MANUAL FOR CUMANA.DOS and Disassembly for RipDOS v2.9

Boot Sequence and the ! Routine on Page #4

There are two 'ROM' areas available in the Cumana disk interface. One is held in EPROM and is used only at bootup. The second is and area of RAM, into which the DOS is loaded – this is available via the ! command whilst the Oric is on power. This can be paged in and out alongside the original Oric ROM so that both can be used. On a 48k Oric, memory #0000 to #BFFF (48k) is RAM and #C000 to #FFFF (16k) is ROM. The ROM area is used as follows.

ROM	Start Address	End Address	Length
Original Oric ROM	#C000	#FFFF	16k
Interface EPROM	#F800	#FFFF	2k
Extended ROM or DOS	#C000	#FFFF	16k

These 3 ROM areas can be paged in and out, but only one is available at a time. The RAM area is always available, and is used to copy data from one ROM to another. Within the Extended ROM, #C000 to #CFFF (4k) is working space (with a lot unused), #D000 to #DFFF (4k) is normally unused and #E000 to #FFFF (8k) hold the DOS itself.

When the Oric is booted with the Cumana disk interface connected, the EPROM in the Interface is paged in, and the 6502 processor looks for a RESET Address in memory locations #FFFC/D and it finds the address #F800. The 6502 starts executing the code at #F800 in the EPROM, to initialise the Oric completely. With the EPROM paged in, the following sequence takes place:

- Copy the ! handler routine down to page #4, ready for use later.
- Test some of the RAM (DOS area) in the interface.
- Initialise and set-up the Oric, as the original Oric ROM would do.
- Look for the DOS on a system disk (a file called CUMANA.DOS).
- Load it into 8k of RAM, from #6800 to #87FF (Execution address T=#8300).
- Display the DOS version.
- Complete initialisation sequence with 'ORIC EXTENDED BASIC ….' Message.
- Look for a file on the disk called BOOTUP.COM and put either !BOOTUP or a null onto the command line.
- Execute the boot routine in the DOS file (still in RAM) at #8300.

The DOS file, which is still in RAM, then takes over with the following:

- Set screen parameters, page out the EPROM and page in the Extended ROM / DOS area.
- Copy the DOS from #6800 in RAM to #E000 in the Extended ROM / DOS area.
- Page back to the original Oric ROM.
- Jump to accept a command from the input buffer.

If the file BOOTUP.COM was found on the disk, the input buffer will contain !BOOTUP, and this file will be loaded from disk. Otherwise, the input buffer will be empty and the Oric will wait for a command from the keyboard.

This manual contains a disassembly for each of the following:

- Cumana Interface EPROM.
- The ! routine to page in the Extended ROM, execute a DOS command and return to Oric.
- The DOS commands in extended ROM.

Around the time of RipDOS v2.7 (see version history), I realised that there was a lot of unused space in the Cumana disk interface: all of #D000 to #DFFF, and a lot in #C000 too. I started writing code and utilities to use this area without taking up precious RAM. The most useful was a disassembler, and I have included that in the manual as an example of how to use the extended DOS facility available in RipDOS.

DOS Commands

File Specifications

Programs or files (as we will refer to them from now on) are normally given names to identify them. This name or filespec abbreviated to <fsp> can consist of numbers and letters up to a total of six. In addition each filespec can have a three character extension e.g. !SAVE “PROG25.BAS”. As well as the 6 character filespec and extension the disk drive number can be specified. e.g. “1-PROG25.BAS” This is a valid filename with optional extension characters that will operate using drive one.

Below are some examples of correct and incorrect filenames:

Correct:
“JOB.1A” Filename including an extension on the default drive.
“2-FILE85” Filename with no extension on disk drive 2.
“2-JOB1.MY” Filename with extension on drive 2.

Incorrect:
“1-PROGRAM.25” Name too long (6 characters max).
“FRED+3” ‘+’ is not alphanumeric and is therefore not allowed.

Ambiguous filenames

You will notice that some DOS instructions include <afsp> in the command line meaning ambiguous filespec. Commands that have <afsp> will allow you to specify not only a single file but a range of files if you wish. The way in which to do this is by using the two wildcard characters * and ?. ‘*’ specifies any number of letters and/or digits in any combination and ‘?’ specifies any single letter or digit. The only requirement is that you must not specify characters after the ‘*’ in either the name part or the extension part of the <afsp>.

Below are some correct and incorrect examples using wildcards:

Correct:
“P*.JO?” **-** This will operate on any file with ‘P’ as the first character of the name part and ‘JO’ as the first part of the extension.
“T??E.B?” **-** Only files with four name characters and two extension characters will be looked at e.g. ‘TIME.B1’ ‘TUNE.BA’ ‘TONE.B5’ etc.

Incorrect:
“P*J2.NO?” **-** There are characters after the ‘*’ in the name part of the <afsp>.
“PROG1?.B*B” **-** There is a character after the ‘*’ in the extension part of the <afsp>.

Specifying disk drives

Some instructions contain <source drive> and <target drive>. The drive (number) being read from is the source drive and the drive (number) being written to is the target drive. If you do not specify a drive number where the instruction requires one then the default drive will be allocated.

Information messages

From time to time, messages are displayed by the system to keep you informed of what is happening. These messages are self explanatory e.g. **Saving …..** PROG1.BAS

Disk Operating System Commands

All Cumana DOS commands for the Oric are preceded by a special character known as ‘pling’ or ‘!’ this is achieved by pressing the shift key and number 1 e.g. !FORMAT.

Where you see <RETURN> in the examples, press the RETURN key on the key board after typing the line. The following is a summary of the Cumana DOS commands in alphabetical order.

!BACKUP <source drive> TO <target drive>

Backup is a command that allows a complete copy to be made from any master diskette. One or more disk drives can be specified and single or dual drive backups are possible. For an explanation of the backup command see Chapter 5 "Making a backup". Remember the target disk will be totally overwritten, it is advisable to ensure that you no longer require any of the previous data.

!CALL address

RipDOS modifications to !CALL:

- This is a new command, added to enable a machine code routine sitting in the shadow ROM area to be called directly from BASIC.
- The syntax is: !CALL address   followed by any other information / variables required by the routine called.
- !CALL works just like CALL, but has the shadow ROM #C000 to #FFFF paged in, instead of the normal Oric ROM.
- On exit (RTS), the main ROM is paged back in again.
- The address called can be any from #0000 to #FFFF, not necessarily above #C000.

!COPY

The copy command is probably the most versatile and therefore the most complex function of the DOS. It allows you to copy the contents of an existing (old) file or set of files, to a new file or set of files. This is a generalised view however there are a number of variations and these are:

The new file(s) can be on the same diskette as the old file(s) or on a different diskette. A number of old files can be merged into one new file and you can give the new file the same name as the old file or change it. Copying (either to the same or a different diskette) can be done using one or more disk drives.

The new file(s) can be given a protect or unprotect status to guard against accidental writes to the diskette, or alternatively the new file can assume the status of the old file. In addition to the above variations there are a number of constraints and these are fully explained in the notes which follow the options.

Notes:

1. The drive that is used to copy from is the source drive
2. The drive being copied to is the target drive.
3. The file being copied is the old file.
4. The file being created is the new file.

Instructions:

!COPY “old filename” TO “new filename” ,(opt) ,(opt) ,(opt) <RETURN>

Caution: Make sure you specify the correct drive number in the filename i.e. source drive goes with old filename, and target drive goes with new filename.

Copy options: There are three options which you can specify as part of the instruction and they are described as follows:

,P or ,N (This is the write protect option). Only one of these can be specified. If you want the new file to be write protected or locked specify ,P. If you want the new file to be unprotected then specify ,N. If you want the new file to assume the same protection as the old file then do not specify either.

,C (Single drive option). If you want to use one disk drive for copying, specify ,C. This will ensure that the messages 'Insert source disk' and 'Insert target disk' are displayed as appropriate so that you know when to change diskettes. (Caution - avoid mixing them up!)

,O or ,M (over-write or merge options). Only one of these can be specified. It is often necessary to create a new file with the same name as an existing file (for example if you were updating files). Ordinarily, this would result in a message telling you that a file with the same name already exists and that no copying will take place. If however you specify ,O the existing file will be overwritten and a message telling you so will be displayed. The exception of course, is when the disk or file is write protected which renders the ,O option ineffective.

You can also merge a number of files into one file by using the ,M option in conjunction with the wildcard characters. For example, the instruction:

```
!COPY "0-JOB*.B?" TO "PROG1" ,M
```

Will select all files (on drive 0 disk) that satisfy "JOB*.B?" and merge them into one file called "PROG1".

If you are not sure about the use of wildcard characters refer to the explanation earlier. To keep you informed, a message is displayed each time an old file is merged into a new file.

Notes:
1. When a filename is created as a result of copying, a message to that effect is displayed.
2. If the new filename is the same as the old filename, then the new filename need not be specified.
3. The old filename need not include the disk drive number if copying from the default drive. The same applies to the new filename.
4. If you are not using the default drive, then you must specify the drive number in old and new filenames, even if the drive number is the same.
5. You can copy all the files on a disk in one instruction, either by using the '*' to give a totally ambiguous filename, or by specifying the disk drive number without quotes. See examples 5 and 6 below. This method of copying files is often faster than a backup especially if there are only a few files on the disk.

The following examples are given to show the many ways in which the copy instruction can be used.

1. !COPY "0-FRED.EXT" TO "1-JOE.12" ,P <RETURN>
   To copy the file FRED.EXT on drive 0 to file JOE.12 on drive l, and give it a protected status.

2. !COPY "1-ALEX" TO 2 ,N <RETURN>
   To copy the file ALEX on drive 1 to a file of the same name on drive 2 and give it an unprotected status.

3. !COPY "0-JOB*" TO 1 <RETURN>
   To copy all the files whose names begin with JOB but which have no extension characters, from drive 0 to 1. The destination files have the same protection status as the source files.

4. !COPY "0-JOB.*" TO "2-BOX.205" ,P ,M <RETURN>
   To merge all files with the name JOB and that have extensions on drive 0 to a file called BOX.205 on drive 2 with the protected status. Note the file BOX.205 should not already exist on drive 2 else the error file exists will be displayed.

5. !COPY "0-*.*" TO 1 <RETURN> or !COPY 0 TO 1 <RETURN>
   To copy all the files on drive 0 disk to the disk in drive 1 with no changes to the protection status.

6. !COPY "0-*.*" TO 0 ,C <RETURN> or
   !COPY 0 TO 0 ,C <RETURN> or
   !COPY 0 ,C <RETURN>
   To copy all the files on one disk to another disk using the default drive only (drive 0).

!DEL <afsp>

The delete command is used to remove a specified file or set of files from the disk. The name of the file(s) is removed from the directory and the space that was previously occupied is free to be used by another file.

Notes:
1. You cannot delete a file that has the write protect status set. If you attempt to do so the message "File is write protected" will be displayed.
2. The directory command will tell you if a file is write protected.

!DIR <afsp> and !LDIR

The name and position of every file saved on the disk is held in a reserved area of the disk known as the directory. In addition, the length of the file and its write protect status is also kept and together these items can be displayed along with the filename. When requesting a directory the ambiguous filename can be used so that you can specify the drive number and a range of files.

Examples:

!DIR <RETURN>
This will simply display all the files on the default drive.

!DIR 1 <RETURN>
This will display all the files on drive 1.

!DIR "2-JOB.*" <RETURN>
This will display all the files on drive 2 that have the filename JOB and any extensions.

When you request a directory of all the files the display will look something like this:

```
Directory of Drive 1-MY DISK

ZAPPER.GAM     8P          PIRATE.BAS    65P
FROGGY.COM    23           FREDDY.TXT    66P
DELTA          1

5 Files                    535 Blocks free
```

Where:
On the top line 1-MY DISK tells you the drive requested was drive 1 and the name given to the disk when formatting was MY DISK. The bottom line gives the number of files, followed by the number of sectors free that can still be used for future files. The actual files are shown between the top and bottom lines: in this case 5 files are shown. The number shown after the filename is the length of the file in sectors and the letter 'P' shows files that are write protected, in this case 3 files are write protected and 2 are not. If there are more files in the directory than the screen can display, the space bar can be used as a scroll stop i.e. press the space bar once to stop the screen scrolling, press again to continue.

Notes:
1. For a 40 track drive with 16 sectors per track the total number of sectors on a disk is 640.
2. Of these, one sector is reserved for every 15 directory entries.

RipDOS modifications to !DIR:
- !LDIR added as a new command
- It works exactly the same as !DIR, but directs the output to a printer.

!DRV <drive number>

Some DOS commands give the user the option to specify a drive number in the command line. If the number is not specified then the default drive is used. The command !DRV is used to alter the value of the default drive e.g.

!DRV 2 <RETURN>
will set the default drive to drive 2 and any command issued with the default drive option will now go to drive 2

Notes:
1. This command is only of use with more than one drive unit.
2. Typing !DRV <RETURN> and not specifying a drive number will set the default drive to 0.

!DSTEP <val>

This command is used to alter the disk drive seek step time. Where <val> is a numeric value either 6, 12, 20 or 30 ms (milliseconds). Your master system diskette is factory pre-set to 12 ms. If you intend altering the seek access time of the disk drive make sure the drive can operate at the new speed. Permanent damage may result if the disk drive mechanics are operated at the wrong speed. If in doubt consult the manufacturer's recommended operating limits. After pressing the reset button on the interface the disk drive head will seek to track zero at 30 ms.

Notes:
1. When a new dstep speed is issued the system tracks on the disk are updated so that the new speed will operate each time the system is used.
2. A temporary change in speed can be achieved by write protecting the disk and then using the dstep command.

RipDOS modifications to !DSTEP:
- !DSTEP changes only the onboard information.
- Use !SET to update the system information on the disk in the default drive..

!FORMAT "disk name"

This command is used to format, initialise and name a disk in a specified drive unit. The disk name can be a maximum of any nine characters (alphanumeric, symbolic, and spaces) and can be prefixed by a drive number as in filenames i.e.

!FORMAT “1-MY DISK” <RETURN>

A message will ask you to load the disk in the specified drive and press <RETURN>
Formatting will then take place and a message will tell you when the operation has finished.

!LOAD “filename” (,D) (,N) (,J) (,A<address>)

This command is used to load into memory, either a basic program or a machine code program. If the program you wish to load was saved with either:

A) The auto option (for basic programs), or
B) The transfer address (for machine code programs)

then the program will auto-run as soon as it has loaded.

You will note there are four options that you can specify:

,D - This will display the start, end and transfer addresses in hexadecimal (for a clear understanding see !SAVE).

,N - If the file was saved using the auto run option or transfer (execution) address, this option will prevent it from running after the file has loaded.

,J - Use this option if you want the file to be joined on to the end of a file currently in memory. If the program you are loading was saved using the auto run option then once loading has finished the two joined programs will run automatically. You can prevent auto run by specifying the ,N option. The J option should only be used with basic programs.

,A <address> - When a file is saved, the start and end addresses are saved with the file and so when loading the computer knows where to put the file. If you want the program to load at an address other than the start address saved with the file, then the program can be forced to load at a different address using the ,A <address> option. Note: The address can be either decimal or hexadecimal, in the latter case prefix the value with the '#' symbol.

RipDOS modifications to !LOAD:

- the DOS is NOT paged out if the auto start (Transfer) address is in the shadow ROM area (#C000 to #FFFF).
- If the T address is in normal RAM, or there is no T address specified, then the DOS is still paged out.

!NAME "diskname"

RipDOS modifications to !NAME:

- This is a new command, added to name or re-name a disk.
- Diskname can be up to 9 characters, and can include the drive number (as in !FORMAT)

!PROT <afsp> (,opt)

This command is used to change the protection status of the ambiguous filename.

Where (,opt) is
,P to protect a file from writing.
,N to remove protection from a file.
,I to protect a file from writing and remove its name from the directory listing when displayed i.e. to make the file invisible.

Note: if no option is specified the P option is given by default.

!READ d,t,s,a

RipDOS modifications to !READ:

- This is a new command, added to read a defined single sector from a disk.
- d is the disc drive number.
- t is the track number (0 to 39 / 79).
- s is the sector number (1 to 16).
- a is the address at which to load the block into memory (256 bytes will be loaded).
- It has not been tested on a double sided disk drive.

!REN <old fsp > TO <new fsp>

This command is used to rename a file on the current default drive where <old fsp> is the name you wish to change and <new fsp> is the name you wish to change to.

Notes:

1. Ambiguous filenames cannot be used.
2. The new filename must not exist on the diskette prior to command execution or else an error message will result.

!SAVE <filename> (,AUTO)

Three types of file can be saved using the !SAVE command. The above example shows how to save basic programs, where <filename> is the name you wish to call the file, and the auto option decides whether the file will automatically run after loading.

!SAVE <filename> (,A start) (,E end) (,T transfer) <RETURN>
This example shows how to save machine code programs where, <filename> is the name of the file, A and E define the start and end of the block of memory to be saved and T decides the transfer (execution) address that the file will begin execution. The !SAVE command can also be used for saving a block of memory. Use the above example but omit the ,T option, this will prevent the file from automatically running after it has been loaded.

Notes:
1. Not specifying the options AUTO (for a basic file) or ,T (for a machine code file) means that the program will not run automatically when loaded. You will have to type in RUN <RETURN> for a basic program or CALL <address><RETURN> for a machine code program.
2. If the filename specified already exists, a message to this effect is displayed.

Following are some examples of instructions:

!SAVE "PROG" ,AUTO <RETURN>
A basic program is stored under the filename PROG. The AUTO Option has been specified so the program will automatically run when it is loaded.

!SAVE "PROGM" ,A#B400 ,E#C250 ,T#B420 <RETURN>
An area of machine code memory from address hexadecimal B400 to C250 is saved under the filename PROGM. The transfer address is B420. When file is loaded, the code will begin execution from this address.

!SAVE "PROGA" ,A#B400 ,E#C250 <RETURN>
An area of memory saved as above but this time without the transfer address. When the file is loaded it will not automatically run instead you would have to use CALL.

!SET (drv), (trks), (sides)

The set command is used to tell the operating system how many drive units it has and what type they are. Where (drv) is the disk drive number you wish to set, (trks) is a value either 40 or 80 depending on what type of drive you have, and (sides) is a letter either 'S' for single sided drives or 'D' for double sided drives. The set command works by updating the master system diskette. If the master system diskette is write protected i.e. it has the write enable notch covered, an error message will be displayed. However the resident system will be temporarily set allowing you to use the drive. The master system disk is factory pre-set for both drive 0 and drive 1. To remove a drive from the system type !SET (drv) <RETURN> without any parameters. Examples:

!SET 2,40,D <RETURN>
will set drive 2 as a 40 track double sided drive.

!SET 2 <RETURN>
will remove drive 2 from the system
RipDOS modifications to !SET:
- !SET d,t,s works as above, but changes only the onboard information.
- !SET <Return> copies out the onboard system information to the default drive, including the DSTEP value

!STAT

The stat command is used to find out what has been set with the !SET command. The display should look something like this:

Drive 0 - 40 Tracks Single-sided. Drive 1 - 40 Tracks Single-sided.

RipDOS modifications to !STAT:
- Displays the DSTEP value too.
- !STAT 0 added to display the information from the system track on the default drive.

Text Filing

!OPEN a file (for writing or reading),
!CLOSE a file (from reading or writing),
!GET data from a file,
!PUT data in a file

These four instructions are concerned with data (as opposed to arrays and programs) and are interdependent to a large degree. For example before you can put data in a file, you must OPEN a new file for writing (you cannot put data in an existing file). When writing is finished you must CLOSE the file. Once the file is CLOSED (created) you can OPEN it for reading only, GET data from it and then CLOSE it again.

!OPEN <filename> (,opt)

To open a file for reading or writing. Where <filename> is the name given to the data file and (,opt) is one of two options. They are;

,R	if you wish to read from an existing file (using the GET instruction).
,W	if you wish to write to a new file (using the PUT instruction).

Notes:
1. If you specify the R option be sure that the file already exists on the diskette.
2. If you specify the W option the filename must NOT already exist on the disk.

!CLOSE (,opt)

To close a file that is currently open (for reading or writing). If the file was open for writing, a new file is created by CLOSE. The options are;

,R	if the file was open for reading.
,W	if the file was open for writing.

If you wish to close a read file and a write file together do not specify an option.

!PUT <data>

Note: the PUT data instruction will not work unless a file has been opened for writing i.e. using the W option. The <data> must be in the form of expressions. An expression can be a numeric or string variable. All expressions must be separated by commas. A numeric expression will be written to the disk as a single byte and a string expression will be written to the disk as a series of bytes.

!GET <data>

Note: the GET data instruction will not work unless a file has been opened for reading i.e. using the R option. Each <data> expression can be numeric or string, and expressions must be separated by commas. The expression format is the same as the PUT instruction.

!STORE <array name ><,filename>

To store an array (string, real or integers) under a specified filename. Where <array name> is the name already allocated to the array you wish to store, note the array name should be used without any dimensions.

Notes:
1. The array can have any number of dimensions but they must not be included as part of the array name.
2. The array must already be in memory, you cannot type in the instruction and then the array.
3. If you do not specify an extension with the filename the default extension .DAT will be used.

!RECALL<new array name><,filename>

To read an array (string, real or integer) and load it into RAM under your own array name. Where <new array name> is the name you wish to give the array when loaded, and <filename> is the name the array was stored as. If you do not specify an extension in the filename the default extension. DAT will be used. An array name can have any amount of memory allocated to it i.e. any number of dimensions. If the array specified is larger than the file on the disk, the remainder of the array is filled with 0's or null strings.

If the array specified is smaller than the file on the disk then only enough data to fill the array is read in from the disk.

New Feature of RipDOS

A routine has been added so that an additional table of commands and their code can be installed in an unused area of memory from #D000 to #DFFF.

The facility, which allows a new set of commands to be added, works through the normal command interpreter routine in the DOS.

If an unrecognised command is encountered, the routine looks to see if there is another look up table in page #D000. If it finds a match there, it will run the routine. If it does not find a match, it will come up with an error message.

The command word table MUST start at #D000 and each entry must be separated with a null (0). Don't forget that some words may be partly or completely tokenised. For example, FORGET will be represented by #8D (FOR) #BE (GET) #00 (null to separate).

#D000 to #D0FF is available for the new table of words, which MUST end with #FF after the null of the last entry.

Memory #D0C0 to D0FF is reserved for the corresponding table of addresses for the new routines. Each address is 2 bytes long with the low byte first. There must be a 2 byte entry for each new command word in the first part of the table. Each address must be 1 less than the starting address for its routine, since 1 is added when the routine is called using the RTS instruction.

Memory #D100 to DFFF is available for the routines themselves. This is in an unused area of DOS memory, and provides huge flexibility for providing your own set of additional utilities without using RAM.

There is an example available in a separate file, with its own instruction manual and full disassembly. XTRA01.DOS gives a few extra commands, including a full machine code disassembler, without using any ORIC RAM.

Disassembly for Cumana Disk Interface EPROM

Initialise the computer

```
#F800    78          SEI            Disable IRQ interrupts
#F801    D8          CLD            Clear the decimal flag
#F802    A2 FF       LDX #FF
#F804    9A          TXS            Initialise stack pointer
#F805    E8          INX            Set A, X and Y to zero
#F806    8A          TXA
#F807    A8          TAY
#F808    CA          DEX            Wait about 0.3 seconds
#F809    D0 FD       BNE F808
#F80B    88          DEY
#F80C    D0 FA       BNE F808
#F80E    9D 00 C0    STA C000,X     Fill page #C000 with zero
#F811    9D 00 C1    STA C100,X     Fill page #C100 with zero
#F814    9D 00 02    STA 0200,X     Fill page #0200 with zero
#F817    95 00       STA 00,X       Fill page #00 with zero
#F819    E8          INX
#F81A    D0 F2       BNE F80E
#F81C    A2 8B       LDX #8B        Copy 139 bytes from
#F81E    BD 32 FB    LDA FB32,X     #FB33 down to page 4 at
#F821    9D 6D 04    STA 046D,X     #046E for the ! handler
#F824    CA          DEX
#F825    D0 F7       BNE F81E
#F827    A0 00       LDY #00        Test the RAM in the interface
#F829    B9 00 C0    LDA C000,Y     for 1 page #C000
#F82C    AA          TAX
#F82D    A9 55       LDA #55
#F82F    99 00 C0    STA C000,Y
#F832    D9 00 C0    CMP C000,Y
#F835    D0 13       BNE F84A       Branch if error found
#F837    A9 AA       LDA #AA
#F839    99 00 C0    STA C000,Y
#F83C    D9 00 C0    CMP C000,Y
#F83F    D0 09       BNE F84A       Branch if error found
#F841    8A          TXA
#F842    99 00 C0    STA C000,Y
#F845    C8          INY
#F846    D0 E1       BNE F829       Loop to continue testing RAM
#F848    F0 1B       BEQ F865       otherwise carry on with set-up routine
```

Error in Interface RAM

```
#F84A    A0 00       LDY #00        Error found in RAM
#F84C    A9 1A       LDA #1A        Turn the screen black
#F84E    99 80 BB    STA BB80,Y
#F851    99 80 BC    STA BC80,Y
#F854    99 80 BD    STA BD80,Y
#F857    99 80 BE    STA BE80,Y
#F85A    99 FE BE    STA BEFE,Y
#F85D    C8          INY
#F85E    D0 EE       BNE F84E       Load offset for error message
#F860    A2 26       LDX #26        ‘Interface adjustment required’
#F862    4C 6A F9    JMP F96A       Jump to error routine
```

```
#F865   A9 FF         LDA #FF        Carry on setting up system
#F867   A2 97         LDX #97
#F869   85 A9         STA A9         Set line number to command mode
#F86B   85 A6         STA A6         Set HIMEM
#F86D   86 A7         STX A7
#F86F   85 A2         STA A2         Set pointer for bottom of string area
#F871   86 A3         STX A3
#F873   8D C1 02      STA 02C1       Set address for character set
#F876   8E C2 02      STX 02C2
#F879   A2 1B         LDX #1B
#F87B   BD BD FB      LDA FBBD,X     Copy ‘fetch next non-space character’
#F87E   95 E2         STA E2,X       routine down to #E2 >
#F880   CA            DEX
#F881   10 F8         BPL F87B
#F883   A9 0F         LDA #0F
#F885   8D 4E 02      STA 024E       Set keyboard initial repeat delay
#F888   A9 02         LDA #02
#F88A   8D 4F 02      STA 024F       Set keyboard successive repeat delay
#F88D   A2 12         LDX #12        Copy vector JMP statements
#F88F   BD 77 FA      LDA FA77,X     from #FA77 down to
#F892   9D 38 02      STA 0238,X     #0238 >
#F895   CA            DEX
#F896   10 F7         BPL F88F
#F898   A9 4C         LDA #4C        JMP instruction
#F89A   85 1A         STA 1A         to print ‘Ready’
#F89C   85 21         STA 21         for USR command
#F89E   85 C3         STA C3         to evaluate numeric functions
#F8A0   8D FB 02      STA 02FB       for ‘&’ routine
#F8A3   A9 B0         LDA #B0        Store #CCB0 for ‘Ready’
#F8A5   A2 CC         LDX #CC
#F8A7   85 1B         STA 1B
#F8A9   86 1C         STX 1C
#F8AB   A9 36         LDA #36        Store #D336 for USR
#F8AD   A2 D3         LDX #D3
#F8AF   85 22         STA 22
#F8B1   86 23         STX 23
#F8B3   8D FC 02      STA 02FC       Store #D336 for &
#F8B6   8E FD 02      STX 02FD
#F8B9   A9 90         LDA #90        Store #0490 for !
#F8BB   A2 04         LDX #04
#F8BD   8D F5 02      STA 02F5
#F8C0   8E F6 02      STX 02F6
#F8C3   A9 00         LDA #00
#F8C5   8D FE 04      STA 04FE
#F8C8   8D FF 04      STA 04FF
#F8CB   20 F9 FC      JSR FCF9       Call a routine in the original ROM at
#F8CE   B8 F8         DTA #F8B8      #F8B8 for NMI service routine
#F8D0   A9 07         LDA #07
#F8D2   8D 6C 02      STA 026C       Ink 7
#F8D5   A9 10         LDA #10
#F8D7   8D 6B 02      STA 026B       Paper 0
#F8DA   A9 50         LDA #50
#F8DC   85 31         STA 31         Set screen line width
#F8DE   A9 30         LDA #30
#F8E0   85 32         STA 32         8-multiple line width
#F8E2   A9 03         LDA #03
#F8E4   85 C2         STA C2         String pointer size
#F8E6   A9 00         LDA #00
#F8E8   85 D7         STA D7         Sign extend byte
#F8EA   85 88         STA 88         Temporary string stack
#F8EC   85 2F         STA 2F         Next byte to/from cassette
```

```
#F8EE   48          PHA
#F8EF   8D 00 05    STA 0500     Zero start of basic
#F8F2   8D 01 05    STA 0501
#F8F5   8D 02 05    STA 0502
#F8F8   8D F7 02    STA 02F7
#F8FB   85 2E       STA 2E       Enable screen output
#F8FD   8D F1 02    STA 02F1     Printer off
#F900   8D F2 02    STA 02F2     Edit off
#F903   8D F4 02    STA 02F4     TROFF
#F906   A9 88       LDA #88
#F908   85 85       STA 85       String block stack pointer
#F90A   A9 02       LDA #02
#F90C   8D C0 02    STA 02C0     Set screen to TEXT mode
#F90F   A9 01       LDA #01
#F911   A0 05       LDY #05
#F913   85 9A       STA 9A       Set pointers to
#F915   84 9B       STY 9B       Start of Basic
#F917   A9 03       LDA #03
#F919   85 9C       STA 9C       End of Basic
#F91B   84 9D       STY 9D
#F91D   85 9E       STA 9E       End of Variables
#F91F   84 9F       STY 9F
#F921   85 A0       STA A0       End of Arrays
#F923   84 A1       STY A1
```

Insert system disk and find DOS

```
#F925   A2 00       LDX #00      Message offset for ‘Insert system disk’
#F927   20 8A FA    JSR FA8A     Print message
#F92A   A2 D8       LDX #D8
#F92C   8E 10 03    STX 0310
#F92F   A0 0B       LDY #0B
#F931   20 DF FB    JSR FBDF     Read from disk
#F934   A9 13       LDA #13      Set input address from disk to #C013
#F936   A0 C0       LDY #C0
#F938   8D 03 C0    STA C003
#F93B   8C 04 C0    STY C004
#F93E   A9 00       LDA #00      Set to Track 0
#F940   A2 01       LDX #01      Set to Sector 1
#F942   20 D7 FB    JSR FBD7     Load the block at Track A, Sector X
#F945   AD 1B C0    LDA C01B     Fetch the DSTEP value
#F948   29 03       AND #03
#F94A   8D 1B C0    STA C01B
#F94D   A2 07       LDX #07      Copy 8 bytes of system track information
#F94F   BD 23 C0    LDA C023,X
#F952   9D 23 C1    STA C123,X
#F955   CA          DEX
#F956   10 F7       BPL F94F
#F958   A2 08       LDX #08      Set file name to load CUMANA.DOS
#F95A   BD 5D FA    LDA FA5D,X
#F95D   9D 2C C1    STA C12C,X   Set filename in #C12C >
#F960   CA          DEX
#F961   10 F7       BPL F95A
#F963   20 1A FA    JSR FA1A     Look for file CUMANA.DOS
#F966   D0 08       BNE F970     Branch if successful
#F968   A2 14       LDX #14      otherwise load offset for ‘No system on disk’ error
#F96A   20 8A FA    JSR FA8A     Print the error message
#F96D   4C 6D F9    JMP F96D     and hang up.
```

Load DOS into RAM

```
#F970     BD 2F C0        LDA C02F,X      File CUMANA.DOS found
#F973     8D 01 C0        STA C001        Set the Track to read it in
#F976     BD 2E C0        LDA C02E,X      X is the position of file in the directory
#F979     8D 02 C0        STA C002        Set the Sector to read it in
#F97C     20 DD FB        JSR FBDD        Read in the (first) block
#F97F     AD 27 C0        LDA C027        Get the Start address for loading
#F982     AE 28 C0        LDX C028
#F985     8D 4B C1        STA C14B        Save Start address for loading (#6800)
#F988     8E 4C C1        STX C14C
#F98B     85 0C           STA 0C          Save Start address for loading
#F98D     86 0D           STX 0D
#F98F     AD 2B C0        LDA C02B        Get the T address
#F992     AE 2C C0        LDX C02C
#F995     8D 4D C1        STA C14D        Save T address (#8300)
#F998     8E 4E C1        STX C14E
#F99B     A2 0A           LDX #0A         Set offset for first block of file
#F99D     BD 23 C0        LDA C023,X      Get the number of bytes in this block
#F9A0     F0 16           BEQ F9B8        Branch if done
#F9A2     8D 41 C1        STA C141        Save as counter
#F9A5     A0 00           LDY #00         Counter for reading bytes
#F9A7     E8              INX
#F9A8     BD 23 C0        LDA C023,X      Get a byte of the file
#F9AB     91 0C           STA (0C),Y      Store it at the required address
#F9AD     E6 0C           INC 0C          Increment save address
#F9AF     D0 02           BNE F9B3
#F9B1     E6 0D           INC 0D
#F9B3     CE 41 C1        DEC C141        Decrement byte pointer
#F9B6     D0 EF           BNE F9A7        Go back for the next byte until end of block
#F9B8     AD 23 C0        LDA C023        Get Track for next block
#F9BB     AE 24 C0        LDX C024        Get Sector for next block
#F9BE     F0 07           BEQ F9C7        Branch if end of file
#F9C0     20 D7 FB        JSR FBD7        Read in next block of file
#F9C3     A2 02           LDX #02         Set offset for second (and rest) block of file
#F9C5     D0 D6           BNE F99D        Branch back to copy the bytes down
#F9C7     A2 13           LDX #13         Code for message 'No system on disk'
#F9C9     20 8A FA        JSR FA8A        Print message on status line.  WHY?
#F9CC     AD 4B C1        LDA C14B        Re-set the Start address pointer in #0C/D
#F9CF     85 0C           STA 0C
#F9D1     AD 4C C1        LDA C14C
#F9D4     85 0D           STA 0D
#F9D6     A0 00           LDY #00         Initialise byte counter
#F9D8     B1 0C           LDA (0C),Y      Get first byte of file (low byte of DOS entry address)
#F9DA     8D 96 04        STA 0496        Save it
#F9DD     C8              INY
#F9DE     B1 0C           LDA (0C),Y      Get second byte of file (high byte of DOS entry address)
#F9E0     8D 9B 04        STA 049B        Save it
#F9E3     A9 02           LDA #02
#F9E5     8D 80 BB        STA BB80        Set status line
#F9E8     C8              INY
#F9E9     B1 0C           LDA (0C),Y      Get next byte of file (DOS version)
#F9EB     99 80 BB        STA BB80,Y      Put DOS version on status line
#F9EE     D0 F8           BNE F9E8        Loop until end of DOS version
#F9F0     A0 00           LDY #00
#F9F2     B9 E7 FA        LDA FAE7,Y      Do 'ORIC EXTENDED BASIC ….' message
#F9F5     F0 09           BEQ FA00        Branch at end of message
#F9F7     AA              TAX
#F9F8     20 F9 FC        JSR FCF9        Call routine in original ROM at
#F9FB     7C F7           DTA  F77C       #F77C to print character (in X) to screen
#F9FD     C8              INY
```

```
#F9FE    D0 F2              BNE F9F2        Loop to end of message
```

Look for file BOOTUP.COM

```
#FA00    A2 08              LDX #08         Index for data
#FA02    BD 66 FA           LDA FA66,X      Set up filename to BOOTUP.COM
#FA05    9D 2C C1           STA C12C,X
#FA08    BD 6F FA           LDA FA6F,X
#FA0B    95 35              STA 35,X        Set name of program to !BOOTUP
#FA0D    CA                 DEX
#FA0E    10 F2              BPL FA02        Loop until all characters done
#FA10    20 1A FA           JSR FA1A        Search disk for file BOOTUP.COM
#FA13    D0 02              BNE FA17        Branch if file found
#FA15    86 35              STX 35          or nullify !BOOTUP command if not present
#FA17    6C 4D C1           JMP (C14D)      Jump to transfer address (#8300) in DOS in RAM
                                            to copy DOS from #6800 > in RAM
                                            to #E000 > in Shadow ROM area.
```

Search directories for filename

```
#FA1A    A9 23              LDA #23         Search directories for file of  name specified in #C12C >
#FA1C    8D 03 C0           STA C003
#FA1F    A9 C0              LDA #C0
#FA21    8D 04 C0           STA C004        Set the address at which to load the block
#FA24    AD 26 C1           LDA C126        Get Track for first directory
#FA27    AE 25 C1           LDX C125        Get Sector for first directory
#FA2A    20 D7 FB           JSR FBD7        Load the block at Track A, Sector X (directory)
#FA2D    A2 03              LDX #03
#FA2F    8E 3F C1           STX C13F
#FA32    A0 00              LDY #00
#FA34    BD 23 C0           LDA C023,X      Get character of filename
#FA37    F0 12              BEQ FA4B        Branch if no file in this slot, and move to next slot
#FA39    B9 2C C1           LDA C12C,Y      Compare for the specified filename
#FA3C    DD 23 C0           CMP C023,X
#FA3F    D0 0A              BNE FA4B        Move to next block if match fails
#FA41    E8                 INX
#FA42    C8                 INY
#FA43    C0 09              CPY #09
#FA45    90 F2              BCC FA39        Match all 9 characters of filename
#FA47    AE 3F C1           LDX C13F
#FA4A    60                 RTS             Exit when match found

#FA4B    AD 3F C1           LDA C13F        Increment counters to next block in directory
#FA4E    18                 CLC
#FA4F    69 10              ADC #10
#FA51    AA                 TAX
#FA52    90 DB              BCC FA2F        Carry on looking in same block
#FA54    AD 23 C0           LDA C023        Get Track for next directory block
#FA57    AE 24 C0           LDX C024        Get Sector for next directory block
#FA5A    D0 CE              BNE FA2A        Carry on looking unless end of directories
#FA5C    60                 RTS             Exit if end of directories reached
```

Data for specified filenames

```
#FA5D    43 55 4D 41 4E     DTA CUMAN
#FA62    41 44 4F 53        DTA ADOS
#FA66    42 4F 4F 54 55 50  DTA BOOTUP
#FA6C    43 4F 4D           DTA COM
#FA6F    21 42 4F 4F 54     DTA !BOOT
#FA74    55 50 00           DTA  UP
```

```
                                        Data to be copied down to #0238 > .  Vectors for
#FA77    4C 7C F7          JMP F77C     Print character to screen
#FA7A    4C 78 EB          JMP EB78     Get key
#FA7D    4C C1 F5          JMP F5C1     Send byte to printer
#FA80    4C 65 F8          JMP F865     Print to status line
#FA83    4C 22 EE          JMP EE22     IRQ routine
#FA86    4C B2 F8          JMP F8B2     NMI routine
#FA89    40                RTI
```

Interface message handler

```
#FA8A    A0 1A             LDY #1A      Clear the status line
#FA8C    A9 20             LDA #20
#FA8E    99 82 BB          STA BB82,Y
#FA91    88                DEY
#FA92    D0 FA             BNE FA8E
#FA94    BD A1 FA          LDA FAA1,X   Get character of message
#FA97    F0 07             BEQ FAA0     00 = last character of message
#FA99    99 82 BB          STA BB82,Y   Put message on status line
#FA9C    E8                INX
#FA9D    C8                INY
#FA9E    D0 F4             BNE FA94     Loop until end of message
#FAA0    60                RTS
```

Data for system and error messages

```
#FAA1    49 6E 73 65 72    DTA Inser
#FAA6    74 20 73 79 73    DTA t sys
#FAAB    74 65 6D 20 64    DTA tem d
#FAB0    69 73 6B 2E 00    DTA isk.
#FAB5    0C 4E 6F 20 73    DTA  No s
#FABA    79 73 74 65 6D    DTA ystem
#FABF    20 6F 6E 20 64    DTA  on d
#FAC4    69 73 6B 00 0C    DTA isk
#FAC9    49 6E 74 65 72    DTA Inter
#FACE    66 61 63 65 20    DTA face
#FAD3    61 64 6A 75 73    DTA adjus
#FAD8    74 6D 65 6E 74    DTA tment
#FADD    20 72 65 71 75    DTA  requ
#FAE2    69 72 65 64 00    DTA ired
#FAE7    0C 4F 52 49 43    DTA  ORIC
#FAEC    20 45 58 54 45    DTA  EXTE
#FAF1    4E 44 45 44 20    DTA NDED
#FAF6    42 41 53 49 43    DTA BASIC
#FAFB    20 56 31 2E 31    DTA  V1.1
#FB00    0D 0A 60 20 31    DTA   ` 1
#FB05    39 38 33 20 54    DTA 983 T
#FB0A    41 4E 47 45 52    DTA ANGER
#FB0F    49 4E 45 0D 0A    DTA INE
#FB14    0A 0A 20 33 37    DTA    37
#FB19    36 33 31 20 42    DTA 631 B
#FB1E    59 54 45 53 20    DTA YTES
#FB23    46 52 45 45 0D    DTA FREE
#FB28    0A 0A 0A 52 65    DTA    Re
#FB2D    61 64 79 0D 0A 00 DTA ady
```

Data to copy to #046D > for ! handler

```
#FB33   C9 C8         CMP #C8       See separate disassembly at #046E
#FB35   F0 0D         BEQ FB44
#FB37   C9 27         CMP #27
#FB39   F0 09         BEQ FB44
#FB3B   C9 3A         CMP #3A
#FB3D   B0 05         BCS FB44
#FB3F   E9 2F         SBC #2F
#FB41   38            SEC
#FB42   E9 D0         SBC #D0
#FB44   60            RTS
#FB45   04 00 00 00   DTA
#FB49   4C 00 00      JMP 0000
#FB4C   4C E3 04      JMP 04E3
#FB4F   4C D3 04      JMP 04D3
#FB52   4C DB 04      JMP 04DB
#FB55   A9 00         LDA #00
#FB57   8D 81 04      STA 0481
#FB5A   A9 12         LDA #12
#FB5C   8D 85 04      STA 0485
#FB5F   A9 E0         LDA #E0
#FB61   8D 86 04      STA 0486
#FB64   08            PHP
#FB65   78            SEI
#FB66   8D 82 04      STA 0482
#FB69   68            PLA
#FB6A   8D 83 04      STA 0483
#FB6D   AD 80 04      LDA 0480
#FB70   48            PHA
#FB71   AD 81 04      LDA 0481
#FB74   20 E3 04      JSR 04E3
#FB77   AD 83 04      LDA 0483
#FB7A   48            PHA
#FB7B   AD 82 04      LDA 0482
#FB7E   28            PLP
#FB7F   20 84 04      JSR 0484
#FB82   08            PHP
#FB83   78            SEI
#FB84   8D 82 04      STA 0482
#FB87   68            PLA
#FB88   8D 83 04      STA 0483
#FB8B   68            PLA
#FB8C   20 E3 04      JSR 04E3
#FB8F   AD 83 04      LDA 0483
#FB92   48            PHA
#FB93   AD 82 04      LDA 0482
#FB96   28            PLP
#FB97   60            RTS
#FB98   08            PHP
#FB99   BA            TSX
#FB9A   FE 02 01      INC 0102,X
#FB9D   4C 44 02      JMP 0244
#FBA0   08            PHP
#FBA1   BA            TSX
#FBA2   FE 02 01      INC 0102,X
#FBA5   4C 47 02      JMP 0247
#FBA8   78            SEI
#FBA9   29 02         AND #02
#FBAB   8D 81 04      STA 0481
#FBAE   AD 80 04      LDA 0480
```

```
#FBB1    29 FD       AND #FD
#FBB3    0D 81 04    ORA 0481
#FBB6    8D 80 04    STA 0480
#FBB9    8D 14 03    STA 0314
#FBBC    60          RTS
                                   Data to copy down to Page #00
#FBBD    E6 E9       INC E9        Routine at #E2
#FBBF    D0 02       BNE FBC3      It holds the current programme position
#FBC1    E6 EA       INC EA        and is used to step through spaces until
#FBC3    AD B9 EC    LDA ECB9      it finds the next non-space character
#FBC6    C9 20       CMP #20
#FBC8    F0 F3       BEQ FBBD      (Branch for next character if space found)
#FBCA    4C 6E 04    JMP 046E      Test for statement delimiter or number
                                   Set to Write  - unused?
#FBCD    8D 01 C0    STA C001      Save the Track
#FBD0    8E 02 C0    STX C002      Save the Sector
#FBD3    A0 A0       LDY #A0       Flag for Write
#FBD5    D0 08       BNE FBDF      Branch always
```

Read from disk

```
#FBD7    8D 01 C0    STA C001      Save the Track
#FBDA    8E 02 C0    STX C002      Save the Sector
#FBDD    A0 80       LDY #80       Flag for Read
                                   Read (or write?) the block at Track A, Sector X
#FBDF    20 EC FC    JSR FCEC
#FBE2    20 E9 FB    JSR FBE9
#FBE5    20 F1 FC    JSR FCF1
#FBE8    60          RTS

#FBE9    8C 05 C0    STY C005      Part of read / write routine
#FBEC    AD 00 C0    LDA C000
#FBEF    29 03       AND #03
#FBF1    AA          TAX
#FBF2    BD D1 FC    LDA FCD1,X
#FBF5    2C 01 C0    BIT C001
#FBF8    10 02       BPL FBFC
#FBFA    09 10       ORA #10
#FBFC    8D 14 03    STA 0314
#FBFF    AE 80 04    LDX 0480
#FC02    8D 80 04    STA 0480
#FC05    29 6C       AND #6C
#FC07    85 F3       STA F3
#FC09    8A          TXA
#FC0A    29 6C       AND #6C
#FC0C    C5 F3       CMP F3
#FC0E    F0 23       BEQ FC33
#FC10    A9 52       LDA #52
#FC12    85 F3       STA F3
#FC14    A9 C1       LDA #C1
#FC16    85 F4       STA F4
#FC18    AD 05 C0    LDA C005
#FC1B    48          PHA
#FC1C    A9 C0       LDA #C0
#FC1E    8D 05 C0    STA C005
#FC21    20 8E FC    JSR FC8E
#FC24    68          PLA
#FC25    8D 05 C0    STA C005
#FC28    AD FE 04    LDA 04FE
#FC2B    D0 76       BNE FCA3
```

```
#FC2D       AD 12 03              LDA 0312
#FC30       8D 11 03              STA 0311
#FC33       A9 00                 LDA #00
#FC35       8D 06 C0              STA C006
#FC38       20 5C FC              JSR FC5C
#FC3B       D0 02                 BNE FC3F
#FC3D       18                    CLC
#FC3E       60                    RTS
#FC3F       29 18                 AND #18
#FC41       F0 60                 BEQ FCA3
#FC43       AD 06 C0              LDA C006
#FC46       30 5B                 BMI FCA3
#FC48       D0 05                 BNE FC4F
#FC4A       EE 06 C0              INC C006
#FC4D       D0 E9                 BNE FC38
#FC4F       09 80                 ORA #80
#FC51       8D 06 C0              STA C006
#FC54       A0 08                 LDY #08
#FC56       20 D5 FC              JSR FCD5
#FC59       90 DD                 BCC FC38
#FC5B       60                    RTS
#FC5C       AD 03 C0              LDA C003
#FC5F       85 F3                 STA F3
#FC61       AD 04 C0              LDA C004
#FC64       85 F4                 STA F4
#FC66       A9 10                 LDA #10
#FC68       2C 05 C0              BIT C005
#FC6B       70 1F                 BVS FC8C
#FC6D       AD 01 C0              LDA C001
#FC70       29 7F                 AND #7F
#FC72       CD 11 03              CMP 0311
#FC75       F0 0A                 BEQ FC81
#FC77       8D 13 03              STA 0313
#FC7A       A0 1C                 LDY #1C
#FC7C       20 D5 FC              JSR FCD5
#FC7F       B0 22                 BCS 5CA3
#FC81       AD 02 C0              LDA C002
#FC84       8D 12 03              STA 0312
#FC87       AD 05 C0              LDA C005
#FC8A       29 20                 AND #20
#FC8C       D0 17                 BNE FCA5
#FC8E       20 BA FC              JSR FCBA
#FC91       58                    CLI
#FC92       AD 18 03              LDA 0318
#FC95       30 FB                 BMI FC92
#FC97       AD 13 03              LDA 0313
#FC9A       91 F3                 STA (F3),Y
#FC9C       C8                    INY
#FC9D       D0 F3                 BNE FC92
#FC9F       E6 F4                 INC F4
#FCA1       D0 EF                 BNE FC92
#FCA3       38                    SEC
#FCA4       60                    RTS
#FCA5       20 BA FC              JSR FCBA
#FCA8       58                    CLI
#FCA9       AD 18 03              LDA 0318
#FCAC       30 FB                 BMI FCA9
#FCAE       B1 F3                 LDA (F3),Y
#FCB0       8D 13 03              STA 0313
#FCB3       C8                    INY
#FCB4       D0 F3                 BNE FCA9
#FCB6       E6 F4                 INC F4
```

```
#FCB8    D0 EF          BNE FCA9
#FCBA    AC 05 C0       LDY C005
#FCBD    78             SEI
#FCBE    8C 10 03       STY 0310
#FCC1    AD 80 04       LDA 0480
#FCC4    09 01          ORA #01
#FCC6    29 FD          AND #FD
#FCC8    8D 80 04       STA 0480
#FCCB    8D 14 03       STA 0314
#FCCE    A0 00          LDY #00
#FCD0    60             RTS

#FCD1    04 24 44 64    DTA            Data for read / write

#FCD5    98             TYA
#FCD6    29 FC          AND #FC
#FCD8    0D 1B C0       ORA C01B
#FCDB    A8             TAY
#FCDC    20 BD FC       JSR FCBD
#FCDF    20 E8 FC       JSR FCE8
#FCE2    29 18          AND #18
#FCE4    D0 BD          BNE FCA3
#FCE6    18             CLC
#FCE7    60             RTS
#FCE8    18             CLC
#FCE9    58             CLI
#FCEA    90 FE          BCC FCEA

#FCEC    48             PHA            Part of read / write routine
#FCED    A9 40          LDA #40
#FCEF    D0 03          BNE FCF4
#FCF1    48             PHA
#FCF2    A9 C0          LDA #C0
#FCF4    8D 0E 03       STA 030E
#FCF7    68             PLA
#FCF8    60             RTS
```

Call to original Oric ROM

```
#FCF9    08             PHP            Save all registers
#FCFA    48             PHA
#FCFB    98             TYA
#FCFC    48             PHA
#FCFD    8A             TXA
#FCFE    48             PHA
#FCFF    BA             TSX            Increment the return address (on the stack) by 2
#FD00    BD 05 01       LDA 0105,X     and copy the original return address
#FD03    85 0E          STA 0E         into #000E/F
#FD05    18             CLC
#FD06    69 02          ADC #02
#FD08    9D 05 01       STA 0105,X
#FD0B    BD 06 01       LDA 0106,X
#FD0E    85 0F          STA 0F
#FD10    69 00          ADC #00
#FD12    9D 06 01       STA 0106,X
#FD15    A0 01          LDY #01
#FD17    B1 0E          LDA (0E),Y
#FD19    8D 85 04       STA 0485       Fetch the 2 bytes following the call
#FD1C    C8             INY            and store in #0485/6 as the address
#FD1D    B1 0E          LDA (0E),Y     to call in the original ROM
```

```
#FD1F    8D 86 04      STA 0486
#FD22    A9 02         LDA #02
#FD24    8D 81 04      STA 0481      Set marker at #0481
#FD27    68            PLA           Recover all the registers
#FD28    AA            TAX
#FD29    68            PLA
#FD2A    A8            TAY
#FD2B    68            PLA
#FD2C    28            PLP
#FD2D    4C 9F 04      JMP 049F      Perform the call to the original ROM and return
```

NMI Routine

```
#FD30    48            PHA
#FD31    AD 81 04      LDA 0481
#FD34    48            PHA
#FD35    AD 85 04      LDA 0485
#FD38    48            PHA
#FD39    AD 86 04      LDA 0486
#FD3C    48            PHA
#FD3D    AD 80 04      LDA 0480
#FD40    29 FE         AND #FE
#FD42    8D 80 04      STA 0480
#FD45    8D 14 03      STA 0314
#FD48    A9 8D         LDA #8D
#FD4A    8D 85 04      STA 0485
#FD4D    A9 04         LDA #04
#FD4F    8D 86 04      STA 0486
#FD52    A9 02         LDA #02
#FD54    8D 81 04      STA 0481
#FD57    20 9F 04      JSR 049F
#FD5A    68            PLA
#FD5B    8D 86 04      STA 0486
#FD5E    68            PLA
#FD5F    8D 85 04      STA 0485
#FD62    68            PLA
#FD63    8D 81 04      STA 0481
#FD66    68            PLA
#FD67    40            RTI
```

IRQ Routine

```
#FD68    2C 14 03      BIT 0314
#FD6B    30 18         BMI FD85
#FD6D    AD 80 04      LDA 0480
#FD70    29 FE         AND #FE
#FD72    8D 80 04      STA 0480
#FD75    8D 14 03      STA 0314
#FD78    68            PLA
#FD79    68            PLA
#FD7A    68            PLA
#FD7B    AD 10 03      LDA 0310
#FD7E    29 5D         AND #5D
#FD80    8D FE 04      STA 04FE
#FD83    58            CLI
#FD84    60            RTS
#FD85    48            PHA
#FD86    8A            TXA
#FD87    48            PHA
```

```
#FD88     AD 81 04          LDA 0481
#FD8B     48                PHA
#FD8C     AD 85 04          LDA 0485
#FD8F     48                PHA
#FD90     AD 86 04          LDA 0486
#FD93     48                PHA
#FD94     A9 8A             LDA #8A
#FD96     8D 85 04          STA 0485
#FD99     A9 04             LDA #04
#FD9B     8D 86 04          STA 0486
#FD9E     A9 02             LDA #02
#FDA0     8D 81 04          STA 0481
#FDA3     20 9F 04          JSR 049F
#FDA6     68                PLA
#FDA7     8D 86 04          STA 0486
#FDAA     68                PLA
#FDAB     8D 85 04          STA 0485
#FDAE     68                PLA
#FDAF     8D 81 04          STA 0481
#FDB2     68                PLA
#FDB3     AA                TAX
#FDB4     68                PLA
#FDB5     40                RTI

#FDB6     30 FD             DTA #FD30       Not sure if these are used
#FDB8     00 F8             DTA #F800
#FDBA     68 FD             DTA #FD68
#FDBC     00 B7             DTA
#FDBE     65 10             DTA
#FDC0     40                RTI

#FDC1 to  FF FF FF FF FF    DTA             63 bytes spare
#FDFB     FF FF FF FF FF    DTA
#FE00     00 00 00 00 00    DTA             128 bytes spare
#FE7B     00 00 00 00 00    DTA
#FE80     FF FF FF FF FF    DTA             128 bytes spare
#FEFB     FF FF FF FF FF    DTA
#FF00     00 00 00 00 00    DTA             128 bytes spare
#FF7B     00 00 00 00 00    DTA
#FF80     FF FF FF FF FF    DTA             122 bytes spare
#FFF5     FF FF FF FF FF    DTA

#FFFA     30 FD             DTA #FD30       NMI Vector
#FFFC     00 F8             DTA #F800       Reset Vector
#FFFE     68 FD             DTA #FD68       IRQ Vector
```

Disassembly for ! on Page #4

Test for statement delimiter or number

```
#046E    C9 C8       CMP #C8     Test for ELSE
#0470    F0 0D       BEQ 047F    Exit if ELSE with Z set
#0472    C9 27       CMP #27     Test for ‘ (REM)
#0474    F0 09       BEQ 047F    Exit if remark with Z set
#0476    C9 3A       CMP #3A     Test for a number between 0 and 9
#0478    B0 05       BCS 047F    Exit with Carry flag set if not 0 to 9
#047A    E9 2F       SBC #2F
#047C    38          SEC
#047D    E9 D0       SBC #D0
#047F    60          RTS         Exit with Carry flag clear if 0 to 9 found
```

Data, flags and vector JMPs

```
#0480    84          DTA         Flag used to Page shadow ROM in / out
#0481    00          DTA         Flag used to Page shadow ROM in / out
#0482    00          DTA         Used to save value of Accumulator
#0483    72          DTA         Used to save status register
#0484    4C 12 E0    DTA         Vector for JMP #E012, but address can be changed
#0487    4C E3 04    DTA         Vector for JMP #04E3 (Page)
#048A    4C D3 04    DTA         Vector for JMP #04D3 (IRQ)
#048D    4C DB 04    DTA         Vector for JMP #04DB (NMI)
```

Process the ! command

```
#0490    A9 00       LDA #00
#0492    8D 81 04    STA 0481
#0495    A9 12       LDA #12     Set vector address in #0485 to start of DOS #E012
#0497    8D 85 04    STA 0485
#049A    A9 E0       LDA #E0
#049C    8D 86 04    STA 0486
#049F    08          PHP         Push status register onto stack
#04A0    78          SEI         Disable IRQ interrupts
#04A1    8D 82 04    STA 0482    Save A
#04A4    68          PLA         Get status register back into A
#04A5    8D 83 04    STA 0483    Save status register
#04A8    AD 80 04    LDA 0480    Get Paging flag
#04AB    48          PHA         Save it on stack  (PLA is at #04C6)
#04AC    AD 81 04    LDA 0481    Get Paging flag
#04AF    20 E3 04    JSR 04E3    Page the shadow ROM in
#04B2    AD 83 04    LDA 0483    Recover the saved values status register
#04B5    48          PHA         Save it on the stack
#04B6    AD 82 04    LDA 0482    Recover the saved value of A
#04B9    28          PLP         Re-set the status register to original value
#04BA    20 84 04    JSR 0484    Call to DOS
#04BD    08          PHP         Push status register onto stack
#04BE    78          SEI         Disable IRQ interrupts
#04BF    8D 82 04    STA 0482    Save A
#04C2    68          PLA         Get status register back into A
#04C3    8D 83 04    STA 0483    Save status register
#04C6    68          PLA         Get paging flag back
#04C7    20 E3 04    JSR 04E3    Page the shadow ROM out
```

```
#04CA   AD 83 04   LDA 0483     Recover the saved values status register
#04CD   48         PHA          Save it on the stack
#04CE   AD 82 04   LDA 0482     Recover the saved value of A
#04D1   28         PLP          Re-set the status register to original value
#04D2   60         RTS          Exit

#04D3   08         PHP
#04D4   BA         TSX
#04D5   FE 02 01   INC 0102,X
#04D8   4C 44 02   JMP 0244     Jump to IRQ

#04DB   08         PHP
#04DC   BA         TSX
#04DD   FE 02 01   INC 0102,X
#04E0   4C 47 02   JMP 0247     Jump to NMI
```

Page shadow ROM in / out

```
#04E3   78         SEI          Disable IRQ interrupts
#04E4   29 02      AND #02      Reset the Page flags
#04E6   8D 81 04   STA 0481     and page the shadow ROM in or out
#04E9   AD 80 04   LDA 0480     depending on the flags
#04EC   29 FD      AND #FD
#04EE   0D 81 04   ORA 0481
#04F1   8D 80 04   STA 0480
#04F4   8D 14 03   STA 0314
#04F7   60         RTS
```

Disassembly for Rip DOS V2.9

```
#E000    12 E0                              Entry address #E012
#E002    52 69 70           DTA Rip         Message Rip DOS V2.9
#E005    20 44 4F 53 20     DTA DOS
#E00A    56 32 2E 39 00     DTA V2.9
#E00F    00 00 00
```

Match primary commands

```
#E012    BA                 TSX             Entry point into DOS
#E013    8E 07 C0           STX C007
#E016    A9 00              LDA #00
#E018    8D FF 04           STA 04FF
#E01B    A6 EA              LDX EA
#E01D    D0 03              BNE E022
#E01F    8E FD 04           STX 04FD
#E022    A2 FF              LDX #FF         Try to match command word
#E024    8E 40 C1           STX C140        Command word counter
#E027    A0 FF              LDY #FF
#E029    EE 40 C1           INC C140
#E02C    E8                 INX             X counts along the lookup table
#E02D    C8                 INY             Y counts characters in the command word
#E02E    BD 40 FF           LDA FF40,X
#E031    C9 FF              CMP #FF
#E033    F0 28              BEQ E05D        Exit if end of table reached with no match found
#E035    48                 PHA
#E036    68                 PLA
#E037    F0 0C              BEQ E045        Accept command word when null reached
#E039    D1 E9              CMP (E9),Y
#E03B    F0 EF              BEQ E02C        Accept the current character and go back for another
#E03D    E8                 INX             otherwise move on to try the next word in the table
#E03E    BD 40 FF           LDA FF40,X
#E041    D0 FA              BNE E03D
#E043    F0 E2              BEQ E027
#E045    98                 TYA             Command word accepted
#E046    18                 CLC             Move text pointer on to the end of command word
#E047    65 E9              ADC E9           (Y = number of characters in word)
#E049    85 E9              STA E9
#E04B    90 02              BCC E04F
#E04D    E6 EA              INC EA
#E04F    AD 40 C1           LDA C140        Retrieve command word counter
#E052    0A                 ASL             and double it as index for address
#E053    AA                 TAX
#E054    BD C1 FF           LDA FFC1,X      Fetch the address (-1) for the command word
#E057    48                 PHA             Push it onto the stack
#E058    BD C0 FF           LDA FFC0,X
#E05B    48                 PHA
#E05C    60                 RTS             and JMP to it via the RTS
```

Match secondary commands

```
#E05D    A2 FF              LDX #FF         Same process as above,
#E05F    8E 40 C1           STX C140        but looking in a second command table
#E062    A0 FF              LDY #FF         held at #D000 onwards
#E064    EE 40 C1           INC C140
#E067    E8                 INX
```

```
#E068   C8              INY
#E069   BD 00 D0        LDA D000,X      Secondary command word table starts at #D000
#E06C   C9 FF           CMP #FF         Exit if end of table reached with no match
#E06E   F0 28           BEQ E098        (the start up routine puts #FF into #D000
#E070   48              PHA             so there will be an immediate exit
#E071   68              PLA             if extra commands have not been added)
#E072   F0 0C           BEQ E080        Accept command word when null reached
#E074   D1 E9           CMP (E9),Y
#E076   F0 EF           BEQ E067        Accept the current character and go back for another
#E078   E8              INX             otherwise move on to try the next word in the table
#E079   BD 00 D0        LDA D000,X
#E07C   D0 FA           BNE E078
#E07E   F0 E2           BEQ E062
#E080   98              TYA             Secondary command word accepted
#E081   18              CLC             Move text pointer on to the end of command word
#E082   65 E9           ADC E9
#E084   85 E9           STA E9
#E086   90 02           BCC E08A
#E088   E6 EA           INC EA
#E08A   AD 40 C1        LDA C140        Retrieve command word couture
#E08D   0A              ASL             and double it as index for address
#E08E   AA              TAX
#E08F   BD C1 D0        LDA D0C1,X      Fetch the address (-1) for the command word
#E092   48              PHA             Push it onto the stack
#E093   BD C0 D0        LDA D0C0,X
#E096   48              PHA
#E097   60              RTS             and JMP to it via the RTS

#E098   A2 09           LDX #09         End up here if no match found for command word
#E09A   BD C2 E0        LDA E0C2,X      and look for a file called ??????.COM to execute
#E09D   9D 2B C1        STA C12B,X      Set up space for the filename
#E0A0   CA              DEX             and .COM extension
#E0A1   D0 F7           BNE E09A
#E0A3   8E 2B C1        STX C12B
#E0A6   20 E8 00        JSR 00E8        Re-fetch the current character
#E0A9   9D 2C C1        STA C12C,X      Fill in the rest of the filename from text
#E0AC   E8              INX
#E0AD   E0 07           CPX #07
#E0AF   B0 0C           BCS E0BD        Branch to error if name too long
#E0B1   20 E2 00        JSR 00E2        Fetch the next character
#E0B4   D0 F3           BNE E0A9
#E0B6   A9 00           LDA #00
#E0B8   85 A9           STA A9
#E0BA   4C 06 E1        JMP E106        Jump to load the file ???????.COM
#E0BD   A2 05           LDX #05         Code for ‘Invalid filename’ error
#E0BF   4C 02 F7        JMP F702        Print error message and exit

#E0C2   20 20 20 20 20  DTA             Data for filename
#E0C7   20 20 43 4F 4D  DTA   COM
#E0CC   60              RTS
#E0CD   60              RTS
#E0CE   60              RTS
#E0CF   60              RTS
#E0D0   60              RTS
#E0D1   60              RTS

#E0D2   AD 4E C1        LDA C14E        Patch for m/c files which auto run after loading
#E0D5   C9 C0           CMP #C0
#E0D7   90 03           BCC E0DC        Leave DOS paged in if the T address
#E0D9   6C 4D C1        JMP (C14D)      is in shadow ROM area, >#C000
#E0DC   68              PLA             Otherwise, put the T address -1
#E0DD   8D 50 C1        STA C150        onto the stack, to jump there by RTS
```

```
#E0E0   68         PLA          once the normal ROM has been paged back in
#E0E1   8D 51 C1   STA C151
#E0E4   68         PLA
#E0E5   8D 52 C1   STA C152
#E0E8   AE 4E C1   LDX C14E
#E0EB   AC 4D C1   LDY C14D
#E0EE   D0 01      BNE E0F1
#E0F0   CA         DEX
#E0F1   88         DEY
#E0F2   8A         TXA
#E0F3   48         PHA
#E0F4   98         TYA
#E0F5   48         PHA
#E0F6   AD 52 C1   LDA C152
#E0F9   48         PHA
#E0FA   AD 51 C1   LDA C151
#E0FD   48         PHA
#E0FE   AD 50 C1   LDA C150
#E101   48         PHA
#E102   60         RTS
```

!LOAD

```
#E103   20 66 F4   JSR F466     Set up filename
#E106   A2 05      LDX #05      Initialise #C14B to #C150 to zero
#E108   A9 00      LDA #00      (flags and addresses)
#E10A   9D 4B C1   STA C14B,X
#E10D   CA         DEX
#E10E   10 FA      BPL E10A
#E110   20 E8 00   JSR 00E8     Re-fetch the current character
#E113   F0 4B      BEQ E160     Branch if end of line reached
#E115   20 B3 F5   JSR F5B3     Dispose of comma and get next character
#E118   C9 4A      CMP #4A      Check for J (Join)
#E11A   D0 12      BNE E12E     Branch if no join
#E11C   A5 9C      LDA 9C       Perform Join
#E11E   38         SEC
#E11F   E9 02      SBC #02
#E121   8D 4B C1   STA C14B
#E124   A5 9D      LDA 9D
#E126   E9 00      SBC #00
#E128   8D 4C C1   STA C14C
#E12B   4C 5B E1   JMP E15B
#E12E   C9 44      CMP #44      Check for D (Display start, end and transfer addresses)
#E130   D0 05      BNE E137
#E132   8D 4F C1   STA C14F     Set flag to display addresses
#E135   F0 24      BEQ E15B     and go on to get next character
#E137   C9 41      CMP #41      Check for A (new address specified for loading)
#E139   D0 14      BNE E14F
#E13B   8D 50 C1   STA C150     Set flag for new address for loading
#E13E   20 E2 00   JSR 00E2     Fetch the next character
#E141   20 A3 F7   JSR F7A3     Call routine in original ROM at
#E144   53 E8                   #E853 to get a 2 byte integer
#E146   8C 4B C1   STY C14B     Save the transfer address
#E149   8D 4C C1   STA C14C
#E14C   4C 10 E1   JMP E110     Go back for more parameters
#E14F   C9 4E      CMP #4E      Check for N (prevent AUTO run)
#E151   F0 05      BEQ E158
#E153   A2 02      LDX #02      Code for ‘Invalid command end’ error
#E155   4C 02 F7   JMP F702     Print error message and exit
#E158   8D 50 C1   STA C150     Set flag to inhibit AUTO run
```

```
#E15B     20 E2 00       JSR 00E2       Fetch the next character
#E15E     D0 B5          BNE E115       Process other parameters if present
#E160     AE 2B C1       LDX C12B       Get the drive number
#E163     20 E1 E5       JSR E5E1       Check for illegal drive number
#E166     20 C2 F5       JSR F5C2       Look for the file of name specified
#E169     D0 05          BNE E170
#E16B     A2 01          LDX #01        Code for ‘File not found’ error
#E16D     4C 02 F7       JMP F702       Print error message and exit
#E170     A2 01          LDX #01        Code for ‘Loading …’ message
#E172     20 39 F5       JSR F539       Print message and filename
#E175     AE 3F C1       LDX C13F
#E178     BD 32 C0       LDA C032,X
#E17B     29 01          AND #01
#E17D     F0 01          BEQ E180
#E17F     EA             NOP
#E180     BD 2F C0       LDA C02F,X     Get track data for first block to load
#E183     8D 01 C0       STA C001
#E186     BD 2E C0       LDA C02E,X     Get sector data for first block to load
#E189     8D 02 C0       STA C002
#E18C     20 10 F4       JSR F410       Read in some data from disk
#E18F     A2 00          LDX #00
#E191     A0 02          LDY #02
#E193     AD 4B C1       LDA C14B       Test if alternative load address specified
#E196     0D 4C C1       ORA C14C
#E199     D0 0C          BNE E1A7
#E19B     B9 25 C0       LDA C025,Y     Otherwise, set load address from header block
#E19E     8D 4B C1       STA C14B
#E1A1     B9 26 C0       LDA C026,Y
#E1A4     8D 4C C1       STA C14C
#E1A7     38             SEC
#E1A8     AD 4B C1       LDA C14B       Work out displacement for loading address
#E1AB     F9 25 C0       SBC C025,Y
#E1AE     99 25 C0       STA C025,Y
#E1B1     AD 4C C1       LDA C14C
#E1B4     F9 26 C0       SBC C026,Y
#E1B7     99 26 C0       STA C026,Y
#E1BA     18             CLC
#E1BB     B9 25 C0       LDA C025,Y     Work out new end address
#E1BE     79 27 C0       ADC C027,Y
#E1C1     99 27 C0       STA C027,Y
#E1C4     B9 26 C0       LDA C026,Y
#E1C7     79 28 C0       ADC C028,Y
#E1CA     99 28 C0       STA C028,Y
#E1CD     B9 29 C0       LDA C029,Y     Copy Transfer address
#E1D0     8D 4D C1       STA C14D
#E1D3     B9 2A C0       LDA C02A,Y
#E1D6     8D 4E C1       STA C14E
#E1D9     AD 4F C1       LDA C14F       See if address display was requested
#E1DC     F0 2F          BEQ E20D       Branch if not
#E1DE     20 0B F8       JSR F80B       Print out start, end and transfer addresses
#E1E1     20 98 F5       JSR F598       Print CR and LF for new line
#E1E4 -   EA             NOP
#E20C     EA             NOP
#E20D     AD 4B C1       LDA C14B       Put Start address into #0C and #0D
#E210     85 0C          STA 0C
#E212     AD 4C C1       LDA C14C
#E215     85 0D          STA 0D
#E217     A2 0A          LDX #0A
#E219     BD 23 C0       LDA C023,X     Number of bytes in sector to load
#E21C     F0 16          BEQ E234
#E21E     8D 41 C1       STA C141       Save number of bytes to load
#E221     A0 00          LDY #00
```

```
#E223     E8          INX
#E224     BD 23 C0    LDA C023,X     Load the bytes and copy them to destination address
#E227     91 0C       STA (0C),Y
#E229     E6 0C       INC 0C
#E22B     D0 02       BNE E22F
#E22D     E6 0D       INC 0D
#E22F     CE 41 C1    DEC C141
#E232     D0 EF       BNE E223       Continue till all bytes in block loaded
#E234     AD 23 C0    LDA C023       Set up track and sector details for next block
#E237     8D 01 C0    STA C001
#E23A     AD 24 C0    LDA C024
#E23D     F0 0A       BEQ E249       Branch if end reached
#E23F     8D 02 C0    STA C002
#E242     20 10 F4    JSR F410       Read in next block
#E245     A2 02       LDX #02
#E247     D0 D0       BNE E219       Go back for more blocks
#E249     AD 4E C1    LDA C14E       Get high byte of Transfer address
#E24C     F0 09       BEQ E257
#E24E     AD 50 C1    LDA C150
#E251     F0 01       BEQ E254
#E253     60          RTS
#E254     4C D2 E0    JMP E0D2
#E257     AD 4D C1    LDA C14D       Get Low byte of Transfer address
#E25A     F0 F7       BEQ E253       RTS if both zero – i.e. machine code, non AUTO
#E25C     C9 02       CMP #02
#E25E     90 0A       BCC E26A
#E260     AD 50 C1    LDA C150
#E263     D0 05       BNE E26A
#E265     20 A3 F7    JSR F7A3       Call routine in original ROM at
#E268     3A C7                      #C73A to set the start of BASIC
#E26A     20 A3 F7    JSR F7A3       Call routine in original ROM at
#E26D     5F C5                      #C55F to set line link pointers
#E26F     A5 91       LDA 91         Reset:
#E271     18          CLC            End of Basic
#E272     69 02       ADC #02        End of Variables
#E274     85 9C       STA 9C         End of Arrays
#E276     85 9E       STA 9E         Bottom of Strings
#E278     85 A0       STA A0
#E27A     A5 92       LDA 92
#E27C     69 00       ADC #00
#E27E     85 9D       STA 9D
#E280     85 9F       STA 9F
#E282     85 A1       STA A1
#E284     A5 A6       LDA A6
#E286     A6 A7       LDX A7
#E288     85 A2       STA A2
#E28A     86 A3       STX A3
#E28C     20 A3 F7    JSR F7A3       Call routine in original ROM at
#E28F     52 C9                      #C952 to perform a RESTORE
#E291     60          RTS            and exit
```

!SAVE

```
#E292     20 26 EB    JSR EB26       Close file for writing
#E295     20 66 F4    JSR F466       Set up filename
#E298     20 27 F5    JSR F527       Check that there are no wildcards
#E29B     AE 2B C1    LDX C12B       Get the drive number
#E29E     20 E1 E5    JSR E5E1       Check for valid drive number
#E2A1     20 C2 F5    JSR F5C2       Look for the specified filename
#E2A4     F0 05       BEQ E2AB       Branch if it does not exist
```

```
#E2A6   A2 09      LDX #09    Code for ‘File already exists’ error
#E2A8   4C 02 F7   JMP F702   Print error message and exit
#E2AB   A2 02      LDX #02    Message code for ‘Saving..’
#E2AD   20 60 F5   JSR F560   Do Saving …. Filename message
#E2B0   20 02 F6   JSR F602   Set up track / sector for first block
#E2B3   D0 05      BNE E2BA   Branch if there is enough disk space
#E2B5   A2 0A      LDX #0A    Code for ‘Insufficient disk space’ error
#E2B7   4C 02 F7   JMP F702   Print error message and exit
#E2BA   A2 FF      LDX #FF    Set flags etc
#E2BC   8E 25 C0   STX C025
#E2BF   E8         INX
#E2C0   8E 26 C0   STX C026
#E2C3   8E 3B C1   STX C13B   Write protect status
#E2C6   8E 2B C0   STX C02B   Transfer address / basic / code flag
#E2C9   8E 2C C0   STX C02C   Transfer address / basic / code flag
#E2CC   20 E8 00   JSR 00E8   Re-fetch the current character
#E2CF   F0 0D      BEQ E2DE   If end of statement, process as non-AUTO BASIC
#E2D1   20 B3 F5   JSR F5B3   Dispose of comma and get next character
#E2D4   C9 C7      CMP #C7    Test for keyword AUTO
#E2D6   D0 22      BNE E2FA   Branch if not AUTO
#E2D8   20 E2 00   JSR 00E2   Fetch the next character
#E2DB   A9 02      LDA #02
#E2DD   2C A9 01   BIT 01A9   #E2DE gives LDA #01
#E2E0   8D 2B C0   STA C02B   Save flag: 1=AUTO, 2=not AUTO
#E2E3   A5 9A      LDA 9A     Set up start addresses for BASIC
#E2E5   A6 9B      LDX 9B
#E2E7   8D 27 C0   STA C027
#E2EA   8E 28 C0   STX C028
#E2ED   A5 9C      LDA 9C     Set up end addresses for BASIC
#E2EF   A6 9D      LDX 9D
#E2F1   8D 29 C0   STA C029
#E2F4   8E 2A C0   STX C02A
#E2F7   4C 45 E3   JMP E345   Branch
#E2FA   C9 41      CMP #41    Test for A (start address for code)
#E2FC   D0 42      BNE E340   Branch if not the A option
#E2FE   20 E2 00   JSR 00E2   Fetch the next character
#E301   20 A3 F7   JSR F7A3   Call routine in original ROM at
#E304   53 E8                 #E853 to get a 2 byte integer
#E306   8C 27 C0   STY C027   Save the A address
#E309   8D 28 C0   STA C028
#E30C   20 B3 F5   JSR F5B3   Dispose of comma and get next character
#E30F   C9 45      CMP #45    Test for E (end address for code)
#E311   D0 2D      BNE E340   Branch if not the E option
#E313   20 E2 00   JSR 00E2   Fetch the next character
#E316   20 A3 F7   JSR F7A3   Call routine in original ROM at
#E319   53 E8                 #E853 to get a 2 byte integer
#E31B   8C 29 C0   STY C029   Save the E address
#E31E   8D 2A C0   STA C02A
#E321   20 E8 00   JSR 00E8   Re-fetch current character
#E324   F0 1F      BEQ E345   Branch if end of statement
#E326   20 B3 F5   JSR F5B3   Dispose of comma and get next character
#E329   C9 54      CMP #54    Test for T (transfer address for code)
#E32B   D0 13      BNE E340   Branch if not the E option
#E32D   20 E2 00   JSR 00E2   Fetch the next character
#E330   20 A3 F7   JSR F7A3   Call routine in original ROM at
#E333   53 E8                 #E853 to get a 2 byte integer
#E335   8C 2B C0   STY C02B   Save the T address
#E338   8D 2C C0   STA C02C
#E33B   20 E8 00   JSR 00E8   Re-fetch the current character
#E33E   F0 05      BEQ E345   Branch if end of statement
#E340   A2 02      LDX #02    Code for ‘Invalid command end’ error
#E342   4C 02 F7   JMP F702   Print error message and exit
```

```
#E345   AD 27 C0    LDA C027      Set up start address for save
#E348   AE 28 C0    LDX C028
#E34B   85 0C       STA 0C
#E34D   86 0D       STX 0D
#E34F   AD 29 C0    LDA C029      Set up end address for save
#E352   AE 2A C0    LDX C02A
#E355   8D 4B C1    STA C14B
#E358   8E 4C C1    STX C14C
#E35B   AD 02 C0    LDA C002      Set up sector
#E35E   8D 37 C1    STA C137
#E361   AD 01 C0    LDA C001      Set up track
#E364   8D 38 C1    STA C138
#E367   A2 0A       LDX #0A
#E369   A0 00       LDY #00
#E36B   8E 40 C1    STX C140
#E36E   E8          INX
#E36F   B1 0C       LDA (0C),Y    Copy first blocks worth of file into
#E371   9D 23 C0    STA C023,X    C023 onwards, after the header data
#E374   AD 4B C1    LDA C14B      Test for last data byte
#E377   C5 0C       CMP 0C
#E379   D0 07       BNE E382
#E37B   AD 4C C1    LDA C14C
#E37E   C5 0D       CMP 0D
#E380   F0 18       BEQ E39A      Move to last block if end of file
#E382   E6 0C       INC 0C        Increment pointers
#E384   D0 02       BNE E388
#E386   E6 0D       INC 0D
#E388   E8          INX
#E389   D0 E4       BNE E36F      Continue until page is full
#E38B   20 58 F4    JSR F458      Write the block out to disk
#E38E   20 69 F6    JSR F669      Set up the next block on the disk
#E391   D0 03       BNE E396      Test if disk full
#E393   4C B5 E2    JMP E2B5      Jump to Insufficient disk space error
#E396   A2 02       LDX #02
#E398   D0 CF       BNE E369      Go back and do next block
#E39A   A9 00       LDA #00       Set marker to indicate last block
#E39C   8D 23 C0    STA C023
#E39F   8D 24 C0    STA C024
#E3A2   E8          INX
#E3A3   F0 0A       BEQ E3AF
#E3A5   8A          TXA
#E3A6   A8          TAY
#E3A7   A9 00       LDA #00
#E3A9   99 23 C0    STA C023,Y    Fill the rest of the block with null characters
#E3AC   C8          INY
#E3AD   D0 FA       BNE E3A9
#E3AF   20 58 F4    JSR F458      Write the block out to disk
#E3B2   20 B1 F6    JSR F6B1      Update the directory
#E3B5   4C D1 F6    JMP F6D1      Update system tack and exit
```

!DIR

```
#E3B8   20 66 F4    JSR F466      Set up filename
#E3BB   AE 2B C1    LDX C12B      Get the drive number
#E3BE   20 E1 E5    JSR E5E1      Check for illegal drive number
#E3C1   20 E2 F6    JSR F6E2      Load system track
#E3C4   A2 08       LDX #08
#E3C6   BD 3B C0    LDA C03B,X    Set up disk name in #C17B on
#E3C9   9D 7B C1    STA C17B,X
#E3CC   CA          DEX
```

```
#E3CD     10 F7          BPL E3C6
#E3CF     20 98 F5       JSR F598         Print CR and LF for new line
#E3D2     A2 0F          LDX #0F          Code for ‘Directory of Drive’
#E3D4     20 29 F7       JSR F729         Print system message
#E3D7     AD 2B C1       LDA C12B         Get the drive number
#E3DA     09 30          ORA #30
#E3DC     20 9F F5       JSR F59F         Print the character
#E3DF     A9 2D          LDA #2D          ‘-‘ character
#E3E1     20 9F F5       JSR F59F         Print the character
#E3E4     A2 00          LDX #00
#E3E6     BD 7B C1       LDA C17B,X       Get character for the disk name
#E3E9     20 9F F5       JSR F59F         Print the character
#E3EC     E8             INX
#E3ED     E0 09          CPX #09
#E3EF     D0 F5          BNE E3E6
#E3F1     20 98 F5       JSR F598         Print CR and LF for new line
#E3F4     20 98 F5       JSR F598         Print CR and LF for new line
#E3F7     A9 00          LDA #00
#E3F9     8D 40 C1       STA C140
#E3FC     20 C5 F5       JSR F5C5         Fetch next matching filename
#E3FF     F0 54          BEQ E455         Exit when all done
#E401     BD 32 C0       LDA C032,X       Ignore if the file is ‘invisible’
#E404     29 40          AND #40
#E406     D0 48          BNE E450
#E408     20 AE F5       JSR F5AE         Print a space
#E40B     20 46 F5       JSR F546         Print out the filename
#E40E     AC 3F C1       LDY C13F
#E411     B9 2D C0       LDA C02D,Y       Fetch number of sectors taken by the file
#E414     BE 2C C0       LDX C02C,Y
#E417     20 57 F7       JSR F757         Print out 2 byte integer in decimal
#E41A     AC 3F C1       LDY C13F
#E41D     B9 32 C0       LDA C032,Y       Fetch the write protect status
#E420     10 03          BPL E425
#E422     A9 50          LDA #50          P character
#E424     2C A9 20       BIT 20A9         #E425 = LDA #A9 (space character)
#E427     20 9F F5       JSR F59F         Print the character
#E42A     EE 40 C1       INC C140         Increment counter for files
#E42D     AD 40 C1       LDA C140         Test if odd or even
#E430     29 01          AND #01          (for 2 per line)
#E432     F0 05          BEQ E439         Branch if even
#E434     20 AE F5       JSR F5AE         Print a space
#E437     90 17          BCC E450         Branch always
#E439     20 98 F5       JSR F598         Print CR and LF for new line
#E43C     20 A3 F7       JSR F7A3         Call routine in original ROM at
#E43F     3B 02                           #023B – jump to the GET KEY routine
#E441     10 0D          BPL E450         Branch if no key pressed
#E443     C9 20          CMP #20          Test for space bar (scroll)
#E445     D0 05          BNE E44C         branch (or await next key press)
#E447     20 A3 F7       JSR F7A3         Call routine in original ROM at
#E44A     E8 C5                           #C5E8 to read key from keyboard
#E44C     C9 1B          CMP #1B          Test for Escape key
#E44E     F0 2B          BEQ E47B         Exit if ESC pressed
#E450     20 F0 F5       JSR F5F0         Look for next  matching file
#E453     D0 AC          BNE E401         Branch back unless end reached
#E455     AD 40 C1       LDA C140
#E458     29 01          AND #01
#E45A     F0 03          BEQ E45F
#E45C     20 98 F5       JSR F598         Print CR and LF for new line
#E45F     20 98 F5       JSR F598         Print CR and LF for new line
#E462     AE 40 C1       LDX C140         Get number of files
#E465     20 55 F7       JSR F755         Print out number of files
#E468     A2 10          LDX #10          Code for ‘Files’
```

```
#E46A    20 29 F7     JSR F729      Print system message
#E46D    AE 27 C1     LDX C127      Get number of blocks free
#E470    AD 28 C1     LDA C128
#E473    20 57 F7     JSR F757      Print out 2 byte integer in decimal
#E476    A2 11        LDX #11       Code for ‘Blocks free’
#E478    20 29 F7     JSR F729      Print system message
#E47B    4C 98 F5     JMP F598      Print new line and exit
```

!DEL

```
#E47E    20 26 EB     JSR EB26      Close file for writing
#E481    20 66 F4     JSR F466      Set up filename
#E484    AE 2B C1     LDX C12B      Get the drive number
#E487    20 E1 E5     JSR E5E1      Check for illegal drive number
#E48A    20 C2 F5     JSR F5C2      Look for file
#E48D    D0 05        BNE E494      Branch if found
#E48F    A2 01        LDX #01       Code for ‘File not found’ error
#E491    4C 02 F7     JMP F702      Print error message and exit
#E494    A2 08        LDX #08
#E496    BD 2C C1     LDA C12C,X
#E499    C9 3F        CMP #3F       Look for wildcards in the filename
#E49B    F0 05        BEQ E4A2      Branch if found
#E49D    CA           DEX
#E49E    10 F6        BPL E496      Look for more wildcards
#E4A0    30 20        BMI E4C2      Branch if no wildcards
#E4A2    20 98 F5     JSR F598      Print CR and LF for new line
#E4A5    20 46 F5     JSR F546      Print out the filename
#E4A8    A2 03        LDX #03       Code for (Y/N)
#E4AA    20 29 F7     JSR F729      Print system message
#E4AD    20 A3 F7     JSR F7A3      Call routine in original ROM at
#E4B0    E8 C5                      #C5E8 to read key from keyboard
#E4B2    C9 59        CMP #59       Test for Y = Yes
#E4B4    08           PHP
#E4B5    F0 02        BEQ E4B7
#E4B7    A9 4E        LDA #4E       if not Y then load N for No
#E4B9    20 9F F5     JSR F59F      Print the character
#E4BC    28           PLP
#E4BD    F0 03        BEQ E4C2      Branch to delete file
#E4BF    4C DE E4     JMP E4DE      otherwise jump to look for next file
#E4C2    AC 3F C1     LDY C13F
#E4C5    B9 32 C0     LDA C032,Y
#E4C8    10 08        BPL E4D2
#E4CA    A2 12        LDX #12       Code for ‘Write protected’
#E4CC    20 29 F7     JSR F729      Print system message
#E4CF    4C DE E4     JMP E4DE      and jump to look for next file
#E4D2    20 E9 E4     JSR E4E9      Delete file
#E4D5    AD 3D C1     LDA C13D      Re-load directory parameters
#E4D8    AE 3E C1     LDX C13E
#E4DB    20 0A F4     JSR F40A      Read from disk – block specified by track A / sector X
#E4DE    20 F0 F5     JSR F5F0      Carry on looking for matching files
#E4E1    D0 BF        BNE E4A2      Branch if match
#E4E3    20 98 F5     JSR F598      Print CR and LF for new line
#E4E6    4C D1 F6     JMP F6D1      Update system tack and exit
#E4E9    A2 00        LDX #00       Delete the file
#E4EB    B9 2C C0     LDA C02C,Y    Copy file information
#E4EE    9D 35 C1     STA C135,X
#E4F1    C8           INY
#E4F2    E8           INX
#E4F3    E0 06        CPX #06
#E4F5    D0 F4        BNE E4EB
```

```
#E4F7    CE 25 C0      DEC C025      Decrement number of files in directory
#E4FA    AE 3F C1      LDX C13F      Get displacement along directory
#E4FD    A0 10         LDY #10
#E4FF    A9 00         LDA #00
#E501    9D 23 C0      STA C023,X    Fill directory entry with nulls
#E504    E8            INX
#E505    88            DEY
#E506    D0 F9         BNE E501
#E508    20 06 F4      JSR F406      Re-write directory to disk
#E50B    AD 01 C0      LDA C001      Track counter
#E50E    8D 3D C1      STA C13D
#E511    AD 02 C0      LDA C002      Sector counter
#E514    8D 3E C1      STA C13E
#E517    AE 39 C1      LDX C139      Read in sector of final block
#E51A    AD 3A C1      LDA C13A
#E51D    20 0A F4      JSR F40A      Read from disk – block specified by track A / sector X
#E520    AD 23 C1      LDA C123      Copy ‘next available sector’
#E523    8D 24 C0      STA C024      info to final block
#E526    AD 24 C1      LDA C124
#E529    8D 23 C0      STA C023
#E52C    20 06 F4      JSR F406      Write sector out again
#E52F    AD 37 C1      LDA C137
#E532    8D 23 C1      STA C123
#E535    AD 38 C1      LDA C138
#E538    8D 24 C1      STA C124
#E53B    18            CLC
#E53C    AD 27 C1      LDA C127      Update the number of blocks free
#E53F    6D 35 C1      ADC C135      by adding the number of blocks
#E542    8D 27 C1      STA C127      taken by this file
#E545    AD 28 C1      LDA C128
#E548    6D 36 C1      ADC C136
#E54B    8D 28 C1      STA C128
#E54E    38            SEC
#E54F    AD 29 C1      LDA C129      Update the number of blocks used
#E552    ED 35 C1      SBC C135      ready for updating the system track
#E555    8D 29 C1      STA C129      on completion
#E558    AD 2A C1      LDA C12A
#E55B    ED 36 C1      SBC C136
#E55E    8D 2A C1      STA C12A
#E561    60            RTS
```

!REN

```
#E562    20 66 F4      JSR F466      Set up first filename
#E565    AE 2B C1      LDX C12B      Get the drive number
#E568    20 E1 E5      JSR E5E1      Check for illegal drive number
#E56B    20 E8 00      JSR 00E8      Re-fetch the current character
#E56E    C9 C3         CMP #C3       Look for ‘TO’ token
#E570    D0 4D         BNE E5BF      Branch to error if not found
#E572    20 C2 F5      JSR F5C2      Look for specified file on disk
#E575    F0 4B         BEQ E5C2      Branch if file not found
#E577    A2 09         LDX #09       Copy first file name
#E579    BD 2B C1      LDA C12B,X
#E57C    9D 7A C1      STA C17A,X
#E57F    CA            DEX
#E580    10 F7         BPL E579
#E582    20 E2 00      JSR 00E2      Fetch the next character
#E585    20 66 F4      JSR F466      Set up second filename
#E588    20 27 F5      JSR F527      Check for wildcards – not allowed
#E58B    AD 2B C1      LDA C12B      Check that the 2 drive numbers are the same
```

```
#E58E    CD 7A C1      CMP C17A
#E591    D0 35         BNE E5C8       Branch to error if they are not
#E593    20 C2 F5      JSR F5C2       Look for second file on disk
#E596    D0 2D         BNE E5C5       Branch to error if it already exists
#E598    A2 09         LDX #09        Swap over the 2 file names in memory
#E59A    BD 2B C1      LDA C12B,X
#E59D    BC 7A C1      LDY C17A,X
#E5A0    9D 7A C1      STA C17A,X
#E5A3    98            TYA
#E5A4    9D 2B C1      STA C12B,X
#E5A7    CA            DEX
#E5A8    10 F0         BPL E59A
#E5AA    20 C2 F5      JSR F5C2       Find the first filename on the disk
#E5AD    A0 00         LDY #00        Copy down the second name
#E5AF    B9 7B C1      LDA C17B,Y
#E5B2    9D 23 C0      STA C023,X
#E5B5    E8            INX
#E5B6    C8            INY
#E5B7    C0 09         CPY #09
#E5B9    D0 F4         BNE E5AF
#E5BB    20 06 F4      JSR F406       Write out the new filename
#E5BE    60            RTS            and exit
#E5BF    A2 0F         LDX #0F        Code for ‘Missing to’ error
#E5C1    2C A2 01      BIT 01A2       #E5C2 LDX #01 – code for ‘File not found’ error
#E5C4    2C A2 09      BIT 09A2       #E5C5 LDX #09 – code for ‘File already exists’ error
#E5C7    2C A2 10      BIT 10A2       #E5C8 LDX #10 – ‘Re-named file not on same disk’ error
#E5CA    4C 02 F7      JMP F702       Print error message and exit
```

!DRV

```
#E5CD    A2 00         LDX #00        Default drive number
#E5CF    20 E8 00      JSR 00E8       Re-fetch the current character
#E5D2    B0 06         BCS E5DA       Branch if not 0-9 and use default
#E5D4    E9 2F         SBC #2F        Convert ASCII character to number
#E5D6    AA            TAX
#E5D7    20 E2 00      JSR 00E2       Fetch the next character
#E5DA    20 E1 E5      JSR E5E1       Check for illegal drive number
#E5DD    8E 0C C0      STX C00C       Store default drive number and
#E5E0    60            RTS            exit
#E5E1    E0 04         CPX #04        Error if  drive no > 3
#E5E3    B0 09         BCS E5EE
#E5E5    BD 13 C0      LDA C013,X     or if drive not SET
#E5E8    F0 04         BEQ E5EE
#E5EA    8E 00 C0      STX C000       Save drive number and exit
#E5ED    60            RTS
#E5EE    A2 04         LDX #04        Code for ‘Bad drive number’ error
#E5F0    4C 02 F7      JMP F702       Print error message and exit
```

!BACKUP

```
#E5F3    AD 0C C0      LDA C00C       Set the default drive number
#E5F6    8D 7A C1      STA C17A
#E5F9    8D 7B C1      STA C17B
#E5FC    20 E8 00      JSR 00E8       Re-fetch the current character
#E5FF    F0 27         BEQ E628       Branch to backup if end of statement
#E601    20 A3 F7      JSR F7A3       Call routine in original ROM at
#E604    C8 D8                        #D8C8 to get a single byte expression
#E606    20 E1 E5      JSR E5E1       Check for illegal drive number
#E609    8E 7A C1      STX C17A
```

```
#E60C    20 E8 00     JSR 00E8      Re-fetch the current character
#E60F    F0 17        BEQ E628      Branch to backup if end of statement
#E611    C9 C3        CMP #C3       Look for ‘TO’ token
#E613    F0 03        BEQ E618      Branch if found
#E615    4C BF E5     JMP E5BF      otherwise jump to ‘Missing to’ error
#E618    20 E2 00     JSR 00E2      Fetch the next character
#E61B    F0 0B        BEQ E628      Branch to backup if end of statement
#E61D    20 A3 F7     JSR F7A3      Call routine in original ROM at
#E620    C8 D8                      #D8C8 to get a single byte expression
#E622    20 E1 E5     JSR E5E1      Check for illegal drive number
#E625    8E 7B C1     STX C17B      and save as target
#E628    AE 7A C1     LDX C17A      Save number of tracks
#E62B    BD 17 C0     LDA C017,X    on side 2 of
#E62E    8D 82 C1     STA C182      source drive
#E631    BD 13 C0     LDA C013,X    Save number of tracks
#E634    8D 81 C1     STA C181      on side 1 of source drive
#E637    EC 7B C1     CPX C17B      Branch if same drive number
#E63A    F0 25        BEQ E661
#E63C    AC 7B C1     LDY C17B      Test for incompatible drives, if
#E63F    D9 13 C0     CMP C013,Y    number of tracks on source does not equal
#E642    F0 05        BEQ E649      number of tracks on target
#E644    A2 1B        LDX #1B       Code for ‘Incompatible drives’ error
#E646    4C 02 F7     JMP F702      Print error message and exit
#E649    A2 07        LDX #07       Code for ‘Load disks for backup from #’
#E64B    20 29 F7     JSR F729      Print system message
#E64E    AD 7A C1     LDA C17A      Get source drive number
#E651    09 30        ORA #30
#E653    20 9F F5     JSR F59F      Print the character
#E656    A2 08        LDX #08       Code for ‘to #’
#E658    20 29 F7     JSR F729      Print system message
#E65B    AD 7B C1     LDA C17B      Get target drive number
#E65E    4C 69 E6     JMP E669      Jump to print it
#E661    A2 04        LDX #04       Code for ‘Load source disk on drive’
#E663    20 29 F7     JSR F729      Print system message
#E666    AD 7A C1     LDA C17A      Get source drive number
#E669    09 30        ORA #30
#E66B    20 9F F5     JSR F59F      Print the character
#E66E    A2 09        LDX #09       Code for ‘and press Return’
#E670    20 29 F7     JSR F729      Print system message
#E673    20 86 F5     JSR F586      Wait for Return
#E676    A9 00        LDA #00       Set to Track 0
#E678    8D 7C C1     STA C17C
#E67B    A9 01        LDA #01       Set to Sector 1
#E67D    8D 7D C1     STA C17D
#E680    AD 7A C1     LDA C17A      Set drive number for source
#E683    8D 00 C0     STA C000
#E686    A0 80        LDY #80       Read in a block of data
#E688    20 E5 E6     JSR E6E5
#E68B    D0 46        BNE E6D3      Branch if disk error
#E68D    AD 7B C1     LDA C17B      Set up target drive number
#E690    8D 00 C0     STA C000
#E693    CD 7A C1     CMP C17A      Test if source and target are the same
#E696    D0 08        BNE E6A0      Skip prompt if they are not the same
#E698    A2 06        LDX #06       Code for ‘Load target disk and press Return’
#E69A    20 29 F7     JSR F729      Print system message
#E69D    20 86 F5     JSR F586      Wait for Return
#E6A0    A0 A0        LDY #A0       Write block of data out to target drive
#E6A2    20 E5 E6     JSR E6E5
#E6A5    D0 2F        BNE E6D6      Branch if drive error
#E6A7    AD 01 C0     LDA C001      Re-set Track number
#E6AA    8D 7C C1     STA C17C
#E6AD    AD 02 C0     LDA C002      Re-set Sector number
```

```
#E6B0   8D 7D C1    STA C17D
#E6B3   AD 7A C1    LDA C17A    Re-set source drive number
#E6B6   8D 00 C0    STA C000
#E6B9   CD 7B C1    CMP C17B    Test if source and target are the same
#E6BC   D0 08       BNE E6C6    Skip prompt if they are not the same
#E6BE   A2 05       LDX #05     Code for ‘Load source disk and press Return’
#E6C0   20 29 F7    JSR F729    Print system message
#E6C3   20 86 F5    JSR F586    Wait for Return
#E6C6   AD 01 C0    LDA C001    Get track number
#E6C9   CD 81 C1    CMP C181    Compare with the number of tracks on source disk
#E6CC   D0 B2       BNE E680    Go back for more if not finished
#E6CE   A2 0A       LDX #0A     Message code for ‘Backup complete.’
#E6D0   4C 29 F7    JMP F729    Print system message and exit
#E6D3   A2 0B       LDX #0B     Code for ‘Disk error’ message
#E6D5   2C A2 0C    BIT 0CA2    #E6D6 LDX #0C – ‘Drive’
#E6D8   20 29 F7    JSR F729    Print system message
#E6DB   A2 0D       LDX #0D     Code for ‘Track’
#E6DD   20 29 F7    JSR F729    Print system message
#E6E0   20 A3 F7    JSR F7A3    Call routine in original ROM at
#E6E3   03 C0                   #C003 to RESTART BASIC
#E6E5   AD 7C C1    LDA C17C    Save track and sector
#E6E8   AE 7D C1    LDX C17D    details for next output
#E6EB   8D 01 C0    STA C001
#E6EE   8E 02 C0    STX C002
#E6F1   8C 40 C1    STY C140    Set read / write marker
#E6F4   A9 0A       LDA #0A
#E6F6   8D 80 C1    STA C180
#E6F9   A9 00       LDA #00     Set LOAD address to #1000
#E6FB   8D 03 C0    STA C003
#E6FE   A9 10       LDA #10
#E700   8D 04 C0    STA C004
#E703   AC 40 C1    LDY C140    Get read / write marker
#E706   20 12 F4    JSR F412    Read / write a sector
#E709   AD 02 C0    LDA C002
#E70C   18          CLC
#E70D   69 02       ADC #02
#E70F   C9 11       CMP #11
#E711   90 04       BCC E717
#E713   E9 10       SBC #10
#E715   49 03       EOR #03
#E717   8D 02 C0    STA C002
#E71A   EE 04 C0    INC C004
#E71D   AD 04 C0    LDA C004
#E720   29 0F       AND #0F
#E722   D0 DF       BNE E703
#E724   38          SEC
#E725   AD 82 C1    LDA C182
#E728   F0 02       BEQ E72C
#E72A   A9 7F       LDA #7F
#E72C   6D 01 C0    ADC C001
#E72F   8D 01 C0    STA C001
#E732   CE 80 C1    DEC C180
#E735   D0 CC       BNE E703
#E737   60          RTS
```

!COPY

```
#E738   20 26 EB    JSR EB26    Perform a CLOSE
#E73B   AD 0C C0    LDA C00C    Set up default drive number
#E73E   8D 7A C1    STA C17A
```

```
#E741     8D 84 C1      STA C184
#E744     20 66 F4      JSR F466      Set up filename
#E747     AE 2B C1      LDX C12B      Get the drive number
#E74A     20 E1 E5      JSR E5E1      Check for illegal drive number
#E74D     A2 09         LDX #09       Copy first filename and drive number
#E74F     BD 2B C1      LDA C12B,X
#E752     9D 7A C1      STA C17A,X
#E755     CA            DEX
#E756     10 F7         BPL E74F
#E758     20 E8 00      JSR 00E8      Re-fetch the current character
#E75B     C9 C3         CMP #C3       Test for TO token
#E75D     F0 03         BEQ E762      Branch if found
#E75F     4C BF E5      JMP E5BF      Missing TO error and exit
#E762     20 E2 00      JSR 00E2      Fetch the next character
#E765     20 66 F4      JSR F466      Set up filename
#E768     AE 2B C1      LDX C12B      Get the drive number
#E76B     20 E1 E5      JSR E5E1      Check for illegal drive number
#E76E     A2 09         LDX #09       Copy the second filename and drive number
#E770     BD 2B C1      LDA C12B,X
#E773     9D 84 C1      STA C184,X
#E776     CA            DEX
#E777     10 F7         BPL E770
#E779     A9 80         LDA #80
#E77B     8D 8F C1      STA C18F      Set Protect flag
#E77E     8D 90 C1      STA C190      Set Over write flag
#E781     8D 91 C1      STA C191      Set for single drive option
#E784     20 E8 00      JSR 00E8      Re-fetch the current character
#E787     F0 35         BEQ E7BE      Branch if end of statement
#E789     20 B3 F5      JSR F5B3      Dispose of comma and get next character
#E78C     C9 50         CMP #50       Test for P option (Protect)
#E78E     F0 04         BEQ E794
#E790     C9 4E         CMP #4E       Test for N option (Non protect)
#E792     D0 05         BNE E799
#E794     8D 8F C1      STA C18F      Set the P/N flag
#E797     F0 20         BEQ E7B9      Branch always to get next character
#E799     C9 4F         CMP #4F       Test for O (over write option)
#E79B     D0 05         BNE E7A2      Branch if not found
#E79D     8D 90 C1      STA C190      Set the O flag
#E7A0     F0 17         BEQ E7B9      Branch always to get next character
#E7A2     C9 43         CMP #43       Test for C (single drive)
#E7A4     D0 0E         BNE E7B4      Branch if not C
#E7A6     8D 91 C1      STA C191      Set single drive flag
#E7A9     AD 7A C1      LDA C17A      Check if the drive numbers are the same
#E7AC     CD 84 C1      CMP C184
#E7AF     F0 08         BEQ E7B9      Branch if they are the same
#E7B1     A2 12         LDX #12       Code for ‘Target drive not source drive’
#E7B3     2C A2 07      BIT 07A2      #E7B4 LDX #07 – ‘Illegal attribute’
#E7B6     4C 02 F7      JMP F702      Print error message and exit
#E7B9     20 E2 00      JSR 00E2      Fetch the next character
#E7BC     D0 CB         BNE E789      Go back to test next character if not end of statement
#E7BE     A2 08         LDX #08       Copy down first filename
#E7C0     BD 7B C1      LDA C17B,X
#E7C3     9D 2C C1      STA C12C,X
#E7C6     CA            DEX
#E7C7     10 F7         BPL E7C0
#E7C9     AD 7A C1      LDA C17A      Get first drive number
#E7CC     8D 00 C0      STA C000      and save it
#E7CF     AD 91 C1      LDA C191      Branch if not the single drive option
#E7D2     30 15         BMI E7E9
#E7D4     A2 04         LDX #04       Code for ‘Load source disk on drive’
#E7D6     20 29 F7      JSR F729      Print system message
#E7D9     AD 00 C0      LDA C000      Get the drive number
```

```
#E7DC    09 30       ORA #30
#E7DE    20 9F F5    JSR F59F      Print the character
#E7E1    A2 09       LDX #09       Code for ‘and press Return’
#E7E3    20 29 F7    JSR F729      Print system message
#E7E6    20 86 F5    JSR F586      Wait for the Return
#E7E9    20 E2 F6    JSR F6E2      Load system track
#E7EC    20 C5 F5    JSR F5C5      Get filename and locate the file in directory
#E7EF    D0 03       BNE E7F4      Branch if found
#E7F1    4C C2 E5    JMP E5C2      Jump to ‘File not found’ error and exit
#E7F4    20 15 F5    JSR F515      copy filename
#E7F7    AD 01 C0    LDA C001      Copy Track and Sector numbers
#E7FA    AE 02 C0    LDX C002
#E7FD    8D B3 C1    STA C1B3
#E800    8E B4 C1    STX C1B4
#E803    AD 3F C1    LDA C13F      and the displacement along the directory
#E806    8D B5 C1    STA C1B5
#E809    A9 00       LDA #00       Set #C003/4 and #0C/D to #0600
#E80B    A2 06       LDX #06       (address for loading the file)
#E80D    8D 03 C0    STA C003
#E810    85 0C       STA 0C
#E812    8E 04 C0    STX C004
#E815    86 0D       STX 0D
#E817    A9 00       LDA #00       Set counter for sectors
#E819    8D 41 C1    STA C141
#E81C    AE 37 C1    LDX C137      Get track and sector details
#E81F    AD 38 C1    LDA C138      for first block to copy
#E822    20 0A F4    JSR F40A      Read from disk – block specified by track A / sector X
#E825    EE 41 C1    INC C141      Increment Sector counter
#E828    A0 01       LDY #01       Get next track / sector numbers
#E82A    B1 0C       LDA (0C),Y    into A and X
#E82C    AA          TAX
#E82D    88          DEY
#E82E    B1 0C       LDA (0C),Y
#E830    E0 00       CPX #00       Test for end
#E832    F0 0B       BEQ E83F      Branch if no more to do
#E834    EE 04 C0    INC C004      Increment page pointers for next block
#E837    E6 0D       INC 0D
#E839    A4 0D       LDY 0D
#E83B    C0 B0       CPY #B0
#E83D    90 E3       BCC E822      Go back to read in next sector
#E83F    8D BA C1    STA C1BA      Track for next block
#E842    8E BB C1    STX C1BB      Sector for next block
#E845    AD 91 C1    LDA C191      Test for single drive option
#E848    30 08       BMI E852      Skip message if 2 drives
#E84A    A2 06       LDX #06       Code for ‘Load target disk and press Return’
#E84C    20 29 F7    JSR F729      Print system message
#E84F    20 86 F5    JSR F586      Wait for Return
#E852    AD 84 C1    LDA C184      Set target drive number
#E855    8D 00 C0    STA C000
#E858    20 E2 F6    JSR F6E2      Load system track
#E85B    AD 85 C1    LDA C185
#E85E    A2 08       LDX #08       Counter for characters in filename
#E860    BD 85 C1    LDA C185,X    Get character from second file name
#E863    C9 3F       CMP #3F       Test for ? character
#E865    F0 03       BEQ E86A      and ignore any present (leaves a space instead)
#E867    9D 2C C1    STA C12C,X    Save the character
#E86A    CA          DEX           Decrement counter
#E86B    10 F3       BPL E860      Go back for the rest if not finished
#E86D    20 C2 F5    JSR F5C2      Look for filename in directory
#E870    F0 13       BEQ E885      Branch to save if not found
#E872    AD 90 C1    LDA C190      Test for overwrite
#E875    10 03       BPL E87A      Carry on if over write requested
```

```
#E877   4C ED E8    JMP E8ED      Jump to ‘Already exists’ error and exit
#E87A   AC 3F C1    LDY C13F      Check if write protected
#E87D   B9 32 C0    LDA C032,Y
#E880   30 65       BMI E8E7      Branch to ‘Write protected’
#E882   20 E9 E4    JSR E4E9      Delete the file
#E885   20 02 F6    JSR F602      Set up first sector for Save
#E888   D0 03       BNE E88D      Branch if OK, otherwise
#E88A   4C B5 E2    JMP E2B5      Jump to ‘Insufficient disk space’ error
#E88D   A9 00       LDA #00       Re-set the address for the loaded file
#E88F   A2 06       LDX #06
#E891   85 0C       STA 0C
#E893   86 0D       STX 0D
#E895   A0 01       LDY #01
#E897   B1 0C       LDA (0C),Y    Load a byte
#E899   F0 08       BEQ E8A3      Branch if finished
#E89B   B9 23 C0    LDA C023,Y    Switch top next available sector
#E89E   91 0C       STA (0C),Y
#E8A0   88          DEY
#E8A1   10 F8       BPL E89B
#E8A3   A5 0C       LDA 0C        Set up address for writing file
#E8A5   8D 03 C0    STA C003
#E8A8   A5 0D       LDA 0D
#E8AA   8D 04 C0    STA C004
#E8AD   20 06 F4    JSR F406      Write a sector out to disk
#E8B0   A9 23       LDA #23       Set up load address for next sector
#E8B2   A2 C0       LDX #C0
#E8B4   8D 03 C0    STA C003
#E8B7   8E 04 C0    STX C004
#E8BA   CE 41 C1    DEC C141      Decrement sector counter
#E8BD   F0 0A       BEQ E8C9      End if no more sectors to do
#E8BF   E6 0D       INC 0D
#E8C1   20 69 F6    JSR F669      Set up next sector
#E8C4   D0 CF       BNE E895      Go back and do it unless
#E8C6   4C B5 E2    JMP E2B5      Insufficient disk space error
#E8C9   A9 00       LDA #00       Default for N status
#E8CB   AC 8F C1    LDY C18F      Load P/N status
#E8CE   30 09       BMI E8D9      Branch if no change specified
#E8D0   C0 50       CPY #50       Test for P
#E8D2   D0 02       BNE E8D6
#E8D4   A9 80       LDA #80       Set for P
#E8D6   8D 3B C1    STA C13B      Store #80 for P or #00 for N
#E8D9   20 B1 F6    JSR F6B1      Update the directory
#E8DC   20 D1 F6    JSR F6D1      Update system tack
#E8DF   AD 90 C1    LDA C190
#E8E2   10 06       BPL E8EA      Branch to ‘Overwritten’ or use
#E8E4   A9 13       LDA #13       Code for ‘Created
#E8E6   2C A9 12    BIT 12A9      #E8E7 LDA #12 – ‘Write protected’
#E8E9   2C A9 15    BIT 15A9      #E8EA LDA #15 – ‘Overwritten’
#E8EC   2C A9 14    BIT 14A9      #E8ED LDA #14 – ‘Already exists’
#E8EF   48          PHA           Save the message code
#E8F0   20 6D F5    JSR F56D      Print filename to screen
#E8F3   20 AE F5    JSR F5AE      Print a space
#E8F6   68          PLA           Get the message code back
#E8F7   AA          TAX
#E8F8   20 29 F7    JSR F729      Print system message
#E8FB   20 98 F5    JSR F598      Print CR and LF for new line
#E8FE   AD 91 C1    LDA C191      C status
#E901   30 08       BMI E90B      Branch if not in single drive mode, otherwise
#E903   A2 05       LDX #05       Code for ‘Load source disk and press RETURN’
#E905   20 29 F7    JSR F729      Print system message
#E908   20 86 F5    JSR F586      Wait for RETURN
#E90B   AD 7A C1    LDA C17A      Re-set source drive number
```

```
#E90E    8D 00 C0       STA C000
#E911    A2 08          LDX #08        Copy filename
#E913    BD 7B C1       LDA C17B,X
#E916    9D 2C C1       STA C12C,X
#E919    CA             DEX
#E91A    10 F7          BPL E913
#E91C    20 E2 F6       JSR F6E2       Load system track
#E91F    AD B3 C1       LDA C1B3       Get location of current Track
#E922    AE B4 C1       LDX C1B4       and Sector
#E925    20 0A F4       JSR F40A       Read from disk – block specified by track A / sector X
#E928    AD B5 C1       LDA C1B5       Recover displacement along directory
#E92B    8D 3F C1       STA C13F
#E92E    20 F0 F5       JSR F5F0       Carry on looking in Directory
#E931    F0 03          BEQ E936       Branch if end
#E933    4C F4 E7       JMP E7F4       Go back and copy next file
#E936    60             RTS
```

!STORE

```
#E937    20 8D E9       JSR E98D       Check syntax of statement
#E93A    20 FB F7       JSR F7FB       Open the file for writing and print 'Saving …'
#E93D    A0 05          LDY #05        Output as header on the STORE file:
#E93F    B9 A7 02       LDA 02A7,Y          #02A7/8 – variable type
#E942    20 BD EB       JSR EBBD            #02A9/A – start address
#E945    88             DEY                 #02AB/C – end address - 1
#E946    10 F7          BPL E93F
#E948    AD A7 02       LDA 02A7       Test variable type
#E94B    30 17          BMI E964       Branch if $ variables
#E94D    A0 00          LDY #00        Store Real / % variables
#E94F    20 E1 E9       JSR E9E1       Check if at end
#E952    F0 0D          BEQ E961       Close and exit if done
#E954    B1 0C          LDA (0C),Y     Fetch a byte
#E956    20 BD EB       JSR EBBD       Store it
#E959    E6 0C          INC 0C         Increment address pointers
#E95B    D0 F2          BNE E94F
#E95D    E6 0D          INC 0D
#E95F    D0 EE          BNE E94F       Go back for more
#E961    4C 26 EB       JMP EB26       Close file for writing and exit
#E964    20 E1 E9       JSR E9E1       Do $ variables.  Check if at end
#E967    F0 F8          BEQ E961       Close and exit if last $ done
#E969    A0 00          LDY #00
#E96B    B1 0C          LDA (0C),Y     Fetch a byte
#E96D    48             PHA            Save it
#E96E    20 BD EB       JSR EBBD       Output number of bytes in current string
#E971    68             PLA            Get the byte back
#E972    F0 14          BEQ E988       Branch if the $ is empty
#E974    AA             TAX            otherwise save as X for counter
#E975    A0 02          LDY #02        Set up pointer for $ in #00D1/2
#E977    B1 0C          LDA (0C),Y
#E979    99 D0 00       STA 00D0,Y
#E97C    88             DEY
#E97D    D0 F8          BNE E977       Go back for second byte of pointer
#E97F    B1 D1          LDA (D1),Y     Get byte of string
#E981    20 BD EB       JSR EBBD       Store it
#E984    C8             INY
#E985    CA             DEX
#E986    D0 F7          BNE E97F       Loop until current $ done
#E988    20 FC E9       JSR E9FC       Increment pointers
#E98B    90 D7          BCC E964       and branch always back for more
```

Syntax and set-up subroutines for STORE and RECALL

```
#E98D     A9 40          LDA #40        Set STORE / RECALL flag
#E98F     85 2B          STA 2B
#E991     20 A3 F7       JSR F7A3       Call routine in original ROM at
#E994     88 D1                         #D188 to get a variable from text (the array name)
#E996     A9 00          LDA 00         Reset flags
#E998     85 2B          STA 2B
#E99A     A5 28          LDA 28         Save variable type bytes
#E99C     A4 29          LDY 29
#E99E     8D A7 02       STA 02A7
#E9A1     8C A8 02       STY 02A8
#E9A4     A0 02          LDY #02        Save offset bytes for next array
#E9A6     B1 CE          LDA (CE),Y
#E9A8     8D A9 02       STA 02A9
#E9AB     C8             INY
#E9AC     B1 CE          LDA (CE),Y
#E9AE     8D AA 02       STA 02AA
#E9B1     A5 CE          LDA CE         Save pointer to current array
#E9B3     48             PHA
#E9B4     A5 CF          LDA CF
#E9B6     48             PHA
#E9B7     20 B3 F5       JSR F5B3       Dispose of comma and get next character
#E9BA     20 DA F7       JSR F7DA       Set-up file name with DAT extension
#E9BD     68             PLA            Recover pointer for array
#E9BE     85 CF          STA CF
#E9C0     68             PLA
#E9C1     85 CE          STA CE
#E9C3     20 E8 00       JSR 00E8       Re-fetch the current character
#E9C6     F0 03          BEQ E9CB       Branch if end of statement
#E9C8     4C 18 EB       JMP EB18       otherwise – Invalid command end error
#E9CB     20 A3 F7       JSR F7A3       Call routine in original ROM at
#E9CE     9E EA                         #EA9E to use part of STORE routine
#E9D0     60             RTS            Exit
#E9D1     AD AD 02       LDA 02AD       Test if at end of array
#E9D4     CD AF 02       CMP 02AF       Exit with Z flag set if contents of
#E9D7     D0 08          BNE E9E1       #02AD/E  = #02AF/B0
#E9D9     AD AE 02       LDA 02AE
#E9DC     CD B0 02       CMP 02B0
#E9DF     F0 0C          BEQ E9ED
#E9E1     A5 0C          LDA 0C         Test if at end of array
#E9E3     CD AB 02       CMP 02AB       Exit with Z flag set if contents of
#E9E6     D0 05          BNE E9ED       #000C/D = #02AB/C
#E9E8     A5 0D          LDA 0D
#E9EA     CD AC 02       CMP 02AC
#E9ED     60             RTS
#E9EE     A9 03          LDA #03        Advance the pointer for $ variables
#E9F0     18             CLC            by adding 3 to #02AD/E
#E9F1     6D AD 02       ADC 02AD
#E9F4     8D AD 02       STA 02AD
#E9F7     90 03          BCC E9FC
#E9F9     EE AE 02       INC 02AE
#E9FC     A9 03          LDA #03        Advance the pointer for $ variables
#E9FE     18             CLC            by adding 3 to #000C/D
#E9FF     65 0C          ADC 0C
#EA01     85 0C          STA 0C
#EA03     90 03          BCC EA08
#EA05     E6 0D          INC 0D
#EA07     18             CLC
#EA08     60             RTS
```

!RECALL

```
#EA09    20 A3 F7     JSR F7A3        Call routine in original ROM at
#EA0C    50 D6                        #D650 – garbage collection routine
#EA0E    20 8D E9     JSR E98D        Check syntax, array type and set up space for array
#EA11    20 42 F8     JSR F842        Open file for reading and print ‘Loading..’
#EA14    A0 03        LDY #03         Counter for 4 bytes
#EA16    20 90 EC     JSR EC90        Fetch a byte (from header)
#EA19    99 AD 02     STA 02AD,Y      and save as start and end pointers
#EA1C    88           DEY
#EA1D    10 F7        BPL EA16
#EA1F    20 90 EC     JSR EC90        Fetch next byte from header (variable type)
#EA22    CD A8 02     CMP 02A8
#EA25    F0 06        BEQ EA2D        Accept if there is a match
#EA27    20 20 EB     JSR EB20        Close file again
#EA2A    4C 9B EA     JMP EA9B        Illegal attribute error if wrong type of variable specified
#EA2D    20 90 EC     JSR EC90        Fetch next byte from header (variable type)
#EA30    CD A7 02     CMP 02A7
#EA33    D0 F2        BNE EA27        Error if second byte doesn’t match too
#EA35    AD A7 02     LDA 02A7        Test for $ variable
#EA38    30 1F        BMI EA59        Branch if $ variable
#EA3A    A0 00        LDY #00         otherwise do Real and % variables
#EA3C    20 D1 E9     JSR E9D1        Test if all done
#EA3F    F0 15        BEQ EA56        Close and exit if all done
#EA41    20 90 EC     JSR EC90        Fetch another byte from file
#EA44    91 0C        STA (0C),Y      Save it in memory
#EA46    E6 0C        INC 0C          Increment pointers
#EA48    D0 02        BNE EA4C
#EA4A    E6 0D        INC 0D
#EA4C    EE AD 02     INC 02AD
#EA4F    D0 EB        BNE EA3C
#EA51    EE AE 02     INC 02AE
#EA54    D0 E6        BNE EA3C        Go back and do more
#EA56    4C 20 EB     JMP EB20        Close and exit
#EA59    20 D1 E9     JSR E9D1        Do $ array - Test if all done
#EA5C    F0 F8        BEQ EA56        Close and exit if all done
#EA5E    A0 00        LDY #00
#EA60    20 90 EC     JSR EC90        Fetch length of $ and save in array block
#EA63    91 0C        STA (0C),Y
#EA65    48           PHA
#EA66    68           PLA
#EA67    F0 1B        BEQ EA84        Branch if empty string
#EA69    20 A3 F7     JSR F7A3        Call routine in original ROM at
#EA6C    AB D5                        #D5AB to get slot in memory for a new string
#EA6E    A0 00        LDY #00         Recall string to memory,
#EA70    AA           TAX             using string length as a counter
#EA71    20 90 EC     JSR EC90        Fetch a byte
#EA74    91 D1        STA (D1),Y
#EA76    C8           INY
#EA77    CA           DEX
#EA78    D0 F7        BNE EA71
#EA7A    A0 02        LDY #02         Set $ pointer in array block
#EA7C    B9 D0 00     LDA 00D0,Y
#EA7F    91 0C        STA (0C),Y
#EA81    88           DEY
#EA82    D0 F8        BNE EA7C
#EA84    20 EE E9     JSR E9EE        Advance pointers by 3
#EA87    90 D0        BCC EA59        and go back for more
```

!OPEN

```
#EA89   20 DA F7    JSR F7DA     Set up filename with default DAT extension
#EA8C   20 B3 F5    JSR F5B3     Dispose of comma and get next character
#EA8F   AA          TAX
#EA90   20 E2 00    JSR 00E2     Fetch the next character
#EA93   E0 52       CPX #52      Check for R (Read)
#EA95   F0 09       BEQ EAA0
#EA97   E0 57       CPX #57      Check for W (Write)
#EA99   F0 2B       BEQ EAC6
#EA9B   A2 07       LDX #07      Code for ‘Illegal attribute’ error
#EA9D   4C 02 F7    JMP F702     Print error message and exit
#EAA0   AD 59 C1    LDA C159     OPEN to Read
#EAA3   20 F0 EA    JSR EAF0     Test validity of request to open
#EAA6   D0 03       BNE EAAB     Accept if file exists
#EAA8   4C C2 E5    JMP E5C2     Jump to ‘File not found’ error
#EAAB   8E 59 C1    STX C159     otherwise, open the file for Reading
#EAAE   BD 2F C0    LDA C02F,X
#EAB1   8D 00 C3    STA C300
#EAB4   BD 2E C0    LDA C02E,X
#EAB7   8D 01 C3    STA C301
#EABA   AD 2B C1    LDA C12B     Drive number
#EABD   8D 58 C1    STA C158
#EAC0   A2 00       LDX #00
#EAC2   8E 5A C1    STX C15A
#EAC5   60          RTS

#EAC6   AD 5B C1    LDA C15B     OPEN to Write
#EAC9   20 F0 EA    JSR EAF0     Test validity of request to write
#EACC   F0 05       BEQ EAD3     Accept if the file does not exist
#EACE   A2 09       LDX #09      Code for ‘File already exists’ error
#EAD0   4C 02 F7    JMP F702     Print error message and exit
#EAD3   20 02 F6    JSR F602     Check for sufficient disk space
#EAD6   D0 03       BNE EADB     Branch if OK
#EAD8   4C B5 E2    JMP E2B5     Jump to ‘Insufficient disk space’ error
#EADB   A9 02       LDA #02      otherwise open the file for writing
#EADD   8D 5C C1    STA C15C
#EAE0   A9 00       LDA #00
#EAE2   8D 3B C1    STA C13B
#EAE5   20 5B EC    JSR EC5B     Copy a block of bytes
#EAE8   A9 01       LDA #01
#EAEA   8D 5B C1    STA C15B
#EAED   60          RTS
#EAEE   EA          NOP
#EAEF   EA          NOP
#EAF0   F0 05       BEQ EAF7     Test for file already open
#EAF2   A2 0B       LDX #0B      Code for ‘File open’ error
#EAF4   4C 02 F7    JMP F702     Print error message and exit
#EAF7   20 27 F5    JSR F527     Check for wildcards when not allowed
#EAFA   AE 2B C1    LDX C12B     Get the drive number
#EAFD   20 E1 E5    JSR E5E1     Check for illegal drive number
#EB00   4C C2 F5    JMP F5C2     Look for specified file on disk
```

!CLOSE

```
#EB03   20 E8 00    JSR 00E8     Re-fetch the current character
#EB06   F0 15       BEQ EB1D     If end of statement, close for reading and writing
#EB08   AA          TAX
#EB09   20 E2 00    JSR 00E2     Fetch the next character
#EB0C   E0 2C       CPX #2C      Test for , (comma)
```

```
#EB0E    F0 F8          BEQ EB08       Loop to get new character
#EB10    E0 52          CPX #52        Test for R (Close for reading)
#EB12    F0 0C          BEQ EB20       Branch to close for reading routine
#EB14    E0 57          CPX #57        Test for W (Close for writing)
#EB16    F0 0E          BEQ EB26       Branch to close for writing routine
#EB18    A2 02          LDX #02        Code for ‘Invalid command end’ error
#EB1A    4C 02 F7       JMP F702       Print error message and exit
#EB1D    20 26 EB       JSR EB26       Close for writing, then
#EB20    A9 00          LDA #00        Close for reading
#EB22    8D 59 C1       STA C159
#EB25    60             RTS
#EB26    AD 5B C1       LDA C15B       Close for writing
#EB29    F0 2F          BEQ EB5A       Branch if file not open (already closed)
#EB2B    AD 73 C1       LDA C173
#EB2E    F0 2A          BEQ EB5A
#EB30    A9 00          LDA #00        Fill rest of block with 0 (zero / null)
#EB32    AC 5C C1       LDY C15C
#EB35    99 00 C2       STA C200,Y
#EB38    C8             INY
#EB39    C0 02          CPY #02
#EB3B    D0 F8          BNE EB35
#EB3D    20 5E EB       JSR EB5E       Write final block to disk
#EB40    20 73 EC       JSR EC73       Part of GET routine
#EB43    AD 3D C1       LDA C13D
#EB46    8D 01 C0       STA C001
#EB49    AD 3E C1       LDA C13E
#EB4C    8D 02 C0       STA C002
#EB4F    20 7F EC       JSR EC7F       Part of GET routine
#EB52    20 B1 F6       JSR F6B1       Update the directory
#EB55    20 D1 F6       JSR F6D1       Update system tack
#EB58    A9 00          LDA #00
#EB5A    8D 5B C1       STA C15B       Mark as closed
#EB5D    60             RTS            Exit
#EB5E    AD 65 C1       LDA C165       Write final block to disk
#EB61    8D 00 C0       STA C000
#EB64    A9 00          LDA #00        Set pointers for data transfer to disk
#EB66    8D 03 C0       STA C003
#EB69    A9 C2          LDA #C2
#EB6B    8D 04 C0       STA C004
#EB6E    AD 73 C1       LDA C173
#EB71    8D 02 C0       STA C002       Sector
#EB74    AD 74 C1       LDA C174
#EB77    8D 01 C0       STA C001       Track
#EB7A    4C 06 F4       JMP F406       Output block to disk and return
```

!PUT

```
#EB7D    AD 5B C1       LDA C15B       Test if file open
#EB80    F0 75          BEQ EBF7       Branch to ‘File not open’ error
#EB82    20 A3 F7       JSR F7A3       Call routine in original ROM at
#EB85    17 CF                         #CF17 to evaluate an expression
#EB87    24 28          BIT 28         Test type of expression
#EB89    10 26          BPL EBB1       Branch if numeric (not $)
#EB8B    20 A3 F7       JSR F7A3       Call routine in original ROM at
#EB8E    D0 D7                         #D7D0 to check string type
#EB90    AA             TAX            Save length of string as counter
#EB91    20 BD EB       JSR EBBD       Put on disk
#EB94    8A             TXA            Restore string length
#EB95    F0 0B          BEQ EBA2       Branch if zero
#EB97    A0 00          LDY #00
```

```
#EB99     B1 91          LDA (91),Y      Loop to put $ on disk
#EB9B     20 BD EB       JSR EBBD        Put on disk
#EB9E     C8             INY
#EB9F     CA             DEX             Decrement length of string to copy
#EBA0     D0 F7          BNE EB99        Loop until string done
#EBA2     20 E8 00       JSR 00E8        Re-fetch the current character
#EBA5     F0 09          BEQ EBB0        Exit if end of statement
#EBA7     C9 2C          CMP #2C         Test for , (comma)
#EBA9     D0 05          BNE EBB0        Return if not
#EBAB     20 E2 00       JSR 00E2        Fetch the next character
#EBAE     D0 D2          BNE EB82        Evaluate next expression if not end of line
#EBB0     60             RTS
#EBB1     20 A3 F7       JSR F7A3        Call routine in original ROM at
#EBB4     CB D8                          #D8CB to get single byte integer
#EBB6     8A             TXA
#EBB7     20 BD EB       JSR EBBD        Put on disk
#EBBA     4C A2 EB       JMP EBA2        Go back for more
#EBBD     8D 40 C1       STA C140        Put character (byte) on disk
#EBC0     8E 41 C1       STX C141        Save A, X and Y
#EBC3     98             TYA
#EBC4     48             PHA
#EBC5     AC 5C C1       LDY C15C        Get counter for position in block
#EBC8     D0 13          BNE EBDD        Branch if space in block
#EBCA     20 73 EC       JSR EC73        Open up another block on disk
#EBCD     20 7F EC       JSR EC7F
#EBD0     20 69 F6       JSR F669        Set up sector for save
#EBD3     D0 03          BNE EBD8        Branch if successful
#EBD5     4C B5 E2       JMP E2B5        ‘Insufficient disk space’ error
#EBD8     20 5B EC       JSR EC5B        Copy block of data
#EBDB     A0 02          LDY #02
#EBDD     AD 40 C1       LDA C140
#EBE0     99 00 C2       STA C200,Y
#EBE3     C8             INY
#EBE4     8C 5C C1       STY C15C
#EBE7     D0 03          BNE EBEC
#EBE9     20 5E EB       JSR EB5E        Output block of data to disk
#EBEC     68             PLA             Recover X and Y
#EBED     A8             TAY
#EBEE     AE 41 C1       LDX C141
#EBF1     60             RTS             and exit
```

!GET

```
#EBF2     AD 59 C1       LDA C159        Test if file is open for reading
#EBF5     D0 05          BNE EBFC        Accept if it is
#EBF7     A2 1C          LDX #1C         Code for ‘File not open’ error
#EBF9     4C 02 F7       JMP F702        Print error message and exit
#EBFC     20 A3 F7       JSR F7A3        Call routine in original ROM at
#EBFF     88 D1                          #D188 to get variable from text
#EC01     20 90 EC       JSR EC90        Fetch next byte from disk
#EC04     90 05          BCC EC0B        Branch if not end if file
#EC06     A2 1D          LDX #1D         Code for ‘File end’ error
#EC08     4C 02 F7       JMP F702        Print error message and exit
#EC0B     24 28          BIT 28          Test for $ type
#EC0D     30 2C          BMI EC3B        Branch if $
#EC0F     24 29          BIT 29          Test if Integer % variable
#EC11     30 1E          BMI EC31        Branch if Integer
#EC13     A8             TAY             Otherwise do Real Variable
#EC14     20 A3 F7       JSR F7A3        Call routine in original ROM at
#EC17     B6 D4                          #D4B6 to convert integer to floating point number
```

```
#EC19    A6 B6        LDX B6
#EC1B    A4 B7        LDY B7
#EC1D    20 A3 F7     JSR F7A3       Call routine in original ROM at
#EC20    AD DE                       #DEAD to pack FPA into memory
#EC22    20 E8 00     JSR 00E8       Re-fetch the current character
#EC25    F0 09        BEQ EC30       Branch to exit if end of statement
#EC27    C9 2C        CMP #2C        Test for comma ,
#EC29    D0 05        BNE EC30       Branch to exit if not a comma
#EC2B    20 E2 00     JSR 00E2       Fetch the next character
#EC2E    D0 CC        BNE EBFC       Go back to do more
#EC30    60           RTS            Exit at end of statement
#EC31    A0 01        LDY #01        Do Integer variable %
#EC33    91 B6        STA (B6),Y     Store the integer
#EC35    88           DEY
#EC36    98           TYA
#EC37    91 B6        STA (B6),Y
#EC39    F0 E7        BEQ EC22       Go back for more
                                     Do string variable $
#EC3B    20 A3 F7     JSR F7A3       Call routine in original ROM at
#EC3E    AB D5                       #D5AB to get slot in memory for a new string
#EC40    A6 D0        LDX D0         Fetch length of $
#EC42    F0 0B        BEQ EC4F       Branch if $ is empty
#EC44    A0 00        LDY #00
#EC46    20 90 EC     JSR EC90       Get a character from disk
#EC49    91 D1        STA (D1),Y     Store it away
#EC4B    C8           INY            Increment address counter
#EC4C    CA           DEX            Decrement $ length
#EC4D    D0 F7        BNE EC46       Go back for more unless last character
#EC4F    A0 02        LDY #02
#EC51    B9 D0 00     LDA 00D0,Y
#EC54    91 B6        STA (B6),Y
#EC56    88           DEY
#EC57    10 F8        BPL EC51
#EC59    30 C7        BMI EC22       Go back for more

#EC5B    AD 23 C0     LDA C023
#EC5E    AE 24 C0     LDX C024
#EC61    8D 00 C2     STA C200
#EC64    8E 01 C2     STX C201
#EC67    A2 1C        LDX #1C
#EC69    BD 23 C1     LDA C123,X     Copies 30 bytes
#EC6C    9D 5D C1     STA C15D,X     from #C123
#EC6F    CA           DEX            to #C15D
#EC70    10 F7        BPL EC69
#EC72    60           RTS

#EC73    A2 1C        LDX #1C
#EC75    BD 5D C1     LDA C15D,X     Copies 30 bytes
#EC78    9D 23 C1     STA C123,X     from #C15D
#EC7B    CA           DEX            to #C123
#EC7C    10 F7        BPL EC75
#EC7E    60           RTS

#EC7F    AD 2B C1     LDA C12B       Set start of data pointer to #C023
#EC82    8D 00 C0     STA C000
#EC85    A9 23        LDA #23
#EC87    8D 03 C0     STA C003
#EC8A    A9 C0        LDA #C0
#EC8C    8D 04 C0     STA C004
#EC8F    60           RTS

#EC90    8E 41 C1     STX C141       Save X
```

```
#EC93    AE 5A C1     LDX C15A      Test for end of current data block
#EC96    F0 0B        BEQ ECA3      Get another block from disk if needed
#EC98    BD 00 C3     LDA C300,X    Get the next byte
#EC9B    EE 5A C1     INC C15A      Increment the pointer to the next byte
#EC9E    18           CLC
#EC9F    AE 41 C1     LDX C141      Recover X
#ECA2    60           RTS           Return with byte in A

#ECA3    AD 58 C1     LDA C158      Get another block of data from disk into memory
#ECA6    8D 00 C0     STA C000
#ECA9    A9 00        LDA #00       Set address pointer to #C300 for block
#ECAB    8D 03 C0     STA C003
#ECAE    A9 C3        LDA #C3
#ECB0    8D 04 C0     STA C004
#ECB3    38           SEC
#ECB4    AE 01 C3     LDX C301      Test for file end
#ECB7    F0 E6        BEQ EC9F      Exit if file end
#ECB9    98           TYA
#ECBA    48           PHA
#ECBB    AD 00 C3     LDA C300
#ECBE    20 0A F4     JSR F40A      Read from disk – block specified by track A / sector X
#ECC1    68           PLA
#ECC2    A8           TAY
#ECC3    A2 02        LDX #02
#ECC5    8E 5A C1     STX C15A
#ECC8    D0 CE        BNE EC98      Exit
```

!FORMAT

```
#ECCA    20 BE F4     JSR F4BE      Set up drive number and disk name
#ECCD    AE 2B C1     LDX C12B      Get the drive number
#ECD0    20 E1 E5     JSR E5E1      Check for illegal drive number
#ECD3    A2 16        LDX #16       Code for ‘Load disk on drive’
#ECD5    20 29 F7     JSR F729      Print system message
#ECD8    AD 00 C0     LDA C000      Add the drive number
#ECDB    09 30        ORA #30       and
#ECDD    20 9F F5     JSR F59F      Print the character
#ECE0    A2 09        LDX #09       Code for ‘and press RETURN’
#ECE2    20 29 F7     JSR F729      Print system message
#ECE5    20 86 F5     JSR F586      Wait for Return
#ECE8    20 98 F5     JSR F598      Print CR and LF for new line
#ECEB    AC 00 C0     LDY C000      Copy drive parameters
#ECEE    B9 13 C0     LDA C013,Y
#ECF1    8D 81 C1     STA C181      Side 1 = 40/80 tracks
#ECF4    B9 17 C0     LDA C017,Y
#ECF7    8D 82 C1     STA C182      Side 2 = 40/80 tracks
#ECFA    A9 00        LDA #00       Set #000C/D to #1000
#ECFC    85 0C        STA 0C        (address to copy data to)
#ECFE    A9 10        LDA #10
#ED00    85 0D        STA 0D
#ED02    A9 00        LDA #00
#ED04    8D 23 C0     STA C023
#ED07    8D 80 C1     STA C180
#ED0A    A8           TAY           Zero X and Y
#ED0B    AA           TAX
#ED0C    A9 01        LDA #01
#ED0E    8D 40 C1     STA C140      Counter
#ED11    8D 24 C0     STA C024
#ED14    A9 00        LDA #00       Set #000C/D to #1000 again
#ED16    85 0C        STA 0C        (address to copy data to)
```

```
#ED18   A9 10        LDA #10
#ED1A   85 0D        STA 0D
#ED1C   20 22 EE     JSR EE22        Copy in set 1 of data bytes
#ED1F   A2 0B        LDX #0B         Offset for set 2 of data to write from table
#ED21   20 22 EE     JSR EE22        Copy in set 2 of data bytes
#ED24   AD 40 C1     LDA C140
#ED27   91 0C        STA (0C),Y      Save more byte from counter
#ED29   C8           INY             Increment storage address
#ED2A   D0 02        BNE ED2E
#ED2C   E6 0D        INC 0D
#ED2E   A2 14        LDX #14         Offset for set 3 of data to write
#ED30   20 22 EE     JSR EE22        Copy in set 3 of data bytes
#ED33   EE 40 C1     INC C140
#ED36   AD 40 C1     LDA C140        Counter
#ED39   C9 11        CMP #11
#ED3B   90 E2        BCC ED1F        Loop 16 times
#ED3D   A2 27        LDX #27         Offset for set 4 of data to write
#ED3F   20 22 EE     JSR EE22        Copy in set 4 of data bytes
#ED42   A0 08        LDY #08
#ED44   20 12 F4     JSR F412        Write data out to disk
#ED47   A9 70        LDA #70         Set address in #0C/D to #1070
#ED49   85 0C        STA 0C
#ED4B   A9 10        LDA #10
#ED4D   85 0D        STA 0D
#ED4F   20 44 EE     JSR EE44        Not sure what this does
#ED52   A2 00        LDX #00
#ED54   A0 00        LDY #00
#ED56   AD 80 C1     LDA C180
#ED59   29 7F        AND #7F
#ED5B   91 0C        STA (0C),Y
#ED5D   AD 80 C1     LDA C180
#ED60   10 01        BPL ED63
#ED62   C8           INY
#ED63   98           TYA
#ED64   A0 01        LDY #01
#ED66   91 0C        STA (0C),Y
#ED68   A0 2B        LDY #2B
#ED6A   BD 26 C0     LDA C026,X
#ED6D   91 0C        STA (0C),Y
#ED6F   C8           INY
#ED70   BD 36 C0     LDA C036,X
#ED73   91 0C        STA (0C),Y
#ED75   A5 0C        LDA 0C
#ED77   18           CLC
#ED78   69 64        ADC #64
#ED7A   85 0C        STA 0C
#ED7C   90 02        BCC ED80
#ED7E   E6 0D        INC 0D
#ED80   E6 0D        INC 0D
#ED82   E8           INX
#ED83   EC 81 C1     CPX C181
#ED86   90 CC        BCC ED54
#ED88   A9 00        LDA #00
#ED8A   8D 03 C0     STA C003
#ED8D   A9 10        LDA #10
#ED8F   8D 04 C0     STA C004
#ED92   A0 F0        LDY #F0
#ED94   20 12 F4     JSR F412        Write to disk
#ED97   A0 5B        LDY #5B
#ED99   AD 23 C0     LDA C023
#ED9C   8D 80 C1     STA C180
#ED9F   30 A6        BMI ED47
```

```
#EDA1     D0 A1          BNE ED44
#EDA3     A9 20          LDA #20
#EDA5     A2 00          LDX #00
#EDA7     9D 23 C0       STA C023,X
#EDAA     CA             DEX
#EDAB     D0 FA          BNE EDA7
#EDAD     A2 07          LDX #07        Copy 8 bytes
#EDAF     BD 13 C0       LDA C013,X
#EDB2     9D 23 C0       STA C023,X
#EDB5     CA             DEX
#EDB6     10 F7          BPL EDAF
#EDB8     A9 00          LDA #00
#EDBA     8D 34 C0       STA C034
#EDBD     8D 36 C0       STA C036
#EDC0     8D 38 C0       STA C038
#EDC3     8D 39 C0       STA C039
#EDC6     8D 3A C0       STA C03A
#EDC9     A9 07          LDA #07
#EDCB     8D 33 C0       STA C033
#EDCE     A9 04          LDA #04
#EDD0     8D 35 C0       STA C035
#EDD3     18             CLC
#EDD4     AD 81 C1       LDA C181
#EDD7     6D 82 C1       ADC C182
#EDDA     A2 04          LDX #04
#EDDC     0A             ASL
#EDDD     2E 38 C0       ROL C038
#EDE0     CA             DEX
#EDE1     D0 F9          BNE EDDC
#EDE3     38             SEC
#EDE4     E9 02          SBC #02
#EDE6     8D 37 C0       STA C037
#EDE9     B0 03          BCS EDEE
#EDEB     CE 38 C0       DEC C038
#EDEE     A2 08          LDX #08        Copy disk name onto system track
#EDF0     BD 42 C1       LDA C142,X     with spaces to fill out if required
#EDF3     F0 03          BEQ EDF8
#EDF5     9D 3B C0       STA C03B,X
#EDF8     CA             DEX
#EDF9     10 F5          BPL EDF0
#EDFB     A9 23          LDA #23        Set #C003/4 to point to #C023
#EDFD     A2 C0          LDX #C0
#EDFF     8D 03 C0       STA C003
#EE02     8E 04 C0       STX C004
#EE05     A9 00          LDA #00
#EE07     A2 01          LDX #01
#EE09     20 00 F4       JSR F400       Write out system track
#EE0C     A9 00          LDA #00        Fill page starting at #C023 with zeros
#EE0E     AA             TAX
#EE0F     9D 23 C0       STA C023,X
#EE12     CA             DEX
#EE13     D0 FA          BNE EE0F
#EE15     A2 04          LDX #04
#EE17     20 00 F4       JSR F400       Write out a track
#EE1A     A2 17          LDX #17        Code for 'Formatting complete'
#EE1C     20 29 F7       JSR F729       Print system message
#EE1F     4C E8 00       JMP 00E8       Exit to get next character
```

The next routine enters with a storage address in #0C/D an offset for it in Y, and an offset for a data table in X. The data table is at #EE81 and consists of pairs of bytes. The first byte is the byte to store, and the second byte is the number of times to store it. On exit, #0C/D and Y points to the next free byte for storage

```
#EE22     BD 81 EE       LDA EE81,X     Read a byte from data block later on
```

```
#EE25    C9 FF       CMP #FF       Test for last byte
#EE27    F0 1A       BEQ EE43      Exit if done
#EE29    48          PHA           Save the first byte
#EE2A    E8          INX           Increment data table offset counter
#EE2B    BD 81 EE    LDA EE81,X    Read second byte
#EE2E    E8          INX           Increment data table offset counter
#EE2F    8E 41 C1    STX C141      Save data table offset counter
#EE32    AA          TAX           Shift second byte to X
#EE33    68          PLA           Recover first byte
#EE34    91 0C       STA (0C),Y    Store first byte
#EE36    C8          INY           Increment indirect address on #000C/D
#EE37    D0 02       BNE EE3B
#EE39    E6 0D       INC 0D
#EE3B    CA          DEX           Decrement counter (second byte)
#EE3C    D0 F6       BNE EE34      Loop back
#EE3E    AE 41 C1    LDX C141      Recover data table offset counter
#EE41    D0 DF       BNE EE22      Branch always to test for more pairs.
#EE43    60          RTS

#EE44    A2 10       LDX #10       Set counter for 16 bytes
#EE46    AC 24 C0    LDY C024
#EE49    98          TYA
#EE4A    18          CLC
#EE4B    69 03       ADC #03
#EE4D    C9 11       CMP #11
#EE4F    90 02       BCC EE53
#EE51    E9 10       SBC #10
#EE53    8D 24 C0    STA C024
#EE56    99 35 C0    STA C035,Y
#EE59    AD 23 C0    LDA C023
#EE5C    99 25 C0    STA C025,Y
#EE5F    CA          DEX           Decrement counter
#EE60    D0 E4       BNE EE46      Loop back for more
#EE62    AE 82 C1    LDX C182
#EE65    F0 04       BEQ EE6B
#EE67    49 80       EOR #80
#EE69    30 0D       BMI EE78
#EE6B    18          CLC
#EE6C    69 01       ADC #01
#EE6E    CD 81 C1    CMP C181
#EE71    90 05       BCC EE78
#EE73    A9 00       LDA #00
#EE75    99 35 C0    STA C035,Y
#EE78    8D 23 C0    STA C023
#EE7B    99 25 C0    STA C025,Y
#EE7E    A2 00       LDX #00
#EE80    60          RTS
                                   Pairs of data bytes for FORMAT
#EE81    4E 28       DTA           Set 1 - 40 bytes of #4E
#EE83    00 0C       DTA           12 bytes of #00
#EE85    F6 03       DTA           3 bytes of #F6
#EE87    FC 01       DTA           1 byte of #FC
#EE89    4E 28 FF    DTA           40 bytes of #4E and end marker (96 bytes in all)
#EE8C    00 0C       DTA           Set 2 - 12 bytes of #00
#EE8E    F5 03       DTA           3 bytes of #F5
#EE90    FE 01       DTA           1 byte of #FE
#EE92    00 02 FF    DTA           2 bytes of #00 and end marker (18 bytes in all)
#EE95    01 01       DTA           Set 3 - 1 byte of #01
#EE97    F7 01       DTA           1 byte of #F7
#EE99    4E 16       DTA           22 bytes of #4E
#EE9B    00 0C       DTA           12 bytes of #00
```

```
#EE9D    F5 03       DTA           3 bytes of #F5 and end marker (39 bytes in all)
#EE9F    FB 01       DTA           1 byte of #FB
#EEA1    40 00       DTA           256 bytes of #40
#EEA3    F7 01       DTA           1 byte of #F7
#EEA5    4E 28 FF    DTA           40 bytes of #4E and end marker (298 bytes in all)
#EEA8    4E 00 FF    DTA           Set 4 - 256 bytes of #4E and end marker
#EEAB    4E 00 FF    DTA           Set 5 - 256 bytes of #4E and end marker
#EEAE    4E 00 FF    DTA           Set 6 - 256 bytes of #4E and end marker
```

!STAT

```
#EEB1    20 31 EF    JSR EF31      Copy current STAT to #C023 >
#EEB4    20 E8 00    JSR 00E8      Re-fetch the current character
#EEB7    F0 0F       BEQ EEC8      Branch if end of statement
#EEB9    C9 30       CMP #30       Test for 0
#EEBB    F0 05       BEQ EEC2      Accept if STAT 0
#EEBD    A2 07       LDX #07       Code for ‘Illegal attribute’ error
#EEBF    4C 02 F7    JMP F702      Print error message and exit
#EEC2    20 E2 00    JSR 00E2      Fetch the next character
#EEC5    20 F3 F7    JSR F7F3      Read system info from drive 0
#EEC8    20 98 F5    JSR F598      Print CR and LF for new line
#EECB    A2 1A       LDX #1A       Code for ‘DSTEP –‘
#EECD    20 29 F7    JSR F729      Print system message
#EED0    AC 2B C0    LDY C02B      Fetch DSTEP code
#EED3    BE 28 F0    LDX F028,Y    and look up its value in table
#EED6    20 55 F7    JSR F755      Print it out followed by
#EED9    A2 1B       LDX #1B       Code for ‘ms’
#EEDB    20 29 F7    JSR F729      Print system message
#EEDE    20 98 F5    JSR F598      Print CR and LF for new line
#EEE1    A0 00       LDY #00       Initialise to drive 0
#EEE3    8C 40 C1    STY C140
#EEE6    AC 40 C1    LDY C140
#EEE9    C0 04       CPY #04       Stop at drive 4
#EEEB    B0 43       BCS EF30      Exit if all done
#EEED    B9 23 C0    LDA C023,Y    Test for drive n
#EEF0    F0 39       BEQ EF2B      Branch if not present
#EEF2    A2 0C       LDX #0C       Code for ‘Drive’
#EEF4    20 29 F7    JSR F729      Print system message
#EEF7    20 AE F5    JSR F5AE      Print a space
#EEFA    AD 40 C1    LDA C140      Fetch drive number
#EEFD    18          CLC
#EEFE    69 30       ADC #30       turn it into an ASCII character
#EF00    20 9F F5    JSR F59F      Print the character (drive number)
#EF03    20 AE F5    JSR F5AE      Print a space
#EF06    A9 2D       LDA #2D       - character
#EF08    20 9F F5    JSR F59F      Print the character
#EF0B    AC 40 C1    LDY C140      Fetch drive number
#EF0E    B9 23 C0    LDA C023,Y    Fetch number of tracks
#EF11    AA          TAX
#EF12    20 55 F7    JSR F755      Print out number of tracks
#EF15    A2 0D       LDX #0D       Code for ‘Track’
#EF17    20 29 F7    JSR F729      Print system message
#EF1A    A2 18       LDX #18       Code for ‘, Single-sided’
#EF1C    AC 40 C1    LDY C140      Fetch drive number
#EF1F    B9 27 C0    LDA C027,Y    Test for single or double sided
#EF22    F0 01       BEQ EF25      Skip increment message number if single sided
#EF24    E8          INX           Code for ‘, Double-sided’
#EF25    20 29 F7    JSR F729      Print system message
#EF28    20 98 F5    JSR F598      Print CR and LF for new line
#EF2B    EE 40 C1    INC C140      Increment drive number
```

```
#EF2E     D0 B6          BNE EEE6       Branch always
#EF30     60             RTS

#EF31     A2 08          LDX #08        Copy 9 bytes of system information
#EF33     BD 13 C0       LDA C013,X     from #C013>
#EF36     9D 23 C0       STA C023,X     to     #C023>
#EF39     CA             DEX
#EF3A     10 F7          BPL EF33
#EF3C     60             RTS
```

!SET

```
#EF3D     20 E8 00       JSR 00E8       Re-fetch the current character
#EF40     D0 0E          BNE EF50       Branch if not end of statement
#EF42     20 F3 F7       JSR F7F3       Read in system track
#EF45     20 31 EF       JSR EF31       Copy 9 bytes
#EF48     4C 06 F4       JMP F406       Write out to disk and exit
#EF4B     A2 04          LDX #04        Code for 'Bad drive number' error
#EF4D     4C 02 F7       JMP F702       Print error message and exit
#EF50     B0 F9          BCS EF4B       Error if character not 0-9
#EF52     E9 2F          SBC #2F
#EF54     C9 04          CMP #04
#EF56     B0 F3          BCS EF4B       Error if drive number > 3
#EF58     8D 40 C1       STA C140       Save drive number
#EF5B     A8             TAY
#EF5C     A9 00          LDA #00
#EF5E     99 13 C0       STA C013,Y     Set drive parameter to 0 (not present)
#EF61     99 17 C0       STA C017,Y
#EF64     20 E2 00       JSR 00E2       Fetch the next character
#EF67     F0 37          BEQ EFA0       Exit if end of statement (drive removed from system)
#EF69     20 B3 F5       JSR F5B3       Dispose of comma and get next character
#EF6C     20 A3 F7       JSR F7A3       Call routine in original ROM at
#EF6F     53 E8                         #E853 to get a 2 byte integer
#EF71     C9 00          CMP #00
#EF73     D0 09          BNE EF7E       Error if number >255
#EF75     98             TYA
#EF76     C9 28          CMP #28
#EF78     F0 09          BEQ EF83       Accept if 40 (track)
#EF7A     C9 50          CMP #50
#EF7C     F0 05          BEQ EF83       Accept if 80 (track)
#EF7E     A2 07          LDX #07        Code for 'Illegal attribute' error
#EF80     4C 02 F7       JMP F702       Print error message and exit
#EF83     AC 40 C1       LDY C140       Fetch drive number
#EF86     99 13 C0       STA C013,Y     Save number of tracks
#EF89     20 B3 F5       JSR F5B3       Dispose of comma and get next character
#EF8C     A2 00          LDX #00        Default for single sided
#EF8E     C9 53          CMP #53        Test for S (single sided)
#EF90     F0 07          BEQ EF99       Accept if S
#EF92     C9 44          CMP #44        Test for D (double sided)
#EF94     D0 E8          BNE EF7E       Reject if not D
#EF96     BE 13 C0       LDX C013,Y     Fetch number of tracks for side 2
#EF99     8A             TXA
#EF9A     99 17 C0       STA C017,Y     Save number of tracks on side 2 (or 0)
#EF9D     20 E2 00       JSR 00E2       Fetch the next character
#EFA0     60             RTS
#EFA1     60             RTS
```

!PROT

```
#EFA2    20 66 F4     JSR F466      Set up filename
#EFA5    AE 2B C1     LDX C12B      Get the drive number
#EFA8    20 E1 E5     JSR E5E1      Check for illegal drive number
#EFAB    A9 80        LDA #80       Default code for P status
#EFAD    A2 3F        LDX #3F
#EFAF    8D 3D C1     STA C13D
#EFB2    8E 3E C1     STX C13E
#EFB5    20 E8 00     JSR 00E8      Re-fetch the current character
#EFB8    F0 2F        BEQ EFE9      Branch if end of statement
#EFBA    20 B3 F5     JSR F5B3      Dispose of comma and get next character
#EFBD    A8           TAY
#EFBE    20 E2 00     JSR 00E2      Fetch the next character
#EFC1    C0 4E        CPY #4E       Test for N (Unprotect)
#EFC3    D0 06        BNE EFCB      Branch if not N
#EFC5    A2 3F        LDX #3F
#EFC7    A9 00        LDA #00       Change code to N status
#EFC9    F0 E4        BEQ EFAF      Branch always to save codes
#EFCB    C0 50        CPY #50       Test for P (Protect)
#EFCD    D0 07        BNE EFD6      Branch if not P
#EFCF    AD 3D C1     LDA C13D
#EFD2    09 80        ORA #80       Set code for P
#EFD4    30 D9        BMI EFAF      Branch to save codes
#EFD6    C0 49        CPY #49       Test for I (Invisible)
#EFD8    D0 0A        BNE EFE4      Error if not I
#EFDA    A2 3F        LDX #3F
#EFDC    AD 3D C1     LDA C13D
#EFDF    09 40        ORA #40       Set code for I
#EFE1    4C AF EF     JMP EFAF      Branch always to save codes
#EFE4    A2 07        LDX #07       Code for ‘Illegal attribute’ error
#EFE6    4C 02 F7     JMP F702      Print error message and exit
#EFE9    20 C2 F5     JSR F5C2      Look for file of specified name
#EFEC    D0 05        BNE EFF3      Branch if found
#EFEE    A2 01        LDX #01       Code for ‘File not found’ error
#EFF0    4C 02 F7     JMP F702      Print error message and exit
#EFF3    AE 3F C1     LDX C13F
#EFF6    BD 32 C0     LDA C032,X
#EFF9    2D 3E C1     AND C13E      Reset the NPI codes
#EFFC    0D 3D C1     ORA C13D
#EFFF    9D 32 C0     STA C032,X
#F002    20 06 F4     JSR F406      Write track back to disk
#F005    20 F0 F5     JSR F5F0      Carry on looking through directory for matching filenames
#F008    D0 E9        BNE EFF3      Branch if another matching file found
#F00A    60           RTS           otherwise exit
```

!DSTEP

```
#F00B    20 A3 F7     JSR F7A3      Call routine in original ROM at
#F00E    53 E8                      #E853 to get a 2 byte integer
#F010    C9 00        CMP #00
#F012    D0 0B        BNE F01F      Error if > 255
#F014    98           TYA
#F015    A2 03        LDX #03
#F017    DD 28 F0     CMP F028,X    Compare with the acceptable values in table
#F01A    F0 08        BEQ F024      Branch if match found
#F01C    CA           DEX
#F01D    10 F8        BPL F017      Loop to test other permitted values
#F01F    A2 07        LDX #07       Code for ‘Illegal attribute’ error
#F021    4C 02 F7     JMP F702      Print error message and exit
```

```
#F024    8E 1B C0       STX C01B      Save DSTEP value
#F027    60             RTS           Exit
#F028    06 0C 14 1E                  Data bytes for DSTEP values
```

!NAME

```
#F02C    20 BE F4       JSR F4BE      Set up disk name
#F02F    AE 2B C1       LDX C12B      Get the drive number
#F032    20 E1 E5       JSR E5E1      Check for illegal drive number
#F035    20 F1 F6       JSR F6F1      Read in system sector
#F038    A2 08          LDX #08       Copy in new disk name
#F03A    BD 42 C1       LDA C142,X
#F03D    D0 02          BNE F041
#F03F    A9 20          LDA #20       Use space for nulls
#F041    9D 3B C0       STA C03B,X
#F044    CA             DEX
#F045    10 F3          BPL F03A      Loop for each character of the disk name
#F047    4C 06 F4       JMP F406      Write out system track and exit
```

!LDIR

```
#F04A    20 A3 F7       JSR F7A3      Call routine in original ROM at
#F04D    16 C8                        #C816 to set output to printer
#F04F    20 B8 E3       JSR E3B8      Do !DIR
#F052    20 A3 F7       JSR F7A3      Call routine in original ROM at
#F055    2F C8                        #C82F to set output to screen
#F057    60             RTS           Exit

#F058 -  60             RTS           Spare - 936 bytes of blank space
#F2FF    60             RTS
```

!READ

```
#F300    20 E8 00       JSR 00E8      Re-fetch the current character
#F303    B0 25          BCS F32A      Error if not 0 - 9
#F305    E9 2F          SBC #2F
#F307    C9 04          CMP #04
#F309    B0 1F          BCS F32A      Error if > 3
#F30B    8D 40 C1       STA C140      Save drive number
#F30E    AA             TAX
#F30F    20 E1 E5       JSR E5E1      Check for illegal drive number
#F312    20 E2 00       JSR 00E2      Fetch the next character
#F315    20 B3 F5       JSR F5B3      Dispose of comma and get next character
#F318    20 A3 F7       JSR F7A3      Call routine in original ROM at
#F31B    53 E8                        #E853 to get a 2 byte integer
#F31D    C9 00          CMP #00
#F31F    D0 09          BNE F32A      Error if track > 255
#F321    98             TYA           Accept if < number of tracks on drive
#F322    AC 40 C1       LDY C140      Drive number
#F325    D9 13 C0       CMP C013,Y
#F328    90 05          BCC F32F      Accept
#F32A    A2 07          LDX #07       Code for ‘Illegal attribute’ error
#F32C    4C 02 F7       JMP F702      Print error message and exit
#F32F    8D 01 C0       STA C001      Save Track number
#F332    20 B3 F5       JSR F5B3      Dispose of comma and get next character
#F335    20 A3 F7       JSR F7A3      Call routine in original ROM at
#F338    53 E8                        #E853 to get a 2 byte integer
#F33A    C9 00          CMP #00
```

```
#F33C      D0 EC         BNE F32A      Error if Sector > 255
#F33E      98            TYA
#F33F      F0 E9         BEQ F32A      Error if Sector = 0
#F341      C9 11         CMP #11
#F343      B0 E5         BCS F32A      Error if Sector > 16
#F345      8D 02 C0      STA C002      Save Sector number
#F348      20 B3 F5      JSR F5B3      Dispose of comma and get next character
#F34B      20 A3 F7      JSR F7A3      Call routine in original ROM at
#F34E      53 E8                       #E853 to get a 2 byte integer
#F350      A9 23         LDA #23       Leaves address at which to load the sector in #0033/4
#F352      A0 C0         LDY #C0
#F354      8D 03 C0      STA C003      Address at which to load the track
#F357      8C 04 C0      STY C004      Address at which to load the track
#F35A      20 10 F4      JSR F410      Load the required sector into #C023 >
#F35D      A0 00         LDY #00       Copy the sector to the required memory location
#F35F      B9 23 C0      LDA C023,Y
#F362      91 33         STA (33),Y
#F364      C8            INY
#F365      D0 F8         BNE F35F      Loop for 256 bytes
#F367      60            RTS           Exit
```

!CALL

```
#F368      20 A3 F7      JSR F7A3      Call routine in original ROM at
#F36B      53 E8                       #E853 to get a 2 byte integer
#F36D      6C 33 00      JMP (0033)    Jump to the routine address, with the shadow ROM paged in
#F370      60            RTS

#F371 -    60            RTS           Spare - 143 bytes of blank space
#F3FF      60            RTS
```

Write to disk

```
#F400      8D 01 C0      STA C001      Save Track number
#F403      8E 02 C0      STX C002      Save Sector number
#F406      A0 A0         LDY #A0       Flag for Write
#F408      D0 08         BNE F412      Branch always
```

Read from disk

Read in a block of data from disk, using Track number in A and Sector number in X

```
#F40A      8D 01 C0      STA C001      Save Track number
#F40D      8E 02 C0      STX C002      Save Sector number
#F410      A0 80         LDY #80       Flag for Read
#F412      20 80 F9      JSR F980      Read or write as determined by Y
#F415      B0 01         BCS F418      Error if C set
#F417      60            RTS           otherwise exit
#F418      AD FE 04      LDA 04FE      Determine type of error
#F41B      29 40         AND #40
#F41D      F0 05         BEQ F424      Branch if disk error
#F41F      A2 1A         LDX #1A       Code for ‘Disk write protected’ error
#F421      4C 02 F7      JMP F702      Print error message and exit
#F424      A2 0B         LDX #0B       Code for ‘Disk error’
#F426      20 29 F7      JSR F729      Print system message
#F429      AE FE 04      LDX 04FE
#F42C      20 55 F7      JSR F755      Print out single byte number in decimal form
#F42F      A2 0C         LDX #0C       Code for ‘Drive’
#F431      20 29 F7      JSR F729      Print system message
```

```
#F434    AE 2B C1    LDX C12B     Get the drive number
#F437    20 55 F7    JSR F755     Print out single byte number in decimal form
#F43A    A2 0D       LDX #0D      Code for 'Track'
#F43C    20 29 F7    JSR F729     Print system message
#F43F    AE 01 C0    LDX C001     Get track number
#F442    20 55 F7    JSR F755     Print out single byte number in decimal form
#F445    A2 0E       LDX #0E      Code for 'sector'
#F447    20 29 F7    JSR F729     Print system message
#F44A    AE 02 C0    LDX C002     Get sector number
#F44D    20 55 F7    JSR F755     Print out single byte number in decimal form
#F450    20 98 F5    JSR F598     Print CR and LF for new line
#F453    20 A3 F7    JSR F7A3     Call routine in original ROM at
#F456    03 C0                    #C003 to RESTART BASIC
#F458    8A          TXA
#F459    18          CLC
#F45A    ED 40 C1    SBC C140
#F45D    AE 40 C1    LDX C140
#F460    9D 23 C0    STA C023,X
#F463    4C 06 F4    JMP F406
```

Get filename from text

This routine gets a filename from text into a temporary storage area at #C142 >.   It then copies it down to #C12C > into 6.3 format for filename.extension, padding it out with spaces or wildcards as appropriate.

```
#F466    20 BE F4    JSR F4BE     Read filename from text into $ buffer at #C142
#F469    A2 08       LDX #08      Counter
#F46B    A0 00       LDY #00      Index
#F46D    B9 42 C1    LDA C142,Y   Fetch first character
#F470    D0 03       BNE F475     Branch if string present
#F472    A9 3F       LDA #3F      Use ? (wildcard) if no string
#F474    2C A9 20    BIT 20A9     #F475 – LDA #20 (space character)
#F477    9D 2C C1    STA C12C,X   Save character as space or ?
#F47A    CA          DEX          Continue for 9 characters
#F47B    10 FA       BPL F477     to end up with 9 spaces or 9x? if no string
#F47D    A2 00       LDX #00
#F47F    A9 06       LDA #06      Number of characters in filename
#F481    20 97 F4    JSR F497     Copy the first part of the filename
#F484    B9 42 C1    LDA C142,Y   Fetch next character
#F487    F0 28       BEQ F4B1     Exit if end
#F489    C8          INY          Increment character counter
#F48A    C9 2E       CMP #2E      Test for  .
#F48C    F0 05       BEQ F493     Branch if  .  found
#F48E    A2 05       LDX #05      Code for 'Invalid filename' error
#F490    4C 02 F7    JMP F702     Print error message and exit
#F493    A2 06       LDX #06      Index for second part of filename
#F495    A9 03       LDA #03      Number of characters in extension
#F497    8D 41 C1    STA C141     Save number of characters to process
#F49A    B9 42 C1    LDA C142,Y   Fetch a character
#F49D    F0 12       BEQ F4B1     Exit if done
#F49F    C9 2E       CMP #2E      Test for .
#F4A1    F0 0E       BEQ F4B1     Exit if  .  found
#F4A3    C8          INY          Increment character counter
#F4A4    C9 2A       CMP #2A      Test for  *
#F4A6    F0 0A       BEQ F4B2     Branch if  *  used to fill with  ? wildcards
#F4A8    9D 2C C1    STA C12C,X   otherwise use the character found
#F4AB    E8          INX          Next character
#F4AC    CE 41 C1    DEC C141     Decrement character counter
#F4AF    D0 E9       BNE F49A     Do more if present
#F4B1    60          RTS          otherwise exit
```

```
#F4B2   A9 3F       LDA #3F       ?  character
#F4B4   9D 2C C1    STA C12C,X    Fill up the rest of the space with wildcards
#F4B7   E8          INX
#F4B8   CE 41 C1    DEC C141
#F4BB   D0 F7       BNE F4B4
#F4BD   60          RTS

                                  Read filename from text into $ buffer
#F4BE   AD 0C C0    LDA C00C      Set default drive number
#F4C1   8D 2B C1    STA C12B
#F4C4   A2 09       LDX #09       Empty the space for the filename
#F4C6   A9 00       LDA #00       by filling with 0 (null)
#F4C8   9D 42 C1    STA C142,X
#F4CB   CA          DEX
#F4CC   10 FA       BPL F4C8
#F4CE   20 E8 00    JSR 00E8      Re-fetch the current character
#F4D1   F0 41       BEQ F514      Exit if done
#F4D3   B0 08       BCS F4DD      Branch if not 0 - 9
#F4D5   E9 2F       SBC #2F       Use as drive number
#F4D7   8D 2B C1    STA C12B      Save specified drive number
#F4DA   4C E2 00    JMP 00E2      Get next character and exit
#F4DD   20 A3 F7    JSR F7A3      Call routine in original ROM at
#F4E0   17 CF                     #CF17 to evaluate an expression
#F4E2   20 A3 F7    JSR F7A3      Call routine in original ROM at
#F4E5   CD D7                     #D7CD to check string type
#F4E7   8D 40 C1    STA C140      Save length of string found
#F4EA   A0 00       LDY #00       Index
#F4EC   C9 02       CMP #02       Test string length
#F4EE   90 12       BCC F502      Branch if less than 2 characters
#F4F0   C8          INY
#F4F1   B1 91       LDA (91),Y    Fetch a character from the string
#F4F3   88          DEY
#F4F4   C9 2D       CMP #2D       Test for  -  character
#F4F6   D0 0A       BNE F502      Branch if not  -  character
#F4F8   B1 91       LDA (91),Y
#F4FA   38          SEC
#F4FB   E9 30       SBC #30       Use as drive number
#F4FD   8D 2B C1    STA C12B      Save drive number
#F500   A0 02       LDY #02
#F502   A2 00       LDX #00
#F504   CC 40 C1    CPY C140      Compare with string length
#F507   B0 0B       BCS F514      Exit when finished
#F509   B1 91       LDA (91),Y    Fetch a character from the string
#F50B   9D 42 C1    STA C142,X    Store it
#F50E   E8          INX
#F50F   C8          INY
#F510   E0 0A       CPX #0A
#F512   90 F0       BCC F504      Continue for up to 9 characters
#F514   60          RTS           then exit

                                  Copy filename and its data bytes
#F515   A0 00       LDY #00       Counter
#F517   AE 3F C1    LDX C13F      Get displacement and copy filename and its data
#F51A   BD 23 C0    LDA C023,X    from the directory (in #C023 >)
#F51D   99 2C C1    STA C12C,Y    into memory slot at #C12C
#F520   E8          INX
#F521   C8          INY
#F522   C0 10       CPY #10       Continue for 16 bytes
#F524   90 F4       BCC F51A
#F526   60          RTS
```

Check for wildcards if not allowed

```
#F527   A2 08      LDX #08       Counter for characters
#F529   BD 2C C1   LDA C12C,X    Fetch a character
#F52C   C9 3F      CMP #3F       Test for ?
#F52E   F0 04      BEQ F534      Error if wildcard found
#F530   CA         DEX
#F531   10 F6      BPL F529      Continue for 9 characters
#F533   60         RTS           Exit if none found
#F534   A2 08      LDX #08       Code for ‘Wildcards not allowed’ error
#F536   4C 02 F7   JMP F702      Print error message and exit
```

Print message then filename

```
#F539   EA         NOP           taking filename from #C023 >
#F53A   EA         NOP
#F53B   EA         NOP
#F53C   EA         NOP
#F53D   20 29 F7   JSR F729      Print system message
#F540   20 46 F5   JSR F546      Print out the filename
#F543   4C 98 F5   JMP F598      Print new line and exit

#F546   AE 3F C1   LDX C13F      Print out filename
#F549   A0 06      LDY #06       Counter for filename
#F54B   20 55 F5   JSR F555      Print 6 characters
#F54E   A9 2E      LDA #2E       then the  .
#F550   20 9F F5   JSR F59F      Print the character
#F553   A0 03      LDY #03       Counter for extension
#F555   BD 23 C0   LDA C023,X
#F558   20 9F F5   JSR F59F      Print the character
#F55B   E8         INX
#F55C   88         DEY
#F55D   D0 F6      BNE F555      Loop until required number of characters printed
#F55F   60         RTS           Return or exit

#F560   EA         NOP           Print out message and filename
#F561   EA         NOP           taking filename from #C12C >
#F562   EA         NOP
#F563   EA         NOP
#F564   20 29 F7   JSR F729      Print system message
#F567   20 6D F5   JSR F56D      Print out the filename
#F56A   4C 98 F5   JMP F598      New line and exit

#F56D   A2 00      LDX #00       Print out filename to screen
#F56F   A0 06      LDY #06       Counter for filename
#F571   20 7B F5   JSR F57B      Print 6 characters
#F574   A9 2E      LDA #2E       then the  .
#F576   20 9F F5   JSR F59F      Print the character
#F579   A0 03      LDY #03       Counter for extension
#F57B   BD 2C C1   LDA C12C,X
#F57E   20 9F F5   JSR F59F      Print the character
#F581   E8         INX
#F582   88         DEY
#F583   D0 F6      BNE F57B      Loop until required number of characters printed
#F585   60         RTS           Return or exit
```

Wait for RETURN key

```
#F586   20 A3 F7   JSR F7A3      Call routine in original ROM at
```

```
#F589    E8 C5                          #C5E8 to read key from keyboard
#F58B    C9 0D          CMP #0D         Test for RETURN
#F58D    F0 09          BEQ F598        Branch if found
#F58F    C9 1B          CMP #1B         Test for ESC – Restart basic
#F591    D0 F3          BNE F586        Go back for another key
#F593    20 A3 F7       JSR F7A3        Call routine in original ROM at
#F596    03 C0                          #C003 to RESTART BASIC
#F598    20 A3 F7       JSR F7A3        Call routine in original ROM at
#F59B    F0 CB                          CBF0 to print CR and LF for new line
#F59D    60             RTS             Print new line and exit
#F59E    EA             NOP
```

Print character to screen

```
#F59F    EA             NOP
#F5A0    EA             NOP
#F5A1    EA             NOP
#F5A2    EA             NOP
#F5A3    EA             NOP
#F5A4    20 A3 F7       JSR F7A3        Call routine in original ROM at
#F5A7    D9 CC                          #CCD9 to print character (in A) to screen
#F5A9    18             CLC
#F5AA    60             RTS
#F5AB    20 AE F5       JSR F5AE        Print 2 spaces
#F5AE    A9 20          LDA #20         Print 1 space
#F5B0    4C 9F F5       JMP F59F        Print the character

#F5B3    20 E8 00       JSR 00E8        Re-fetch the current character
#F5B6    C9 2C          CMP #2C         Dispose of comma and fetch next character
#F5B8    F0 05          BEQ F5BF
#F5BA    A2 02          LDX #02         Code for ‘Invalid command end’ error
#F5BC    4C 02 F7       JMP F702        Print error message and exit
#F5BF    4C E2 00       JMP 00E2        Fetch next character
```

Look for file in directory

```
#F5C2    20 E2 F6       JSR F6E2        Read in system track
#F5C5    AD 26 C1       LDA C126        Get track
#F5C8    AE 25 C1       LDX C125        Get sector
#F5CB    20 0A F4       JSR F40A        Read from disk – block specified by track A / sector X
#F5CE    A2 03          LDX #03         Displacement for filename in directory
#F5D0    8E 3F C1       STX C13F        Save it for later if there is a match
#F5D3    A0 00          LDY #00         Set counter for characters in filename
#F5D5    BD 23 C0       LDA C023,X      Is there a file in this slot in the directory?
#F5D8    F0 16          BEQ F5F0        Move to next slot if there is not
#F5DA    B9 2C C1       LDA C12C,Y      Get character from required filename
#F5DD    C9 3F          CMP #3F         ? character
#F5DF    F0 05          BEQ F5E6        Accept a match if it’s a wildcard
#F5E1    DD 23 C0       CMP C023,X      Otherwise try to match it with file in directory
#F5E4    D0 0A          BNE F5F0        Move to next slot if character match fails
#F5E6    E8             INX             Move on to next character
#F5E7    C8             INY
#F5E8    C0 09          CPY #09         Stop at 9 characters (6.3)
#F5EA    90 EE          BCC F5DA        Go back for next character
#F5EC    AE 3F C1       LDX C13F        Get displacement of matched file in directory
#F5EF    60             RTS             Exit with match

#F5F0    AD 3F C1       LDA C13F        Carry on looking in same directory
#F5F3    18             CLC             Increment X by 16
```

```
#F5F4    69 10        ADC #10
#F5F6    AA           TAX
#F5F7    90 D7        BCC F5D0      Go back for more in same directory, or
#F5F9    AD 23 C0     LDA C023      get pointers to the next directory
#F5FC    AE 24 C0     LDX C024
#F5FF    D0 CA        BNE F5CB      Carry on looking if there is another directory
#F601    60           RTS           otherwise exit with Z set when at end of directory.
```

Set up first sector for save

```
#F602    AD 26 C1     LDA C126      Track to locate next directory block
#F605    AE 25 C1     LDX C125      Sector to locate next directory block
#F608    20 0A F4     JSR F40A      Read directory block from disk, specified by track A,  sector X
#F60B    AD 25 C0     LDA C025
#F60E    C9 0F        CMP #0F
#F610    D0 28        BNE F63A      Branch if directory block is not full
#F612    AD 23 C0     LDA C023      otherwise get the track
#F615    AE 24 C0     LDX C024      and sector for the next directory block
#F618    D0 EE        BNE F608      Branch unless finished
#F61A    AD 23 C1     LDA C123
#F61D    F0 49        BEQ F668      Exit if no more directory blocks (disk full?)
#F61F    8D 24 C0     STA C024      Create a new directory block
#F622    AD 24 C1     LDA C124
#F625    8D 23 C0     STA C023
#F628    20 06 F4     JSR F406      Write out to disk
#F62B    20 8B F6     JSR F68B      Set up sector for save
#F62E    A9 00        LDA #00       Empty new directory block
#F630    AA           TAX
#F631    9D 23 C0     STA C023,X
#F634    E8           INX
#F635    D0 FA        BNE F631
#F637    20 06 F4     JSR F406      Write new block out to disk
#F63A    A2 03        LDX #03       Step through directory, looking for
#F63C    BD 23 C0     LDA C023,X    first empty slot
#F63F    F0 07        BEQ F648      Branch when empty slot found
#F641    8A           TXA
#F642    18           CLC           otherwise move on 16 bytes to
#F643    69 10        ADC #10       the next slot in the directory
#F645    AA           TAX
#F646    D0 F4        BNE F63C
#F648    8E 3F C1     STX C13F      Save the displacement along the directory
#F64B    AD 01 C0     LDA C001      Save track and sector information
#F64E    8D 3D C1     STA C13D      for the directory
#F651    AD 02 C0     LDA C002
#F654    8D 3E C1     STA C13E
#F657    A9 00        LDA #00       Zero the number of blocks taken by the file
#F659    8D 35 C1     STA C135
#F65C    8D 36 C1     STA C136
#F65F    20 69 F6     JSR F669      Set up a sector for Save
#F662    8E 37 C1     STX C137
#F665    8D 38 C1     STA C138
#F668    60           RTS

#F669    20 8B F6     JSR F68B      Set up sector for save
#F66C    F0 1C        BEQ F68A      Disk full ?
#F66E    EE 29 C1     INC C129
#F671    D0 03        BNE F676
#F673    EE 2A C1     INC C12A
#F676    EE 35 C1     INC C135
#F679    D0 03        BNE F67E
#F67B    EE 36 C1     INC C136
#F67E    AD 01 C0     LDA C001
#F681    AE 02 C0     LDX C002
```

```
#F684    8E 39 C1    STX C139
#F687    8D 3A C1    STA C13A
#F68A    60          RTS

#F68B    AE 23 C1    LDX C123     Fetch Sector
#F68E    F0 20       BEQ F6B0     Exit if Sector 0
#F690    AD 24 C1    LDA C124     Fetch Track
#F693    20 0A F4    JSR F40A     Read from disk – block specified by track A / sector X
#F696    38          SEC          Reduce the ‘number of sectors free’ by 1
#F697    AD 27 C1    LDA C127
#F69A    E9 01       SBC #01
#F69C    8D 27 C1    STA C127
#F69F    B0 03       BCS F6A4
#F6A1    CE 28 C1    DEC C128
#F6A4    AD 23 C0    LDA C023     Mark the next available sector  / track
#F6A7    AE 24 C0    LDX C024
#F6AA    8D 24 C1    STA C124
#F6AD    8E 23 C1    STX C123
#F6B0    60          RTS
```

Update directory after save

```
#F6B1    AD 3D C1    LDA C13D     Fetch Track for current directory
#F6B4    AE 3E C1    LDX C13E     Fetch Sector for current directory
#F6B7    20 0A F4    JSR F40A     Read from disk – block specified by track A / sector X
#F6BA    A2 00       LDX #00      Copy 16 bytes of information
#F6BC    AC 3F C1    LDY C13F     about the new file into the directory
#F6BF    BD 2C C1    LDA C12C,X
#F6C2    99 23 C0    STA C023,Y
#F6C5    C8          INY
#F6C6    E8          INX
#F6C7    E0 10       CPX #10
#F6C9    90 F4       BCC F6BF
#F6CB    EE 25 C0    INC C025     Increment the number of files in the directory
#F6CE    4C 06 F4    JMP F406     Write the directory out to disk again.
```

Update system track after save

```
#F6D1    20 F1 F6    JSR F6F1     Read in system information
#F6D4    A2 07       LDX #07      and replace 8 bytes of system information
#F6D6    BD 23 C1    LDA C123,X
#F6D9    9D 33 C0    STA C033,X
#F6DC    CA          DEX
#F6DD    10 F7       BPL F6D6
#F6DF    4C 06 F4    JMP F406     Write to Track A, Sector X

#F6E2    20 F1 F6    JSR F6F1     Read in system information
#F6E5    A2 07       LDX #07      and copy 8 bytes of system information from it
#F6E7    BD 33 C0    LDA C033,X   to #C123 >
#F6EA    9D 23 C1    STA C123,X
#F6ED    CA          DEX
#F6EE    10 F7       BPL F6E7
#F6F0    60          RTS
```

Read in system information

```
#F6F1    A9 23       LDA #23      Set address for reading in system track
#F6F3    A0 C0       LDY #C0
#F6F5    8D 03 C0    STA C003
#F6F8    8C 04 C0    STY C004
```

```
#F6FB    A9 00        LDA #00      Set to Track 0
#F6FD    A2 01        LDX #01      Set to Sector 1
#F6FF    4C 0A F4     JMP F40A     Read in the specified block from Track A, Sector X
```

ERROR Messages

```
#F702    8E FF 04     STX 04FF     Save error code
#F705    AD FD 04     LDA 04FD     Test if error messages are inhibited
#F708    F0 0F        BEQ F719     Branch if they are not
#F70A    AE 07 C0     LDX C007     otherwise recover stack pointer
#F70D    9A           TXS
#F70E    20 E8 00     JSR 00E8     Re-fetch the current character
#F711    F0 05        BEQ F718     Branch to exit if end of statement
#F713    20 E2 00     JSR 00E2     Fetch the next character
#F716    D0 FB        BNE F713     Continue until end of statement reached
#F718    60           RTS          exit

#F719    A9 00        LDA #00      Set up address for error messages
#F71B    85 91        STA 91       Store address for start of error
#F71D    A9 FC        LDA #FC      messages at #91/2
#F71F    85 92        STA 92
#F721    20 31 F7     JSR F731     Print out the Xth message
#F724    20 A3 F7     JSR F7A3     Call routine in original ROM at
#F727    96 C4                     #C496 to use part of error message routine
```

SYSTEM Messages

```
#F729    A9 90        LDA #90      Set up address for system messages
#F72B    85 91        STA 91       Store address for start of system
#F72D    A9 FD        LDA #FD      messages at #91/2
#F72F    85 92        STA 92
                                   Print out the Xth message
#F731    A0 00        LDY #00      Re-set character counter
#F733    B1 91        LDA (91),Y   Get character of message
#F735    08           PHP          Save flag (high bit set if last character of message)
#F736    C8           INY          Increment character counter
#F737    D0 02        BNE F73B
#F739    E6 92        INC 92
#F73B    28           PLP          Test character and branch to get next character
#F73C    10 F5        BPL F733     if not the last character of the current message
#F73E    CA           DEX          Decrement the message index code
#F73F    D0 F2        BNE F733     Loop back if correct message not reached yet
#F741    B1 91        LDA (91),Y   Get character from correct message
#F743    08           PHP          Save flag (high bit set if last character of message)
#F744    29 7F        AND #7F      Get rid of the high bit if present
#F746    EA           NOP
#F747    EA           NOP
#F748    EA           NOP
#F749    20 9F F5     JSR F59F     Print the character
#F74C    C8           INY          Increment the address pointer
#F74D    D0 02        BNE F751
#F74F    E6 92        INC 92
#F751    28           PLP          Get the flag back
#F752    10 ED        BPL F741     Go back for next character if not end of message
#F754    60           RTS
```

Print out number in decimal form

```
#F755   A9 00       LDA #00       Print out single byte number in decimal form
#F757   85 D1       STA D1        Print out 2 byte number in decimal form
#F759   86 D2       STX D2
#F75B   A2 90       LDX #90
#F75D   38          SEC
#F75E   20 A3 F7    JSR F7A3      Call routine in original ROM at
#F761   31 DF                     #DF31 part of FPA routine to set mantissa to X
#F763   20 A3 F7    JSR F7A3      Call routine in original ROM at
#F766   D5 E0                     #E0D5 to convert number to string
#F768   85 DE       STA DE        Save string pointer
#F76A   84 DF       STY DF        Save string pointer
#F76C   A0 FF       LDY #FF
#F76E   C8          INY
#F76F   B1 DE       LDA (DE),Y    Fetch a character from the string
#F771   D0 FB       BNE F76E      Get the next one if not the end of string
#F773   C0 05       CPY #05       Carry on for up to 6 characters
#F775   B0 0C       BCS F783
#F777   EA          NOP           get here with the length of the string in Y
#F778   EA          NOP           and print (6-Y) spaces before the string
#F779   EA          NOP
#F77A   EA          NOP
#F77B   20 AE F5    JSR F5AE      Print a space (before the string)
#F77E   C8          INY
#F77F   C0 05       CPY #05       Fill out with spaces to 6 characters
#F781   90 F8       BCC F77B
#F783   A9 01       LDA #01
#F785   A8          TAY
#F786   20 A3 F7    JSR F7A3      Call routine in original ROM at
#F789   B0 CC                     #CCB0 to print out string
#F78B   60          RTS
```

Print byte as 2 ASCII in hex

```
#F78C   48          PHA           Save the number
#F78D   4A          LSR           Get the high nibble
#F78E   4A          LSR
#F78F   4A          LSR
#F790   4A          LSR
#F791   20 97 F7    JSR F797      Print out the high nibble
#F794   68          PLA           Get the number back
#F795   29 0F       AND #0F       Get the low nibble
#F797   18          CLC
#F798   69 30       ADC #30       Convert it into ASCII character
#F79A   C9 3A       CMP #3A
#F79C   90 02       BCC F7A0      Add 6 for A to F
#F79E   69 06       ADC #06
#F7A0   4C 9F F5    JMP F59F      Print out the character and return
```

Original Oric ROM call handler

```
#F7A3   08          PHP           Save all the registers
#F7A4   48          PHA
#F7A5   98          TYA
#F7A6   48          PHA
#F7A7   8A          TXA
#F7A8   48          PHA
#F7A9   BA          TSX           Increment the return address (on the stack) by 2
```

```
#F7AA    BD 05 01        LDA 0105,X      and copy the original return address
#F7AD    85 0E           STA 0E          into #000E/F
#F7AF    18              CLC
#F7B0    69 02           ADC #02
#F7B2    9D 05 01        STA 0105,X
#F7B5    BD 06 01        LDA 0106,X
#F7B8    85 0F           STA 0F
#F7BA    69 00           ADC #00
#F7BC    9D 06 01        STA 0106,X
#F7BF    A0 01           LDY #01
#F7C1    B1 0E           LDA (0E),Y
#F7C3    8D 85 04        STA 0485        Fetch the 2 bytes following the call
#F7C6    C8              INY             and store in #0485/6 as the address
#F7C7    B1 0E           LDA (0E),Y      to call in the original ROM
#F7C9    8D 86 04        STA 0486
#F7CC    A9 02           LDA #02
#F7CE    8D 81 04        STA 0481        Set marker at #0481
#F7D1    68              PLA             Recover all the registers
#F7D2    AA              TAX
#F7D3    68              PLA
#F7D4    A8              TAY
#F7D5    68              PLA
#F7D6    28              PLP
#F7D7    4C 9F 04        JMP 049F        Perform the call to the original ROM and return
```

Set up filename.DAT

```
#F7DA    20 66 F4        JSR F466        Set up filename
#F7DD    AD 32 C1        LDA C132        Get first character of extension
#F7E0    C9 20           CMP #20         Test for blank
#F7E2    D0 0B           BNE F7EF        Exit if an extension is specified
#F7E4    A2 02           LDX #02         otherwise use DAT
#F7E6    BD F0 F7        LDA F7F0,X
#F7E9    9D 32 C1        STA C132,X
#F7EC    CA              DEX
#F7ED    10 F7           BPL F7E6
#F7EF    60              RTS
#F7F0    44 41 54        DTA             DAT (default extension for STORE and RECALL)

#F7F3    A9 00           LDA #00         Read in system sector
#F7F5    8D 00 C0        STA C000
#F7F8    4C F1 F6        JMP F6F1        Jump to read in system information

#F7FB    A2 02           LDX #02         Message code for Saving ....
#F7FD    20 60 F5        JSR F560        Print Saving … + filename message
#F800    20 C6 EA        JSR EAC6        Open file to Write
#F803    4C 4A F8        JMP F84A
#F806    60              RTS
#F807    60              RTS
#F808    60              RTS
#F809    60              RTS
#F80A    60              RTS
                                         Display  A,E,T addresses for code ,D switch
#F80B    AD 4C C1        LDA C14C        Get the A address
#F80E    20 8C F7        JSR F78C        Print out byte as 2 hex characters
#F811    AD 4B C1        LDA C14B
#F814    20 8C F7        JSR F78C        Print out byte as 2 hex characters
#F817    20 AE F5        JSR F5AE        Print a space
#F81A    B9 28 C0        LDA C028,Y      Get the E address
#F81D    20 8C F7        JSR F78C        Print out byte as 2 hex characters
```

```
#F820       B9 27 C0        LDA C027,Y
#F823       20 8C F7        JSR F78C        Print out byte as 2 hex characters
#F826       20 AE F5        JSR F5AE        Print a space
#F829       AE 4D C1        LDX C14D
#F82C       AD 4E C1        LDA C14E
#F82F       F0 07           BEQ F838        Branch if no T address
#F831       20 8C F7        JSR F78C        Print out byte as 2 hex characters
#F834       8A              TXA
#F835       4C 8C F7        JMP F78C        Print out byte as 2 hex characters
#F838       E0 02           CPX #02
#F83A       D0 05           BNE F841
#F83C       A2 1C           LDX #1C         Code for ‘AUTO’
#F83E       20 29 F7        JSR F729        Print system message
#F841       60              RTS

#F842       A2 01           LDX #01         Message code for Loading ... for Recall
#F844       20 60 F5        JSR F560        Print out Loading… + filename
#F847       20 A0 EA        JSR EAA0        Open file to Read
#F84A       AD A9 02        LDA 02A9
#F84D       AC AA 02        LDY 02AA
#F850       85 0C           STA 0C
#F852       84 0D           STY 0D
#F854       60              RTS

#F855 -     60              RTS             Spare - 299 bytes of blank space
#F97F       60              RTS

#F980       20 96 FA        JSR FA96        To / from disk
#F983       20 89 F9        JSR F989        On entry, Y=#80 for Read
#F986       4C 9B FA        JMP FA9B        or Y=#A0 for Read

#F989       8C 05 C0        STY C005        Part of read / write routine
#F98C       AD 00 C0        LDA C000
#F98F       29 03           AND #03
#F991       AA              TAX
#F992       BD 7B FA        LDA FA7B,X
#F995       2C 01 C0        BIT C001
#F998       10 02           BPL F99C
#F99A       09 10           ORA #10
#F99C       8D 14 03        STA 0314
#F99F       AE 80 04        LDX 0480
#F9A2       8D 80 04        STA 0480
#F9A5       29 6C           AND #6C
#F9A7       85 F3           STA F3
#F9A9       8A              TXA
#F9AA       29 6C           AND #6C
#F9AC       C5 F3           CMP F3
#F9AE       F0 2B           BEQ F9DB
#F9B0       C0 10           CPY #10
#F9B2       90 27           BCC F9DB
#F9B4       C0 F0           CPY #F0
#F9B6       F0 23           BEQ F9DB
#F9B8       A9 52           LDA #52
#F9BA       85 F3           STA F3
#F9BC       A9 C1           LDA #C1
#F9BE       85 F4           STA F4
#F9C0       AD 05 C0        LDA C005
#F9C3       48              PHA
#F9C4       A9 C0           LDA #C0
#F9C6       8D 05 C0        STA C005
#F9C9       20 38 FA        JSR FA38
#F9CC       68              PLA
```

```
#F9CD     8D 05 C0        STA C005
#F9D0     AD FE 04        LDA 04FE
#F9D3     D0 78           BNE FA4D
#F9D5     AD 12 03        LDA 0312
#F9D8     8D 11 03        STA 0311
#F9DB     A9 00           LDA #00
#F9DD     8D 06 C0        STA C006
#F9E0     20 04 FA        JSR FA04
#F9E3     D0 02           BNE F9E7
#F9E5     18              CLC
#F9E6     60              RTS
#F9E7     29 18           AND #18
#F9E9     F0 62           BEQ FA4D
#F9EB     AD 06 C0        LDA C006
#F9EE     30 5D           BMI FA4D
#F9F0     D0 05           BNE F9F7
#F9F2     EE 06 C0        INC C006
#F9F5     D0 E9           BNE F9E0
#F9F7     09 80           ORA #80
#F9F9     8D 06 C0        STA C006
#F9FC     A0 08           LDY #08
#F9FE     20 7F FA        JSR FA7F
#FA01     90 DD           BCC F9E0
#FA03     60              RTS
#FA04     AD 03 C0        LDA C003
#FA07     85 F3           STA F3
#FA09     AD 04 C0        LDA C004
#FA0C     85 F4           STA F4
#FA0E     A9 10           LDA #10
#FA10     2C 05 C0        BIT C005
#FA13     10 6A           BPL FA7F
#FA15     70 1F           BVS FA36
#FA17     AD 01 C0        LDA C001
#FA1A     29 7F           AND #7F
#FA1C     CD 11 03        CMP 0311
#FA1F     F0 0A           BEQ FA2B
#FA21     8D 13 03        STA 0313
#FA24     A0 1C           LDY #1C
#FA26     20 7F FA        JSR FA7F
#FA29     B0 22           BCS FA4D
#FA2B     AD 02 C0        LDA C002
#FA2E     8D 12 03        STA 0312
#FA31     AD 05 C0        LDA C005
#FA34     29 20           AND #20
#FA36     D0 17           BNE FA4F
#FA38     20 64 FA        JSR FA64
#FA3B     58              CLI
#FA3C     AD 18 03        LDA 0318
#FA3F     30 FB           BMI FA3C
#FA41     AD 13 03        LDA 0313
#FA44     91 F3           STA (F3),Y
#FA46     C8              INY
#FA47     D0 F3           BNE FA3C
#FA49     E6 F4           INC F4
#FA4B     D0 EF           BNE FA3C
#FA4D     38              SEC
#FA4E     60              RTS
#FA4F     20 64 FA        JSR FA64
#FA52     58              CLI
#FA53     AD 18 03        LDA 0318
#FA56     30 FB           BMI FA53
#FA58     B1 F3           LDA (F3),Y
```

```
#FA5A    8D 13 03       STA 0313
#FA5D    C8             INY
#FA5E    D0 F3          BNE FA53
#FA60    E6 F4          INC F4
#FA62    D0 EF          BNE FA53
#FA64    AC 05 C0       LDY C005
#FA67    78             SEI
#FA68    8C 10 03       STY 0310
#FA6B    AD 80 04       LDA 0480
#FA6E    09 01          ORA #01
#FA70    29 FD          AND #FD
#FA72    8D 80 04       STA 0480
#FA75    8D 14 03       STA 0314
#FA78    A0 00          LDY #00
#FA7A    60             RTS

#FA7B    84 A4 C4 E4    DTA            Data for read / write

#FA7F    98             TYA
#FA80    29 FC          AND #FC
#FA82    0D 1B C0       ORA C01B
#FA85    A8             TAY
#FA86    20 67 FA       JSR FA67
#FA89    20 92 FA       JSR FA92
#FA8C    29 18          AND #18
#FA8E    D0 BD          BNE FA4D
#FA90    18             CLC
#FA91    60             RTS
#FA92    18             CLC
#FA93    58             CLI
#FA94    90 FE          BCC FA94

#FA96    48             PHA            Part of read / write routine
#FA97    A9 40          LDA #40
#FA99    D0 03          BNE FA9E
#FA9B    48             PHA
#FA9C    A9 C0          LDA #C0
#FA9E    8D 0E 03       STA 030E
#FAA1    68             PLA
#FAA2    60             RTS

#FAA3 -  60             RTS            Spare - 93 bytes of blank space
#FAFF    60             RTS
```

BOOT Routine

This routine is only used at bootup, to initialise the computer and copy DOS up from RAM to #E000 - #FFFF.
At boot, the DOS is loaded at #6800 and this routine begins at #8300. It is called via the transfer address in the saved ROM at the end of the boot routine in the interface.

```
#FB00    A9 28          LDA #28
#FB02    8D 57 02       STA 0257       Set screen width to 40 characters
#FB05    A9 50          LDA #50
#FB07    8D 56 02       STA 0256       Set Printer width to 80 characters
#FB0A    78             SEI            Disable interrupts
#FB0B    A9 84          LDA #84        Page to extended ROM
#FB0D    8D 80 04       STA 0480
#FB10    8D 14 03       STA 0314
#FB13    A2 00          LDX #00        Initialise counters for
#FB15    A0 20          LDY #20        number of bytes to copy (#2000)
```

```
#FB17     BD 00 68      LDA 6800,X     Copy DOS from #6800 in RAM
#FB1A     9D 00 E0      STA E000,X     to #E000 in extended ROM
#FB1D     E8            INX
#FB1E     D0 F7         BNE FB17       Loop back to complete a page
#FB20     EE 19 83      INC 8319       Increment high byte of the address for LDA (#FB17)
#FB23     EE 1C 83      INC 831C       Increment high byte of the address for STA (#FB1A)
#FB26     88            DEY
#FB27     D0 EE         BNE FB17       Loop back until all pages done
#FB29     A9 FF         LDA #FF
#FB2B     8D 00 D0      STA D000       Zero the ‘extra command table’ in RipDOS
#FB2E     A9 02         LDA #02        Page back to original ROM
#FB30     20 E6 04      JSR 04E6
#FB33     A9 07         LDA #07        Set PAPER=7 (white)
#FB35     8D E1 02      STA 02E1
#FB38     20 04 F2      JSR F204       PAPER in original ROM
#FB3B     A9 00         LDA #00        INK=0 (black)
#FB3D     8D E1 02      STA 02E1
#FB40     20 10 F2      JSR F210       INK in original ROM
#FB43     A2 34         LDX #34        Set pointers
#FB45     A0 00         LDY #00
#FB47     58            CLI            Enable interrupts
#FB48     4C BD C4      JMP C4BD       Jump to accept line from input buffer
                                       This will automatically load a file called
                                       BOOTUP.COM if there is one on the disk

#FB4B -   60            RTS            Spare - 47 bytes of blank space
#FB79     60            RTS
```

NMI routine

```
#FB7A     48            PHA
#FB7B     AD 81 04      LDA 0481
#FB7E     48            PHA
#FB7F     AD 85 04      LDA 0485
#FB82     48            PHA
#FB83     AD 86 04      LDA 0486
#FB86     48            PHA
#FB87     AD 80 04      LDA 0480
#FB8A     29 FE         AND #FE
#FB8C     8D 80 04      STA 0480
#FB8F     8D 14 03      STA 0314
#FB92     A9 DB         LDA #DB
#FB94     8D 85 04      STA 0485
#FB97     A9 04         LDA #04
#FB99     8D 86 04      STA 0486
#FB9C     A9 02         LDA #02
#FB9E     8D 81 04      STA 0481
#FBA1     20 9F 04      JSR 049F
#FBA4     68            PLA
#FBA5     8D 86 04      STA 0486
#FBA8     68            PLA
#FBA9     8D 85 04      STA 0485
#FBAC     68            PLA
#FBAD     8D 81 04      STA 0481
#FBB0     68            PLA
#FBB1     40            RTI
```

IRQ routine

```
#FBB2     2C 14 03          BIT 0314
#FBB5     30 18             BMI FBCF
#FBB7     AD 80 04          LDA 0480
#FBBA     29 FE             AND #FE
#FBBC     8D 80 04          STA 0480
#FBBF     8D 14 03          STA 0314
#FBC2     68                PLA
#FBC3     68                PLA
#FBC4     68                PLA
#FBC5     AD 10 03          LDA 0310
#FBC8     29 5D             AND #5D
#FBCA     8D FE 04          STA 04FE
#FBCD     58                CLI
#FBCE     60                RTS
#FBCF     48                PHA
#FBD0     8A                TXA
#FBD1     48                PHA
#FBD2     AD 81 04          LDA 0481
#FBD5     48                PHA
#FBD6     AD 85 04          LDA 0485
#FBD9     48                PHA
#FBDA     AD 86 04          LDA 0486
#FBDD     48                PHA
#FBDE     A9 D3             LDA #D3
#FBE0     8D 85 04          STA 0485
#FBE3     A9 04             LDA #04
#FBE5     8D 86 04          STA 0486
#FBE8     A9 02             LDA #02
#FBEA     8D 81 04          STA 0481
#FBED     20 9F 04          JSR 049F
#FBF0     68                PLA
#FBF1     8D 86 04          STA 0486
#FBF4     68                PLA
#FBF5     8D 85 04          STA 0485
#FBF8     68                PLA
#FBF9     8D 81 04          STA 0481
#FBFC     68                PLA
#FBFD     AA                TAX
#FBFE     68                PLA
#FBFF     40                RTI
```

Error message data

```
#FC00     FF                DTA  File         Error message #00 (start)
#FC01     46 69 6C 65       DTA  File         Error message #01
#FC05     20 6E 6F 74 20    DTA  not
#FC0A     66 6F 75 6E E4    DTA found
#FC0F     49 6E 76 61 6C    DTA Inval         Error message #02
#FC14     69 64 20 63 6F    DTA id co
#FC19     6D 6D 61 6E 64    DTA mmand
#FC1E     20 65 6E E4       DTA  end
#FC22     FF                DTA               Error message #03 (none)
#FC23     42 61 64 20 64    DTA Bad d         Error message #04
#FC28     72 69 76 65 20    DTA rive
#FC2D     6E 75 6D 62 65 F2 DTA number
#FC33     49 6E 76 61       DTA Inva          Error message #05
#FC37     6C 69 64 20 66    DTA lid f
#FC3C     69 6C 65 6E 61    DTA ilena
```

```
#FC41     6D E5               DTA me
#FC41     FF                  DTA            Error message #06 (none)
#FC44     49 6C               DTA Il         Error message #07
#FC46     6C 65 67 61 6C      DTA legal
#FC4B     20 61 74 74 72      DTA attr
#FC50     69 62 75 74 E5      DTA ibute
#FC55     57 69 6C 64 63      DTA Wildc      Error message #08
#FC5A     61 72 64 73 20      DTA ards
#FC5F     6E 6F 74 20 61      DTA not a
#FC64     6C 6C 6F 77 65 E4   DTA llowed
#FC6A     46 69 6C 65         DTA File       Error message #09
#FC6E     20 61 6C 72 65      DTA alre
#FC73     61 64 79 20 65      DTA ady e
#FC78     78 69 73 74 F3      DTA xists
#FC7D     49 6E 73 75 66      DTA Insuf      Error message #0A
#FC82     66 69 63 69 65      DTA ficie
#FC87     6E 74 20 64 69      DTA nt di
#FC8C     73 6B 20 73 70      DTA sk sp
#FC91     61 63 E5            DTA ace
#FC94     46 69 6C 65 20      DTA File       Error message #0B
#FC99     6F 70 65 EE         DTA open
#FC9D     FF                  DTA            Error message #0C (none)
#FC9E     FF                  DTA            Error message #0D (none)
#FC9F     FF                  DTA            Error message #0E (none)
#FCA0     4D 69 73 73 69      DTA Missi      Error message #0F
#FCA5     6E 67 20 27 74      DTA ng 't
#FCAA     6F 27 A0            DTA o'
#FCAD     52 65               DTA Re         Error message #10
#FCAF     6E 61 6D 65 64      DTA named
#FCB4     20 66 69 6C 65      DTA file
#FCB9     20 6E 6F 74 20      DTA not
#FCBE     6F 6E 20 73 61      DTA on sa
#FCC3     6D 65 20 64 69      DTA me di
#FCC8     73 EB               DTA sk
#FCCA     FF                  DTA            Error message #11 (none)
#FCCB     54 61               DTA Ta         Error message #12
#FCCD     72 67 65 74 20      DTA rget
#FCD2     64 72 69 76 65      DTA drive
#FCD7     20 6E 6F 74 20      DTA not
#FCDC     73 6F 75 72 63 65   DTA source
#FCE2     20 64 72 69 76 E5   DTA  drive
#FCE8     44 65 73            DTA Des        Error message #13
#FCEB     74 69 6E 61 74      DTA tinat
#FCF0     69 6F 6E 20 6E      DTA ion n
#FCF5     6F 74 20 73 70      DTA ot sp
#FCFA     65 63 69 66 69      DTA ecifi
#FCFF     65 E4               DTA ed
#FD01     FF                  DTA            Error message #14 (none)
#FD02     FF                  DTA            Error message #15 (none)
#FD03     FF                  DTA            Error message #16 (none)
#FD04     FF                  DTA            Error message #17 (none)
#FD05     FF                  DTA            Error message #18 (none)
#FD06     FF                  DTA            Error message #19 (none)
#FD07     44 69               DTA Di         Error message #1A
#FD09     73 6B 20 77 72      DTA sk wr
#FD0E     69 74 65 20 70      DTA ite p
#FD13     72 6F 74 65 63      DTA rotec
#FD18     74 65 E4            DTA ted
#FD1B     49 6E               DTA In         Error message #1B
#FD1D     63 6F 6D 70 61      DTA compa
#FD22     74 69 62 6C 65 20   DTA tible
#FD28     64 72 69 76 65 F3   DTA drives
```

```
#FD2E     46 69 6C              DTA Fil          Error message #1C
#FD31     65 20 6E 6F 74        DTA e not
#FD36     20 6F 70 65 EE        DTA open
#FD3B     46 69 6C 65 20        DTA File         Error message #1D
#FD40     65 6E E4              DTA end
#FD43     60 60 60 60 60 60     DTA              Space for more messages
#FD49     60 60 60 60 60 60     DTA
#FD4F     60 60 60 60 60        DTA
#FD54     60 60 60 60 60        DTA
#FD59     60 60 60 60 60        DTA
#FD5E     60 60 60 60 60        DTA
#FD63     60 60 60 60 60        DTA
#FD68     60 60 60 60 60        DTA
#FD6D     60 60 60 60 60        DTA
#FD72     60 60 60 60 60        DTA
#FD77     60 60 60 60 60        DTA
#FD7C     60 60 60 60 60        DTA
#FD81     60 60 60 60 60        DTA
#FD86     60 60 60 60 60        DTA
#FD8B     60 60 60 60 60        DTA
```

System message data

```
#FD90     FF                    DTA Load         System message #00 (none)
#FD91     4C 6F 61 64           DTA Load         System message #01
#FD95     69 6E 67 2E 2E A0     DTA ing..
#FD9B     53 61 76 69           DTA Savi         System message #02
#FD9F     6E 67 2E 2E A0        DTA ng..
#FDA4     20 28 59 2F 4E        DTA  (Y/N        System message #03
#FDA9     29 3A A0              DTA ):
#FDAC     4C 6F  61 64 20 73    DTA Load s       System message #04
#FDB2     6F 75 72 63 65 20     DTA ource
#FDB8     64 69 73 6B 20        DTA disk
#FDBD     6F 6E 20 64 72        DTA on dr
#FDC2     69 76 65 A0           DTA ive
#FDC6     4C 6F 61 64 20 73     DTA Load s       System message #05
#FDCC     6F 75 72 63 65        DTA ource
#FDD1     20 64 69 73 6B        DTA disk
#FDD6     20 61 6E 64 20        DTA and
#FDDB     70 72 65 73 73        DTA press
#FDE0     20 52 45 54 55        DTA RETU
#FDE5     52 4E A0              DTA RN
#FDE8     4C 6F 61 64 20 74     DTA Load t       System message #06
#FDEE     61 72 67 65 74 20     DTA arget
#FDF4     64 69 73 6B 20        DTA disk
#FDF9     61 6E 64 20 70        DTA and p
#FDFE     72 65 73 73 20 52     DTA ress R
#FE04     45 54 55 52 4E A0     DTA RETURN
#FE0A     4C 6F 61              DTA Loa          System message #07
#FE0D     64 20 64 69 73        DTA d dis
#FE12     6B 73 20 66 6F        DTA ks fo
#FE17     72 20 62 61 63        DTA r bac
#FE1C     6B 75 70 20 66        DTA kup f
#FE21     72 6F 6D 20 A3        DTA rom #
#FE26     20 54 6F 20 A3        DTA  To #        System message #08
#FE2B     0D 0A 61 6E 64        DTA and          System message #09
#FE30     20 70 72 65 73        DTA pres
#FE35     73 20 52 45 54        DTA s RET
#FE3A     55 52 4E A0           DTA URN
#FE3E     42 61 63 6B 75 70     DTA Backup       System message #0A
```

```
#FE44     20 63 6F 6D 70        DTA comp
#FE49     6C 65 74 65 AE        DTA lete.
#FE4E     44 69 73 6B 20        DTA Disk          System message #0B
#FE53     65 72 72 6F F2        DTA error
#FE58     20 44 72 69 76 E5     DTA  Drive        System message #0C
#FE5E     20 54 72 61 63 EB     DTA  Track        System message #0D
#FE64     20 73 65              DTA   se          System message #0E
#FE67     63 74 6F F2           DTA ctor
#FE6B     44 69 72 65 63 74     DTA Direct        System message #0F
#FE71     6F 72 79 20 6F        DTA ory o
#FE76     66 20 64 72 69        DTA f dri
#FE7B     76 65 A0              DTA ve
#FE7E     20 46 69 6C 65 73     DTA  Files        System message #10
#FE84     20 A0                 DTA
#FE86     20 42 6C 6F           DTA  Blo          System message #11
#FE8A     63 6B 73 20 66        DTA cks f
#FE8F     72 65 65 A0           DTA ree
#FE93     20 57 72 69 74 65     DTA  Write        System message #12
#FE99     20 70 72 6F 74        DTA prot
#FE9E     65 63 74 65 E4        DTA ected
#FEA3     43 72 65 61 74 65 E4  DTA Created       System message #13
#FEAA     41 6C 72              DTA Alr           System message #14
#FEAD     65 61 64 79 20        DTA eady
#FEB2     65 78 69 73 74 F3     DTA exists
#FEB8     4F 76 65 72 77        DTA Overw         System message #15
#FEBD     72 69 74 74 65 EE     DTA ritten
#FEC3     4C 6F 61              DTA Loa           System message #16
#FEC6     64 20 64 69 73        DTA d dis
#FECB     6B 20 6F 6E 20        DTA k on
#FED0     64 72 69 76 65 A0     DTA drive
#FED6     46 6F 72 6D           DTA Form          System message #17
#FEDA     61 74 74 69 6E        DTA attin
#FEDF     67 20 63 6F 6D        DTA g com
#FEE4     70 6C 65 74 E5        DTA plete
#FEE9     2C 20 53 69 6E        DTA , Sin         System message #18
#FEEE     67 6C 65 2D 73        DTA gle-s
#FEF3     69 64 65 E4           DTA ided
#FEF7     2C 20 44 6F 75 62     DTA , Doub        System message #19
#FEFD     6C 65 2D 73 69        DTA le-si
#FF02     64 65 E4              DTA ded
#FF05     20 44 53 54 45 50     DTA  DSTEP        System message #1A
#FF0B     20 20 20 AD           DTA    -
#FF0F     20 6D F3              DTA ms            System message #1B
#FF12     41 55 54 CF           DTA AUTO          System message #1C
#FF16     60 60 60 60 60        DTA               Space for more messages
#FF1B     60 60 60 60 60        DTA               RTS in case called in error
#FF20     60 60 60 60 60        DTA
#FF25     60 60 60 60 60        DTA
#FF2A     60 60 60 60 60        DTA
#FF2F     60 60 60 60 60        DTA
#FF34     60 60 60 60 60        DTA
#FF39     60 60 60 60 60        DTA
#FF3E     60 60                 DTA
```

Table of keywords

```
#FF40     4C 4F 41 44 00        DTA LOAD
#FF45     53 41 56 45 00        DTA SAVE
#FF4A     44 49 52 00           DTA DIR
#FF4E     44 45 4C 00           DTA DEL
```

```
#FF52    44 52 56 00           DTA DRV
#FF56    52 45 4E 00           DTA REN
#FF5A    42 41 43 4B 55 50 00  DTA BACKUP
#FF61    43 4F 50 59 00        DTA COPY
#FF66    4F 50 45 4E 00        DTA OPEN
#FF6B    43 4C 4F 53 45 00     DTA CLOSE
#FF71    BE 00                 DTA GET
#FF73    50 55 54 00           DTA PUT
#FF77    8D 4D 41 54 00        DTA FORMAT
#FF7C    53 54 41 54 00        DTA STAT
#FF81    53 45 54 00           DTA SET
#FF85    44 CB 00              DTA DSTEP
#FF88    50 52 4F 54 00        DTA PROT
#FF8D    82 00                 DTA STORE
#FF8F    83 00                 DTA RECALL
#FF91    4E 41 4D 45 00        DTA NAME
#FF96    95 00                 DTA READ
#FF98    BF 00                 DTA CALL
#FF9A    4C 44 49 52 00        DTA LDIR
#FF9F    FF                    DTA              End of table marker
#FFA0    60 60 60 60 60 60     DTA              Space for 6 more commands
#FFA6    60 60 60 60 60 60     DTA
#FFAC    60 60 60 60 60        DTA              (RTS in case called in error)
#FFB1    60 60 60 60 60        DTA
#FFB6    60 60 60 60 60        DTA
#FFBB    60 60 60 60 60        DTA
```

Address Table

Addresses for DOS routines (1 less than the address for the routine as it is incremented by the RTS)

```
#FFC0    02 E1                 DTA              To give #E103 for !LOAD
#FFC2    91 E2                 DTA              To give #E292 for !SAVE
#FFC4    B7 E3                 DTA              To give #E3B8 for !DIR
#FFC6    7D E4                 DTA              To give #E47E for !DEL
#FFC8    CC E5                 DTA              To give #E5CD for !DRV
#FFCA    61 E5                 DTA              To give #E562 for !REN
#FFCC    F2 E5                 DTA              To give #E5F3 for !BACKUP
#FFCE    37 E7                 DTA              To give #E738 for !COPY
#FFD0    88 EA                 DTA              To give #EA89 for !OPEN
#FFD2    02 EB                 DTA              To give #EB03 for !CLOSE
#FFD4    F1 EB                 DTA              To give #EBF2 for !GET
#FFD6    7C EB                 DTA              To give #EB7D for !PUT
#FFD8    C9 EC                 DTA              To give #ECCA for !FORMAT
#FFDA    B0 EE                 DTA              To give #EEB1 for !STAT
#FFDC    3C EF                 DTA              To give #EF3D for !SET
#FFDE    0A F0                 DTA              To give #F00B for !DSTEP
#FFE0    A1 EF                 DTA              To give #EFA2 for !PROT
#FFE2    36 E9                 DTA              To give #E937 for !STORE
#FFE4    08 EA                 DTA              To give #EA09 for !RECALL
#FFE6    2B F0                 DTA              To give #F02C for !NAME
#FFE8    FF F2                 DTA              To give #F300 for !READ
#FFEA    67 F3                 DTA              To give #F368 for !CALL
#FFEC    49 F0                 DTA              To give #F04A for !LDIR
#FFEE    60 60                 DTA              Spare
#FFF0    60 60                 DTA              Spare
#FFF2    60 60                 DTA              Spare
#FFF4    60 60                 DTA              Spare
#FFF6    60 60                 DTA              Spare
#FFF8    60 60                 DTA              Spare
#FFFA    7A FB                 DTA              NMI address
```

```
#FFFC     00 00          DTA          RST address
#FFFE     B2 FB          DTA          IRQ address
```

Version history for RipDOS

These pages explain the changes that I have made to CUMANA.DOS V1.3 as supplied with my 5.25 inch disc drive. Each version is the same as the previous version, with the changes described.

RipDOS V2.0

1 This is CUMANA DOS V1.3, re organised in memory, without any added features

E012	6812	Entry point for command interpreter
E103	6913	Command routines
F400	7C00	General subroutines
F980	8180	Disk read / write
FB00	8300	DOS copy up routine
FB7A	837A	NMI routine
FBB2	83B2	IRQ routine
FC00	8400	Error messages
FD90	8590	System messages
FF40	8740	Command word table
FFC0	87C0	Routine address look up table
FFFA	87FA	NMI / RESET / IRQ vectors

All of the blocks except NMI & IRQ have spare capacity - for expansion.

RipDOS V2.1

2 Patch to page out the DOS before auto running a machine code program.
(2a in V2.7 modifies this again)

3 !LOAD routine changed to display the Transfer address (if present) even if ,A or ,N switches also used.

RipDOS V2.2

4 !STORE added to work as per CUMANA manual. Works for real. integer and string arrays.

5 !RECALL also added to work as per manual

RipDOS V2.3

6 Start-up routine changed to give black ink on white paper.

1a Also realised that a file called 'BOOTUP.COM' will auto-load if present on the disk, and that command files of this type can be accessed at any time through the DOS by using !FILENAME to run FILENAME.COM. E.g. !BOOTUP will run the file BOOTUP.COM

RipDOS V2.4

7 !STAT re-written to give the onboard system information, including the DSTEP value.

7a !STAT 0 added to give the same information, but from the system track on the disk in default drive

8 !SET d,t,s works as per the CUMANA manual, but changes only the onboard information

8a !SET <Return> copies out the onboard system information to the default drive, including the DSTEP value

9 !DSTEP re-written and moved. It works as per CUMANA manual, but changes only the onboard information. Use !SET to update the system information on the disk in the default drive.

10 !NAME added as a new disc command. It is used to give a name to (or change the name of) a disc. The syntax is

!NAME “diskname” <Return>

Disksname can be up to 9 characters, and can include the drive number (as in !FORMAT)

RIPDOS V2.5

11 !READ command added to enable a single disk sector to be read in. The syntax is:

! READ d,t,s,a <RETURN>

where d is the disc drive number, t is the track number (0 to 39 / 79), s is the sector number (1 to 16), and a is the address at which to load the block (256 bytes will be loaded). The routine sits in DOS between #F300 and #F360 (#7B00 to 7B60). It has not been tested on a double sided disk drive.

RIPDOS V2.6

2 !CALL command added to enable a machine code routine sitting in the shadow ROM area to be called directly from BASIC. The syntax is:

!CALL address followed by any other information required by the routine called.

!CALL works just like CALL, but has the shadow ROM #C000 to #FFFF paged in. On exit (RTS), the main ROM is paged back in again. The address called can be any from #0000 to #FFFF, not necessarily above #C000.

RipDOS V2.7

2a In V2.1, a patch was added to page out the DOS before auto running a machine code program. In V2.7, the DOS is NOT paged out if the auto start (Transfer) address is in the shadow ROM area (#C000 to #FFFF). If the T address is in normal RAM, or there is no T address specified, then the DOS is still paged out.

13 A routine has also been added so that an additional table of commands and their code can be installed in an unused area of memory from #D000 to #DFFF

The facility, which allows a new set of commands to be added, works through the normal command interpreter routine in the DOS. If an unrecognised command is encountered, the routine looks to see if there is another look up table in page #D000. If it finds a match there, it will run the routine. If it does not find a match, it will come up with an error message.

The command word table MUST start at #D000 and each entry must be separated with a null (0). Don't forget that some words may be partly or completely tokenised. For example, FORGET will be represented by #8D (FOR) #BE (GET) #00 (null to separate).

#D000 to #D0FF is available for the new table of words, which MUST end with #FF after the null of the last entry.

Memory #D0C0 to D0FF is reserved for the corresponding table of addresses for the new routines. Each address is 2 bytes long with the low byte first. There must be a 2 byte entry for each new command word in the first part of the table. Each address must be 1 less than the starting address for its routine, since 1 is added when the routine is called using the RTS instruction.

Memory #D100 to DFFF is available for the routines themselves. This is in an unused area of DOS memory, and provides huge flexibility for providing your own set of additional utilities without using RAM.

RipDOS V2.8

3a In V2.1, a patch was added so that the ,D option with !LOAD displays the Transfer address, even if ,A or ,N was also specified. This is retained, and a message ‘AUTO’ is displayed for the Transfer address if the file is BASIC. A new line is also printed.

14 The PRINT CHARACTER routines have all been rationalised to use ROM #F598 to print a new line and ROM #CCD9 to print the accumulator. This allows output to be directed to a printer if required.

15 In CUMANA.DOS, Loading ..... / Saving ..... messages were not displayed if A9 contains zero. i.e. if the current line number is less than 256! The test is removed, so that the message is always displayed.

16 !STORE and !RECALL now give Loading ... / Saving ...... messages.

17 !LDIR added as a new command. Its use and syntax is the same as !DIR, but output is directed to a printer. Routine address #F04A to #F057 (#784A to 7857).

Appendix 1   RipDOS use of ROM area #C000 - #DFFF

Hex Address in ROM	Use
C000	Drive number in use
C001	Track counter
C002	Sector counter
C003	Pointer
C004	Pointer
C005	Read / write flag. #A0 = Write, #80 = Read
C006	
C007	Save stack pointer whilst in DOS ROM
C008	
C009	
C00A	
C00B	
C00C	Default drive number. Set by !DRV
C00D	
C00E	
C00F	
C010	
C011	
C012	
C013	For drive 0 - 0 if no drive, 40 for 40 track, 80 for 80 track
C014	For drive 1 - 0 if no drive, 40 for 40 track, 80 for 80 track
C015	For drive 2 - 0 if no drive, 40 for 40 track, 80 for 80 track
C016	For drive 3 - 0 if no drive, 40 for 40 track, 80 for 80 track
C017	For drive 0 - tracks on side 2. 0 for single sided, 40 or 80
C018	For drive 1 - tracks on side 2. 0 for single sided, 40 or 80
C019	For drive 2 - tracks on side 2. 0 for single sided, 40 or 80
C01A	For drive 3 - tracks on side 2. 0 for single sided, 40 or 80
C01B	DSTEP value. 0, 1, 2, 3 for 6, 12, 20, 30 mS
C01C	
C01D	
C01E	
C01F	
C020	
C021	
C022	
C023	
C024	
C025	SAVE default #FF
C026	SAVE default 0

Hex Address in ROM	Use
C027	Start address for SAVE - 2 bytes
C028	
C029	End address for SAVE - 2 bytes
C02A	
C02B	Transfer address for code. 1=basic. 2=basic, Auto
C02C	0=basic
C02D	
C02E	Buffer for data going out on SAVE - to #C122
C02F	
C030	
C031	
C032	
C033	System information
C034	System information
C035	System information
C036	System information
C037	System information
C038	System information
C039	System information
C03A	System information
C03B	Disk name
C03C	Disk name
C03D	Disk name
C03E	Disk name
C03F	Disk name
C040	Disk name
C041	Disk name
C042	Disk name
C043	Disk name
C122	End of buffer for data going out on SAVE
C123	System information
C124	System information
C125	System information
C126	System information
C127	System information - No of blocks free in !DIR
C128	System information - No of blocks free in !DIR
C129	System information
C12A	System information
C12B	Drive number
C12C	Filename
C12D	Filename
C12E	Filename

Hex Address in ROM	Use
C12F	Filename
C130	Filename
C131	Filename
C032	Filename extension
C133	Filename extension
C134	Filename extension
C135	Number of sectors taken by file - 2 bytes
C136	
C137	Sector - first block
C138	Track - first block
C139	Sector - final block
C13A	Track - final block
C13B	0 for not protected, #80 for protected
C13C	
C13D	!PROT - 0 for N, #80 for P, #C0 for I
C13E	!PROT - #3F for N, P and I
C13F	
C140	Miscellaneous counter and temporary storage area
C141	Miscellaneous counter and temporary storage area
C142	Name of disk / file
C143	Name of disk / file
C144	Name of disk / file
C145	Name of disk / file
C146	Name of disk / file
C147	Name of disk / file
C148	Name of disk / file
C149	Name of disk / file
C14A	Name of disk / file
C14B	Start address for LOAD - 2 bytes
C14C	
C14D	Transfer address for LOAD if specified - 2 bytes
C14E	
C14F	D flag for LOAD - #44 gives display
C150	A / N flag for LOAD
C151	
C152	
C153	
C154	
C155	
C156	
C157	
C158	

Hex Address in ROM	Use
C159	Flag for file being open for reading (GET). 0 = not open
C15A	Counter for tracking GET data
C15B	Flag for file being open for writing (PUT). 0 = not open
C15C	Counter for tracking PUT data
C15D - C179	
C17A	Drive number
C17B	Disk name / filename
C17C	Disk name / filename
C17D	Disk name / filename
C17E	Disk name / filename
C17F	Disk name / filename
C180	Disk name / filename
C181	Disk name / filename
C182	Disk name / filename
C183	Disk name / filename
C184	Drive number
C185	Filename for COPY TO
C186	Filename for COPY TO
C187	Filename for COPY TO
C188	Filename for COPY TO
C189	Filename for COPY TO
C18A	Filename for COPY TO
C18B	Filename for COPY TO
C18C	Filename for COPY TO
C18D	Filename for COPY TO
C18E	
C18F	COPY flag. #4E = N, #50 = P, #80 = same (default)
C190	COPY flag. #4F = O. Default #80
C191	COPY flag. Single drive option. #43 for C
C192 - C1B2	
C1B3	Track used in COPY
C1B4	Sector used in COPY
C1B5	Displacement used in COPY
C1B6	
C1B7	
C1B8	
C1B9	
C1BA	Track of next block to look for in COPY
C1BB	Sector of next block to look for in COPY
C1BC - C1FF	
C200	Page reserved for !PUT
C300	Page reserved for !GET

Hex Address in ROM	Use
C400 - CFFF	Unused?
D000 - DFFF	Empty - available for user defined routines

Appendix 2 RipDOS use of ROM area #E000 - #FFFF

Hex Address in ROM	Hex for sub	Routine /sub routine	Spare bytes
E000		Message RipDOS V2.7	
E012	E022	Try to match primary command word	
E012	E05D	Try to match secondary command word	
E012	E098	No match for command word	
E012	E0C2	Data	
E012		Command interpreter - entry point from page 4	
E0D2		Patch for m/c files which auto run after loading	
E103		!LOAD	
E292		!SAVE	
E3B8		!DIR	
E47E		!DEL	
E562		!REN	
E5CD		!DRV	
E5F3		!BACKUP	
E738		!COPY	
E937	E98D	Store and Recall syntax and set-up	
E937		!STORE	
EA09		!RECALL	
EA89	EAA0	Open to read	
EA89	EAC6	Open to write	
EA89		!OPEN	
EB03		!CLOSE	
EB7D		!PUT	
EBF2		!GET	
ECCA	EE81	Data bytes for format	
ECCA		!FORMAT	
EEB1		!STAT	
EF3D		!SET	
EFA2		!PROT	
F00B	F028	Data	
F00B		!DSTEP	
F02C		!NAME	
F04A		!LDIR	
F058		Spare	936
F300		!READ	
F368		!CALL	
F371		Spare	143
F400		Read / Write	
F466		Set up filename in page C1	

Hex Address in ROM	Hex for sub	Routine /sub routine	Spare bytes
F4BE		Read filename from text into $ buffer	
F515		Copy filename down	
F527		Check for wildcards if not allowed	
F539		Print out message and filename from buffer	
F546		Print out filename from buffer	
F560		Print out message and filename	
F56D		Print out filename to screen	
F586		Get RETURN	
F598		Print CR and LF	
F59F		Print accumulator to screen	
F5AB		Print 2 spaces	
F5AE		Print 1 space	
F5B3		Dispose of comma and fetch next character	
F5C2		Look for file of specified name in directory	
F5C5		Same but does not load system track first for directory	
F5F0		Carry on looking in same directory	
F602		Set up first sector for save	
F669		Set up sectors for save	
F6B1		Update directory after save	
F6D1		Update system track after save	
F6E2		Read in system information	
F6F1		Read in system information	
F6F5		Read in system information	
F702		Error message / system message routine	
F729		System message handler	
F755		Print out single byte number in decimal form	
F757		Print out 2 byte number in decimal form	
F78C		Print byte as 2 ASCII in hex	
F7A3		Original ROM call handler	
F7DA		Set up filename	
F7F3		Read in system sector	
F7FB		Display Saving .... for Store	
F80B		Display A,E,T addresses for code ,D switch	
F842		Display Loading ... for Recall	
F855		Spare	299
F980		To / from disk	
FAA3		Spare	93
FB00		Routine to copy ROM up from RAM to #E000 - #FFFF	
FB4B		Spare	47
FB7A		NMI routine	
FBB2		IRQ routine	

Hex Address in ROM	Hex for sub	Routine /sub routine	Spare bytes
FC00		Error message data + space	
FD90		System message data + space	
FF40		Table of keywords for command interpreter + space	
FFC0		Addresses for corresponding routines + space	
FFFA		NMI address	
FFFC		RST address	
FFFE		IRQ address	

Appendix 3 List of Error Messages

Error message #01	File not found
Error message #02	Invalid command end
Error message #03	none
Error message #04	Bad drive number
Error message #05	Invalid filename
Error message #06	none
Error message #07	Illegal attribute
Error message #08	Wildcards not allowed
Error message #09	File already exists
Error message #0A	Insufficient disk space
Error message #0B	File open
Error message #0C	None
Error message #0D	None
Error message #0E	None
Error message #0F	Missing ‘to’
Error message #10	Renamed file not on same disk
Error message #11	None
Error message #12	Target drive not source drive
Error message #13	Destination not specified
Error message #14	None
Error message #15	None
Error message #16	None
Error message #17	None
Error message #18	None
Error message #19	None
Error message #1A	Disk write protected
Error message #1B	Incompatible drives
Error message #1C	File not open
Error message #1D	File end

Appendix 4 List of System Messages

System message #01	Loading..
System message #02	Saving..
System message #03	(Y/N):
System message #04	Load source disk on drive
System message #05	Load source disk and press RETURN
System message #06	Load target disk and press RETURN
System message #07	Load disks for backup from #
System message #08	To #
System message #09	and press RETURN
System message #0A	Backup complete.
System message #0B	Disk error
System message #0C	Drive
System message #0D	Track
System message #0E	sector
System message #0F	Directory of drive
System message #10	Files
System message #11	Blocks free
System message #12	Write protected
System message #13	Created
System message #14	Already exists
System message #15	Overwritten
System message #16	Load disk on drive
System message #17	Formatting complete
System message #18	, Single-sided
System message #19	, Double-sided
System message #1A	DSTEP -
System message #1B	ms
System message #1C	AUTO

Appendix 5 Data on Cumana disk System Track

Byte	Use
0	Drive 0, side 1 - Number of tracks (0, 40, 80)
1	Drive 1, side 1
2	Drive 2, side 1
3	Drive 3, side 1
4	Drive 0, side 2
5	Drive 1, side 2
6	Drive 2, side 2
7	Drive 3, side 2
8	DSTEP value
9 - F	
10	Sector for next available sector
11	Track for next available sector
12	Sector pointer to first directory
13	Track pointer to first directory
14 - 15	Number of blocks free on disk - 2 bytes
16 - 17	Number of blocks used on disk - 2 bytes
18 - 20	Disk name
21 - on	Not used

Appendix 6   Data on Cumana disk Directory Track

Byte	Use
0	Track - pointer to next directory (0 if no more)
1	Sector - pointer to next directory (0 if no more)
2	Number of files in directory (max = 15)
3 - 8	Filename - null if no file in slot
9 - B	Filename extension
C - D	Number of sectors taken by file - 2 bytes
E	Sector of first block of program
F	Track of first block of program
10	Sector of last block of program
11	Track of last block of program
12	P, N, I status
13 - 22	Repeat sequence in 16 byte blocks
etc.	Same format for data files, including STORE

Appendix 7   Data on Cumana disk File Tracks

Byte	Use
	First block (program, code, memory created by !SAVE)
0	Track pointer to next block
1	Sector pointer to next block
2	#FF = suitable for !LOAD
3	00
4 - 5	Start address for !LOAD - 2 bytes
6 - 7	End address for !LOAD - 2 bytes
8	T address / program type - 2 bytes:
9	0000 = code no T, 0001 = basic, 0002 = basic AUTO, ABCD = code T address
A	Number of bytes in this block. #F5 = full
B	Data bytes from here on
	Subsequent blocks created by !SAVE
0	Track pointer to next block. Null if no more
1	Sector pointer to next block. Null if no more
2	Number of bytes in this block. #FD = full
3	Data bytes from here on
	Blocks created by !OPEN / !PUT
0	Track pointer to next block. Null if no more
1	Sector pointer to next block. Null if no more
2	Data - for single byte data. For $ - length of $ (e.g. 3) followed by string
3	Data - for single byte data. For $ - string character
4	Data - for single byte data. For $ - string character
5	Data - for single byte data. For $ - string character (last)
6	Data - for single byte data. For $ - length of $ followed by string
7	Data - for single byte data. For $ - string character
8	etc.
	Blocks created by !STORE
0	Track pointer to next block. Null if no more
1	Sector pointer to next block. Null if no more
2 - 3	End address + 1 from which data was saved - 2 bytes
4 - 5	Start address from which array was saved - 2 bytes
6	Bit 7 set for % integers
7	#FF for strings
8	Array block copied from memory. For $ - Length of $ followed by $.