAM Drive sizes
- A large (63238 byte) **T**ransient **P**rogram **A**rea for external programs
- The separate *system files* disk used by v1.9 is no longer required
- All *system files* are pre-installed to the internal RAM Drive at boot
- **COPY** and **DUMP** commands are now built-in, and not external programs
- All internal commands accept wildcards, including **REName** and **COPY**
- Ability to software-swap a pair of drives, so **A:** can be **B:** and vice versa
- **FATREAD.COM** for importing files from PC formatted *FAT12/16/32* media

¹ Rather than the *AL-Patch* version which sacrificed v1.9's internal RAM drive

² Edwin Blink's container format allowing BDOS and Pro-DOS to share flash cards

Table of Contents

1 - Getting Started	1
1.1 Producing a BOOT Disk	1
1.2 What is Pro-DOS?	1
1.3 CP/M 2.2 Compatibility	2
1.4 Floppy Disk Performance	3
2 - Introducing Pro-DOS	4
2.1 Booting	4
2.2 Disk Drives	6
2.3 Command Line	8
2.3.1 Changing Drives and User Areas	9
2.3.2 Entering Commands	10
2.3.3 Filenames, Programs and Drive Prefixes	11
2.3.4 Wildcard Filenames	13
2.3.5 Printer Echoing	14
2.3.6 Scroll Pausing	14
2.3.7 Aborting	14
3 - Internal Commands	15
3.1 CLS	15
3.2 COPY	15
3.3 DIR	16
3.4 DIRS	18
3.5 DISK	18
3.6 DRIVES	19
3.7 DUMP	19
3.8 ERA	20
3.9 FEEDS	21
3.10 PALETTE	21
3.11 REN	23
3.12 SAVE	24
3.13 SWAP	25
3.14 TYPE	26
3.15 USER	27
4 - External Utilities	28
4.1 ATTRIB.COM	29
4.2 BATCH.COM	31
4.3 CRC.COM	33
4.4 FATREAD.COM	35
4.5 FORMAT.COM	40
4.5.1 Floppy Disks	40
4.5.2 Flash Cards	41
4.5.2.1 Formatting Native Pro-DOS Cards	42
4.5.2.2 Formatting Legacy BDOS Cards	43
4.6 SAMREAD	44
5 - Appendix A - H19/Z19/VT52 Terminal Codes	45
6 - Appendix B - Keyboard Return Codes	48

1 - Getting Started

A SAM Coupé (256k RAM is fine) with at least one floppy drive is recommended to run Pro-DOS v2.0, from now on simply known as Pro-DOS or v2.0. Alternatively, you can run Pro-DOS using Simon Owen's excellent [SimCoupé](#) emulator.

Those already familiar with Pro-DOS v1.9 should still take the time to read this manual at least once. v2.0 has a number of changes and features that differ from v1.9 and to get the best out of v2.0 it is useful to know what these are.

v2.0 is completely self-contained and does not require any of v1.9's disks, utility programs or documentation.

A basic knowledge of CP/M, using a command-line environment and also a basic understanding of MS-DOS (for the *FATREAD* utility) will prove useful when using v2.0.

1.1 Producing a *BOOT* Disk

v2.0 comes as a single **.DSK** disk image (named **Pro-DOS-v2.dsk** in the distribution .ZIP) and this image can be used directly in the SimCoupé emulator.

If you wish to use Pro-DOS on a real SAM the first thing you must do is copy the **.DSK** image to a real floppy. The tool of choice here is Simon Owen's excellent [SAMdisk](#) utility. Once the disk image has been copied, insert the disk into your SAM and press **F9** to boot.

1.2 What is Pro-DOS?

Pro-DOS is an operating system (OS) providing compatibility with the original *Digital Research CP/M 2.2* system. This means a whole world of software designed to run under CP/M 2.2 is available on your SAM. Pro-DOS is a CP/M 2.2 workalike operating system.

When searching out CP/M software it is worth checking to see if it was designed to run under CP/M 2.2. Some programs require a later version of CP/M, commonly known as *CP/M 3* or *CP/M PLUS*, and these are likely to fail if you try and run them in a CP/M 2.2 environment. However, CP/M 2.2 is by far the most popular version and has the largest number of programs available.

Some CP/M programs must be *installed* for the system/machine you intend to run them on. This usually means providing the terminal (screen) configuration codes and sometimes the cursor-key codes returned by the keyboard. Programs requiring installation usually provide tools and instructions. It is normally a simple, one-off task.

Appendix A contains information on the terminal (screen) escape sequences. Pro-DOS emulates *H19/Z19/VT52* terminals.

Appendix B contains information on the codes returned by the cursor and delete keys to programs running under Pro-DOS.

Pro-DOS is independent of the SAM ROM and DOS and runs on any SAM configuration regardless of RAM size and/or storage devices attached. However, as it is a disk operating system having at least one mass storage device is recommended!

Pro-DOS will self-boot providing it is the first program saved on a floppy disk. This means you do not have to pre-boot a SAM, BDOS or MASTERDOS disk to load Pro-DOS. Section ([2.1](#)) describes the various ways Pro-DOS can be made to boot.

After booting, Pro-DOS automatically adjusts to make the best use of the resources found in your SAM. If you have a 256k single-drive SAM, or a 512k SAM with a floppy drive, twin CF ATOM-Lite, Trinity Interface and four external 1MB RAM packs, it adapts accordingly.

After Pro-DOS has booted it will display a system information screen to show all of the drives available to you.

1.3 CP/M 2.2 Compatibility

Every effort has been made to ensure Pro-DOS is as compatible as possible to CP/M 2.2 and v2.0 offers higher compatibility than v1.9.

However, as Pro-DOS doesn't use any of *Digital Research's* original CP/M 2.2 code and, unlike true CP/M 2.2, it also runs in banked RAM, there could be times where compatibility is compromised. To minimise this, any part of the system that should be visible to external programs remains so. However, programs that rely on patching the CCP¹ and/or BDOS² code will certainly fail, as these areas are not visible to external programs.

¹ Console Command Processor - part of the OS responsible for user interaction

² Basic Disk Operating System - part of the OS responsible for disk operations

It's worth pointing out that any external programs that fail may not be true CP/M 2.2 programs!

A program might have been written to run under one of several CP/M 2.2 CCP extensions, such as the popular *ZCPR* system. Any documentation that came with the program should inform you.

However, not all programs designed to make use of CCP extensions will fail. Most will detect that they are running under plain CP/M 2.2 and adjust accordingly. An example here is a popular CP/M word processor called *ZDE.COM*. Although ZDE can use ZCPR features it will also run quite happily under plain CP/M 2.2. This is an example of a well-behaved program. Not all programs are well behaved!

Another potential pitfall would be programs designed to use certain hardware specific to a particular system. For example, a program designed to talk to the hardware clock on an *Osborne I* will almost certainly not perform as expected!

However, such a program would also fail when run on *any* CP/M 2.2 system other than the one it was designed for.

It's worth mentioning that many of the programs found in the huge *Walnut Creek* and *Oak* archives¹ are specific to certain types of CP/M hardware. These might prove problematic when trying to run them on different CP/M hardware such as the SAM Coupé.

It is always wise to read documentation that comes with a program, and look for any assumptions made about specific hardware.

Your aim should be to run programs designed for generic CP/M 2.2, and doing so usually means it will be a painless task getting them up and running under Pro-DOS.

1.4 Floppy Disk Performance

When transferring **.DSK** or **.CPM** disk images to a floppy disk for use on a real SAM you should use [SAMDisk](#) with a version \geq v3.5.

SAMDisk $<$ v3.5 does not write the correct interleave for Pro-DOS and accesses to disks produced by $<$ v3.5 will be noticeably slower.

For best floppy disk performance in Pro-DOS you should use disks formatted by the system's own *FORMAT.COM* utility.

¹ Huge archives of CP/M programs which can be downloaded freely on-line

2 - Introducing Pro-DOS

Pro-DOS v2.0 comes as a single self-contained program on one disk; unlike v1.9, which had separate boot, and *system files* disks.

v2.0 comes supplied as a single **.DSK** disk image and those wishing to use it on their real SAM Coupé, rather than on the SimCoupé emulator, should also download a copy of Simon Owen's excellent *SAMdisk* tool. Producing a *boot disk* was covered in section [1.1](#).

2.1 Booting

Pro-DOS gives you a number of booting options:

1. As a loadable program from a *SAMDOS* format disk¹
2. As a self-booting DOS from a *SAMDOS* format disk²
3. As a self-booting DOS from a *Pro-DOS* format disk³
4. As a self-booting DOS from a CF card in the *ATOM-Lite*⁴

Option 2 will be the most familiar method. Simply format a floppy disk using any *SAMDOS* on your SAM. Don't make it a DOS disk though; it should be left blank. Next, copy Pro-DOS (it is a single code-type file) to the disk, ensuring it is the first file saved. This is the same technique used to make bootable *SAMDOS* disks.

This disk will now automatically boot Pro-DOS when it is placed in your SAM and you press **F9**.

If you prefer to load and run programs manually, as in option 1 above, then **LOAD** the Pro-DOS code file to address **32768** and run it using **CALL 32768**. The Pro-DOS code file saved on the original v2.0 disk image is set to auto-run. If for some reason it doesn't, then the above addresses always apply.

Once Pro-DOS has booted it detects the storage devices and RAM configuration of your SAM and presents a breakdown of this on the initial start-up screen.

The following page shows examples of three SAM configurations, the first being a 256k SAM with a single floppy drive:

¹ *SAMDOS* is used generically to mean *SAMDOS*, *BDOS* or *MASTERDOS*

² Make sure Pro-DOS is the first thing saved on the disk

³ See the section detailing the Pro-DOS *FORMAT* utility

⁴ Requires that your SAM contains the AL-BOOT ROM


```

Pro-DOS v2.0                                     TPA: 63238 bytes
Programmed by Chris Pile
(C) Digital Reality 1991-2014

Drives: A: = RAM Drive, Size: 124k - Internal
        B: = Floppy

A0>_

```

Next is a 512k SAM with one floppy drive, an ATOM-Lite with two CF cards and four 1MB external RAM packs:

```

Pro-DOS v2.0                                     TPA: 63238 bytes
Programmed by Chris Pile
(C) Digital Reality 1991-2014

Drives: A: = RAM Drive, Size: 380k - Internal
        B: = Floppy

        C: = ATOM-Lite: MASTER Hitachi CV 7.2.2
              7 x 1016k logical disks - Native Pro-DOS

        D: = ATOM-Lite: SLAVE TOSHIBA THNCF128MBA
              122 x 1016k logical disks - Unformatted

        E: = RAM Drive, Size: 1016k - External
        F: = RAM Drive, Size: 1016k - External
        G: = RAM Drive, Size: 1016k - External
        H: = RAM Drive, Size: 1016k - External

Change logical disks on CF/SD/MMC cards using the "disk" command
An unformatted CF/SD/MMC card is not available until formatted

A0>_

```

Finally, a 512k SAM with one floppy drive, an ATOM-Lite with one CF card, a Quazar Trinity Interface with SDHC card and a single 1MB external RAM pack:

```

Pro-DOS v2.0                                     TPA: 63238 bytes
Programmed by Chris Pile
(C) Digital Reality 1991-2014

Drives: A: = RAM Drive, Size: 380k - Internal
        B: = Floppy

        C: = ATOM-Lite: SLAVE TOSHIBA THNCF128MBA
              122 x 1016k logical disks - Legacy BDOS

        D: = TRINITY: SDHC Card: 7.9GB Capacity
              7579 x 1016k logical disks - Native Pro-DOS

        E: = RAM Drive, Size: 1016k - External

Change logical disks on CF/SD/MMC cards using the "disk" command

A0>_

```

The **TPA: 63238 bytes** information shown on the start-up screen is the amount of RAM (or **T**ransient **P**rogram **A**rea) available to programs running under the system. Pro-DOS v2.0 offers one of the largest TPA's of any CP/M 2.2 system. The TPA amount remains constant, regardless of the amount of RAM installed in your SAM.

2.2 Disk Drives

Pro-DOS v2.0 supports up to eight drives, and allows you to perform file operations between any of them at any time.

Pro-DOS always provides you with an internal RAM Drive¹, and it becomes the default (**A:**) after boot. If you prefer the RAM Drive wasn't drive **A:** you can change it for the duration of your session using the *SWAP* command explained in the "Internal Commands" section of this manual.

The RAM Drive comes pre-installed with utility programs, removing the need for the separate *system files* disk required by v1.9.

A 256k SAM will get a 124k RAM Drive, and a 512k SAM gets 380k.

Pro-DOS supports any of the following physical drive layouts:

- No physical drives - only the internal RAM Drive.
- Single floppy drive
- Twin floppy drives
- No floppy drives and an ATOM-Lite interface²
- Single floppy drive and an ATOM-Lite interface
- No floppy drives and a Trinity interface
- Single floppy drive and a Trinity interface
- Twin floppy drives and a Trinity interface
- No floppy drives, a Trinity interface and an ATOM-Lite interface
- Single floppy drive, a Trinity interface and an ATOM-Lite interface

The ATOM-Lite can have either a single or a dual CF-adapter. On dual adapters Pro-DOS treats each CF card as a separate drive.

Pro-DOS makes use of any 1MB external RAM packs attached by using each one found as a separate 1MB RAM Drive.

Pro-DOS imposes an eight drive limit, so if you have a floppy, the internal RAM Drive, an ATOM-Lite with dual CF cards, a Trinity with a flash card and four external RAM packs then Pro-DOS will omit one of the RAM packs to bring down the total drive number to eight.

¹ An area of RAM reserved as a fast (but volatile) virtual disk drive

² Pro-DOS requires that the ATOM-Lite is installed in the right-hand drive bay

RAM Drives are used exactly the same as physical drives. But remember that their content is lost when you switch off your SAM!

v2.0 floppy disks use the same format as v1.9, which is also the same as the *Amstrad PCW9256*¹. Disks have a data area of 720k with 706k available for storage. Shown as 720k/706k from now on.

*CF/MMC/SD/SDHC*² cards in the ATOM-Lite and Trinity interfaces are divided into a number of *logical disks*. Any of the logical disks can be selected using the *DISK* command described in the "*Internal Commands*" section of this manual.

Users of BDOS³ on the SAM will already be familiar with logical disks, BDOS calls them records and uses the *RECORD* command to change them. *DISK* in Pro-DOS is synonymous with *RECORD*.

Pro-DOS treats flash cards in the ATOM-Lite and Trinity as fixed hard drives. "Hot Swapping" is not supported!

When Pro-DOS boots it tries to detect the format of any flash cards. The format can either be native Pro-DOS or BDOS. If it is neither then Pro-DOS will report an unformatted card. This doesn't mean empty, it simply means Pro-DOS could not reliably detect its format.

Automatic detection of cards that have had BDOS or Pro-DOS CP/M records randomly injected into them via SAMDisk will not be reliable. Reliable detection relies on cards that have been previously formatted by BDOS or Pro-DOS.

Pro-DOS format logical disks share the same structure as external 1MB RAM Drives, 1024k with a total of 1016k usable storage space.

BDOS format logical disks use a container structure devised by Edwin Blink⁴ allowing standard CP/M disks to reside inside BDOS records. They share the same 720k/706k format as CP/M floppies.

Pro-DOS can also format individual 720k/706k CP/M containers on BDOS format cards, allowing you to mix CP/M and BDOS logical disks on one BDOS format card.

See *FORMAT* in the "*External Utilities*" section of this manual for more information on disk formats and the formatting and booting options available for floppy disks and flash cards in Pro-DOS.

¹ 160 tracks (80 per side) of 9 sectors, 1 reserved track, 256 directory entries

² Non-volatile flash media that Pro-DOS can use in either the ATOM-Lite or Trinity

³ A DOS for the SAM Coupé allowing the use of hard drives and flash cards

⁴ The creator of BDOS and the ATOM-Lite interface

2.3 Command Line

The final thing you will see on the start-up information screen, after the main drive information, is:

A0>_

This is the *command prompt* and shows you the drive currently selected and also the user area within that drive - user areas are described later. After the > symbol you will notice a flashing cursor. The cursor marks the beginning of *the command line*, which is your interface with Pro-DOS.

The command line is how you issue system commands, how you change drives and user areas and how you run external programs. In fact, pretty much everything you do in Pro-DOS will require you to enter a command of some description.

There are a number of basic editing functions available when you are at the command line, including a history buffer containing the last eight commands entered.

The list below shows the editing functions available:

- | | | |
|-----------------------|---|---|
| • Left Arrow | - | Move cursor one character left |
| • Shift + Left Arrow | - | Move cursor one word left |
| • Right Arrow | - | Move cursor one character right |
| • Shift + Right Arrow | - | Move cursor one word right |
| • Up Arrow | - | Move up in the command history |
| • Down Arrow | - | Move down in the command history |
| • DELETE | - | Delete one character to the left of the cursor |
| • Shift + DELETE | - | Delete one character to the right of the cursor |
| • ESCape | - | Clears the entire command line |
| • CNTRL + Q | - | Moves cursor to the start of the line |
| • CNTRL + W | - | Moves cursor to the end of the line |
| • CNTRL + R | - | Delete everything to the left of the cursor |
| • CNTRL + T | - | Delete everything to the right of the cursor |
| • CNTRL + P | - | Toggle screen-to-printer echoing ¹ |
| • CNTRL + S | - | Toggle scroll pausing ² |

These editing functions are also available to external programs that use the *Read Console Buffer* system call. So you may find you can use these functions in some programs.

The command history functions, however, are only available while you are at the system command prompt.

¹ Screen output also echoes to a printer when "P" is shown in the top right corner

² Scrolling will pause after each screen when "S" is shown in the top right corner

2.3.1 Changing Drives and User Areas

After booting, the command prompt will show drive **A:**, user area **0** which, until changed, becomes the default drive and user area for all subsequent disk operations.

User areas are covered later on in the manual, but for now think of them as 16 (0-15) independent areas on the same physical disk.

Almost like subdirectories but without the flexibility!

You can change to any available drive in your system by typing the drive letter followed by a colon. For example, to move to drive **B:** you would type:

B:

Followed by the *RETURN* key. Character case on the command line is not important and **b:** would have been valid.

When you change drives you remain in the same user area. For example, if you are in user area **5** on drive **A:** and move to drive **B:** your prompt would look like:

B5>_

Changing user areas usually requires the *USER* command. Pro-DOS offers this command and it is described in the "*Internal Commands*" section of this manual, along with more on user areas.

Pro-DOS allows you to change user areas using a shortcut similar to changing drives. So to change to user area **13** you can type:

13:

The colon is important and must be included and you follow it by pressing *RETURN*. If you were on drive **A:** when you issued the user change your prompt would now look like:

A13>_

Showing that you are in user area **13** of the same drive. Most of the time the user area does not change when you change drives, and the drive does not change when you change user areas.

However, Pro-DOS allows you to simultaneously change the drive and user area in one shortcut. This is done by combining the drive letter and user area number.

For example, entering:

B4:

Would change the drive to drive **B:**, and the user area to **4**. The command prompt would change to indicate this and would look like:

B4>_

Order of the individual elements is important, with the drive letter specified first and no spaces between drive, user area or colon.

So, **4B:** would not be valid and neither would **B 4:**

Any changed drive and/or user area becomes the permanent default for all subsequent disk operations until you, or an external program, changes them again.

Suitable error messages are displayed if you try selecting a drive that, for some reason, is unavailable.

An error message is also displayed if you try selecting a user area >15. User areas are always between 0 and 15.

2.3.2 *Entering Commands*

Issuing any of the system's internal commands and the running of executable *.COM* files are all performed via the command line. You enter commands by typing what you require and pressing *RETURN*.

To recap, character case is not important and there are a number of editing features available as listed on page 8. You can hold down the *SHIFT* key when typing if, for some reason, you did require uppercase characters to be entered.

Uppercase can be maintained by pressing the *CAPS* key. The cursor will be shown as a solid block when in *caps-lock* mode. Just press *CAPS* again to revert back to lowercase.

You can recall up to eight previously entered commands by pressing *cursor up* or *cursor down*. MS-DOS users will be familiar with this.

Each new command entered will push the oldest one out of the command history buffer, meaning Pro-DOS maintains a list of the eight most recent commands.

The command line has a 70-character limit when at the command prompt. External programs using the system's *Read Console Buffer* call are not subjected to the same limit.

2.3.3 *Filenames, Programs and Drive Prefixes*

Filenames in Pro-DOS are the same as those found in other CP/M systems and consist of two parts. The first part is the filename and the second part is the filename extension. A full stop is used to separate the two parts and most filenames will have some sort of filename extension. The first part may be up to eight characters in length, and the second up to three.

Spaces and the characters <>, :=|[]; are not allowed in filenames.

The list below are all valid filenames:

```
README.TXT
RUN_ME.COM
SOMEDATA.D
HELLO
MYSOURCE.ASM
```

Filenames that have a *.COM* extension (like *RUN_ME.COM* above) are executable programs and could be anything from a game of chess to a word processor. Most of the *system files* pre-installed to the internal RAM Drive are executable CP/M programs.

Think of *.COM* as an abbreviation for **COM**mand and, as a result, *.COM* files automatically run once loaded.

To run a particular program you don't have to include the *.COM* part of its name (although you can), just enter everything up to the full stop. Therefore, to execute the program *RUN_ME.COM* from the examples above the command prompt would look something like:

```
A0>run_me
```

Notice the *.COM* part has been omitted. After typing a program's name you simply press *RETURN*. Then, providing the file actually exists, Pro-DOS loads it from the disk and runs it.

You can access files from disks in other drives and in different user areas without permanently selecting the drive or user area. To do this you prefix the filename with the required drive letter, or drive letter and user area, or simply the user area. Then follow any of those combinations with a colon and, finally, the filename.

The list below shows a number of valid ways to load and run the *RUN_ME.COM* program from different locations in the system:

```
1 .... A0:>b:run_me
2 .... A0:>b7:run_me
3 .... A0:>10:run_me
```

1. Would search for and load *RUN_ME.COM* from the current user area (**0**, as shown in the *command prompt*) on the disk in drive **B**:

2. Would search for and load *RUN_ME.COM* from user area **7** on the disk in drive **B**:

3. Would search for and load *RUN_ME.COM* from user area **10** on the disk in the current drive (**A**;) as shown in the *command prompt*.

Think of prefixes as temporarily changing drives and/or user areas while Pro-DOS searches for and loads the required program.

Pro-DOS restores the original drive and user area before executing the program. To permanently change drives and user areas you must issue the proper commands. Changing drives was covered in section [2.3.1](#), and information about user areas can be found in the "Internal Commands" section of this manual.

Often programs require additional parameters. For example, a file copier may need a destination drive as well as a source file to copy.

Parameters are given by separating the first one from the filename by one or more spaces, and then separating each parameter from the previous by the same method.

If *RUN_ME.COM* was actually a file copier, and you wanted to copy a file called *HELLO.DAT* from the disk in drive **B**: to the disk in drive **C**: you might issue a command similar to:

```
A0>run_me b:hello.dat c:
```

In this example the current user area would apply to both drives.

2.3.4 Wildcard Filenames

Pro-DOS allows *wildcard* filenames in the parameters of its internal commands and external utilities. These wildcards are ***** and **?**

The **?** character matches any single character, for example:

?E?LO.COM

Would match both of these filenames:

HELLO.COM MELLO.COM

But wouldn't match these:

HELLO.TXT MELLOW.COM

The ***** character matches all remaining characters to the right of its current position in any part of the filename, for example:

***.COM**

Would match both of these filenames:

HELLO.COM SOMEFILE.COM

But wouldn't match these:

HELLO.DOC SOMEFILE.ASM

You can mix the ***** and **?** characters, for example:

SO?E*.B*

Would match both of these filenames:

SOMEFILE.BAT SONE.BOM

But wouldn't match these:

SOMEFILE.DAT SONIC.BOM

Finally, you can use ***.*** to match any filename or all files.

2.3.5 Printer Echoing

Sometimes you might want all of the text output to the screen also echoed to your printer¹.

Pro-DOS gives you the ability to switch printer echoing on and off by pressing **CNTRL + P**. When printer echoing is switched on you will see an uppercase "P" in the top right of the screen.

Pressing **CNTRL + P** again would switch off printer echoing.

Programs designed to produce printer output (word processors for example) **will not** require you to switch on printer echoing. They will call appropriate system functions to send output to the printer.

There is also an internal command (*FEEDS*), which allows you to switch off and on any line feed characters sent to your printer. The system defaults to sending line feeds.

2.3.6 Scroll Pausing

When displaying a large amount of text information, like sending a document to the screen for example, the system will present it as one continuously scrolling block. Sometimes it can be useful to pause after each screen to give yourself a chance to read it.

Pro-DOS supports scroll pausing, and you can turn this on and off by pressing **CNTRL + S**. When scroll pausing is switched on you will see an uppercase "S" in the top right of the screen.

Pressing **CNTRL + S** again would turn off scroll pausing.

When the system pauses scrolling you will see the "S" in the top right-hand corner start to flash. To scroll on one more screen simply press any key. You can also press **CNTRL + S** to continue scrolling whilst simultaneously switching off scroll pausing.

2.3.7 Aborting

Pressing **CNTRL + C** will usually abort any process in Pro-DOS, and this often applies to external programs running under Pro-DOS too.

¹ Standard parallel printers plugged into port 1 of the MGT Comms Interface

3 - Internal Commands

Pro-DOS provides 15 internal commands. As internal commands are built-in, rather than loaded from disk, they are always available.

3.1 CLS

This command takes no additional parameters. It allows you to clear the screen if things are starting to look a little untidy.

3.2 COPY

Unlike v1.9's external copy program, which had to be loaded from disk, v2.0's copy is built-in and provides comprehensive copying functions. It allows you to copy between drives, between drives and user areas, or just between user areas. Renaming files as they are copied is supported, as is wildcard copying and renaming.

Below are some examples of various combinations. Although the prompt (**A0>**) is shown here you wouldn't actually type it in:

```
1 .... A0>copy *.* b:
2 .... A0>copy b10:*.com
3 .... A0>copy c:*.asm b4:*.z80
4 .... D0>copy 5:old.com 7:new.com
5 .... E2>copy d3:*.*
```

1. Copies all files from the current drive and user area (**A0**) to the same user area on drive **B:**, keeping the same names.

2. Copies all files with a **.COM** extension from drive **B:** user area **10** to the current drive and user area (**A0**), keeping the same names.

3. Copies all files with a **.ASM** extension from drive **C:** user area **0** to drive **B:** user area **4**, renaming the **.ASM** extension to **.Z80** on each copy.

4. Copies the file **OLD.COM** from user area **5** of the current drive (**D**) to user area **7** of the same drive, renaming the copy **NEW.COM**

5. Copies all files from drive **D:** user area **3** to the current drive and user area (**E2**), keeping the same names.

3.3 DIR

Is the disk **DIR**ectory command, and a command you will use a lot.

When used without parameters it gives you a list (or directory) of all the *non-hidden* files in the current user area on the disk in the current drive. Which might look something like:

```
A0>dir

A0:ATTRIB .COM | CRC      .COM | FORMAT  .COM | SAMREAD .COM | SYSTEM  .BIN
A0:BATCH  .COM | FATREAD .COM | README  .TXT
8 Files, 322k free

A0>_
```

The directory is always shown in alphabetically sorted columns.

Also shown is the total number of files and how much free space there is left on the disk. A *DIR* command issued for a flash card will also show the number of the currently selected *logical disk*, as in the following example:

```
B0>dir d:

D0:ALLOC .COM | CHEK15 .LBR | EWADE .COM | HIDE .COM | SARGON .COM
D0:ANIMAL .BBC | CIT .LBR | F-INDEX .BBC | LT31 .COM | SD .COM
D0:ANIMAL .DAT | CKCRC .LBR | F-RAND0 .BBC | MBASIC .COM | SD .DOC
D0:AUTOMENU.COM | CONTENTS.DOC | F-RAND1 .BBC | MERGE .BBC | SPZ37 .COM
D0:B29 .COM | CONVERT .COM | F-RAND2 .BBC | MLOAD .COM | STAT .COM
D0:BATCH .ZSM | COPY .COM | F-RSER1 .BBC | NCRC .COM | SURVEY .COM
D0:BBCBASIC.COM | COPY .PRN | F-RSER2 .BBC | OCOPY .COM | TABS .COM
D0:BBCBASIC.TXT | CRC .COM | F-RSTD .BBC | PALETTE .COM | TABS .ZSM
D0:BBCDIST .MAC | CRC .OLD | F-WESER1.BBC | PALETTE .ZSM | TEST .SUB
D0:BUFFTEST.COM | CRCKLIST.CRC | F-WESER2.BBC | PROTECT .COM | WADE .DOC
D0:BUFFTEST.PRN | DSTAT .COM | F-WSER1 .BBC | PROTECT .ZSM | WADE .LBR
D0:BUFFTEST.ZSM | DUMP .COM | F-WSER2 .BBC | PTEST .ZSM | ZAP .COM
D0:CERTIFY .COM | EDDY .COM | F-WSTD .BBC | READ .ME | ZDE .COM
D0:CHARS .COM | EOL .COM | FORMAT .COM | SAMREAD .COM | ZSID .COM
D0:CHARS .PRN | EOL .ZSM | HELP .COM | SAMREAD .ZSM | ZSM .COM
D0:CHARS .ZSM
76 Files, 436k free on logical disk 78

B0>_
```

Drive/user prefixes ([2.3.3](#)) and filename wildcards ([2.3.4](#)) can be used to list files in any user area on any drive or logical disk.

The following examples show some more *DIR* action:

List all **.COM** files on drive **E:**, user area **12** from the currently selected drive and user area (**B0**):

```
B0>dir e12:*.com

E12:ACOPY   .COM | CONVERT .COM | FORMAT   .COM | PALETTE .COM | SUBMIT   .COM
E12:ALLOC   .COM | COPY    .COM | HELP     .COM | PIP     .COM | SURVEY   .COM
E12:AUTOMENU.COM | CRC     .COM | HIDE     .COM | PROTECT .COM | TABS     .COM
E12:B29     .COM | DSTAT   .COM | LT31     .COM | SAMREAD .COM | ZAP      .COM
E12:BBCBASIC.COM | DUMP    .COM | MBASIC   .COM | SARGON  .COM | ZDE      .COM
E12:BUFFTEST.COM | EDDY    .COM | MLOAD    .COM | SD      .COM | ZSID     .COM
E12:CERTIFY .COM | EOL     .COM | NCRC     .COM | SPZ37   .COM | ZSM      .COM
E12:CHARS   .COM | EWADE   .COM | OCOPY    .COM | STAT    .COM
39 Files, 310k free

B0>_
```

List all **.ZSM** files on drive **E:**, user area **12**, when that drive and user area are the currently selected drive and user area (**E12**):

```
E12>dir *.zsm

E12:BATCH   .ZSM | CHARS   .ZSM | PALETTE .ZSM | PTEST   .ZSM | TABS     .ZSM
E12:BUFFTEST.ZSM | EOL     .ZSM | PROTECT .ZSM | SAMREAD .ZSM
9 Files, 310k free

E12>_
```

List all files beginning with a letter "**F**" in user area **0** of the selected *logical disk* (**78**) on the flash card in drive **D:** from the currently selected drive and user area (**E12**):

```
E12>dir d0:f*.*

D0:F-INDEX .BBC | F-RAND2 .BBC | F-RSTD   .BBC | F-WSER1 .BBC | F-WSTD   .BBC
D0:F-RAND0 .BBC | F-RSER1 .BBC | F-WSER1.BBC | F-WSER2 .BBC | FORMAT   .COM
D0:F-RAND1 .BBC | F-RSER2 .BBC | F-WSER2.BBC
13 Files, 436k free on logical disk 78

E12>_
```

List all files beginning with a letters "**BU**" in user area **5** of the currently selected drive (**F**):

```
F0>dir 5:bu*.*

F5:BUFFTEST.COM | BUFFTEST.PRN | BUFFTEST.ZSM
3 Files, 324k free

F0>_
```

3.4 DIRS

Functionally identical to the *DIR* command, except it lists files that have their *system attribute* set. These files are usually hidden.

Pro-DOS provides an external utility (*ATTRIB.COM*) to allow you to set/reset the system attribute on any file or files.

3.5 DISK

Is used to select an individual *logical disk* on a flash card.

When used without parameters *DISK* gives you a summary of all available flash cards in your system.

For example:

```
A0>disk  
  
C: Flash Card : ATOM CF (bdos) : 9 logical disks, no disk selected  
D: Flash Card : TRINITY (pdos) : 122 logical disks, disk 78 selected  
  
A0>_
```

Here drive **C:** is a flash card in the ATOM-Lite, it is in BDOS format, contains **9** logical disks and has no logical disk currently selected.

Drive **D:** is a flash card in the Trinity, it is in native Pro-DOS format, contains **122** logical disks and logical disk **78** is currently selected.

To change logical disks on any flash card in your system you supply its drive letter, a colon and then the logical disk number.

So to select logical disk **5** on the flash card in drive **D:** you would:

A0>disk d:5

Logical disks remain permanently selected until you use the *DISK* command again to change to a different one.

If your system only has a single flash drive you can choose to omit the drive letter and simply supply the required logical disk number.

You can also choose to omit the drive letter if the currently selected drive (shown in the command prompt) contains the flash card you want to change logical disks on.

In these cases the logical disk is applied automatically to either the single available flash drive, or the currently selected flash drive.

3.6 DRIVES

This command takes no additional parameters. It simply presents a list of the available drives in your system.

Sometimes it can be useful to see which drive letters belong to which drives, especially if you have a full eight drive system and you've used *SWAP* to change the order of some of them!

Here's an example:

```
D0>drives

A: RAM Drive : Internal
B: Floppy    : Left Bay
C: Flash Card : ATOM CF (bdos) : 9 logical disks, no disk selected
D: Flash Card : ATOM CF (pdos) : 122 logical disks, disk 78 selected
E: RAM Drive  : External
F: RAM Drive  : External
G: RAM Drive  : External
H: RAM Drive  : External

D0>_
```

3.7 DUMP

Allows the viewing of files at the byte level. *DUMP* does this by producing a HEX and ASCII display of a file's contents to the screen.

Content can also be echoed to your printer using *CNTRL + P*.

DUMP requires one filename parameter.

Drive/user prefixes ([2.3.3](#)) and filename wildcards ([2.3.4](#)) can be used to view files in any user area on any drive or logical disk.

When dumping files using wildcards you will be given the option to view or skip each file as it is found.

The following shows an example output from the *DUMP* command used on a file called *SOMEFILE.COM* from user area 5 on the disk in the current drive (A):

```
A0>dump 5:somefile.com

000000: 0E 27 59 CD 05 00 21 DC AC A7 ED 52 28 06 11 46  .'Y...!....R(..F
000010: 05 C3 0F 05 11 80 00 0E 1A CD 05 00 0E 19 CD 05  .....
000020: 00 32 23 05 0E 20 1E FF CD 05 00 32 2A 05 D9 AA  .2#.. ....2*...
000030: 28 0F F2 3B 01 11 30 05 C3 0F 05 5A 0E 20 CD 05  (..;..0....Z.  ..
000040: 00 11 00 0A 21 00 00 7B 06 08 AC 67 29 30 08 7C  ....!..{...g)0.|
000050: EE 10 67 7D EE 21 6F 10 F3 EB 73 24 72 25 EB 1C  ..g}.!o...s$r%..
000060: 20 E2 21 6C 00 CD C2 04 28 14 11 62 05 4E 2D 7E  .!l....(..b.N~
000070: FE 2B C2 15 05 3E 49 B9 C2 15 05 32 FF 01 21 5C  .+...>I....2..!\

OK

A0>_
```

3.8 ERA

Is used to **ERA**se a file or files from any user area on any drive or logical disk. Drive/user prefixes ([2.3.3](#)) and filename wildcards ([2.3.4](#)) can be used to selectively erase files.

ERA requires one filename parameter, here are some examples:

```
1 .... A0>era *.*
2 .... B0>era somefile.com
3 .... A0>era d7:*.tmp
4 .... E3>era b:temp.*
5 .... F0>era 6:hello*.t?p
```

1. Erase **all** files from the current drive and user area (A0). When requesting erasure of all files the system will ask for confirmation.

2. Erase *SOMEFILE.COM* from the current drive and user area (B0).

3. Erase all files with a *.TMP* extension from drive **D:**, user area 7.

4. Erase all files called *TEMP* with any extension name from the current user area (3) on drive **B:**

5. Erase all files that have the first five characters of their filename as *HELLO* and an extension with the first character as *T*, any second character and the third character as *P* from user area 6 on the currently selected drive (F).

3.9 FEEDS

This command takes no additional parameters.

Each time you enter this command it toggles between telling the system not to send or to send a line feed character to your printer after each carriage return. The system default is to send line feeds.

The current state will be displayed after each use of the command and the example below shows line feeds getting switched off, and then back on again:

```
A4>feeds
Are off
A4>feeds
Are on
A4>_
```

3.10 PALETTE

Allows you to change any (or all) of the component parts of the system's on-screen colours.

PALETTE requires one to five additional parameters, which are:

- i=#** (Ink colour)
- p=#** (Paper colour)
- b=#** (Cursor's bright colour)
- d=#** (Cursor's dark colour)
- u=#** (Colour of the character under the cursor)

PALETTE always requires at least one parameter to work with and you can give the parameters in any order.

Separate multiple parameters using a space. Any duplicates will be ignored, the system will use the last occurrence of any duplicates.

The **#** represents any numeric value between 0 and 127 and the value corresponds directly to the colours shown in chapter five, page 66 of the *MGT SAM Coupé User's Guide*.

The *ink* and *paper* components need no explaining.

The remaining components control the flashing cursor colours, with **U** controlling the ink colour of any character underneath the cursor.

The underneath ink colour is more prominent when the cursor is a solid block (*caps-lock* mode) rather than the usual underline.

Here are some examples:

```
1 .... A0>palette i=127
2 .... A0>palette p=2 i=127
3 .... A0>palette b=127 d=7 u=42
```

1. Changes the *ink* to bright white.
2. Changes the *ink* to bright white, and the *paper* to dark red.
3. Changes the *bright cursor* to white, the *dark cursor* to grey, and the *ink* of the character underneath the cursor to bright red.

To restore Pro-DOS v2.0's default (retro green) colours use:

```
A0>palette i=72 p=0 b=76 d=4 u=0
```

If you preferred the white on blue scheme of Pro-DOS v1.9 use:

```
A0>palette i=127 p=27 b=127 d=127 u=27
```

To simulate the look of a retro amber monitor you can use:

```
A0>palette i=98 p=0 b=106 d=0 u=98
```

Providing the parameters are valid, the system changes the colours immediately and asks you to confirm that they are OK.

Answering *yes* (press Y) returns to the command prompt with the new colours in place. Answering *no* (press N) restores the previous colour scheme before returning you to the command prompt.

If you can't see the on-screen text because of a bad choice of ink/paper colours simply press N (or wait 10 seconds) and the system will restore your previous colours.

If you have a Quazar Trinity Ethernet Interface attached, Pro-DOS will save your colours to its EEPROM. The next time Pro-DOS boots, and providing the Trinity is still attached, your chosen palette colours will be restored.

3.11 REN

Is used to **REName** a file or files from any user area on any drive or logical disk. Drive/user prefixes ([2.3.3](#)) and filename wildcards ([2.3.4](#)) can be used to selectively rename files.

REN requires two filename parameters separated by one or more spaces. Here are some examples:

```
1 .... A0>ren oldfile.com newfile.com
2 .... B5>ren a0:*.asm *.z80
3 .... C0>ren 5:b*.* d*.*
4 .... A0>ren d:test.com newname
```

1. The rename command at its simplest. This example changes the name of the file *OLDFILE.COM* to *NEWFILE.COM*.

2. Renames all files with a *.ASM* extension in user area **0** on drive **A:**, keeping their original name but changing the extension to *.Z80*. *A.ASM* would become *A.Z80*, *TEST.ASM* becomes *TEST.Z80* and so on. This is an example of a wildcard rename.

3. Renames all files that start with the letter "**B**" in user area **5** on the current drive (**C**) changing the first letter to "**D**". *BOB.COM* becomes *DOB.COM*, *BADDY.DAT* becomes *DADDY.DAT* and so on.

4. Renames the file *TEST.COM* in the current user area (**0**) on drive **D:** to *NEWNAME*

The second filename parameter doesn't require a drive and/or user prefix. The *REN* command always uses any drive and/or user prefix given in the first filename as its target.

If you do specify a drive and/or user prefix in the second parameter the system automatically removes it before proceeding.

Wildcard renaming a large number of files can be a slow process, so the *REN* command displays a small rotating "*time wheel*" in the bottom right of the screen to indicate it is still working.

This can be useful on flash drives because they are silent, and it is not always clear that they are being read from or written to.

3.12 SAVE

Allows you to save the contents of the system's *TPA*¹.

SAVE requires two parameters separated by one or more spaces.

The first parameter is a (decimal) number between **0** and **255** giving the total number of 256-byte blocks to save. The second parameter is the filename.

Drive/user prefixes ([2.3.3](#)) are allowed in the filename, but the filename cannot contain wildcards, it must be unambiguous.

Here are some examples:

```
1 .... A0>save 4 one-k.bin
2 .... B0>save 255 d7:bigfile.bin
3 .... D5>save 8 0:two-k.bin
4 .... E0>save 0 empty.dat
```

1. Saves a 1024-byte (4x256) file called *ONE-K.BIN* to the current drive and user area (**A0**).

2. Saves a 65280-byte (255x256) file called *BIGFILE.BIN* to user area **7** on drive **D:**. This is the largest file you can save using *SAVE*.

3. Saves a 2048-byte (8x256) file called *TWO-K.BIN* to user area **0** of the current drive (**D**).

4. Saves a zero-length file called *EMPTY.DAT* to the current drive and user area (**E0**).

You can use zero-length files as a software disk label. Although zero-length files take no space they do use up one directory entry.

SAVE is not particularly useful and think of it as a CP/M 2.2 legacy command. It has been included in case you wanted to use some of the older CP/M development tools, as these often relied on *SAVE*.

Like the original CP/M 2.2 *SAVE* command, the Pro-DOS version will automatically overwrite any existing file without warning or error.

¹ System TPA runs from address &100 (256 decimal)

3.13 SWAP

Allows you to "software-swap" any pair of drives in the system.

SWAP requires two drive letter parameters separated by a colon.

Pro-DOS v2.0 sets its internal RAM Drive (pre-installed with system utilities) as drive **A:** and the left-hand floppy drive as drive **B:**

This is different to v1.9, which always set drive **A:** as the left-hand floppy and its internal RAM Drive was either **B:**, in the case of a single-drive SAM or **C:**, on a SAM with twin drives. Some users may prefer this layout and *SWAP* allows you to achieve this.

The example below shows the use of *SWAP* (and *DRIVES* to display the information) to change v2.0's twin-floppy layout to equal v1.9's:

```
A0>drives
A: RAM Drive   : Internal
B: Floppy      : Left Bay
C: Floppy      : Right Bay

A0>swap a:c
OK

C0>swap a:b
OK

C0>drives
A: Floppy      : Left Bay
B: Floppy      : Right Bay
C: RAM Drive   : Internal

C0>_
```

A drive remains swapped until you swap it again or reload Pro-DOS, and swapped drives apply to programs as well as the command line.

So swapping the internal RAM Drive to drive **C:** would mean all running programs will "see" drive **C:** as the internal RAM Drive.

If you *SWAP* the current drive (shown in the command prompt) the system follows to wherever this drive ends up.

You can see this in action in the example above as the command prompt changes from **A0>** to **C0>** after the first *SWAP*.

3.14 TYPE

Allows you to view text (ASCII) files on your screen with the option to echo them to your printer using *CNTRL + P*.

TYPE requires one filename parameter.

Drive/user prefixes ([2.3.3](#)) and filename wildcards ([2.3.4](#)) can be used to view files in any user area on any drive or logical disk.

When viewing files using wildcards you will be given the option to view or skip each file as it is found.

The internal RAM Drive holds a number of external system utility programs. As well as the programs it also holds a small information file called *README.TXT*, which can be viewed using *TYPE*.

The example below shows *TYPE* being used to view this information file from its default drive and user area (**A0**):

```
A0>type readme.txt

+-----+
| This RAM Drive contains all of the external system utilities described in |
| chapter four of the user manual. It is safe to erase any (or all) of the |
| files on this drive if you are not planning to use them during a session. |
+-----+

OK

A0>_
```

TYPE handles text files with MS-DOS line endings, usually a carriage return followed by a line feed, and it will also handle text files with UNIX style line endings, usually just a line feed.

The ability to handle both types can be useful when viewing text files created on some of the earlier CP/M systems.

Perhaps more useful is the ability to use *TYPE* to view text files from the archives of some early *Bulletin Boards Systems*, as these were often saved with UNIX style line endings.

When viewing large documents it can be useful to pause after each screen. You can achieve this by pressing *CNTRL + S* (and observing an uppercase "S" in the top right of the screen) as described in section [2.3.6](#).

3.15 USER

Is the final internal command and its function is to allow you to select any of the 16 available user areas on a disk.

USER takes one parameter, a decimal value between **0** and **15**.

User areas are used to separate files that reside on a single disk. For example, you could have all your text editors in user area **0**, all your assemblers in user area **1** and so on. User areas are a legacy of CP/M 2.2 and are not nearly as useful as they sound!

Disk operations are performed on the current drive and from the current user area, as shown in the command prompt. Which means programs cannot usually "see" files outside the current user area.

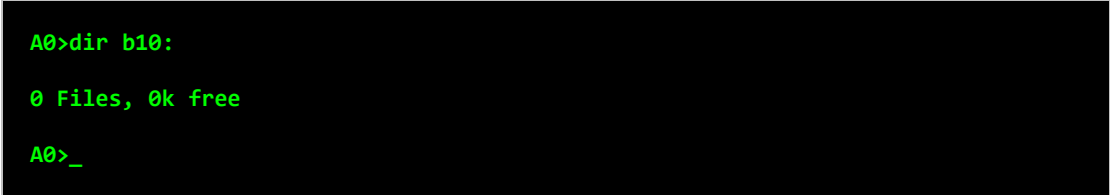
This explains why user areas are not that useful!

However, in Pro-DOS you can temporarily override the current user area using a drive/user prefix as described in section [2.3.3](#). This allows you to use commands such as *DIR*, run a program from or copy files from/to other user areas without having to permanently change to them first.

Pro-DOS offers a preferred shortcut method to change user areas as described in section [2.3.1](#) and, similar to the *SAVE* command, *USER* is a legacy command included for CP/M 2.2 compatibility reasons.

User areas will affect *DIR* and *DIRS* too. Those commands always show the total disk space remaining on a disk after subtracting the space used by *all* files in *all* user areas.

So although *DIR* or *DIRS* might not show you any files, they might also say there is less than 706k available on the disk! Like this:



```
A0>dir b10:  
  
0 Files, 0k free  
  
A0>_
```

In this example there are no files and also no available disk space!

It is actually saying there are no visible files *in user area 10* and the capacity has been used by files elsewhere on the disk.

4 - External Utilities

Pro-DOS v1.9 came on two disks, the second of which was labelled *System Files* and contained programs used to perform occasional tasks such as disk formatting, batch processing and so on.

Pro-DOS v2.0 doesn't require a second disk. Instead, the system files (or utilities) used to perform occasional tasks are pre-installed to the internal RAM Drive at boot time.

You may not plan to use any of the system utilities during a session, in which case you can erase any (or all) of them to free up space on the internal RAM Drive. The system utilities are always restored every time you boot Pro-DOS.

v2.0 provides 6 external utilities, all are standard *.COM* programs.

As a little reminder, you execute (or run) *.COM* programs by typing their filename, include a drive/user prefix if they are on a different drive, followed by any required parameters separated from the filename by one or more spaces. Then press *RETURN*.

As the utilities are more complex than the internal commands they all give basic usage information¹ when run without parameters.

Here is the information *ATTRIB.COM* provides:

```
A0>attrib

Usage:  ATTRIB "filespec" +/-r +/-h

SET 'read only' = ATTRIB "filespec" +r
RESET 'hidden' = ATTRIB "filespec" -h

You can combine flags, for example: ATTRIB "filespec" -h +r

To display current file attributes: ATTRIB "filespec"

filespec = FILENAME or X:FILENAME or X0:FILENAME or 0:FILENAME
           X is an optional drive letter, and 0 is an optional
           user number between 0 and 15

Wildcard filenames accepted

A0>_
```

The rest of this manual describes the external utility programs, their function and parameter requirements.

¹ The *CRC.COM* utility is an exception to this (see its entry in section [4.3](#))

4.1 ATTRIB.COM

Is used to *set* or *reset* the file attributes of a file or group of files. It can also show the attributes of a file or group of files.

ATTRIB takes a filename parameter followed by an optional number of *set/reset* switches. You can use drive/user prefixes ([2.3.3](#)) and filename wildcards ([2.3.4](#)) to selectively process files in any user area on any drive or logical disk.

The file attributes are:

SYSTEM (*HIDDEN*)

Files with the *system* attribute set are hidden from the standard *DIR* command. Pro-DOS provides the *DIRS* command to view them. Hidden files can be used like visible files and the *system* attribute serves only to hide them from *DIR*.

READ-ONLY

Files with the *read-only* attribute set are protected from accidental erasure. Protection is at software level and does not protect against disk formatting or programs that reset the read-only attribute.

The only protection against formatting is physical write protection.

Files can have any of these attributes set or reset. It is perfectly valid to have files that are both hidden and read-only.

Running *ATTRIB* with just a filename parameter will show you the current attributes of a particular file or files.

Here's an example:

```
A4>attrib *.*  
  
FILE.1  
RH FILE.2  
FILE.3  
R FILE.4  
  
A4>_
```

Here you can see *FILE.2* is both *read-only* and *hidden*, and *FILE.4* is just *read-only*. The letters **R** (read-only) and **H** (hidden) denote the attribute is *SET*, no letter denotes the attribute is *RESET*.

To alter the attributes of a file or group of files you provide one or more switch parameters after the filename. The switches are:

- +r** To *set* the read-only attribute
- r** To *reset* the read-only attribute
- +h** To *set* the system (hidden) attribute
- h** To *reset* the system (hidden) attribute

Here is an example of setting the *read-only* attributes on all files:

```
A4>0:attrib *.* +r

R ONE
R TWO
R THREE
R FOUR

A4>_
```

You can see an **R** indicator by each file to indicate that it has had its attribute set. In this example a user area prefix has been used to force *ATTRIB.COM* to load from user area **0**.

Using the modified files from the example above, the next example shows resetting the *read-only* attribute on all files beginning with the letter "T" whilst simultaneously setting their *system* attribute and, finally, displaying the attributes of all of the files:

```
A4>0:attrib t*.* -r +h

H TWO
H THREE

A4>0:attrib *.*

R ONE
H TWO
H THREE
R FOUR

A4>_
```

You can include as many switches as you like, the system always uses the last one it finds of any duplicates.

Although one or more spaces are required between the filename and the switches, the switches do not require spaces between them. So a switch pair given as **+h-r** would be perfectly acceptable.

4.2 BATCH.COM

Allows you to execute a group of commands as if they were entered one-by-one on the command line. It does this by reading the lines from a file known as a *batch* file.

Batch files are text files saved with a *.BAT* extension. Alternatively, the extension *.SUB* can be used if you wish, as the original CP/M *SUBMIT.COM* program used *.SUB* and Pro-DOS is compatible with output from that program.

BATCH takes a filename parameter followed by an optional number of *string replacements*, each separated by one or more spaces. A drive/user prefix ([2.3.3](#)) can be used on the batch file's filename but it cannot contain wildcards, it must be unambiguous.

Specifying *.BAT* (or *.SUB*) is not required as *BATCH* automatically searches for both, with *.BAT* files taking precedence.

Batch files are useful in situations where you find yourself typing a lot of commands. Assembler sessions are a good example of this and assembling and linking a single file might look like:

```
A0>asem source.z80 myprog.hex
    Assembly complete: 0000 Errors.
A0>link myprog.hex
    Linked File OK. Saved size 512 bytes.
A0>copy myprog.com b:
A0>_
```

Using a *.BAT* file could have reduced the process to:

A0>batch assemble source myprog b:

Which would load and run *BATCH.COM* and, once loaded, it would process the lines found in *ASSEMBLE.BAT* (or *ASSEMBLE.SUB*).

source, **myprog** and **b:** are *string replacements* passed to the batch file, with each replacement having a corresponding number.

The first parameter after the batch-file name is **\$1**, in this example the word **source**, **\$2** equals **myprog** and **\$3** would be **b:**

The contents of *ASSEMBLE.BAT* file might look something like:

```
assem $1.z80 $2.hex
link $2.hex
copy $2.com $3
```

In this example *BATCH* would replace **\$1** with the string "**source**", all **\$2**'s with "**myprog**" and **\$3** with "**b:**".

You can use as many string replacements as you like, remembering that each replacement has a corresponding **\$n** number matching its position. String replacement **\$1** is always the first parameter after the batch-file name given to *BATCH*.

If you need to use the **\$** symbol in one of the lines in a batch file just type it twice. *BATCH* replaces **\$\$** with a single **\$**.

BATCH terminates the processing of a batch file if any system errors occur, like trying to write to a write-protected disk for example. You can manually terminate the processing of a batch file by holding down any key until control is returned.

CP/M does not allow *nesting* of batch files. However, you can *chain* them by loading *BATCH.COM* at the end of a batch file and giving it another *.BAT* file to process. When chaining you need to remember that the string replacement numbers change for subsequent batch files. Take the following example:

```
dir $1
copy $2 $3
batch $4 $5 $6
```

If this file was saved as *CHAIN.BAT* you might invoke it by:

```
A0>batch chain b: file1 file2 newbatch new1 new2
```

Inside *CHAIN.BAT* **\$1** is replaced by "**b:**", **\$2** by "**file1**" and so on.

When *BATCH* is run again from inside *CHAIN.BAT* it is given the file **newbatch**, which was **\$4**'s replacement. Inside *NEWBATCH.BAT* a **\$1**, and not the **\$5** it would have been inside *CHAIN.BAT*, gets replaced by "**new1**". **\$n** variables reset when *BATCH.COM* loads.

Running v1.9's BATCH under v2.0 is not recommended as v1.9's BATCH writes to memory that now contains code!

4.3 CRC.COM

Allows you to checksum a file or group of files. *CRC* can also produce *integrity files* that contain lists of files to check and their stored checksums. It can also check files from within integrity files.

CRC takes a filename parameter followed by an optional switch to request production of an integrity file. You can use drive/user prefixes ([2.3.3](#)) and filename wildcards ([2.3.4](#)) to selectively process files in any user area on any drive or logical disk.

Running *CRC* without a filename parameter (or just a drive/user) forces it to search for an integrity file called *CHECKSUM.CRC*. It expects to find this in the current drive/user area shown in the command prompt, or a drive/user area you specified. When *CHECKSUM.CRC* cannot be found a usage screen will be displayed.

Here are two examples:

A0>crc e5:

Expects to find *CHECKSUM.CRC* in user area **5** on drive **E**:

A0>crc

Expects to find *CHECKSUM.CRC* in the current drive/user area (**A0**).

When an integrity file is found *CRC* proceeds to check the files listed within. It expects to find these files in the same place it found the *CHECKSUM.CRC* file, which will be the default drive/user area, or the drive user area you specified.

The output from *CRC* will look something like:

```
A0>crc e5:

Checking the files in CHECKSUM.CRC:

MBASIC .COM = AC87 - OK
HELP .COM = 55E1 - OK
XSUB .COM = E9D4 - OK

RESULTS
-----
Correct: 3
Corrupt: 0
Missing: 0

A0>_
```

To produce an integrity file you must specify which file or files you want it to contain, followed by the integrity file switch **+i**

Here are some examples:

```
A0>crc *.* +i
```

Would checksum all files on the current drive and user area (**A0**) and produce a *CHECKSUM.CRC* integrity file in the same place.

```
A0>crc e6:*.* +i
```

Would checksum all files in user area **6** on drive **E:** and produce a *CHECKSUM.CRC* integrity file in user area **6** on drive **E:**

```
D4>a0:crc 3:*.com +i
```

Would checksum all *.COM* files in user area **3** on the current drive (**D**) and produce a *CHECKSUM.CRC* integrity file in user area **3** on drive **D:**. In this example a drive/user prefix has been used to force *CRC.COM* to load from drive **A:**, user area **0**.

It is a good idea to check there is enough space available for the *CHECKSUM.CRC* file on the disk where it will be created, otherwise *CRC* will abort with a disk/directory full error. The maximum space it will occupy (in bytes) is always $(number-of-files*13)+6$.

If you want to checksum a file or a group of files but do not want to produce a *CHECKSUM.CRC* integrity file then omit the **+i** switch.

For example:

```
A0>crc *.*
```

Would checksum all files on the current drive and user area (**A0**) and only display the results on screen.

```
A0>crc 5:*.com
```

Would checksum all *.COM* files in user area **5** on the current drive (**A**) and only display the results on screen.

The *CHECKSUM.CRC* file contains its own sum that *CRC* ensures is correct before checking the files listed within.

4.4 **FATREAD.COM**

Allows the copying of files from *FAT12/16/32*¹ formatted storage media. It also allows traversal and copying of subdirectories.

FATREAD takes a single drive/colon parameter, like this:

A0>fatread c:

This first thing *FATREAD* does is check the media in the drive is actually FAT formatted, and this could be a 720k floppy disk or a CF/MMC/SD/SDHC flash card.

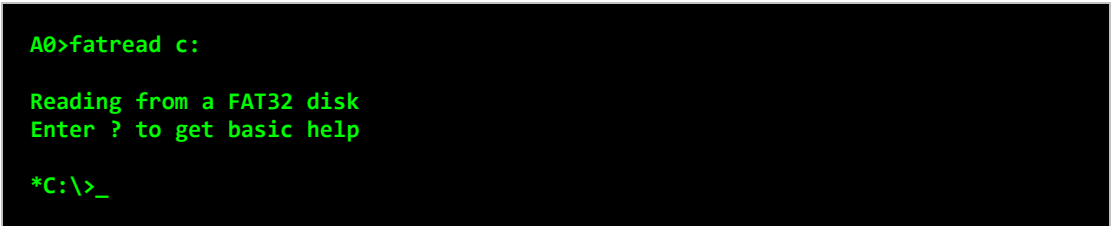
If the media is FAT formatted it will become the *source disk* for the entire *FATREAD* session. If you wanted a different flash card as the source disk you must first exit *FATREAD* and then reload it with the new card's drive letter as the parameter.

With floppy disks you can simply eject the disk and replace it with another FAT formatted disk, there is no need to reload *FATREAD*.

FATREAD handles all FAT12/16/32 format disks providing they are not partitioned. Floppy disks must be FAT12, 720k DS/DD².

Once it is established that the source disk is suitable you will see some information followed by a command prompt.

It looks something like:



```
A0>fatread c:
Reading from a FAT32 disk
Enter ? to get basic help
*C:\>_
```

The first thing to notice is the command prompt as this is different to the usual CP/M/Pro-DOS prompt. Those familiar with MS-DOS should feel at home as it behaves similar.

The prompt always shows the FAT source disk and the * character is used as a reminder that you are inside *FATREAD*'s environment.

¹ A disk format used by MS-DOS/Windows. Compatible with many other OSes

² Double-Sided, Double-Density - the same as a SAM floppy disk

FATREAD provides a command-line environment through which to process files on FAT formatted storage media. It does this using a set of five commands, which are:

DIR
CD
COPY
EXIT
CLS

EXIT and **CLS** do exactly what you expect, and you can also exit by pressing *CNTRL* + *C* when the cursor is at the start of the line.

The remaining commands are described over the next few pages. A basic understanding of an MS-DOS command line, including paths and subdirectories, will prove useful when using *FATREAD*.

The examples below all used a FAT32 formatted CF card in drive **C:** which contained the entire (500MB) "*Walnut Creek*" CP/M archive.

DIR

Is used to display the contents of the current base directory or a directory specified as a path. You can also include filenames and filenames can include wildcards as described in section ([2.3.4](#)).

Note, although filenames can include wildcards pathnames cannot.

Here is an example showing the *root directory*¹ of the source disk:

```
*C:\>dir

DIR-> $RECYCLE.BIN | DIR-> KAYPRO | DIR-> KILDALL
DIR-> LAMBDA | DIR-> MBUG | DIR-> OSBORNE
DIR-> SIMTEL | DIR-> STARLET | DIR-> UTILS
DIR-> ZSYS | 00-INDEX.TXT | ALLFILES.TXT
CDROM .CFG | DIRS .TXT | GO .BAT
LOOKUP .EXE | README .TXT | TREEINFO.NCD
VIEW .EXE | DIR-> _BBS | DIR-> BEEHIVE
DIR-> COMMODOR | DIR-> CPM | DIR-> CPMHELP
DIR-> CPMINFO | DIR-> DEMON | DIR-> DOCS
DIR-> EMULATOR | DIR-> ENTERPRS | DIR-> FOG
DIR-> JSAGE

9 File(s), 22 Dir(s)

*C:\>_
```

¹ The very first directory on any FAT formatted storage media

Although this looks similar to a CP/M directory you will notice some filenames are marked with **DIR->**. This signifies a subdirectory.

Subdirectories can contain files and potentially more subdirectories. A chain of subdirectories is known as *the path*.

FATREAD does not support FAT (Windows) long filenames, and always displays/processes the short version. As all long filenames have an equivalent short version this does not present a problem.

The following example shows the contents of a subdirectory called **UTILS**, which is found off the root directory:

```
*C:\>dir utils

DIR-> .           | DIR-> ..          |      00-INDEX.TXT
      00README.TXT |      DESCRIPT.ION |      FILES .BBS
      LOOKUP .DAT  |      WILDCAT .TXT | DIR-> 22DISK
DIR-> CPM          | DIR-> CPM86       | DIR-> DOS
DIR-> SQUEEZE

      6 File(s), 7 Dir(s)

*C:\>_
```

Here you can see two special directory entries "." and "..". These are found in all directories off the root and provide a quick way of accessing the current directory "." or the previous directory "..".

The final example of the **DIR** command shows all files with a *.TXT* extension found in a subdirectory called **CPM**, which in turn is contained in a subdirectory called **UTILS** found off the root:

```
*C:\>dir utils\cpm\*.txt

      00-INDEX.TXT |      WILDCAT .TXT

      2 File(s), 0 Dir(s)

*C:\>_
```

Each element of the path is separated using a "\", the same way paths and filenames are separated in MS-DOS. The parts that make up the path cannot contain wildcards, but the filename part can.

Also, no source disk must be specified in either the pathname or the filename. The source disk is pre-selected when you first load *FATREAD* and is always the disk shown in the command prompt.

To prevent you having to type in a full path each time, it can be useful to move to a particular subdirectory and fix this as the base for subsequent operations. This can be achieved using:

CD

CD stands for **C**hange **D**irectory and you follow the **CD** with a path to a directory you wish to set as the base. The path cannot contain wildcards and consists of directory names only.

Here's an example:

```
*C:\>cd utils\squeeze

*C:\UTILS\SQUEEZE>dir

DIR-> .          | DIR-> ..          |          00-INDEX.TXT
      00SQ-USQ.DOC |          ALUSQ2   .COM |          DESCRIPT.ION
      FILES   .BBS |          MAKESQ   |          MAKEUSQ
      SQ      .1   |          SQ       .C  |          SQ       .EXE
      SQ      .H   |          SQCOM    .H  |          SQDEBUG .C
      SQIO    .C   |          TR1      .C  |          TR2     .C
      USQ     .C   |          USQ      .EXE |          USQ     .H
      USQ     .OBJ |          UTR      .C  |          UTR     .OBJ
      WILDCAT .TXT
                                23 File(s), 2 Dir(s)

*C:\UTILS\SQUEEZE>_
```

Here *FATREAD* has set the subdirectory **SQUEEZE** contained in the subdirectory **UTILS** found off the root as the base directory for all subsequent operations.

You can shortcut back to the main root directory at any time by:

**cd **

The base directory path is always shown in the command prompt.

Note the **DIR** command in the above example listed the contents of the base directory. In this example if you wanted to list the root but didn't want to move from the **SQUEEZE** directory you could use:

dir ..\..\.

Or the following, which would do the same:

**dir **

The first example used the special ".." directory to step backwards, each step separated by a "\". The second example used the "\" to force the root directory. Here are two more examples:

```
dir ..\*.txt
```

Would list files with a *.TXT* extension in the directory below the current base directory and:

```
dir \lambda
```

Would list the contents of the directory **LAMBDA** which is expected to be found off the main root.

In both examples *FATREAD* stays in the base directory shown in the prompt, and only *moves* internally for the duration of the command.

The final command is the heart of *FATREAD* and is the one allowing you to copy a file or group of files from a FAT formatted disk to any of the CP/M format disks in your system. The command is:

COPY

All references to pathnames and filenames described in the **DIR** and **CD** commands apply, and **COPY** can take an additional parameter to specify the destination drive or drive/user to copy the files to. If no destination is specified then it defaults to the system drive and user area selected when you ran *FATREAD*.

Wildcard renaming of copies is allowed. Here are some examples:

```
copy *.txt e0:*.doc
```

Copies files with a *.TXT* extension from the base directory (the path in the command prompt) on the source disk to the CP/M disk in drive **E:**, user area **0**. This example would rename the extension of each copy to *.DOC*, whilst preserving the main filename.

```
copy \cpm\utils\asmutl b:
```

Copies all files from the subdirectory **ASMUTL** off the main root via **\CPM\UTILS** to the default user area on the CP/M disk in drive **B:**.

Copy will copy all files contained in a named subdirectory but it does not recurs into other subdirectories inside a named subdirectory.

4.5 *FORMAT.COM*

Allows you to prepare the structure of floppy disks and flash cards for use with Pro-DOS, it also allows you to produce CP/M disks or flash cards that automatically boot Pro-DOS¹.

FORMAT requires slightly different parameters for floppy disks and flash cards, so each storage medium is described separately.

4.5.1 *Floppy Disks*

FORMAT requires a single drive/colon parameter and, optionally, a "make bootable" switch.

To prepare a standard (non-bootable) 720k/706k CP/M floppy disk:

```
A0>format b:
```

Here *FORMAT* expects to find a writable floppy disk in drive **B**:

You will be given a warning to inform you that all data on the disk will be erased, and confirmation is required to continue the format.

v2.0's *FORMAT* writes a slightly different sector interleave² to v1.9 and gives a slight performance gain to floppy disks operations.

FORMAT can also produce CP/M floppy disks that will boot directly to Pro-DOS via the *F9* key. To do this use the following:

```
A0>format b: +b
```

The **+b** switch tells *FORMAT* that you wish to make a standard 720k/706k CP/M disk bootable, and to do this *FORMAT* expects to find a copy of *SYSTEM.BIN* in the current drive/user area. If it can't be found there it will also check in user area **0** of the internal RAM Drive. *SYSTEM.BIN* contains the main OS code and this is required to make bootable disks or flash cards.

Bootable CP/M disks lose 32k of their capacity, making them 674k.

The advantage is they are available immediately to Pro-DOS after boot without having to change from the SAM format boot disk first.

¹ Flash cards are bootable on the ATOM-Lite if you have the AL-BOOT ROM

² Remains fully compatible with v1.9 and v1.9 can use v2.0 format floppy disks

4.5.2 Flash Cards

Flash cards require formatting before Pro-DOS can use them.

There are two exceptions, the first being any flash card already formatted for use with *BDOS*¹. Pro-DOS will recognise these and allow you to use any of the individual BDOS records as a virtual CP/M disk. If an individual record is not already formatted as a CP/M disk then it must be done so before Pro-DOS can use it.

The second is the use of *FATREAD*, which expects flash cards to be FAT12/16/32 formatted and not in native Pro-DOS or BDOS format.

Pro-DOS divides the total capacity of a flash card into a number of smaller *logical disks*. Each logical disk acts like a single disk and everything you can do on a normal floppy/RAM disk you can do on a logical disk. Pro-DOS provides the *DISK* command (described in section [3.5](#)) to select between different logical disks on flash cards.

Pro-DOS is designed to handle two logical disk formats, which it calls **Native Pro-DOS** and **Legacy BDOS**. The information screen shown after booting tells you the format of any flash cards found.

1. Native Pro-DOS:

This is the format preferred by Pro-DOS. It divides a flash card into a number of 1024k (1MB) logical disks, each having 1016k usable capacity and is the same format used by the external RAM Drives. This format also supports a boot-block, allowing it to boot directly to Pro-DOS if used with an ATOM-Lite and the AL-BOOT ROM².

2. Legacy BDOS:

A format devised by Edwin Blink for a patch he wrote for Pro-DOS v1.9 allowing it to use BDOS formatted flash cards. It does this by writing a standard 720k/706k CP/M "*disk*" inside a large container file that resides in a standard BDOS record. This allows you to mix CP/M virtual disks and standard BDOS records on one BDOS card.

The disadvantages are no boot capability and only 706k-storage capacity in each container, the same as a standard floppy disk.

In terms of raw data throughput both formats are identical.

¹ A DOS for the SAM Coupé allowing the use of hard drives and flash cards

² A replacement ROM allowing the SAM to boot from a CF card in the ATOM-Lite

4.5.2.1 *Formatting Native Pro-DOS Cards*

Before Pro-DOS can divide a flash card into a number of native (1MB) logical disks you must do a complete format. This is done in a similar way to floppy disks by entering:

A0>format d:

Here **d:** represents one of your flash card drives.

Similar to floppy disks you can request that a boot-block is written to the card. This allows native format cards to boot Pro-DOS when used in an ATOM-Lite on a SAM containing the AL-BOOT ROM.

The "*make bootable*" switch is the same as for floppy disks:

A0>format d: +b

Formatted native Pro-DOS cards have three more format options available to them. The first allows you to format individual logical disks anywhere on the card, and is achieved by adding a colon/disk number to the drive letter:

A0>format d:100

This would format a single logical disk, in this case *disk 100*, on the flash card in drive **d:**. The disk number should be within the range of the card and *FORMAT* will check that the number is valid.

Formatting an individual logical disk will erase its content, but the content of all other logical disks on the card remain untouched.

If you didn't initially add a boot-block to a native format card, and now you require one, you don't have to reformat the entire card. You can simply add the boot-block using the following:

A0>format d:0

Likewise, to prevent native format cards from booting you can remove the boot-block using the following:

A0>format d:0-

Adding or removing a boot-block can be done as often as you like, it has no affect on the data in any of the logical disks on the card.

4.5.2.2 *Formatting Legacy BDOS Cards*

Edwin Blink (creator of BDOS and the ATOM-Lite) wrote a patch for Pro-DOS v1.9 allowing it to use flash cards with BDOS and the ATOM-Lite. The patch achieved this by using a special disk file and removing v1.9's internal RAM Drive to make room for its code.

The special file Edwin devised is a large container-like file written to a standard BDOS *record*. This file is called "*CP/M DISK*" and is 720k in size. The file contains (hence container) the layout of a standard Pro-DOS 720k/706k floppy disk and a signature Pro-DOS uses to ensure a particular BDOS record contains a valid CP/M "*disk*".

Any remaining space in a BDOS record that contains a CP/M disk file is free for BDOS to save standard SAM files. This allows you to share BDOS and CP/M virtual disks¹ on one BDOS format flash card.

FORMAT allows you to create these special "*CP/M DISK*" files in any record on a BDOS format card. Pro-DOS does not use the short record² sometimes found at the end of flash cards, it requires a full (800k/780k) record. Pro-DOS takes care of this automatically and does not include short records as part of the total record count.

To format a BDOS record for use with Pro-DOS you:

A0>format d:25

Which would create a "*CP/M DISK*" container in **RECORD 25** of a BDOS formatted flash card in drive **d**:

Once formatted you can select this record just like you would any other logical disk in Pro-DOS by using the *DISK* command.

BDOS records formatted for use with Pro-DOS will be renamed to "*Pro-DOS CPM DISK*", and this includes the name in the record list.

Formatting individual BDOS records, or formatting the entire flash card to the native Pro-DOS format, are the only formatting options available for cards already in the BDOS format.

Formatting an individual BDOS record will erase its content, but the content of all other records on the card remain untouched.

¹ Virtual disks are either BDOS "*records*" or Pro-DOS "*disks*" and are synonymous

² A BDOS record with less than the full 780k free space available

4.6 SAMREAD

Allows you to import code files from SAMDOS¹ format floppy disks. It also allows you to import any files from standard 720k, FAT12, MS-DOS format floppy disks.

SAMREAD takes a single drive/colon parameter, like this:

A0>samread b:

Tells *SAMREAD* to look at the disk in drive **b:**, which it expects to be a SAMDOS or 720k, FAT12, MS-DOS format floppy disk.

This utility differs from other Pro-DOS utilities in that it doesn't use a command-line environment. This is a legacy from Pro-DOS v1.9's version of *SAMREAD* and was purposely retained in this rewrite to maintain a sense of familiarity.

SAMREAD works with floppy disks only.

If you are planning to import files from a FAT12 (MS-DOS) floppy disk then the *FATREAD* utility, described in section [4.4](#), is a better choice as it is not as restrictive as *SAMREAD*.

As *SAMREAD* isn't command-line driven you have to use the cursor keys to move around the grid of files and/or MS-DOS subdirectories listed on screen. You then select or deselect files using another set of keys. Full instructions are given inside *SAMREAD*.

Because *SAMREAD* is fairly simple and, hopefully, intuitive enough to use, I won't spend any more time describing it here. Except to mention a few things to remember:

- 1.** It only works with floppy disks.
- 2.** On SAMDOS format disks you can only import code-type files.
- 3.** It sometimes changes SAM filenames to make them CP/M legal.

The original Pro-DOS v1.9 *SAMREAD* utility was designed as a way of getting code files written on the SAM onto your CP/M disks. And that was it! No frills, just basic file importing. This version is not that different, other than having the ability to import from MS-DOS floppy disks too. This addition might prove useful if you own a SAM without any mass flash storage devices.

¹ SAMDOS is used generically to mean SAMDOS, BDOS or MASTERDOS

5 - Appendix A - H19/Z19/VT52 Terminal Codes

Pro-DOS emulates the screen functions of the *Heath H19/H89* and *Zenith Z19/Z89* terminals. Both offer a superset of the *DEC VT52* terminal and are backwardly compatible to it. These were popular terminals from the 1970s and used by many CP/M computers.

The terminal is driven by *escape sequences* consisting of the escape character (27 decimal, &1B hex) followed by one or more additional ASCII characters. Character case is important in the sequences.

CP/M programs often need installing, which is traditionally done by a small install program that modifies parts of the main program for the terminal escape sequences supported by the OS. These install programs often provide a list of popular terminals to select.

First choice should be the *Heath H19/H89*, if that isn't available choose a *Zenith Z19/Z89*. Failing those, choose the good old *VT52*!

When an install program provides the *H19/Z19/VT52* options it can sometimes be better to ignore them and set up the terminal escape sequences manually, providing the program gives you that option.

The reason is that install programs often provide only basic cursor addressing escape sequences, and ignore more powerful sequences like "*clear to end of line*", "*insert line*", "*delete line*", Etc.

This means that installed programs often default to printing space characters to "*clear to end of line*" for example. Internally this is much slower, and performance gains can be achieved if you can manually set up all of the escape sequences a program requires.

The escape sequences used to insert/delete lines are always worth using. Text editors will see a big performance boost here.

Pro-DOS emulates all of the main *H19/Z19/VT52* screen functions, and also includes the *Kaypro*¹ cursor positioning sequence.

Remember, should an install program ask you to provide the escape sequences as ASCII strings then character case is important.

The following page lists the escape sequences supported, their function and also their (hexadecimal) byte-codes:

¹ A popular brand of CP/M computer from the 1980s

Pro-DOS v2.0 H19/Z19/VT52 Escape Sequences

Sequence	HEX Bytes	Function	Notes
ESC A	1B 41	Cursor Up	No effect if on top line
ESC B	1B 42	Cursor Down	No effect if on bottom line
ESC C	1B 43	Cursor Right	No effect if at right edge
ESC D	1B 44	Cursor Left	No effect if at left edge
ESC E	1B 45	CLS and Home Cursor	Clear screen and reset the cursor to the top left
ESC H	1B 48	Home Cursor	Reset cursor to the top left
ESC I	1B 49	Reverse Line Feed	Cursor up. If on the top line the screen scrolls down
ESC J	1B 4A	Clear to End of Screen	Clears from the cursor* to the bottom right of the screen
ESC K	1B 4B	Clear to End of Line	Clears from the cursor* to the end of the line
ESC L	1B 4C	Insert Line	Inserts a line at the cursor position, lines below are scrolled down and cursor X position is reset to zero
ESC M	1B 4D	Delete Line	Deletes a line at the cursor position, lines below are scrolled up and cursor X position is reset to zero
ESC Y row col	1B 59 20 20	Position Cursor	Moves the cursor position to row, col. A &20 offset is always added to row, col. So top-left equals &20,&20
ESC = row col	1B 3D 20 20	Position Cursor	Kaypro version of ESC Y
ESC b	1B 62	Clear to Start of Screen	Duplicate version of ESC d
ESC d	1B 64	Clear to Start of Screen	Clears from the cursor* to the top left of the screen
ESC e	1B 65	Cursor ON	Turns the visible cursor ON
ESC f	1B 66	Cursor OFF	Turns the visible cursor OFF
ESC j	1B 6A	Save Cursor Position	Saves cursor X,Y position
ESC k	1B 6B	Restore Cursor Position	Restores saved X,Y position
ESC l	1B 6C	Clear Row	Clears the current cursor line and resets the cursor X position to zero
ESC o	1B 6F	Clear to Start of Line	Clears from the cursor* to the start of the line
ESC p	1B 70	Set Inverse Video Mode	Use inverse colours
ESC q	1B 71	Set Normal Video Mode	Use normal colours

* All Functions that clear from the cursor position also clear the character under the cursor. None of them affect the position of the cursor on screen

As well as escape sequences, Pro-DOS provides additional control codes that also manipulate the screen and cursor position.

These are all single byte codes with a value < 32 decimal (&20 hex) and several are shortcuts to an equivalent escape sequence.

You often find these shortcuts used in software designed for use on older systems, especially the *Kaypro* brand of CP/M computers.

Table of Pro-DOS v2.0 Terminal Control Codes

Control	HEX Byte	Function	Notes
CTRL-G	07	Bell Sound	Makes a "ding" sound!
CTRL-H	08	Backspace	Cursor left, if cursor is at the left edge then it wraps to the right edge and moves up
CTRL-I	08	TAB	Outputs the required number of spaces to the next TAB stop
CTRL-J	0A	Line Feed	Cursor down. Screen scrolls up if cursor is on the bottom line
CTRL-K	0B	Vertical TAB	Cursor up. No effect when the cursor is already on the top line
CTRL-L	0C	CLS and Home Cursor	Kaypro version of ESC E
CTRL-M	0D	Carriage Return	Resets cursor X position to zero
CTRL-W	17	Clear to End of Screen	Kaypro version of ESC J
CTRL-X	18	Clear to End of Line	Kaypro version of ESC K
CTRL-Z	1A	CLS and Home Cursor	Alternative version of CTRL-L
CTRL-^	1E	Home Cursor	Kaypro version of ESC H

It is worth pointing out that you are free to use any of the shortcuts in place of their equivalent escape sequence. They are all single bytes and process faster than their escape sequence equivalent.

With that in mind, you should always choose the shortcut versions when installing a program. Providing the installer allows of course.

Pro-DOS is an *auto-wrap* terminal and the cursor will automatically wrap to the next line when it steps beyond column 80. If the next line is > row 24 the screen will scroll. Software may have been written for a terminal that doesn't auto-wrap, and may not display its output correctly. If auto-wrap matters to a particular program it usually gives you the option to specify the wrap characteristics of your terminal. WordStar is a classic example here.

6 - Appendix B - Keyboard Return Codes

The codes returned by Pro-DOS v2.0 for the unshifted cursor keys and unshifted delete key are different to those of Pro-DOS v1.9.

This was done to bring the cursor codes in line with some of the more popular CP/M computers. As a result, Pro-DOS v2.0 now returns the traditional *WordStar Diamond*¹ codes for the unshifted cursor keys and a standard backspace for the unshifted delete key.

v2.0 offers three additional sets of codes, including the original codes returned by v1.9. Holding one of the shift keys and simultaneously pressing the required cursor key accesses them.

Changing the codes returned by v2.0 may mean you have to run the installer for some programs again. I have tried to minimise the need by offering popular cursor key return codes via the shift keys. Programs already configured for v1.9 will still work, providing you hold the *shift* key when you need the cursor keys or the delete key.

***UNSHIFTED* = WordStar Diamond (v2.0's default)**

<i>Action</i>	<i>Control Character and HEX Byte Returned</i>
Cursor Left	CTRL-S (13)
Cursor Right	CTRL-D (04)
Cursor Up	CTRL-E (05)
Cursor Down	CTRL-X (18)

***SHIFTED* = Original Pro-DOS v1.9**

<i>Action</i>	<i>Control Character and HEX Byte Returned</i>
Cursor Left	CTRL-\ (1C)
Cursor Right	CTRL-] (1D)
Cursor Up	CTRL-_ (1F)
Cursor Down	CTRL-^ (1E)

¹ CTRL-S/D/E/X codes to equal *Left*, *Right*, *Up* and *Down*

SYMBOL-SHIFTED = Kaypro/ADM-3A

<i>Action</i>	<i>Control Character and HEX Byte Returned</i>
Cursor Left	CTRL-H (08)
Cursor Right	CTRL-L (0C)
Cursor Up	CTRL-K (0B)
Cursor Down	CTRL-J (0A)

CNTRL-SHIFTED = Generic

<i>Action</i>	<i>Control Character and HEX Byte Returned</i>
Cursor Left	CTRL-] (1D)
Cursor Right	CTRL-\ (1C)
Cursor Up	CTRL-^ (1E)
Cursor Down	CTRL-_ (1F)

The delete key in Pro-DOS v2.0 returns the opposite code to v1.9. To return the same code as v1.9 hold down any of the shift keys when pressing delete. The table below shows the return codes:

DELETE KEY

<i>Shift Held</i>	<i>Control Character and HEX Byte Returned</i>
NONE	CTRL-H (08)
SHIFT	DEL (7F)
SYMBOL	DEL (7F)
CNTRL	DEL (7F)

On a final note, the keyboard buffer routines have been improved in v2.0. They no longer buffer huge numbers of cursor up/down codes when scrolling through large documents with a text editor.