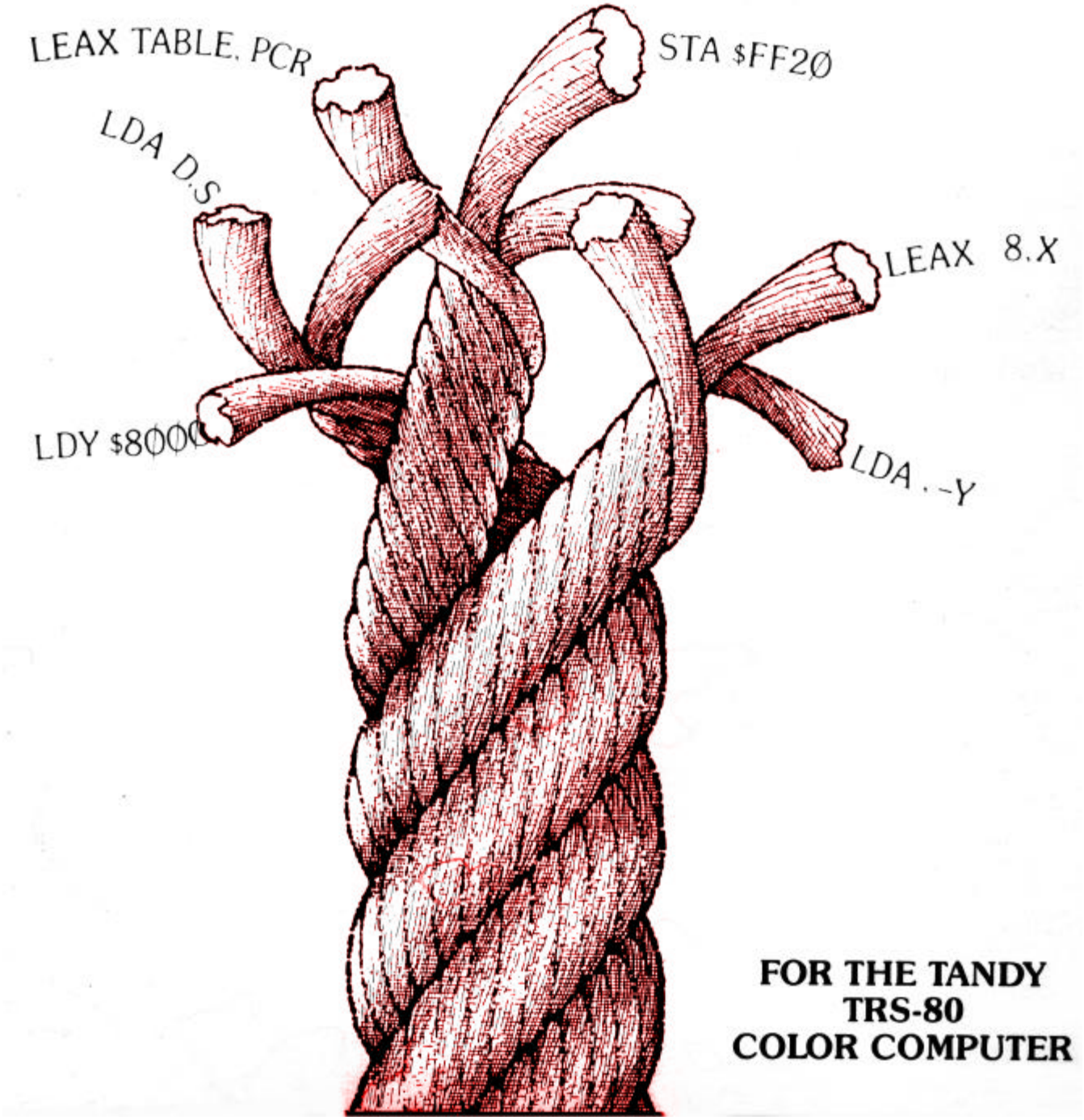
DISK BASIC UNRAVELLED II



FOR THE TANDY TRS-80 COLOR COMPUTER

TABLE OF CONTENTS

APPENDICES

FOREWORD

Due to the many requests for the Unravelled Series produced by Spectral Associates, and the fact that these books are rare and no longer in production, I have taken it upon myself to reproduce them in electronic .PDF (Adobe Acrobat®) format.

I have re-disassembled the ROMs listed in this book, and added all the comments from the Original Extended Basic Unravelled Book. Some changes were made to make the book a little easier to read.

1. The comments have been cleaned up some. In cases where a comments continued onto the next line, a * is placed in the Labels column, as well as a * at the beginning of each line of the comment. In cases where the previous comment used this format, a = was used. This was done in the original, but not all comments stuck to this format.
2. I have renumbered all the line numbers. Each Appendix (with code) starts at Line ØØØ1.
3. Some spell checking, and context checking was done to verify accuracy.
4. I used the Letter Gothic MT Bold Font. This allows for display of Slashed Zeros. I thought it important to be able to distinguish between Ø and O.
5. All the Hex code now shows the Opcodes.

There were other minor changes that were made to make viewing a little better. If any discrepancies arise, please let me know so that I may correct the errors. I can be contacted at: mailto:wzydhek@internetcds.com

Special Thanks to Jean-François Morin for pointing out those Oops to me. I d like to also thank those who have either given me, or loaned me their copy of the original Unravelled Series.

About Me

My name is Walter K. Zydhek. I ve been a Computer Hobbyist since 1984 when I received my 1st Tandy Color Computer 2 for Christmas. It had 32K of ram, Cassette, and one Cartridge. I quickly learned to program in Basic and then moved into Assembly.

Over the next few years, I saved to purchase the Multi-Pak Interface, Disk Drives, Modem, OS-9, and various Odds and Ends.

I moved to Tampa Florida and in the move, My CoCo was damaged. I then replaced it with the CoCo 3. WOW what a difference. I added the 512K Ram Upgrade, A CM-8 color monitor, and joined the Carolwood CoCo Club. (Thanks Jean-François for reminding me of the name.)

I had a couple of close friends that helped me explore the world of CoCo and by this time, I knew that my CoCo would be my friend forever. I give special thanks to Steve Cohn, who helped me get started with ADOS. Two other people whose names I can t remember were very beneficial to my mastering of the CoCo.

Shortly after getting my CoCo 3, I started BBS ing. Wow, a whole new world. My knowledge just kept growing.

A few years later, I moved to Oregon, then to Phoenix, Arizona to attend school. I studied Electronics Technology at Phoenix Institute of Technology. In the second year, we studied Micro-processor Theory. For our labs, we just happen to use the Tandy Color Computer 3 (for studying 68Ø9 Processors). I had it made. In this class I added an EPROM programmer/reader to my list of hardware. My favorite instructor, Gary Angle & I spent many hours sharing information on the CoCo. At one time, we shared a joint project to disassemble ROMs from industrial machinery, which used the 68Ø9 Processor. Using the CoCo to read the ROMs to work with.

I even had a BBS running under OS-9 at one time. RiBBS I think it was. Very similar to QuickBBS and RemoteAccess BBS for the PC.

In 1991, I finally converted over to PC, but never forgetting my CoCo. About 5 years ago, My CoCo and all related material was stolen from me. And the CoCo world was just a memory.

In the last 2 Years, my love for the CoCo has re-kindled. I have been partially content to use a CoCo Emulator for my PC. I tried the CoCo 2 Emulator by Jeff Vavasour. This was OK, but a lot was left out. I then purchased the CoCo 3 Emulator. Much better, but would not use Double Sided Disks . Although it did have a Virtual Hard Drive for use in OS-9.

I then wanted to better the CoCo Emulator, add use of PC hardware, Add Double Sided Disk functionality, and even make it Windows Native, instead of a Dos Box. Unfortunately I could not get the source code for the CoCo 3 Emulator.

I then turned to Paul Burgin s Dragon 2/Coco 2 Emulator. This had source code available and with a small $2Ø.ØØ donation, was able to get the source code to additional portions of his program. I have tinkered with it, but came to understand that I needed more info on the CoCo. I have looked all over the net and found quite a lot of useful information, but what I really needed was the Unravelled Series.

I was able to find someone that had Extended Basic Unravelled and Disk Basic Unravelled (He sent them to me for free). And a friend of mine had Super Extended Basic Unravelled (A copy I gave him years ago). Unfortunately, the books are not in the best of shape, and the type is hard to read, and with so many people looking for the books, I decided to re-do them in Electronic format.

I ask everyone that obtains copies of this electronic document to PLEASE give freely. These books are for educational/informational use only. These books are no longer in publication and Spectral Associates no longer in business. Do not use these books for financial gain, as that would most certainly abuse the Copyright Laws that I have already bruised by re-producing them.

Other than that, enjoy the books!! I ll add more information to them as I get it. I plan on adding more Memory Map information, as well as hardware info in the coming months. But for now, take advantage of this fine resource.

Walter K. Zydhek

INTRODUCTION

Disk Basic Unravelled will provide the reader with a complete detailed and fully commented assembly listing of the Disk Operating System (DOS) of Radio Shack s COLOR BASIC. It is not within the scope of this book to teach the neophyte how to construct a DOS or to be able to completely understand the COLOR DOS. The reader will need to have a basic knowledge of 68Ø9 assembly language programming to be able to take full advantage of the opportunities, which this book presents. It is also assumed that the reader is familiar with the contents of the Disk Basic Users manual, which contains a general description of the overall operation of Disk Basic and much useful information concerning the physical and logical format of the tracks & sectors. Disk Basic Unravelled will allow the reader to be able to completely understand the theory behind COLOR DOS to the point of being able to modify it for his own purposes or add extra commands or functions to the DOS.

No attempt will be made to re explain the functions of BASIC and Extended Basic, which were explained in the previous two books of the BASIC Unravelled series. The reader should be aware of the fact that Color Disk Basic is not a stand-alone system. There are many direct calls into the Basic and Extended Basic ROMs. These calls are not explained in this book; the reader will have to refer to the Basic and Extended Basic Unravelled books in order to get a full explanation of these ROM calls. A complete memory map of the system operating variables is given at the beginning of the DOS assembly listing and a symbol table showing the location of all referenced routines and tables is at the end of the listing.

All of the ROMs used in the Color Computer have undergone revisions since the inception of the machine. The disk ROMs have undergone the most severe change of the three ROMs. The first disk RON (Revision 1.Ø) used only 6K of the available 8K RON space and the second disk RON (Revision 1.1) used approximately 6.5K of ROM with the majority of the .5K increase going to correct bugs in the first ROM and to add the DOS command to Disk Basic. That leaves 1.5K of free ROM space in the latest version of Disk Basic, which is available to the user if he has a 64K machine. It is not recommended that this free ROM space be permanently allocated by any user since the Disk Basic ROMs in the Dragon computer (a British clone of the Color Computer) use the entire 8K ROM space and have added several new disk BASIC commands. This means that the commands are also probably available to Radio Shack and version 1.2 of the BASIC ROM, which may contain some of these commands, will be coming along sometime.

The new revisions of the Color Basic and Extended Basic ROMs kept the majority of the code in the same position in the ROM. In the case of the Disk Basic revisions, however, no effort was made to keep the code in the same position. There are two reasons for this: the first is that there were so many changes that it would have been very difficult to maintain the position and secondly, there was 2K of additional ROM space available so why try to maintain the position. The total positional difference between the two versions makes it very difficult to have one assembly listing which owners of either Disk Basic ROM may use. To solve this problem, an assembly listing of both versions is contained in the book. The 1.1 version will be the most useful since it has had most of the bugs, which were in the 1.Ø version corrected. The complete memory map will not be given for the 1.Ø version since the memory maps for both versions of the ROM are identical.

HOW TO USE THIS BOOK

Disk BASIC Unravelled is a commented, disassembled listing of the TRS-80 Color Computer Disk BASIC ROM. The author has never seen any kind of source listing for the Color Computer ROMs, so the comments and disassembly are 100% unique. Some of the variable label literals, which were used, have come from published memory maps of systems, which use a BASIC similar to that used in the Color Computer.

The labels used in the disassembly correspond to absolute addresses in RAM preceded by an L . The labels correspond to the ad-dresses in Version 1.0 of the ROM, which may cause some confusion when trying to cross-index the 1.0 and 1.1 versions.

Literal labels have been assigned to RAM variables (memory locations that contain data which may change) and some ROM routines and data tables. The symbol table in Appendix D will allow the user to locate the address of the literal label. If the address is between 0 and $989, the literal is a RAM variable, the description of which will be found in appendix A, the Memory Map. If the address is between $8000 and $9FFF, the label will be found in the Extended BASIC listing; if it is between $A000 and $BFFF, the label is in the Color BASIC listing and if it is between $C000 and $DFFF, the label is in the Disk BASIC listing. Some of the literal values such as SKP1, SECLEN, etc. are values not associated with an address. They are defined at the beginning of the Memory Map (appendix A) in the table of EQUATES (EQU). There is an additional group of EQUates at the beginning of the Disk Basic disassembly listing (Appendices B & C).

The > symbol will occasionally appear to the left of the address of an instruction. This symbol is used to indicate that a JMP, JSR or LBxx instruction is being used when a BRA, BSR or Bxx instruction would suffice. These instructions may be replaced by their short versions in order to save a few bytes if necessary.

There are several places in the original object code where an instruction of the form LDA 0,R (where R = X,Y,U,S) has been used. These have been replaced by instructions of the form LDA ,R which are more efficient in terms of processor time (one cycle shorter).

The different versions of the ROMs provided in this book are kept in one large disk file with conditional assembly flags which allow the assembly of whichever version is desired by merely changing a single flag in the source listing. This is a convenient method of keeping track of the different versions of the ROMs but it can cause havoc with the line numbers at the extreme left of the disassembly listing. The line numbers keep track of EVERY line in the source listing regardless of whether or not that particular line is assembled. If when using the disassembly listings, you notice a gap in the line numbers it means that the missing line numbers correspond to a section of code, which was skipped during the assembly of that particular listing. This invariably means that there is a difference in the ROMs at that particular point.

DESCRIPTION OF DISK BASIC

Disk Basic will allow the Color Computer to communicate with a floppy disk drive in order to rapidly store large amounts of data on a non volatile medium. Disk Basic is different from Extended Basic in the manner that Extended Basic provided the user with a package of graphics commands AND several useful non graphics commands, whereas Disk Basic provides ONLY disk oriented commands with no additional commands (there is approximately 1.5K of wasted space where something else could have been provided). Accordingly, any discussion of Disk Basic will center around only the TRS 8Ø Color Computer s DOS (Disk Operating System)

As computers have evolved over the years, one of the biggest problems faced was the storage of the large amounts of data and programs, which the computer must have access to. The amount of random access storage available to the user was relatively small compared to the total amount of storage required. Random access memory is very fast, fairly expensive and volatile (it is lost when the power is turned off). The first method of mass storage used was magnetic tape, which was non volatile and cheap, but slow. Then came the floppy disk which was non volatile, not quite so cheap, and faster than magnetic tape. Presently the floppy disk is the primary system for mass storage in microcomputers.

A floppy disk is a round piece of magnetic tape shaped like a record on which data is magnetically recorded. Somehow the data, which is stored on the disk, must be capable of being transferred to and from the computer s random access memory. This is a very complex task, which requires many things in order to be done properly. There must be a mechanical method of moving the disk and transferring the magnetic data to and from the disk. This job is performed by the disk drive. Also, there must be a method of formatting and transferring data to and from the computer s RAM and the disk drive. In the Color Computer the disk controller board does this. The majority of the work done by the disk controller is performed by the 1793 Floppy Disk Controller (FDC) which is an integrated circuit as complicated as the 68Ø9 chip. In order to make the process orderly and logical there must be an overall controlling format or procedure for sending data to and from the 1793 (which will only provide primitive transference of blocks of data to and from the drive). The Disk Operating System (DOS) provides this overall control function by establishing a format for storing data and programs on the disk. The DOS provides a method of storing or retrieving blocks or single bytes of data to or from the disk drive.

The 1793 is capable of storing data on a disk in many different formats. For the Color Computer the 1793 is set up to save data on the disk in 35 tracks. Each track is composed of 18 sectors and each sector contains 256 bytes. The DOS treats this raw data as 68 granules with each granule containing 9 sectors, 2 granules per track. The one remaining track is used for the directory and the file allocation table (FAT).

FILE CONTROL BLOCK STRUCTURE

The File Control Block (FCB) is used by the DOS to control the transfer of data between the computer s RAM and the disk. It consists of 25 control bytes and a 256-byte data buffer. The 25 control bytes may have different functions if the file is a random/direct, sequential input or sequential output file. The data buffer is used to collect data so that the disk I/O will only be required when there is a full sector (256 bytes) of data to be input or output to the disk. The use of this buffer speeds up the overall disk I/O by cutting down on the number of times that actual disk accesses are required.

The number of FCBs allowed is set by the FILES command, which initializes the direct page variable FCBACT (the maximum number of FCBs allowed). The DOS always sets up a system FCB directly above the last allocated FCB, which is reserved for the exclusive use of the DOS and is not accessible to the user through BASIC. The system FCB is used when the system requires an FCB for disk I/O during the execution of such commands as MERGE, COPY, SAVE, LOAD, etc. This FCB may be accessed by the user under machine language but care must be exercised to insure that none of the BASIC commands which utilize the system FCB are used when doing so.

The OPEN command is used to initialize the FCB for disk I/O. It keeps track of which byte, sector, track and granule is currently being accessed by the DOS for the file controlled by the FCB. When disk I/O has been completed, the FCB is deactivated with the CLOSE command. When an FCB is closed, it is available for use by another file and once the FCB is used by another file, all of the information used by the previous file is lost. Some of the information must be saved since the user may want to reopen the same file for use later
on. Only six bytes from the FCB must be saved in order to be able to reinitialize an FCB. These six bytes are the file type (1), ASCII flag (1), first granule in file (2) and the number of bytes used in the last sector (2) and they are stored in the directory. A two-byte quantity is used to store number of bytes used in the last sector since the number of bytes may be any number from Ø to 256 ($1ØØ).

Listed below are those FCB control bytes, which are common to all types of files and their relative offset from the start of the FCB.

OFFSET	NAME	DESCRIPTION
Ø	FCBTYP	Single byte code representing the file type under which the file was opened. It may not have any relationship to the actual type of data stored in the file; a sequential file may be opened as a random file and vice versa. The allowed codes are: $1Ø = Sequential input, $2Ø = Sequential output, $4Ø = random, Ø = killed file.
1	FCBDRV	Single byte quantity defining the drive number where the file is located (Ø 3).
2	FCBFGR	Single byte quantity defining the first granule used by the file.
3	FCBCGR	Single byte quantity defining the current granule being accessed by the FCB.

4 FCBSEC Single byte quantity defining the current sector being accessed by the FCB (1 9).

18 FCBDIR Single byte quantity defining the directory entry number for this file (Ø 71).

19 FCBLST Double byte quantity containing the number of bytes used by this file in the last sector of the file.

Listed below are the definitions of the non common FCB control bytes as used by random files.

OFFSET	NAME	DESCRIPTION
5	FCBCPT	Unused
6	FCBPOS	Print position - always zero
7	FCBREC	Double byte quantity containing the current record number being used by the FCB.
8	FCBRLN	Double byte quantity containing the length of a record
11	FCBBUF	Double byte quantity containing a pointer to the absolute address of the start of random file buffer, which is exactly one record length long.
13	FCBSOF	Double byte quantity containing the sector offset to the current position in the record. These bytes are used to keep track of how many sectors from the beginning of a random file the current data being processed is located. These bytes are used to determine if the data in the FCB data buffer are valid for the current record number being processed. The high order byte is often set to $FF to cause new data to be read into the FCB data buffer.
15	FCBFLG	Single byte GET / PUT flag: Ø=GET, 1=PUT.
16		Two unused bytes
21	FCBGET	Double byte quantity containing the number of characters, which have been pulled out of the current record. These bytes are set to zero every time a record is stored in (PUT) or retrieved from (GET) a file.
23	FCBPUT	Double byte quantity containing the number of characters, which have been PUT into the current record. These bytes are set to zero every time a record is stored in (PUT) or retrieved from (GET) a file.

Listed below are the definitions of the non common FCB control bytes as used by sequential files.

OFFSET	NAME	DESCRIPTION
5	FCBCPT	Single byte quantity pointing to the next character to be processed for input files. When this byte is incremented to zero it indicates that the data buffer needs to be refilled. For output files this byte is used to indicate that 256 bytes of the last sector in the file have been used in case a DISK FULL error occurs while searching for an unused granule6 FCBPOS Single byte quantity containing the current print position in the file for output files, unused for input files. A carriage return in the output data stream will reset this value to zero.
7	FCBREC	Double byte quantity containing the number of whole sectors which have been input or output to a file.
9 15		Seven unused bytes.
16	FCBCFL	Single byte cache flag: ØØ=cache empty, $FF=cache full when inputting data, the DOS treats a CR, LF sequence as a CR. Therefore the DOS must look for a LF after a CR and if it does not find a LF, it must save that character for the next time an input character is needed. The cache flag indicates whether or not an extra character, which needs to be saved (cached), has been pulled out of an input file.
17	FCBCDT	Single byte cache data byte. If the cache flag is set the cache data byte is stored here.
23	FCBDFL	Single byte data left flag for input files: ØØ=data still left in file, $FF=no data left in file.
24	FCBLFT	Single byte quantity containing the number of characters left in the data buffer of an input file or the number of characters stored in the data buffer of an output file.

FILE ALLOCATION TABLE

The file allocation table (FAT) is used to keep track of whether or not a granule has been allocated to a file or if it is free. The FAT is composed of six control bytes followed by 68 data bytes one byte for each granule. The FAT is stored on sector two of the directory track (17). A RAM image of the FAT is kept in the disk RAM for each of the four possible drives. Keeping an image of the FAT in RAM helps speed up the overall operation of the DOS by eliminating the need for disk I/O every time the DOS modifies the FAT. Saving the FAT to disk is done approximately every 19 times that a new granule is pulled from the free granule reserve. It is written to disk whenever a file is closed and there are some DOS operations, which force the FAT to be written to disk when that DOS operation allocates a free granule.

Only the DOS uses two of the six control bytes. The first FAT control byte keeps track of how many FCBs are active on the drive for a particular FAT. This byte is used to preclude the loading in of the FAT from disk when there is any active file currently using the FAT. You can imagine the disaster, which would occur if you were creating a file and had allocated some granules to your new file but had not saved the new FAT to disk when the old FAT was loaded into RAM on top of the new FAT. Your new file would be hopelessly gone. For that reason the DOS must not allow the FAT to be loaded into RAM from disk while an FCB is active for that FAT.

The second FAT control byte is used to govern the need to write data from the FAT RAM image to the disk. If the value of this byte is zero it means that the FAT RAM image is an exact copy of what is currently stored on the disk. If the value is non zero, it indicates that the data in the FAT RAM image has been changed since the last time that the FAT was written to disk. The number stored in this byte is an indicator of how many granules have been removed from the FAT since the last FAT to disk write. Some BASIC commands, such as KILL, cause an immediate FAT RAM image to disk write when granules are either freed or allocated. Other commands, which allocate granules, increment the second FAT control byte. This byte is then compared to the disk variable WFATVL and when the second control byte is >= WFATVL, the FAT is written to disk.

The FAT data bytes are used to determine whether or not a granule is free and if it has been allocated they are used to determine to which file the granule belongs. If a data byte is $FF, it means that the granule is free and may be allocated to any file. If a granule has been allocated, it is part of a sector chain, which defines which granules belong to a certain file. The only information required to be able to trace the granule chain is the number of the first granule in the chain. If the first granule of the chain is not known, the chain cannot be traced down backwards.

A granule data byte, which has been allocated, will contain a value, which is the number of the next granule in the granule chain for that file. If the two most significant bits (6,7) of a granule data byte are set, then that granule is the last granule in a file s granule chain. The low order four bits will contain the number of sectors in the last granule, which the file uses. Even though a file may not use all of the sectors in the last granule in the chain, no other file may use the sectors. Disk space is not allocated on a sector basis, it is allocated on a granule basis and the granule may not be broken down. The smallest one-byte file

will still require a full granule to be allocated in order to store the file.

Granules are allocated in such a manner that will cause them to be relatively uniformly spread around the disk. This will lessen wear on the disk by not always allocating certain granules so that the disk drive head will not pass over certain sections of the disk too often. This is a common method used by a DOS in order to increase the life of a disk by spreading out the wear over as large a surface as possible, which could not be done if the granules were allocated on a strictly next in line numerical basis.

THE DIRECTORY

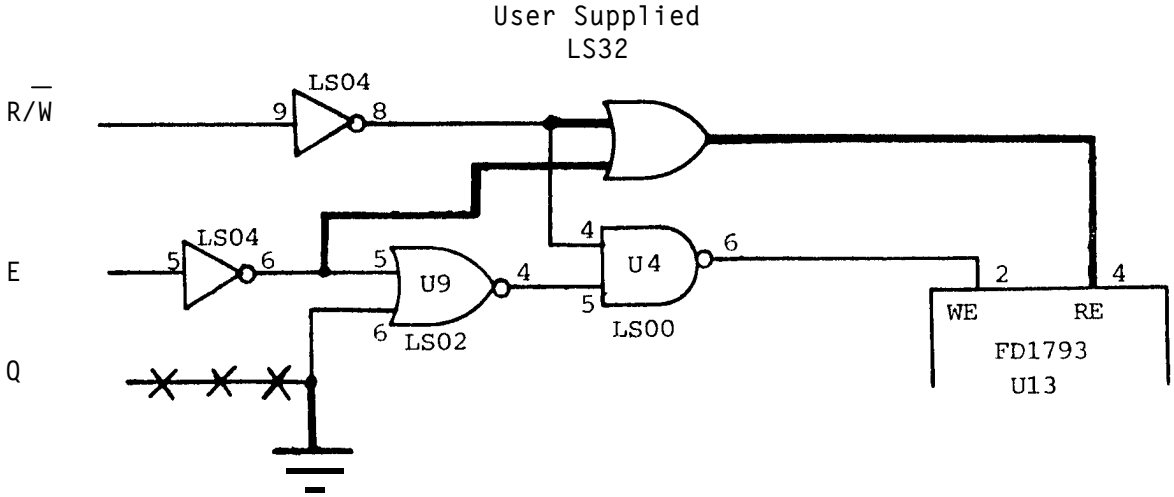
The directory is used by the DOS to keep track of how many files are stored on a disk. Track 17 is reserved for the directory and the file allocation table (FAT). The FAT resides on sector 2 and the directory occupies sectors 3-11. The remaining sectors are not used by the DOS in the current or past revisions to the BASIC ROMs. Each directory entry requires 32 bytes; so eight directory entries will fit in one sector for a total of 72 maximum directory entries. However, one full granule is required for each directory entry and there are only 68 granules on a disk so that only 68 directory entries (files) may exist on a disk at any time.

The format of the 32-byte directory entry is as follows:

Byte	Description
Ø 7	Filename, which is left justified and blank, filled. If byteØ is Ø, then the file has been KILL ed and the directory entry is available for use. If byteØ is $FF, then the entry and all following entries have never been used.
8 1Ø	Filename extension
11	File type: Ø=BASIC, 1=BASIC data, 2=Machine language, 3= Text editor source
12	ASCII flag: Ø=binary or crunched BASIC, $FF=ASCII
13	Number of the first granule in the file
14 15	Number of bytes used in the last sector of the file
16 31	Unused (future use)

DIRECTORY CRASHES

There has been much talk in the Color Computer media about so called directory crashes . These directory crashes seem to occur at random and will result in an unreadable directory, which causes the loss of all data on the directory at the time. The typical solution for this problem seems to be keeping a spare copy of the directory on an unused portion of the disk so that if a crash occurs, the spare copy may be used to restore the directory. A couple of years ago, we were having the same problem and Rodger Rosenbaum, genius extraordinaire, solved the problem by finding the bug in the disk controller which was causing it. Given below is the hardware fix, which will cure the directory crash problem. Spectral Associates does not guarantee nor does it recommend this fix and Spectral Associates will not assume any responsibility or liability for damages caused should any person or entity use or attempt to use the fix.



NOTE: This fix is only valid for the old style disk controllers. The new 5-volt only controllers do not suffer from this problem and should not be modified.

Add the wires shown with bold black lines. In order to install the fix without cutting traces on the PC board, gently lift IC s U9 and U13 out of their sockets, bend pins 6 and 4 respectively out and replace the IC s in their sockets. Then solder wires to the bent out pins.

1793 FLOPPY DISK CONTROLLER DESCRIPTION

The 1793 Floppy Disk Controller (FDC) is the heart of the disk controller card, which interfaces the Color Computer to the disk drive. Only the basic knowledge of the FDC in order to understand the operation of Disk Basic is presented here. If further, detailed information concerning the operation of the FDC is required, the reader is referred to the 1793 data sheet published by the Western Digital Corp. Only those functions of the 1793, which may be accessed by software the Color Computer, will be discussed.

The FDC is responsible for controlling the transference of data between the computer and the disk drive. There are many different disk drives manufactured by different companies and each drive has its own peculiarities, which require the FDC to be set up or used in a slightly different manner. The Color Computer Disk Controller board (which you plug into the cartridge slot) provides most of the support and set up functions, which the FDC requires and an eight-bit latch is used to store the FDC functions, which are software programmable, and those disk drive functions, which must be controlled directly by the computer. This control latch is located at $FF40 and is a write only latch, which means that data may only be written into the latch; no provision has been given to read the contents of the latch. For this reason, the DOS has reserved a byte in the Disk RAM (DRGRAM) which is an image of the contents of $FF40 so that the system software will know the status of the control latch at any time. Listed below are the functions, which may be controlled by DSKREG ($FF40).

bit #	Description
0	Drive select 0
1	Drive select 1
2	Drive select 2
3	Drive motor enable: 0=motors off, 1=motors on
4	Write pre-compensation flag: 0=no pre-comp, 1=pre-comp
5	Density flag: 0=single density, 1=double density
6	Drive select 3
7	Halt flag: 0=halt disabled, 1=halt enabled

$FF40 Control Functions

The drive select flags directly control which drive will communicate with the computer and the motor enable flag will enable or disable the motors of all of the drives. The density flag indicates to the FDC whether the data will be stored onto the diskette in single or double density. Write pre-compensation is used to correct the problem on a double density formatted disk of certain bit patterns causing a bit to shift from its nominal write position and appear at the read data separator early or late. Write pre-compensation rectifies this problem during disk writes by shifting such a bit from its nominal position in the opposite direction to its known read shift. Write precomp is usually necessary only for data written on the tracks on the inner half of the disk. The tracks on which write pre-compensation should be enabled vary from manufacturer and the number of the track at which write precomp is enabled in the Tandy disk is 22. Write precomp is on for tracks with a number greater than 22. The halt flag is used to enable the FDC board to halt the 6809. This is used to enable the Color Computer to operate the disk drives in double density mode at the low (.89 MHz) clock speed at which the Color Computer

runs. When the halt flag is high, the DRQ (Data ReQuest) signal from the FDC will be connected to the halt input of the 68Ø9. This will allow the DRQ signal to control the operation of the 68Ø9 to the extent that the 68Ø9 will not process any instructions while the FDC is processing data to or from the 68Ø9. Writing a zero to bit 7 of will clear the halt flag $FF4Ø or it will be cleared whenever the FDC generates an INTRQ (Interrupt Request) signal, which indicates that the FDC has completed its current command.

Data transfer between the computer and the disk drives through the FDC is accomplished through a series of hardware and software tricks . The slow clock speed of the Color Computer will not allow data to be transferred in the normal method of getting a byte from the computer, giving it to the FDC and then performing status checks until the FDC is not busy. There is just not enough time for this when operating at double density. Part of the Color Computer s solution is a hardware trick whereby the disk controller board will allow the FDC to halt the 68Ø9 while the FDC is storing or retrieving data. The halt flag will allow the DRQ signal from the FDC to halt the 68Ø9 so that the 68Ø9 will wait while the FDC is processing a data request. This trick will allow the 68Ø9 to pass data to the FDC as fast as the FDC can take it by executing a fast loop of: grab a byte from RAM, give it to the FDC and loop back to get another byte. The analogous loop for getting data will also work. The only problem is how to get out of this loop. This problem is solved with software - when an FDC command such as WRITE SECTOR or READ SECTOR is completed an interrupt (INTRQ) is generated by the FDC. The Color Computer connects this INTFQ signal to the Non Maskable Interrupt (NMI) pin of the 68Ø9. This means that whenever an FDC command (except the $DØ FORCE INTERRUPT) is completed, an NMI will be generated. The computer will now be able to tell that an FDC command is over; all that is left is for the computer to know where to go when the command is finished. This is accomplished by storing a jump vector (DNMIVC) in the disk RAM prior to entering the FDC data transfer loop. Another byte in disk RAM is used as a flag (NMIFLG) to indicate that the NMI jump vector should be used. If the NMIFLG is not equal to zero and an NMI is received by the 68Ø9, Disk Basic will cause the NMIFLG to be reset and control will be transferred to the address in DNMIVC. It is exactly this method, which is used to exit from the FDC data transfer routines.

The FDC has four registers, which are used to communicate with the computer. Their functions are described below:

ADDRESS	READ	WRITE
$FF48	STATUS REGISTER	COMMAND REGISTER
$FF49	TRACK REGISTER	TRACK REGISTER
$FF4A	SECTOR REGISTER	SECTOR REGISTER
$FF4B	DATA REGISTER	DATA REGISTER

The track and sector registers are used by the FDC to remember where the disk drive s head is currently located. The data register is used to pass data to and from the FDC. The command register is used to pass one of the eleven possible commands to the FDC. Each of these commands has several different forms, which are explained in the FDC data sheet. The form used by Disk Basic are listed below:

TYPE	COMMAND	CODE
I	RESTORE	$Ø3
I	SEEK	$17

I	STEP	$23
I	STEP IN	$43
I	STEP OUT	$53
II	READ SECTOR	$8Ø
II	WRITE SECTOR	$AØ
III	READ ADDRESS	$CØ
III	READ TRACK	$E4
III	WRITE TRACK	$F4
IV	FORCE INTERRUPT	$DØ

The status register is used to reflect the results of an FDC command. The contents of the status register will vary depending upon the TYPE of FDC command, which was executed. Listed below are the contents of the status register for the various TYPEs.

STATUS REGISTER SUMMARY

BIT	ALL TYPE I COMMANDS	READ ADDRESS	READ SECTOR	READ TRACK	WRITE SECTOR	WRITE TRACK
S7	NOT READY	NOT READY	NOT READY	NOT READY	NOT READY	NOT READY
S6	WRITE PROTECT	Ø	Ø	Ø	WRITE PROTECT	WRITE PROTECT
S5	HEAD LOADED	Ø	RECORD TYPE	Ø	WRITE FAULT	WRITE FAULT
S4	SEEK ERROR	RNF	RNF	Ø	RNF	Ø
S3	CRC ERROR	CRC ERROR	CRC ERROR	Ø	CRC ERROR	Ø
S2	TRACK Ø	LOST DATA	LOST DATA	LOST DATA	LOST DATA	LOST DATA
SI	INDEX	DRO	DRO	DRO	DRO	DRO
S1	BUSY	BUSY	BUSY	BUSY	BUSY	BUSY

STATUS FOR TYPE I COMMANDS

BIT NAME	MEANING
S7 NOT READY	This bit when set indicates the drive is not ready. When reset it indicates that the drive is ready. This bit is an inverted copy of the Ready input and logically or d with MR.
S6 PROTECTED	When set, indicates Write Protect is activated. This bit is an inverted copy at WRPT input.
S5 HEAD LOADED	When set, it indicates the head is loaded and engaged. This bit is a logical and of HLD and HLT signals.
S4 SEEK ERROR	When set, the desired track was not verified. This bit is reset to Ø when updated.
S3 CRC ERROR	CRC encountered in ID field.
S2 TRACK ØØ	When set, indicates Read/Write head is positioned to Track Ø. This bit is an inverted copy of the TRØØ input.
S1 INDEX	When set, indicates index mark detected from drive. This bit is an inverted copy of the IP input.
SØ BUSY	When set, command is in progress. When reset, no command is in progress.

STATUS FOR TYPE II AND III COMMANDS

BIT NAME	MEANING
S7 NOT READY	This bit when set indicates the drive is not ready. When reset, it indicates that the drive is ready. This bit is an inverted copy of the Ready input and or d with MR. The Type II and Ill Commands will not execute unless the drive is ready.
S6 WRITE PROTECT	On Read Record, Not Used. On Read Track, Not Used. On any Write: It indicates a Write Protect. This bit is reset when updated.
S5 RECORD TYPE/WRITE FAULT	On Read Record. It indicates the record type code from data field address mark. 1 = Deleted Data Mark. Ø = Data Mark. On any Write: It indicates a Write Fault. This bit is reset when updated.
S4 RECORD NOT FOUND (RNF)	When set, it indicates that the desired track, sector, or side was not found. This bit is reset when updated.
S3 CRC ERROR	If S4 is set, an error is found in one or more ID fields: otherwise it indicates error in data field. This bit is reset when updated.
S2 LOST DATA	When set, it indicates the computer did not respond to DRQ in one byte time. This bit is reset to zero when updated.
S1 DATA REQUEST	This bit is a copy of the DRQ output. When set, it indicates the DR is full on a Read Operation or the DR is empty on a Write operation. This bit Is reset to zero when updated.
SØ BUSY	When set, command is under execution. When reset, no command Is under execution.

The disk variable DCSTA is not a true reflection of the contents of the FDC status register. Disk Basic filters the status bits of the FDC status register and allow only those errors, which Disk Basic requires to pass through.

MACHINE LANGUAGE FILE INPUT/OUTPUT

The DOS uses a special format for transferring binary files to and from the disk. The format is fairly simple and straightforward and allows the loading of non contiguous blocks of memory from the same file. The only problem is that Radio Shack has not provided a SAVEM function, which will allow the saving of non contiguous blocks of memory into one disk file. This minor problem can be gotten around with the help of a neat utility called JOIN which is included in the Spectral Associates Color Computer Editor Assembler, ULTRA 80CC. This utility will allow the concatenation of as many machine language files as the user requires into one large file. LOADM will then load all of the segments into memory and the segments may overlay one another.

Binary data is stored on the disk as one large block proceeded by a five-byte preamble. The data block is followed by five more bytes which are another preamble if there is another block of data following or the five bytes are a post-amble if there are no further data blocks. The format for the preamble and the post-amble are given below:

BYTE	PREAMBLE	POSTAMBLE
0	00 Preamble flag	$FF Post-amble flag
1,2	Length of data block	Two zero bytes
3,4	Load address	EXEC address

DISK BASIC RAM VARIABLES/BUFFERS

Disk Basic requires a substantial amount of RAM for variables and buffer space. There is not enough room in the direct page for all of the variables so Disk Basic grabs a chunk of RAM immediately above the video display RAM (starting at $600) for its own use. It also uses nine bytes in the direct page, which leaves a total of 17 unused direct page variables for the user when Disk Basic is installed.

At the beginning of Disk RAM are two sector length (256 bytes) I/O buffers, which are primarily used to buffer data transfers to and from the disk controller. DBUFØ is the main I/O buffer and is involved in virtually all disk data transfers. The secondary I/O buffer, DBUF1 is used as a buffer during verify operations and is used as a scratchpad work area or a temporary home for the stack during certain other disk commands such as DSKI$, DSKO$ and DSKINI. Following these two buffers are four buffers for the File Allocation Tables and control blocks for the four drives. After these buffers are the variables, which Disk Basic uses for its own internal purposes, and a description of the function of these variables may be found in the direct page memory map found at the beginning of the disassembled list of Disk Basic.

Disk Basic also allocates two additional blocks of RAM for random file buffer and file control block storage. The area for these storage areas is directly after the disk variable RAM and just before the graphic page reserved area. The random file buffer area is used to save a one record length buffer for each active random file. The random file buffer size may be changed with the FILES command. Each time a random file is OPENed or CLOSEd one record length of memory is allocated or deallocated from the available buffer space in the random file buffer area. If there is not enough space in the random file buffer area to hold one record length, an OB (out of buffer space) error will be generated. Immediately after the random file buffer area is the area reserved for file control blocks. The number of available FCBs may be changed by the FILES command. Each FCB requires 281 bytes of RAM and there is always one more FCB (the system FCB) reserved than the number of FCBs requested by the FILES command.

```
0001   C000   ROMPAK     EQU   $C000
0002
0003   0008   BS         EQU   8               BACKSPACE
0004   000D   CR         EQU   $D              ENTER KEY
0005   001B   ESC        EQU   $1B             ESCAPE CODE
0006   000A   LF         EQU   $A              LINE FEED
0007   000C   FORMF      EQU   $C              FORM FEED
0008   0020   SPACE      EQU   $20             SPACE (BLANK)
0009
0010   003A   STKBUF     EQU   58              STACK BUFFER ROOM
0011   045E   DEBDEL     EQU   $45E            DEBOUNCE DELAY
0012   00FA   LBUFMX     EQU   250             MAX NUMBER OF CHARS IN A BASIC LINE
0013   00FA   MAXLIN     EQU   $FA             MAXIMUM MS BYTE OF LINE NUMBER
0014
0015   2600   DOSBUF     EQU   $2600           RAM LOAD LOCATION FOR THE DOS COMMAND
       0006   FATCON     EQU   6               NUMBER OF CONTROL BYTES BEFORE FAT
       0019   FCBCON     EQU   25              NUMBER OF CONTROL BYTES BEFORE FCB
0016   0020   DIRLEN     EQU   32              NUMBER OF BYTES IN DIRECTORY ENTRY
0017   0100   SECLEN     EQU   256             LENGTH OF SECTOR IN BYTES
0018   0012   SECMAX     EQU   18              MAXIMUM NUMBER OF SECTORS PER TRACK
0019   1200   TRKLEN     EQU   SECMAX*SECLEN   LENGTH OF TRACK IN BYTES
0020   0023   TRKMAX     EQU   35              MAX NUMBER OF TRACKS
0021   004A   FATLEN     EQU   6+(TRKMAX-1)*2  FILE ALLOCATION TABLE LENGTH
0022   0044   GRANMX     EQU   (TRKMAX-1)*2    MAXIMUM NUMBER OF GRANULES
0023   0119   FCBLEN     EQU   SECLEN+25       FILE CONTROL BLOCK LENGTH
0024   0010   INPFIL     EQU   $10             INPUT FILE TYPE
0025   0020   OUTFIL     EQU   $20             OUTPUT FILE TYPE
0026   0040   RANFIL     EQU   $40             RANDOM/DIRECT FILE TYPE
0027
0035              * SUPER EXTENDED BASIC EQUATES
0036   0018   ROWMAX     EQU   24              MAXIMUM NUMBER OF ROWS IN HI-RES PRINT MODE
0037   0000   RAMLINK    EQU   0               DUMMY RAM LINK VECTOR
0038   2000   HRESSCRN   EQU   $2000           ADDRESS OF THE HI-RES SCREEN IN THE CPU'S MEMORY SPACE
0039   C000   HRESBUFF   EQU   $C000           ADDRESS OF THE GET/PUT BUFFERS IN THE CPU'S MEMORY SPACE
0040   DFFF   TMPSTACK   EQU   $DFFF           ADDRESS OF THE HI-RES GRAPHICS STACK IN THE CPU'S MEMORY SPACE
0041   0062   EBHITOK    EQU   $62             FIRST ENHANCED BASIC TOKEN NUMBER
0042   0029   EBHISTOK   EQU   $29             FIRST ENHANCED BASIC FUNCTION TOKEN NUMBER BUG - SHOULD BE $28
0043   0020   CURCHAR    EQU   SPACE           HI-RES CURSOR CHARACTER
0044
0045              * HBUFF HGET/HPUT BUFFER HEADER EQUATES
0046   0000   HB.ADDR    EQU   0               ADDRESS OF THE NEXT BUFFER - 2 BYTES
0047   0002   HB.NUM     EQU   2               NUMBER OF THIS BUFFER - 1 BYTES
0048   0003   HB.SIZE    EQU   3               NUMBER OF BYTES IN THE BUFFER - 2 BYTES
0049   0005   HB.LEN     EQU   5               NUMBER OF BYTES IN THIS HEADER
0050
0051              * VIDEO REGISTER EQUATES
0052              * INIT0 BIT EQUATES
0053   0080   COCO       EQU   $80             1 = Color Computer compatible
0054   0040   MMUEN      EQU   $40             1 = MMU enabled
0055   0020   IEN        EQU   $20             1 = GIME chip IRQ output enabled
0056   0010   FEN        EQU   $10             1 = GIME chip FIRQ output enabled
0057   0008   MC3        EQU   8               1 = RAM at XFEXX is constant
0058   0004   MC2        EQU   4               1 = standard SCS
0059   0002   MC1        EQU   2               ROM map control
0060   0001   MC0        EQU   1               ROM map control
0061
0062              * INTERRUPT REQUEST ENABLED
0063   0020   TMR        EQU   $20             TIMER
0064   0010   HBORD      EQU   $10             HORIZONTAL BORDER
0065   0008   VBORD      EQU   8               VERTICAL BORDER
0066   0004   EI2        EQU   4               SERIAL DATA
0067   0002   EI1        EQU   2               KEYBOARD
0068   0001   EI0        EQU   1               CARTRIDGE
0069
0070              * EXPANDED MEMORY DEFINITIONS
0071   0030   BLOCK 6.0  EQU   $30             BLOCKS $30-$33 ARE THE HI-RES GRAPHICS SCREEN
0072   0031   BLOCK 6.1  EQU   $31             HI-RES GRAPHICS SCREEN
0073   0032   BLOCK 6.2  EQU   $32             HI-RES GRAPHICS SCREEN
0074   0033   BLOCK 6.3  EQU   $33             HI-RES GRAPHICS SCREEN
0075   0034   BLOCK 6.4  EQU   $34             GET/PUT BUFFER
0076   0035   BLOCK 6.5  EQU   $35             STACK AREA FOR HI-RES GRAPHICS COMMAND
0077   0036   BLOCK 6.6  EQU   $36             CHARACTER POINTERS
0078   0037   BLOCK 6.7  EQU   $37             UNUSED BY BASIC
0079
```

```
0080                * BLOCKS $48-$4F ARE USED FOR THE BASIC OPERATING SYSTEM
0081       0038     BLOCK7.0  EQU   $38
0082       0039     BLOCK7.1  EQU   $39
0083       003A     BLOCK7.2  EQU   $3A
0084       003B     BLOCK7.3  EQU   $3B
0085       003C     BLOCK7.4  EQU   $3C
0086       003D     BLOCK7.5  EQU   $3D
0087       003E     BLOCK7.6  EQU   $3E
0088       003F     BLOCK7.7  EQU   $3F
0089
0028                * PSEUDO PSEUDO OPS
0029       0021     SKP1      EQU   $21             OP CODE OF BRN   SKIP ONE BYTE
0030       008C     SKP2      EQU   $8C             OP CODE OF CMPX # - SKIP TWO BYTES
0031       0086     SKP1LD    EQU   $86             OP CODE OF LDA # - SKIP THE NEXT BYTE
0032                *                              AND LOAD THE VALUE OF THAT BYTE INTO ACCA   THIS
0033                *                              IS USUALLY USED TO LOAD ACCA WITH A NON ZERO VALUE
0090
0091
0092 0000                      ORG   0
0093       0000                SETDP 0
0094
0095 0000           ENDFLG     RMB   1              STOP/END FLAG: POSITIVE=STOP, NEG=END
0096 0001           CHARAC     RMB   1              TERMINATOR FLAG 1
0097 0002           ENDCUR     RMB   1              TERMINATOR FLAG 2
0098 0003           TMPLOC     RMB   1              SCRATCH VARIABLE
0099 0004           IFCTR      RMB   1              IF COUNTER - HOW MANY IF STATEMENTS IN A LINE
0100 0005           DIMFLG     RMB   1              *DV* ARRAY FLAG 0=EVALUATE, 1=DIMENSIONING
0101 0006           VALTYP     RMB   1              *DV* *PV TYPE FLAG: 0=NUMERIC, $FF=STRING
0102 0007           GARBFL     RMB   1              *TV STRING SPACE HOUSEKEEPING FLAG
0103 0008           ARYDIS     RMB   1              DISABLE ARRAY SEARCH: 00=ALLOW SEARCH
0104 0009           INPFLG     RMB   1              *TV INPUT FLAG: READ=0, INPUT<>0
0105 000A           RELFLG     RMB   1              *TV RELATIONAL OPERATOR FLAG
0106 000B           TEMPPT     RMB   2              *PV TEMPORARY STRING STACK POINTER
0107 000D           LASTPT     RMB   2              *PV ADDR OF LAST USED STRING STACK ADDRESS
0108 000F           TEMPTR     RMB   2              TEMPORARY POINTER
0109 0011           TMPTR1     RMB   2              TEMPORARY DESCRIPTOR STORAGE (STACK SEARCH)
0110                ** FLOATING POINT ACCUMULATOR #2 (MANTISSA ONLY)
0111 0013           FPA2       RMB   4              FLOATING POINT ACCUMULATOR #2 MANTISSA
0112 0017           BOTSTK     RMB   2              BOTTOM OF STACK AT LAST CHECK
0113 0019           TXTTAB     RMB   2              *PV BEGINNING OF BASIC PROGRAM
0114 001B           VARTAB     RMB   2              *PV START OF VARIABLES
0115 001D           ARYTAB     RMB   2              *PV START OF ARRAYS
0116 001F           ARYEND     RMB   2              *PV END OF ARRAYS (+1)
0117 0021           FRETOP     RMB   2              *PV START OF STRING STORAGE (TOP OF FREE RAM)
0118 0023           STRTAB     RMB   2              *PV START OF STRING VARIABLES
0119 0025           FRESPC     RMB   2              UTILITY STRING POINTER
0120 0027           MEMSIZ     RMB   2              *PV TOP OF STRING SPACE
0121 0029           OLDTXT     RMB   2              SAVED LINE NUMBER DURING A "STOP"
0122 002B           BINVAL     RMB   2              BINARY VALUE OF A CONVERTED LINE NUMBER
0123 002D           OLDPTR     RMB   2              SAVED INPUT PTR DURING A "STOP"
0124 002F           TINPTR     RMB   2              TEMPORARY INPUT POINTER STORAGE
0125 0031           DATTXT     RMB   2              *PV 'DATA' STATEMENT LINE NUMBER POINTER
0126 0033           DATPTR     RMB   2              *PV 'DATA' STATEMENT ADDRESS POINTER
0127 0035           DATTMP     RMB   2              DATA POINTER FOR 'INPUT' & 'READ'
0128 0037           VARNAM     RMB   2              *TV TEMP STORAGE FOR A VARIABLE NAME
0129 0039           VARPTR     RMB   2              *TV POINTER TO A VARIABLE DESCRIPTOR
0130 003B           VARDES     RMB   2              TEMP POINTER TO A VARIABLE DESCRIPTOR
0131 003D           RELPTR     RMB   2              POINTER TO RELATIONAL OPERATOR PROCESSING ROUTINE
0132 003F           TRELFL     RMB   1              TEMPORARY RELATIONAL OPERATOR FLAG BYTE
0133
0134                * FLOATING POINT ACCUMULATORS #3,4 & 5 ARE MOSTLY
0135                * USED AS SCRATCH PAD VARIABLES.
0136                ** FLOATING POINT ACCUMULATOR #3 :PACKED: ($40-$44)
0137 0040           V40        RMB   1
0138 0041           V41        RMB   1
0139 0042           V42        RMB   1
0140 0043           V43        RMB   1
0141 0044           V44        RMB   1
0142                ** FLOATING POINT ACCUMULATOR #4 :PACKED: ($45-$49)
0143 0045           V45        RMB   1
0144 0046           V46        RMB   1
0145 0047           V47        RMB   1
0146 0048           V48        RMB   2
0147                ** FLOATING POINT ACCUMULATOR #5 :PACKED: ($4A $4E)
```

```
0148 004A           V4A       RMB   1
0149 004B           V4B       RMB   2
0150 004D           V4D       RMB   2
0151                ** FLOATING POINT ACCUMULATOR #0
0152 004F           FP0EXP    RMB   1               *PV FLOATING POINT ACCUMULATOR #0 EXPONENT
0153 0050           FPA0      RMB   4               *PV FLOATING POINT ACCUMULATOR #0 MANTISSA
0154 0054           FP0SGN    RMB   1               *PV FLOATING POINT ACCUMULATOR #0 SIGN
0155 0055           COEFCT    RMB   1               POLYNOMIAL COEFFICIENT COUNTER
0156 0056           STRDES    RMB   5               TEMPORARY STRING DESCRIPTOR
0157 005B           FPCARY    RMB   1               FLOATING POINT CARRY BYTE
0158                ** FLOATING POINT ACCUMULATOR #1
0159 005C           FP1EXP    RMB   1               *PV FLOATING POINT ACCUMULATOR #1 EXPONENT
0160 005D           FPA1      RMB   4               *PV FLOATING POINT ACCUMULATOR #1 MANTISSA
0161 0061           FP1SGN    RMB   1               *PV FLOATING POINT ACCUMULATOR #1 SIGN
0162
0163 0062           RESSGN    RMB   1               SIGN OF RESULT OF FLOATING POINT OPERATION
0164 0063           FPSBYT    RMB   1               FLOATING POINT SUB BYTE (FIFTH BYTE)
0165 0064           COEFPT    RMB   2               POLYNOMIAL COEFFICIENT POINTER
0166 0066           LSTTXT    RMB   2               CURRENT LINE POINTER DURING LIST
0167 0068           CURLIN    RMB   2               *PV CURRENT LINE # OF BASIC PROGRAM, $FFFF = DIRECT
0168 006A           DEVCFW    RMB   1               *TV TAB FIELD WIDTH
0169 006B           DEVLCF    RMB   1               *TV TAB ZONE
0170 006C           DEVPOS    RMB   1               *TV PRINT POSITION
0171 006D           DEVWID    RMB   1               *TV PRINT WIDTH
0172 006E           PRTDEV    RMB   1               *TV PRINT DEVICE: 0=NOT CASSETTE, -1=CASSETTE
0173 006F           DEVNUM    RMB   1               *PV DEVICE NUMBER: -3=DLOAD, -2=PRINTER,
0174                *                                                  -1=CASSETTE, 0=SCREEN, 1-15=DISK
0175 0070           CINBFL    RMB   1               *PV CONSOLE IN BUFFER FLAG: 00=NOT EMPTY, $FF=EMPTY
0176 0071           RSTFLG    RMB   1               *PV WARM START FLAG: $55=WARM, OTHER=COLD
0177 0072           RSTVEC    RMB   2               *PV WARM START VECTOR - JUMP ADDRESS FOR WARM START
0178 0074           TOPRAM    RMB   2               *PV TOP OF RAM
0179 0076                     RMB   2               SPARE: UNUSED VARIABLES
0180 0078           FILSTA    RMB   1               *PV FILE STATUS FLAG: 0=CLOSED, 1=INPUT, 2=OUTPUT
0181 0079           CINCTR    RMB   1               *PV CONSOLE IN BUFFER CHAR COUNTER
0182 007A           CINPTR    RMB   2               *PV CONSOLE IN BUFFER POINTER
0183 007C           BLKTYP    RMB   1               *TV CASS BLOCK TYPE: 0=HEADER, 1=DATA, $FF=EOF
0184 007D           BLKLEN    RMB   1               *TV CASSETTE BYTE COUNT
0185 007E           CBUFAD    RMB   2               *TV CASSETTE LOAD BUFFER POINTER
0186 0080           CCKSUM    RMB   1               *TV CASSETTE CHECKSUM BYTE
0187 0081           CSRERR    RMB   1               *TV ERROR FLAG/CHARACTER COUNT
0188 0082           CPULWD    RMB   1               *TV PULSE WIDTH COUNT
0189 0083           CPERTM    RMB   1               *TV BIT COUNTER
0190 0084           CBTPHA    RMB   1               *TV BIT PHASE FLAG
0191 0085           CLSTSN    RMB   1               *TV LAST SINE TABLE ENTRY
0192 0086           GRBLOK    RMB   1               *TV GRAPHIC BLOCK VALUE FOR SET, RESET AND POINT
0193 0087           IKEYIM    RMB   1               *TV INKEY$ RAM IMAGE
0194 0088           CURPOS    RMB   2               *PV CURSOR LOCATION
0195 008A           ZERO      RMB   2               *PV DUMMY - THESE TWO BYTES ARE ALWAYS ZERO
0196 008C           SNDTON    RMB   1               *TV TONE VALUE FOR SOUND COMMAND
0197 008D           SNDDUR    RMB   2               *TV DURATION VALUE FOR SOUND COMMAND
0198
0199                ** THESE BYTES ARE MOVED DOWN FROM ROM
0200                ***                             INIT   DESCRIPTION
0201                *                               VALUE
0202 008F           CMPMID    RMB   1               18     *PV 1200/2400 HERTZ PARTITION
0203 0090           CMP0      RMB   1               24     *PV UPPER LIMIT OF 1200 HERTZ PERIOD
0204 0091           CMP1      RMB   1               10     *PV UPPER LIMIT OF 2400 HERTZ PERIOD
0205 0092           SYNCLN    RMB   2               128    *PV NUMBER OF $55'S TO CASSETTE LEADER
0206 0094           BLKCNT    RMB   1               11     *PV CURSOR BLINK DELAY
0207 0095           LPTBTD    RMB   2               88     *PV BAUD RATE CONSTANT (600)
0208 0097           LPTLND    RMB   2               1      *PV PRINTER CARRIAGE RETURN DELAY
0209 0099           LPTCFW    RMB   1               16     *PV TAB FIELD WIDTH
0210 009A           LPTLCF    RMB   1               112    *PV LAST TAB ZONE
0211 009B           LPTWID    RMB   1               132    *PV PRINTER WIDTH
0212 009C           LPTPOS    RMB   1               0      *PV LINE PRINTER POSITION
0213 009D           EXECJP    RMB   2               LB4AA  *PV JUMP ADDRESS FOR EXEC COMMAND
0214
0215                ** THIS ROUTINE PICKS UP THE NEXT INPUT CHARACTER FROM
0216                ** BASIC. THE ADDRESS OF THE NEXT BASIC BYTE TO BE
0217                ** INTERPRETED IS STORED AT CHARAD.
0218
0219 009F 0C A7     GETNCH    INC   <CHARAD+1       *PV INCREMENT LS BYTE OF INPUT POINTER
0220 00A1 26 02               BNE   GETCCH          *PV BRANCH IF NOT ZERO (NO CARRY)
0221 00A3 0C A6               INC   <CHARAD         *PV INCREMENT MS BYTE OF INPUT POINTER
```

```
0222 00A5 B6        GETCCH     FCB   $B6             *PV OP CODE OF LDA EXTENDED
0223 00A6           CHARAD           2               *PV THESE 2 BYTES CONTAIN ADDRESS OF THE CURRENT
0224                *                                    CHARACTER WHICH THE BASIC INTERPRETER IS
0225                *                                    PROCESSING
0226 00A8 7E AA 1A            JMP   BROMHK          JUMP BACK INTO THE BASIC RUM
0227
0228 00AB           VAB        RMB   1               = LOW ORDER FOUR BYTES OF THE PRODUCT
0229 00AC           VAC        RMB   1               = OF A FLOATING POINT MULTIPLICATION
0230 00AD           VAD        RMB   1               = THESE BYTES ARE USE AS RANDOM DATA
0231 00AE           VAE        RMB   1               = BY THE RND STATEMENT
0232
0233                * EXTENDED BASIC VARIABLES
0234 00AF           TRCFLG     RMB   1               *PV TRACE FLAG Ø=OFF ELSE=ON
0235 00B0           USRADR     RMB   2               *PV ADDRESS OF THE START OF USR VECTORS
0236 00B2           FORCOL     RMB   1               *PV FOREGROUND COLOR
0237 00B3           BAKCOL     RMB   1               *PV BACKGROUND COLOR
0238 00B4           WCOLOR     RMB   1               *TV WORKING COLOR BEING USED BY EX BASIC
0239 00B5           ALLCOL     RMB   1               *TV ALL PIXELS IN THIS BYTE SET TO COLOR OF VB3
0240 00B6           PMODE      RMB   1               *PV PMODE'S MODE ARGUMENT
0241 00B7           ENDGRP     RMB   2               *PV END OF CURRENT GRAPHIC PAGE
0242 00B9           HORBYT     RMB   1               *PV NUMBER OF BYTES/HORIZONTAL GRAPHIC LINE
0243 00BA           BEGGRP     RMB   2               *PV START OF CURRENT GRAPHIC PAGE
0244 00BC           GRPRAM     RMB   1               *PV START OF GRAPHIC RAM (MS BYTE)
0245 00BD           HORBEG     RMB   2               *DV* *PV HORIZ COORD - START POINT
0246 00BF           VERBEG     RMB   2               *DV* *PV VERT COORD - START POINT
0247 00C1           CSSYAL     RMB   1               *PV SCREEN'S COLOR SET ARGUMENT
0248 00C2           SETFLG     RMB   1               *PV PRESET/PSET FLAG: Ø=PRESET, 1=PSET
0249 00C3           HOREND     RMB   2               *DV* *PV HORIZ COORD - ENDING POINT
0250 00C5           VEREND     RMB   2               *DV* *PV VERT COORD - ENDING POINT
0251 00C7           HORDEF     RMB   2               *PV HORIZ COORD - DEFAULT COORD
0252 00C9           VERDEF     RMB   2               *PV VERT COORD - DEFAULT COORD
0253
0254                * EXTENDED BASIC SCRATCH PAD VARIABLES
0255 00CB           VCB        RMB   2
0256 00CD           VCD        RMB   2
0257 00CF           VCF        RMB   2
0258 00D1           VD1        RMB   2
0259 00D3           VD3        RMB   1
0260 00D4           VD4        RMB   1
0261 00D5           VD5        RMB   1
0262 00D6           VD6        RMB   1
0263 00D7           VD7        RMB   1
0264 00D8           VD8        RMB   1
0265 00D9           VD9        RMB   1
0266 00DA           VDA        RMB   1
0267
0268 00DB           CHGFLG     RMB   1               *TV FLAG TO INDICATE IF GRAPHIC DATA HAS BEEN CHANGED
0269 00DC           TMPSTK     RMB   2               *TV STACK POINTER STORAGE DURING PAINT
0270 00DE           OCTAVE     RMB   1               *PV OCTAVE VALUE (PLAY)
0271 00DF           VOLHI      RMB   1               *DV* *PV VOLUME HIGH VALUE (PLAY)
0272 00E0           VOLLOW     RMB   1               *DV* *PV VOLUME LOW VALUE (PLAY)
0273 00E1           NOTELN     RMB   1               *PV NOTE LENGTH (PLAY)
0274 00E2           TEMPO      RMB   1               *PV TEMPO VALUE (PLAY)
0275 00E3           PLYTMR     RMB   2               *TV TIMER FOR THE PLAY COMMAND
0276 00E5           DOTYAL     RMB   1               *TV DOTTED NOTE TIMER SCALE FACTOR
0277 00E6           HRMODE     EQU   *               SUPER EXTENDED BASIC HI-RES MODE
0278 00E6           DLBAUD     RMB   1               *DV* *PV DLOAD BAUD RATE CONSTANT $B0=300, $2C=1200
0279 00E7           HRWIDTH    EQU   *               SUPER EXTENDED BASIC HI-RES TEXT MODE
0280 00E7           TIMOUT     RMB   1               *DV* *PV DLOAD TIMEOUT CONSTANT
0281 00E8           ANGLE      RMB   1               *DV* *PV ANGLE VALUE (DRAW)
0282 00E9           SCALE      RMB   1               *DV* *PV SCALE VALUE (DRAW)
0283
0284                * DSKCON VARIABLES
0285 00EA           DCOPC      RMB   1               *PV DSKCON OPERATION CODE Ø-3
0286 00EB           DCDRV      RMB   1               *PV DSKCON DRIVE NUMBER Ø 3
0287 00EC           DCTRK      RMB   1               *PV DSKCON TRACK NUMBER Ø 34
0288 00ED           DSEC       RMB   1               *PV DSKCON SECTOR NUMBER 1-18
0289 00EE           DCBPT      RMB   2               *PV DSKCON DATA POINTER
0290 00F0           DCSTA      RMB   1               *PV DSKCON STATUS BYTE
0291
0292 00F1           FCBTMP     RMB   2               TEMPORARY FCB POINTER
0293
0294 00F3                      RMB   13              SPARE: UNUSED VARIABLES
0295
```

```
0296
0297                *                               BASIC  EXBASI(DOSBASIC
0298
0299 0100          SW3VEC     RMB   3              $XXXX  $XXXX  $3B3B  SWI3 VECTOR
0300 0103          SW2VEC     RMB   3              $XXXX  $XXXX  $3B3B  SWI2 VECTOR
0301 0106          SWIVEC     RMB   3              $XXXX  $XXXX  $XXXX  SWI VECTOR
0302 0109          NMIVEC     RMB   3              $XXXX  $XXXX  $D7AE  NMI VECTOR
0303 010C          IRQVEC     RMB   3              $A9B3  $894C  $D7BC  IRQ VECTOR
0304 010F          FRQVEC     RMB   3              $A0F6  $A0F6  $A0F6  FIRQ VECTOR
0305
0306 0112          TIMVAL
0307 0112          USRJMP     RMB   3              JUMP ADDRESS FOR BASIC'S USR FUNCTION
0308               *          RMB   2              TIMER VALUE FOR EXBAS
0309               *          RMB   1              UNUSED BY EXBAS OR DISK BASIC
0310 0115          RVSEED     RMB   1              * FLOATING POINT RANDOM NUMBER SEED EXPONENT
0311 0116                     RMB   4              * MANTISSA: INITIALLY SET TO $804FC75259
0312 011A          CASFLG     RMB   1              UPPER CASE/LOWER CASE FLAG: $FF=UPPER, 0=LOWER
0313 011B          DEBVAL     RMB   2              KEYBOARD DEBOUNCE DELAY (SET TO $45E)
0314 011D          EXPJMP     RMB   3              JUMP ADDRESS FOR EXPONENTIATION
0315               **                              INITIALLY SET TO ERROR FOR BASIC, $8489 FOR EX BASIC
0316
0317                ***       COMMAND INTERPRETATION VECTOR TABLE
0318
0319                ** FOUR SETS OF 10 BYTE TABLES:
0320
0321
0322                ** THE LAST USED TABLE MUST BE FOLLOWED BY A ZERO BYTE
0323                * THE JUMP TABLE VECTORS (3,4 AND 8,9) POINT TO THE JUMP TABLE FOR
0324                * THE FIRST TABLE. FOR ALL OTHER TABLES, THESE VECTORS POINT TO A
0325                * ROUTINE WHICH WILL VECTOR YOU TO THE CORRECT JUMP TABLE.
0326                * SUPER ENHANCED BASIC HAS MODIFIED THIS SCHEME SO THAT THE USER
0327                * TABLE MAY NOT BE ACCESSED.  ANY ADDITIONAL TABLES WILL HAVE TO BE
0328                * ACCESSED FROM A NEW COMMAND HANDLER.
0329
0330                *         BYTE  DESCRIPTION
0331                *         0     NUMBER OF RESERVED WORDS
0332                *         1,2   LOOKUP TABLE OF RESERVED WORDS
0333                *         3,4   JUMP TABLE FOR COMMANDS (FIRST TABLE)
0334                *               VECTOR TO EXPANSION COMMAND HANDLERS (ALL BUT FIRST TABLE)
0335                *         5     NUMBER OF SECONDARY FUNCTIONS
0336                *         6,7   LOOKUP TABLE OF SECONDARY FUNCTIONS (FIRST TABLE)
0337                *               VECTOR TO EXPANSION SECONDARY COMMAND HANDLERS (ALL BUT
0338                *               FIRST TABLE)
0339                *         8,9   JUMP TABLE FOR SECONDARY FUNCTIONS
0340                *         10    0 BYTE - END OF TABLE FLAG (LAST TABLE ONLY)
0341
0342 0120          COMVEC     RMB   10             BASIC'S TABLE
0343 012A                     RMB   10             EX BASIC'S TABLE
0344 0134                     RMB   10             DISC BASIC'S TABLE (UNUSED BY EX BASIC)
0345
0346                **** USR FUNCTION VECTOR ADDRESSES (EX BASIC ONLY)
0347 013E                     RMB   2              USR 0 VECTOR
0348 0140                     RMB   2              USR 1
0349 0142                     RMB   2              USR 2
0350 0144                     RMB   2              USR 3
0351 0146                     RMB   2              USR 4
0352 0148                     RMB   2              USR 5
0353 014A                     RMB   2              USR 6
0354 014C                     RMB   2              USR 7
0355 014E                     RMB   2              USR 8
0356 0150                     RMB   2              USR 9
0357
0358                *** THE ABOVE 20 BYTE USR ADDR VECTOR TABLE IS MOVED TO
0359                *** $95F-$972 BY DISC BASIC. THE 20 BYTES FROM $13E-$151
0360                *** ARE REDEFINED AS FOLLOWS:
0361
0362                *         RMB   10             USER (SPARE) COMMAND INTERPRETATION TABLE SPACE
0363                *         FCB   0              END OF COMM INTERP TABLE FLAG
0364                *         RMB   9              UNUSED BY DISK BASIC
0365
0366                *         COMMAND INTERPRETATION TABLE VALUES
0367                *               BYTE           BASIC  EX BAS DISK BASIC
0368                *               0              53                BASIC TABLE
0369                *               1,2            $AA66
```

```
0370                *               3,4             $AB67
0371                *               5               20
0372                *               6,7             $AB1A
0373                *               8,9             $AA29
0374
0375                *               0                       25              EX BASIC TABLE
0376                *               1,2                     $8183
0377                *               3,4                     $813C  $CE2E    ($CF0A 2.1)
0378                *               5                       14
0379                *               6,7                     $821E
0380                *               8,9                     $8168  $CE56    ($CF32 2.1)
0381
0382                *               0                       19 (20 2.1)     DISK BASIC TABLE
0383                *               1,2                     $C17F
0384                *               3,4                     $C2C0
0385                *               5                       6
0386                *               6,7                     $C201
0387                *               8,9                     $C236
0388
0389
0390 0152          KEYBUF     RMB   8               KEYBOARD MEMORY BUFFER
0391 015A          POTVAL     RMB   1               LEFT VERTICAL JOYSTICK DATA
0392 015B                     RMB   1               LEFT HORIZONTAL JOYSTICK DATA
0393 015C                     RMB   1               RIGHT VERTICAL JOYSTICK DATA
0394 015D                     RMB   1               RIGHT HORIZONTAL JOYSTICK DATA
0395
0396                * BASIC'S RAM VECTORS - INITIALIZED TO RTS BY COLOR BASIC
0397                * 25 SETS OF 3 BYTE INSTRUCTIONS WHICH ARE CALLED BY COLOR BASIC
0398                * EXTENDED AND DISK BASIC. THEIR PURPOSE IS TO ALLOW ENHANCEMENTS (SUCH
0399                * AS EX BASIC AND DOS BASIC) AS MORE ROMS ARE ADDED TO THE
0400                * SYSTEM BY EFFECTIVELY ALLOWING MORE CODE TO BE ADDED TO THE
0401                * ROUTINES IN EARLIER ROMS. THIS NEW CODE IS LOCATED IN THE NEW ROMS
0402                * AND THE ADDRESS TO GET TO THE NEW CODE IS IN BYTES 1 & 2 OF THE
0403                * RAM VECTOR. BYTE 0 WILL CONTAIN A $7E WHICH IS THE FIRST BYTE OF
0404                * THE JMP INSTRUCTION.
0405                * THE FIRST ADDRESS IN THIS TABLE IS THE ADDRESS IN BASIC WHICH
0406                * CALLS THE RAM VECTOR, THE SECOND ADDRESS IS THE VALUE WHICH
0407                * EX BASIC PUTS IN THE RAM VECTOR (IF ANY) AND THE THIRD ADDRESS
0408                * IS THE VALUE WHICH DISK BASIC PUTS THERE (IF ANY)
0409
0410
0411                *                               2.0    2.1    1.0    1.1
0412 015E          RVEC0      RMB   3               $A5F6          $C426  $C44B  OPEN COMMAND
0413 0161          RVEC1      RMB   3               $A5B9          $C838  $C888  DEVICE NUMBER VALIDITY CHECK
0414 0164          RVEC2      RMB   3               $A35F          $C843  $C893  SET PRINT PARAMETERS
0415 0167          RVEC3      RMB   3               $A282  $8273   $CB4A  $CC1C  CONSOLE OUT
0416 016A          RVEC4      RMB   3               $A176  $8CF1   $C58F  $C5BC  CONSOLE IN
0417 016D          RVEC5      RMB   3               $A3ED          $C818  $C848  INPUT DEVICE NUMBER CHECK
0418 0170          RVEC6      RMB   3               $A406          $C81B  $C84B  PRINT DEVICE NUMBER CHECK
0419 0173          RVEC7      RMB   3               $A426          $CA3B  $CAE9  CLOSE ALL FILES
0420 0176          RVEC8      RMB   3               $A42D  $8286   $CA4B  $CAF9  CLOSE ONE FILE
0421 0179          RVEC9      RMB   3               $B918  $8E90   $8E90  $8E90  PRINT
0422 017C          RVEC10     RMB   3               $B061          $CC5B  $CD35  INPUT
0423 017F          RVEC11     RMB   3               $A549          $C859  $C8A9  BREAK CHECK
0424 0182          RVEC12     RMB   3               $A390          $C6B7  $C6E4  INPUTTING A BASIC LINE
0425 0185          RVEC13     RMB   3               $A4BF          $CA36  $CAE4  TERMINATING BASIC LINE INPUT
0426 0188          RVEC14     RMB   3               $A5CE          $CA60  $C90C  EOF COMMAND
0427 018B          RVEC15     RMB   3               $B223  $8846   $CDF6  $CED2  EVALUATE AN EXPRESSION
0428 018E          RVEC16     RMB   3               $AC46          $C6B7  $C6E4  RESERVED FOR ON ERROR GOTO COMMAND
0429 0191          RVEC17     RMB   3               $AC49  $88F0   $C24D  $C265  ERROR DRIVER
0430 0194          RVEC18     RMB   3               $AE75  $829C   $C990  $CA3E  RUN
0431 0197          RVEC19     RMB   3               $BD22  $87EF                 ASCII TO FLOATING POINT CONVERSION
0432 019A          RVEC20     RMB   3               $AD9E  $82B9          $C8B0  BASIC'S COMMAND INTERPRETATION LOOP
0433 019D          RVEC21     RMB   3               $A8C4                        RESET/SET/POINT COMMANDS
0434 01A0          RVEC22     RMB   3               $A910                        CLS
0435                *                               $8162                        EXBAS' SECONDARY TOKEN HANDLER
0436                *                               $8AFA                        EXBAS' RENUM TOKEN CHECK
0437                *                               $975C          $C29A  $C2B2  EXBAS' GET/PUT
0438 01A3          RVEC23     RMB   3               $B821  $8304                 CRUNCH BASIC LINE
0439 01A6          RVEC24     RMB   3               $B7C2                        UNCRUNCH BASIC LINE
0440
0441 01A9          STRSTK     RMB   8*5             STRING DESCRIPTOR STACK
0442 01D1          CFNBUF     RMB   9               CASSETTE FILE NAME BUFFER
0443 01DA          CASBUF     RMB   256             CASSETTE FILE DATA BUFFER
```

```
0444 02DA           LINHDR     RMB   2               LINE INPUT BUFFER HEADER
0445 02DC           LINBUF     RMB   LBUFMX+1        BASIC LINE INPUT BUFFER
0446 03D7           STRBUF     RMB   41              STRING BUFFER
0447
0448 0400           VIDRAM     RMB   200             VIDEO DISPLAY AREA
0449
0450                *START OF ADDITIONAL RAM VARIABLE STORAGE (DISK BASIC ONLY)
0451 0600           DBUF0      RMB   SECLEN          I/O BUFFER #0
0452 0700           DBUF1      RMB   SECLEN          I/O BUFFER #1
0453 0800           FATBL0     RMB   FATLEN          FILE ALLOCATION TABLE - DRIVE 0
0454 084A           FATBL1     RMB   FATLEN          FILE ALLOCATION TABLE - DRIVE 1
0455 0894           FATBL2     RMB   FATLEN          FILE ALLOCATION TABLE - DRIVE 2
0456 08DE           FATBL3     RMB   FATLEN          FILE ALLOCATION TABLE - DRIVE 3
0457 0928           FCBV1      RMB   16*2            FILE BUFFER VECTORS (15 USER, 1 SYSTEM)
0458 0948           RNBFAD     RMB   2               START OF FREE RANDOM FILE BUFFER AREA
0459 094A           FCBADR     RMB   2               START OF FILE CONTROL BLOCKS
0460 094C           DNAMBF     RMB   8               DISK FILE NAME BUFFER
0461 0954           DEXTBF     RMB   3               DISK FILE EXTENSION NAME BUFFER
0462 0957           DFLTYP     RMB   1               *DV* DISK FILE TYPE: 0=BASIC, 1=DATA, 2=MACHINE
0463                *                                     LANGUAGE, 3=TEXT EDITOR SOURCE FILE
0464 0958           DASCFL     RMB   1               *DV* ASCII FLAG: 0=CRUNCHED OR BINARY, $FF=ASCII
0465 0959           DRUNFL     RMB   1               RUN FLAG: (IF BIT 1=1 THEN RUN, IF BIT 0=1, THEN CLOSE
0466                *                                ALL FILES BEFORE RUNNING)
0467 095A           DEFDRV     RMB   1               DEFAULT DRIVE NUMBER
0468 095B           FCBACT     RMB   1               NUMBER OF FCBS ACTIVE
0469 095C           DRESFL     RMB   1               RESET FLAG: <>0 WILL CAUSE A 'NEW' & SHUT DOWN ALL FCBS
0470 095D           DLOADFL    RMB   1               LOAD FLAG: CAUSE A 'NEW' FOLLOWING A LOAD ERROR
0471 095E           DMRGFL     RMB   1               MERGE FLAG: 0=N0 MERGE, $FF=MERGE
0472 095F           DUSRVC     RMB   20              DISK BASIC USR COMMAND VECTORS
0473                *** DISK FILE WORK AREA FOR DIRECTORY SEARCH
0474                *   EXISTING FILE
0475 0973           V973       RMB   1               SECTOR NUMBER
0476 0974           V974       RMB   2               RAM DIRECTORY IMAGE ADDRESS
0477 0976           V976       RMB   1               FIRST GRANULE NUMBER
0478                *   UNUSED FILE
0479 0977           V977       RMB   1               SECTOR NUMBER
0480 0978           V978       RMB   2               RAM DIRECTORY IMAGE ADDRESS
0481
0482 097A           WFATVL     RMB   2               WRITE FAT VALUE: NUMBER OF FREE GRANULES WHICH MUST BE TAKEN
0483                                                 FROM THE FAT TO TRIGGER A WRITE FAT TO DISK SEQUENCE
0484 097C           DFFLEN     RMB   2               DIRECT ACCESS FILE RECORD LENGTH
0485 097E           DR0TRK     RMB   4               CURRENT TRACK NUMBER, DRIVES 0,1,2,3
0486 0982           NMIFLG     RMB   1               NMI FLAG: 0=DON'T VECTOR <>0=YECTOR OUT
0487 0983           DNMIVC     RMB   2               NMI VECTOR: WHERE TO JUMP FOLLOWING AN NMI
0488                *                                INTERRUPT IF THE NMI FLAG IS SET
0489 0985           RDYTMR     RMB   1               MOTOR TURN OFF TIMER
0490 0986           DRGRAM     RMB   1               RAM IMAGE OF DSKREG ($FF40)
0491 0987           DVERFL     RMB   1               VERIFY FLAG: 0=OFF, $FF=ON
0492 0988           ATTCTR     RMB   1               READ/WRITE ATTEMPT COUNTER: NUMBER OF TIMES THE
0493                *                                DISK WILL ATTEMPT TO RETRIEVE OR WRITE DATA
0494                *                                BEFORE IT GIVES UP AND ISSUES AN ERROR.
0495
0496 0989           DFLBUF     RMB   SECLEN          INITIALIZED TO SECLEN BY DISKBAS
0497
0498                *RANDOM FILE RESERVED AREA
0499
0500                *FILE CONTROL BLOCKS AND BUFFERS
0501
0502                *GRAPHIC PAGE RESERVED AREA
0503
0504                *BASIC PROGRAM
0505
0506                *VARIABLE STORAGE AREA
0507
0508                *ARRAY STORAGE AREA
0509
0510
0511                * FREE MEMORY
0512
0513
0514                *STACK
0515
0516                *STRING SPACE
0517
```

```
0518                *USER PROGRAM RESERVED AREA
0519
0520                *END OF RAM
0521
0522 8000          ORG       $8000
0523
0524 8000                    RMB   $2000           EXTENDED BASIC ROM
0525 A000                    RMB   $2000           COLOR BASIC ROM
0526 C000          ROMPAK    EQU   *
0527 C000          DOSBAS    RMB   $2000           DISK BASIC ROM/ENHANCED BASIC INIT CODE
0528 E000                    RMB   $1F00           ENHANCED BASIC
0529
0530                * START OF ADDITIONAL VARIABLES USED BY SUPER EXTENDED BASIC
0531 FE00          H.CRSLOC  RMB   2              CURRENT LOCATION OF CURSOR
0532 FE02          H.CURSX   RMB   1              X POSITION OF CURSOR
0533 FE03          H.CURSY   RMB   1              Y POSITION OF CURSOR
0534 FE04          H.COLUMN  RMB   1              COLUMNS ON HI-RES ALPHA SCREEN
0535 FE05          H.ROW     RMB   1              ROWS ON HI-RES ALPHA SCREEN
0536 FE06          H.DISPEN  RMB   2              END OF HI-RES DISPLAY SCREEN
0537 FE08          H.CRSATT  RMB   1              CURRENT CURSOR'S ATTRIBUTES
0538 FE09                    RMB   1              UNUSED
0539 FE0A          H.FCOLOR  RMB   1              FOREGROUND COLOR
0540 FE0B          H.BCOLOR  RMB   1              BACKGROUND COLOR
0541 FE0C          H.ONBRK   RMB   2              ON BRK GOTO LINE NUMBER
0542 FE0E          H.ONERR   RMB   2              ON ERR GOTO LINE NUMBER
0543 FE10          H.ERROR   RMB   1              ERROR NUMBER ENCOUNTERED OR $FF (NO ERROR)
0544 FE11          H.ONERRS  RMB   2              ON ERR SOURCE LINE NUMBER
0545 FE13          H.ERLINE  RMB   2              LINE NUMBER WHERE ERROR OCCURRED
0546 FE15          H.ONBRKS  RMB   2              ON BRK SOURCE LINE NUMBER
0547 FE17          H.ERRBRK  RMB   1              STILL UNKNOWN, HAS TO DO WITH ERR, BRK
0548 FE18          H.PCOUNT  RMB   1              PRINT COUNT, CHARACTERS TO BE HPRINTED
0549 FE19          H.PBUF    RMB   80             PRINT BUFFER, HPRINT CHARS. STORED HERE
0550 FE69                    RMB   132            UNUSED
0551 FED           INT.FLAG  RMB   1              INTERRUPT VALID FLAG. 0=NOT VALID, $55=VALID
0552                * TABLE OF JUMP VECTORS TO INTERRUPT SERVICING ROUTINES
0553 FEEE          INT.JUMP
0554 FEEE          INT.SWI3  RMB   3
0555 FEF1          INT.SWI2  RMB   3
0556 FEF4          INT.FIRQ  RMB   3
0557 FEF7          INT.IRQ   RMB   3
0558 FEFA          INT.SWI   RMB   3
0559 FEFD          INT.NMI   RMB   3
0560
0561                * I/O AREA
0562
0563 FF00          PIA0      EQU   *              PERIPHERAL INTERFACE ADAPTER ONE
0564
0565 FF00          BIT0      KEYBOARD ROW 1 AND RIGHT JOYSTICK SWITCH 1
0566               BIT1      KEYBOARD ROW 2 AND LEFT JOYSTICK SWITCH 1
0567               BIT2      KEYBOARD ROW 3 AND RIGHT JOYSTICK SWITCH 2
0568               BIT3      KEYBOARD ROW 4 AND LEFT JOYSTICK SWITCH 2
0569               BIT4      KEYBOARD ROW 5
0570               BIT5      KEYBOARD ROW 6
0571               BIT6      KEYBOARD ROW 7
0572               BIT7      JOTSTICK COMPARISON IINPUT
0573
0574 FF01          BIT0      CONTROL OF HSYNC (63.5ps)    0 = IRQ* TO CPU DISABLED
0575                         INTERRUPT                    1 = IRQ* TO CPU ENABLED
0576               BIT1      CONTROL OF INTERRUPT         0 = FLAG SET ON FALLING EDGE OF HS
0577                         POLARITY                     1 = FLAG SET ON RISING EDGE OF HS
0578               BIT2      NORMALLY 1                   0 = CHANGES FF00 TO DATA DIRECTION
0579               BIT3      SEL 1                        LSB OF TWO ANALOG MUX SELECT LINES
0580               BIT4      ALWAYS 1
0581               BIT5      ALWAYS 1
0582               BIT6      NOT USED
0583               BIT7      HORIZONTAL SYNC INTERRUPT FLAG
0584
0585 FF02          BIT0      KEYBOARD COLUMN 1
0586               BIT1      KEYBOARD COLUMN 2
0587               BIT2      KEYBOARD COLUMN 3
0588               BIT3      KEYBOARD COLUMN 4
0589               BIT4      KEYBOARD COLUMN 5
0590               BIT5      KEYBOARD COLUMN 6
0591               BIT6      KEYBOARD COLUMN 7 / RAM SIZE OUTPUT
```

```
0592                BIT7      KEYBOARD COLUMN 8
0593
0594 FF03           BIT0      CONTROL OF VSYNC (16.667ms)  0 = IRQ* TO CPU DISABLED
0595                          INTERRUPT                    1 = IRQ* TO CPU ENABLED
0596                BIT1      CONTROL OF INTERRUPT         0 = FLAG SET ON FALLING EDGE OF FS
0597                          POLARITY                     1 = FLAG SET ON RISING EDGE OF FS
0598                BIT2      NORMALLY 1                   0 = CHANGES FF02 TO DATA DIRECTION
0599                BIT3      SEL 2                        MSB OF TWO ANALOG MUX SELECT LINES
0600                BIT4      ALWAYS 1
0601                BIT5      ALWAYS 1
0602                BIT6      NOT USED
0603                BIT7      FIELD SYNC INTERRUPT FLAG
0604
0605 FF04                     RMB   28              PIA0 IMAGES
0606 FF20           DA
0607 FF20           PIA1      EQU   *               PERIPHERAL INTERFACE ADAPTER TWO
0608
0609 FF20           BIT0      CASSETTE DATA INPUT
0610                BIT1      RS-232C DATA OUTPUT
0611                BIT2      6 BIT D/A LSB
0612                BIT3      6 BIT D/A
0613                BIT4      6 BIT D/A
0614                BIT5      6 BIT D/A
0615                BIT6      6 BIT D/A
0616                BIT7      6 BIT D/A MSB
0617
0618 FF21           BIT0      CONTROL OF CD                0 = FIRQ* TO CPU DISABLED
0619                          (RS-232C STATUS)             1 = FIRQ* TO CPU ENABLED
0620                BIT1      CONTROL OF INTERRUPT         0 = FLAG SET ON FALLING EDGE OF CD
0621                          POLARITY                     1 = FLAG SET ON RISING EDGE OF CD
0622                BIT2      NORMALLY 1                   0 = CHANGES FF20 TO DATA DIRECTION
0623                BIT3      CASSETTE MOTOR CONTROL       0 = OFF       1 = ON
0624                BIT4      ALWAYS 1
0625                BIT5      ALWAYS 1
0626                BIT6      NOT USED
0627                BIT7      CD INTERRUPT FLAG
0628
0629 FF22           BIT0      RS-232C DATA INPUT
0630                BIT1      SINGLE BIT SOUND OUTPUT
0631                BIT2      RAM SIZE INPUT
0632                BIT3      RGB MONITOR SENSING INPUT    CSS
0633                BIT4      VDG CONTROL OUTPUT           GM0 & UPPER/LOWER CASE*
0634                BIT5      VDG CONTROL OUTPUT           GM1 & INVERT
0635                BIT6      VDG CONTROL OUTPUT           GM2
0636                BIT7      VDG CONTROL OUTPUT           A*/G
0637
0638 FF23           BIT0      CONTROL OF CARTRIDGE         0 = FIRQ* TO CPU DISABLED
0639                          INTERRUPT                    1 = FIRQ* TO CPU ENABLED
0640                BIT1      CONTROL OF INTERRUPT         0 = FLAG SET ON FALLING EDGE OF CART*
0641                          POLARITY                     1 = FLAG SET ON RISING EDGE OF CART*
0642                BIT2      NORMALLY 1                   0 = CHANGES FF22 TO DATA DIRECTION
0643                BIT3      SOUND ENABLE
0644                BIT4      ALWAYS 1
0645                BIT5      ALWAYS 1
0646                BIT6      NOT USED
0647                BIT7      CARTRIDGE INTERRUPT FLAG
0648
0649 FF24                     RMB   28              PIA1 IMAGES
0650 FF40           PIA2
0651 FF40           DSKREG    RMB   1               DISK CONTROL REGISTER
0652
0653 FF40           BIT0      DRIVE SELECT 0
0654                BIT1      DRIVE SELECT 1
0655                BIT2      DRIVE SELECT 2
0656                BIT3      DRIVE MOTOR ENABLE           0 = MOTORS OFF       1 = MOTORS ON
0657                BIT4      WRITE PRECOMPENSATION        0 = NO PRECOMP       1 = PRECOMP
0658                BIT5      DENSITY FLAG                 0 = SINGLE           1 = DOUBLE
0659                BIT6      DRIVE SELECT 3
0660                BIT7      HALT FLAG                    0 = DISABLED         1 = ENABLED
0661
0662 FF41                     RMB   7               DSKREG IMAGES
0663
0664                * FLOPPY DISK CONTROLLER INTERNAL REGISTERS
0665 FF48           FDCREG    RMB   1               STATUS/COMMAND REGISTER
```

```
0666
0667                COMMANDS   TYPE  COMMAND               CODE
0668                           I     RESTORE               $Ø3
0669                           I     SEEK                  $17
0670                           I     STEP                  $23
0671                           I     STEP IN               $43
0672                           I     STEP OUT              $53
0673                           II    READ SECTOR           $8Ø
0674                           II    WRITE SECTOR          $AØ
0675                           III   READ ADDRESS          $CØ
0676                           III   READ TRACK            $E4
0677                           III   WRITE TRACK           $F4
0678                           IV    FORCE INTERRUPT       $DØ
0679
0680                STATUS     BIT   TYPE I          READ ADDRESS/SECTOR/TRACK  WRITE SECTOR/TRACK
0681                           SØ    BUSY            BUSY                       BUSY
0682                           S1    INDEX           DRQ                        DRQ
0683                           S2    TRACK Ø         LOST DATA                  LOST DATA
0684                           S3    CRC ERROR       CRC ERROR (EXCEPT TRACK)   CRC ERROR (EXCEPT TRACK)
0685                           S4    SEEK ERROR      RNF (EXCEPT TRACK)         RNF (EXCEPT TRACK)
0686                           S5    HEAD LOADED     RECORD TYPE (SECTOR ONLY)  WRITE FAULT
0687                           S6    WRITE PROTECT                              WRITE PROTECT
0688                           S7    NOT READY       NOT READY                  NOT READY
0689
0690 FF49                      RMB   1               TRACK REGISTER
0691 FF4A                      RMB   1               SECTOR REGISTER
0692 FF4B                      RMB   1               DATA REGISTER
0693 FF4C                      RMB   4               FDCREG IMAGES
0694
0695 FF5Ø                      RMB   16              UNUSED SPACE
0696 FF6Ø                      RMB   1               X COORDINATE FOR X-PAD
0697 FF61                      RMB   1               Y COORDINATE FOR X-PAD
0698 FF62                      RMB   1               STATUS REGISTER FOR X-PAD
0699 FF63                      RMB   5               UNUSED
0700                * RS-232 PROGRAM PAK
0701 FF68                      RMB   1               READ/WRITE DATA REGISTER
0702 FF69                      RMB   1               STATUS REGISTER
0703 FF6A                      RMB   1               COMMAND REGISTER
0704 FF6B                      RMB   1               CONTROL REGISTER
0705 FF6C                      RMB   4
0706 FF7Ø                      RMB   13
0707 FF7D                      RMB   1               SOUND/SPEECH CARTRIDGE RESET
0708 FF7E                      RMB   1               SOUND/SPEECH CARTRIDGE READ/WRITE
0709 FF7F                      RMB   1               MULTI-PAK PROGRAMMING REGISTER
0710
0711 FF8Ø                      RMB   64              RESERVED FOR FUTURE EXPANSION
0712
0713                * VIDEO CONTROL REGISTERS
0714 FF9Ø          INIT0       RMB   1               INITIALIZATION REGISTER Ø
0715
0716 FF9Ø          BIT0        MCØ                         ROM MAP CONTROL (SEE TABLE BELOW)
0717               BIT1        MC1                         ROM MAP CONTROL (SEE TABLE BELOW)
0718               BIT2        MC2                         STANDARD SCS
0719               BIT3        MC3                         1 = DRAM AT ØxFEXX IS CONSTANT
0720               BIT4        FEN                         1 = CHIP FIRQ OUTPUT ENABLED
0721               BIT5        IEN                         1 = CHIP IRQ OUTPUT ENABLED
0722               BIT6        M/P                         1 = MMU ENABLED
0723               BIT7        COCO                        1 = COCO 1 & 2 COMPATIBLE
0724
0725                           MC1   MCØ             ROM MAPPING
0726                           Ø     x               16K INTERNAL, 16K EXTERNAL
0727                           1     Ø               32K INTERNAL
0728                           1     1               32L EXTERNAL (EXCEPT FOR VECTORS)
0729
0730 FF91          INIT1       RMB   1               INITIALIZATION REGISTER 1
0731
0732 FF91          BIT0        TR                          MMU TASK REGISTER SELECT
0733               BIT1
0734               BIT2
0735               BIT3
0736               BIT4
0737               BIT5        TINS                        TIMER INPUT SELECT: 1=7Øns, Ø=63ns
0738               BIT6
0739               BIT7
```

```
0740
0741
0742 FF92          IRQENR    RMB   1             IRQ INTERRUPT ENABLE REGISTER
0743
0744 FF92          BIT0      EI0                       CARTRIDGE IRQ ENABLED
0745               BIT1      EI1                       KEYBOARD IRQ ENABLED
0746               BIT2      EI2                       SERIAL DATA IRQ ENABLED
0747               BIT3      VBORD                     VERTICAL BORDER IRQ ENABLED
0748               BIT4      HBORD                     HORIZONTAL BORDER IRQ ENABLED
0749               BIT5      TMR                       INTERRUPT FROM TIMER ENABLED
0750               BIT6
0751               BIT7
0752
0753 FF93          FIRQENR   RMB   1             FIRQ INTERRUPT ENABLE REGISTER
0754
0755 FF93          BIT0      EI0                       CARTRIDGE FIRQ ENABLED
0756               BIT1      EI1                       KEYBOARD FIRQ ENABLED
0757               BIT2      EI2                       SERIAL DATA FIRQ ENABLED
0758               BIT3      VBORD                     VERTICAL BORDER FIRQ ENABLED
0759               BIT4      HBORD                     HORIZONTAL BORDER FIRQ ENABLED
0760               BIT5      TMR                       INTERRUPT FROM TIMER ENABLED
0761               BIT6
0762               BIT7
0763
0764 FF94          V.TIMER   RMB   2             TIMER REGISTER
0765 FF96                    RMB   2             RESERVED FOR FUTURE EXPANSION
0766
0767 FF98          VIDEOREG  RMB   1             VIDEO MODE REGISTER
0768
0769 FF98          BIT0      LPR0                      LINES PER ROW (SEE TABLE BELOW)
0770               BIT1      LPR1                      LINES PER ROW (SEE TABLE BELOW)
0771               BIT2      LPR2                      LINES PER ROW (SEE TABLE BELOW)
0772               BIT3      H50                       1 = 50 Hz VERTICAL REFRESH
0773               BIT4      MOCH                      1 = MONOCHROME (ON COMPOSITE)
0774               BIT5      BPI                       1 = BURST PHASE INVERTED
0775               BIT6
0776               BIT7      BP                        0 = ALPHA, 1 = BIT PLANE
0777
0778               LPR2      LPR1  LPR0          LINES PER CHARACTER ROW
0779               0         0     0             1            (GRAPHICS MODES)
0780               0         0     1             2            (COCO 1 & 2 ONLY)
0781               0         1     0             3            (COCO 1 & 2 ONLY)
0782               0         1     1             8
0783               1         0     0             9
0784               1         0     1             (RESERVED)
0785               1         1     0             12
0786               1         1     1             (RESERVED)
0787
0788 FF99          VIDEOREG  RMB   1             VIDEO MODE REGISTER
0789
0790 FF99          BIT0      CRES0                     COLOR RESOLUTION
0791               BIT1      CRES1                     COLOR RESOLUTION
0792               BIT2      HRES0                     HORIZONTAL RESOLUTION
0793               BIT3      HRES1                     HORIZONTAL RESOLUTION
0794               BIT4      HRES2                     HORIZONTAL RESOLUTION
0795               BIT5      LPF0                      LINES PER FIELD (SEE TABLE BELOW)
0796               BIT6      LPF1                      LINES PER FIELD (SEE TABLE BELOW)
0797               BIT7
0798
0799                         LPF1  LPF0          LINES PER FIELD
0800                         0     0             192
0801                         0     1             200
0802                         1     0             RESERVED
0803                         1     1             225
0804
0805               * VIDEO RESOLUTION
0806                         ALPHA: BP = 0, COCO = 0
0807                              MODE           HRES2  HRES1  HRES0  CRES1  CRES0
0808                              32 CHARACTER   0             0             1
0809                              40 CHARACTER   0             1             1
0810                              80 CHARACTER   1             1             1
0811                         GRAPHICS: BP = 1, COCO = 0
0812                              PIXELSxCOLORS  HRES2  HRES1  HRES0  CRES1  CRES0
0813                              640x4          1      1      1      0      1
```

```
                              640x2          1     Ø     1     Ø     Ø
                              512x4          1     1     Ø     Ø     1
                              512x2          1     Ø     Ø     Ø     Ø
                              32Øx16         1     1     1     1     Ø
                              32Øx4          1     Ø     1     Ø     1
                              256x16         1     1     Ø     1     Ø
                              256x4          1     Ø     Ø     Ø     1
                              256x2          Ø     1     Ø     Ø     Ø
                              16Øx16         1     Ø     1     1     Ø

          * COCO MODE SELECTION
                                        DISPLAY MODE        REG. FF22
                                        V2    V1    VØ      7     6     5     4     3
          ALPHA                         Ø     Ø     Ø       Ø     x     x     Ø     CSS
          ALPHA INVERTED                Ø     Ø     Ø       Ø     x     x     Ø     CSS
          SEMIGRAPHICS 4                Ø     Ø     Ø       Ø     x     x     Ø     x
          64x64 COLOR GRAPHICS          Ø     Ø     1       1     Ø     Ø     Ø     CSS
          128x64 GRAPHICS               Ø     Ø     1       1     Ø     Ø     1     CSS
          128x64 COLOR GRAPHICS         Ø     1     Ø       1     Ø     1     Ø     CSS
          128x96 GRAPHICS               Ø     1     1       1     Ø     1     1     CSS
          128x96 COLOR GRAPHICS         1     Ø     Ø       1     1     Ø     Ø     CSS
          128x96 GRAPHICS               1     Ø     1       1     1     Ø     1     CSS
          128x96 COLOR GRAPHICS         1     1     Ø       1     1     1     Ø     CSS
          256x192 GRAPHICS              1     1     Ø       1     1     1     1     CSS

          * ALPHANUMERIC MODES
                    TEXT SCREEN MEMORY
                         EVEN BYTE
                         BITØ           CHARACTER BIT Ø
                         BIT1           CHARACTER BIT 1
                         BIT2           CHARACTER BIT 2
                         BIT3           CHARACTER BIT 3
                         BIT4           CHARACTER BIT 4
                         BIT5           CHARACTER BIT 5
                         BIT6           CHARACTER BIT 6
                         BIT7

                         ODD BYTE
                         BITØ           BGNDØ  BACKGROUND COLOR BIT (PALETTE ADDR)
                         BIT1           BGND1  BACKGROUND COLOR BIT (PALETTE ADDR)
                         BIT2           BGND2  BACKGROUND COLOR BIT (PALETTE ADDR)
                         BIT3           FGBDØ  FOREGROUND COLOR BIT (PALETTE ADDR)
                         BIT4           FGND1  FOREGROUND COLOR BIT (PALETTE ADDR)
                         BIT5           FGND2  FOREGROUND COLOR BIT (PALETTE ADDR)
                         BIT6           UNDLN  CHARACTERS ARE UNDERLINED
                         BIT7           BLINK  CHARACTERS BLINK AT 1/2 SEC. RATE
                    * ATTRIBUTES NOT AVAILABLE WHEN COCO = 1
          * GRAPHICS MODES
                    16 COLOR MODES: (CRES1=1, CRESØ = Ø)
                         BYTE FROM DRAM
                         BITØ           PAØ, SECOND PIXEL
                         BIT1           PA1, SECOND PIXEL
                         BIT2           PA2, SECOND PIXEL
                         BIT3           PA3, SECOND PIXEL
                         BIT4           PAØ, FIRST PIXEL
                         BIT5           PA1, FIRST PIXEL
                         BIT6           PA2, FIRST PIXEL
                         BIT7           PA3, FIRST PIXEL
                    4 COLOR MODES: (CRES1=Ø, CRESØ = 1)
                         BYTE FROM DRAM
                         BITØ           PAØ, FOURTH PIXEL
                         BIT1           PA1, FOURTH PIXEL
                         BIT2           PAØ, THIRD PIXEL
                         BIT3           PA1, THIRD PIXEL
                         BIT4           PAØ, SECOND PIXEL
                         BIT5           PA1, SECOND PIXEL
                         BIT6           PAØ, FIRST PIXEL
                         BIT7           PA1, FIRST PIXEL
                    2 COLOR MODES: (CRES1=Ø, CRESØ = Ø)
                         BYTE FROM DRAM
                         BITØ           PAØ, EIGHTH PIXEL
                         BIT1           PAØ, SEVENTH PIXEL
                         BIT2           PAØ, SIXTH PIXEL
                         BIT3           PAØ, FIFTH PIXEL
```

```
0888                                BIT4            PA0, FORTH PIXEL
0889                                BIT5            PA0, THIRD PIXEL
0890                                BIT6            PA0, SECOND PIXEL
0891                                BIT7            PA0, FIRST PIXEL
0892                * PALETTE ADDRESSES
0893                                ADDRESS         PA3    PA2    PA1    PA0
0894                                FFB0            0      0      0      0
0895                                FFB1            0      0      0      1
0896                                FFB2            0      0      1      0
0897                                FFB3            0      0      1      1
0898                                FFB4            0      1      0      0
0899                                FFB5            0      1      0      1
0900                                FFB6            0      1      1      0
0901                                FFB7            0      1      1      1
0902                                FFB8            1      0      0      0
0903                                FFB9            1      0      0      1
0904                                FFBA            1      0      1      0
0905                                FFBB            1      0      1      1
0906                                FFBC            1      1      0      0
0907                                FFBD            1      1      0      1
0908                                FFBE            1      1      1      0
0909                                FFBF            1      1      1      1
0910
0911 FF9A           V.BORDER   RMB   1              BORDER REGISTER
0912
0913 FF9A           BIT0       BLU0                        BLUE LSB
0914                BIT1       GRN0                        GREEN LSB
0915                BIT2       RED0                        RED LSB
0916                BIT3       BLU1                        BLUE MSB
0917                BIT4       GRN1                        GREEN MSB
0918                BIT5       RED1                        RED MSB
0919                BIT6
0920                BIT7
0921
0922 FF9B                      RMB   1              RESERVED
0923 FF9C           V.SCROLL   RMB   1              VERTICAL SCROLL REGISTER
0924
0925 FF9C           BIT0       VSC0
0926                BIT1       VSC1
0927                BIT2       VSC2
0928                BIT3       VSC3
0929                BIT4
0930                BIT5
0931                BIT6
0932                BIT7
0933                * IN COCO MODE, THE VSC'S MUST BE INITIALIZED TO $0F
0934
0935 FF9D           V.OFSET1   RMB   1              VERTICAL OFFSET 1 REGISTER
0936
0937 FF9D           BIT0       Y11
0938                BIT1       Y12
0939                BIT2       Y13
0940                BIT3       Y14
0941                BIT4       Y15
0942                BIT5       Y16
0943                BIT6       Y17
0944                BIT7       Y18
0945
0946 FF9E           V.OFSET0   RMB   1              VERTICAL OFFSET 0 REGISTER
0947
0948 FF9E           BIT0       Y3
0949                BIT1       Y4
0950                BIT2       Y5
0951                BIT3       Y6
0952                BIT4       Y7
0953                BIT5       Y8
0954                BIT6       Y9
0955                BIT7       Y10
0956                * IN COCO MODE, Y9-Y15 ARE NOT EFFECTIVE, AND ARE CONTROLLED BY
0957                  SAM BITS F0-F6. ALSO IN COCO MODE, Y16-Y18 SHOULD BE 1, ALL OTHERS 0
0958
0959 FF9F           H.OFSET0   RMB   1              HORIZONTAL OFFSET 0 REGISTER
0960
0961 FF9F           BIT0       X0                  HORIZONTAL OFFSET ADDRESS
```

```
0962                BIT1      X1                  HORIZONTAL OFFSET ADDRESS
0963                BIT2      X2                  HORIZONTAL OFFSET ADDRESS
0964                BIT3      X3                  HORIZONTAL OFFSET ADDRESS
0965                BIT4      X4                  HORIZONTAL OFFSET ADDRESS
0966                BIT5      X5                  HORIZONTAL OFFSET ADDRESS
0967                BIT6      X6                  HORIZONTAL OFFSET ADDRESS
0968                BIT7      XVEN                HORIZONTAL VIRTUAL ENABLE
0969                * HVEN ENABLES A HORIZONTAL SCREEN WIDTH OF 128 BYTES REGARDLESS OF THE
097Ø                  HRES BITS AND CRES BITS SELECTED. THIS WILL ALLOW A 'VIRTUAL' SCREEN
0971                  SOMEWHAT LARGER THAN THE DISPLAYED SCREEN.  THE USER CAN MOVE THIS
0972                  'WINDOW' (THE DISPLAYED SCREEN) BY MEANS OF THE HORIZONTAL OFFSET
0973                  BITS.  IN CHARACTER MODE, THE SCREEN WIDTH IS 128 CHARACTERS REGARDLESS
0974                  OF ATTRIBUTE (OR 64, IF DOUBLE-WIDE IS SELECTED).
0975
0976 FFAØ          MMUREG    RMB   16              MEMORY MANAGEMENT UNIT REGISTERS (6 BITS)
0977
0978                * RELATIONSHIP BETWEEN DATA IN TASK REGISTER AND GENERATED ADDRESS
0979                                BIT             D5     D4     D3     D2     D1     DØ
098Ø                                CORRESPONDING
0981                                MEMORY ADDRESS  A18    A17    A16    A15    A14    A13
0982
0983                * DATA FROM THE MMU IS THEN USED AS THE UPPER 6 ADDRESS LINES (A13-A18)
0984                  FOR MEMORY ACCESS
0985                                ADDRESS RANGE   TR     A15    A14    A13    MMU LOCATION
0986                                XØØØØ - X1FFF   Ø      Ø      Ø      Ø      FFAØ
0987                                X2ØØØ - X3FFF   Ø      Ø      Ø      1      FFA1
0988                                X4ØØØ - X5FFF   Ø      Ø      1      Ø      FFA2
0989                                X6ØØØ - X7FFF   Ø      Ø      1      1      FFA3
099Ø                                X8ØØØ - X9FFF   Ø      1      Ø      Ø      FFA4
0991                                XAØØØ - XBFFF   Ø      1      Ø      1      FFA5
0992                                XCØØØ - XDFFF   Ø      1      1      Ø      FFA6
0993                                XEØØØ - XFFFF   Ø      1      1      1      FFA7
0994
0995                                XØØØØ - X1FFF   1      Ø      Ø      Ø      FFA8
0996                                X2ØØØ - X3FFF   1      Ø      Ø      1      FFA9
0997                                X4ØØØ - X5FFF   1      Ø      1      Ø      FFAA
0998                                X6ØØØ - X7FFF   1      Ø      1      1      FFAB
0999                                X8ØØØ - X9FFF   1      1      Ø      Ø      FFAC
1ØØØ                                XAØØØ - XBFFF   1      1      Ø      1      FFAD
1Ø01                                XCØØØ - XDFFF   1      1      1      Ø      FFAE
1Ø02                                XEØØØ - XFFFF   1      1      1      1      FFAF
1Ø03
1Ø04 FFBØ          PALETREG  RMB   16              COLOR PALETTE REGISTERS (6 BITS)
1Ø05
1Ø06                                DATA BIT        D5     D4     D3     D2     D1     DØ
1Ø07                                RGB OUTPUT      R1     G1     B1     RØ     GØ     BØ
1Ø08                                COMP. OUTPUT    I1     IØ     P3     P2     P1     PØ
1Ø09
1Ø10                * FOR COCO COMPATIBILITY, THE FOLLOWING SHOULD BE LOADED ON INITIALIZATION
1Ø11                  (RGB VALUES) FOR PAL VERSION, IGNORE TABLE FOR COMPOSITE
1Ø12                          FFBØ  GREEN           $12
1Ø13                          FFB1  YELLOW          $36
1Ø14                          FFB2  BLUE            $Ø9
1Ø15                          FFB3  RED             $24
1Ø16                          FFB4  BUFF            $3F
1Ø17                          FFB5  CYAN            $1Ø
1Ø18                          FFB6  MAGENTA         $2D
1Ø19                          FFB7  ORANGE          $26
1Ø2Ø                          FFB8  BLACK           $ØØ
1Ø21                          FFB9  GREEN           $12
1Ø22                          FFBA  BLACK           $ØØ
1Ø23                          FFBB  BUFF            $3F
1Ø24                          FFBC  BLACK           $ØØ
1Ø25                          FFBD  GREEN           $12
1Ø26                          FFBE  BLACK           $ØØ
1Ø27                          FFBF  ORANGE          $26
1Ø28
1Ø29 FFCØ          SAMREG    EQU   *               SAM CONTROL REGISTERS
1Ø3Ø
1Ø31 FFCØ          VØCLR     RMB   1               CLEAR COCO GRAPHICS MODE VØ
1Ø32 FFC1          VØSET     RMB   1               SET COCO GRAPHICS MODE VØ
1Ø33 FFC2          V1CLR     RMB   1               CLEAR COCO GRAPHICS MODE V1
1Ø34 FFC3          V1SET     RMB   1               SET COCO GRAPHICS MODE V1
1Ø35 FFC4          V2CLR     RMB   1               CLEAR COCO GRAPHICS MODE V2
```

```
1036 FFC5           V2SET      RMB   1              SET COCO GRAPHICS MODE V2
1037 FFC6           FØCLR      RMB   1              CLEAR COCO GRAPHICS OFFSET FØ
1038 FFC7           FØSET      RMB   1              SET COCO GRAPHICS OFFSET FØ
1039 FFC8           F1CLR      RMB   1              CLEAR COCO GRAPHICS OFFSET F1
1040 FFC9           F1SET      RMB   1              SET COCO GRAPHICS OFFSET F1
1041 FFCA           F2CLR      RMB   1              CLEAR COCO GRAPHICS OFFSET F2
1042 FFCB           F2SET      RMB   1              SET COCO GRAPHICS OFFSET F2
1043 FFCC           F3CLR      RMB   1              CLEAR COCO GRAPHICS OFFSET F3
1044 FFCD           F3SET      RMB   1              SET COCO GRAPHICS OFFSET F3
1045 FFCE           F4CLR      RMB   1              CLEAR COCO GRAPHICS OFFSET F4
1046 FFCF           F4SET      RMB   1              SET COCO GRAPHICS OFFSET F4
1047 FFDØ           F5CLR      RMB   1              CLEAR COCO GRAPHICS OFFSET F5
1048 FFD1           F5SET      RMB   1              SET COCO GRAPHICS OFFSET F5
1049 FFD2           F6CLR      RMB   1              CLEAR COCO GRAPHICS OFFSET F6
1050 FFD3           F6SET      RMB   1              SET COCO GRAPHICS OFFSET F6
1051 FFD4                      RMB   4              RESERVED
1052 FFD8           R1CLR      RMB   1              CLEAR CPU RATE, (Ø.89 MHz)
1053 FFD9           R1SET      RMB   1              SET CPU RATE, (1.78 MHz)
1054 FFDA                      RMB   4              RESERVED
1055 FFDE           ROMCLR     RMB   1              ROM DISABLED
1056 FFDF           ROMSET     RMB   1              ROM ENABLED
1057
1058 FFEØ                      RMB   18             RESERVED FOR FUTURE MPU ENHANCEMENTS
1059                *          INTERRUPT VECTORS
1060 FFF2           SWI3       RMB   2
1061 FFF4           SWI2       RMB   2
1062 FFF6           FIRQ       RMB   2
1063 FFF8           IRQ        RMB   2
1064 FFFA           SWI        RMB   2
1065 FFFC           NMI        RMB   2
1066 FFFE           RESETV     RMB   2
```

```
0001        00 E0            DHITOK   EQU  $E1                          HIGHEST 1.1 DISK TOKEN
0002        00 32            CYEAR    EQU  '2'
0003                         *
0004                         *
0005                         *
0006                         **
0007                         **** FILE ALLOCATION TABLE FORMAT
0008                         **
0009                         *
0010                         * THE FILE ALLOCATION TABLE (FAT) CONTAINS THE STATUS OF THE GRANULES ON A DISKETTE.
0011                         * THE FAT CONTAINS 6 CONTROL BYTES FOLLOWED BY 68 DATA BYTES (ONE PER GRANULE). ONLY THE
0012                         * FIRST TWO OF THE SIX CONTROL BYTES ARE USED. A VALUE OF $FF IS SAVED IN UNALLOCATED
0013                         * GRANULES. IF BITS 6 & 7 OF THE DATA BYTE ARE SET, THE GRANULE IS THE LAST GRANULE
0014                         * IN A FILE AND BITS 0-5 ARE THE NUMBER OF USED SECTORS IN THAT GRANULE. IF BITS 6 & 7
0015                         * ARE NOT SET, THE DATA BYTE CONTAINS THE NUMBER OF THE NEXT GRANULE IN THE FILE.
0016
0017                         * OFFSETS TO FAT CONTROL BYTES
0018        00 00            FAT0     EQU  0                           ACTIVE FILE COUNTER : DISK TO RAM FAT IMAGE DISABLE
0019        00 01            FAT1     EQU  1                           VALID DATA FLAG: 0=DISK DATA VALID, <> 0 = NEW FAT
0020                         *                                         DATA - DISK DATA INVALID
0021                         *            2 TO 5                       NOT USED
0022        00 06            FATCON   EQU  6                           OFFSET TO START OF FAT DATA (68 BYTES)
0023                         *
0024                         **
0025                         **** DIRECTORY ENTRY FORMAT
0026                         **
0027                         *
0028                         * THE DIRECTORY IS USED TO KEEP TRACK OF HOW MANY FILES ARE STORED ON A DISKETTE
0029                         * AND WHERE THE FILE IS STORED ON THE DISK. THE FIRST GRANULE USED BY THE FILE WILL
0030                         * ALLOW THE FAT TO TRACK DOWN ALL OF THE GRANULES USED BY THE FILE. IF THE FIRST
0031                         * BYTE OF THE DIRECTORY ENTRY IS ZERO, THE FILE HAS BEEN KILLED;
0032                         * IF THE FIRST BYTE IS $FF THEN THE DIRECTORY ENTRY HAS NEVER BEEN USED.
0033                         *
0034                         *            BYTE                          DESCRIPTION
0035
0036        00 00            DIRNAM   EQU  0                           FILE NAME
0037        00 08            DIREXT   EQU  8                           FILE EXTENSION
0038        00 0B            DIRTYP   EQU  11                          FILE TYPE
0039        00 0C            DIRASC   EQU  12                          ASCII FLAG
0040        00 0D            DIRGRN   EQU  13                          FIRST GRANULE IN FILE
0041        00 0E            DIRLST   EQU  14                          NUMBER OF BYTES IN LAST SECTOR
0042                         *            16 TO 31                     UNUSED
0043                         *
0044                         **
0045                         **** FILE CONTROL BLOCK FORMAT
0046                         **
0047                         *
0048                         * THE FILE STRUCTURE OF COLOR TRS DOS IS CONTROLLED BY A FILE CONTROL BLOCK (FCB)
0049                         * THE FCB CONTAINS 25 CONTROL BYTES AND A SECTOR LONG (256 BYTES) DATA BUFFER.
0050                         * THE CONTROL BYTES CONTROL THE ORDERLY FLOW OF DATA FROM THE COMPUTER'S RAM TO
0051                         * THE DISKETTE AND VICE VERSA. THE OPEN COMMAND INITIALIZES THE FCB; THE INPUT,
0052                         * OUTPUT, WRITE, PRINT, GET AND PUT COMMANDS TRANSFER DATA THROUGH THE FCB AND
0053                         * THE CLOSE COMMAND TURNS OFF THE FCB.
0054
0055                         * TABLES OF OFFSETS TO FCB CONTROL BYTES
0056
0057                         ***** RANDOM FILE
0058                         *            BYTE                          DESCRIPTION
0059        00 00            FCBTYP   EQU  0                           FILE TYPE: $40=RANDOM/DIRECT, 0=CLOSED
0060        00 01            FCBDRV   EQU  1                           DRIVE NUMBER
0061        00 02            FCBFGR   EQU  2                           FIRST GRANULE IN FILE
0062        00 03            FCBCGR   EQU  3                           CURRENT GRANULE BEING USED
0063        00 04            FCBSEC   EQU  4                           CURRENT SECTOR BEING USED (1-9)
0064                         *            5                            UNUSED
0065        00 06            FCBPOS   EQU  6                           CURRENT PRINT POSITION - ALWAYS ZERO IN RANDOM FILES
0066        00 07            FCBREC   EQU  7                           CURRENT RECORD NUMBER
0067        00 09            FCBRLN   EQU  9                           RANDOM FILE RECORD LENGTH
0068        00 0B            FCBBUF   EQU  11                          POINTER TO START OF THIS FILE'S RANDOM ACCESS BUFFER
0069        00 0D            FCBSOF   EQU  13                          SECTOR OFFSET TO CURRENT POSITION IN RECORD
0070        00 0F            FCBFLG   EQU  15                          GET/PUT FLAG: 0=PUT, 1=PUT
0071                         *            16,17                        NOT USED
0072        00 12            FCBDIR   EQU  18                          DIRECTORY ENTRY NUMBER (0-71)
0073        00 13            FCBLST   EQU  19                          NUMBER OF BYTES IN LAST SECTOR OF FILE
0074        00 15            FCBGET   EQU  21                          'GET' RECORD COUNTER: HOW MANY CHARACTERS HAVE BEEN
0075                         *                                         PULLED OUT OF THE CURRENT RECORD
0076        00 17            FCBPUT   EQU  23                          'PUT' RECORD COUNTER: POINTER TO WHERE IN THE RECORD THE NEXT
0077                         *                                         BYTE WILL BE 'PUT'
0078        00 19            FCBCON   EQU  25                          OFFSET TO START OF FCB DATA BUFFER (256 BYTES)
0079
0080                         ***** SEQUENTIAL FILE
0081                         *            BYTE                          DESCRIPTION
0082        00 00            FCBTYP   EQU  0                           FILE TYPE: $10=INPUT, $20=OUTPUT, 0=CLOSED
0083        00 01            FCBDRV   EQU  1                           DRIVE NUMBER
0084        00 02            FCBFGR   EQU  2                           FIRST GRANULE IN FILE
0085        00 03            FCBCGR   EQU  3                           CURRENT GRANULE BEING USED
0086        00 04            FCBSEC   EQU  4                           CURRENT SECTOR BEING USED (1-9)
0087        00 05            FCBCPT   EQU  5                           INPUT FILE: CHARACTER POINTER - POINTS TO NEXT CHARACTER IN
0088                         *                                         FILE TO BE PROCESSED.
0089                         *                                         OUTPUT FILE: FULL SECTOR FLAG - IF IT IS 1 WHEN THE FILE IS
0090                         *                                         CLOSED IT MEANS 256 BYTES OF THE LAST SECTOR HAVE BEEN USED.
0091        00 06            FCBPOS   EQU  6                           CURRENT PRINT POSITION
0092        00 07            FCBREC   EQU  7                           CURRENT RECORD NUMBER: HOW MANY WHOLE SECTORS HAVE BEEN
0093                         *                                         INPUT OR OUTPUT TO A FILE.
0094                         *            9 TO 15                      UNUSED
0095        00 10            FCBRLN   EQU  16                          CACHE FLAG: 00=CACHE EMPTY, $FF=CACHE FULL
0096        00 11            FCBCDT   EQU  17                          CACHE DATA BYTE
```

```
0097        00 12            FCBDIR  EQU   18                          DIRECTORY ENTRY NUMBER (0-71)
0098        00 13            FCBLST  EQU   19                          NUMBER OF BYTES IN LAST SECTOR OF FILE
0099                         *               21,22                     UNUSED
0100        00 17            FCBDFL  EQU   23                          INPUT FILE ONLY: DATA LEFT FLAG: 0=DATA LEFT, $FF=NO DATA (EMPTY)
0101        00 18            FCBLFT  EQU   24                          NUMBER OF CHARACTERS LEFT IN BUFFER (INPUT FILE)
0102                         *                                         NUMBER OF CHARS STORED IN BUFFER (OUTPUT FILE)
0103        00 19            FCBCON  EQU   25                          OFFSET TO FCB DATA BUFFER (256 BYTES)
0104
0105                                 ORG   $C000
0106
0107   C000 44 4B            DOSBAS  FCC   'DK'
0108   C002 20 08            LC002   BRA   LC00C
0109
0110   C004 D7 5F            DCNVEC  FDB   DSKCON                       DSKCON POINTER
0111   C006 00 EA            DSKVAR  FDB   DCOPC                        ADDRESS OF DSKCON VARIABLES
0112   C008 DF 4C            DSINIT  FDB   DOSINI                       DISK INITIALIZATION VECTOR
0113   C00A DF 00            DOSVEC  FDB   DOSCOM                       DOS COMMAND VECTOR
0114
0115                         **** ZERO OUT THE RAM USED BY DISK BASIC
0116   C00C 8E 06 00         LC00C   LDX   #DBUF0                       POINT X TO START OF DISK RAM
0117   C00F 6F 80            LC00F   CLR   ,X+                          CLEAR A BYTE
0118   C011 8C 09 89                 CMPX  #DFLBUF                     END OF DISK'S RAM?
0119   C014 26 F9                    BNE   LC00F                       NO - KEEP CLEARING
0120   C016 8E C1 09                 LDX   #LC109                      POINT X TO ROM IMAGE OF COMMAND INTERPRETATION TABLE
0121   C019 CE 01 34                 LDU   #COMVEC+20                  POINT U TO RAM ADDRESS OF SAME
0122   C01C C6 0A                    LDB   #10                         10 BYTES PER TABLE
0123   C01E BD A5 9A                 JSR   LA59A                       MOVE (B) BYTES FROM (X) TO (U)
0124   C021 CC B2 77                 LDD   #LB277                      SYNTAX ERROR ADDRESS
0125   C024 ED 43                    STD   $03,U                       * SET JUMP TABLE ADDRESSES OF THE USER COMMAND
0126   C026 ED 48                    STD   $08,U                       * INTERPRETATION TABLE TO POINT TO SYNTAX ERROR
0127   C028 6F C4                    CLR   ,U                          CLEAR BYTE 0 OF USER TABLE (DOESN'T EXIST FLAG)
0128   C02A 6F 45                    CLR   $05,U                       SET NUMBER OF SECONDARY USER TOKENS TO ZERO
0129   C02C CC CF 0A                 LDD   #DXCVEC                     * SAVE NEW
0130   C02F FD 01 2D                 STD   COMVEC+13                   * POINTERS TO EXBAS
0131   C032 CC CF 32                 LDD   #DXIVEC                     * COMMAND AND SECONDARY
0132   C035 FD 01 32                 STD   COMVEC+18                   * COMMAND INTERPRETATION ROUTINES
0133                         **** MOVE THE NEW RAM VECTORS FROM ROM TO RAM
0134   C038 CE 01 5E                 LDU   #RVEC0                      POINT U TO 1ST RAM VECTOR
0135   C03B 86 7E            LC03B   LDA   #$7E                        OP CODE OF JMP INSTRUCTION
0136   C03D B7 01 A0                 STA   RVEC22                      SET 1ST BYTE OF 'GET'/'PUT' RAM VECTOR TO 'JMP'
0137   C040 A7 C0                    STA   ,U+                         SET 1ST BYTE OF RAM VECTOR TO 'JMP'
0138   C042 EC 81                    LDD   ,X++                        GET RAM VECTOR FROM ROM
0139   C044 ED C1                    STD   ,U++                        STORE IT IN RAM
0140   C046 8C C1 39                 CMPX  #LC139                      COMPARE TO END OF ROM VALUES
0141   C049 26 F0                    BNE   LC03B                       BRANCH IF NOT ALL VECTORS MOVED
0142   C04B 8E C2 B2                 LDX   #DVEC22                     GET ROM VALUE OF 'GET'/'PUT' RAM VECTOR
0143   C04E BF 01 A1                 STX   RVEC22+1                    SAVE IT IN RAM
0144   C051 8E C8 B0                 LDX   #DVEC20                     GET DISK COMMAND INTERPRETATION LOOP RAM VECTOR
0145   C054 BF 01 9B                 STX   RVEC20+1                    SAVE IN RAM VECTOR TABLE
0146                         **** INITIALIZE DISK BASIC'S USR VECTORS
0147   C057 8E 09 5F                 LDX   #DUSRVC                     POINT X TO START OF DISK BASIC USR VECTORS
0148   C05A 9F B0                    STX   USRADR                      SAVE START ADDRESS IN USRADR
0149   C05C CE B4 4A                 LDU   #LB44A                      POINT U TO ADDRESS OF 'FUNCTION CALL' ERROR
0150   C05F C6 0A                    LDB   #$0A                        10 USER VECTORS TO INITIALIZE
0151   C061 EF 81            LC061   STU   ,X++                        SET USR VECTOR TO 'FC' ERROR
0152   C063 5A                       DECB                              DECREMENT USR VECTOR COUNTER
0153   C064 26 FB                    BNE   LC061                       BRANCH IN NOT DONE WITH ALL 10 VECTORS
0154   C066 8E D8 A1                 LDX   #DNMISV                     GET ADDRESS OF NMI SERVICING ROUTINE
0155   C069 BF 01 0A                 STX   NMIVEC+1                    SAVE IT IN NMI VECTOR
0156   C06C 86 7E                    LDA   #$7E                        OP CODE OF JMP
0157   C06E B7 01 09                 STA   NMIVEC                      MAKE THE NMI VECTOR A JMP
0158   C071 8E D8 AF                 LDX   #DIRQSV                     GET ADDRESS OF DISK BASIC IRQ SERVICING ROUTINE
0159   C074 BF 01 0D                 STX   IRQVEC+1                    SAVE IT IN IRQVEC
0160   C077 86 13                    LDA   #$13                        = INITIALIZE WRITE FAT
0161   C079 B7 09 7A                 STA   WFATVL                      = TO DISK TRIGGER VALUE
0162   C07C 7F 08 00                 CLR   FATBL0                      *
0163   C07F 7F 08 4A                 CLR   FATBL1                      * INITIALIZE THE ACTIVE FILE COUNTER OF
0164   C082 7F 08 94                 CLR   FATBL2                      * EACH FAT TO ZERO. THIS WILL CAUSE THE FATS
0165   C085 7F 08 DE                 CLR   FATBL3                      * TO THINK THERE ARE NO ACTIVE FILES
0166   C088 8E 09 89                 LDX   #DFLBUF                     = GET THE STARTING ADDRESS OF THE
0167   C08B BF 09 48                 STX   RNBFAD                      = RANDOM FILE BUFFER FREE AREA AND DAVE IT AS THE
0168                         *                                         = START ADDRESS OF FREE RAM FOR RANDOM FILE BUFFERS
0169   C08E 30 89 01 00              LEAX  $0100,X                     SAVE 256 BYTES FOR RANDOM FILE BUFFERS INITIALLY
0170   C092 BF 09 4A                 STX   FCBADR                      SAVE START ADDRESS OF FCBS
0171   C095 30 01                    LEAX  $01,X                       * ADD ONE AND SAVE THE STARTING
0172   C097 BF 09 28                 STX   FCBV1                       * ADDRESS OF FCB1
0173   C09A 6F 00                    CLR   FCBTYP,X                    CLEAR THE FIRST BYTE OF FCB 1 (CLOSE FCB)
0174   C09C 30 89 01 19              LEAX  FCBLEN,X                    POINT X TO FCB 2
0175   C0A0 BF 09 2A                 STX   FCBV1+2                     SAVE ITS STARTING ADDRESS IN FCB VECTOR TABLE
0176   C0A3 6F 00                    CLR   FCBTYP,X                    CLEAR THE FIRST BYTE OF FCB 2 (CLOSE FCB)
0177   C0A5 30 89 01 19              LEAX  FCBLEN,X                    * POINT X TO SYSTEM FCB - THIS FCB WILL ONLY
0178                         *                                         * BE USED TO COPY, LOAD, SAVE, MERGE, ETC
0179   C0A9 BF 09 2C                 STX   FCBV1+4                     SAVE ITS ADDRESS IN THE FCB VECTOR TABLE
0180   C0AC 6F 00                    CLR   FCBTYP,X                    CLEAR THE FIRST BYTE OF SYSTEM FCB (CLOSE FCB)
0181   C0AE 86 02                    LDA   #$02                        * SET THE NUMBER OF ACTIVE RESERVED
0182   C0B0 B7 09 5B                 STA   FCBACT                      * FILE BUFFERS TO 2 (1,2)
0183   C0B3 30 89 01 19              LEAX  FCBLEN,X                    POINT X TO ONE PAST THE END OF SYSTEM FCB
0184   C0B7 1F 10                    TFR   X,D                         SAVE THE ADDRESS IN ACCD
0185   C0B9 5D                       TSTB                              ON AN EVEN 256 BYTE BOUNDARY?
0186   C0BA 27 01                    BEQ   LC0BD                       YES
0187   C0BC 4C                       INCA                              NO - ADD 256 TO ADDRESS
0188   C0BD 85 01            LC0BD   BITA  #$01                        * CHECK TO SEE IF ACCD IS ON AN EVEN
0189   C0BF 27 01                    BEQ   LC0C2                       * 512 BYTE (ONE GRAPHIC PAGE) BOUNDARY - ADD
0190   C0C1 4C                       INCA                              * 256 (INCA) TO IT IF NOT
0191   C0C2 1F 89            LC0C2   TFR   A,B                         COPY ACCA TO ACCB
0192   C0C4 CB 18                    ADDB  #$18                        SAVE ENOUGH ROOM FOR 4 GRAPHICS PAGES (PCLEAR 4)
```

```
0193   C0C6 D7 19                    STB   TXTTAB                        SAVE NEW START OF BASIC ADDRESS
0194   C0C8 BD 96 EC                 JSR   L96EC                         INITIALIZE EXBAS VARIABLES & DO A NEW
0195   C0CB 96 BA                    LDA   BEGGRP                        GET THE START OF CURRENT GRAPHICS PAGE
0196   C0CD 8B 06                    ADDA  #$06                          ADD 1.5K (6 X 256 = ONE GRAPHICS PAGE)
0197   C0CF 97 B7                    STA   ENDGRP                        SAVE NEW END OF GRAPHICS PAGE
0198   C0D1 AD 9F C0 08              JSR   [DSINIT]                      INITIALIZE SWI2,3 JUMP ADDRESSES
0199   C0D5 8D 19                    BSR   LC0F0                         GO INITIALIZE THE FLOPPY DISK CONTROLLER
0200   C0D7 1C AF                    ANDCC #$AF                          TURN ON IRQ AND FIRQ
0201   C0D9 8E C1 38                 LDX   #LC139-1                      POINT X TO DISK BASIC COPYRIGHT MESSAGE
0202   C0DC BD B9 9C                 JSR   STRINOUT                      PRINT COPYRIGHT MESSAGE TO SCREEN
0203   C0DF 8E C0 E7                 LDX   #DKWMST                       GET DISK BASIC WARM START ADDRESS
0204   C0E2 9F 72                    STX   RSTVEC                        SAVE IT IN RESET VECTOR
0205   C0E4 7E A0 E2                 JMP   LA0E2                         JUMP BACK TO BASIC
0206
0207   C0E7 12              DKWMST   NOP                                 WARM START INDICATOR
0208   C0E8 8D 06                    BSR   LC0F0                         INITIALIZE THE FLOPPY DISK CONTROLLER
0209   C0EA BD D2 D2                 JSR   LD2D2                         CLOSE FILES AND DO MORE INITIALIZATION
0210   C0ED 7E 80 C0                 JMP   XBWMST                        JUMP TO EXBAS' WARM START
0211   C0F0 7F 09 82        LC0F0    CLR   NMIFLG                        RESET NMI FLAG
0212   C0F3 7F 09 85                 CLR   RDYTMR                        RESET DRIVE NOT READY TIMER
0213   C0F6 7F 09 86                 CLR   DRGRAM                        RESET RAM IMAGE OF DSKREG (MOTORS OFF)
0214   C0F9 7F FF 40                 CLR   DSKREG                        RESET DISK CONTROL REGISTER
0215   C0FC 86 D0                    LDA   #$D0                          FORCE INTERRUPT COMMAND OF 1793
0216   C0FE B7 FF 48                 STA   FDCREG                        SEND IT TO 1793
0217   C101 1E 88                    EXG   A,A                           * DELAY
0218   C103 1E 88                    EXG   A,A                           * DELAY SOME MORE
0219   C105 B6 FF 48                 LDA   FDCREG                        GET 1793 STATUS (CLEAR REGISTER)
0220   C108 39                       RTS
0221
0222                         * DISK BASIC COMMAND INTERP TABLES
0223   C109 14              LC109    FCB   20                            20 DISK BASIC 1.1 COMMANDS
0224   C10A C1 92                    FDB   LC192                         DISK BASIC'S COMMAND DICTIONARY
0225   C10C C2 38                    FDB   LC238                         COMMAND JUMP TABLE
0226   C10E 06                       FCB   06                            6 DISK BASIC SECONDARY FUNCTIONS
0227   C10F C2 19                    FDB   LC219                         SECONDARY FUNCTION TABLE
0228   C111 C2 4E                    FDB   LC24E                         SECONDARY FUNCTION JUMP TABLE
0229
0230                         * RAM HOOKS FOR DISK BASIC
0231   C113 C4 4B C8 88 C8 93 LC113    FDB   DVEC0,DVEC1,DVEC2
0232   C119 CC 1C C5 BC C8 48          FDB   DVEC3,DVEC4,DVEC5
0233   C11F C8 4B CA E9 CA F9          FDB   DVEC6,DVEC7,DVEC8
0234   C125 8E 90 CD 35 C8 A9          FDB   XVEC9,DVEC10,DVEC11
0235   C12B C6 E4 CA E4 C9 0C          FDB   DVEC12,DVEC13,DVEC14
0236   C131 CE D2 C6 E4                FDB   DVEC15,DVEC12
0237   C135 C2 65 CA 3E                FDB   DVEC17,DVEC18
0238
0239                         * DISK BASIC COPYRIGHT MESSAGE
0240   C139 44 49 53 4B 20 45 LC139    FCC   'DISK EXTENDED COLOR BASIC 1.1'
0241   C13F 58 54 45 4E 44 45
0242   C145 44 20 43 4F 4C 4F
0243   C14B 52 20 42 41 53 49
0244   C151 43 20 31 2E 31
0245   C156 0D                         FCB   CR
0246   C157 43 4F 50 59 52 49          FDB   'COPYRIGHT (C) 198'
0247   C15D 47 48 54 20 28 43
0248   C163 29 20 31 39 38
0249   C168 32                         FCB   CYEAR
0250   C169 20 42 59 20 54 41          FCC   ' BY TANDY'
0251   C16F 4E 44 59
0252   C172 0D                         FCB   CR
0253   C173 55 4E 44 45 52 20          FCC   'UNDER LICENSE FROM MICROSOFT'
0254   C179 4C 49 43 45 4E 53
0255   C17F 45 20 46 52 4F 4D
0256   C185 20 4D 49 43 52 4F
0257   C18B 53 4F 46 54
0258   C18F 0D 0D 00                   FDB   CR,CR,0
0259
0260
0261                           * DISK BASIC COMMAND DICTIONARY TABLE
0262                           *                                           TOKEN #
0263   C192 44 49 D2           LC192    FCC   'DI',$80+'R'                  CE
0264   C195 44 52 49 56 C5             FCC   'DRIV',$80+'E'                CF
0265   C19A 46 49 45 4C C4             FCC   'FIEL',$80+'D'                D0
0266   C19F 46 49 4C 45 D3             FCC   'FILE',$80+'S'                D1
0267   C1A4 4B 49 4C CC                FCC   'KIL',$80+'L'                 D2
0268   C1A8 4C 4F 41 C4                FCC   'LOA',$80+'D'                 D3
0269   C1AC 4C 53 45 D4                FCC   'LSE',$80+'T'                 D4
0270   C1B0 4D 45 52 47 C5             FCC   'MERG',$80+'E'                D5
0271   C1B5 52 45 4E 41 4D C5          FCC   'RENAM',$80+'E'               D6
0272   C1BB 52 53 45 D4                FCC   'RSE',$80+'T'                 D7
0273   C1BF 53 41 56 C5                FCC   'SAV',$80+'E'                 D8
0274   C1C3 57 52 49 54 C5             FCC   'WRIT',$80+'E'                D9
0275   C1C8 56 45 52 49 46 D9          FCC   'VERIF',$80+'Y'               DA
0276   C1CE 55 4E 4C 4F 41 C4          FCC   'UNLOA',$80+'D'               DB
0277   C1D4 44 53 4B 49 4E C9          FCC   'DSKIN',$80+'I'               DC
0278   C1DA 42 41 43 4B 55 D0          FCC   'BACKU',$80+'P'               DD
0279   C1E0 43 4F 50 D9                FCC   'COP',$80+'Y'                 DE
0280   C1E4 44 53 4B 49 A4             FCC   'DSKI',$80+'$'                DF
0281   C1E9 44 53 4B 4F A4             FCC   'DSKO',$80+'$'                E0
0282   C1EE 44 4F D3                   FCC   'DO',$80+'S'                  E1
0283
0284
0285                           * DISK BASIC COMMAND JUMP TABLE
0286                           *                                           COMMAND / TOKEN #
0287   C1F1 CC A9              LC1F1    FDB   DIR                           DIR / CE
0288   C1F3 CE C5                       FDB   DRIVE                         DRIVE / CF
```

```
0289   C1F5 D0 BC                FDB   FIELD                       FIELD / D0
0290   C1F7 D1 5C                FDB   FILES                       FILES / D1
0291   C1F9 C6 EF                FDB   KILL                        KILL / D2
0292   C1FB CA 48                FDB   LOAD                        LOAD / D3
0293   C1FD D1 02                FDB   LSET                        LSET / D4
0294   C1FF CA 39                FDB   MERGE                       MERGE / D5
0295   C201 D0 1B                FDB   RENAME                      RENAME / D6
0296   C203 D1 01                FDB   RSET                        RSET / D7
0297   C205 C9 E0                FDB   SAVE                        SAVE / D8
0298   C207 D0 66                FDB   WRITE                       WRITE / D9
0299   C209 D7 4E                FDB   VERIFY                      VERIFY / DA
0300   C20B D2 33                FDB   UNLOAD                      UNLOAD / DB
0301   C20D D5 99                FDB   DSKINI                      DSKINI /DC
0302   C20F D2 62                FDB   BACKUP                      BACKUP / DD
0303   C211 D3 B9                FDB   COPY                        COPY / DE
0304   C213 D4 ED                FDB   DSKI                        DSKI$ / DF
0305   C215 D5 62                FDB   DSKO                        DSKO$ / E0
0306   C217 D6 EC                FDB   DOS                         DOS / E1
0307
0308
0309                         * SECONDARY FUNCTION DICTIONARY TABLE
0310                         *                                       TOKEN #
0311   C219 43 56 CE         LC219 FCC   'CV',$80+'N'               A2
0312   C21C 46 52 45 C5            FCC   'FRE',$80+'E'               A3
0313   C220 4C 4F C3               FCC   'LO',$80+'C'                A4
0314   C223 4C 4F C6               FCC   'LO',$80+'F'                A5
0315   C226 4D 4B 4E A4            FCC   'MKN',$80+'$'               A6
0316   C22A 41 D3                  FCC   'A',$80+'S'                 A7
0317
0318
0319                         * DISK BASIC SECONDARY FUNCTION JUMP TABLE
0320                         *                                       FUNCTION / TOKEN #
0321   C22C CD F4            LC22C FDB   CVN                         CVN / A2
0322   C22E CE 9C                  FDB   FREE                        FREE / A3
0323   C230 CE 10                  FDB   LOC                         LOC / A4
0324   C232 CE 37                  FDB   LOF                         LOF / A5
0325   C234 CE 02                  FDB   MKN$                        MKN$ / A6
0326   C236 B2 77                  FDB   AS                          AS / A7
0327
0328                         *DISK BASIC COMMAND INTERPRETATION HANDLER
0329   C238 81 E1            LC238 CMPA  #DHITOK                    *COMPARE TO HIGHEST DISK BASIC TOKEN
0330   C23A 22 08                  BHI   LC244                       *AND BRANCH IF HIGHER
0331   C23C 8E C1 F1               LDX   #LC1F1                      POINT X TO DISK BASIC COMMAND JUMP TABLE
0332   C23F 80 CE                  SUBA  #$CE                        SUBTRACT OUT LOWEST DISK BASIC COMMAND TOKEN
0333   C241 7E AD D4               JMP   LADD4                       JUMP TO BASIC'S COMMAND HANDLER
0334   C244 81 E1            LC244 CMPA  #DHITOK                    COMPARE TO HIGHEST DISK BASIC TOKEN
0335   C246 10 23 F0 2D            LBLS  LB277                       'SYNTAX' ERROR IF < DISK BASIC COMMAND TOKEN
0336   C24A 6E 9F 01 41            JMP   [COMVEC+33]                 PROCESS A USER COMMAND TOKEN
0337                         *DISK BASIC SECONDARY COMMAND INTERPRETATION HANDLER
0338   C24E C1 4E            LC24E CMPB  #($A7-$80)*2               *COMPARE MODIFIED SECONDARY TOKEN TO
0339   C250 23 04                  BLS   LC256                       *HIGHEST DISK BASIC TOKEN & BRANCH IF HIGHER
0340   C252 6E 9F 01 46            JMP   [,COMVEC+38]                JUMP TO USER SECONDARY COMMAND HANDLER
0341   C256 C0 44            LC256 SUBB  #($A2-$80)*2               *SUBTRACT OUT THE SMALLEST SECONDARY
0342   C258 34 04                  PSHS  B                           *DISK TOKEN & SAVE MODIFIED TOKEN ON THE STACK
0343   C25A BD B2 62               JSR   LB262                       SYNTAX CHECK FOR '(' AND EVALUATE EXPRESSION
0344   C25D 35 04                  PULS  B                           RESTORE MODIFIED TOKEN
0345   C25F 8E C2 2C               LDX   #LC22C                      POINT X TO SECONDARY COMMAND JUMP TABLE
0346   C262 7E B2 CE               JMP   LB2CE                       JUMP TO BASIC'S SECONDARY COMMAND HANDLER
0347
0348                         * ERROR DRIVER RAM VECTOR
0349   C265 35 20            DVEC17 PULS Y                          PUT THE RETURN ADDRESS INTO Y
0350   C267 BD AD 33               JSR   LAD33                       RESET THE CONT FLAG, ETC
0351   C26A BD D2 D2               JSR   LD2D2                       INITIALIZE SOME DISK VARIABLES AND CLOSE FILES
0352   C26D 34 24                  PSHS  Y,B                         PUT RETURN ADDRESS AND ERROR NUMBER ON THE STACK
0353   C26F BD CA E9               JSR   DVEC7                       CLOSE ALL FILES
0354   C272 35 04                  PULS  B                           GET THE ERROR NUMBER BACK
0355   C274 C1 36                  CMPB  #2*27                       COMPARE TO THE LOWEST DISK ERROR NUMBER
0356   C276 10 25 C6 76            LBCS  XVEC17                      BRANCH TO EXBAS ERROR HANDLER IF NOT DISK ERROR NUMBER
0357   C27A 32 62                  LEAS  $02,S                       PURGE RETURN ADDRESS OFF THE STACK
0358   C27C BD A7 E9               JSR   LA7E9                       TURN OFF THE CASSETTE MOTOR
0359   C27F BD A9 74               JSR   LA974                       DISABLE THE ANALOG MULTIPLEXER
0360   C282 0F 6F                  CLR   DEVNUM                      SET DEVICE NUMBER TO THE SCREEN
0361   C284 BD B9 5C               JSR   LB95C                       SEND A CR TO THE SCREEN
0362   C287 BD B9 AF               JSR   LB9AF                       SEND A '?' TO THE SCREEN
0363   C28A 8E C2 5A               LDX   #LC290-2*27                 POINT X TO DISK BASIC'S ERROR TABLE
0364   C28D 7E AC 60               JMP   LAC60                       JUMP TO BASIC'S ERROR HANDLER
0365
0366                         * DISK BASIC ERROR MESSAGES
0367   C290 42 52            LC290 FCC   'BR'                       27 BAD RECORD
0368   C292 44 46                  FCC   'DF'                        28 DISK FULL
0369   C294 4F 42                  FCC   'OB'                        29 OUT OF BUFFER SPACE
0370   C296 57 50                  FCC   'WP'                        30 WRITE PROTECTED
0371   C298 46 4E                  FCC   'FN'                        31 BAD FILE NAME
0372   C29A 46 53                  FCC   'FS'                        32 BAD FILE STRUCTURE
0373   C29C 41 45                  FCC   'AE'                        33 FILE ALREADY EXISTS
0374   C29E 46 4F                  FCC   'FO'                        34 FIELD OVERFLOW
0375   C2A0 53 45                  FCC   'SE'                        35 SET TO NON-FIELDED STRING
0376   C2A2 56 46                  FCC   'VF'                        36 VERIFICATION ERROR
0377   C2A4 45 52                  FCC   'ER'                        37 WRITE OR INPUT PAST END OF RECORD
0378
0379                         * DISK FILE EXTENSIONS
0380   C2A6 42 41 53         BASEXT FCC  'BAS'                      BASIC FILE EXTENSION
0381   C2A9 20 20 20         DEFEXT FCC  '   '                      BLANK (DEFAULT) FILE EXTENSION
0382   C2AC 44 41 54         DATEXT FCC  'DAT'                      DATA FILE EXTENSION
0383   C2AF 42 49 4E         BINEXT FCC  'BIN'                      BINARY FILE EXTENSION
0384
```

```
0385                        * CLS RAM VECTOR
0386   C2B2 34 11           DVEC22  PSHS  X,CC                        SAVE X REG AND STATUS
0387   C2B4 AE 63                   LDX   $03,S                       LOAD X WITH CALLING ADDRESS
0388   C2B6 8C 97 5F                CMPX  #L975F                      COMING FROM EXBAS' GET/PUT?
0389   C2B9 26 04                   BNE   LC2BF                       NO
0390   C2BB 81 23                   CMPA  #'#'                        NUMBER SIGN (GET#, PUT#)?
0391   C2BD 27 02                   BEQ   LC2C1                       BRANCH IF GET OR PUT TO RANDOM FILE
0392   C2BF 35 91           LC2BF   PULS  CC,X,PC                     RESTORE X REG, STATUS AND RETURN
0393
0394                        * GET/PUT TO A DIRECT/RANDOM FILE
0395   C2C1 32 65           LC2C1   LEAS  $05,S                       PURGE RETURN ADDRESS AND REGISTERS OFF OF THE STACK
0396   C2C3 BD C8 2E                JSR   LC82E                       EVALUATE DEVICE NUMBER & SET FCB POINTER
0397   C2C6 9F F1                   STX   FCBTMP                      SAVE FCB POINTER
0398   C2C8 6F 88 15                CLR   FCBGET,X                    * RESET THE GET
0399   C2CB 6F 88 16                CLR   FCBGET+1,X                  * DATA POINTER
0400   C2CE 6F 88 17                CLR   FCBPUT,X                    = RESET THE PUT
0401   C2D1 6F 88 18                CLR   FCBPUT+1,X                  = DATA POINTER
0402   C2D4 6F 06                   CLR   FCBPOS,X                    RESET PRINT POSITION COUNTER
0403   C2D6 A6 01                   LDA   FCBDRV,X                    *GET THE FCB DRIVE NUMBER AND
0404   C2D8 97 EB                   STA   DCDRV                       *SAVE IT IN DSKCON VARIABLE
0405   C2DA 9D A5                   JSR   GETCCH                      GET CURRENT INPUT CHARACTER FROM BASIC
0406   C2DC 27 0C                   BEQ   LC2EA                       BRANCH IF END OF LINE
0407   C2DE BD B2 6D                JSR   SYNCOMMA                    SYNTAX CHECK FOR COMMA
0408   C2E1 BD B7 3D                JSR   LB73D                       EVALUATE EXPRESSION - RETURN IN (X)
0409   C2E4 1F 10                   TFR   X,D                         SAVE RECORD NUMBER IN ACCD
0410   C2E6 9E F1           LC2E6   LDX   FCBTMP                      POINT X TO FCB
0411   C2E8 ED 07                   STD   FCBREC,X                    SAVE RECORD NUMBER IN FCB
0412   C2EA EC 07           LC2EA   LDD   FCBREC,X                    GET RECORD NUMBER
0413   C2EC 27 1D                   BEQ   LC30B                       'BAD RECORD' ERROR IF RECORD NUMBER = 0
0414   C2EE BD C6 85                JSR   LC685                       INCREMENT RECORD NUMBER
0415   C2F1 EC 09                   LDD   FCBRLN,X                    * GET RANDOM FILE RECORD LENGTH AND RANDOM FILE
0416   C2F3 AE 0B                   LDX   FCBBUF,X                    * BUFFER POINTER AND SAVE THEM ON THE STACK -
0417   C2F5 34 16                   PSHS  X,B,A                       * THESE ARE THE INITIAL VALUES OF A TEMPORARY
0418                        *                                        * RECORD LENGTH COUNTER AND RANDOM BUFFER
0419                        *                                        * POINTER WHICH ARE MAINTAINED ON THE STACK
0420   C2F7 30 5E                   LEAX  $-2,U                       POINT X TO (RECORD NUMBER -1)
0421   C2F9 BD 9F B5                JSR   L9FB5                       MULT (UNSIGNED) RECORD LENGTH X (RECORD NUMBER -1)
0422   C2FC 34 60                   PSHS  U,Y                         SAVE PRODUCT ON THE STACK
0423   C2FE A6 E0                   LDA   ,S+                         CHECK MS BYTE OF PRODUCT
0424   C300 26 09                   BNE   LC30B                       'BR' ERROR IF NOT ZERO (RECORD NUMBER TOO BIG)
0425   C302 35 10                   PULS  X                           * PULL THE BOTTOM 3 PRODUCT BYTES OFF THE STACK;
0426   C304 35 04                   PULS  B                           * TOP TWO IN X, BOTTOM IN ACCB; ACCB POINTS TO
0427                        *                                        * THE FIRST BYTE OF THE SECTOR USED BY THIS RECORD,
0428                        *                                        * (X) CONTAINS THE SECTOR OFFSET (IN WHICH SECTOR
0429                        *                                        * FROM THE START THE BYTE IS LOCATED)
0430   C306 8C 02 64        LC306   CMPX  #(TRKMAX-1)                 612 SECTORS MAX IN A RANDOM FILE
0431   C309 25 05                   BLO   LC310                       BRANCH IF RECORD LENGTH O.K.
0432   C30B C6 36           LC30B   LDB   #2*27                       'BAD RECORD' ERROR
0433   C30D 7E AC 46                JMP   LAC46                       JUMP TO ERROR HANDLER
0434   C310 DE F1           LC310   LDU   FCBTMP                      POINT U TO FCB
0435   C312 AC 4D                   CMPX  FCBSOF,U                    * COMPARE SAVED SECTOR OFFSET TO THE CURRENT SECTOR OFFSET
0436   C314 10 27 00 B7             LBEQ  LC3CF                       * BEING PROCESSED - DO NOT PROCESS A NEW SECTOR IF THEY ARE EQUAL
0437   C318 34 14                   PSHS  X,B                         SAVE BYTE AND SECTOR OFFSET TO RECORD START ON STACK
0438   C31A A6 4F                   LDA   FCBFLG,U                    * CHECK FCB GET/PUT FLAG AND
0439   C31C 27 06                   BEQ   LC324                       * BRANCH IF IT WAS A GET
0440   C31E 6F 4F                   CLR   FCBFLG,U                    FORCE GET/PUT TO 'PUT'
0441   C320 C6 03                   LDB   #$03                        DSKCON WRITE OP CODE
0442   C322 8D 33                   BSR   LC357                       GO WRITE A SECTOR - SAVE 'PUT' DATA ON DISK
0443                        * CONVERT THE SECTOR OFFSET TO A GRANULE AND SECTOR NUMBER
0444   C324 EC 61           LC324   LDD   $01,S                       * GET THE NUMBER OF SECTORS TO THE START OF
0445   C326 BD C7 84                JSR   LC784                       * THIS RECORD NUMBER AND CONVERT THEM TO A GRANULE OFFSET
0446   C329 34 04                   PSHS  B                           SAVE GRANULE OFFSET ON THE STACK
0447   C32B BD C7 79                JSR   LC779                       MULTIPLY GRANULE NUMBER X 9 - CONVERT TO NUMBER OF SECTORS
0448   C32E 50                      NEGB                              * NEGATE LS BYTE OF GRANULE OFFSET AND ADD THE
0449   C32F EB 63                   ADDB  $03,S                       * LS BYTE OF SECTOR OFFSET - ACCB = SECTOR
0450                        *                                        * NUMBER (0-8) CORRESPONDING TO THE SECTOR NUMBER WITHIN A
0451                        *                                        * GRANULE OF THE LAST SECTOR OF THE SECTOR OFFSET
0452   C331 5C                      INCB                              = ADD ONE - SECTORS SAVED IN THE FCB; START
0453   C332 E7 44                   STB   FCBSEC,U                    = AT 1 NOT 0 - SAVE IT IN THE FCB
0454   C334 E6 42                   LDB   FCBFGR,U                    GET FIRST GRANULE IN FILE
0455   C336 BD C7 55                JSR   LC755                       POINT X TO FAT
0456   C339 33 06                   LEAU  FATCON,X                    POINT U TO FAT DATA
0457   C33B A6 E4                   LDA   ,S                          GET NUMBER OF GRANULES OFFSET TO RECORD
0458   C33D 4C                      INCA                              ADD ONE (COMPENSATE FOR DECA BELOW)
0459   C33E 30 C4           LC33E   LEAX  ,U                          POINT X TO FAT DATA
0460   C340 3A                      ABX                               POINT X TO CORRECT GRANULE
0461   C341 4A                      DECA                              DECREMENT GRANULE COUNTER
0462   C342 27 37                   BEQ   LC37B                       BRANCH IF CORRECT GRANULE FOUND
0463   C344 E7 E4                   STB   ,S                          SAVE GRANULE ADDRESS ON STACK
0464   C346 E6 84                   LDB   ,X                          GET NEXT GRANULE IN FILE
0465   C348 C1 C0                   CMPB  #$C0                        LAST GRANULE IN FILE?
0466   C34A 25 F2                   BLO   LC33E                       NO - KEEP LOOKING
0467
0468                        * THE GRANULE BEING SEARCHED FOR IS NOT PRESENTLY DEFINED IN THIS RANDOM FILE
0469   C34C E6 E4                   LDB   ,S                          GET OFFSET TO LAST GRANULE
0470   C34E 0D D8                   TST   VD8                         * CHECK GET/PUT FLAG
0471   C350 26 14                   BNE   LC366                       * AND BRANCH IF PUT
0472   C352 C6 2E           LC352   LDB   #2*23                       'INPUT PAST END OF FILE' ERROR
0473   C354 7E AC 46                JMP   LAC46                       JUMP TO ERROR HANDLER
0474   C357 30 C8 19        LC357   LEAX  FCBCON,U                    POINT X TO FCB DATA BUFFER
0475
0476                        * READ/WRITE A SECTOR. ENTER WITH OP CODE IN ACCB, BUFFER PTR IN X
0477   C35A D7 EA           LC35A   STB   DCOPC                       SAVE DSKCON OPERATION CODE VARIABLE
0478   C35C 9F EE                   STX   DCBPT                       SAVE DSKCON LOAD BUFFER VARIABLE
0479   C35E 30 C4                   LEAX  ,U                          POINT X TO FCB
0480   C360 BD C7 63                JSR   LC763                       CONVERT FCB TRACK AND SECTOR TO DSKCON VARIABLES
```

```
0481   C363 7E D6 F2                JMP   LD6F2                          READ/WRITE A TRACK OR SECTOR
0482
0483                        * 'PUT' DATA INTO A GRANULE NOT PRESENTLY INCLUDED IN THIS FILE
0484   C366 34 12           LC366   PSHS  X,A                            SAVE GRANULE COUNTER AND POINTER TO LAST USED GRANULE
0485   C368 BD C7 BF                JSR   LC7BF                          FIND FIRST FREE GRANULE IN FAT
0486   C36B 1F 89                   TFR   A,B                            SAVE FREE GRANULE NUMBER IN ACCB
0487   C36D 35 42                   PULS  A,U                            PULL LAST GRANULE POINTER AND COUNTER OFF OF STACK
0488   C36F E7 C4                   STB   ,U                             SAVE NEWLY FOUND GRANULE NUMBER IN ADDRESS OF LAST GRANULE
0489   C371 4A                      DECA                                 DECREMENT GRANULE COUNTER
0490   C372 26 F2                   BNE   LC366                          GET ANOTHER GRANULE IF NOT DONE
0491   C374 34 14                   PSHS  X,B                            SAVE POINTER TO LAST GRANULE AND OFFSET
0492   C376 BD C7 1E                JSR   LC71E                          WRITE FAT TO DISK
0493   C379 35 14                   PULS  B,X                            RESTORE POINTER AND OFFSET
0494
0495                        * WHEN CORRECT GRANULE IS FOUND, FIND THE RIGHT SECTOR
0496   C37B 32 61           LC37B   LEAS  $01,S                          REMOVE GRAN NUMBER FROM STACK
0497   C37D DE F1                   LDU   FCBTMP                         POINT U TO FCB
0498   C37F E7 43                   STB   FCBCGR,U                       SAVE CURRENT GRANULE IN FCB
0499   C381 86 FF                   LDA   #$FF                           *SET FCBSOF,U TO ILLEGAL SECTOR OFFSET WHICH WILL
0500   C383 A7 4D                   STA   FCBSOF,U                       *FORCE NEW SECTOR DATA TO BE READ IN
0501   C385 A6 84                   LDA   ,X                             GET CURRENT GRANULE
0502   C387 81 C0                   CMPA  #$C0                           IS IT THE LAST GRANULE?
0503   C389 25 27                   BLO   LC3B2                          NO
0504   C38B 84 3F                   ANDA  #$3F                           MASK OFF LAST GRANULE FLAG BITS
0505   C38D A1 44                   CMPA  FCBSEC,U                       * COMPARE CALCULATED SECTOR TO CURRENT SECTOR IN FCB
0506   C38F 24 21                   BHS   LC3B2                          * AND BRANCH IF CALCULATED SECTOR IS > LAST SECTOR IN FILE
0507   C391 96 D8                   LDA   VD8                            = CHECK GET/PUT FLAG: IF 'GET' THEN 'INPUT
0508   C393 27 BD                   BEQ   LC352                          = PAST END OF FILE' ERROR
0509   C395 A6 44                   LDA   FCBSEC,U                       * GET CURRENT SECTOR NUMBER FROM FCB,
0510   C397 8A C0                   ORA   #$C0                           * OR IN THE LAST GRANULE FLAG BITS
0511   C399 A7 84                   STA   ,X                             * AND SAVE IN FAT
0512   C39B BD C5 A9                JSR   LC5A9                          WRITE FAT TO DISK IF NECESSARY
0513   C39E AE 49                   LDX   FCBRLN,U                       * GET RECORD LENGTH AND CHECK TO
0514   C3A0 8C 01 00                CMPX  #SECLEN                        * SEE IF IT IS SECLEN (EXACTLY ONE SECTOR)
0515   C3A3 26 08                   BNE   LC3AD                          BRANCH IF IT IS NOT EXACTLY ONE SECTOR
0516   C3A5 AC C8 13                CMPX  FCBLST,U                       =BRANCH IF THE NUMBER OF BYTES IN THE LAST SECTOR
0517   C3A8 27 08                   BEQ   LC3B2                          =IS SET TO ONE SECTOR (SECLEN)
0518   C3AA 86 81                   LDA   #$81                           *SET THE PRESAVED FLAG (BIT15) AND FORCE
0519   C3AC 21 4F           LC3AC   BRN   $C3FD                          *THE NUMBER OF BYTES IN LAST SECTOR TO 256
0520   C3AD 4F              LC3AD   CLRA                                 SET THE NUMBER OF BYTES IN LAST SECTOR TO ZERO
0521   C3AE 5F                      CLRB                                 CLEAR LS BYTE OF ACCD
0522   C3AF ED C8 13                STD   FCBLST,U                       SAVE THE NUMBER OF BYTES IN LAST SECTOR
0523   C3B2 C6 02           LC3B2   LDB   #$02                           DSKCON READ OP CODE
0524   C3B4 AE 49                   LDX   FCBRLN,U                       * GET RECORD LENGTH AND COMPARE
0525   C3B6 8C 01 00                CMPX  #SECLEN                        * IT TO SECLEN - EXACTLY ONE SECTOR
0526   C3B9 26 0D                   BNE   LC3C8                          BRANCH IF NOT EXACTLY ONE SECTOR LONG
0527   C3BB 32 67                   LEAS  $07,S                          CLEAN UP STACK
0528   C3BD AE 4B                   LDX   FCBBUF,U                       POINT X TO START OF RANDOM FILE BUFFER
0529   C3BF 96 D8                   LDA   VD8                            * CHECK GET/PUT FLAG AND
0530   C3C1 27 02                   BEQ   LC3C5                          * BRANCH IF GET
0531   C3C3 C6 03                   LDB   #$03                           DSKCON WRITE OP CODE
0532 > C3C5 7E C3 5A        LC3C5   JMP   LC35A                          READ/WRITE A SECTOR
0533 > C3C8 BD C3 57        LC3C8   JSR   LC357                          READ A SECTOR INTO FCB DATA BUFFER
0534   C3CB 35 14                   PULS  B,X                            * GET BACK THE BYTE OFFSET TO RECORD: X = NUMBER OF
0535                        *                                            * SECTORS; ACCB = BYTE POINTER IN SECTOR
0536   C3CD AF 4D                   STX   FCBSOF,U                       SAVE SECTOR OFFSET IN FCB
0537   C3CF 34 04           LC3CF   PSHS  B                              SAVE BYTE OFFSET ON STACK
0538   C3D1 BD C7 55                JSR   LC755                          POINT X TO FILE ALLOCATION TABLE
0539   C3D4 30 06                   LEAX  FATCON,X                       MOVE X TO FAT DATA
0540   C3D6 E6 43                   LDB   FCBCGR,U                       GET CURRENT GRANULE NUMBER
0541   C3D8 3A                      ABX                                  POINT X TO PROPER GRANULE IN FAT
0542   C3D9 A6 84                   LDA   ,X                             * GET CURRENT GRANULE AND CHECK TO
0543   C3DB 81 C0                   CMPA  #$C0                           * SEE IF IT IS LAST GRANULE
0544   C3DD 25 2B                   BLO   LC40A                          BRANCH IF THIS GRANULE IS < LAST GRANULE
0545   C3DF 84 3F                   ANDA  #$3F                           MASK OFF LAST GRANULE FLAG BITS
0546   C3E1 A1 44                   CMPA  FCBSEC,U                       * COMPARE LAST SECTOR USED IN GRANULE TO
0547   C3E3 26 25                   BNE   LC40A                          * CALCULATED SECTOR; BRANCH IF NOT EQUAL
0548   C3E5 EC C8 13                LDD   FCBLST,U                       GET NUMBER OF BYTES IN LAST SECTOR
0549   C3E8 84 7F                   ANDA  #$7F                           MASK OFF PRESAVED FLAG (BIT 15)
0550   C3EA 34 06                   PSHS  B,A                            SAVE NUMBER OF BYTES IN LAST SECTOR ON STACK
0551   C3EC 4F                      CLRA                                 * LOAD ACCB WITH THE BYTE OFFSET TO CURRENT
0552   C3ED E6 62                   LDB   $02,S                          * RECORD AND ADD THE REMAINING RECORD LENGTH
0553   C3EF E3 63                   ADDD  $03,S                          * TO IT - ACCD = END OF RECORD OFFSET
0554   C3F1 10 A3 E1                CMPD  ,S++                           =COMPARE THE END OF RECORD OFFSET TO THE NUMBER OF
0555   C3F4 23 14                   BLS   LC40A                          =BYTES USED IN THE LAST SECTOR
0556   C3F6 0D D8                   TST   VD8                            * CHECK GET/PUT FLAG AND BRANCH IF 'GET'
0557   C3F8 10 27 FF 56             LBEQ  LC352                          * TO 'INPUT PAST END OF FILE' ERROR
0558
0559                        * IF LAST USED SECTOR, CALCULATE HOW MANY BYTES ARE USED
0560                        * IF DATA IS BEING 'PUT' PASTH THE CURRENT END OF FILE
0561   C3FC 10 83 01 00             CMPD  #SECLEN                        COMPARE TO ONE SECTOR'S LENGTH
0562   C400 23 03                   BLS   LC405                          BRANCH IF REMAINDER OF RECORD LENGTH WILL FIT IN THIS SECTOR
0563   C402 CC 01 00                LDD   #SECLEN                        FORCE NUMBER OF BYTES = ONE SECTOR LENGTH
0564   C405 8A 80           LC405   ORA   #$80                           * SET PRE-SAVED FLAG BIT - ALL PUT RECORDS ARE
0565                        *                                            * WRITTEN TO DISK BEFORE LEAVING 'PUT'
0566   C407 ED C8 13                STD   FCBLST,U                       SAVE NUMBER OF BYTES USED IN LAST SECTOR
0567   C40A 35 04           LC40A   PULS  B                              PULL BYTE OFFSET OFF OF THE STACK
0568   C40C 30 C8 19                LEAX  FCBCON,U                       POINT X TO FCB DATA BUFFER
0569   C40F 3A                      ABX                                  MOVE X TO START OF RECORD
0570   C410 EE 62                   LDU   $02,S                          POINT U TO CURRENT POSITION IN RANDOM FILE BUFFER
0571   C412 34 04                   PSHS  B                              SAVE BYTE OFFSET ON STACK
0572   C414 86 FF                   LDA   #-1                            * CONVERT ACCD INTO A NEGATIVE 2 BYTE NUMBER
0573                        *                                            * REPRESENTING THE REMAINING UNUSED BYTES IN THE SECTOR
0574   C416 E3 61                   ADDD  $01,S                          * ADD TEMPORARY RECORD LENGTH COUNTER (SUBTRACT
0575                        *                                            * REMAINING BYTES FROM TEMPORARY RECORD LENGTH)
0576   C418 24 07                   BHS   LC421                          BRANCH IF THERE ARE ENOUGH UNUSED BYTES TO FINISH THE RECORD
```

```
0577   C41A ED 61              STD   $01,S                    SAVE NEW TEMPORARY RECORD LENGTH COUNTER
0578   C41C 35 04              PULS  B                        RESTORE BYTE COUNTER
0579   C41E 50                 NEGB                           * NEGATE IT - ACCB = THE NUMBER OF BYTES
0580                          *                               * AVAILABLE TO A RECORD IN THIS SECTOR
0581   C41F 20 08              BRA   LC429                    MOVE THE DATA
0582
0583                          * BRANCH HERE IF REMAINING RECORD LENGTH WILL FIT IN
0584                          * WHAT'S LEFT OF THE CURRENTLY SELECTED SECTOR
0585   C421 E6 62              LC421 LDB   $02,S              GET REMAINING RECORD LENGTH
0586   C423 6F 61              CLR   $01,S                    * CLEAR THE TEMPORARY RECORD LENGTH
0587   C425 6F 62              CLR   $02,S                    * COUNTER ON THE STACK
0588   C427 32 61              LEAS  $01,S                    PURGE BYTE OFFSET FROM STACK
0589   C429 96 D8              LC429 LDA   VD8                * CHECK GET/PUT FLAG AND
0590   C42B 27 02              BEQ   LC42F                    * BRANCH IF GET
0591   C42D 1E 13              EXG   X,U                      SWAP SOURCE AND DESTINATION POINTERS
0592   C42F BD A5 9A           LC42F JSR   LA59A              TRANSFER DATA FROM SOURCE TO DESTINATION BUFFERS
0593   C432 EF 62              STU   $02,S                    SAVE NEW TEMP RECORD POINTER ON THE STACK (GET)
0594
0595                          * MOVE DATA FROM FCB DATA BUFFER TO THE RANDOM FILE BUFFER IF 'GET'
0596                          * OR FROM RANDOM FILE BUFFER TO FCB DATA BUFFER IF 'PUT'
0597   C434 DE F1              LDU   FCBTMP                   POINT U TO FCB
0598   C436 96 D8              LDA   VD8                      * CHECK GET/PUT FLAG AND
0599   C438 27 04              BEQ   LC43E                    * BRANCH IF GET
0600   C43A A7 4F              STA   FCBFLG,U                 SAVE 'PUT' FLAG IN THE FCB
0601   C43C AF 62              STX   $02,S                    SAVE NEW TEMPORARY RECORD POINTER ON STACK (PUT)
0602   C43E AE 4D              LC43E LDX   FCBSOF,U           * GET SECTOR OFFSET COUNTER AND
0603   C440 30 01              LEAX  $01,X                    * ADD ONE TO IT
0604   C442 5F                 CLRB                           SET BYTE OFFSET = 0
0605   C443 EE E4              LDU   ,S                       * CHECK THE LENGTH OF THE TEMPORARY RECORD LENGTH
0606   C445 10 26 FE BD        LBNE  LC306                    * COUNTER AND KEEP MOVING DATA IF <> 0
0607   C449 35 96              PULS  A,B,X,PC                 * PULL TEMPORARY RECORD LENGTH AND
0608                          *                               * BUFFER ADDRESS OFF STACK AND RETURN
0609
0610                          * OPEN RAM HOOK
0611   C44B 32 62              DVEC0 LEAS  $02,S              PULL RETURN ADDRESS OFF OF THE STACK
0612   C44D BD B1 56           JSR   LB156                    EVALUATE AN EXPRESSION
0613   C450 BD B6 A4           JSR   LB6A4                    *GET MODE(I,O,R) - FIRST BYTE OF STRING EXPRESSION
0614   C453 34 04              PSHS  B                        *AND SAVE IT ON STACK
0615   C455 BD A5 A2           JSR   LA5A2                    GET DEVICE NUMBER
0616   C458 5D                 TSTB                           SET FLAGS
0617   C459 10 2F E1 A6        LBLE  LA603                    BRANCH IF NOT A DISK FILE
0618   C45D 35 02              PULS  A                        GET MODE
0619   C45F 34 06              PSHS  B,A                      SAVE MODE AND DEVICE NUMBER (FILE NUMBER)
0620   C461 0F 6F              CLR   DEVNUM                   SET DEVICE NUMBER TO SCREEN
0621   C463 BD B2 6D           JSR   SYNCOMMA                 SYNTAX CHECK FOR COMMA
0622   C466 8E C2 AC           LDX   #DATEXT                  POINT TO 'DAT' FOR EXTENSION
0623   C469 BD C9 38           JSR   LC938                    GET FILENAME FROM BASIC
0624   C46C CC 01 FF           LDD   #$01FF                   DEFAULT DISK FILE TYPE AND ASCII FLAG
0625   C46F FD 09 57           STD   DFLTYP                   SAVE DEFAULT VALUES: DATA, ASCII
0626   C472 8E 01 00           LDX   #SECLEN                  DEFAULT RECORD LENGTH - 1 PAGE
0627   C475 9D A5              JSR   GETCCH                   GET CHAR FROM BASIC
0628   C477 27 08              BEQ   LC481                    BRANCH IF END OF LINE
0629   C479 BD B2 6D           JSR   SYNCOMMA                 SYNTAX CHECK FOR COMMA
0630   C47C BD B3 E6           JSR   LB3E6                    EVALUATE EXPRESSION
0631   C47F 9E 52              LDX   FPA0+2                   GET EVALUATED EXPRESSION
0632   C481 BF 09 7C           LC481 STX   DFFLEN             RECORD LENGTH
0633   C484 10 27 EF C2        LBEQ  LB44A                    IF = 0, THEN 'ILLEGAL FUNCTION CALL'
0634   C488 BD A5 C7           JSR   LA5C7                    ERROR IF ANY FURTHER CHARACTERS ON LINE
0635   C48B 35 06              PULS  A,B                      GET MODE AND FILE NUMBER
0636
0637                          * OPEN DISK FILE FOR READ OR WRITE
0638   C48D 34 02              LC48D PSHS  A                  SAVE MODE ON STACK
0639   C48F BD C7 49           JSR   LC749                    POINT X TO FCB FOR THIS FILE
0640   C492 10 26 E1 86        LBNE  LA61C                    'FILE ALREADY OPEN' ERROR IF FILE OPEN
0641   C496 9F F1              STX   FCBTMP                   SAVE FILE BUFFER POINTER
0642   C498 BD C7 9D           JSR   LC79D                    MAKE SURE FILE ALLOC TABLE IS VALID
0643   C49B BD C6 8C           JSR   LC68C                    SCAN DIRECTORY FOR 'FILENAME.EXT'
0644   C49E 35 04              PULS  B                        GET MODE
0645   C4A0 86 10              LDA   #INPFIL                  INPUT TYPE FILE
0646   C4A2 34 02              PSHS  A                        SAVE FILE TYPE ON STACK
0647   C4A4 C1 49              CMPB  #'I'                     INPUT MODE?
0648   C4A6 26 1F              BNE   LC4C7                    BRANCH IF NOT
0649
0650                          * OPEN A SEQUENTIAL FILE FOR INPUT
0651   C4A8 BD C6 E5           JSR   LC6E5                    CHECK TO SEE IF DIRECTORY MATCH IS FOUND
0652   C4AB BD C8 07           JSR   LC807                    CHECK TO SEE IF FILE ALREADY OPEN
0653   C4AE BE 09 74           LDX   V974                     GET RAM DIRECTORY BUFFER
0654   C4B1 EC 0B              LDD   DIRTYP,X                 GET FILE TYPE AND ASCII FLAG
0655   C4B3 FD 09 57           STD   DFLTYP                   SAVE IN RAM IMAGE
0656   C4B6 8D 75              BSR   LC52D                    INITIALIZE FILE BUFFER CONTROL BLOCK
0657   C4B8 BD C6 27           JSR   LC627                    GO FILL DATA BUFFER
0658   C4BB BD C7 55           LC4BB JSR   LC755              POINT X TO PROPER FILE ALLOCATION TABLE
0659   C4BE 6C 00              INC   FAT0,X                   ADD ONE TO FAT ACTIVE FILE COUNTER
0660   C4C0 9E F1              LDX   FCBTMP                   GET FILE BUFFER POINTER
0661   C4C2 35 02              PULS  A                        GET FILE TYPE
0662   C4C4 A7 00              STA   FCBTYP,X                 SAVE IT IN FCB
0663   C4C6 39                 RTS
0664   C4C7 68 E4              LC4C7 ASL   ,S                 SET FILE TYPE TO OUTPUT
0665   C4C9 C1 4F              CMPB  #'O'                     FILE MODE = OUTPUT?
0666   C4CB 26 1B              BNE   LC4E8                    BRANCH IF NOT
0667
0668                          * OPEN A SEQUENTIAL FILE FOR OUTPUT
0669   C4CD 7D 09 73           TST   V973                     DOES FILE EXIST ON DIRECTORY?
0670   C4D0 27 0F              BEQ   LC4E1                    BRANCH IF NOT
0671   C4D2 BD C6 FC           JSR   LC6CF                    KILL THE OLD FILE
0672   C4D5 B6 09 73           LDA   V973                     * GET DIRECTORY SECTOR NUMBER OF OLD FILE AND
```

```
0673   C4D8 B7 09 77              STA   V977                        * SAVE IT AS FIRST FREE DIRECTORY ENTRY
0674   C4DB BE 09 74              LDX   V974                        =GET RAM DIRECTORY IMAGE OF OLD FILE AND
0675   C4DE BF 09 78              STX   V978                        =SAVE IT AS FIRST FREE DIRECTORY ENTRY
0676
0677   C4E1 BD C5 67       LC4E1  JSR   LC567                       SET UP NEW DIRECTORY ENTRY ON DISK
0678   C4E4 8D 52                 BSR   LC538                       INITIALIZE FILE BUFFER
0679   C4E6 20 D3                 BRA   LC4BB                       FLAG AND MAP FCB AS BEING USED
0680   C4E8 C1 52          LC4E8  CMPB  #'R'                        FILE MODE = R (RANDOM)?
0681   C4EA 27 06                 BEQ   LC4F2                       BRANCH IF SO
0682   C4EC C1 44                 CMPB  #'D'                        FILE MODE = D (DIRECT)?
0683   C4EE 10 26 E1 24           LBNE  LA616                       'BAD FILE MODE' ERROR IF NOT
0684
0685                       * OPEN A RANDOM/DIRECT FILE
0686   C4F2 68 E4          LC4F2  ASL   ,S                          SET FILE TYPE TO DIRECT
0687   C4F4 FC 09 48              LDD   RNBFAD                      * GET ADDRESS OF RANDOM FILE BUFFER AREA
0688   C4F7 34 06                 PSHS  B,A                         * AND SAVE IT ON THE STACK
0689   C4F9 F3 09 7C              ADDD  DFFLEN                      ADD THE RECORD LENGTH
0690   C4FC 25 06                 BLO   LC504                       'OB' ERROR IF SUM > $FFFF
0691   C4FE 10 B3 09 4A           CMPD  FCBADR                      IS IT > THAN FCB DATA AREA?
0692   C502 23 05                 BLS   LC509                       BRANCH IF NOT
0693   C504 C6 3A          LC504  LDB   #2*29                       'OUT OF BUFFER SPACE' ERROR
0694   C506 7E AC 46              JMP   LAC46                       JUMP TO ERROR HANDLER
0695   C509 34 06          LC509  PSHS  B,A                         SAVE END OF RANDOM BUFFER ON STACK
0696   C50B 7D 09 73              TST   V973                        DID THIS FILE EXIST
0697   C50E 26 04                 BNE   LC514                       BRANCH IF SO
0698   C510 8D 55                 BSR   LC567                       SET UP NEW FILE IN DIRECTORY
0699   C512 20 05                 BRA   LC519                       INITIALIZE FCB
0700   C514 86 FF          LC514  LDA   #$FF                        * SET FILE TYPE MATCH = $FF (ILLEGAL VALUE) -
0701   C516 BD C8 07              JSR   LC807                       * THIS WILL FORCE ANY OPEN MATCHED FILE TO CAUSE
0702                              *                                 * A 'FILE ALREADY OPEN' ERROR
0703   C519 8D 12          LC519  BSR   LC52D                       INITIALIZE FCB
0704   C51B 63 0D                 COM   FCBSOF,X                    * SET FCBSOF,X TO $FF (ILLEGAL SECTOR OFFSET) WHICH WILL
0705                              *                                 * FORCE NEW SECTOR DATA TO BE READ IN DURING GET/PUT
0706   C51D 6C 08                 INC   FCBREC+1,X                  INITIALIZE RECORD NUMBER = 1
0707   C51F 35 46                 PULS  A,B,U                       U = START OF RANDOM FILE BUFFER AREA, ACCD = END
0708   C521 FD 09 48              STD   RNBFAD                      SAVE NEW START OF RANDOM FILE BUFFER AREA
0709   C524 EF 0B                 STU   FCBBUF,X                    SAVE BUFFER START IN FCB
0710   C526 FE 09 7C              LDU   DFFLEN                      * GET RANDOM FILE RECORD LENGTH
0711   C529 EF 09                 STU   FCBRLN,X                    * AND SAVE IT IN FCB
0712   C52B 20 8E                 BRA   LC4BB                       SET FAT FLAG, SAVE FILE TYPE IN FCB
0713
0714                       * INITIALIZE FCB DATA FOR INPUT
0715   C52D 8D 09          LC52D  BSR   LC538                       INITIALIZE FCB
0716   C52F FE 09 74              LDU   V974                        GET RAM DIRECTORY IMAGE
0717   C532 EE 4E                 LDU   DIRLIST,U                   *GET NUMBER OF BYTES IN LAST SECTOR OF FILE
0718   C534 EF 88 13              STU   FCBLST,X                    *SAVE IT IN FCB
0719   C537 39                    RTS
0720                       * INITIALIZE FILE CONTROL BLOCK
0721   C538 9E F1          LC538  LDX   FCBTMP                      GET CURRENT FILE BUFFER
0722   C53A C6 19                 LDB   #FCBCON                     CLEAR FCB CONTROL BYTES
0723   C53C 6F 80          LC53C  CLR   ,X+                         CLEAR A BYTE
0724   C53E 5A                    DECB                              DECREMENT COUNTER
0725   C53F 26 FB                 BNE   LC53C                       BRANCH IF NOT DONE
0726   C541 9E F1                 LDX   FCBTMP                      GET CURRENT FILE BUFFER ADDRESS BACK
0727   C543 96 EB                 LDA   DCDRV                       *GET CURRENT DRIVE NUMBER AND
0728   C545 A7 01                 STA   FCBDRV,X                    *SAVE IT IN FCB
0729   C547 B6 09 76              LDA   V976                        =GET FIRST GRANULE -
0730   C54A A7 02                 STA   FCBFGR,X                    =SAVE IT AS THE STARTING GRANULE NUMBER AND
0731   C54C A7 03                 STA   FCBCGR,X                    =SAVE IT AS CURRENT GRANULE NUMBER
0732   C54E F6 09 73              LDB   V973                        GET DIRECTORY SECTOR NUMBER
0733   C551 C0 03                 SUBB  #$03                        SUBTRACT 3 - DIRECTORY SECTORS START AT 3
0734   C553 58                    ASLB                              * MULTIPLY SECTORS
0735   C554 58                    ASLB                              * BY 8 (8 DIRECTORY
0736   C555 58                    ASLB                              * ENTRIES PER SECTOR)
0737   C556 34 04                 PSHS  B                           SAVE SECTOR OFFSET
0738   C558 FC 09 74              LDD   V974                        GET RAM DIRECTORY IMAGE
0739   C55B 83 06 00              SUBD  #DBUF0                      SUBTRACT RAM OFFSET
0740   C55E 86 08                 LDA   #$08                        8 DIRECTORY ENTRIES/SECTOR
0741   C560 3D                    MUL                               NOW ACCA CONTAINS 0-7
0742   C561 AB E0                 ADDA  ,S+                         ACCA CONTAINS DIRECTORY ENTRY (0-71)
0743   C563 A7 88 12              STA   FCBDIR,X                    SAVE DIRECTORY ENTRY NUMBER
0744   C566 39                    RTS
0745
0746                       * SET UP DIRECTORY AND UPDATE FILE ALLOCATION TABLE ENTRY IN FIRST UNUSED SECTOR
0747   C567 C6 38          LC567  LDB   #28*2                       'DISK FULL' ERROR
0748   C569 B6 09 77              LDA   V977                        GET SECTOR NUMBER OF FIRST EMPTY DIRECTORY ENTRY
0749   C56C 10 27 E6 D6           LBEQ  LAC46                       'DISK FULL' ERROR IF NO EMPTY DIRECTORY ENTRIES
0750   C570 B7 09 73              STA   V973                        SAVE SECTOR NUMBER OF FIRST EMPTY DIRECTORY ENTRY
0751   C573 97 ED                 STA   DSEC                        SAVE SECTOR NUMBER IN DSKCON REGISTER
0752   C575 C6 02                 LDB   #$02                        READ OP CODE
0753   C577 D7 EA                 STB   DCOPC                       SAVE IN DSKCON REGISTER
0754   C579 BD D6 F2              JSR   LD6F2                       READ SECTOR
0755   C57C BE 09 78              LDX   V978                        * GET ADDRESS OF RAM IMAGE OF UNUSED DIRECTORY
0756   C57F BF 09 74              STX   V974                        * ENTRY AND SAVE AS CURRENT USED RAM IMAGE
0757   C582 33 84                 LEAU  ,X                          (TFR X,U) POINT U TO DIRECTORY RAM IMAGE
0758   C584 C6 20                 LDB   #DIRLEN                     SET COUNTER TO CLEAR 32 BYTES (DIRECTORY ENTRY)
0759   C586 6F 80          LC586  CLR   ,X+                         CLEAR BYTE
0760   C588 5A                    DECB                              DECREMENT COUNTER
0761   C589 26 FB                 BNE   LC586                       CONTINUE IF NOT DONE
0762   C58B 8E 09 4C              LDX   #DNAMBF                     POINT TO FILENAME AND EXTENSION RAM IMAGE
0763   C58E C6 0B                 LDB   #11                         11 BYTES IN FILENAME AND EXTENSION
0764   C590 BD A5 9A              JSR   LA59A                       MOVE B BYTES FROM X TO U
0765   C593 FC 09 57              LDD   DFLTYP                      GET FILE TYPE AND ASCII FLAG
0766   C596 ED 40                 STD   $00,U                       SAVE IN RAM IMAGE
0767   C598 C6 21                 LDB   #33                         FIRST GRANULE TO CHECK
0768   C59A BD C7 BF              JSR   LC7BF                       FIND THE FIRST FREE GRANULE
```

```
0769   C59D B7 09 76              STA   V976                        SAVE IN RAM
0770   C5A0 A7 42                 STA   $02,U                       SAVE IN RAM IMAGE OF DIRECTORY TRACK
0771   C5A2 C6 03                 LDB   #$03                       * GET WRITE OPERATION CODE AND SAVE
0772   C5A4 D7 EA                 STB   DCOPC                       * IT IN DSKCON REGISTER
0773   C5A6 BD D6 F2              JSR   LD6F2                       GO WRITE A SECTOR IN DIRECTORY
0774   C5A9 34 56          LC5A9  PSHS  U,X,B,A                     SAVE REGISTERS
0775   C5AB BD C7 55              JSR   LC755                       POINT X TO FILE ALLOCATION TABLE
0776   C5AE 6C 01                 INC   FAT1,X                      INDICATE NEW DATA IN FILE ALLOC TABLE
0777   C5B0 A6 01                 LDA   FAT1,X                      GET NEW DATA FLAG
0778   C5B2 B1 09 7A              CMPA  WFATVL                      * HAVE ENOUGH GRANULES BEEN REMOVED FROM THE FAT TO
0779                       *                                        * CAUSE THE FAT TO BE WRITTEN TO THE DISK
0780   C5B5 25 03                 BLO   LC5BA                       RETURN IF NO NEED TO WRITE OUT ALLOCATION TABLE
0781   C5B7 BD C7 1E              JSR   LC71E                       WRITE FILE ALLOCATION SECTOR TO DISK
0782   C5BA 35 D6          LC5BA  PULS  A,B,X,U,PC                  RESTORE REGISTERS
0783
0784                       * CONSOLE IN RAM VECTOR
0785   C5BC 96 6F          DVEC4  LDA   DEVNUM                       GET DEVICE NUMBER
0786   C5BE 10 2F C7 2F           LBLE  XVEC4                       BRANCH IF NOT DISK FILE
0787   C5C2 32 62                 LEAS  $02,S                       GET RID OF RETURN ADDRESS
0788   C5C4 34 14          LC5C4  PSHS  X,B                         SAVE REGISTERS
0789   C5C6 0F 70                 CLR   CINBFL                      CLEAR BUFFER NOT EMPTY FLAG
0790   C5C8 8E 09 26              LDX   #FCBV1-2                    POINT TO FILE BUFFER VECTOR TABLE
0791   C5CB D6 6F                 LDB   DEVNUM                      GET ACTIVE DISK FILE NUMBER
0792   C5CD 58                    ASLB                              TIMES 2 - TWO BYTES PER FCB ADDRESS
0793   C5CE AE 85                 LDX   B,X                         NOW X POINTS TO FILE BUFFER
0794   C5D0 E6 84                 LDB   FCBTYPE,X                   GET FILE TYPE
0795   C5D2 C1 40                 CMPB  #RANFIL                     IS THIS A RANDOM (DIRECT) FILE?
0796   C5D4 26 16                 BNE   LC5EC                       BRANCH IF NOT
0797
0798                       * GET A BYTE FROM A RANDOM FILE - RETURN CHAR IN ACCA
0799   C5D6 EC 88 15              LDD   FCBGET,X                    GET THE RECORD COUNTER
0800   C5D9 10 A3 09              CMPD  FCBRLN,X                    *COMPARE TO RECORD LENGTH AND
0801   C5DC 24 20                 BHS   LC5FE                       *BRANCH TO BUFFER EMPTY IF >= RECORD LENGTH
0802   C5DE C3 00 01              ADDD  #$0001                      = ADD ONE TO RECORD POINTER AND
0803   C5E1 ED 88 15              STD   FCBGET,X                    = SAVE IT IN FCB
0804   C5E4 AE 0B                 LDX   FCBBUF,X                    * POINT X TO START OF RANDOM FILE BUFFER AND
0805   C5E6 30 8B                 LEAX  D,X                         * ADD THE RECORD COUNTER TO IT
0806   C5E8 A6 1F                 LDA   $-1,X                       GET A CHARACTER FROM THE BUFFER
0807   C5EA 35 94                 PULS  B,X,PC                      RESTORE REGISTERS AND RETURN
0808                       * GET A BYTE FROM A SEQUENTIAL FILE
0809   C5EC E6 88 10       LC5EC  LDB   FCBCFL,X                    * TEST THE CACHE FLAG AND BRANCH IF AN
0810   C5EF 27 08                 BEQ   LC5F9                       * EXTRA CHARACTER HAS NOT BEEN READ FROM FILE
0811   C5F1 A6 88 11              LDA   FCBCDT,X                    GET THE CACHE CHARACTER
0812   C5F4 6F 88 10              CLR   FCBCFL,X                    CLEAR THE CACHE FLAG
0813   C5F7 35 94                 PULS  B,X,PC                      RESTORE REGISTERS AND RETURN
0814
0815   C5F9 E6 88 17       LC5F9  LDB   FCBDFL,X                    IS ANY DATA LEFT?
0816   C5FC 27 04                 BEQ   LC602                       BRANCH IF SO
0817   C5FE 03 70                 COM   CINBFL                      SET FLAG TO BUFFER EMPTY
0818   C600 35 94                 PULS  B,X,PC                      RESTORE REGISTERS AND RETURN
0819
0820   C602 E6 05          LC602  LDB   FCBCPT,X                    GET CHARACTER POINTER
0821   C604 6C 05                 INC   FCBCPT,X                    ADD ONE TO CHARACTER POINTER
0822   C606 6A 88 18              DEC   FCBLFT,X                    DECREMENT NUMBER OF CHARACTERS LEFT IN FILE BUFFER
0823   C609 27 06                 BEQ   LC611                       IF LAST CHARACTER, GO GET SOME MORE
0824   C60B 3A                    ABX                               ADD CHARACTER COUNTER TO X
0825   C60C A6 88 19              LDA   FCBCON,X                    GET DATA CHARACTER (SKIP PAST 25 FCB CONTROL BYTES
0826   C60F 35 94                 PULS  B,X,PC
0827                       * GET A CHARACTER FROM FCB DATA BUFFER - RETURN CHAR IN ACCA
0828   C611 34 60          LC611  PSHS  U,Y                         SAVE REGISTERS
0829   C613 4F                    CLRA                              *
0830   C614 33 8B                 LEAU  D,X                         * POINT U TO CORRECT CHARACTER
0831   C616 A6 C8 19              LDA   FCBCON,U                    =GET DATA CHAR (SKIP PAST 25 CONTROL BYTES)
0832   C619 34 02                 PSHS  A                           =AND SAVE DATA CHARACTER ON STACK
0833   C61B 6F 05                 CLR   FCBCPT,X                    RESET CHAR POINTER TO START OF BUFFER
0834   C61D A6 01                 LDA   FCBDRV,X                    GET DRIVE NUMBER AND SAVE IT IN
0835   C61F 97 EB                 STA   DCDRV                       DSKCON VARIABLE
0836   C621 8D 04                 BSR   LC627                       GO READ A SECTOR - FILL THE BUFFER
0837   C623 35 62                 PULS  A,Y,U                       RESTORE REGISTERS AND DATA CHARACTER
0838   C625 35 94                 PULS  B,X,PC                      RESTORE REGISTERS AND RETURN
0839                       * REFILL THE FCB INPUT DATA BUFFER FOR SEQUENTIAL FILES
0840   C627 A6 04          LC627  LDA   FCBSEC,X                    GET CURRENT SECTOR NUMBER
0841   C629 4C             LC629  INCA                              ADD ONE
0842   C62A 34 02                 PSHS  A                           SAVE NEW SECTOR NUMBER ON THE STACK
0843   C62C 81 09                 CMPA  #$09                        NINE SECTORS PER GRANULE
0844   C62E 23 01                 BLS   LC631                       BRANCH IF <= 9
0845   C630 4F                    CLRA                              SET TO SECTOR ZERO
0846   C631 A7 04          LC631  STA   FCBSEC,X                    SAVE SECTOR NUMBER
0847   C633 E6 03                 LDB   FCBCGR,X                    GET GRANULE NUMBET TO FAT POINTER
0848   C635 33 84                 LEAU  ,X                          POINT U TO FCB (TFR X,U)
0849   C637 BD C7 55              JSR   LC755                       POINT X TO PROPER FILE ALLOCATION TABLE
0850   C63A 3A                    ABX                               ADD OLD GRANULE NUMBER TO FAT POINTER
0851   C63B E6 06                 LDB   FATCON,X                    GET GRANULE NUMBER (6 CONTROL BYTES AT FRONT OF FAT)
0852   C63D 30 C4                 LEAX  ,U                          POINT X TO FCB
0853   C63F C1 C0                 CMPB  #$C0                        IS CURRENT GRANULE LAST ONE IN FILE?
0854   C641 24 0A                 BHS   LC64D                       YES
0855   C643 35 02                 PULS  A                           GET SECTOR NUMBER
0856   C645 80 0A                 SUBA  #10                         WAS IT 10? - OVERFLOW TO NEXT GRANULE IF SO
0857   C647 26 15                 BNE   LC65E                       BRANCH IF NOT
0858   C649 E7 03                 STB   FCBCGR,X                    SAVE NEW GRANULE NUMBER
0859   C64B 20 DC                 BRA   LC629                       SET VARIABLES FOR NEW GRANULE
0860   C64D C4 3F          LC64D  ANDB  #$3F                        GET NUMBER OF SECTORS USED IN THIS GRANULE
0861   C64F C1 09                 CMPB  #$09                        9 SECTORS / GRANULE
0862   C651 23 05                 BLS   LC658                       BRANCH IF OK
0863   C653 C6 40          LC653  LDB   #2*32                       'BAD FILE STRUCTURE' ERROR
0864   C655 7E AC 46              JMP   LAC46                       ERROR DRIVER
```

```
0865   C658 E0 E0          LC658   SUBB  ,S+                          SUBTRACT CURRENT SECTOR NUMBER AND PULS A
0866   C65A 25 21                  BLO   LC67D                         BRANCH IF PAST LAST SECTOR
0867   C65C 1F 98                  TFR   B,A                           SECTOR NUMBER TO ACCA
0868   C65E 34 02          LC65E   PSHS  A                             SAVE SECTOR NUMBER DIFFERENCE
0869   C660 8D 23                  BSR   LC685                         INCREMENT RECORD NUMBER
0870   C662 86 02                  LDA   #$02                          *GET READ OPERATION CODE
0871   C664 97 EA                  STA   DCOPC                         *AND SAVE IT IN DSKCON VARIABLE
0872   C666 BD C7 63               JSR   LC763                         GET PROPER TRACK AND SECTOR TO DSKCON VARIABLES
0873   C669 33 88 19               LEAU  FCBCON,X                      * POINT U TO START OF FCB DATA BUFFER
0874   C66C DF EE                  STU   DCBPT                         * AND SAVE IT IN DSKCON VARIABLE
0875   C66E BD D6 F2               JSR   LD6F2                         GO READ A SECTOR INTO FCB BUFFER
0876   C671 6F 88 18               CLR   FCBLFT,X                      NUMBER OF CHARS LEFT IN BUFFER = 256
0877   C674 E6 E0                  LDB   ,S+                           GET SECTOR NUMBER OFF STACK
0878   C676 26 0C                  BNE   LC684                         RETURN IF DATA LEFT; FALL THRU IF LAST SECTOR
0879   C678 EC 88 13               LDD   FCBLST,X                      GET NUMBER OF BYTES IN THE LAST SECTOR
0880   C67B 26 04                  BNE   LC681                         BRANCH IF SOME BYTES IN LAST SECTOR
0881   C67D 5F             LC67D   CLRB                                SET NUMBER OF REMAINING BYTES = 256
0882   C67E 63 88 17               COM   FCBDFL,X                      SET DATA LEFT FLAG TO $FF
0883   C681 E7 88 18       LC681   STB   FCBLFT,X                      SAVE THE NUMBER OF CHARS LEFT IN BUFFER
0884   C684 39             LC684   RTS
0885
0886   C685 EE 07          LC685   LDU   FCBREC,X                      GET CURRENT RECORD NUMBER
0887   C687 33 41                  LEAU  $01,U                         BUMP IT
0888   C689 EF 07                  STU   FCBREC,X                      PUT IT BACK
0889   C68B 39                     RTS
0890
0891                       * SCAN DIRECTORY FOR FILENAME.EXT FOUND IN DNAMBF. IF FILENAME FOUND,
0892                       * RETURN WITH SECTOR NUMBER IN V973, GRANULE IN V976 AND RAM BUFFER
0893                       * CONTAINING DIRECTORY DATA IN V974. IF DISK IS FULL THEN V973,
0894                       * V977 = 0. THE FIRST UNUSED SECTOR RETURNED IN V977, RAM IMAGE IN V978
0895   C68C 7F 09 73       LC68C   CLR   V973                          CLEAR SECTOR NUMBER
0896   C68F 7F 09 77               CLR   V977                          CLEAR TEMP SECTOR COUNTER
0897   C692 CC 11 02               LDD   #$1102                        TRACK 17 (DIRECTORY), READ OPERATION CODE
0898   C695 97 EC                  STA   DCTRK                         SAVE TRACK NUMBER
0899   C697 D7 EA                  STB   DCOPC                         SAVE OPERATION CODE (READ)
0900   C699 C6 03                  LDB   #$03                          READ SECTOR 3 (FIRST DIRECTORY SECTOR)
0901   C69B D7 ED          LC69B   STB   DSEC                          SAVE SECTOR NUMBER IN DSKCON VARIABLE
0902   C69D CE 06 00               LDU   #DBUF0                        *BUFFER AREA NUMBER 0 AS DATA BUFFER - SAVE
0903   C6A0 DF EE                  STU   DCBPT                         *IN DSKCON VARIABLE
0904   C6A2 BD D6 F2               JSR   LD6F2                         GO READ A SECTOR
0905   C6A5 FF 09 74       LC6A5   STU   V974                          SAVE RAM DIRECTORY BUFFER ADDRESS
0906   C6A8 31 C4                  LEAY  ,U                            POINT Y TO DIRECTORY BUFFER
0907   C6AA A6 C4                  LDA   ,U                            GET A BYTE FROM BUFFER
0908   C6AC 26 28                  BNE   LC6D6                         BRANCH IF NOT ZERO - FILE IS ACTIVE
0909   C6AE 8D 29                  BSR   LC6D9                         SET UNUSED FILE POINTERS IF ENTRY HAS BEEN KILLED
0910   C6B0 8E 09 4C       LC6B0   LDX   #DNAMBF                       POINT TO DISK FILE NAME BUFFER
0911   C6B3 A6 80          LC6B3   LDA   ,X+                           *COMPARE THE FILENAME AND EXTENSION
0912   C6B5 A1 C0                  CMPA  ,U+                           *STORED IN RAM AT DNAMBF TO THE DIRECTORY
0913   C6B7 26 0E                  BNE   LC6C7                         *ENTRY STORED AT ,U (BRANCH IF MISMATCH)
0914   C6B9 8C 09 57               CMPX  #DNAMBF+11                    AT END OF FILE NAME BUFFER?
0915   C6BC 26 F5                  BNE   LC6B3                         BRANCH IF NOT DONE CHECKING FILENAME
0916   C6BE F7 09 73               STB   V973                          SAVE SECTOR NUMBER IN DSKCON VARIABLE
0917   C6C1 A6 42                  LDA   FCBFGR,U                      *GET NUMBER OF FIRST GRANULE IN FILE
0918   C6C3 B7 09 76               STA   V976                          *AND SAVE IT IN V976
0919   C6C6 39                     RTS
0920
0921   C6C7 33 A8 20       LC6C7   LEAU  DIRLEN,Y                      GET NEXT DIRECTORY ENTRY (DIRLEN BYTES PER ENTRY)
0922   C6CA 11 83 07 00            CMPU  #DBUF0+SECLEN                 AT END OF BUFFER?
0923   C6CE 26 D5                  BNE   LC6A5                         CHECK NEXT ENTRY IF NOT AT END
0924   C6D0 5C                     INCB                                NEXT SECTOR
0925   C6D1 C1 0B                  CMPB  #11                           11 SECTORS MAX IN DIRECTORY
0926   C6D3 23 C6                  BLS   LC69B                         BRANCH IF MORE SECTORS
0927   C6D5 39                     RTS
0928
0929   C6D6 43             LC6D6   COMA                                COMPLEMENT FIRST BYTE IN DIRECTORY EMTRY
0930   C6D7 26 D7                  BNE   LC6B0                         BRANCH IF FILE IS ACTIVE - FALL THRU IF NOT USED
0931
0932                       * SET POINTERS FOR FIRST UNUSED DIRECTORY ENTRY
0933   C6D9 B6 09 77       LC6D9   LDA   V977                          UNUSED ENTRY ALREADY FOUND?
0934   C6DC 26 06                  BNE   DVEC12                        RETURN IF UNUSED ENTRY ALREADY FOUND
0935   C6DE F7 09 77               STB   V977                          SECTOR CONTAINING THIS DIRECTORY ENTRY
0936   C6E1 FF 09 78               STU   V978                          POINTS TO RAM AREA WHERE DIRECTORY DATA IS STORED
0937   C6E4 39             DVEC12  RTS
0938
0939   C6E5 C6 34          LC6E4   LDB   #2*26                         'NE' ERROR
0940   C6E7 7D 09 73               TST   V973                          WAS A DIRECTORY MATCH FOUND?
0941   C6EA 26 F8                  BNE   DVEC12                        RETURN IF FOUND
0942   C6EC 7E AC 46               JMP   LAC46                         JUMP TO ERROR HANDLER IF NOT FOUND
0943
0944                       * KILL COMMAND
0945   C6EF BD C9 35       KILL    JSR   LC935                         GET FILENAME.EXT FROM BASIC
0946   C6F2 BD A5 C7               JSR   LA5C7                         'SYNTAX' ERROR IF MORE CHARACTERS ON LINE
0947   C6F5 BD C7 9D               JSR   LC79D                         GET VALID FAT DATA
0948   C6F8 8D 92                  BSR   LC68C                         TEST FOR FILE NAME MATCH IN DIRECTORY
0949   C6FA 8D E9                  BSR   LC6E5                         MAKE SURE THE FILE EXISTED
0950   C6FC 86 FF          LC6FC   LDA   #$FF                          * MATCH FILE TYPE = $FF; THIS WILL CAUSE AN 'AO'
0951                       *                                           * ERROR TO BE GENERATED IF ANY FILE TYPE IS OPEN
0952   C6FE BD C8 07               JSR   LC807                         CHECK TO MAKE SURE FILE IS NOT OPEN
0953   C701 BE 09 74               LDX   V974                          *GET RAM IMAGE OF DIRECTORY
0954   C704 6F 84                  CLR   DIRNAM,X                      *AND ZERO FIRST BYTE - KILL FILE
0955   C706 C6 03                  LDB   #$03                          =WRITE OPERATION CODE - SAVE
0956   C708 D7 EA                  STB   DCOPC                         =IT IN DSKCON VARIABLE
0957   C70A BD D6 F2               JSR   LD6F2                         WRITE A SECTOR
0958   C70D E6 0D                  LDB   DIRGRN,X                      GET NUMBER OF FIRST GRANULE IN FILE
0959   C70F 8D 44          LC70F   BSR   LC755                         POINT X TO PROPER FILE ALLOCATION TABLE
0960   C711 30 06                  LEAX  FATCON,X                      SKIP 6 CONTROL BYTES
```

```
0961   C713 3A                  ABX                             POINT TO CORRECT ENTRY
0962   C714 E6 84               LDB   ,X                        GET NEXT GRANULE
0963   C716 86 FF               LDA   #$FF                      *GET FREE GRANULE FLAG AND
0964   C718 A7 84               STA   ,X                        *MARK GRANULE AS FREE
0965   C71A C1 C0               CMPB  #$C0                      WAS THIS THE LAST GRANULE?
0966   C71C 25 F1               BLO   LC70F                     * KEEP FREEING GRANULES IF NOT LAST ONE
0967                   *                                        * WRITE FILE ALLOCATION SECTOR TO DIRECTORY - DO NOT WRITE
0968                   *                                        * THE SIX CONTROL BYTES AT THE START OF THE FAT TO THE DISK
0969   C71E CE 06 00    LC71E   LDU   #DBUF0                    =POINT U TO DISK BUFFER 0 AND
0970   C721 DF EE               STU   DCBPT                     =SAVE IT AS DSKCON VARIABLE
0971   C723 CC 11 03            LDD   #$1103                    * WRITE DIRECTORY TRACK - SAVE
0972   C726 97 EC               STA   DCTRK                     * TRACK AND WRITE OPERATION CODE IN
0973   C728 D7 EA               STB   DCOPC                     * DSKCON VARIABLES
0974   C72A C6 02               LDB   #$02                      = GET FILE ALLOCATION SECTOR AND
0975   C72C D7 ED               STB   DSEC                      = SAVE IN DSKCON VARIABLE
0976   C72E 8D 25               BSR   LC755                     POINT X TO PROPER FILE ALLOCATION TABLE
0977   C730 6F 01               CLR   FAT1,X                    RESET FLAG INDICATING VALID FAT DATA HAS BEEN STORED ON DISK
0978   C732 30 06               LEAX  FATCON,X                  MOVE (X) TO START OF GRANULE DATA
0979   C734 C6 44               LDB   #GRANMX                   68 BYTES IN FAT
0980   C736 BD A5 9A            JSR   LA59A                     MOVE ACCB BYTES FROM FAT RAM IMAGE TO DBUF0
0981
0982                   * ZERO OUT ALL OF THE BYTES IN THE FAT SECTOR WHICH DO NOT CONTAIN THE GRANULE DATA
0983   C739 6F C0       LC739   CLR   ,U+                       CLEAR A BYTE
0984   C73B 11 83 07 00         CMPU  #DBUF0+SECLEN             FINISHED THE WHOLE SECTOR?
0985   C73F 26 F8               BNE   LC739                     NO
0986   C741 7E D6 F2            JMP   LD6F2                     WRITE A SECTOR
0987
0988                   * ENTER WITH ACCB CONTAINING FILE NUMBER (1-15); EXIT WITH X POINTING
0989                   * TO CORRECT FILE BUFFER; FLAGS SET ACCORDING TO FILE TYPE.
0990
0991   C744 34 04       LC744   PSHS  B                         SAVE FILE NUMBER ON STACK
0992   C746 D6 6F               LDB   DEVNUM                    GET DEVICE NUMBER (FILE NUMBER)
0993   C748 8C                  CMPX  #$3404                    SKIP TWO BYTES
0994   C749 34 04       LC719   PSHS  B                         SAVE FILE NUMBER ON STACK
0995   C74B 58                  ASLB                            X2: 2 BYTES PER POINTER
0996   C74C 8E 09 26            LDX   #FCBV1-2                  POINT X TO START OF FCB POINTERS
0997   C74F AE 85               LDX   B,X                       POINT X TO PROPER FCB
0998   C751 E6 00               LDB   FCBTYP,X                  SET FLAGS ACCORDING TO FILE TYPE
0999   C753 35 84               PULS  B,PC                      RESTORE FILE NUMBER
1000
1001                   * POINT X TO DRIVE ALLOCATION TABLE
1002
1003   C755 34 06       LC755   PSHS  B,A                       SAVE ACCD ON STACK
1004   C757 96 EB               LDA   DCDRV                     GET DRIVE NUMBER
1005   C759 C6 4A               LDB   #FATLEN                   GET LENGTH OF FILE ALLOCATION TABLE
1006   C75B 3D                  MUL                             MULTIPLY BY DRIVE NUMBER TO GET OFFSET
1007   C75C 8E 08 00            LDX   #FATBL0                   START OF FILE ALLOCATION TABLE
1008   C75F 30 8B               LEAX  D,X                       POINT TO RIGHT TABLE
1009   C761 35 86               PULS  A,B,PC                    RESTORE ACCD
1010
1011                   * CONVERT GRANULE NUMBER TO TRACK & SECTOR NUMBER - X MUST BE POINTING TO CORRECT
1012                   * FCB; THE TRACK AND SECTOR NUMBER WILL BE STORED IN DSKCON REGISTERS
1013   C763 E6 03       LC763   LDB   FCBCGR,X                  GET GRANULE NUMBER
1014   C765 54                  LSRB                            DIVIDE BY 2 - 2 GRANULES / TRACK
1015   C766 D7 EC               STB   DCTRK                     TRACK NUMBER
1016   C768 C1 11               CMPB  #17                       TRACK 17 = DIRECTORY TRACK
1017   C76A 25 02               BLO   LC76E                     BRANCH IF < DIRECTORY TRACK
1018   C76C 0C EC               INC   DCTRK                     INCR TRACK NUMBER IF > DIRECTORY TRACK
1019   C76E 58          LC76E   ASLB                            MULTIPLY TRACK NUMBER BY 2
1020   C76F 50                  NEGB                            NEGATE GRANULE NUMBER
1021   C770 EB 03               ADDB  FCBCGR,X                  B=0 IF EVEN GRANULE; 1 IF ODD
1022   C772 8D 05               BSR   LC779                     RETURN B=0 FOR EVEN GRANULE NUMBER, B=9 FOR ODD GRANULE NUMBER
1023   C774 EB 04               ADDB  FCBSEC,X                  ADD SECTOR NUMBER
1024   C776 D7 ED               STB   DSEC                      SAVE SECTOR NUMBER
1025   C778 39                  RTS
1026                   * MULTIPLY ACCD BY 9
1027   C779 34 06       LC778   PSHS  B,A                       TEMP STORE ACCD ON STACK
1028   C77B 58                  ASLB                            *
1029   C77C 49                  ROLA                            * MULTIPLY BY 2
1030   C77D 58                  ASLB                            =
1031   C77E 49                  ROLA                            = MULTIPLY BY FOUR
1032   C77F 58                  ASLB                            *
1033   C780 49                  ROLA                            * MULTIPLY BY EIGHT
1034   C781 E3 E1               ADDD  ,S++                      ADD ONE = MULTIPLY BY NINE
1035   C783 39                  RTS
1036
1037                   * CONVERT ACCD INTO A GRANULE NUMBER - RETURN RESULT IN ACCB;
1038                   * ENTER WITH ACCD CONTAINING A NUMBER OF SECTORS. RETURN IN ACCB
1039                   * THE NUMBER (0-67) CORRESPONDING TO THE NUMBER OF COMPLETE
1040                   * GRANULES CONTAINED IN THAT MANY SECTORS.
1041                   * DIVIDE BY 90, MULTIPLY BY 10 IS FASTER THAN DIVIDE BY 9
1042   C784 6F E2       LC784   CLR   ,-S                       CLEAR A TEMPORARY SLOT ON THE STACK
1043   C786 6C E4       LC756   INC   ,S                        * DIVIDE ACCD BY 90 - SAVE THE
1044   C788 83 00 5A            SUBD  #9*10                     * QUOTIENT+1 ON THE STACK - REMAINDER
1045   C78B 2A F9               BPL   LC786                     * IN ACCB
1046   C78D A6 E4               LDA   ,S                        = PUT THE QUOTIENT+1 IN ACCA AND
1047   C78F E7 E4               STB   ,S                        = SAVE REMAINDER ON STACK
1048   C791 C6 0A               LDB   #10                       * MULTIPLY (QUOTIENT+1)
1049   C793 3D                  MUL                             * BY 10
1050   C794 35 02               PULS  A                         PUT THE REMAINDER IN ACCA
1051   C796 5A          LC796   DECB                            * DECREMENT THE GRANULE COUNT BY ONE FOR
1052   C797 8B 09               ADDA  #$09                      * EVERY NINE SECTORS (1 GRANULE) IN THE
1053   C799 2B FB               BMI   LC796                     * REMAINDER - COMPENSATE FOR THE + 1 IN QUOTIENT+1
1054   C79B 4F                  CLRA                            CLEAR MS BYTE OF ACCD
1055   C79C 39          LC79C   RTS
1056
```

```
1057                            * MAKE SURE RAM FILE ALLOCATION TABLE DATA IS VALID
1058   C79D 8D B6               LC79D   BSR   LC755                           POINT X TO FAT FOR THE CORRECT DRIVE NUMBER
1059   C79F 6D 00                       TST   FAT0,X                          CHECK TO SEE IF ANY FILES ARE ACTIVE
1060   C7A1 26 F9                       BNE   LC79C                           RETURN IF ANY FILES ACTIVE IN THIS FAT
1061   C7A3 6F 01                       CLR   FAT1,X                          RESET FAT DATA VALID FLAG
1062   C7A5 33 06                       LEAU  FATCON,X                        LOAD U WITH START OF GRANULE DATA BUFFER
1063   C7A7 8E 06 00                    LDX   #DBUF0                          BUFFER FOR DISK TRANSFER
1064   C7AA 9F EE                       STX   DCBPT                           PUT IN DSKCON PARAMETER
1065   C7AC CC 11 02                    LDD   #$1102                          DIRECTORY TRACK, READ SECTOR
1066   C7AF 97 EC                       STA   DCTRK                           STORE IN DSKCON TRACK NUMBER
1067   C7B1 D7 EA                       STB   DCOPC                           STORE IN DSKCON OP CODE
1068   C7B3 C6 02                       LDB   #$02                            GET SECTOR NUMBER 2 (FILE ALLOCATION TABLE)
1069   C7B5 D7 ED                       STB   DSEC                            STORE IN DSKCON PARAMETER
1070   C7B7 BD D6 F2                    JSR   LD6F2                           GO READ SECTOR
1071   C7BA C6 44                       LDB   #GRANMX                         TRANSFER FILE ALLOCATION TABLE TO FILE ALLOC TABLE BUFFER
1072   C7BC 7E A5 9A                    JMP   LA59A                           MOVE B BYTES FROM (X) TO (U)
1073
1074                            * FIND FIRST FREE GRANULE - ENTER WITH ACCB CONTAINING
1075                            * GRANULE FROM WHICH TO START SEARCHING. THE FOUND GRANULE
1076                            * IS MARKED BY STORING A $C0 IN THE GRANULE'S DATA BYTE
1077                            * TO INDICATE THAT IT IS THE LAST GRANULE IN THE FILE.
1078                            * RETURN WITH FIRST FREE GRANULE FOUND IN ACCA
1079   C7BF 8D 94               LC7BF   BSR   LC755                           POINT X TO FILE ALLOC TABLE
1080   C7C1 30 06                       LEAX  FATCON,X                        SKIP CONTROL BYTES
1081   C7C3 4F                          CLRA                                  USE ACCA AS GRANULE COUNTER
1082   C7C4 C4 FE                       ANDB  #$FE                            MASK OFF BIT ZERO OF SEARCH GRANULE
1083   C7C6 6F E2                       CLR   ,-S                             INITIALIZE AND SAVE A BYTE ON STACK (DIRECTION FLAG)
1084   C7C8 63 85               LC7C8   COM   B,X                             IS THIS GRANULE FREE? ($FF=FREE)
1085   C7CA 27 31                       BEQ   LC7FD                           BRANCH IF IT IS
1086   C7CC 63 85                       COM   B,X                             RESTORE GRANULE DATA
1087   C7CE 4C                          INCA                                  ADD ONE TO GRANULE COUNTER
1088   C7CF 81 44                       CMPA  #GRANMX                         GRANMX GEANULES PER DISK
1089   C7D1 24 25                       BHS   LC7F8                           BRANCH IF ALL GRANULES CHECKED (DISK FULL)
1090   C7D3 5C                          INCB                                  INCR TO NEXT GRANULE
1091   C7D4 C5 01                       BITB  #$01                            IS BIT 0 SET?
1092   C7D6 26 F0                       BNE   LC7C8                           BRANCH IF ODD GRANULE NUMBER (SAME TRACK)
1093   C7D8 34 06                       PSHS  B,A                             SAVE GRANULE COUNTER AND CURRENT GRANULE NUMBER
1094   C7DA C0 02                       SUBB  #$02                            SUBTRACT ONE TRACK (2 GRANULES)
1095   C7DC 63 62                       COM   $02,S                           COMPLEMENT DIRECTION FLAG
1096   C7DE 26 0C                       BNE   LC7EC                           BRANCH EVERY OTHER TIME
1097   C7E0 E0 E0               LC7E0   SUBB  ,S+                             SUBTRACT THE GRANULE COUNTER FROM THE CURRENT GRANULE NUMBER
1098   C7E2 2A 04                       BPL   LC7E8                           BRANCH IF LOWER BOUND NOT EXCEEDED
1099   C7E4 E6 E4                       LDB   ,S                              RESTORE CURRENT GRANULE NUMBER IF LOWER BOUND EXCEEDED
1100   C7E6 63 61               LC7E6   COM   $01,S                           * COMPLEMENT FLAG - IF GRANULE NUMBER HAS EXCEEDED
1101                            *                                             * BOUNDS ON EITHER THE HI OR LO SIDE, FORCE IT TO GO IN
1102                            *                                             * THE DIRECTION OPPOSITE THE EXCEEDED BOUND
1103   C7E8 32 61               LC7E8   LEAS  $01,S                           CLEAN UP STACK
1104   C7EA 20 DC                       BRA   LC7C8                           CHECK FOR ANOTHER FREE GRANULE
1105
1106   C7EC EB E0               LC7EC   ADDB  ,S+                             ADD THE GRANULE COUNTER TO THE CURRENT GRANULE NUMBER
1107   C7EE C1 44                       CMPB  #GRANMX                         GRANMX GRANULES PER DISK
1108   C7F0 25 F6                       BLO   LC7E8                           BRANCH IF UPPER BOUND NOT EXCEEDED
1109   C7F2 E6 E4                       LDB   ,S                              * RESTORE CURRENT GRANULE COUNT AND GO TWICE
1110   C7F4 C0 04                       SUBB  #$04                            * AS FAR AS USUAL IN OPPOSITE DIRECTION IF UPPER BOUND EXCEEDED
1111   C7F6 20 EE                       BRA   LC7E6                           KEEP SEARCHING
1112   C7F8 C6 38               LC7F8   LDB   #2*28                           'DISK FULL' ERROR
1113   C7FA 7E AC 46                    JMP   LAC46                           JUMP TO ERROR HANDLER
1114
1115                            * POINT X TO FIRST FREE GRANULE POSITION IN THE FILE ALLOCATION
1116                            * TABLE AND MARK THE POSITION WITH A LAST GRANULE IN FILE MARKER
1117   C7FD 32 61               LC7FD   LEAS  $01,S                           CLEAR UP STACK - REMOVE DIRECTION FLAG
1118   C7FF 1F 98                       TFR   B,A                             GRANULE NUMBER TO ACCA
1119   C801 3A                          ABX                                   POINT X TO FIRST FOUND GRANULE
1120   C802 C6 C0                       LDB   #$C0                            LAST GRANULE FLAG
1121   C804 E7 84                       STB   ,X                              MARK THE FIRST FOUND GRANULE AS THE LAST GRANULE
1122   C806 39                  LC806   RTS
1123
1124                            * CHECK ALL ACTIVE FILES TO MAKE SURE A FILE IS NOT ALREADY OPEN - TO BE OPEN
1125                            * A FILE BUFFER MUST MATCH THE DRIVE NUMBER AND FIRST GRANULE NUMBER
1126                            * IN RAM DIRECTORY ENTRY AND THE FCB TYPE MUST NOT MATCH THE FILE TYPE IN ACCA
1127                            * AN 'AO' ERROR WILL NOT BE GENERATED IF A FILE IS BEING OPENED FOR
1128                            * THE SAME MODE THAT IT HAS ALREADY BEEN OPENED UNDER.
1129
1130   C807 34 02               LC807   PSHS  A                               SAVE FILE TYPE ON STACK
1131   C809 F6 09 5B                    LDB   FCBACT                          NUMBER OF CURRENTLY OPEN FILES
1132   C80C 5C                          INCB                                  ADD ONE MORE TO FILE COUNTER
1133   C80D BD C7 49            LC80D   JSR   LC749                           POINT X TO FCB OF THIS FILE
1134   C810 27 17                       BEQ   LC829                           BRANCH IF BUFFER NOT BEING USED
1135   C812 96 EB                       LDA   DCDRV                           * GET DRIVE NUMBER AND CHECK TO SEE IF IT
1136   C814 A1 01                       CMPA  FCBDRV,X                        * MATCHES THE DRIVE NUMBER FOR THIS BUFFER
1137   C816 26 11                       BNE   LC829                           FILE EXISTS ON ANOTHER DRIVE
1138   C818 FE 09 74                    LDU   V974                            GET RAM DIRECTORY AREA
1139   C81B A6 4D                       LDA   DIRGRAN,U                       GET FIRST GRANULE IN FILE
1140   C81D A1 02                       CMPA  FCBFGR,X                        DOES IT MATCH THIS FILE BUFFER?
1141   C81F 26 08                       BNE   LC829                           NO
1142   C821 A6 00                       LDA   FCBTYP,X                        GET FILE TYPE OF THIS BUFFER
1143   C823 A1 E4                       CMPA  ,S                              DOES IT MATCH THE ONE WE ARE LOOKING FOR?
1144   C825 10 26 DD F3                 LBNE  LA61C                           'FILE ALREADY OPEN' ERROR IF NOT
1145   C829 5A                  LC829   DECB                                  DECR FILE COUNTER
1146   C82A 26 E1                       BNE   LC80D                           BRANCH IF HAVEN'T CHECKED ALL ACTIVE FILES
1147   C82C 35 82                       PULS  A,PC                            RESTORE FILE TYPE AND RETURN
1148
1149   C82E BD A5 A5            LC82E   JSR   LA5A5                           EVALUATE AN EXPRESSION (DEVICE NUMBER)
1150   C831 0F 6F                       CLR   DEVNUM                          SET DEVICE NUMBER TO SCREEN
1151   C833 5D                          TSTB                                  TEST NEW DEVICE NUMBER
1152   C834 10 2F EC 12                 LBLE  LB44A                           'FC' ERROR IF DEVICE NUMBER NOT A DISK FILE
```

```
1153   C838 BD C7 49                JSR   LC749                      POINT X TO FCB
1154   C83B A6 00                   LDA   FCBTYP,X                   TEST IF BUFFER IS IN USE
1155   C83D 10 27 DB BA             LBEQ  LA3FB                      'FILE NOT OPEN' ERROR
1156   C841 81 40                   CMPA  #RANFIL                    DIRECT/RANDOM FILE?
1157   C843 27 C1                   BEQ   LC806                      RETURN IF RANDOM
1158   C845 7E A6 16        LC856   JMP   LA616                      BAD FILE MODE ERROR IF NOT RANDOM
1159
1160                        * INPUT DEVICE NUMBER CHECK RAM HOOK
1161   C848 86 10           DVEC5   LDA   #INPFIL                    INPUT FILE TYPE
1162   C84A 8C              LC84A   CMPX  #8620                      SKIP TWO BYTES
1163
1164                        * PRINT DEVICE NUMBER CHECK RAM HOOK
1165   C84B 86 20           DVEC6   LDA   #OUTFIL                    OUTPUT FILE TYPE
1166   C84D 0D 6F                   TST   DEVNUM                     * CHECK DEVICE NUMBER AND RETURN IF
1167   C84F 2F B5                   BLE   LC806                      * NOT A DISK FILE
1168   C851 AF E4                   STX   ,S                         = REPLACE SUBROUTINE RETURN ADDRESS WITH X REGISTER -
1169                        *                                        = THIS IS THE SAME AS LEAS 2,S AND PSHS X
1170   C853 BD C7 44                JSR   LC744                      POINT X TO FCB
1171   C856 34 06                   PSHS  B,A                        SAVE ACCB AND FILE TYPE ON STACK
1172   C858 A6 00                   LDA   FCBTYP,X                   GET FILE TYPE
1173   C85A 10 27 DB 9D             LBEQ  LA3FB                      'FILE NOT OPEN' ERROR
1174   C85E 81 40                   CMPA  #RANFIL                    RANDOM FILE?
1175   C860 27 06                   BEQ   LC868                      BRANCH IF RANDOM FILE
1176   C862 A1 E4                   CMPA  ,S                         IS THIS FCB OF THE PROPER TYPE?
1177   C864 26 DF                   BNE   LC845                      'FILE MODE' ERROR IF NOT
1178   C866 35 96           LC866   PULS  A,B,X,PC                   RESTORE ACCB,X,ACCA (FILE TYPE) AND RETURN
1179
1180   C868 AE 64           LC868   LDX   $04,S                      * GET CALLING ADDRESS FROM THE STACK AND
1181   C86A 8C B0 0C                CMPX  #LB00C                     * RETURN UNLESS COMING FROM
1182   C86D 26 F7                   BNE   LC866                      * BASIC'S 'INPUT' STATEMENT
1183   C86F BD B2 6D                JSR   SYNCOMMA                   SYNTAX CHECK FOR A COMMA
1184   C872 81 22                   CMPA  #'"'                       CHECK FOR A DOUBLE QUOTE
1185   C874 26 0B                   BNE   LC881                      RETURN TO BASIC'S 'INPUT' COMMAND
1186   C876 BD B2 44                JSR   LB244                      STRIP PROMPT STRING FROM BASIC AND PUT IT ON THE STRING STACK
1187   C879 BD B6 57                JSR   LB657                      PURGE THE STRING PUT ON THE STRING STACK
1188   C87C C6 3B                   LDB   #';'                       SEMICOLON
1189   C87E BD B2 6F                JSR   LB26F                      DO A SYNTAX CHECK FOR SEMICOLON
1190   C881 8E B0 1E        LC881   LDX   #LB01E                     GET MODIFIED REENTRY POINT INTO BASIC
1191   C884 AF 64                   STX   $04,S                      AND PUT IT INTO THE RETURN ADDRESS ON THE STACK
1192   C886 35 96                   PULS  A,B,X,PC                   RETURN TO BASIC
1193
1194                        * DEVICE NUMBER VALIDITY CHECK RAM HOOK
1195   C888 2F 25           DVEC1   BLE   LC8AF                      RETURN IF NOT A DISK FILE
1196   C88A F1 09 5B                CMPB  FCBACT                     COMPARE DEVICE NUMBER TO HIGHEST POSSIBLE
1197   C88D 10 22 DD 8E             LBHI  LA61F                      'DEVICE NUMBER' ERROR IF TOO BIG
1198   C891 35 90                   PULS  X,PC                       RETURN
1199
1200                        * SET PRINT PARAMETERS RAM HOOK
1201   C893 0D 6F           DVEC2   TST   DEVNUM                      *CHECK DEVICE NUMBER AND
1202   C895 2F 18                   BLE   LC8AF                      *RETURN IF NOT DISK FILE
1203   C897 32 62                   LEAS  $02,S                      PURGE RETURN ADDRESS OFF OF THE STACK
1204   C899 34 16                   PSHS  X,B,A                      SAVE REGISTERS
1205   C89B 0F 6E                   CLR   PRTDEV                     SET PRINT DEVICE NUMBER TO NON-CASSETTE
1206   C89D BD C7 44                JSR   LC744                      POINT X TO FCB
1207   C8A0 E6 06                   LDB   FCBPOS,X                   GET PRINT POSITION
1208   C8A2 4F                      CLRA                             PRINT WIDTH (256)
1209   C8A3 8E 10 00                LDX   #$1000                     TAB FIELD WIDTH AND TAB ZONE
1210   C8A6 7E A3 7C                JMP   LA37C                      SAVE THE PRINT PARAMETERS
1211
1212                        * BREAK CHECK RAM HOOK
1213   C8A9 0D 6F           DVEC11  TST   DEVNUM                      * CHECK DEVICE NUMBER AND RETURN
1214   C8AB 2F 02                   BLE   LC8AF                      * IF NOT A DISK FILE
1215   C8AD 32 62                   LEAS  $02,S                      = PURGE RETURN ADDRESS OFF OF THE STACK - DON'T
1216   C8AF 39              LC8AF   RTS                              = DO A BREAK CHECK IF DISK FILE
1217
1218                        * COMMAND INTERPRETATION RAM HOOK
1219   C8B0 32 62           DVEC20  LEAS  $02,S                      PURGE RETURN ADDRESS OFF OF THE STACK
1220   C8B2 1C AF           LC8B2   ANDCC #$AF                       ENABLE IRQ & FIRQ
1221   C8B4 7F FF 02                CLR   PIA0+2                     STROBE ALL KEYS (COLUMN STROBE)
1222   C8B7 B6 FF 00                LDA   PIA0                       READ KEYBOARD ROWS
1223   C8BA 43                      COMA                             INVERT KEYBOARD ROW DATA
1224   C8BB 84 7F                   ANDA  #$7F                       MASK OFF JOYSTICK INPUT BIT
1225   C8BD 27 03                   BEQ   LC8C2                      BRANCH IF NO KEY DOWN
1226   C8BF BD AD EB                JSR   LADEB                      GO DO A BREAK CHECK IF A KEY IS DOWN
1227   C8C2 9E A6           LC8C2   LDX   CHARAD                     GET INPUT POINTER INTO X
1228   C8C4 9F 2F                   STX   TINPTR                     TEMP SAVE IT
1229   C8C6 A6 80                   LDA   ,X+                        SEARCH FOR THE END OF CURRENT LINE
1230   C8C8 27 07                   BEQ   LC8D1                      BRANCH IF END OF LINE
1231   C8CA 81 3A                   CMPA  #':'                       CHECK FOR END OF SUB LINE, TOO
1232   C8CC 27 25                   BEQ   LC8F3                      BRANCH IF END OF SUB LINE
1233   C8CE 7E B2 77                JMP   LB277                      'SYNTAX' ERROR IF NOT END OF LINE
1234   C8D1 A6 81           LC8D1   LDA   ,X++                       *GET MS BYTE OF ADDRESS OF NEXT BASIC LINE
1235   C8D3 97 00                   STA   ENDFLAG                    *AND SAVE IT IN CURLIN
1236   C8D5 26 03                   BNE   LC8DA                      BRANCH IF NOT END OF PROGRAM
1237   C8D7 7E AE 15                JMP   LAE15                      GO 'STOP' THE SYSTEM
1238   C8DA EC 80           LC8DA   LDD   ,X+                        *GET LINE NUMBER OF THIS LINE AND
1239   C8DC DD 68                   STD   CURLIN                     *SAVE IT IN CURLIN
1240   C8DE 9F A6                   STX   CHARAD                     RESET BASIC'S INPUT POINTER
1241   C8E0 96 AF                   LDA   TRCFLG                     * CHECK THE TRACE FLAG AND
1242   C8E2 27 0F                   BEQ   LC8F3                      * BRANCH IF TRACE OFF
1243   C8E4 86 5B                   LDA   #'<'                       < LEFT DELIMITER OF TRON
1244   C8E6 BD A2 82                JSR   LA282                      SEND CHARACTER TO CONSOLE OUT
1245   C8E9 96 68                   LDA   CURLIN                     GET NUMBER OF CURRENT LINE NUMBER
1246   C8EB BD BD CC                JSR   LBDCC                      CONVERT ACCD TO DECIMAL & PRINT IT ON SCREEN
1247   C8EE 86 5D                   LDA   #'>'                       > RIGHT DELIMITER OF TRON
1248   C8F0 BD A2 82                JSR   LA282                      SEND A CHARACTER TO CONSOLE OUT
```

```
1249   C8F3 9D 9F          LC8F3   JSR   GETNCH                    GET NEXT CHARACTER FROM BASIC
1250   C8F5 1F A9                  TFR   CC,B                      SAVE STATUS REGISTER IN ACCB
1251   C8F7 81 98                  CMPA  #$98                      CSAVE TOKEN?
1252   C8F9 26 03                  BNE   LC8FE                     NO
1253   C8FB 7E 83 16               JMP   L8316                     GO CHECK FOR CSAVEM
1254   C8FE 81 97          LC8FE   CMPA  #$97                      CLOAD TOKEN?
1255   C900 26 03                  BNE   LC905                     NO
1256   C902 7E 83 11               JMP   L8311                     JUMP TO EXBAS' CLOAD ROUTINE
1257   C905 1F 9A          LC905   TFR   B,CC                      RESTORE STATUS REGISTER
1258   C907 BD AD C6               JSR   LADC6                     LOOP THROUGH BASIC'S MAIN INTERPRETATION LOOP
1259   C90A 20 A6                  BRA   LC8B2
1260
1261                       * EOF RAM HOOK
1262   C90C 32 62          DVEC14  LEAS  $02,S                     PURGE RETURN ADDRESS OFF OF THE STACK
1263   C90E 96 6F                  LDA   DEVNUM                    * GET DEVICE NUMBER AND SAVE
1264   C910 34 02                  PSHS  A                         * IT ON THE STACK
1265   C912 BD A5 AE               JSR   LA5AE                     STRIP DEVICE NUMBER OFF OF INPUT LINE
1266   C915 BD A3 ED               JSR   LA3ED                     VERIFY THAT THE FILE TYPE WAS 'INPUT'
1267   C918 0D 6F                  TST   DEVNUM                    * CHECK DEVICE NUMBER AND
1268   C91A 10 2F DC BC            LBLE  LA5DA                     * BRANCH BACK TO  BASIC'S EOF IF NOT DISK FILE
1269   C91E BD C7 44               JSR   LC744                     POINT X TO FCB
1270   C921 E6 00                  LDB   FCBTYP,X                  GET FILE TYPE
1271   C923 C1 40                  CMPB  #RANFIL                   RANDOM FILE?
1272   C925 10 27 DC ED            LBEQ  LA616                     'FM' BAD FILE MODE ERROR IF RANDOM
1273   C929 5F                     CLRB                            FILE NOT EMPTY FLAG - SET TO NOT EMPTY
1274   C92A A6 88 10               LDA   FCBCFL,X                  *CHECK THE CACHE FLAG - BRANCH IF
1275   C92D 26 03                  BNE   LC932                     *THERE IS A CHARACTER WHICH HAS BEEN CACHED
1276   C92F E6 88 17               LDB   FCBDFL,X                  GET SEQUENTIAL INPUT FILE STATUS
1277   C932 7E A5 E4       LC932   JMP   LA5E4                     LINK BACK TO BASIC'S EOF STATEMENT
1278
1279                       * GET FILENAME/EXTENSION: DRIVE NUMBER FROM BASIC
1280   C935 8E C2 A9       LC935   LDX   #DEFEXT                   POINT TO '    ' BLANK (DEFAULT) EXTENSION
1281   C938 6F E2                  CLR   ,-S                       CLEAR A BYTE ON STACK FOR USE AS A DRIVES FLAG
1282   C93A B6 09 5A               LDA   DEFDRV                    * GET DEFAULT DISK NUMBER
1283   C93D 97 EB                  STA   DCDRV                     * STORE IN DSKCON PARAMETER
1284   C93F CE 09 4C               LDU   #DNAMBF                   DISK FILENAME BUFFER
1285   C942 CC 20 08               LDD   #$2008                    STORE 8 BLANKS IN RAM (DEFAULT FILE NAME)
1286   C945 A7 C0          LC945   STA   ,U+                       STORE A BLANK IN FILE NAME
1287   C947 5A                     DECB                            DECREMENT COUNTER
1288   C948 26 FB                  BNE   LC945                     BRANCH IF NOT DONE
1289   C94A C6 03                  LDB   #$03                      3 BYTES IN EXTENSION
1290   C94C BD A5 9A               JSR   LA59A                     MOVE B BYTES FROM (X) TO (U)
1291   C94F BD 87 48               JSR   L8748                     EVALUATE A STRING EXPRESSION
1292   C952 33 84                  LEAU  ,X                        POINT U TO START OF STRING
1293   C954 C1 02                  CMPB  #$02                      * CHECK LENGTH OF STRING AND
1294   C956 25 12                  BLO   LC96A                     * BRANCH IF < 2
1295   C958 A6 41                  LDA   $01,U                     = GET 2ND CHARACTER IN STRING AND
1296   C95A 81 3A                  CMPA  #':'                      = CHECK FOR COLON
1297   C95C 26 0C                  BNE   LC96A                     BRANCH IF NO DRIVE NUMBER
1298   C95E A6 C4                  LDA   ,U                        * GET 1ST CHARACTER
1299   C960 81 30                  CMPA  #'0'                      * IN STRING AND
1300   C962 25 06                  BLO   LC96A                     * CHECK TO SEE
1301   C964 81 33                  CMPA  #'3'                      * IF IT IS IN
1302   C966 22 02                  BHI   LC96A                     * THE RANGE 0-3
1303   C968 8D 33                  BSR   LC99D                     GET DRIVE NUMBER
1304   C96A 8E 09 4C       LC96A   LDX   #DNAMBF                   POINT X TO FILE NAME BUFFER
1305   C96D 5C                     INCB                            COMPENSATE FOR DECB BELOW
1306   C96E 5A             LC96E   DECB                            DECREMENT STRING LENGTH
1307   C96F 26 0C                  BNE   LC97D                     BRANCH IF MORE CHARACTERS IN STRING
1308   C971 32 61                  LEAS  $01,S                     CLEAN UP STACK - REMOVE DRIVE FLAG
1309   C973 8C 09 4C       LC973   CMPX  #DNAMBF                   POINTER STILL AT START OF BUFFER?
1310   C976 26 67                  BNE   LC9DF                     RETURN IF NOT
1311   C978 C6 3E          LC978   LDB   #2*31                     'BAD FILENAME' ERROR IF NULL FILENAME
1312   C97A 7E AC 46               JMP   LAC46                     ERROR HANDLER
1313   C97D A6 C0          LC97D   LDA   ,U+                       GET A CHARACTER FROM STRING
1314   C97F 81 2E                  CMPA  #'.'                      LOOK FOR PERIOD?
1315   C981 27 2D                  BEQ   LC9B0                     YES
1316   C983 81 2F                  CMPA  #'/'                      SLASH?
1317   C985 27 29                  BEQ   LC9B0                     YES
1318   C987 81 3A                  CMPA  #':'                      COLON?
1319   C989 27 09                  BEQ   LC994                     YES
1320   C98B 8C 09 54               CMPX  #DEXTBF                   COMPARE POINTER TO END OF FILENAME BUFFER
1321   C98E 27 E8                  BEQ   LC978                     'BAD FILENAME' ERROR - FILENAME TOO LONG
1322   C990 8D 3E                  BSR   LC9D0                     PUT A CHARACTER IN FILENAME
1323   C992 20 DA                  BRA   LC96E                     GET ANOTHER CHARACTER FROM STRING
1324   C994 8D DD          LC994   BSR   LC973                     'BAD FILENAME' ERROR IF NO FILENAME YET
1325   C996 8D 05                  BSR   LC99D                     GET DRIVE NUMBER
1326   C998 5D                     TSTB                            * CHECK LENGTH OF STRING
1327   C999 26 DD                  BNE   LC978                     * ''BAD FILENAME' ERROR IF MORE CHARACTERS LEFT
1328   C99B 35 82          LC99B   PULS  A,PC                      REMOVE DRIVES FLAG FROM STACK AND RETURN
1329
1330                       * GRAB DRIVE NUMBER
1331   C99D 63 62          LC99D   COM   $02,S                     TOGGLE DRIVE FLAG
1332   C99F 27 D7                  BEQ   LC978                     'BAD FILENAME' ERROR IF DRIVE NUMBER DEFINED TWICE
1333   C9A1 A6 C1                  LDA   ,U++                      ASCII VALUE OF DRIVE NUMBER TO ACCA
1334   C9A3 C0 02                  SUBB  #$02                      DECREMENT STRING LENGTH BY 2 FOR DRIVE (:X)
1335   C9A5 80 30                  SUBA  #'0'                      SUBTRACT ASCII BIAS
1336   C9A7 25 CF                  BLO   LC978                     DRIVE NUMBER TOO LOW - 'BAD FILENAME' ERROR
1337   C9A9 81 03                  CMPA  #$03                      MAX OF 4 DRIVES
1338   C9AB 22 CB                  BHI   LC978                     DRIVE NUMBER TOO HIGH - 'BAD FILENAME' ERROR
1339   C9AD 97 EB                  STA   DCDRV                     STORE IN DSKCON DRIVE NUMBER
1340   C9AF 39                     RTS
1341
1342                       * GRAB EXTENSION
1343   C9B0 8D C1          LC9B0   BSR   LC973                     'BAD FILENAME' ERROR IF NO FILENAME YET
1344   C9B2 8E 09 57               LDX   #DFLTYP                   POINT X TO END OF EXTENSION BUFFER
```

```
1345   C9B5 86 20                  LDA  #SPACE                     BLANK
1346   C9B7 A7 82           LC9B7  STA  ,-X                        *
1347   C9B9 8C 09 54               CMPX #DEXTBF                    * FILL EXTENSION WITH
1348   C9BC 26 F9                  BNE  LC9B7                      * BLANKS (DEFAULT)
1349   C9BE 5A              LC9BE  DECB                            DECREMENT STRING COUNTER
1350   C9BF 27 DA                  BEQ  LC99B                      RETURN IF ZERO
1351   C9C1 A6 C0                  LDA  ,U+                        GET A CHARACTER FROM STRING
1352   C9C3 81 3A                  CMPA #':'                       *CHECK FOR DRIVE SEPARATOR
1353   C9C5 27 CD                  BEQ  LC994                      *
1354   C9C7 8C 09 57               CMPX #DFLTYP                    =CHECK FOR END OF ESTENSION RAM BUFFER &
1355   C9CA 27 AC                  BEQ  LC978                      = 'BAD FILENAME' ERROR IF EXTENSION TOO LONG
1356   C9CC 8D 02                  BSR  LC9D0                      PUT A CHARACTER IN EXTENSION BUFFER
1357   C9CE 20 EE                  BRA  LC9BE                      GET ANOTHER EXTENSION CHARACTER
1358
1359                        * INSERT CHARACTER INTO FILENAME OR EXTENSION
1360   C9D0 A7 80           LC9D0  STA  ,X+                        STORE CHARACTER IN FILENAME BUFFER
1361   C9D2 27 A4                  BEQ  LC978                      'BAD FILENAME' ERROR; ZEROES ARE ILLEGAL
1362   C9D4 81 2E                  CMPA #'.'                       PERIOD?
1363   C9D6 27 A0                  BEQ  LC978                      'BAD FILENAME' ERROR IF PERIOD
1364   C9D8 81 2F                  CMPA #'/'                       SLASH?
1365   C9DA 27 9C                  BEQ  LC978                      'BAD FILENAME' ERROR IF SLASH
1366   C9DC 4C                     INCA                            CHECK FOR $FF
1367   C9DD 27 99                  BEQ  LC978                      'BAD FILENAME' ERROR IF $FF
1368   C9DF 39              LC9DF  RTS
1369
1370                        * SAVE COMMAND
1371   C9E0 81 4D           SAVE   CMPA #'M'                       *
1372   C9E2 10 27 05 82            LBEQ LCF68                      *BRANCH IF SAVEM
1373   C9E6 8D 4B                  BSR  LCA33                      GO GET FILENAME, ETC. FROM BASIC
1374   C9E8 9E 8A                  LDX  ZERO                       ZERO OUT X REG
1375   C9EA BF 09 57               STX  DFLTYP                     SET FILE TYPE AND ASCII FLAG TO ZERO
1376   C9ED 9D A5                  JSR  GETCCH                     GET CURRENT INPUT CHARACTER FROM BASIC
1377   C9EF 27 21                  BEQ  LCA12                      BRANCH IF END OF LINE
1378   C9F1 BD B2 6D               JSR  SYNCOMMA                   SYNTAX CHECK FOR COMMA
1379   C9F4 C6 41                  LDB  #'A'                       *ASCII FILE?
1380   C9F6 BD B2 6F               JSR  LB26F                      *SYNTAX CHECK ON CONTENTS OF ACCB
1381   C9F9 26 E4                  BNE  LC9DF                      RETURN IF NO MORE CHARACTERS ON LINE
1382   C9FB 73 09 58               COM  DASCFL                     SET CRUNCHED/ASCII FLAG TO ASCII
1383   C9FE 8D 04                  BSR  LCA04                      OPEN A SEQUENTIAL FILE FOR OUTPUT
1384   CA00 4F                     CLRA                            SET ZERO FLAG - CAUSE ENTIRE FILE TO BE LISTED
1385   CA01 7E B7 64               JMP  LIST                       'LIST' THE FILE TO CONSOLE OUT
1386
1387                        * OPEN A SEQUENTIAL FILE FOR INPUT/OUTPUT - USE THE SYSTEM
1388                        * FCB LOCATED AT THE TOP OF FCBS
1389   CA04 86 4F           LCA04  LDA  #'O'                       OUTPUT FILE TYPE
1390   CA06 8C              LCA06  CMPX #$8649                     SKIP TWO BYTES
1391   CA07 86 49           LCA07  LDA  #'I'                       INPUT FILE TYPE
1392   CA09 F6 09 5B               LDB  FCBACT                     GET NUMBER OF RESERVED FILES CURRENTLY RESERVED
1393   CA0C 5C                     INCB                            ADD ONE - USE ONE ABOVE HIGHEST RESERVED FCB
1394   CA0D D7 6F                  STB  DEVNUM                     SAVE IT IN DEVICE NUMBER
1395   CA0F 7E C4 8D               JMP  LC48D                      OPEN A FILE & INITIALIZE FCB
1396                        * SAVE A CRUNCHED FILE - A PREAMBLE OF THREE BYTES WILL PRECEED CRUNCHED
1397                        * FILES: BYTE 1 = $FF, 2,3 = LENGTH OF BASIC PROGRAM
1398   CA12 8D F0           LCA12  BSR  LCA04                      OPEN A SEQUENTIAL FILE FOR OUTPUT
1399   CA14 86 FF                  LDA  #$FF                       BASIC FILE FLAG
1400   CA16 BD CC 24               JSR  LCC24                      CONSOLE OUT
1401   CA19 DC 1B                  LDD  VARTAB                     LOAD ACCD WITH START OF VARIABLES
1402   CA1B 93 19                  SUBD TXTTAB                     SUBTRACT START OF BASIC
1403   CA1D BD CC 24               JSR  LCC24                      CONSOLE OUT FILE LENGTH MS BYTE
1404   CA20 1F 98                  TFR  B,A                        PULL LS BYTE INTO ACCA
1405   CA22 BD CC 24               JSR  LCC24                      CONSOLE OUT FILE LENGTH LS BYTE
1406   CA25 9E 19                  LDX  TXTTAB                     POINT X TO START OF BASIC
1407   CA27 A6 80           LCA27  LDA  ,X+                        GET BYTE FROM BASIC
1408   CA29 BD CC 24               JSR  LCC24                      SEND TO CONSOLE OUT
1409   CA2C 9C 1B                  CMPX VARTAB                     COMPARE TO END OF BASIC
1410   CA2E 26 F7                  BNE  LCA27                      KEEP GOING IF NOT AT END
1411   CA30 7E A4 2D               JMP  LA42D                      CLOSE FILE
1412   CA33 8E C2 A6               LDX  #BASEXT                    POINT TO 'BAS' EXTENSION (DEFAULT)
1413   CA36 7E C9 38               JMP  LC938                      GET FILENAME.EXT FROM BASIC
1414
1415                        * MERGE COMMAND
1416   CA39 4F              MERGE  CLRA                            RUN FLAG (0 = DON'T RUN)
1417   CA3A C6 FF                  LDB  #$FF                       MERGE FLAG ($FF = MERGE)
1418   CA3C 20 12                  BRA  LCA50                      GO LOAD THE FILE
1419
1420                        * RUN RAM VECTOR
1421   CA3E 81 22           DVEC18 CMPA #'"'                       CHECK FOR FILENAME DELIMITER (DOUBLE QUOTE)
1422   CA40 10 26 B8 58            LBNE XVEC18                     NONE - JUMP TO EXBAS RUN RAM HOOK
1423   CA44 86 02                  LDA  #$02                       RUN FLAG - DON'T CLOSE ALL FILES BEFORE RUN
1424   CA46 20 07                  BRA  LCA4F                      LOAD THE FILE
1425
1426                        * LOAD COMMAND
1427   CA48 81 4D           LOAD   CMPA #'M'                       *
1428   CA4A 10 27 05 73            LBEQ LCFC1                      *BRANCH IF LOADM
1429   CA4E 4F                     CLRA                            RUN FLAG = ZERO (DON'T RUN)
1430   CA4F 5F              LCA4F  CLRB                            CLEAR MERGE FLAG
1431   CA50 B7 09 59        LCA50  STA  DRUNFL                     RUN FLAG (0 = DON'T RUN, 2 = RUN)
1432   CA53 F7 09 5E               STB  DMRGFL                     MERGE FLAG (0 = NO MERGE, $FF = MERGE)
1433   CA56 8D DB                  BSR  LCA33                      GO GET FILENAME, ETC. FROM BASIC
1434   CA58 9D A5                  JSR  GETCCH                     GET CURRENT INPUT CHAR
1435   CA5A 27 10                  BEQ  LCA6C                      BRANCH IF END OF LINE
1436   CA5C BD B2 6D               JSR  SYNCOMMA                   SYNTAX CHECK FOR COMMA
1437   CA5F C6 52                  LDB  #'R'                       *
1438   CA61 BD B2 6F               JSR  LB26F                      *IS NEXT CHAR 'R'? RUN AFTER LOAD
1439   CA64 BD A5 C7               JSR  LA5C7                      SYNTAX ERROR IF ANY MORE CHARS ON LINE
1440   CA67 86 03                  LDA  #$03                       *SET FLAGS TO RUN AND CLOSE ALL FILES
```

```
1441   CA69 B7 09 59              STA   DRUNFL                    *BEFORE THE FILE IS RUN
1442   CA6C 8D 99          LCA6C  BSR   LCA07                     GRAB FCB FOR INPUT FILE
1443   CA6E B6 09 58              LDA   DASCFL                    *CHECK ASCII FLAG AND BRANCH
1444   CA71 27 0B                 BEQ   LCA7E                     *IF CRUNCHED BASIC FILE
1445   CA73 7D 09 5E              TST   DMRGFL                    IS THIS A MERGE?
1446   CA76 26 03                 BNE   LCA7B                     BRANCH IF MERGE
1447   CA78 BD AD 19              JSR   LAD19                     DO A 'NEW' - ERASE VARIABLES, RESET VARIABLES
1448   CA7B 7E AC 7C       LCA7B  JMP   LAC7C                     GO TO BASIC'S MAIN LOOP, IT WILL LOAD PROGRAM
1449
1450                       * LOAD IN A CRUNCHED BASIC FILE
1451   CA7E B6 09 57       LCA7E  LDA   DFLTYP                    *CHECK FILE TYPE (MUST BE BASIC:0) & CHECK
1452   CA81 BA 09 5E              ORA   DMRGFL                    *MERGE FLAG (MUST BE NO MERGE: 0)
1453   CA84 10 26 DB 8E           LBNE  LA616                     'BAD FILE MODE' ERROR IF MERGE OR NON-BASIC
1454   CA88 BD AD 19              JSR   LAD19                     DO A 'NEW' - RESET POINTERS, ERASE VARIABLES
1455   CA8B 73 09 5D              COM   DLODFL                    * SET THE LOAD FLAG TO $FF - THIS WILL CAUSE A NEW TO
1456                       *                                     * OCCUR IF AN ERROR OCCURS WHILE THE PROGRAM IS BEING LOADED
1457   CA8E BD CD BC              JSR   LCDBC                     GET CHAR FROM BUFFER - SHOULD BE $FF
1458   CA91 BD CD BC              JSR   LCDBC                     GET ANOTHER - MS BYTE OF LENGTH
1459   CA94 34 02                 PSHS  A                         SAVE MS BYTE ON STACK
1460   CA96 BD CD BC              JSR   LCDBC                     LS BYTE OF LENGTH OF PROGRAM
1461   CA99 1F 89                 TFR   A,B                       PUT LS BYTE INTO ACCB
1462   CA9B 35 02                 PULS  A                         NOW ACCD CONTAINS LENGTH OF PROGRAM
1463   CA9D D3 19                 ADDD  TXTTAB                    ADD BEGINNING OF BASIC
1464   CA9F BD AC 37              JSR   LAC37                     SEE OF ENOUGH ROOM IN RAM FOR THIS FILE
1465   CAA2 9E 19                 LDX   TXTTAB                    GET START OF BASIC
1466   CAA4 BD C5 C4       LCAA4  JSR   LC5C4                     READ A CHAR FROM CONSOLE IN
1467   CAA7 D6 70                 LDB   CINBFL                    BUFFER EMPTY?
1468   CAA9 26 04                 BNE   LCAAF                     BRANCH IF SO
1469   CAAB A7 80                 STA   ,X+                       STORE CHAR
1470   CAAD 20 F5                 BRA   LCAA4                     GET ANOTHER CHARACTER
1471
1472   CAAF 7F 09 5D       LCAAF  CLR   DLODFL                    CLEAR LOAD FLAG - LOAD WAS ERROR FREE
1473   CAB2 9F 1B                 STX   VARTAB                    SAVE NEW START OF VARIABLES
1474                       * MAKE SURE LAST THREE BYTES LOADED WERE ZERO
1475   CAB4 C6 03                 LDB   #$03                      CHECK THREE BYTES
1476   CAB6 A6 82          LCAB6  LDA   ,-X                       CHECK A BYTE
1477   CAB8 26 03                 BNE   LCABD                     BRANCH IF NON-ZERO
1478   CABA 5A                    DECB                            DECREMENT COUNTER
1479   CABB 26 F9                 BNE   LCAB6                     KEEP CHECKING IF NOT DONE
1480   CABD 9E 1B          LCABD  LDX   VARTAB                    GET START OF VARIABLES
1481   CABF 9F 1B          LCABF  STX   VARTAB                    SAVE START OF VARIABLES
1482   CAC1 6F 80                 CLR   ,X+                       CLEAR A BYTE
1483   CAC3 5A                    DECB                            DECREMRNT COUNTER
1484   CAC4 2A F9                 BPL   LCABF                     KEEP CLEARING BYTES IF NOT DONE
1485   CAC6 BD A4 2D       LCAC6  JSR   LA42D                     CLOSE SELECTED FILE
1486   CAC9 BD AD 21              JSR   LAD21                     DO PART OF NEW - ERASE VARIABLES, RESET INPUT PTR
1487   CACC BD 82 9C              JSR   XVEC18                    INITIALIZE EXBAS GRAPHICS VARIABLES
1488   CACF BD AC EF              JSR   LACEF                     RELOCATE ALL THE BASIC NEXT LINE POINTERS
1489   CAD2 77 09 59              ASR   DRUNFL                    CHECK LSB OF RUN FLAG
1490   CAD5 25 03                 BLO   LCADA                     BRANCH IF DON'T CLOSE ALL FILES
1491   CAD7 BD A4 26              JSR   LA426                     CLOSE ALL FILES
1492   CADA 77 09 59       LCADA  ASR   DRUNFL                    TEST BIT 1 OF RUN FLAG
1493   CADD 10 25 E2 BD           LBCS  LAD9E                     BRANCH TO COMM INTERPRETATION LOOP IF BIT 1 SET
1494   CAE1 7E AC 73              JMP   LAC73                     RETURN TO DIRECT MODE
1495
1496   CAE4 0D 6F          DVEC13 TST   DEVNUM                    * CHECK DEVICE NUMBER AND
1497   CAE6 2E DE                 BGT   LCAC6                     * TRY TO RUN FILE IF IT IS A DISK FILE
1498   CAE8 39                    RTS
1499
1500                       * CLOSE ALL FILE BUFFERS RAM VECTOR
1501   CAE9 F6 09 5B       DVEC7  LDB   FCBACT                    GET THE NUMBER OF RESERVED FILE BUFFERS
1502   CAEC 5C                    INCB                            ADD ONE
1503   CAED 34 04          LCAED  PSHS  B                         SAVE IT
1504   CAEF D7 6F                 STB   DEVNUM                    STORE IT IN DEVICE NUMBER
1505   CAF1 8D 0E                 BSR   LCB01                     CLOSE FILE
1506   CAF3 35 04                 PULS  B                         GET BACK NUMBER OF FILE BUFFERS
1507   CAF5 5A                    DECB                            DECREMENT FILE BUFFER COUNTER
1508   CAF6 26 F5                 BNE   LCAED                     BRANCH IF ALL FILES NOT CLOSED
1509   CAF8 39             LCAF8  RTS
1510
1511                       * CLOSE FILE RAM HOOK
1512   CAF9 0D 6F          DVEC8  TST   DEVNUM                    * CHECK DEVICE NUMBER AND RETURN
1513   CAFB 10 2F B7 87           LBLE  XVEC8                     * IF NOT A DISK FILE
1514   CAFF 32 62                 LEAS  $02,S                     PURGE RETURN ADDRESS OFF OF THE STACK
1515   CB01 BD C7 44       LCB01  JSR   LC744                     POINT X TO CORRECT FCB
1516   CB04 0F 6F                 CLR   DEVNUM                    SET DEVICE NUMBER TO SCREEN
1517   CB06 9F F1          LCB06  STX   FCBTMP                    SAVE FILE BUFFER POINTER
1518   CB08 A6 00                 LDA   FCBTYP,X                  GET THE TYPE OF THIS FILE
1519   CB0A 27 EC                 BEQ   LCAF8                     RETURN IF FILE NOT OPEN
1520   CB0C 34 02                 PSHS  A                         SAVE FILE TYPE
1521   CB0E 6F 00                 CLR   FCBTYP,X                  CLOSE THE FILE - ZERO OUT THE FILE TYPE
1522   CB10 E6 01                 LDB   FCBDRV,X                  * GET DRIVE NUMBER AND
1523   CB12 D7 EB                 STB   DCDRV                     * SAVE IT IN DSKCON VARIABLE
1524   CB14 81 20                 CMPA  #OUTFIL                   = CHECK FOR OUTPUT TYPE AND
1525   CB16 26 19                 BNE   LCB31                     = BRANCH IF NOT OUTPUT TYPE FILE
1526
1527                       * CLOSE A SEQUENTIAL OUTPUT FILE
1528   CB18 E6 88 18              LDB   FCBLFT,X                  GET THE NUMBER OF CHARACTERS IN BUFFER
1529   CB1B 86 80                 LDA   #$80                      * SET THE PRE-SAVED BIT TO INDICATE THAT THE DATA
1530                       *                                     * HAS ALREADY BEEN SAVED ON DISK
1531   CB1D AA 05                 ORA   FCBCPT,X                  'OR' IN THE FULL SECTOR FLAG
1532   CB1F ED 88 13              STD   FCBLST,X                  SAVE THE NUMBER OF BYTES USED IN THE LAST SECTOR
1533   CB22 6C 04                 INC   FCBSEC,X                  INCREMENT THE SECTOR NUMBER
1534   CB24 E6 03                 LDB   FCBCGR,X                  GET THE CURRENT GRANULE NUMBER
1535   CB26 BD C7 55              JSR   LC755                     POINT X TO FILE ALLOCATION TABLE
1536   CB29 A7 01                 STA   FAT1,X                    SET FAT DATA NOT VALID FLAG (ACCA <> 0)
```

```
1537   CB2B 3A                  ABX                         ADD GRANULE OFFSET TO FAT POINTER
1538   CB2C 6C Ø6               INC   FATCON,X              * INCREMENT GRANULE DATA (ADD ONE SECTOR TO LAST
1539                           *                           * GRANULE) SKIP PAST THE SIX FAT CONTROL BYTES
1540   CB2E 7E CB C3   LCB2E    JMP   LCBC3                 UPDATE FAT AND DIRECTORY
1541   CB31 81 4Ø      LCB31    CMPA  #RANFIL               RANDOM FILE?
1542   CB33 26 F9               BNE   LCB2E                 NO - UPDATE FAT AND DIRECTORY IF SEQUENTIAL INPUT FILE
1543
1544                           * CLOSE A RANDOM FILE
1545   CB35 EC Ø9               LDD   FCBRLN,X              GET RECORD LENGTH
1546   CB37 AE ØB               LDX   FCBBUF,X              POINT X TO RANDOM FILE BUFFER
1547   CB39 31 8B               LEAY  D,X                   POINT Y TO END OF RANDOM FILE BUFFER
1548   CB3B 34 36               PSHS  Y,X,B,A               SAVE POINTERS ON STACK
1549   CB3D 31 E4               LEAY  ,S                    POINT Y CURRENT STACK POINTER
1550   CB3F DE 1B               LDU   VARTAB                GET START OF VARIABLES
1551   CB41 11 93 1D   LCB41    CMPU  ARYTAB                COMPARE TO START OF ARRAYS
1552   CB44 27 ØE               BEQ   LCB54                 BRANCH IF ALL VARIABLES CHECKED
1553   CB46 A6 41               LDA   $Ø1,U                 GET 2ND BYTE OF VARIABLE NAME
1554   CB48 33 42               LEAU  $Ø2,U                 MOVE POINTER TO START OF DESCRIPTOR
1555   CB4A 2A Ø2               BPL   LCB4E                 BRANCH IF VARIABLE - NUMERIC
1556   CB4C 8D 28               BSR   LCB76                 ADJUST STRING VARIABLE IF IN RANDOM FILE BUFFER
1557   CB4E 33 45      LCB4E    LEAU  $Ø5,U                 MOVE POINTER TO NEXT VARIABLE
1558   CB5Ø 2Ø EF               BRA   LCB41                 PROCESS ANOTHER VARIABLE
1559   CB52 35 4Ø      LCB52    PULS  U                     GET ADDRESS OF NEXT ARRAY TO U
1560   CB54 11 93 1F   LCB54    CMPU  ARYEND                COMPARE TO END OF ARRAYS
1561   CB57 27 3A               BEQ   LCB93                 BRANCH IF END OF ARRAYS
1562   CB59 1F 3Ø               TFR   U,D                   * SAVE ARRAY START IN ACCD, ADD OFFSET
1563   CB5B E3 42               ADDD  $Ø2,U                 * TO NEXT ARRAY AND SAVE ADDRESS OF
1564   CB5D 34 Ø6               PSHS  B,A                   * NEXT ARRAY ON THE STACK
1565   CB5F A6 41               LDA   $Ø1,U                 GET 2ND LETTER OF VARIABLE NAME
1566   CB61 2A EF               BPL   LCB52                 BRANCH IF NUMERIC
1567   CB63 E6 44               LDB   $Ø4,U                 GET THE NUMBER OF DIMENSIONS
1568   CB65 58                  ASLB                        X2:2 BYTES PER DIMENSION
1569   CB66 CB Ø5               ADDB  #$Ø5                  5 BYTES CONSTANT PER ARRAY DESCRIPTOR
1570   CB68 4F                  CLRA                        CLEAR MSB OF OFFSET - (ONLY 125 DIMENSIONS ALLOWED)
1571   CB69 33 CB               LEAU  D,U                   POINT U TO START OF THIS ARRAY'S VARIABLES
1572   CB6B 11 A3 E4   LCB6B    CMPU  ,S                    AT END OF THIS ARRAY?
1573   CB6E 27 E2               BEQ   LCB52                 YES
1574   CB7Ø 8D Ø4               BSR   LCB76                 ADJUST STRING VARIABLE IF IN RANDOM FILE BUFFER
1575   CB72 33 45               LEAU  $Ø5,U                 MOVE POINTER TO NEXT DESCRIPTOR
1576   CB74 2Ø F5               BRA   LCB6B                 CHECK NEXT VARIABLE
1577                           *
1578                           * CHECK TO SEE IF A STRING IS LOCATED IN THE RANDOM FILE BUFFER AREA. IF IT IS
1579                           * THE RANDOM FILE BUFFER IN QUESTION, IT WILL BE DELETED. IF IT IS HIGHER IN THE RANDOM
1580                           * FILE BUFFER SPACE THAN THE BUFFER IN QUESTION, THE LENGTH OF THE CURRENT
1581                           * BUFFER WILL BE SUBTRACTED FROM THE ADDRESS OF THE STRING BECAUSE THE CURRENT
1582                           * BUFFER IS BEING DELETED (CLOSED).
1583   CB76 AE 42      LCB76    LDX   $Ø2,U                 POINT X TO START OF STRING
1584   CB78 BC Ø9 48            CMPX  RNBFAD                COMPARE TO START OF FREE RANDOM FILE BUFFER AREA
1585   CB7B 24 ØE               BHS   LCB8B                 RETURN IF > START OF FREE RANDOM FILE BUFFER AREA
1586   CB7D AC 22               CMPX  $Ø2,Y                 COMPARE TO START OF THIS FILE'S RANDOM BUFFER
1587   CB7F 25 ØA               BLO   LCB8B                 RETURN IF < START OF THIS FILE'S RANDOM BUFFER
1588   CB81 AC 24               CMPX  $Ø4,Y                 COMPARE TO END OF THIS FILE'S RANDOM BUFFER
1589   CB83 25 Ø7               BLO   LCB8C                 RETURN IF < END OF THIS FILE'S RANDOM BUFFER
1590   CB85 1F 1Ø               TFR   X,D                   SAVE POINTER IN ACCD
1591   CB87 A3 A4               SUBD  ,Y                    SUBTRACT RECORD LENGTH FROM START OF STRING ADDRESS
1592   CB89 ED 42               STD   $Ø2,U                 SAVE NEW START OF STRING ADDRESS
1593   CB8B 39         LCB8B    RTS
1594   CB8C 6F C4      LCB8C    CLR   ,U                    CLEAR THE LENGTH OF THE STRING
1595   CB8E 6F 42               CLR   $Ø2,U                 * CLEAR THE ADDRESS
1596   CB9Ø 6F 43               CLR   $Ø3,U                 * OF THE STRING
1597   CB92 39                  RTS
1598                           * REMOVE RESERVED SPACE IN RANDOM FILE BUFFER FOR A 'CLOSED' RANDOM FILE
1599                           * ADJUST THE START OF RANDOM FILE BUFFER POINTER IN ALL RANDOM FCBS
1600   CB93 F6 Ø9 5B   LCB93    LDB   FCBACT                GET THE NUMBER OF ACTIVE FILES
1601   CB96 5C                  INCB                        ADD ONE
1602   CB97 34 Ø4      LCB97    PSHS  B                     SAVE FILES COUNT ON THE STACK
1603   CB99 BD C7 49            JSR   LC749                 POINT X TO FCB
1604   CB9C A6 ØØ               LDA   FCBTYP,X              GET FILE TYPE
1605   CB9E 81 4Ø               CMPA  #RANFIL               IS IT A RANDOM FILE?
1606   CBAØ 26 ØB               BNE   LCBAD                 BRANCH IF NOT
1607   CBA2 EC ØB               LDD   FCBBUF,X              GET START OF THIS FILE'S RANDOM FILE BUFFER
1608   CBA4 1Ø A3 24            CMPD  $Ø4,Y                 * COMPARE TO END OF RANDOM FILE BUFFER AREA AND
1609   CBA7 25 Ø4               BLO   LCBAD                 * BRANCH IF < END OF RANDOM FILE BUFFER AREA
1610   CBA9 A3 A4               SUBD  ,Y                    = SUBTRACT RECORD LENGTH OF SELECTED FILE
1611   CBAB ED ØB               STD   FCBBUF,X              = SAVE NEW START OF RANDOM FILE BUFFER
1612   CBAD 35 Ø4      LCBAD    PULS  B                     GET THE FILES COUNTER
1613   CBAF 5A                  DECB                        DECREMENT FILES COUNTER
1614   CBBØ 26 E5               BNE   LCB97                 BRANCH IF ALL FILES NOT DONE
1615   CBB2 35 56               PULS  A,B,X,U               * U = END OF RANDOM FILE BUFFER, X = START OF RANDOM
1616                                                        * FILE BUFFER, ACCD = RECORD LENGTH
1617
1618                           ** THIS WOULD PROBABLY BE THE MOST CONVENIENT PLACE TO FIX THE BUG WHICH
1619                           ** CAUSES THE SYSTEM TO HANG IF AN ERROR IS ENCOUNTERED DURING 'COPY'
1620
1621                           *       CMPU  FCBADR                * IS THE END OF THIS FCB'S BUFFER ABOVE THE END
1622                           *                                   * OF THE START OF THE FCB AREA
1623                           *       BLO   LCBB4                 NO - FREE UP THE SPACE USED BY THIS FILE IN RANDOM BUFFER
1624                           *       LDX   #DFLBUF               YES - DOING A 'COPY'; RESET START OF RANDOM BUFFER
1625                           *       BRA   LCBCØ
1626                           * RANDOM FILE BUFFER AREA
1627                           * REMOVE RESERVED SPACE FOR CLOSED FILE FROM RANDOM FILE BUFFER SPACE
1628
1629   CBB4 11 B3 Ø9 48 LCBB4   CMPU  RNBFAD                AT THE BOTTOM OF FREE RANDOM BUFFER AREA?
1630   CBB8 27 Ø6               BEQ   LCBCØ                 BRANCH IF THERE
1631   CBBA A6 CØ               LDA   ,U+                   = GRAB A SOURCE BYTE AND
1632   CBBC A7 8Ø               STA   ,X+                   = MOVE IT TO DESTINATION
```

```
1633  CBBE 2Ø F4                BRA   LCBB4                     KEEP MOVING BYTES
1634  CBCØ BF Ø9 48     LCBCØ   STX   RNBFAD                    SAVE NEW START OF FREE RANDOM BUFFER AREA
1635  CBC3 BD C7 55     LCBC3   JSR   LC755                     POINT X TO PROPER FILE ALLOCATION TABLE
1636  CBC6 6A ØØ                DEC   FATØ,X                    REMOVE ONE ACTIVE FILE
1637  CBC8 6D Ø1                TST   FAT1,X                    NEW DATA IN FAT RAM IMAGE?
1638  CBCA 27 Ø3                BEQ   LCBCF                     NO
1639  CBCC BD C7 1E             JSR   LC71E                     WRITE OUT FILE ALLOCATION TABLE TO DISK
1640  CBCF 9E F1        LCBCF   LDX   FCBTMP                    GET FILE BUFFER POINTER
1641  CBD1 35 Ø2                PULS  A                         GET FILE TYPE
1642  CBD3 81 2Ø                CMPA  #OUTFIL                   IS IT A SEQUENTIAL OUTPUT FILE?
1643  CBD5 27 Ø8                BEQ   LCBDF                     YES
1644  CBD7 81 4Ø                CMPA  #RANFIL                   IS IT A RANDOM FILE?
1645  CBD9 26 BØ                BNE   LCB8B                     RETURN IF NOT A RANDOM FILE (SEQUENTIAL INPUT)
1646  CBDB A6 ØF                LDA   FCBFLG,X                  * TEST THE GET/PUT FLAG AND
1647  CBDD 27 ØA                BEQ   LCBE9                     * BRANCH IF 'GET'
1648
1649                            * WRITE CONTENTS OF FILE BUFFER TO DISK
1650  CBDF BD C7 63     LCBDF   JSR   LC763                     GET PROPER TRACK & SECTOR NUMBERS
1651  CBE2 33 88 19             LEAU  FCBCON,X                  POINT U TO START OF FCB DATA
1652  CBE5 DF EE                STU   DCBPT                     SET UP FILE BUFFER POINTER FOR DSKCON
1653  CBE7 8D 2C                BSR   LCC15                     GO WRITE A SECTOR
1654  CBE9 A6 88 13     LCBE9   LDA   FCBLST,X                  CHECK THE PRE-SAVED FLAG
1655  CBEC 2A 9D                BPL   LCB8B                     RETURN IF RECORD HAS ALREADY BEEN SAVED ON DISK
1656  CBEE E6 88 12             LDB   FCBDIR,X                  GET DIRECTORY NUMBER OF THIS FILE
1657  CBF1 C4 Ø7                ANDB  #$Ø7                      8 ENTRIES PER SECTOR
1658  CBF3 86 2Ø                LDA   #DIRLEN                   DIRLEN BYTES PER DIRECTORY ENTRY
1659  CBF5 3D                   MUL                             GET SECTOR OFFSET FOR THIS ENTRY
1660  CBF6 CE Ø6 ØØ             LDU   #DBUFØ                    * GET READ/WRITE BUFFER Ø AND
1661  CBF9 DF EE                STU   DCBPT                     * SAVE IT IN DSKCON REGISTER
1662  CBFB 31 CB                LEAY  D,U                       Y POINTS TO CORRECT DIRECTORY ENTRY
1663  CBFD E6 88 12             LDB   FCBDIR,X                  GET DIRECTORY ENTRY NUMBER
1664  CCØØ 54                   LSRB                            *
1665  CCØ1 54                   LSRB                            *
1666  CCØ2 54                   LSRB                            * DIVIDE BY 8; EIGHT DIRECTORY ENTRIES PER SECTOR
1667  CCØ3 CB Ø3                ADDB  #$Ø3                      ADD BIAS; FIRST 3 SECTORS NOT DIRECTORY
1668  CCØ5 D7 ED                STB   DSEC                      STORE SECTOR NUMBER
1669  CCØ7 CC 11 Ø2             LDD   #$11Ø2                    DIRECTORY TRACK - READ OP CODE
1670  CCØA 97 EC                STA   DCTRK                     STORE TRACK NUMBER
1671  CCØC 8D Ø9                BSR   LCC17                     GO READ DIRECTORY
1672  CCØE EC 88 13             LDD   FCBLST,X                  GET NUMBER OF BYTES IN THE LAST SECTOR
1673  CC11 84 7F                ANDA  #$7F                      MASK OFF THE PRE-SAVED FLAG
1674  CC13 ED 2E                STD   DIRLIST,Y                 SAVE NUMBER OF BYTES IN LAST SECTOR OF FILE IN DIRECTORY
1675  CC15 C6 Ø3        LCC15   LDB   #$Ø3                      WRITE OP CODE
1676  CC17 D7 EA        LCC17   STB   DCOPC                     SAVE DSKCON OP CODE VARIABLE
1677  CC19 7E D6 F2             JMP   LD6F2                     GO READ/WRITE SECTOR
1678
1679                            * CONSOLE OUT RAM HOOK
1680  CC1C ØD 6F        DVEC3   TST   DEVNUM                    CHECK DEVICE NUMBER
1681  CC1E 1Ø 2F B6 51          LBLE  XVEC3                     BRANCH TO EX BASIC IF NOT A DISK FILE
1682  CC22 32 62                LEAS  $Ø2,S                     POP RETURN OFF STACK
1683                            * SEND A CHARACTER IN ACCA TO A DISK FILE. A CARRIAGE RETURN WILL RESET THE
1684                            * PRINT POSITION AND CONTROL CODES WILL NOT INCREMENT THE PRINT POSITION.
1685  CC24 34 16        LCC24   PSHS  X,B,A                     SAVE REGISTERS
1686  CC26 8E Ø9 26             LDX   #FCBV1-2                  POINT X TO TABLE OF FILE NUMBER VECTORS
1687  CC29 D6 6F                LDB   DEVNUM                    GET CURRENT FILE NUMBER
1688  CC2B 58                   ASLB                            2 BYTES PER FCB ADDRESS
1689  CC2C AE 85                LDX   B,X                       POINT X TO PROPER FCB
1690  CC2E E6 84                LDB   FCBTYP,X                  GET FILE TYPE
1691  CC3Ø C1 1Ø                CMPB  #INPFIL                   IS IT AN INPUT FILE?
1692  CC32 27 36                BEQ   LCC6A                     RETURN IF SO
1693  CC34 81 ØD                CMPA  #CR                       CARRIAGE RETURN (ENTER)
1694  CC36 26 Ø2                BNE   LCC3A                     NO
1695  CC38 6F Ø6                CLR   FCBPOS,X                  CLEAR PRINT POSITION IF CARRIAGE RETURN
1696  CC3A 81 2Ø        LCC3A   CMPA  #SPACE                    *
1697  CC3C 25 Ø2                BLO   LCC4Ø                     *BRANCH IF CONTROL CHAR
1698  CC3E 6C Ø6                INC   FCBPOS,X                  INCREMENT PRINT POSITION
1699  CC4Ø C1 4Ø        LCC4Ø   CMPB  #RANFIL                   IS IT RANDOM FILE?
1700  CC42 26 1A                BNE   LCC5E                     BRANCH IF NOT RANDOM
1701                            * PUT A BYTE INTO A RANDOM FILE
1702  CC44 EC 88 17             LDD   FCBPUT,X                  GET 'PUT' BYTE COUNTER
1703  CC47 C3 ØØ Ø1             ADDD  #$ØØØ1                    ADD ONE
1704  CC4A 1Ø A3 Ø9             CMPD  FCBRLN,X                  COMPARE TO RECORD LENGTH
1705  CC4D 1Ø 22 Ø1 7A          LBHI  LCDCB                     'FR' ERROR IF 'PUT' BYTE COUNTER > RECORD LENGTH
1706  CC51 ED 88 17             STD   FCBPUT,X                  SAVE NEW 'PUT' BYTE COUNTER
1707  CC54 AE ØB                LDX   FCBBUF,X                  POINT TO RANDOM FILE BUFFER POINTER
1708  CC56 3Ø 8B                LEAX  D,X                       POINT TO ONE PAST END OF CURRENT RECORD DATA
1709  CC58 35 Ø2                PULS  A                         PULL DATA FROM STACK
1710  CC5A A7 1F                STA   -1,X                      STORE IN DATA BUFFER
1711  CC5C 35 94                PULS  B,X,PC                    RESTORE REGISTERS AND RETURN
1712
1713                            * WRITE A BYTE TO SEQUENTIAL OUTPUT FILE
1714  CC5E 6C 88 18     LCC5E   INC   FCBLFT,X                  INCREMENT CHARACTER COUNT
1715  CC61 E6 88 18             LDB   FCBLFT,X                  * GET CHARACTER COUNT AND BRANCH
1716  CC64 27 Ø6                BEQ   LCC6C                     * IF THE BUFFER IS FULL
1717  CC66 3A                   ABX                             ADD CHARACTER COUNT TO FCB ADDRESS
1718  CC67 A7 88 18             STA   FCBCON-1,X                STORE NEW CHARACTER (SKIP PAST 25 CONTROL BYTES AT FCB START)
1719  CC6A 35 96        LCC6A   PULS  A,B,X,PC
1720
1721                            * WRITE OUT A FULL BUFFER AND RESET BUFFER
1722  CC6C 34 6Ø        LCC6C   PSHS  U,Y                       SAVE REGISTERS
1723  CC6E A7 89 Ø1 18          STA   SECLEN+FCBCON-1,X         STORE LAST CHARACTER IN BUFFER
1724  CC72 E6 Ø1                LDB   FCBDRV,X                  * GET DRIVE NUMBER AND SAVE
1725  CC74 D7 EB                STB   DCDRV                     * IT IN DSKCON CONTROL TABLE
1726  CC76 6C Ø4                INC   FCBSEC,X                  INCREMENT SECTOR NUMBER
1727  CC78 BD CB DF             JSR   LCBDF                     WRITE THE FILE BUFFER TO DISK
1728  CC7B 31 84                LEAY  ,X                        SAVE FCB POINTER IN Y
```

```
1729   CC7D E6 Ø3                    LDB   FCBCGR,X                      GET GRANULE NUMBER
173Ø   CC7F BD C7 55                 JSR   LC755                         POINT X TO PROPER ALLOCATION TABLE
1731   CC82 3A                       ABX                               ADD THE GRANULE NUMBER TO FAT POINTER
1732   CC83 33 Ø6                    LEAU  FATCON,X                    POINT U TO THE CORRECT GRANULE IN FAT - SKIP PAST
1733                                                                    THE SIX FAT CONTROL BYTES
1734   CC85 A6 24                    LDA   FCBSEC,Y                    GET CURRENT SECTOR FOR THIS GRANULE
1735   CC87 81 Ø9                    CMPA  #$Ø9                        MAX SECTOR NUMBER (9 SECTORS/GRANULE)
1736   CC89 25 ØE                    BLO   LCC99                       BRANCH IF NOT AT END OF GRANULE
1737   CC8B 6A 24                    DEC   FCBSEC,Y                    *DECREMENT SECTOR NUMBER AND INCREMENT ERROR FLAG IN
1738   CC8D 6C 25                    INC   FCBCPT,Y                    *CASE ERROR FOUND WHILE LOOKING FOR NEXT GRANULE
1739                          *                                         THE ERROR FLAG IS USED TO INDICATE THAT ANOTHER SECTOR
174Ø                           *                                         MUST BE ADDED TO THE LENGTH OF FILE FOLLOWING ERROR PROCESSING.
1741   CC8F BD C7 BF                 JSR   LC7BF                       GET NEXT FREE GRANULE
1742   CC92 6F 24                    CLR   FCBSEC,Y                    *CLEAR SECTOR NUMBER AND
1743   CC94 6F 25                    CLR   FCBCPT,Y                    *ERROR FLAG - DISK WAS NOT FULL
1744   CC96 A7 23                    STA   FCBCGR,Y                    SAVE NEW GRANULE IN FCB
1745   CC98 8C 8A CØ                 CMPX  #$8ACØ                      SKIP TWO BYTES   NO DATA STORED IN NEW SECTOR YET
1746   CC99 8A CØ             LCC99  ORA   #$CØ                        FORCE GRANULE NUMBER TO BE FINAL GRANULE IN FILE
1747   CC9B A7 C4                    STA   ,U                          STORE IN MAP
1748   CC9D 3Ø A4                    LEAX  ,Y                          POINT X TO FCB
1749   CC9F BD C6 85                 JSR   LC685                       INCREMENT RECORD NUMBER
175Ø   CCA2 BD C5 A9                 JSR   LC5A9                       UPDATE FILE ALLOCATION TABLE
1751   CCA5 35 6Ø                    PULS  Y,U                         RESTORE REGISTERS
1752   CCA7 35 96                    PULS  A,B,X,PC                    RESTORE REGISTERS AND RETURN
1753
1754                          * DIR COMMAND
1755   CCA9 BD D2 4F          DIR    JSR   LD24F                       SCAN DRIVE NUMBER FROM INPUT LINE
1756   CCAC BD C7 9D                 JSR   LC79D                       GET FAT FOR THIS DRIVE
1757   CCAF BD B9 58                 JSR   LB958                       PRINT CARRIAGE RETURN TO CONSOLE OUT
1758   CCB2 CC 11 Ø2                 LDD   #$11Ø2                      * GET TRACK 17 AND
1759   CCB5 97 EC                    STA   DCTRK                       * READ OP CODE AND
176Ø   CCB7 D7 EA                    STB   DCOPC                       * SAVE IN DSKCON VARIABLES
1761   CCB9 C6 Ø3                    LDB   #$Ø3                        START WITH SECTOR 3 (FIRST DIRECTORY SECTOR)
1762
1763                          * READ A DIRECTORY SECTOR INTO THE I/O BUFFER
1764   CCBB D7 ED             LCCBB  STB   DSEC                        SAVE SECTOR NUMBER IN DSKCON VARIABLE
1765   CCBD 8E Ø6 ØØ                 LDX   #DBUFØ                      * USE I/O BUFFER Ø FOR DATA TRANSFER
1766   CCCØ 9F EE                    STX   DCBPT                       * SAVE IN DSKCON VARIABLE
1767   CCC2 BD D6 F2                 JSR   LD6F2                       READ A SECTOR
1768
1769                          * SEND DIRECTORY INFORMATION TO CONSOLE OUT
177Ø   CCC5 35 4Ø             LCCC5  PULS  U                           SAVE TOP OF STACK
1771   CCC7 BD A5 49                 JSR   LA549                       GO DO A BREAK CHECK
1772   CCCA 34 4Ø                    PSHS  U                           RESTORE STACK
1773   CCCC A6 84                    LDA   DIRNAM,X                    TEST FILE NAME FIRST BYTE
1774   CCCE 27 38                    BEQ   LCD08                       BRANCH IF KILLED
1775   CCDØ 43                       COMA                              FF = END OF DIRECTORY
1776   CCD1 27 44                    BEQ   LCD17                       RETURN IF END OF DIRECTORY
1777   CCD3 34 1Ø                    PSHS  X                           SAVE DIRECTORY POINTER ON STACK
1778   CCD5 C6 Ø8                    LDB   #$Ø8                        NUMBER CHARACTERS TO PRINT
1779   CCD7 BD B9 A2                 JSR   LB9A2                       SEND FILENAME TO CONSOLE OUT
178Ø   CCDA 8D 3F                    BSR   LCD1B                       SEND BLANK TO CONSOLE OUT
1781   CCDC C6 Ø3                    LDB   #$Ø3                        NUMBER CHARACTERS TO PRINT
1782   CCDE BD B9 A2                 JSR   LB9A2                       SEND EXTENSION TO CONSOLE OUT
1783   CCE1 8D 38                    BSR   LCD1B                       SEND BLANK TO CONSOLE OUT
1784   CCE3 E6 ØØ                    LDB   FCBTYP,X                    GET FILE TYPE
1785   CCE5 C1 ØA                    CMPB  #1Ø                         * CHECK THE NUMBER OF DECIMAL DIGITS IN
1786   CCE7 24 Ø2                    BHS   LCCEB                       * ACCB: IF THERE IS ONLY ONE DIGIT,
1787   CCE9 8D 3Ø                    BSR   LCD1B                       * SEND BLANK TO CONSOLE OUT
1788   CCEB 4F                LCCEB  CLRA                              CLEAR MS BYTE OF ACCO
1789   CCEC BD BD CC                 JSR   LBDCC                       PRINT ACCD IN DECIMAL TO CONSOLE OUT
179Ø   CCEF 8D 2A                    BSR   LCD1B                       SEND BLANK TO CONSOLE OUT
1791   CCF1 AE E4                    LDX   ,S                          X NOW POINTS TO DIRECTORY ENTRY
1792   CCF3 86 42                    LDA   #'A'+1                      ASCII BIAS
1793   CCF5 AB ØC                    ADDA  DIRASC,X                    ADD TO ASCII FLAG
1794   CCF7 8D 1F                    BSR   LCD18                       PRINT CHARACTER AND BLANK TO CONSOLE OUT
1795   CCF9 E6 ØD                    LDB   DIRGRN,X                    GET FIRST GRANULE IN FILE
1796   CCFB 8D 21                    BSR   LCD1E                       COUNT GRANULES
1797   CCFD 1F 89                    TFR   A,B                         SAVE COUNT IN ACCB
1798   CCFF 4F                       CLRA                              CLEAR MS BYTE OF ACCD
1799   CDØØ BD BD CC                 JSR   LBDCC                       PRINT ACCO IN DECIMAL TO CONSOLE OUT
18ØØ   CDØ3 BD B9 58                 JSR   LB958                       SEND CARRIAGE RETURN TO CONSOLE OUT
1801   CDØ6 35 1Ø                    PULS  X                           PULL DIRECTORY POINTER OFF OF THE STACK
1802   CDØ8 3Ø 88 2Ø          LCDØ8  LEAX  DIRLEN,X                    MOVE X TO NEXT DIRECTORY ENTRY
1803   CDØB 8C Ø7 ØØ                 CMPX  #DBUFØ+SECLEN               END OF I/O BUFFER?
1804   CDØE 25 B5                    BLO   LCCC5                       BRANCH IF MORE DIRECTORY ENTRIES IN BUFFER
1805   CD1Ø D6 ED                    LDB   DSEC                        GET CURRENT SECTOR
1806   CD12 5C                       INCB                              BUMP COUNT
1807   CD13 C1 12                    CMPB  #SECMAX                     SECMAX SECTORS IN DIRECTORY TRACK
1808   CD15 23 A4                    BLS   LCCBB                       GET NEXT SECTOR
1809   CD17 39                LCD17  RTS                               FINISHED
181Ø   CD18 BD A2 82          LCD18  JSR   LA282                       SEND CHARACTER TO CONSOLE OUT
1811   CD1B 7E B9 AC          LCD1B  JMP   LB9AC                       SEND BLANK TO CONSOLE OUT
1812
1813                          * ENTER WITH ACCB POINTING TO FIRST GRANULE IN A FILE; RETURN THE NUMBER OF
1814                          * GRANULES IN THE FILE IN ACCA, THE GRANULE DATA FOR THE LAST SECTOR IN ACCB
1815   CD1E BD C7 55          LCD1E  JSR   LC755                       POINT X TO FILE ALLOCATION BUFFER
1816   CD21 33 Ø6                    LEAU  FATCON,X                    POINT U TO START OF GRANULE DATA
1817   CD23 4F                       CLRA                              RESET GRANULE COUNTER
1818   CD24 4C                LCD24  INCA                              INCREMENT GRANULE COUNTER
1819   CD25 81 44                    CMPA  #GRANMX                     CHECKED ALL 68 GRANULES?
182Ø   CD27 1Ø 22 F9 28               LBHI  LC653                       YES - 'BAD FILE STRUCTURE' ERROR
1821   CD2B 3Ø C4                    LEAX  ,U                          POINT U TO START OF GRANULE DATA
1822   CD2D 3A                       ABX                               ADD POINTER TO FIRST GRANULE
1823   CD2E E6 84                    LDB   ,X                          GET THIS GRANULE'S CONTROL BYTE
1824   CD3Ø C1 CØ                    CMPB  #$CØ                        IS THIS THE LAST GRANULE IN FILE?
```

```
1825   CD32 25 F0                BLO   LCD24                       NO - KEEP GOING
1826   CD34 39                   RTS
1827
1828                       * INPUT RAM HOOK
1829   CD35 0D 6F          DVEC10 TST   DEVNUM                     * CHECK DEVICE NUMBER AND RETURN
1830   CD37 2F 5E                BLE   LCD97                       * IF NOT A DISK FILE
1831   CD39 8E B0 69             LDX   #LB069                      = CHANGE THE RETURN ADDRESS ON THE STACK TO RE-ENTER BASIC'S INPUT
1832   CD3C AF E4                STX   ,S                          = ROUTINE AT A DIFFERENT PLACE THAN THE CALLING ROUTINE
1833   CD3E 8E 02 DD             LDX   #LINBUF+1                   POINT X TO THE LINE INPUT BUFFER
1834   CD41 C6 2C                LDB   #','                        =
1835   CD43 D7 01                STB   CHARAC                      =COMMA IS READ ITEM SEPARATOR (TEMPORARY STRING SEARCH FLAG)
1836   CD45 96 06                LDA   VALTYP                      * GET VARIABLE TYPE AND BRANCH IF
1837   CD47 26 02                BNE   LCD4B                       * IT IS A STRING
1838   CD49 C6 20                LDB   #SPACE                      SPACE = NUMERIC SEARCH DELIMITER
1839   CD4B 8D 6F          LCD4B BSR   LCDBC                       GET AN INPUT CHARACTER
1840   CD4D 81 20                CMPA  #SPACE                      SPACE?
1841   CD4F 27 FA                BEQ   LCD4B                       YES - GET ANOTHER CHARACTER
1842   CD51 81 22                CMPA  #'"'                        QUOTE?
1843   CD53 26 0A                BNE   LCD5F                       NO
1844   CD55 C1 2C                CMPB  #','                        SEARCH CHARACTER = COMMA?
1845   CD57 26 06                BNE   LCD5F                       NO - NUMERIC SEARCH
1846   CD59 1F 89                TFR   A,B                         * SAVE DOUBLE QUOTE AS
1847   CD5B D7 01                STB   CHARAC                      * THE SEARCH FLAG
1848   CD5D 20 22                BRA   LCD81                       SAVE DOUBLE QUOTES AS FIRST ITEM IN BUFFER
1849
1850   CD5F C1 22          LCD5F CMPB  #'"'                        *
1851   CD61 27 11                BEQ   LCD74                       *BRANCH IF INPUTTING A STRING VARIABLE
1852   CD63 81 0D                CMPA  #CR                         IS THE INPUT CHARACTER A CARRIAGE RETURN
1853   CD65 26 0D                BNE   LCD74                       NO
1854   CD67 8C 02 DD             CMPX  #LINBUF+1                   *IF AT THE START OF INPUTBUFFER, CHECK FOR A
1855   CD6A 27 44                BEQ   LCDB0                       *FOLLOWING LINE FEED AND EXIT ROUTINE
1856   CD6C A6 1F                LDA   -1,X                        =IF THE INPUT CHARACTER PRECEEDING THE CR WAS A LINE FEED,
1857   CD6E 81 0A                CMPA  #LF                         =THEN INSERT THE CR IN THE INPUT STRING, OTHERWISE
1858   CD70 26 3E                BNE   LCDB0                       =CHECK FOR A FOLLOWING LINE FEED AND EXIT THE ROUTINE
1859   CD72 86 0D                LDA   #CR                         RESTORE CARRIAGE RETURN AS THE INPUT CHARACTER
1860   CD74 4D             LCD74 TSTA                              *CHECK FOR A NULL (ZERO) INPUT CHARACTER AND
1861   CD75 27 17                BEQ   LCD8E                       *IGNORE IT IF IT IS A NULL
1862   CD77 91 01                CMPA  CHARAC                      =
1863   CD79 27 1D                BEQ   LCD98                       =CHECK TO SEE IF THE INPUT CHARACTER MATCHES
1864   CD7B 34 04                PSHS  B                           =EITHER ACCB OR CHARAC AND IF IT DOES, THEN
1865   CD7D A1 E0                CMPA  ,S+                         =BRANCH TO CHECK FOR ITEM SEPARATOR OR
1866   CD7F 27 17                BEQ   LCD98                       =TERMINATOR SEQUENCE AND EXIT ROUTINE
1867   CD81 A7 80          LCD81 STA   ,X+                         STORE NEW CHARACTER IN BUFFER
1868   CD83 8C 03 D6             CMPX  #LINBUF+LBUFMX              END OF INPUT BUFFER
1869   CD86 26 06                BNE   LCD8E                       NO
1870   CD88 8D 46                BSR   LCDD0                       GET A CHARACTER FROM CONSOLE IN
1871   CD8A 26 06                BNE   LCD92                       EXIT ROUTINE IF BUFFER EMPTY
1872   CD8C 20 1E                BRA   LCDAC                       CHECK FOR CR OR CR/LF AND EXIT ROUTINE
1873
1874   CD8E 8D 40          LCD8E BSR   LCDD0                       GET A CHARACTER FROM CONSOLE IN
1875   CD90 27 CD                BEQ   LCD5F                       BRANCH IF BUFFER NOT EMPTY
1876   CD92 6F 84          LCD92 CLR   ,X                          PUT A ZERO AT END OF BUFFER WHEN DONE
1877   CD94 8E 02 DC             LDX   #LINBUF                     POINT (X) TO LINBUF - RESET POINTER
1878   CD97 39             LCD97 RTS
1879
1880                       * CHECK FOR ITEM SEPARATOR OR TERMINATOR AND EXIT THE INPUT ROUTINE
1881   CD98 81 22          LCD98 CMPA  #'"'                        QUOTE?
1882   CD9A 27 04                BEQ   LCDA0                       YES
1883   CD9C 81 20                CMPA  #SPACE                      SPACE?
1884   CD9E 26 F2                BNE   LCD92                       NO - EXIT ROUTINE
1885   CDA0 8D 2E                BSR   LCDD0                       GET A CHARACTER FROM CONSOLE IN
1886   CDA2 26 EE                BNE   LCD92                       EXIT ROUTINE IF BUFFER EMPTY
1887   CDA4 81 20                CMPA  #SPACE                      SPACE?
1888   CDA6 27 F8                BEQ   LCDA0                       YES - GET ANOTHER CHARACTER
1889   CDA8 81 2C                CMPA  #','                        COMMA (ITEM SEPARATOR)?
1890   CDAA 27 E6                BEQ   LCD92                       YES - EXIT ROUTINE
1891   CDAC 81 0D          LCDAC CMPA  #CR                         CARRIAGE RETURN?
1892   CDAE 26 08                BNE   LCDB8                       NO
1893   CDB0 8D 1E                BSR   LCDD0                       GET A CHARACTER FROM CONSOLE IN
1894   CDB2 26 DE                BNE   LCD92                       EXIT ROUTINE IF BUFFER EMPTY
1895   CDB4 81 0A                CMPA  #LF                         LINE FEED? TREAT CR,LF AS A CR
1896   CDB6 27 DA                BEQ   LCD92                       YES - EXIT ROUTINE
1897   CDB8 8D 1C          LCDB8 BSR   LCDD6                       BACK UP PTR INPUT POINTER ONE
1898   CDBA 20 D6                BRA   LCD92                       EXIT ROUTINE
1899
1900   CDBC 8D 12          LCDBC BSR   LCDD0                       GET A CHAR FROM INPUT BUFFER - RETURN IN ACCA
1901   CDBE 27 15                BEQ   LCDD5                       RETURN IF BUFFER NOT EMPTY
1902   CDC0 BD C7 44             JSR   LC744                       POINT X TO START OF FILE BUFFER
1903   CDC3 E6 00                LDB   FCBTYP,X                    GET FILE TYPE
1904   CDC5 C1 40                CMPB  #RANFIL                     IS IT RANDOM FILE TYPE?
1905   CDC7 10 26 F5 87          LBNE  LC352                       'INPUT PAST END OF FILE  ERROR IF NOT RANDOM
1906   CDCB C6 4A          LCDCB LDB   #2*37                       'WRITE/INPUT PAST END OF RECORD  ERROR IF RANDOM
1907   CDCD 7E AC 46             JMP   LAC46                       JUMP TO THE ERROR HANDLER
1908
1909   CDD0 BD A1 76       LCDD0 JSR   LA176                       GET A CHAR FROM INPUT BUFFER
1910   CDD3 0D 70                TST   CINBFL                      SET FLAGS ACCORDING TO CONSOLE INPUT FLAG
1911   CDD5 39             LCDD5 RTS
1912
1913                       * MOVE THE INPUT POINTER BACK ONE (DISK FILE)
1914   CDD6 34 14          LCDD6 PSHS  X,B                         SAVE REGISTERS ON STACK
1915   CDD8 BD C7 44             JSR   LC744                       POINT X TO PROPER FCB
1916   CDDB E6 00                LDB   FCBTYP,X                    GET FILE TYPE OF THIS FCB
1917   CDDD C1 40                CMPB  #RANFIL                     IS IT A RANDOM FILE?
1918   CDDF 26 0B                BNE   LCDEC                       BRANCH IF NOT A RANDOM FILE
1919   CDE1 EC 88 15             LDD   FCBGET,X                    *GRAB THE RANDOM FILE 'GET' POINTER,
1920   CDE4 83 00 01             SUBD  #$0001                      *MOVE IT BACK ONE AND RESTORE IT
```

```
1921   CDE7 ED 88 15              STD   FCBGET,X                    *
1922   CDEA 35 94                 PULS  B,X,PC                     RESTORE REGISTERS AND RETURN
1923   CDEC A7 88 11       LCDEC  STA   FCBCDT,X                   SAVE THE CHARACTER IN THE CACHE
1924   CDEF 63 88 10              COM   FCBCFL,X                   SET THE CACHE FLAG TO $FF - DATA IN CACHE
1925   CDF2 35 94                 PULS  B,X,PC                     RESTORE REGISTERS AND RETURN
1926
1927                       * CVN COMMAND
1928   CDF4 BD B6 54       CVN    JSR   LB654                      GET LENGTH AND ADDRESS OF STRING
1929   CDF7 C1 05                 CMPB  #$05                       FIVE BYTES IN A FLOATING POINT NUMBER
1930   CDF9 10 25 E6 4D           LBCS  LB44A                      'FC' ERROR IF <> 5 BYTES
1931   CDFD 0F 06                 CLR   VALTYP                     SET VARIABLE TYPE TO NUMERIC
1932   CDFF 7E BC 14              JMP   LBC14                      COPY A PACKED FP NUMBER FROM (X) TO FPA0
1933
1934                       * MKN$ COMMAND
1935   CE02 BD B1 43       MKN    JSR   LB143                      'TM' ERROR IF VALTYP=STRING
1936   CE05 C6 05                 LDB   #$05                       FIVE BYTES IN A FLOATING POINT NUMBER
1937   CE07 BD B5 0F              JSR   LB50F                      RESERVE FIVE BYTES IN STRING SPACE
1938   CE0A BD BC 35              JSR   LBC35                      PACK FPA0 AND STORE IT IN STRING SPACE
1939   CE0D 7E B6 9B              JMP   LB69B                      SAVE STRING DESCRIPTOR ON STRING STACK
1940
1941                       * LOC COMMAND
1942   CE10 8D 07          LOC    BSR   LCE19                      POINT X TO FILE BUFFER
1943   CE12 EC 07                 LDD   FCBREC,X                   GET RECORD NUMBER (RANDOM FILE) OR SECTOR CTR (SEQUENTIAL)
1944   CE14 DD 52          LCE14  STD   FPA0+2                     *SAVE ACCD IN BOTTOM 2 BYTES OF FPA0 AND
1945   CE16 7E 88 0E              JMP   L880E                      *CONVERT TO FLOATING POINT NUMBER
1946
1947                       * STRIP A DEVICE NUMBER FROM A BASIC STATEMENT, SET PRINT
1948                       * PARAMETERS ACCORDING TO IT - ERROR IF FILE NOT
1949                       * OPEN. RETURN WITH (X) POINTING TO THAT FILE'S FCB
1950   CE19 96 6F          LCE19  LDA   DEVNUM                      * GET CURRENT DEVICE NUMBER AND
1951   CE1B 34 02                 PSHS  A                          * SAVE IT ON THE STACK
1952   CE1D BD B1 43              JSR   LB143                      'TM' ERROR IF VALTYP=STRING
1953   CE20 BD A5 AE              JSR   LA5AE                      CHECK FOR VALID DEVICE NUMBER/SET PRINT PARAMETERS
1954   CE23 0D 6F                 TST   DEVNUM                     * CHECK DEVICE NUMBER
1955   CE25 10 2F E6 21           LBLE  LB44A                      * BRANCH IF NOT DISK FILE 'ILLEGAL FUNCTION CALL'
1956   CE29 BD C7 44              JSR   LC744                      POINT (X) TO FILE BUFFER
1957   CE2C 35 02                 PULS  A                          * GET OLD DEVICE NUMBER OFF OF THE STACK AND
1958   CE2E 97 6F                 STA   DEVNUM                     * SAVE IT AS DEVICE NUMBER
1959   CE30 6D 00                 TST   FCBTYP,X                   IS FILE OPEN?
1960   CE32 10 27 D5 C5           LBEQ  LA3FB                      'FILE NOT OPEN' ERROR IF NOT OPEN
1961   CE36 39                    RTS
1962
1963                       * LOF
1964   CE37 8D E0          LOF    BSR   LCE19                      POINT X TO FILE BUFFER
1965   CE39 A6 01                 LDA   FCBDRV,X                   * GET DRIVE NUMBER AND SAVE IT
1966   CE3B 97 EB                 STA   DCDRV                      * IN DSKCON VARIABLE
1967   CE3D E6 02                 LDB   FCBFGR,X                   GET FIRST GRANULE OF FILE
1968   CE3F 34 10                 PSHS  X                          SAVE FCB POINTER ON STACK
1969   CE41 BD CD 1E              JSR   LCD1E                      FIND TOTAL NUMBER OF GRANULES IN THIS FILE
1970   CE44 4A                    DECA                             SUBTRACT THE LAST GRANULE IN THE FILE
1971   CE45 C4 3F                 ANDB  #$3F                       GET NUMBER OF SECTORS USED IN LAST GRANULE
1972   CE47 34 04                 PSHS  B                          SAVE NUMBER OF SECTORS IN LAST GRANULE ON STACK
1973   CE49 1F 89                 TFR   A,B                        * CONVERT ACCA TO POSITIVE
1974   CE4B 4F                    CLRA                             * 2 BYTE VALUE IN ACCD
1975   CE4C BD C7 79              JSR   LC779                      MULT NUMBER OF FULL GRANULES BY 9
1976   CE4F EB E0                 ADDB  ,S+                        ADD NUMBER SECTORS IN LAST TRACK
1977   CE51 89 00                 ADCA  #$00                       PROPAGATE CARRY TO MS BYTE OF ACCD
1978   CE53 35 10                 PULS  X                          GET FCB POINTER BACK
1979   CE55 34 02                 PSHS  A                          SAVE ACCA ON STACK
1980   CE57 A6 00                 LDA   FCBTYP,X                   * GET FILE TYPE OF THIS FCB AND
1981   CE59 81 40                 CMPA  #RANFIL                    * CHECK TO SEE IF IT'S A RANDOM FILE
1982   CE5B 35 02                 PULS  A                          RESTORE ACCA
1983   CE5D 26 B5                 BNE   LCE14                      IF NOT A RANDOM FILE, THEN THE TOTAL NUMBER OF SECTORS IN THE FILE
1984                       *                                       IS THE LENGTH OF THE FILE
1985
1986                       * CALCULATE LOF FOR A RANDOM FILE - THE LENGTH OF A RANDOM FILE IS THE
1987                       * NUMBER OF RECORDS IN THE FILE.
1988   CE5F 34 10                 PSHS  X                          SAVE FCB POINTER ON STACK
1989   CE61 93 8A                 SUBD  ZERO                       SUBTRACT ZERO FROM ACCD (NUMBER OF SECTORS)
1990   CE63 27 03                 BEQ   LCE68                      BRANCH IF ZERO SECTORS
1991   CE65 83 00 01              SUBD  #$0001                     SUBTRACT ONE SECTOR - THE LAST SECTOR MAY NOT BE IOOZ USED
1992   CE68 8D AA          LCE68  BSR   LCE14                      PUT ACCD INTO FPA0
1993   CE6A D6 4F                 LDB   FP0EXP                     GET EXPONENT OF FPA0
1994   CE6C 27 04                 BEQ   LCE72                      BRANCH IF FPA0 = 0
1995   CE6E CB 08                 ADDB  #$08                       * ADD 8 TO EXPONENT (MULTIPLY FPA0 BY
1996   CE70 D7 4F                 STB   FP0EXP                     * 256 BYTES/SECTOR) AND SAVE NEW EXPONENT
1997   CE72 BD BC 5F       LCE72  JSR   LBC5F                      SAVE NUMBER OF BYTES IN FULL SECTORS IN FPA1
1998   CE75 AE E4                 LDX   ,S                         POINT X TO FCB
1999   CE77 EC 88 13              LDD   FCBLST,X                   GET NUMBER OF BYTES IN LAST SECTOR
2000   CE7A 84 7F                 ANDA  #$7F                       MASK OFF THE PRE-SAVED BYTE
2001   CE7C 8D 96                 BSR   LCE14                      PUT NUMBER BYTES IN LAST SECTOR INTO FPA0
2002   CE7E 0F 62                 CLR   RESSGN                     FORCE SUM SIGN = POSITIVE
2003   CE80 96 5C                 LDA   FP1EXP                     * GET EXPONENTS OF FPA0 AND
2004   CE82 D6 4F                 LDB   FP0EXP                     * FPA1 PRIOR TO ADDITION
2005   CE84 BD B9 C5              JSR   LB9C5                      ADD NUMBER BYTES IN LAST SECTOR TO NUMBER OF
2006                                                               BYTES IN FULL SECTORS
2007   CE87 BD BC 5F              JSR   LBC5F                      SAVE TOTAL NUMBER OF BYTES IN FPA1
2008   CE8A 35 10                 PULS  X                          POINT X TO FCB
2009   CE8C EC 09                 LDD   FCBRLN,X                   * GET RECORD LENGTH
2010   CE8E 8D 84                 BSR   LCE14                      * PUT IT INTO FPA0
2011   CE90 0F 62                 CLR   RESSGN                     FORCE QUOTIENT SIGN = POSITIVE
2012   CE92 96 5C                 LDA   FP1EXP                     * GET EXPONENTS OF FPA0 AND
2013   CE94 D6 4F                 LDB   FP0EXP                     * FPA1 PRIOR TO DIVISION
2014   CE96 BD BB 91              JSR   LBB91                      DIVIDE TOTAL NUMBER OF BYTES BY NUMBER OF BYTES IN A RECORD
2015   CE99 7E BC EE              JMP   INT                        CONVERT FPA0 TO AN INTEGER
2016
```

```
2017                            * FREE COMMAND
2018   CE9C BD B1 43    FREE    JSR   LB143                     * NUMBER TYPE CHECK
2019   CE9F BD B7 0E            JSR   LB70E                     *EVALUATE NUMERIC EXPRESSION AND RETURN VALUE IN ACCB
2020   CEA2 C1 03               CMPB  #$03                      ONLY 4 LEGAL DRIVES
2021   CEA4 10 22 D7 77         LBHI  LA61F                     'DEVICE NUMBER' ERROR IF DRIVE NUMBER IS > 3
2022   CEA8 D7 EB               STB   DCDRV                     SAVE IN DRIVE NUMBER
2023   CEAA BD C7 9D            JSR   LC79D                     GET FILE ALLOCATION TABLE AND STORE IN BUFFER
2024   CEAD BD C7 55            JSR   LC755                     POINT X TO START OF FILE ALLOCATION TABLE BUFFER
2025   CEB0 30 06               LEAX  FATCON,X                  MOVE TO FIRST GRANULE DATA BYTE
2026   CEB2 6F E2               CLR   ,-S                       SPACE FOR FREE GRANULE COUNTER
2027   CEB4 C6 44               LDB   #GRANMX                   GET MAXIMUM NUMBER OF GRANULES
2028   CEB6 A6 80       LCEB6   LDA   ,X+                       GET GRANULE DATA
2029   CEB8 43                  COMA                            *FREE GRANULES $FF
2030   CEB9 26 02               BNE   LCEBD                     *BRANCH IF NOT FREE
2031   CEBB 6C E4               INC   ,S                        INCREMENT FREE GRANULE COUNTER
2032   CEBD 5A          LCEBD   DECB                            DECREMENT GRANULE COUNTER
2033   CEBE 26 F6               BNE   LCEB6                     BRANCH IF NOT DONE
2034   CEC0 35 04               PULS  B                         GET FREE GRANULE COUNTER TO ACCB
2035   CEC2 7E B4 F3            JMP   LB4F3                     LOAD ACCB INTO FPA0
2036
2037                            * DRIVE COMMAND
2038   CEC5 BD B7 0B    DRIVE   JSR   EVALEXPB                  EVALUATE EXPR; RETURN VALUE IN ACCB
2039   CEC8 C1 03               CMPB  #$03                      MAX DRIVE NUMBER = 3
2040   CECA 10 22 D7 51         LBHI  LA61F                     'DEVICE #' ERROR IF DRIVE NUMBER > 3
2041   CECE F7 09 5A            STB   DEFDRV                    SAVE DEFAULT DRIVE NUMBER
2042   CED1 39                  RTS
2043
2044                            * EVALUATE EXPRESSION RAM VECTOR
2045   CED2 A6 64       DVEC15  LDA   $04,S                     = CHECK STACKED PRECEDENCE FLAG AND IF IT IS NOT AN END
2046   CED4 26 13               BNE   LCEE9                     = OF OPERATION, BRANCH TO EXTENDED BASIC'S EXPRESSION
2047                                                            = EVALUATION ROUTINE
2048   CED6 AE 65               LDX   $05,S                     *
2049   CED8 8C AF 9A            CMPX  #LAF9A                    *
2050   CEDB 26 0C               BNE   LCEE9                     * CHECK TWO RETURN ADDRESSES BACK ON THE STACK
2051   CEDD AE 62               LDX   $02,S                     * TO SEE IF THE CALL TO EVALUATE EXPRESSION IS
2052   CEDF 8C B1 66            CMPX  #LB166                    * COMING FROM THE 'LET' COMMAND - BRANCH OUT IF
2053   CEE2 26 05               BNE   LCEE9                     * NOT COMING FROM 'LET'
2054   CEE4 8E CE EC            LDX   #LCEEC                    = IF COMING FROM 'LET', REPLACE THE RETURN ADDR
2055   CEE7 AF 65               STX   $05,S                     = WITH THE DISK BASIC 'LET' MODIFIER ADDRESS
2056   CEE9 7E 88 46    LCEE9   JMP   XVEC15                    EXTENDED BASIC EXPRESSION EVALUATION
2057
2058                            * LET MODIFIER
2059   CEEC 35 02               PULS  A                         PULL VARIABLE TYPE OFF OF THE STACK
2060   CEEE 46                  RORA                            SET CARRY IF SIRING, CLEAR CARRY IF NUMERIC
2061   CEEF BD B1 48            JSR   LB148                     DO A 'TM' CHECK
2062   CEF2 10 27 ED 3D         LBEQ  LBC33                     IF NUMERIC VARIABLE, PACK FPA0 INTO VARDES
2063   CEF6 9E 52               LDX   FPA0+2                    POINT X TO STRING DESCRIPTOR
2064   CEF8 EC 02               LDD   $02,X                     GET ADDRESS OF SIRING
2065   CEFA 10 83 09 89         CMPD  #DFLBUF                   * COMPARE TO START OF RANDOM FILE BUFFERS
2066   CEFE 25 07               BLO   LCF07                     * AND BRANCH IF LOWER
2067   CF00 B3 09 4A            SUBD  FCBADR                    SUBTRACT OUT THE END OF RANDOM FILE BUFFERS
2068   CF03 10 25 E0 AA         LBCS  LAFB1                     BRANCH IF STRING STORED IN RANDOM FILE BUFFER -
2069                                                            MOVE IT INTO THE STRING SPACE
2070   CF07 7E AF A4    LCF07   JMP   LAFA4                     BRANCH BACK TO BASIC S 'LET' COMMAND
2071
2072                            *MODIFIER FOR EXBAS COMMAND INTERPRETATION HANDLER
2073   CF0A 81 CA       DXCVEC  CMPA  #$CA                      TOKEN FOR DLOAD?
2074   CF0C 27 1C               BEQ   LCF2A                     YES
2075   CF0E 81 C8               CMPA  #$C8                      TOKEN FOR PMODE?
2076   CF10 10 26 B2 28         LBNE  L813C                     NO
2077                            * DISK BASIC MODIFIER FOR PMODE - ALLOWS FOR THE RAM THE DOS USES
2078   CF14 9D 9F               JSR   GETNCH                    GET NEXT CHARACTER FROM BASIC
2079   CF16 81 2C               CMPA  #','                      CHECK FOR COMMA
2080   CF18 10 27 C7 34         LBEQ  L9650                     BRANCH IF COMMA
2081   CF1C BD B7 0B            JSR   EVALEXPB                  EVALUATE EXPRESSION; RETURN VALUE IN ACCB
2082   CF1F C1 04               CMPB  #$04                      CHECK FOR PMODE 4
2083   CF21 10 22 E5 25         LBHI  LB44A                     'FC' ERROR IF PMODE > 4
2084   CF25 96 BC               LDA   GRPRAM                    NUMBER BLOCKS BEFORE GRAPHICS PAGES
2085   CF27 7E 96 2E            JMP   L962E                     JUMP TO EXEAS' PMODE COMMAND
2086
2087                            * DISK BASIC DLOAD MODIFIER
2088   CF2A BD A4 29    LCF2A   JSR   LA429                     CLOSE FILES
2089   CF2D 9D 9F               JSR   GETNCH                    GET NEXT CHARACTER FROM BASIC
2090   CF2F 7E 8C 1B            JMP   L8C1B                     JUMP TO EXEAS' DLOAD
2091
2092   CF32 C1 34       DXIVEC  CMPB  #($9A-$80)*2              MODIFIED TOKEN FOR POS
2093   CF34 10 26 B2 30         LBNE  L8168                     IF NOT POS, GO TO EXBAS SECONDARY COMM HANDLER
2094   CF38 BD B2 62            JSR   LB262                     SYNTAX CHECK FOR '(' AND EVALUATE EXPRESSION
2095   CF3B 96 6F               LDA   DEVNUM                    * GET DEVICE NUMBER AND
2096   CF3D 34 02               PSHS  A                         * SAVE IT ON STACK
2097   CF3F BD A5 AE            JSR   LA5AE                     EVALUATE DEVICE NUMBER
2098   CF42 BD A4 06            JSR   LA406                     TEST DEVICE NUMBER
2099   CF45 0D 6F               TST   DEVNUM                    * CHECK DEVICE NUMBER AND BRANCH
2100   CF47 2F 13               BLE   LCF5C                     * IF NOT A DISK FILE
2101   CF49 BD C7 44            JSR   LC744                     POINT X TO FCB
2102   CF4C E6 00               LDB   FCBTYP,X                  GET FILE TYPE
2103   CF4E C1 40               CMPB  #RANFIL                   DIRECT/RANDOM FILE?
2104   CF50 26 0A               BNE   LCF5C                     BRANCH IF NOT A RANDOM FILE
2105   CF52 35 02               PULS  A                         * RESTORE DEVICE NUMBER
2106   CF54 97 6F               STA   DEVNUM                    *
2107   CF56 EC 88 17            LDD   FCBPUT,X                  =GRAB THE 'PUT' DATA ITEM COUNTER AND CONVERT
2108   CF59 7E B4 F4            JMP   GIVABF                    =IT TO A FLOATING POINT NUMBER
2109   CF5C BD A3 5F    LCF5C   JSR   LA35F                     SET PRINT PARAMETERS
2110   CF5F 35 02               PULS  A                         * RESTORE DEVICE NUMBER
2111   CF61 97 6F               STA   DEVNUM                    *
2112   CF63 D6 6C               LDB   DEVPOS                    =GET PRINT POSITION AND
```

```
2113   CF65 7E B4 F3                JMP   LB4F3                          =CONVERT IT TO FLOATING POINT NUMBER IN FPA0
2114
2115                          * SAVEM COMMAND
2116   CF68 9D 9F             LCF568 JSR   GETNCH                         GET NEXT INPUT CHARACTER
2117   CF6A 8D 4F                    BSR   LCFBB                          GET FILENAME, ETC.
2118   CF6C BD 83 6C                 JSR   L836C                          EVALUATE EXPRESSION, PUT II (2 BYTES) ON STACK
2119   CF6F BD 83 6C                 JSR   L836C                          DITTO
2120   CF72 AC 62                    CMPX  $02,S                          COMPARE END ADDRESS TO START ADDRESS
2121   CF74 10 25 E4 D2              LBCS  LB44A                          IF START > END, THEN 'ILLEGAL FUNCTION CALL'
2122   CF78 BD 83 6C                 JSR   L836C                          EVAL EXPRESSION (TRANSFER ADDRESS), PUT ON STACK
2123   CF7B BD A5 C7                 JSR   LA5C7                          SYNTAX ERROR IF ANY MORE CHARS ON THIS LINE
2124   CF7E CC 02 00                 LDD   #$0200                         * FILE TYPE=2, ASCII FLAG = CRUNCHED (0)
2125   CF81 FD 09 57                 STD   DFLTYP                         *
2126   CF84 BD CA 04                 JSR   LCA04                          GET NEXT UNOPEN FILE AND INITIALIZE FCB
2127   CF87 4F                       CLRA                                 *ZERO FLAG - FIRST BYTE OF PREAMBLE
2128   CF88 8D 2B                    BSR   LCFB5                          *WRITE A BYTE TO BUFFER
2129   CF8A EC 62                    LDD   $02,S                          GET END ADDRESS
2130   CF8C A3 64                    SUBD  $04,S                          SUBTRACT THE START ADDRESS
2131   CF8E C3 00 01                 ADDD  #$0001                         THE SAVED DATA BLOCK WILL INCLUDE BOTH THE FIRST AND LAST BYTES
2132   CF91 1F 02                    TFR   D,Y                            SAVE LENGTH IN Y
2133   CF93 8D 1E                    BSR   LCFB3                          WRITE FILE LENGTH TO BUFFER - FIRST ARGUMENT OF PREAMBLE
2134   CF95 EC 64                    LDD   $04,S                          GET THE START ADDRESS
2135   CF97 8D 1A                    BSR   LCFB3                          WRITE OUT THE START ADDRESS - SECOND PREAMBLE ARGUMENT
2136   CF99 AE 64                    LDX   $04,S                          GET START ADDRESS
2137   CF9B A6 80             LCF9B  LDA   ,X+                            GRAB A BYTE
2138   CF9D BD CC 24                 JSR   LCC24                          WRITE IT OUT
2139   CFA0 31 3F                    LEAY  -1,Y                           DECREMENT BYTE COUNTER
2140   CFA2 26 F7                    BNE   LCF9B                          BRANCH IF ALL BYTES NOT DONE
2141   CFA4 86 FF                    LDA   #$FF                           FIRST BYTE OF POSTAMBLE
2142   CFA6 8D 0D                    BSR   LCFB5                          WRITE IT OUT - EOF RECORD
2143   CFA8 4F                       CLRA                                 * FIRST ARGUMENT OF POSTAMBLE IS
2144   CFA9 5F                       CLRB                                 * A DUMMY - ZERO VALUE
2145   CFAA 8D 07                    BSR   LCFB3                          WRITE OUT POSTAMBLE FIRST ARGUMENT
2146   CFAC 35 36                    PULS  A,B,X,Y                        GET CONTROL ADDRESSES FROM THE STACK
2147   CFAE 8D 03                    BSR   LCFB3                          WRITE OUT THE TRANSFER ADDRESS - 2ND ARGUMENT
2148   CFB0 7E A4 2D                 JMP   LA42D                          GO CLOSE ALL FILES
2149
2150                          * WRITE ACCD TO THE BUFFER
2151   CFB3 8D 00             LCFB3  BSR   LCFB5                          WRITE ACCA TO BUFFER, THEN SWAP ACCA,ACCB
2152   CFB5 BD CC 24          LCFB5  JSR   LCC24                          WRITE ACCA TO BUFFER
2153   CFB8 1E 89                    EXG   A,B                            SWAP ACCA,ACCB
2154   CFBA 39                       RTS
2155   CFBB 8E C2 AF          LCFBB  LDX   #BINEXT                        POINT TO .BIN EXTENSION
2156   CFBE 7E C9 38                 JMP   LC938                          GET FILENAME, ETC.
2157
2158                          * LOADM COMMAND
2159   CFC1 9D 9F             LCFC1  JSR   GETNCH                         GET NEXT INPUT CHARACTER
2160   CFC3 8D F6                    BSR   LCFBB                          GET FILENAME, ETC.
2161   CFC5 BD CA 07                 JSR   LCA07                          OPEN NEXT AVAILABLE FILE FOR INPUT
2162   CFC8 FC 09 57                 LDD   DFLTYP                         GET FILE TYPE AND ASCII FLAG
2163   CFCB 83 02 00                 SUBD  #$0200                         FOR LOADM FILE: TYPE=2, ASCII FLAG=0
2164   CFCE 10 26 D6 44              LBNE  LA616                          'BAD FILE MODE' ERROR
2165   CFD2 9E 8A                    LDX   ZERO                           ZERO OUT X REG - DEFAULT VALUE OF OFFSET
2166   CFD4 9D A5                    JSR   GETCCH                         GET CURRENT CHARACTER FROM BASIC
2167   CFD6 27 06                    BEQ   LCFDE                          BRANCH IF END OF LINE - NO OFFSET
2168   CFD8 BD B2 6D                 JSR   SYNCOMMA                       SYNTAX CHECK FOR COMMA
2169   CFDB BD B7 3D                 JSR   LB73D                          EVALUATE EXPRESSION
2170   CFDE 9F D3             LCFDE  STX   VD3                            STORE OFFSET IN VD3
2171   CFE0 BD A5 C7                 JSR   LA5C7                          SYNTAX ERROR IF OTHER CHARACTERS ON LINE
2172
2173                          * GET PREAMBLE/POSTAMBLE
2174   CFE3 BD CD BC          LCFE3  JSR   LCDBC                          GET FIRST BYTE
2175   CFE6 34 02                    PSHS  A                              SAVE IT ON THE STACK
2176   CFE8 8D 29                    BSR   LD013                          GET FIRST ARGUMENT
2177   CFEA 1F 02                    TFR   D,Y                            SAVE IT IN Y
2178   CFEC 8D 25                    BSR   LD013                          GET THE SECOND ARGUMENT
2179   CFEE D3 D3                    ADDD  VD3                            ADD IT TO THE OFFSET
2180   CFF0 DD 9D                    STD   EXECJP                         STORE IT IN THE JUMP ADDRESS OF THE EXEC COMMAND
2181   CFF2 1F 01                    TFR   D,X                            SAVE IT IN X
2182   CFF4 A6 E0                    LDA   ,S+                            GET THE FIRST BYTE OFF OF THE STACK
2183   CFF6 10 26 D4 33              LBNE  LA42D                          CLOSE FILE IF POSTAMBLE (EOF)
2184
2185                          * GET RECORD BYTE(S)
2186   CFFA BD C5 C4          LCFFA  JSR   LC5C4                          GET BYTE FROM BUFFER
2187   CFFD D6 70                    LDB   CINBFL                         GET STATUS OF CONSOLE IN BUFFER
2188   CFFF 27 03                    BEQ   LD004                          BRANCH IF BUFFER NOT EMPTY
2189   D001 7E C3 52                 JMP   LC352                          'INPUT PAST END OF FILE' ERROR
2190   D004 A7 84             LD004  STA   ,X                             STORE BYTE IN MEMORY
2191   D006 A1 80                    CMPA  ,X+                            *TEST TO SEE IF IT STORED PROPERLY AND
2192   D008 27 03                    BEQ   LD00D                          *BRANCH IF PROPER STORE (NOT IN ROM OR BAD RAM)
2193   D00A 7E D7 09                 JMP   LD709                          'I/O ERROR' IF BAD STORE
2194   D00D 31 3F             LD00D  LEAY  -1,Y                           DECREMENT BYTE COUNT
2195   D00F 26 E9                    BNE   LCFFA                          GET NEXT BYTE IF NOT DONE
2196   D011 20 D0                    BRA   LCFE3                          READ ANOTHER PRE/POST AMBLE
2197                          * READ TWO BYTES FROM BUFFER - RETURN THEM IN ACCD
2198   D013 8D 00             LD013  BSR   LD015                          READ A BYTE, SAVE IT IN ACCB
2199   D015 BD CD BC          LD015  JSR   LCDBC                          GET A CHARACTER FROM INPUT BUFFER, RETURN IT IN ACCA
2200   D018 1E 89                    EXG   A,B                            SWAP ACCA,ACCB
2201   D01A 39                       RTS
2202
2203                          * RENAME COMMAND
2204   D01B 9E A6             RENAME LDX   CHARAD                         * SAVE CURRENT INPUT POINTER
2205   D01D 34 10                    PSHS  X                              * ON THE STACK
2206   D01F 8D 35                    BSR   LD056                          GET FILENAME OF SOURCE FILE
2207   D021 96 EB                    LDA   DCDRV                          * SAVE DRIVE NUMBER
2208   D023 34 02                    PSHS  A                              * ON THE STACK
```

```
2209  D025 8D 2A                BSR   LD051                       SYNTAX CHECK FOR 'TO' AND GET NEW FILENAME
2210  D027 35 02                PULS  A                           GET SOURCE DRIVE NUMBER
2211  D029 91 EB                CMPA  DCDRV                       COMPARE TO NEW FILE DRIVE NUMBER
2212  D02B 10 26 E4 1B          LBNE  LB44A                       'FC' ERROR IF F1IES ON DIFFERENT DRIVES
2213  D02F 8D 28                BSR   LD059                       VERIFY THAT NEW FILE DOES NOT ALREADY EXIST
2214  D031 35 10                PULS  X                           * RESTORE INPUT POINTER
2215  D033 9F A6                STX   CHARAD                      *
2216  D035 8D 1F                BSR   LD056                       GET SOURCE FILENAME AGAIN
2217  D037 BD C6 8C             JSR   LC68C                       SCAN DIRECTORY FOR SOURCE FILENAME
2218  D03A BD C6 E5             JSR   LC6E5                       'NE' ERROR IF NOT FOUND
2219  D03D 8D 12                BSR   LD051                       SYNTAX CHECK FOR 'TO' AND GET NEW FILENAME
2220  D03F 8E 09 4C             LDX   #DNAMBF                     POINT X TO FILENAME
2221  D042 FE 09 74             LDU   V974                        POINT U TO DIRECTORY ENTRY OF SOURCE FILE
2222  D045 C6 0B                LDB   #$0B                        11 CHARACTERS IN FILENAME AND EXTENSION
2223  D047 BD A5 9A             JSR   LA59A                       COPY NEW FILENAME TO SOURCE FILE DIRECTORY RAM IMAGE
2224  D04A C6 03                LDB   #$03                        * GET WRITE OP CODE AND
2225  D04C D7 EA                STB   DCOPC                       * SAVE IN DSKCON VARIABLE
2226  D04E 7E D6 F2             JMP   LD6F2                       WRITE NEW DIRECTORY SECTOR
2227
2228                        * DO A SYNTAX CHECK FOR 'TO  AND STRIP A FILENAME FROM BASIC
2229  D051 C6 A5            LD051   LDB   #$A5                      'TO' TOKEN
2230  D053 BD B2 6F             JSR   LB26F                       SYNTAX CHECK FOR 'TO'
2231  D056 7E C9 35         LD056   JMP   LC935                     GET FILENAME FROM BASIC
2232  D059 BD C6 8C         LD059   JSR   LC68C                     SCAN DIRECTORY FOR FILENAME
2233  D05C C6 42                LDB   #33*2                       'FILE ALREADY EXISTS' ERROR
2234  D05E 7D 09 73             TST   V973                        CHECK FOR A MATCH
2235  D061 10 26 DB E1          LBNE  LAC46                       'AE' ERROR IF FILE IN DIRECTORY
2236  D065 39                   RTS
2237
2238                        * WRITE COMMAND
2239  D066 10 27 E8 EE      WRITE   LBEQ  LB958                     PRINT CARRIAGE RETURN TO CONSOLE OUT IF END OF LINE
2240  D06A 8D 03                BSR   LD06F                       GO WRITE AN ITEM LIST
2241  D06C 0F 6F                CLR   DEVNUM                      SET DEVICE NUMBER TO SCREEN
2242  D06E 39               LD06E   RTS
2243  D06F 81 23            LD06F   CMPA  #'#'                      CHECK FOR DEVICE NUMBER FLAG
2244  D071 26 0F                BNE   LD082                       DEFAULT TO CURRENT DEVICE NUMBER IF NONE GIVEN
2245  D073 BD A5 A5             JSR   LA5A5                       SET DEVICE NUMBER; CHECK VALIDITY
2246  D076 BD A4 06             JSR   LA406                       MAKE SURE SELECTED FILE IS AN OUTPUT FILE
2247  D079 9D A5                JSR   GETCCH                      GET CURRENT INPUT CHARACTER
2248  D07B 10 27 E8 D9          LBEQ  LB958                       PRINT CR TO CONSOLE OUT IF END OF LINE
2249  D07F BD B2 6D         LD07F   JSR   SYNCOMMA                  SYNTAX CHECK FOR COMMA
2250  D082 BD B1 56         LD082   JSR   LB156                     EVALUATE EXPRESSION
2251  D085 96 06                LDA   VALTYP                      GET VARIABLE TYPE
2252  D087 26 1E                BNE   LD0A7                       BRANCH IF STRING
2253  D089 BD BD D9             JSR   LBDD9                       CONVERT FP NUMBER TO ASCII STRING
2254  D08C BD B5 16             JSR   LB516                       PUT ON TEMPORARY STRING STACK
2255  D08F BD B9 9F             JSR   LB99F                       PRINT STRING TO CONSOLE OUT
2256
2257                        * PRINT ITEM SEPARATOR TO CONSOLE OUT
2258  D092 9D A5            LD092   JSR   GETCCH                    GET CURRENT CHARACTER
2259  D094 10 27 E8 C0          LBEQ  LB958                       PUT CR TO CONSOLE OUT IF END OF LINE
2260  D098 86 2C                LDA   #','                        COMMA: NON-CASSETTE SEPARATOR
2261  D09A BD A3 5F             JSR   LA35F                       SET PRINT PARAMETERS
2262  D09D 0D 6E                TST   PRTDEV                      * GET CONSOLE PRINT DEVICE AND
2263  D09F 27 02                BEQ   LD0A3                       * BRANCH IF NOT CASSETTE
2264  D0A1 86 0D                LDA   #CR                         GET CARRIAGE RETURN - CASSETTE ITEM SEPARATOR
2265  D0A3 8D 14            LD0A3   BSR   LD0B9                     SEND SEPARATOR TO CONSOLE OUT
2266  D0A5 20 D8                BRA   LD07F                       GET NEXT ITEM
2267
2268                        * PRINT A STRING TO CONSOLE OUT
2269  D0A7 8D 07            LD0A7   BSR   LD0B0                     PRINT LEADING STRING DELIMITER (")
2270  D0A9 BD B9 9F             JSR   LB99F                       PRINT STRING TO CONSOLE OUT
2271  D0AC 8D 02                BSR   LD0B0                       PRINT ENDING STRING DELIMITER (")
2272  D0AE 20 E2                BRA   LD092                       GO PRINT SEPARATOR
2273
2274                        * PRINT STRING DELIMITER (") TO CONSOLE OUT
2275  D0B0 BD A3 5F         LD0B0   JSR   LA35F                     SET PRINT PARAMETERS
2276  D0B3 0D 6E                TST   PRTDEV                      * GET CONSOLE PRINT DEVICE AND
2277  D0B5 26 B7                BNE   LD06E                       * RETURN IF CASSETTE
2278  D0B7 86 22                LDA   #'"'                        QUOTE: NON-CASSETTE STRING DELIMITER
2279  D0B9 7E A2 82         LD0B9   JMP   LA282                     SEND TO CONSOLE OUT
2280
2281                        * FIELD COMMAND
2282  D0BC BD C8 2E         FIELD   JSR   LC82E                     EVALUATE DEVICE NUMBER & VERIFY RANDOM FILE OPEN
2283  D0BF 4F                   CLRA                              *
2284  D0C0 5F                   CLRB                              * CLEAR TOTAL FIELD LENGTH COUNTER
2285  D0C1 34 16                PSHS  X,B,A                       SAVE FCB POINTER & INITIALIZE TOTAL FIELD LENGTH TO ZERO
2286  D0C3 9D A5            LD0C3   JSR   GETCCH                    GET CURRENT INPUT CHARACTER
2287  D0C5 26 02                BNE   LD0C9                       BRANCH IF NOT END OF LINE
2288  D0C7 35 96                PULS  A,B,X,PC                    CLEAN UP STACK AND RETURN
2289  D0C9 BD B7 38         LD0C9   JSR   LB738                     SYNTAX CHECK FOR COMMA, EVALUATE EXPRESSION
2290  D0CC 34 14                PSHS  X,B                         SAVE FIELD LENGTH (ACCB) ON STACK, X IS A DUMMY WHICH WILL
2291                                                              RESERVE 2 BYTES FOR THE ADDRESS WHICH WILL BE CALCULATED BELOW
2292                        * AT THIS POINT THE STACK WILL HAVE THE FOLLOWING INFORMATION ON IT:
2293                        * ,S = FIELD LENGTH   1 2,S = RANDOM FILE BUFFER ADDRESS
2294                        * 3 4,S = TOTAL FIELD LENGTH  5 6,S = FCD POINTER
2295  D0CE 4F                   CLRA                              CLEAR MS BYTE
2296  D0CF E3 63                ADDD  $03,S                       ADD FIELD LENGTH TO TOTAL FIELD LENGTH COUNTER
2297  D0D1 25 07                BLO   LD0DA                       'FO' ERROR IF SUM > $FFFF
2298  D0D3 AE 65                LDX   $05,S                       POINT X TO FCB
2299  D0D5 10 A3 09             CMPD  FCBRLN,X                    * COMPARE TO RECORD LENGTH & BRANCH IF
2300  D0D8 23 05                BLS   LD0DF                       *TOTAL FIELD LENGTH < RECORD LENGTH
2301  D0DA C6 44            LD0DA   LDB   #34*2                     'FIELD OVERFLOW' ERROR
2302  D0DC 7E AC 46             JMP   LAC46                       JUMP TO ERROR DRIVER
2303  D0DF EE 63            LD0DF   LDU   $03,S                     LOAD U WITH OLD TOTAL LENGTH OF ALL FIELDS
2304  D0E1 ED 63                STD   $03,S                       SAVE NEW TOTAL FIELD LENGTH
```

```
2305   D0E3 EC 0B                    LDD   FCBBUF,X                    POINT ACCD TO START OF RANDOM FILE BUFFER
2306   D0E5 33 CB                    LEAU  D,U                         *POINT U TO THIS FIELD'S SLOT IN THE RANDOM
2307   D0E7 EF 61                    STU   $01,S                       *FILE BUFFER AND SAVE IT ON THE STACK
2308   D0E9 C6 FF                    LDB   #$FF                        SECONDARY TOKEN
2309   D0EB BD B2 6F                 JSR   LB26F                       SYNTAX CHECK FOR SECONDARY TOKEN
2310   D0EE C6 A7                    LDB   #$A7                        'AS' TOKEN
2311   D0F0 BD B2 6F                 JSR   LB26F                       SYNTAX CHECK FOR 'AS' TOKEN
2312   D0F3 BD B3 57                 JSR   LB357                       EVALUATE VARIABLE
2313   D0F6 BD B1 46                 JSR   LB146                       'TM' ERROR IF NUMERIC VARIABLE
2314   D0F9 35 44                    PULS  B,U                         * PULL STRING ADDRESS AND LENGTH
2315   D0FB E7 84                    STB   ,X                          * OFF OF THE STACK AND SAVE THEM
2316   D0FD EF 02                    STU   $02,X                       * IN STRING DESCRIPTOR
2317   D0FF 20 C2                    BRA   LD0C3                       CHECK FOR ANOTHER FIELD SPECIFICATION
2318
2319                        * RSET COMMAND
2320   D101 86              RSET     LDA   #$4F                        SKIP ONE BYTE
2321
2322                        * LSET COMMAND
2323   D102 4F              LSET     CLRA                              LSET FLAG = 0
2324   D103 34 02                    PSHS  A                           SAVE RSET($4F),LSET(00) FLAG ON THE STACK
2325   D105 BD B3 57                 JSR   LB357                       EVALUATE FIELD STRING VARIABLE
2326   D108 BD B1 46                 JSR   LB146                       'TM' ERROR IF NUMERIC VARIABLE
2327   D10B 34 10                    PSHS  X                           SAVE STRING DESCRIPTOR ON STACK
2328   D10D AE 02                    LDX   $02,X                       POINT X TO ADDRESS OF STRING
2329   D10F 8C 09 89                 CMPX  #DFLBUF                     * COMPARE STRING ADDRESS TO START OF RANDOM
2330   D112 25 05                    BLO   LD119                       * FILE BUFFER; 'SE' ERROR IF < RANDOM FILE BUFFER
2331   D114 BC 09 4A                 CMPX  FCBADR                      = COMPARE STRING ADDRESS TO TOP OF RANDOM FILE BUFFER
2332   D117 25 05                    BLO   LD11E                       = AREA - BRANCH IF STRING IN RANDOM FILE BUFFER
2333   D119 C6 46           LD119    LDB   #2*35                       'SET TO NON-FIELDED STRING' ERROR
2334   D11B 7E AC 46                 JMP   LAC46                       JUMP TO ERROR HANDLER
2335   D11E C6 B3           LD11E    LDB   #$B3                        *
2336   D120 BD B2 6F                 JSR   LB26F                       * SYNTAX CHECK FOR '=' TOKEN
2337   D123 BD 87 48                 JSR   L8748                       =EVALUATE DATA STRING EXPRESSION; RETURN WITH X
2338                        *                                           =POINTING TO STRING; ACCB = LENGTH
2339   D126 35 20                    PULS  Y                           POINT Y TO FIELD STRING DESCRIPTOR
2340   D128 A6 A4                    LDA   ,Y                          GET LENGTH OF FIELD STRING
2341   D12A 27 2E                    BEQ   LD15A                       RETURN IF NULL STRING
2342   D12C 34 04                    PSHS  B                           SAVE LENGTH OF DATA STRING ON STACK
2343   D12E C6 20                    LDB   #SPACE                      PREPARE TO FILL DATA STRING WITH BLANKS
2344   D130 EE 22                    LDU   $02,Y                       POINT U TO FIELD STRING ADDRESS
2345                        * FILL THE FIELDED STRING WITH BLANKS
2346   D132 E7 C0           LD132    STB   ,U+                         STORE A SPACE IN FIELDED STRING
2347   D134 4A                       DECA                              DECREMENT LENGTH COUNTER
2348   D135 26 FB                    BNE   LD132                       KEEP FILLING W/SPACES IF NOT DONE
2349   D137 E6 E0                    LDB   ,S+                         *GET THE LENGTH OF THE DATA STRING AND
2350   D139 27 1F                    BEQ   LD15A                       *RETURN IF IT IS NULL (ZERO)
2351   D13B E1 A4                    CMPB  ,Y                          =COMPARE LENGTH OF DATA STRING TO LENGTH OF FIELD
2352   D13D 25 04                    BLO   LD143                       =STRING, BRANCH IF FIELD STRING > DATA STRING
2353   D13F E6 A4                    LDB   ,Y                          *GET THE LENGTH OF THE FIELD STRING AND FORCE THE
2354   D141 6F E4                    CLR   ,S                          *RSET/LSET FLAG TO LSET (0) IF DATA STRING LENGTH IS
2355                        *                                           *>= THE FIELD STRING LENGTH. THIS WILL CAUSE THE RIGHT
2356                        *                                           *SIDE OF THE DATA STRING TO BE TRUNCATED
2357   D143 EE 22           LD143    LDU   $02,Y                       LOAD U WITH THE ADDRESS OF THE FIELD STRING
2358   D145 6D E0                    TST   ,S+                         * GET THE RSET/LSET FLAG FROM THE STACK
2359   D147 27 0E                    BEQ   LD157                       * AND BRANCH IF LSET
2360                        * RSET ROUTINE
2361   D149 34 04                    PSHS  B                           SAVE THE NUMBER OF BYTES TO MOVE INTO THE FIELD STRING
2362   D14B 4F                       CLRA                              = TAKE THE 2'S COMPLEMENT OF AN UNSIGNED
2363   D14C 50                       NEGB                              = NUMBER IN ACCB - LEAVE THE DOUBLE BYTE SIGNED
2364   D14D 82 00                    SBCA  #$00                        = RESULT IN ACCD
2365   D14F EB A4                    ADDB  ,Y                          * ADD THE LENGTH OF THE FIELD STRING TO THE INVERSE
2366   D151 89 00                    ADCA  #$00                        * OF THE NUMBER OF BYTES TO BE MOVED
2367   D153 33 CB                    LEAU  D,U                         =ADD RESULT TO START OF FIELD STRING. NOW U
2368                        *                                           =WILL POINT TO (-NUMBER OF BYTES TO MOVE)
2369                        *                                           =FROM THE RIGHT SIDE OF THE FIELD STRING
2370   D155 35 04                    PULS  B                           GET THE NUMBER OF BYTES TO MOVE
2371   D157 7E A5 9A        LD157    JMP   LA59A                       MOVE ACCB BYTES FROM X TO U (DATA TO FIELD STRING)
2372   D15A 35 82           LD15A    PULS  A,PC                        PULL LSET/RSET FLAG OFF OF STACK AND RETURN
2373
2374                        * FILES COMMAND
2375   D15C BD 95 AC        FILES    JSR   L95AC                       RESET SAM DISPLAY PAGE AND VDG MODE
2376   D15F FC 09 4A                 LDD   FCBADR                      GET START OF FILE BUFFERS
2377   D162 83 09 89                 SUBD  #DFLBUF                     SUBTRACT THE START OF RANDOM FILE BUFFER SPACE
2378   D165 34 06                    PSHS  B,A                         SAVE DEFAULT VALUE OF RANDOM FILE BUFFER SPACE ON STACK
2379   D167 F6 09 5B                 LDB   FCBACT                      * GET CURRENT NUMBER OF FCBS
2380   D16A 34 04                    PSHS  B                           * AND SAVE ON THE STACK (DEFAULT VALUE)
2381   D16C 9D A5                    JSR   GETCCH                      GET CURRENT INPUT CHAR
2382   D16E 81 2C                    CMPA  #','                        CHECK FOR COMMA
2383   D170 27 0F                    BEQ   LD181                       BRANCH IF COMMA - NO BUFFER NUMBER PARAMETER GIVEN
2384   D172 BD B7 0B                 JSR   EVALEXPB                    EVALUATE EXPRESSION (BUFFER NUMBER)
2385   D175 C1 0F                    CMPB  #15                         15 FCBS MAX
2386   D177 10 22 E2 CF              LBHI  LB44A                       BRANCH IF > 15 - 'ILLEGAL FUNCTION CALL'
2387   D17B E7 E4                    STB   ,S                          SAVE NUMBER OF FCBS ON STACK
2388   D17D 9D A5                    JSR   GETCCH                      CHECK CURRENT INPUT CHAR
2389   D17F 27 08                    BEQ   LD189                       BRANCH IF END OF LINE
2390   D181 BD B2 6D        LD181    JSR   SYNCOMMA                    SYNTAX CHECK FOR COMMA
2391   D184 BD B3 E6                 JSR   LB3E6                       EVALUATE EXPRESSION, RETURN VALUE IN ACCD
2392   D187 ED 61                    STD   $01,S                       SAVE RANDOM FILE BUFFER SIZE ON STACK
2393   D189 BD CA E9        LD189    JSR   DVEC7                       CLOSE FILES
2394   D18C E6 E4                    LDB   ,S                          * GET THE NUMBER OF BUFFERS TO MAKE AND
2395   D18E 34 04                    PSHS  B                           * INITIALIZE A BUFFER COUNTER ON THE STACK
2396   D190 CC 09 89                 LDD   #DFLBUF                     GET START OF RANDOM FILE BUFFERS
2397   D193 E3 62                    ADDD  $02,S                       ADD THE NEWLY SPECIFIED RANDOM FILE BUFFER SPACE
2398   D195 25 71                    BLO   LD208                       'OUT OF MEMORY' ERROR IF > $FFFF
2399   D197 ED 62                    STD   $02,S                       SAVE START OF FCBS
2400                        * RESERVE SPACE FOR FCBS
```

```
2401   D199 C3 01 19          LD199   ADDD  #FCBLEN                        FCBLEN REQUIRED FOR EACH BUFFER
2402   D19C 25 6A                     BLO   LD208                          'OUT OF MEMORY' ERROR IF > $FFFF
2403   D19E 6A E4                     DEC   ,S                             DECREMENT BUFFER COUNTER
2404   D1A0 2A F7                     BPL   LD199                          *BRANCH IF NOT DONE - THE BPL WILL SET UP ONE MORE BUFFER
2405                          *                                           *THAN THE NUMBER REQUESTED. THIS EXTRA BUFFER IS THE SYSTEM BUFFER
2406                          *                                           *AND IS LOCATED AT THE END OF THE NORMAL FCBS. ONLY SYSTEM ROUTINES
2407                          *                                           *(COPY, BACKUP, MERGE ETC.) MAY ACCESS THIS BUFFER.
2408   D1A2 5D                        TSTB                                 AT AN EXACT 256 BYTE BOUNDARY?
2409   D1A3 27 03                     BEQ   LD1A8                          YES
2410   D1A5 4C                        INCA                                 NO - ADD 256
2411   D1A6 27 60                     BEQ   LD208                          'OUT OF MEMORY' ERROR IF PAST $FFFF
2412   D1A8 85 01             LD1A8   BITA  #$01                           ON A 512 BYTE BOUNDARY?
2413   D1AA 27 03                     BEQ   LD1AF                          YES
2414   D1AC 4C                        INCA                                 NO - ADD 256
2415   D1AD 27 59                     BEQ   LD208                          'OM' ERROR IF PAST $FFFF
2416   D1AF A7 E4             LD1AF   STA   ,S                             SAVE MS BYTE OF NEW GRAPHIC RAM START
2417   D1B1 DC 1B                     LDD   VARTAB                         GET START OF VARIABLES
2418   D1B3 90 BC                     SUBA  GRPRAM                         *SUBTRACT THE OLD GRAPHIC RAM START - ACCD CONTAINS LENGTH
2419                          *                                           *OF PROGRAM PLUS RESERVED GRAPHIC RAM
2420   D1B5 AB E4                     ADDA  ,S                             ADD IN THE AMOUNT OF RAM CALCULATED ABOVE
2421   D1B7 25 4F                     BLO   LD208                          'OUT OF MEMORY' ERROR IF > $FFFF
2422   D1B9 1F 01                     TFR   D,X                            SAVE NEW VARTAB IN X
2423   D1BB 4C                        INCA                                 *ADD 256 - TO GUARANTEE ENOUGH ROOM SINCE ALL CALCULATIONS USE
2424                          *                                           *ONLY THE MSB OF THE ADDRESS
2425   D1BC 27 4A                     BEQ   LD208                          'OUT OF MEMORY' ERROR IF PAST $FFFF
2426   D1BE 10 93 21                  CMPD  FRETOP                         IS IT GREATER THAN THE START OF STRING SPACE
2427   D1C1 24 45                     BHS   LD208                          'OUT OF MEMORY' IF > START OF STRING SPACE
2428   D1C3 4A                        DECA                                 SUBTRACT 256 - COMPENSATE FOR INCA ABOVE
2429   D1C4 93 1B                     SUBD  VARTAB                         SUBTRACT START OF VARIABLES
2430   D1C6 D3 19                     ADDD  TXTTAB                         ADD START OF BASIC
2431   D1C8 1F 02                     TFR   D,Y                            Y HAS NEW START OF BASIC
2432   D1CA A6 E4                     LDA   ,S                             * GET THE GRAPHIC RAM START, SUBTRACT
2433   D1CC 90 BC                     SUBA  GRPRAM                         * THE OLD GRAPHIC RAN START AND SAVE
2434   D1CE 1F 89                     TFR   A,B                            * THE DIFFERENCE IN ACCA AND ACCB
2435   D1D0 9B BA                     ADDA  BEGGRP                         = ADD THE OLD GRAPHIC PAGE START AND
2436   D1D2 97 BA                     STA   BEGGRP                         = STORE THE NEW START OF GRAPHICS RAM
2437   D1D4 DB B7                     ADDB  ENDGRP                         * ADD THE OLD GRAPHIC RAM END ADDRESS AND
2438   D1D6 D7 B7                     STB   ENDGRP                         * STORE THE NEW END OF GRAPHICS RAM
2439   D1D8 35 46                     PULS  A,B,U                          = ACCA=MSB OF START OF GRAPHIC RAM; ACCB=NUMBER OF FILE BUFFERS
2440                          *                                           = U=START OF FILE BUFFERS
2441   D1DA 97 BC                     STA   GRPRAM                         SAVE NEW START OF GRAPHIC RAM
2442   D1DC F7 09 5B                  STB   FCBACT                         NUMBER OF FILE BUFFERS
2443   D1DF FF 09 4A                  STU   FCBADR                         START OF FILE BUFFERS
2444   D1E2 96 68                     LDA   CURLIN                         GET CURRENT LINE NUMBER
2445   D1E4 4C                        INCA                                 ARE WE IN DIRECT MODE?
2446   D1E5 27 08                     BEQ   LD1EF                          YES - MOVE BASIC PROGRAM
2447   D1E7 1F 20                     TFR   Y,D                            MOVE NEW START OF BASIC TO ACCD
2448   D1E9 93 19                     SUBD  TXTTAB                         SUBTRACT OLD START OF BASIC
2449   D1EB D3 A6                     ADDD  CHARAD                         ADD OLD INPUT POINTER
2450   D1ED DD A6                     STD   CHARAD                         SAVE NEW INPUT POINTER
2451   D1EF DE 1B             LD1EF   LDU   VARTAB                         POINT U TO OLD START OF VARIABLES
2452   D1F1 9F 1B                     STX   VARTAB                         SAVE NEW START OF VARIBLES
2453   D1F3 11 93 1B                  CMPU  VARTAB                         * COMPARE OLD START OF VARIABLES TO NEW START OF
2454   D1F6 22 13                     BHI   LD20B                          * VARIABLES & BRANCH IF OLD > NEW
2455                          * MOVE BASIC PROGRAM IF OLD START ADDRESS <= NEW START ADDRESS
2456   D1F8 A6 C2             LD1F8   LDA   ,-U                            GET A BYTE
2457   D1FA A7 82                     STA   ,-X                            MOVE 1T
2458   D1FC 11 93 19                  CMPU  TXTTAB                         AT START OF BASIC PROGRAM?
2459   D1FF 26 F7                     BNE   LD1F8                          NO
2460   D201 10 9F 19                  STY   TXTTAB                         STORE NEW START OF BASIC PROGRAM
2461   D204 6F 3F                     CLR   -1,Y                           RESET START OF PROGRAM FLAG
2462   D206 20 13                     BRA   LD21B                          CLOSE ALL FILES
2463   D208 7E AC 44          LD208   JMP   LAC44                          'OUT OF MEMORY' ERROR
2464                          * MOVE BASIC PROGRAM IF OLD START ADDRESS > NEW START ADDRESS
2465   D20B DE 19             LD20B   LDU   TXTTAB                         POINT U TO OLD START OF BASIC
2466   D20D 10 9F 19                  STY   TXTTAB                         SAVE NEW START OF BASIC
2467   D210 6F 3F                     CLR   -1,Y                           RESET START OF BASIC FLAG
2468   D212 A6 C0             LD212   LDA   ,U+                            GET A BYTE
2469   D214 A7 A0                     STA   ,Y+                            MOVE IT
2470   D216 10 9C 1B                  CMPY  VARTAB                         AT START OF VARIABLES
2471   D219 26 F7                     BNE   LD212                          NO - MOVE ANOTHER BYTE
2472
2473                          * CLOSE ALL FCBS AND RECALCULATE FCB START ADDRESSES
2474   D21B CE 09 28          LD21B   LDU   #FCBV1                         POINT U TO FILE BUFFER POINTERS
2475   D21E BE 09 4A                  LDX   FCBADR                         POINT X TO START OF BUFFERS
2476   D221 5F                        CLRB                                 RESET FILE COUNTER
2477   D222 AF C1             LD222   STX   ,U++                           STORE FILE ADDRESS IN VECTOR TABLE
2478   D224 6F 00                     CLR   FCBTYP,X                       RESET FILE TYPE TO CLOSED
2479   D226 30 89 01 19               LEAX  FCBLEN,X                       GO TO NEXT FCB
2480   D22A 5C                        INCB                                 INCREMENT FILE COUNTER
2481   D22B F1 09 5B                  CMPB  FCBACT                         CLOSE ALL ACTIVE BUFFERS AND SYSTEM FCB
2482   D22E 23 F2                     BLS   LD222                          BRANCH IF NOT DONE
2483   D230 7E 96 CB                  JMP   L96CB                          READJUST LINE NUMBERS, ETC.
2484
2485                          * UNLOAD COMMAND
2486   D233 8D 1A             UNLOAD  BSR   LD24F                          GET DRIVE NUMBER
2487   D235 5F                        CLRB                                 CLEAR FILE COUNTER
2488   D236 5C                LD236   INCB                                 INCREMENT FILE COUNTER
2489   D237 BD C7 49                  JSR   LC749                          POINT X TO FCB
2490   D23A 27 0D                     BEQ   LD249                          BRANCH IF FILE NOT OPEN
2491   D23C A6 01                     LDA   FCBDRV,X                       CHECK DRIVE NUMBER
2492   D23E 91 EB                     CMPA  DCDRV                          DOES IT MATCH THE 'UNLOAD' DRIVE NUMBER?
2493   D240 26 07                     BNE   LD249                          NO MATCH - DO NOT CLOSE THE FILE
2494   D242 34 04                     PSHS  B                              SAVE FILE COUNTER ON THE STACK
2495   D244 BD CB 06                  JSR   LCB06                          CLOSE FCB
2496   D247 35 04                     PULS  B                              RESTORE FILE COUNTER
```

```
2497   D249 F1 Ø9 5B          LD249   CMPB  FCBACT                          CHECKED ALL FILES?
2498   D24C 23 E8                     BLS   LD236                           NO
2499   D24E 39                        RTS
2500                          * GET DRIVE NUMBER FROM BASIC - USE THE DEFAULT DRIVE IF NONE GIVEN
2501   D24F F6 Ø9 5A          LD24F   LDB   DEFDRV                          GET DEFAULT DRIVE NUMBER
2502   D252 9D A5                     JSR   GETCCH                          GET NEXT INPUT CHAR
2503   D254 27 Ø9                     BEQ   LD25F                           USE DEFAULT DRIVE NUMBER IF NONE GIVEN
2504   D256 BD B7 ØB          LD256   JSR   EVALEXPB                        EVALUATE EXPRESSION
2505   D259 C1 Ø3                     CMPB  #$Ø3                            4 DRIVES MAX
2506   D25B 1Ø 22 D3 CØ               LBHI  LA61F                           'DEVICE NUMBER ERROR' IF > 3
2507   D25F D7 EB             LD25F   STB   DCDRV                           STORE IN DSKCON VARIABLE
2508   D261 39                        RTS
2509
2510                          * BACKUP COMMAND
2511   D262 1Ø 27 D3 B9       BACKUP  LBEQ  LA61F                           DEVICE NUMBER ERROR IF NO DRIVE NUMBERS GIVEN
2512   D266 BD 95 AC                  JSR   L95AC                           RESET SAM DISPLAY PAGE AND VOG MODE
2513 > D269 BD D2 56                  JSR   LD256                           * GET SOURCE DRIVE NUMBER AND SAVE
2514   D26C F7 Ø6 FF                  STB   DBUFØ+255                       * IT AT TOP OF DBUFØ (TOP OF NEW STACK)
2515   D26F 9D A5                     JSR   GETCCH                          GET A CHARACTER FROM BASIC
2516   D271 27 Ø8                     BEQ   LD27B                           BRANCH IF END OF LINE
2517   D273 C6 A5                     LDB   #$A5                            TOKEN FOR 'TO'
2518   D275 BD B2 6F                  JSR   LB26F                           SYNTAX CHECK FOR 'TO'
2519 > D278 BD D2 56                  JSR   LD256                           GET DESTINATION DRIVE NUMBER
2520   D27B 1Ø CE Ø6 FF       LD27B   LDS   #DBUFØ+255                      PUT STACK AT TOP OF DBUFØ
2521   D27F 34 Ø4                     PSHS  B                               SAVE DESTINATION DRIVE NUMBER ON STACK
2522   D281 BD A5 C7                  JSR   LA5C7                           SYNTAX ERROR IF NOT END OF LINE
2523   D284 BD CA E9                  JSR   DVEC7                           CLOSE ALL FILES
2524   D287 6F E2                     CLR   ,-S                             CLEAR A TRACK COUNTER ON STACK
2525   D289 8E Ø9 88                  LDX   #DFLBUF-1                       POINT X TO TOP OF DISK RAM VARIABLES
2526   D28C 6C E4             LD28C   INC   ,S                              INCREMENT TRACK COUNTER
2527   D28E 3Ø 89 12 ØØ               LEAX  SECMAX*SECLEN,X                 INCREMENT X BY ONE TRACK
2528   D292 9C 27                     CMPX  MEMSIZ                          COMPARE TO TOP OF NON RESERVED RAN
2529   D294 23 F6                     BLS   LD28C                           KEEP GOING IF MORE FREE RAM LEFT
2530   D296 6A E4                     DEC   ,S                              DECREMENT TRACK COUNTER
2531   D298 1Ø 27 D9 A8               LBEQ  LAC44                           'OM' ERROR IF < 1 TRACK OF FREE RAM
2532   D29C 86 23                     LDA   #TRKMAX                         GET MAXIMUM NUMBER OF TRACKS   INITIALIZE REMAINING TRACKS CTR
2533   D29E 5F                        CLRB                                  INITIALIZE TRACKS WRITTEN COUNTER TO ZERO
2534   D29F 34 Ø6                     PSHS  B,A                             SAVE TRACKS WRITTEN AND REMAINING COUNTERS ON STACK
2535
2536                          * AT THIS POINT THE STACK HAS THE FOLLOWING DATA ON IT:
2537                          * ,S = TRACKS REMAINING COUNTER; 1,S = TRACKS WRITTEN COUNTER
2538                          * 2,S = NUMBER OF TRACKS WHICH FIT IN RAM; 3,S = DESTINATION DRIVE NUMBER
2539                          * 4,S = SOURCE DRIVE NUMBER
2540   D2A1 73 Ø9 5C                  COM   DRESFL                          SET THE DISK RESET FLAG TO CAUSE A RESET
2541   D2A4 5F                LD2A4   CLRB                                  INITIALIZE WRITE TRACK COUNTER TO ZERO
2542   D2A5 5C                LD2A5   INCB                                  ADD ONE TO WRITE TRACK COUNTER
2543   D2A6 6A E4                     DEC   ,S                              * DECREMENT REMAINING TRACKS COUNTER
2544   D2A8 27 Ø4                     BEQ   LD2AE                           * AND BRANCH IF NO TRACKS LEFT
2545   D2AA E1 62                     CMPB  $Ø2,S                           = COMPARE WRITE TRACK COUNTER TO NUMBER OF TRACKS THAT
2546   D2AC 26 F7                     BNE   LD2A5                           = WILL FIT IN RAM AND BRANCH IF ROOM FOR MORE TRACKS IN RAM
2547   D2AE D7 Ø3             LD2AE   STB   TMPLOC                          SAVE THE NUMBER OF TRACKS TO BE TRANSFERRED
2548   D2BØ E6 64                     LDB   $Ø4,S                           GET SOURCE DRIVE NUMBER
2549   D2B2 8D 48                     BSR   LD2FC                           FILL RAM BUFFER WITH TMPLOC TRACKS OF DATA
2550   D2B4 86 FF                     LDA   #$FF                            SET SOURCE/DESTINATION FLAG TO DESTINATION
2551 > D2B6 BD D3 22                  JSR   LD322                           PRINT PROMPT MESSAGE IF NEEDED
2552   D2B9 E6 63                     LDB   $Ø3,S                           GET DESTINATION DRIVE NUMBER
2553   D2BB 8D 42                     BSR   LD2FF                           WRITE TMPLOC TRACKS FROM BUFFER
2554   D2BD 6D E4                     TST   ,S                              TEST TRACKS REMAINING FLAG
2555   D2BF 27 ØC                     BEQ   LD2CD                           BRANCH IF BACKUP DONE
2556   D2C1 4F                        CLRA                                  SET SOURCE/DESTINATION FLAG TO SOURCE
2557 > D2C2 BD D3 22                  JSR   LD322                           PRINT PROMPT MESSAGE IF NEEDED
2558   D2C5 E6 61                     LDB   $Ø1,S                           * GET THE TRACKS WRITTEN COUNTER, ADD THE NUMBER OF
2559   D2C7 DB Ø3                     ADDB  TMPLOC                          * TRACKS MOVED THIS TIME THROUGH LOOP AND
2560   D2C9 E7 61                     STB   $Ø1,S                           * SAVE THE NEW TRACKS WRITTEN COUNTER
2561   D2CB 2Ø D7                     BRA   LD2A4                           COPY SOME MORE TRACKS
2562
2563   D2CD 8D Ø3             LD2CD   BSR   LD2D2                           CHECK FOR DOS INITIALIZATION
2564   D2CF 7E AC 73                  JMP   LAC73                           JUMP BACK TO BASIC S MAIN LOOP
2565
2566   D2D2 35 4Ø             LD2D2   PULS  U                               PUT THE RETURN ADDRESS IN U
2567   D2D4 B6 Ø9 5C                  LDA   DRESFL                          TEST DISK RESET FLAG
2568   D2D7 27 16                     BEQ   LD2EF                           DON T RESET THE DOS IF FLAG NOT SET
2569   D2D9 8E Ø9 28                  LDX   #FCBV1                          POINT X TO TABLE OF FCB ADDRESSES
2570   D2DC 4F                        CLRA                                  SET FILE COUNTER TO ZERO
2571   D2DD 6F 91             LD2DD   CLR   [,X++]                          MARK FCB AS CLOSED
2572   D2DF 4C                        INCA                                  ADD ONE TO FILE COUNTER
2573   D2E0 B1 Ø9 5B                  CMPA  FCBACT                          COMPARE TO NUMBER OF RESERVED FILES
2574   D2E3 23 F8                     BLS   LD2DD                           BRANCH IF ANY FILES NOT SHUT DOWN
2575   D2E5 9E 19                     LDX   TXTTAB                          LOAD X WITH THE START OF BASIC
2576   D2E7 6F 1F                     CLR   -1,X                            SET FIRST BYTE OF BASIC PROGRAM TO ZERO
2577   D2E9 BD AD 19                  JSR   LAD19                           GO DO A 'NEW'
2578   D2EC 7F Ø9 5C                  CLR   DRESFL                          RESET THE DOS RESET FLAG
2579   D2EF B6 Ø9 5D          LD2EF   LDA   DLODFL                          * CHECK THE LOAD RESET FLAG AND
2580   D2F2 27 Ø6                     BEQ   LD2FA                           * BRANCH IF NOT SET
2581   D2F4 7F Ø9 5D                  CLR   DLODFL                          CLEAR THE LOAD RESET FLAG
2582   D2F7 BD AD 19                  JSR   LAD19                           GO DO A 'NEW'
2583   D2FA 6E C4             LD2FA   JMP   ,U                              JUMP BACK TO RETURN ADDRESS SAVED IN U ABOVE
2584
2585   D2FC 86 Ø2             LD2FC   LDA   #$Ø2                            READ OP CODE
2586   D2FE 8C                        CMPX  #$86Ø3                          SKIP TWO BYTES
2587   D2FF 86 Ø3             LD2FF   LDA   #$Ø3                            WRITE OP CODE
2588   D3Ø1 DD EA                     STD   DCOPC                           SAVE IN DSKCON VARIABLE
2589   D3Ø3 A6 63                     LDA   $Ø3,S                           * GET THE NUMBER OF THE TRACK BEING CURRENTLY
2590   D3Ø5 97 EC                     STA   DCTRK                           * WRITTEN AND SAVE IT IN DSKCON VARIABLE
2591   D3Ø7 8E Ø9 89                  LDX   #DFLBUF                         = TRACK BUFFER STARTS AT DFLBUF
2592   D3ØA 9F EE                     STX   DCBPT                           = SAVE IT IN DSKCON VARIABLE
```

```
2593  D30C 96 03                LDA   TMPLOC                    GET NUMBER OF TRACKS TO MOVE
2594  D30E C6 01          LD30E LDB   #$01                      INITIALIZE SECTOR COUNTER TO ONE
2595  D310 D7 ED          LD310 STB   DSEC                      SAVE DSKCON SECTOR VARIABLE
2596  D312 BD D6 F2             JSR   LD6F2                     READ/WRITE A SECTOR
2597  D315 0C EE                INC   DCBPT                     MOVE BUFFER POINTER UP ONE SECTOR (256 BYTES)
2598  D317 5C                   INCB                            INCREMENT SECTOR COUNTER
2599  D318 C1 12                CMPB #SECMAX                    COMPARE TO MAXIMUM NUMBER OF SECTORS PER TRACK
2600  D31A 23 F4                BLS   LD310                     BRANCH IF ANY SECTORS LEFT
2601  D31C 0C EC                INC   DCTRK                     INCREMENT TRACK COUNTER VARIABLE TO NEXT TRACK
2602  D31E 4A                   DECA                            DECREMENT TRACKS TO MOVE COUNTER
2603  D31F 26 ED                BNE   LD30E                     READ MORE TRACKS IF ANY LEFT
2604  D321 39                   RTS
2605
2606  D322 E6 65          LD322 LDB  $05,S                      * GET THE DESTINATION DRIVE NUMBER AND
2607  D324 E1 66                CMPB $06,S                      * COMPARE IT TO THE SOURCE DRIVE NUMBER
2608
2609                      * PRINT SOURCE/DESTINATION DISK SWITCH PROMPT MESSAGE
2610  D326 26 36          LD326 BNE   LD35E                     RETURN IF DRIVE NUMBERS NOT EQUAL
2611  D328 7F 09 85             CLR   RDYTMR                    RESET THE READY TIMER
2612  D32B 7F FF 40             CLR   DSKREG                    CLEAR DSKREG - TURN OFF ALL DISK MOTORS
2613  D32E 7F 09 86             CLR   DRGRAM                    CLEAR DSKREG RAM IMAGE
2614  D331 34 02                PSHS  A                         SAVE SOURCE/DESTINATION FLAG ON STACK
2615  D333 BD A9 28             JSR   LA928                     CLEAR SCREEN
2616  D336 8E D3 5F             LDX   #LD35F                    POINT X TO 'INSERT SOURCE' MESSAGE
2617  D339 C6 0D                LDB   #13                       13 BYTES IN MESSAGE
2618  D33B A6 E0                LDA   ,S+                       GET SOURCE/DESTINATION FLAG FROM THE STACK
2619  D33D 27 05                BEQ   LD344                     BRANCH IF SOURCE
2620  D33F 8E D3 6C             LDX   #LD36C                    POINT X TO 'INSERT DESTINATION' MESSAGE
2621  D342 C6 12                LDB   #18                       18 BYTES IN MESSAGE
2622  D344 BD B9 A2       LD344 JSR   LB9A2                     SEND MESSAGE TO CONSOLE OUT
2623  D347 8E D3 7E             LDX   #LD37E                    POINT X TO 'DISKETTE AND' MESSAGE
2624  D34A C6 1B                LDB   #27                       27 BYTES IN MESSAGE
2625  D34C BD B9 A2             JSR   LB9A2                     SEND MESSAGE TO CONSOLE OUT
2626  D34F CC 64 05             LDD   #$6405                    * SET UP 'SOUND' PARAMETERS
2627  D352 97 8C                STA   SNDTON                    * FOR A BEEP
2628  D354 BD A9 51             JSR   LA951                     JUMP TO 'SOUND' - DO A BEEP
2629  D357 BD A1 71       LD357 JSR   LA171                     GET A CHARACTER FROM CONSOLE IN
2630  D35A 81 0D                CMPA #CR                        * KEEP LOOKING AT CONSOLE IN UNTIL
2631  D35C 26 F9                BNE   LD357                     * YOU GET A CARRIAGE RETURN
2632  D35E 39             LD45E RTS
2633
2634  D35F 49 4E 53 45 52 54 LD35F FCC  'INSERT SOURCE'
2635  D365 20 53 4F 55 52 43
2636  D36B 45
2637  D36C 49 4E 53 45 52 54 LD36C FCC  'INSERT DESTINATION'
2638  D372 20 44 45 53 54 49
2639  D378 4E 41 54 49 4F 4E
2640  D37E 20 44 49 53 4B 45 LD37E FCC  ' DISKETTE AND'
2641  D384 54 54 45 20 41 4E
2642  D38A 44
2643  D38B 0D                   FCB   CR
2644  D38C 50 52 45 53 53 20    FCC   'PRESS 'ENTER''
2645  D392 27 45 4E 54 45 52
2646  D398 27
2647
2648                      * PUSH FILENAME.EXT AND DRIVE NUMBER ONTO THE STACK
2649  D399 35 20          LD399 PULS  Y                         SAVE RETURN ADDRESS IN Y
2650  D39B C6 0B                LDB   #11                       11 CHARACTERS IN FILENAME AND EXTENSION
2651  D39D 8E 09 57             LDX   #DNAMBF+11                POINT X TO TOP OF DISK NAME/EXT BUFFER
2652  D3A0 A6 82          LD3A0 LDA   ,-X                       * GET A CHARACTER FROM FILENAME.
2653  D3A2 34 02                PSHS  A                         * EXT BUFFER AND PUSH IT ONTO THE
2654  D3A4 5A                   DECB                            * STACK - DECREMENT COUNTER AND
2655  D3A5 26 F9                BNE   LD3A0                     * KEEP LOOPING UNTIL DONE
2656  D3A7 96 EB                LDA   DCDRV                     = GET DRIVE NUMBER AND PUSH
2657  D3A9 34 02                PSHS  A                         = IT ONTO THE STACK
2658  D3AB 6E A4                JMP   ,Y                        PSEUDO - RETURN TO CALLING ROUTINE
2659
2660                      * PULL FILENAME.EXT AND DRIVE NUMBER FROM (X) TO RAM
2661  D3AD A6 80          LD3AD LDA   ,X+                       * GET DRIVE NUMBER AND SAVE
2662  D3AF 97 EB                STA   DCDRV                     * IT IN DSKCON VARIABLE
2663  D3B1 C6 0B                LDB   #11                       11 BYTES IN FILENAME AND EXTENSION
2664  D3B3 CE 09 4C             LDU   #DNAMBF                   POINT U TO DISK NAME BUFFER
2665  D3B6 7E A5 9A             JMP   LA59A                     MOVE FILENANE.EXT FROM (X) TO DNAMBF
2666
2667                      * COPY
2668                      * THE COPY PROCESS IS PERFORMED BY COPYING DATA FROM THE SOURCE FILE
2669                      * TO RAM AND THEN COPYING IT TO THE DESTINATION FILE. THE SOURCE AND
2670                      * DESTINATION FILES ARE OPENED AS RANDOM FILES AND BOTH USE THE SYSTEM
2671                      * FCB ABOVE THE RESERVED FCBS. ALL OF AVAILABLE FREE RAM ABOVE THE
2672                      * VARIABLES IS USED AS A COPY BUFFER WHICH SPEEDS UP THE COPYING PROCESS
2673                      * BUT UNFORTUNATELY THE METHOD USED WILL ALLOW AN ERROR ENCOUNTERED DURING
2674                      * THE COPY PROCESS TO 'HANG' THE SYSTEM. THIS IS CAUSED BY POINTING THE FCB'S
2675                      * RANDOM FILE BUFFER POINTER (FCBBUF,X) TO THE FREE RAM BUFFER. AN ERROR
2676                      * WILL THEN CAUSE THE OPEN FILE TO BE CLOSED WITH FCBBUF,X POINTING TO AN
2677                      * AREA IN RAM WHERE THE RANDOM FILE BUFFER CLOSE ROUTINE (LCAE2) WILL NEVER
2678                      * LOOK FOR IT
2679  D3B9 BD C9 35       COPY  JSR   LC935                     * GET SOURCE FILENAME.EXT & DRIVE NUMBER FROM BASIC
2680  D3BC 8D DB                BSR   LD399                     * AND SAVE THEM ON THE STACK
2681  D3BE 6F E2                CLR   ,-S                       CLEAR A BYTE ON STACK - SINGLE DISK COPY (SDC) FLAG
2682  D3C0 9D A5                JSR   GETCCH                    GET CURRENT INPUT CHARACTER
2683  D3C2 27 0A                BEQ   LD3CE                     BRANCH IF END OF LINE - SINGLE DISK COPY
2684  D3C4 63 E4                COM   ,S                        SET SOC FLAG TO $FF (NO SINGLE DISK COPY)
2685  D3C6 C6 A5                LDB   #$A5                      TOKEN FOR 'TO'
2686  D3C8 BD B2 6F             JSR   LB26F                     SYNTAX CHECK FOR 'TO'
2687  D3CB BD C9 35             JSR   LC935                     GET DESTINATION FILENAME.EXT AND DRIVE NUMBER
2688  D3CE 8D C9          LD3CE BSR   LD399                     SAVE DESTINATION FILENAME.EXT & DRIVE NUMBER ON STACK
```

```
2689   D3D0 BD A5 C7            JSR   LA5C7                          SYNTAX ERROR IF MORE CHARACTERS ON LINE
2690   D3D3 BD CA E9            JSR   DVEC7                          CLOSE ALL FILES
2691
2692                          * COUNT THE NUMBER OF SECTORS WORTH OF FREE RAM AVAILABLE
2693   D3D6 6F E2               CLR   ,-S                            CLEAR A SECTOR COUNTER ON THE STACK
2694   D3D8 30 E9 FF 00         LEAX  -SECLEN,S                      POINT X ONE SECTOR LENGTH DOWN FROM THE TOP OF STACK
2695   D3DC 6C E4               LD3DC INC   ,S                       INCREMENT SECTOR COUNTER
2696   D3DE 30 89 FF 00         LEAX  -SECLEN,X                      DECREMENT X BY ONE SECTOR
2697   D3E2 9C 1F               CMPX  ARYEND                         COMPARE TO TOP OF ARRAYS
2698   D3E4 24 F6               BHS   LD3DC                          BRANCH IF NOT AT BOTTOM OF FREE RAM
2699   D3E6 6A E4               DEC   ,S                             DECREMENT SECTOR COUNTER
2700   D3E8 10 27 D8 58         LBEQ  LAC44                          'OM' ERROR IF NOT AT LEAST ONE FULL SECTOR OF FREE RAM
2701   D3EC 30 6E               LEAX  14,S                           POINT X TO START OF SOURCE DATA
2702   D3EE 8D BD               BSR   LD3AD                          PUT SOURCE DATA INTO DNAMBF AND DSKCON
2703   D3F0 BD C6 8C            JSR   LC68C                          SCAN DIRECTORY FOR A MATCH
2704   D3F3 BD C6 E5            JSR   LC6E5                          'NE' ERROR IF MATCH NOT FOUND
2705   D3F6 BE 09 74            LDX   V974                           POINT X TO DIRECTORY RAM IMAGE OF FOUND FILE
2706   D3F9 EE 0E               LDU   DIRLST,X                       * GET NUMBER OF BYTES IN LAST SECTOR AND
2707   D3FB AE 0B               LDX   DIRTYP,X                       * SOURCE FILE TYPE AND ASCII FLAG
2708   D3FD 34 50               PSHS  U,X                            * AND SAVE THEM ON THE STACK
2709   D3FF BD C7 9D            JSR   LC79D                          GET VALID FAT DATA
2710   D402 F6 09 76            LDB   V976                           GET NUMBER OF FIRST GRANULE IN FILE
2711   D405 BD CD 1E            JSR   LCD1E                          * GET THE NUMBER OF GRANULES IN FILE
2712   D408 34 02               PSHS  A                              * AND SAVE IT ON THE STACK
2713   D40A 4A                  DECA                                 SUBTRACT OFF THE LAST GRANULE
2714   D40B C4 3F               ANDB  #$3F                           * MASK OFF LAST GRANULE FLAG BITS AND SAVE THE
2715   D40D 34 04               PSHS  B                              * NUMBER OF SECTORS IN LAST GRANULE ON STACK
2716   D40F 1F 89               TFR   A,B                            SAVE THE NUMBER OF GRANULES IN ACCB
2717   D411 4F                  CLRA                                 CLEAR THE MS BYTE OF ACCD
2718   D412 BD C7 79            JSR   LC779                          MULTIPLY ACCD BY NINE
2719   D415 EB E4               ADDB  ,S                             * ADD THE NUMBER OF SECTORS IN THE LAST
2720   D417 89 00               ADCA  #$00                           * GRANULE TO ACCD
2721   D419 8E 00 01            LDX   #$0001                         INITIALIZE RECORD COUNTER TO ONE
2722   D41C 34 16               PSHS  X,B,A                          INITIALIZE SECTOR AND RECORD COUNTERS ON THE STACK
2723
2724                          * AT THIS POINT THE CONTROL VARIABLES FOR COPY ARE STORED ON THE STACK.
2725                          *   0 1,S = REMAINING SECTORS COUNTER; 2 3,S = RECORD COUNTER
2726                          *     4,S = NUMBER OF SECTORS TO BE COPIED. INITIALLY SET TO NUMBER OF
2727                          *           SECTORS IN THE LAST GRANULE.
2728                          *     5,S = GRAN TEST FLAG. INITIALLY SET TO NUMBER OF GRANS IN FILE
2729                          *     6,S = FILE TYPE; 7,S = ASCII FLAG; 8 9,S = NUMBER OF BYTES IN LAST SECTOR
2730                          *    10,S = NUMBER OF SECTORS WHICH WILL FIT IN THE CURRENTLY AVAILABLE FREE RAM
2731                          * 11-22,S = DESTINATION FILENAME.EXT AND DRIVE NUMBER
2732                          *    23,S = SINGLE DISK COPY FLAG; 24-35,S = SOURCE FILENAME.EXT AND DRIVE NUMBER
2733   D41E 5F                  LD41E CLRB                           SET SECTOR COUNTER TO ZERO
2734   D41F AE E4               LD41F LDX   ,S                       GET THE NUMBER OF SECTORS REMAINING IN THE FILE
2735   D421 27 09               BEQ   LD42C                          BRANCH IF NO SECTORS LEFT
2736   D423 5C                  LD423 INCB                           ADD A SECTOR TO TEMPORARY SECTOR COUNTER
2737   D424 30 1F               LEAX  -1,X                           DECREMENT REMAINING SECTORS COUNTER
2738   D426 27 04               BEQ   LD42C                          BRANCH IF NO SECTORS LEFT
2739   D428 E1 6A               CMPB  10,S                           *COMPARE TEMPORARY COUNTER TO NUMBER OF SECTORS WHICH MAY
2740                          *                                      *BE STORED IN FREE RAM
2741   D42A 26 F7               BNE   LD423                          BRANCH IF STILL ROOM FOR MORE SECTORS
2742   D42C AF E4               LD42C STX   ,S                       SAVE THE NUMBER OF UNCOPIED SECTORS REMAINING IN THE FILE
2743   D42E E7 64               STB   $04,S                          SAVE THE NUMBER OF SECTORS TO BE COPIED THIS TIME THROUGH LOOP
2744   D430 8D 50               BSR   LD482                          'GET' ACCB SECTORS TO RAM BUFFER
2745   D432 86 FF               LDA   #$FF                           SET SOURCE/DESTINATION FLAG TO DESTINATION
2746   D434 8D 40               BSR   LD476                          PRINT PROMPT MESSAGE IF REQUIRED
2747   D436 6D 65               TST   $05,S                          * CHECK THE GRAN TEST FLAG. IF <> 0, IT CONTAINS THE
2748   D438 27 25               BEQ   LD45F                          * NUMBER OF GRANS IN THE FILE AND THE DESTINATION DISK
2749                          *                                      * MUST BE CHECKED FOR ENOUGH ROOM. IF IT IS =0
2750                          *                                      * THEN THE CHECK HAS ALREADY BEEN DONE
2751   D43A 30 6B               LEAX  11,S                           POINT TO DESTINATION FILE PARAMETERS
2752   D43C BD D3 AD            JSR   LD3AD                          GET DESTINATION FILE PARAMETERS FROM STACK
2753   D43F BD D0 59            JSR   LD059                          SCAN DIRECTORY FOR FILE - 'AE' ERROR IF IT EXISTS
2754   D442 BD C7 9D            JSR   LC79D                          GET VALID FAT DATA
2755
2756                          * MAKE SURE THERE ARE ENOUGH FREE GRANULES ON THE DESTINATION DISK
2757   D445 BD C7 55            JSR   LC755                          POINT X TO FAT
2758   D448 30 06               LEAX  FATCON,X                       SKIP PAST THE FAT CONTROL BYTES
2759   D44A A6 65               LDA   $05,S                          GET THE NUMBER OF GRANS IN THE FILE
2760   D44C C6 44               LDB   #GRANMX                        SET GRAN COUNTER TO MAXIMUM
2761   D44E 63 84               LD44E COM   ,X                       * CHECK TO SEE IF A BRAN IS FREE
2762   D450 26 03               BNE   LD455                          * AND BRANCH IF IT IS NOT FREE
2763   D452 4A                  DECA                                 = DECREMENT COUNTER AND BRANCH IF
2764   D453 27 08               BEQ   LD45D                          = THERE ARE ENOUGH FREE GRANULES
2765   D455 63 80               LD455 COM   ,X+                      RESTORE FAT BYTE AND INCREMENT POINTER
2766   D457 5A                  DECB                                 DECREMENT GRAN COUNTER
2767   D458 26 F4               BNE   LD44E                          BRANCH IF ALL GRANS NOT CHECKED
2768   D45A 7E C7 F8            JMP   LC7F8                          'DISK FULL' ERROR
2769   D45D 63 84               LD45D COM   ,X                       RESTORE FAT BYTE
2770   D45F 8D 1B               LD45F BSR   LD47C                    'PUT' DATA FROM RAM BUFFER TO DESTINATION FILE
2771   D461 AE E4               LDX   ,S                             GET THE NUMBER OF REMAINING SECTORS
2772   D463 27 0D               BEQ   LD472                          EXIT ROUTINE IF NO SECTORS LEFT
2773   D465 EC 62               LDD   $02,S                          *
2774   D467 EB 64               ADDB  $04,S                          * GET THE CURRENT RECORD COUNTER, ADD
2775   D469 89 00               ADCA  #$00                           * THE NUMBER OF SECTORS (RECORDS) MOVED
2776   D46B ED 62               STD   $02,S                          * AND SAVE THE NEW RECORD COUNTER
2777   D46D 4F                  CLRA                                 SET SOURCE/DESTINATION FLAG TO SOURCE
2778   D46E 8D 06               BSR   LD476                          PRINT PROMPT MESSAGE IF REQUIRED
2779   D470 20 AC               BRA   LD41E                          KEEP COPYING SECTORS
2780
2781   D472 32 E8 24            LD472 LEAS  36,S                     REMOVE TEMPORARY STORAGE VARIABLES FROM STACK
2782   D475 39                  RTS                                  **** COPY DONE ****
2783
2784   D476 6D E8 19            LD476 TST   25,S                     *CHECK SINGLE DISK COPY FLAG - IF <> ZERO, THEN DON'T
```

```
2785                            *                                       *PRINT THE PROMPT MESSAGE
2786   D479 7E D3 26                    JMP   LD326                     PRINT THE PROMPT MESSAGE IF REQUIRED
2787
2788                            * 'PUT'.'GET' DATA FROM THE DESTINATION/SOURCE FILES
2789   D47C 86 FF               LD47C   LDA   #$FF                      'PUT' FLAG
2790   D47E 30 6D                       LEAX  13,S                      POINT X TO DESTINATION FILENAME DATA
2791   D480 20 04                       BRA   LD486                     GO 'PUT' SOME DATA
2792   D482 4F                  LD482   CLRA                            ZERO IS THE 'GET' FLAG
2793   D483 30 E8 1A                    LEAX  26,S                      POINT X TO THE SOURCE FILENAME DATA
2794   D486 97 D8               LD486   STA   VD8                       SAVE THE 'GET'/'PUT' FLAG
2795   D488 BD D3 AD                    JSR   LD3AD                     GET FILENAME AND DRIVE DATA FROM THE STACK
2796   D48B AE 68                       LDX   $08,S                     * GET ASCII FLAG AND FILE TYPE AND SAVE
2797   D48D BF 09 57                    STX   DFLTYP                    * THEM IN THE DISK RAM VARIABLES
2798   D490 8E 01 00                    LDX   #SECLEN                   = SAVE ONE SECTOR LENGTH IN
2799   D493 BF 09 7C                    STX   DFFLEN                    = RAM RECORD LENGTH VARIABLE
2800   D496 86 52                       LDA   #'R'                      RANDOM FILE TYPE FLAG
2801   D498 F6 09 5B                    LDB   FCBACT                    * GET THE HIGHEST RESERVED FCB NUMBER, ADD ONE
2802   D49B 5C                          INCB                            * AND OPEN A RANDOM FILE WHOSE FCB WILL BE ONE ABOVE
2803   D49C BD C4 8D                    JSR   LC48D                     * THE HIGHEST RESERVED FCB (THE SYSTEM FCB)
2804   D49F 9E F1                       LDX   FCBTMP                    POINT X TO THE 'SYSTEM' FCB
2805   D4A1 CC 01 00                    LDD   #SECLEN                   * SET THE NUMBER OF BYTES IN THE LAST SECTOR
2806   D4A4 ED 88 13                    STD   FCBLST,X                  * OF THE FILE EQUAL TO ONE SECTOR LENGTH
2807   D4A7 E6 66                       LDB   $06,S                     =GET THE NUMBER OF SECTORS TO MOVE AND
2808   D4A9 27 29                       BEQ   LD4D4                     =BRANCH IF NONE LEFT
2809   D4AB D6 D8                       LDB   VD8                       *GRAB THE 'GET'/'PUT' FLAG, 'AND' IT WITH THE
2810   D4AD E4 67                       ANDB  $07,S                     *GRAN TEST FLAG - BRANCH IF 'GET'ING DATA OR THIS IS
2811   D4AF 27 09                       BEQ   LD4BA                     *NOT THE FIRST TIME THROUGH THE LOOP
2812   D4B1 EC 62                       LDD   $02,S                     =GET THE NUMBER OF SECTORS REMAINING TO BE COPIED AND
2813   D4B3 EB 66                       ADDB  $06,S                     =ADD THE NUMBER TO BE COPIED THIS TIME THROUGH LOOP
2814   D4B5 89 00                       ADCA  #$00                      =
2815   D4B7 BD C2 E6                    JSR   LC2E6                     *'PUT' THE LAST RECORD IN THE FILE TO THE SYSTEM FCB.
2816                            *                                       *THE RECORD NUMBER IS IN ACCD.
2817   D4BA 9E F1               LD4BA   LDX   FCBTMP                    POINT X TO THE SYSTEM FCB
2818   D4BC EE 64                       LDU   $04,S                     * GET THE CURRENT RECORD NUMBER
2819   D4BE EF 07                       STU   FCBREC,X                  * AND SAVE IT IN THE FCB
2820   D4C0 E6 66                       LDB   $06,S                     GET THE NUMBER OF THE RECORD (SECTOR) TO MOVE
2821   D4C2 DE 1F                       LDU   ARYEND                    END OF ARRAYS IS THE START OF THE COPY FREE RAM BUFFER
2822   D4C4 34 44               LD4C4   PSHS  U,B                       SAVE SECTOR COUNTER AND BUFFER POINTER ON THE STACK
2823   D4C6 9E F1                       LDX   FCBTMP                    POINT X TO SYSTEM FCB
2824   D4C8 EF 0B                       STU   FCBBUF,X                  *SET THE RANDOM FILE BUFFER POINTER TO THE 'COPY' RAM BUFFER
2825                            *                                       *THIS WILL CAUSE THE SYSTEM TO 'HANG' IF AN ERROR OCCURS DURING COPY.
2826   D4CA BD C2 EA                    JSR   LC2EA                     GO 'GET' OR 'PUT' DATA TO THE SYSTEM FCB
2827   D4CD 6C 61                       INC   $01,S                     ADD 256 (ONE SECTOR) TO THE BUFFER POINTER
2828   D4CF 35 44                       PULS  B,U                       GET THE SECTOR COUNTER AND BUFFER POINER
2829   D4D1 5A                          DECB                            DECREMENT SECTOR COUNTER
2830   D4D2 26 F0                       BNE   LD4C4                     BRANCH IF ALL SECTORS NOT DONE
2831   D4D4 9E F1               LD4D4   LDX   FCBTMP                    POINT X TO SYSTEM FCB
2832   D4D6 CE 09 89                    LDU   #DFLBUF                   * RESET THE RANDOM FILE BUFFER POINTER FOR THE SYSTEM
2833   D4D9 EF 0B                       STU   FCBBUF,X                  * FCB TO THE BOTTOM OF RANDOM FILE BUFFER AREA
2834   D4DB D6 D8                       LDB   VD8                       =GRAB THE 'GET'/'PUT' FLAG, 'AND' IT WITH THE GRAN
2835   D4DD E4 67                       ANDB  $07,S                     =TEST FLAG - CLOSE THE FILE IF 'GET'ING DATA AND
2836   D4DF 27 09                       BEQ   LD4EA                     =THIS IS NOT THE FIRST TIME THROUGH THE LOOP
2837   D4E1 6F 67                       CLR   $07,S                     RESET THE GRAN TEST FLAG IF FIRST TIME THROUGH LOOP
2838   D4E3 EC 6A                       LDD   10,S                      *GET THE NUMBER OF BYTES IN THE LAST SECTOR,
2839   D4E5 8A 80                       ORA   #$80                      *'OR' IN THE PRE-SAVED FLAG AND
2840   D4E7 ED 88 13                    STD   FCBLST,X                  *SAVE THE NUMBER OF BYTES IN THE LAST SECTOR IN THE FCB
2841   D4EA 7E CB 06            LD4EA   JMP   LCB06                     CLOSE THE FILE
2842
2843                            * DSKI$ COMMAND
2844   D4ED 8D 38               DSKI    BSR   LD527                     GET THE DRIVE, TRACK AND SECTOR NUMBERS
2845   D4EF 8D 2B                       BSR   LD51C                     * EVALUATE STRING VARIABLE 1 AND SAVE
2846   D4F1 34 10                       PSHS  X                         * THE DESCRIPTOR ADDRESS ON THE STACK
2847   D4F3 8D 27                       BSR   LD51C                     = EVALUATE STRING VARIABLE 2 AND SAVE
2848   D4F5 34 10                       PSHS  X                         = THE DESCRiPTOR ADDRESS ON THE STACK
2849   D4F7 C6 02                       LDB   #$02                      DSKCON READ OP CODE
2850   D4F9 BD D5 8F                    JSR   LD58F                     REAO A SECTOR INTO DBUF0
2851   D4FC CE 06 80                    LDU   #DBUF0+128                POINT U TO TOP HALF OF DBUF0
2852   D4FF 35 10                       PULS  X                         GET STRING 2 DESCRIPTOR ADDRESS
2853   D501 8D 05                       BSR   LD508                     PUT STRING 2 INTO STRING SPACE
2854   D503 CE 06 00                    LDU   #DBUF0                    POINT U TO BOTTOM HALF OF DBUF0
2855   D506 35 10                       PULS  X                         GET STRING 1 DESCRIPTOR ADDRESS
2856   D508 34 50               LD508   PSHS  U,X                       PUT STRING DESCRIPTOR & SOURCE POINTER ON THE STACK
2857   D50A C6 80                       LDB   #128                      *
2858   D50C BD B5 0F                    JSR   LB50F                     * RESERVE 128 BYTES IN STRING SPACE
2859   D50F 33 84                       LEAU  ,X                        POINT U TO RESERVED STRING SPACE
2860   D511 35 10                       PULS  X                         GET STRING DESCRIPTOR ADDRESS
2861   D513 E7 84                       STB   ,X                        * SAVE DESCRIPTOR DATA (LENGTH AND ADDRESS)
2862   D515 EF 02                       STU   $02,X                     * OF THE NEW STRING
2863   D517 35 10                       PULS  X                         GET THE SOURCE (DBUF0) POINTER
2864   D519 7E A5 9A            LD519   JMP   LA59A                     MOVE SECTOR DATA FROM DBUF0 TO STRING SPACE
2865
2866   D51C BD B2 6D            LD51C   JSR   SYNCOMMA                  SYNTAX CHECK FOR A COMMA
2867   D51F 8E B3 57                    LDX   #LB357                    POINT X TO EVALUATE VARIABLE ROUTINE
2868   D522 8D 2F                       BSR   LD553                     EVALUATE A VARIABLE
2869   D524 7E B1 46            LD524   JMP   LB146                     'TM' ERROR IF NUMERIC VARIABLE
2870
2871                            * EVALUATE DRIVE, TRACK AND SECTOR NUMBERS
2872   D527 BD B7 0B            LD527   JSR   EVALEXPB                  EVALUATE EXPRESSION, RETURN VALUE IN ACCB
2873   D52A C1 03                       CMPB  #$03                      * COMPARE TO 3 (HIGHEST DRIVE NUMBER) -
2874   D52C 22 1C                       BHI   LD54A                     * 'FC' ERROR IF IT S > 3
2875   D52E 34 04                       PSHS  B                         SAVE DRIVE NUMBER ON THE STACK
2876   D530 BD B7 38                    JSR   LB738                     SYNTAX CHECK FOR COMMA. EVALUATE EXPRESSION (TRACK NUMBER)
2877   D533 C1 22                       CMPB  #TRKMAX-1                 * CHECK FOR MAXIMUM TRACK NUMBER
2878   D535 22 13                       BHI   LD54A                     * 'FC' ERROR IF TRACK NUMBER > 34
2879   D537 34 04                       PSHS  B                         SAVE TRACK NUMBER ON THE STACK
2880   D539 BD B7 38                    JSR   LB738                     SYNTAX CHECK FOR COMMA, EVALUATE EXPRESSION (SECTOR NUMBER)
```

```
2881   D53C D7 ED                    STB   DSEC                      SAVE SECTOR NUMBER IN DSKCON VARIABLE
2882   D53E 5A                       DECB                            *USELESS INSTRUCTION. NEXT INSTRUCTION SHOULD JUST
2883   D53F C1 11                    CMPB  #SECMAX-1                 *CHECK FOR MAXIMUM SECTOR NUMBER (SECMAX)
2884   D541 22 Ø7                    BHI   LD54A                     'FC' ERROR IF SECTOR NUMBER TOO BIG
2885   D543 35 Ø6                    PULS  A,B                       * GET TRACK AND DRIVE NUMBER OFF OF
2886   D545 97 EC                    STA   DCTRK                     * THE STACK AND SAVE IN DSKCON
2887   D547 D7 EB                    STB   DCDRV                     * VARIABLES
2888   D549 39                       RTS
2889   D54A 7E B4 4A          LD54A  JMP   LB44A                     JUMP TO 'FC' ERROR
2890
2891   D54D BD B2 6D          LD54D  JSR   SYNCOMMA                  SYNTAX CHECK FOR COMMA
2892   D55Ø 8E B1 56                 LDX   #LB156                    POINT X TO 'EVALUATE EXPRESSION' ROUTINE ADDRESS
2893   D553 D6 EB             LD553  LDB   DCDRV                     * GET THE DSKCON DRIVE, TRACK AND
2894   D555 DE EC                    LDU   DCTRK                     * SECTOR VALUES AND SAVE THEM ON THE STACK
2895   D557 34 44                    PSHS  S,B                       *
2896   D559 AD 84                    JSR   ,X                        GO EVALUATE AN EXPRESSION OR A VARIABLE
2897   D55B 35 44                    PULS  B,S                       * GET THE DRIVE, TRACK AND SECTOR
2898   D55D D7 EB                    STB   DCDRV                     * NUMBERS OFF OF THE STACK AND PUT
2899   D55F DF EC                    STU   DCTRK                     * THEM BACK INTO THE DSKCON VARIABLES
2900   D561 39                       RTS
2901
2902                          * DSKO$ COMMAND
2903   D562 8D C3             DSKO   BSR   LD527                     GET THE DRIVE, TRACK AND SECTOR NUMBERS
2904   D564 8D E7                    BSR   LD54D                     GET THE DESCRIPTOR OF STRING 1
2905   D566 8D BC                    BSR   LD524                     'TM' ERROR IF NUMERIC EXPRESSION
2906   D568 9E 52                    LDX   FPA0+2                    * GET STRING 1 DESCRIPTOR ADDRESS
2907   D56A 34 1Ø                    PSHS  X                         * AND SAVE IT ON THE STACK
2908   D56C 8D DF                    BSR   LD54D                     GET THE DESCRIPTOR OF STRING 2
2909   D56E BD B6 54                 JSR   LB654                     *GET LENGTH AND ADDRESS OF STRING 2 AND
2910   D571 34 14                    PSHS  X,B                       *SAVE THEM ON THE STACK
2911   D573 5F                       CLRB                            SET CLEAR COUNTER TO 256 (FULL SECTOR BUFFER)
2912   D574 8E Ø6 ØØ                LDX   #DBUFØ                    USE DBUFØ AS THE DSKO$ I/O BUFFER
2913   D577 6F 8Ø             LD577  CLR   ,X+                       CLEAR A BYTE IN I/O BUFFER
2914   D579 5A                       DECB                            DECREMENT CLEAR COUNTER
2915   D57A 26 FB                    BNE   LD577                     BRANCH IF ALL 256 BYTES NOT CLEARED
2916   D57C 35 14                    PULS  B,X                       GET THE LENGTH AND ADDRESS OF STRING 2
2917   D57E CE Ø6 8Ø                 LDU   #DBUFØ+128                POINT X TO STRING 2 DESTINATION
2918   D581 8D 96                    BSR   LD519                     MOVE STRING 2 DATA INTO DBUFØ
2919   D583 35 1Ø                    PULS  X                         POINT X TO STRING 1 DESCRIPTOR
2920   D585 BD B6 59                 JSR   LB659                     GET THE LENGTH AND ADDRESS OF STRING 1
2921   D588 CE Ø6 ØØ                LDU   #DBUFØ                    POINT U TO STRING 1 DESTINATION
2922   D58B 8D 8C                    BSR   LD519                     MOVE STRING 1 DATA INTO DBUFØ
2923   D58D C6 Ø3                    LDB   #$Ø3                      DSKCON WRITE OP CODE
2924   D58F 8E Ø6 ØØ                LDX   #DBUFØ                    POINT X TO I/O BUFFER (DBUFØ)
2925   D592 9F EE                    STX   DCBPT                     *
2926   D594 D7 EA                    STB   DCOPC                     * SAVE NEW DSKCON BUFFER POINTER AND OP CODE VARIABLES
2927   D596 7E D6 F2                 JMP   LD6F2                     GO WRITE OUT A SECTOR
2928
2929                          * DSKINI COMMAND
2930   D599 1Ø 27 DØ 82       DSKINI LBEQ  LA61F                     BRANCH TO 'DN' ERROR IF NO DRIVE NUMBER SPECIFIED
2931   D59D BD D2 56                 JSR   LD256                     CALCULATE DRIVE NUMBER
2932   D5AØ C6 Ø4                    LDB   #$Ø4                      SKIP FACTOR DEFAULT VALUE
2933   D5A2 9D A5                    JSR   GETCCH                    GET CURRENT INPUT CHAR FROM BASIC
2934   D5A4 27 ØC                    BEQ   LD5B2                     BRANCH IF END OF LINE
2935   D5A6 BD B7 38                 JSR   LB738                     SYNTAX CHECK FOR COMMA AND EVALUATE EXPRESSION
2936   D5A9 C1 11                    CMPB  #17                       MAX VALUE OF SKIP FACTOR = 16
2937   D5AB 1Ø 24 DE 9B              LBHS  LB44A                     'ILLEGAL FUNCTION CALL' IF BAD SKIP FACTOR
2938   D5AF BD A5 C7                 JSR   LA5C7                     SYNTAX ERROR IF MORE CHARACTERS ON THE LINE
2939   D5B2 34 Ø4             LD5B2  PSHS  B                         SAVE SKIP FACTOR ON THE STACK
2940   D5B4 8E Ø7 12                 LDX   #DBUF1+SECMAX             POINT TO END OF LOGICAL SECTOR NUMBER STORAGE AREA
2941   D5B7 C6 12                    LDB   #SECMAX                   18 SECTORS PER TRACK
2942   D5B9 6F 82             LD5B9  CLR   ,-X                       CLEAR A BYTE IN THE BUFFER
2943   D5BB 5A                       DECB                            CLEARED ALL 18?
2944   D5BC 26 FB                    BNE   LD5B9                     KEEP GOING IF NOT
2945   D5BE 4F                       CLRA                            RESET PHYSICAL SECTOR COUNTER
2946   D5BF 2Ø ØD                    BRA   LD5CE                     START WITH FIRST PHYSICAL SECTOR = 1
2947
2948                          * CALCULATE LOGICAL SECTOR NUMBERS
2949   D5C1 EB E4             LD5C1  ADDB  ,S                        ADD SKIP FACTOR TO LOGICAL SECTOR COUNTER
2950   D5C3 5C                LD5C3  INCB                            ADD ONE TO LOGICAL SECTOR COUNTER
2951   D5C4 C0 12             LD5C4  SUBB  #SECMAX                   SUBTRACT MAX NUMBER OF SECTORS
2952   D5C6 24 FC                    BHS   LD5C4                     BRANCH UNTIL Ø > ACCB >= -18
2953   D5C8 CB 12                    ADDB  #SECMAX                   ADD 18, NOW ACCB IS Ø-17
2954   D5CA 6D 85                    TST   B,X                       IS ANYTHING STORED HERE ALREADY?
2955   D5CC 26 F5                    BNE   LD5C3                     YES - GET ANOTHER SECTOR
2956   D5CE 4C                LD5CE  INCA                            * INCREMENT PHYSICAL SECTOR NUMBER AND
2957   D5CF A7 85                    STA   B,X                       * SAVE IT IN THE RAM BUFFER
2958   D5D1 81 12                    CMPA  #SECMAX                   FINISHED WITH ALL SECTORS?
2959   D5D3 25 EC                    BLO   LD5C1                     NO - KEEP GOING
2960   D5D5 32 61                    LEAS  $Ø1,S                     REMOVE SKIP FACTOR FROM STACK
2961   D5D7 8E 22 ØF                 LDX   #DFLBUF+$1888-2           GET TOP OF RAM USED BY DSKINI
2962   D5DA 9C 27                    CMPX  MEMSIZ                    IS IT > CLEARED AREA?
2963   D5DC 1Ø 22 D6 64              LBHI  LAC44                     'OUT OF MEMORY' ERROR IF > CLEARED AREA
2964   D5EØ BD CA E9                 JSR   DVEC7                     CLOSE ALL FILES
2965   D5E3 73 Ø9 5C                 COM   DRESFL                    SET RESET FLAG TO $FF - THIS WILL CAUSE A DOS RESET
2966   D5E6 1Ø CE Ø8 ØØ              LDS   #DBUF1+SECLEN             SET STACK TO TOP OF DBUF1
2967   D5EA BD 95 AC                 JSR   L95AC                     RESET SAM TO DISPLAY PAGE ZERO AND ALPHA GRAPHICS
2968   D5ED 86 ØØ                    LDA   #$ØØ                      YOU COULD DELETE THIS INSTRUCTION AND CHANGE FOLLOWING STA TO CLR
2969   D5EF 97 EA                    STA   DCOPC                     RESTORE HEAD TO TRACK ZERO DSKCON OP CODE
2970   D5F1 ØF EC                    CLR   DCTRK                     SET DSKCON TRACK VARIABLE TO TRACK ZERO
2971   D5F3 BD D6 F2                 JSR   LD6F2                     RESTORE HEAD TO TRACK ZERO
2972   D5F6 7F Ø9 85                 CLR   RDYTMR                    RESET THE READY TIMER
2973   D5F9 86 C0                    LDA   #$C0                      * FOC READ ADDRESS CODE
2974   D5FB B7 FF 48                 STA   FDCREG                    *
2975   D5FE BD D7 D1                 JSR   LD7D1                     CHECK DRIVE READY - WAIT UNTIL READY
2976   D601 27 1D                    BEQ   LD62Ø                     BRANCH IF DRIVES READY
```

```
2977   D603 7E D6 88                JMP   LD688                      ERROR IF DRIVES NOT READY
2978   D606 81 16           LD606   CMPA  #22                        = CHECK FOR TRACK 22 (PRECOMPENSATION)
2979   D608 25 08                   BLO   LD612                      = AND BRANCH IF < TRACK 22 - NO PRECOMP
2980   D60A B6 09 86                LDA   DRGRAM                     * GET THE RAM IMAGE OF DSKREG, 'OR'
2981   D60D 8A 10                   ORA   #$10                       * IN THE PRECOMPENSATION FLAG AND
2982   D60F B7 FF 40                STA   DSKREG                     * SEND IT TO DSKREG
2983   D612 86 53           LD612   LDA   #$53                       = GET STEP IN COMMAND
2984   D614 B7 FF 48                STA   FDCREG                     = AND SEND IT TO THE 1793
2985   D617 1E 88                   EXG   A,A                        * DELAY AFTER ISSUING COMMAND TO 1793
2986   D619 1E 88                   EXG   A,A                        *
2987   D61B BD D7 D1                JSR   LD7D1                      CHECK DRIVE READY
2988   D61E 26 68                   BNE   LD688                      BRANCH IF NOT READY - ISSUE AN ERROR
2989   D620 BD D7 F0        LD620   JSR   LD7F0                      WAIT A WHILE
2990   D623 8D 6C                   BSR   LD691                      BUILD A FORMATTED TRACK IN RAM
2991   D625 10 8E FF 4B             LDY   #FDCREG+3                  Y POINTS TO 1793 DATA REGISTER
2992   D629 1A 50                   ORCC  #$50                       DISABLE INTERRUPTS
2993   D62B 8E D6 4F                LDX   #LD64F                     * GET RETURN ADDRESS AND STORE
2994   D62E BF 09 83                STX   DNMIVC                     * IT IN THE NON MASKABLE INTERRUPT VECTOR
2995   D631 8E 09 89                LDX   #DFLBUF                    POINT X TO THE FORMATTED TRACK RAM IMAGE
2996   D634 B6 FF 48                LDA   FDCREG                     RESET STATUS OF THE 1793
2997   D637 86 FF                   LDA   #$FF                       * ENABLE THE NMI FLAG TO VECTOR
2998   D639 B7 09 82                STA   NMIFLG                     * OUT OF AN I/O LOOP UPON AN NMI INTERRUPT
2999   D63C C6 F4                   LDB   #$F4                       = GET WRITE TRACK COMMAND AND
3000   D63E F7 FF 48                STB   FDCREG                     = SEND TO 1793
3001   D641 B6 09 86                LDA   DRGRAM                     * GET THE DSKREG RAM IMAGE AND 'OR' IN THE
3002   D644 8A 80                   ORA   #$80                       * FLAG WHICH WILL ENABLE THE 1793 TO HALT
3003   D646 B7 FF 40                STA   DSKREG                     * THE 6809. SEND RESULT TO DSKREG
3004   D649 E6 80           LD649   LDB   ,X+                        = GET A BYTE FROM THE FORMATTED TRACK
3005   D64B E7 A4                   STB   ,Y                         = RAM IMAGE, SEND IT TO THE 1793 AND
3006   D64D 20 FA                   BRA   LD649                      = LOOP BACK TO GET ANOTHER BYTE
3007
3008   D64F B6 FF 48        LD64F   LDA   FDCREG                     GET STATUS
3009   D652 1C AF                   ANDCC #$AF                       ENABLE INTERRUPTS
3010   D654 84 44                   ANDA  #$44                       * KEEP ONLY WRITE PROTECT & LOST DATA
3011   D656 97 F0                   STA   DCSTA                      * AND SAVE IT IN THE DSKCON STATUS BYTE
3012   D658 26 2E                   BNE   LD688                      BRANCH IF ERROR
3013   D65A 0C EC                   INC   DCTRK                      SKIP TO THE NEXT TRACK
3014   D65C 96 EC                   LDA   DCTRK                      GET THE TRACK NUMBER
3015   D65E 81 23                   CMPA  #TRKMAX                    WAS IT THE LAST TRACK
3016   D660 26 A4                   BNE   LD606                      NO - KEEP GOING
3017
3018                        * VERIFY THAT ALL SECTORS ARE READABLE
3019   D662 86 02                   LDA   #$02                       = GET THE DSKCON READ OP CODE
3020   D664 97 EA                   STA   DCOPC                      = AND SAVE IT IN THE DSKCON VARIABLE
3021   D666 8E 06 00                LDX   #DBUF0                     * POINT THE DSKCON BUFFER POINTER
3022   D669 9F EE                   STX   DCBPT                      * TO DBUF0
3023   D66B CE 07 00                LDU   #DBUF1                     POINT U TO THE LOGICAL SECTOR NUMBERS
3024   D66E 4F                      CLRA                             RESET THE TRACK COUNTER TO ZERO
3025   D66F 97 EC           LD66F   STA   DCTRK                      SET THE DSKCON TRACK VARIABLE
3026   D671 5F                      CLRB                             RESET THE SECTOR COUNTER
3027   D672 A6 C5           LD672   LDA   B,U                        GET THE PHYSICAL SECTOR NUMBER
3028   D674 97 ED                   STA   DSEC                       SAVE DSKCON SECTOR VARIABLE
3029   D676 BD D6 F2                JSR   LD6F2                      READ A SECTOR
3030   D679 5C                      INCB                             * INCREMENT THE SECTOR COUNTER
3031   D67A C1 12                   CMPB  #SECMAX                    * AND COMPARE IT TO MAXIMUM SECTOR NUMBER
3032   D67C 25 F4                   BLO   LD672                      * AND KEEP LOOPING IF MORE SECTORS LEFT
3033   D67E 96 EC                   LDA   DCTRK                      = GET THE CURRENT TRACK NUMBER
3034   D680 4C                      INCA                             = ADD ONE TO IT, COMPARE TO THE MAXIMUM TRACK
3035   D681 81 23                   CMPA  #TRKMAX                    = NUMBER AND KEEP LOOPING IF
3036   D683 25 EA                   BLO   LD66F                      = THERE ARE STILL TRACKS TO DO
3037   D685 7E D2 CD                JMP   LD2CD                      GO CHECK FOR A DOS RESET
3038   D688 7F 09 86        LD688   CLR   DRGRAM                     CLEAR RAM IMAGE OF DSKREG
3039   D68B 7F FF 40                CLR   DSKREG                     CLEAR DSKREG - TURN DISK MOTORS OFF
3040 > D68E 7E D7 01                JMP   LD701                      PROCESS DRIVES NOT READY ERROR
3041
3042                        * BUILD A FORMATTED TRACK OF DATA IN RAM STARTING AT DFLBUF.
3043
3044   D691 8E 09 89        LD691   LDX   #DFLBUF                    START TRACK BUFFER AT DFLBUF
3045   D694 CC 20 4E                LDD   #$204E                     GET SET TO WRITE 32 BYTES OF $4E
3046   D697 8D 29                   BSR   LD6C2                      GO WRITE GAP IV
3047   D699 5F                      CLRB                             RESET SECTOR COUNTER
3048   D69A 34 04           LD69A   PSHS  B                          SAVE SECTOR COUNTER
3049   D69C CE 07 00                LDU   #DBUF1                     POINT U TO THE TABLE OF LOGICAL SECTORS
3050   D69F E6 C5                   LDB   B,U                        * GET LOGICAL SECTOR NUMBER FROM TABLE AND
3051   D6A1 D7 ED                   STB   DSEC                       * SAVE IT IN THE DSKCON VARIABLE
3052   D6A3 CE D6 D4                LDU   #LD6D4                     POINT U TO TABLE OF SECTOR FORMATTING DATA
3053   D6A6 C6 03                   LDB   #$03                       * GET FIRST 3 DATA BLOCKS AND
3054   D6A8 8D 1E                   BSR   LD6C8                      * WRITE THEM TO BUFFER
3055   D6AA 96 EC                   LDA   DCTRK                      = GET TRACK NUMBER AND STORE 1T
3056   D6AC A7 80                   STA   ,X+                        = IN THE RAM BUFFER
3057   D6AE 6F 80                   CLR   ,X+                        CLEAR A BYTE (SIDE NUMBER) IN BUFFER
3058   D6B0 96 ED                   LDA   DSEC                       * GET SECTOR NUMBER AND
3059   D6B2 A7 80                   STA   ,X+                        * STORE IT IN THE BUFFER
3060   D6B4 C6 09                   LDB   #$09                       = GET THE LAST NINE DATA BLOCKS AND
3061   D6B6 8D 10                   BSR   LD6C8                      = WRITE THEM TO THE BUFFER
3062   D6B8 35 04                   PULS  B                          GET SECTOR COUNTER
3063   D6BA 5C                      INCB                             NEXT SECTOR
3064   D6BB C1 12                   CMPB  #SECMAX                    18 SECTORS PER TRACK
3065   D6BD 25 DB                   BLO   LD69A                      BRANCH IF ALL SECTORS NOT DONE
3066   D6BF CC C8 4E                LDD   #$C84E                     WRITE 200 BYTES OF $4E AT END OF TRACK
3067
3068                        * WRITE ACCA BYTES OF ACCB INTO BUFFER
3069   D6C2 E7 80           LD6C2   STB   ,X+                        STORE A BYTE IN THE BUFFER
3070   D6C4 4A                      DECA                             DECREMENT COUNTER
3071   D6C5 26 FB                   BNE   LD6C2                      BRANCH IF ALL BYTES NOT MOVED
3072   D6C7 39                      RTS
```

```
3073   D6C8 34 04        LD6C8   PSHS  B                       SAVE THE COUNTER ON THE STACK
3074   D6CA EC C1                LDD   ,U++                    GET TWO BYTES OF DATA FROM THE TABLE
3075   D6CC 8D F4                BSR   LD6C2                   WRITE ACCA BYTES OF ACCB INTO THE BUFFER
3076   D6CE 35 04                PULS  B                       * GET THE COUNTER BACK, DECREMENT
3077   D6D0 5A                   DECB                          * IT AND BRANCH IF ALL DATA BLOCKS
3078   D6D1 26 F5                BNE   LD6C8                   * NOT DONE
3079   D6D3 39                   RTS
3080
3081                     * DATA USED TO FORMAT A SECTOR ON THE DISK
3082
3083                     * THESE DATA ARE CLOSE TO THE IBM SYSTEM 34 FORMAT FOR 256 BYTE SECTORS.
3084                     * DOUBLE DENSITY. THE FORMAT GENERALLY CONFORMS TO THAT SPECIFIED ON THE
3085                     * 1793 DATA SHEET. THE GAP SIZES HAVE BEEN REDUCED TO THE MINIMUM
3086                     * ALLOWABLE. THE IBM FORMAT USES $40 AS THE FILL CHARACTER FOR THE DATA
3087                     * BLOCKS WHILE COLOR DOS USES AN $FF AS THE FILL CHARACTER.
3088   D6D4 08 00        LD6D4   FCB   8,0                     SYNC FIELD
3089   D6D6 03 F5                FCB   3,$F5
3090   D6D8 01 FE                FCB   1,$FE                   ID ADDRESS MARK (AM1)
3091                     * TRACK, SIDE, AND SECTOR NUMBERS ARE INSERTED HERE
3092   D6DA 01 01                FCB   1,1                     SECTOR SIZE (256 BYTE SECTORS)
3093   D6DC 01 F7                FCB   1,$F7                   CRC REQUEST
3094   D6DE 16 4E                FCB   22,$4E                  GAP II (POST-ID GAP)
3095   D6E0 0C 00                FCB   12,0                    SYNC FIELD
3096   D6E2 03 F5                FCB   3,$F5
3097   D6E4 01 FB                FCB   1,$FB                   DATA ADDRESS MARK (AM2)
3098   D6E6 00 FF                FCB   0,$FF                   DATA FIELD (256 BYTES)
3099   D6E8 01 F7                FCB   1,$F7                   CRC REQUEST
3100   D6EA 18 4E                FCB   24,$4E                  GAP III (POST DATA GAP)
3101
3102
3103                     * DOS COMMAND
3104   D6EC 26 54        DOS     BNE   LD742                   RETURN IF ARGUMENT GIVEN
3105   D6EE 6E 9F C0 0A          JMP   [DOSVEC]                JUMP TO THE DOS COMMAND
3106
3107   D6F2 34 04        LD6F2   PSHS  B                       SAVE ACCB
3108   D6F4 C6 05                LDB   #$05                    5 RETRIES
3109   D6F6 F7 09 88             STB   ATTCTR                  SAVE RETRY COUNT
3110   D6F9 35 04                PULS  B                       RESTORE ACCB
3111   D6FB 8D 62        LD6FB   BSR   DSKCON                  GO EXECUTE COMMAND
3112   D6FD 0D F0                TST   DCSTA                   CHECK STATUS
3113   D6FF 27 0D                BEQ   LD70E                   BRANCH IF NO ERRORS
3114   D701 96 F0        LD701   LDA   DCSTA                   GET DSKCON ERROR STATUS
3115   D703 C6 3C                LDB   #2*30                   'WRITE PROTECTED' ERROR
3116   D705 85 40                BITA  #$40                    CHECK BIT 6 OF STATUS
3117   D707 26 02                BNE   LD70B                   BRANCH IF WRITE PROTECT ERROR
3118   D709 C6 28        LD709   LDB   #2*20                   'I/O ERROR'
3119   D70B 7E AC 46     LD70B   JMP   LAC46                   JUMP TO ERROR DRIVER
3120   D70E 34 02        LD70E   PSHS  A                       SAVE ACCA
3121   D710 96 EA                LDA   DCOPC                   GET OPERATION CODE
3122   D712 81 03                CMPA  #$03                    CHECK FOR WRITE SECTOR COMMAND
3123   D714 35 02                PULS  A                       RESTORE ACCA
3124   D716 26 2A                BNE   LD742                   RETURN IF NOT WRITE SECTOR
3125   D718 7D 09 87             TST   DVERFL                  CHECK VERIFY FLAG
3126   D71B 27 25                BEQ   LD742                   RETURN IF NO VERIFY
3127   D71D 34 56                PSHS  U,X,B,A                 SAVE REGISTERS
3128   D71F 86 02                LDA   #$02                    READ OPERATION CODE
3129   D721 97 EA                STA   DCOPC                   STORE TO DSKCON PARAMETER
3130   D723 DE EE                LDU   DCBPT                   POINT U TO WRITE BUFFER ADDRESS
3131   D725 8E 07 00             LDX   #DBUF1                  * ADDRESS OF VERIFY BUFFER
3132   D728 9F EE                STX   DCBPT                   * TO DSKCON VARIABLE
3133   D72A 8D 33                BSR   DSKCON                  GO READ SECTOR
3134   D72C DF EE                STU   DCBPT                   RESTORE WRITE BUFFER
3135   D72E 86 03                LDA   #$03                    WRITE OP CODE
3136   D730 97 EA                STA   DCOPC                   SAVE IN DSKCON VARIABLE
3137   D732 96 F0                LDA   DCSTA                   CHECK STATUS FOR THE READ OPERATION
3138   D734 26 0D                BNE   LD743                   BRANCH IF ERROR
3139   D736 5F                   CLRB                          CHECK 256 BYTES
3140   D737 A6 80        LD737   LDA   ,X+                     GET BYTE FROM WRITE BUFFER
3141   D739 A1 C0                CMPA  ,U+                     COMPARE TO READ BUFFER
3142   D73B 26 06                BNE   LD743                   BRANCH IF NOT EQUAL
3143   D73D 5A                   DECB                          * DECREMENT BYTE COUNTER AND
3144   D73E 26 F7                BNE   LD737                   * BRANCH IF NOT DONE
3145   D740 35 56                PULS  A,B,X,U                 RESTORE REGISTERS
3146   D742 39           LD742   RTS
3147   D743 35 56        LD743   PULS  A,B,X,U                 RESTORE REGISTERS
3148   D745 7A 09 88             DEC   ATTCTR                  DECREMENT THE VERIFY COUNTER
3149   D748 26 B1                BNE   LD6FB                   BRANCH IF MORE TRIES LEFT
3150   D74A C6 48                LDB   #2*36                   'VERIFY ERROR'
3151   D74C 20 BD                BRA   LD70B                   JUMP TO ERROR HANDLER
3152
3153                     * VERIFY COMMAND
3154   D74E 5F           VERIFY  CLRB                          OFF FLAG = 0
3155   D74F 81 AA                CMPA  #$AA                    OFF TOKEN ?
3156   D751 27 07                BEQ   LD75A                   YES
3157   D753 53                   COMB                          ON FLAG = $FF
3158   D754 81 88                CMPA  #$88                    ON TOKEN
3159   D756 10 26 DB 1D          LBNE  LB277                   BRANCH TO 'SYNTAX ERROR' IF NOT ON OR OFF
3160   D75A F7 09 87     LD75A   STB   DVERFL                  SET VERIFY FLAG
3161   D75D 0E 9F                JMP   GETNCH                  GET NEXT CHARACTER FROM BASIC
3162
3163                     * DSKCON ROUTINE
3164   D75F 34 76        DSKCON  PSHS  U,Y,X,B,A               SAVE REGISTERS
3165   D761 86 05                LDA   #$05                    * GET RETRY COUNT AND
3166   D763 34 02                PSHS  A                       * SAVE IT ON THE STACK
3167   D765 7F 09 85     LD765   CLR   RDYTMR                  RESET DRIVE NOT READY TIMER
3168   D768 D6 EB                LDB   DCDRV                   GET DRIVE NUMBER
```

```
3169   D76A 8E D8 9D            LDX   #LD89D                    POINT X TO DRIVE ENABLE MASKS
3170   D76D B6 09 86            LDA   DRGRAM                     GET DSKREG IMAGE
3171   D770 84 A8               ANDA  #$A8                       KEEP MOTOR STATUS, DOUBLE DENSITY. HALT ENABLE
3172   D772 AA 85               ORA   B,X                        'OR' IN DRIVE SELECT DATA
3173   D774 8A 20               ORA   #$20                       'OR' IN DOUBLE DENSITY
3174   D776 D6 EC               LDB   DCTRK                      GET TRACK NUMBER
3175   D778 C1 16               CMPB  #22                        PRECOMPENSATION STARTS AT TRACK 22
3176   D77A 25 02               BLO   LD77E                      BRANCH IF LESS THAN 22
3177   D77C 8A 10               ORA   #$10                       TURN ON WRITE PRECOMPENSATION IF >= 22
3178   D77E 1F 89        LD77E  TFR   A,B                        SAVE PARTIAL IMAGE IN ACCB
3179   D780 8A 08               ORA   #$08                       'OR' IN MOTOR ON CONTROL BIT
3180   D782 B7 09 86            STA   DRGRAM                     SAVE IMAGE IN RAM
3181   D785 B7 FF 40            STA   DSKREG                     PROGRAM THE 1793 CONTROL REGISTER
3182   D788 C5 08               BITB  #$08                       = WERE MOTORS ALREADY ON?
3183   D78A 26 06               BNE   LD792                      = DON'T WAIT FOR IT TO COME UP TO SPEED IF ALREADY ON
3184   D78C BD A7 D1            JSR   LA7D1                      * WAIT A WHILE
3185   D78F BD A7 D1            JSR   LA7D1                      * WAIT SOME MORE FOR MOTOR TO COME UP TO SPEED
3186   D792 8D 3D        LD792  BSR   LD7D1                      WAIT UNTIL NOT BUSY OR TIME OUT
3187   D794 26 0A               BNE   LD7A0                      BRANCH IF TIMED OUT (DOOR OPEN. NO DISK, NO POWER. ETC.)
3188   D796 0F F0               CLR   DCSTA                      CLEAR STATUS REGISTER
3189   D798 8E D8 95            LDX   #LD895                     POINT TO COMMAND JUMP VECTORS
3190   D79B D6 EA               LDB   DCOPC                      GET COMMAND
3191   D79D 58                  ASLB                             2 BYTES PER COMMAND JUMP ADDRESS
3192   D79E AD 95               JSR   [B,X]                      GO DO IT
3193   D7A0 35 02        LD7A0  PULS  A                          GET RETRY COUNT
3194   D7A2 D6 F0               LDB   DCSTA                      GET STATUS
3195   D7A4 27 0B               BEQ   LD7B1                      BRANCH IF NO ERRORS
3196   D7A6 4A                  DECA                             DECREMENT RETRIES COUNTER
3197   D7A7 27 08               BEQ   LD7B1                      BRANCH IF NO RETRIES LEFT
3198   D7A9 34 02               PSHS  A                          SAVE RETRY COUNT ON STACK
3199   D7AB 8D 0B               BSR   LD7B8                      RESTORE HEAD TO TRACK 0
3200   D7AD 26 F1               BNE   LD7A0                      BRANCH IF SEEK ERROR
3201   D7AF 20 B4               BRA   LD765                      GO TRY COMMAND AGAIN IF NO ERROR
3202   D7B1 86 78        LD7B1  LDA   #120                       120*1/60 = 2 SECONDS (1/60 SECOND FOR EACH IRQ INTERRUPT)
3203   D7B3 B7 09 85            STA   RDYTMR                     WAIT 2 SECONDS BEFORE TURNING OFF MOTOR
3204   D7B6 35 F6               PULS  A,B,X,Y,U,PC               RESTORE REGISTERS - EXIT DSKCON
3205                     * RESTORE HEAD TO TRACK 0
3206   D7B8 8E 09 7E     LD7B8  LDX   #DR0TRK                    POINT TO TRACK TABLE
3207   D7BB D6 EB               LDB   DCDRV                      GET DRIVE NUMBER
3208   D7BD 6F 85               CLR   B,X                        ZERO TRACK NUMBER
3209   D7BF 86 03               LDA   #$03                       * RESTORE HEAD TO TRACK 0, UNLOAD THE HEAD
3210   D7C1 B7 FF 48            STA   FDCREG                     * AT START, 30 MS STEPPING RATE
3211   D7C4 1E 88               EXG   A,A                        =
3212   D7C6 1E 88               EXG   A,A                        = WAIT FOR 1793 TO RESPOND TO COMMAND
3213   D7C8 8D 07               BSR   LD7D1                      WAIT TILL DRIVE NOT BUSY
3214   D7CA 8D 24               BSR   LD7F0                      WAIT SOME MORE
3215   D7CC 84 10               ANDA  #$10                       1793 STATUS : KEEP ONLY SEEK ERROR
3216   D7CE 97 F0               STA   DCSTA                      SAVE IN DSKCON STATUS
3217   D7D0 39           LD7D0  RTS
3218                     * WAIT FOR THE 1793 TO BECOME UNBUSY. IF IT DOES NOT BECOME UNBUSY,
3219                     * FORCE AN INTERRUPT AND ISSUE A  DRIVE NOT READY  1793 ERROR.
3220   D7D1 9E 8A        LD7D1  LDX   ZERO                       GET ZERO TO X REGISTER - LONG WAIT
3221   D7D3 30 1F        LD7D3  LEAX  -1,X                       DECREMENT LONG WAIT COUNTER
3222   D7D5 27 08               BEQ   LD7DF                      IF NOT READY BY NOW, FORCE INTERRUPT
3223   D7D7 B6 FF 48            LDA   FDCREG                     * GET 1793 STATUS AND TEST
3224   D7DA 85 01               BITA  #$01                       * BUSY STATUS BIT
3225   D7DC 26 F5               BNE   LD7D3                      BRANCH IF BUSY
3226   D7DE 39                  RTS
3227   D7DF 86 D0        LD7DF  LDA   #$D0                       * FORCE INTERRUPT COMMAND - TERMINATE ANY COMMAND
3228   D7E1 B7 FF 48            STA   FDCREG                     * IN PROCESS. DO NOT GENERATE A 1793 INTERRUPT REQUEST
3229   D7E4 1E 88               EXG   A,A                        * WAIT BEFORE READING 1793
3230   D7E6 1E 88               EXG   A,A                        *
3231   D7E8 B6 FF 48            LDA   FDCREG                     RESET INTRQ (FDC INTERRUPT REQUEST)
3232   D7EB 86 80               LDA   #$80                       RETURN DRIVE NOT READY STATUS IF THE DRIVE DID NOT BECOME UNBUSY
3233   D7ED 97 F0               STA   DCSTA                      SAVE DSKCON STATUS BYTE
3234   D7EF 39                  RTS
3235                     * MEDIUM DELAY
3236   D7F0 8E 22 2E     LD7F0  LDX   #8750                      DELAY FOR A WHILE
3237   D7F3 30 1F        LD7F3  LEAX  -1,X                       * DECREMENT DELAY COUNTER AND
3238   D7F5 26 FC               BNE   LD7F3                      * BRANCH IF NOT DONE
3239   D7F7 39                  RTS
3240                     * READ ONE SECTOR
3241   D7F8 86 80        LD7F8  LDA   #$80                       $80 IS READ FLAG (1793 READ SECTOR)
3242   D7FA 8C           LD7FA  CMPX  #$86A0                     SKIP TWO BYTES
3243                     * WRITE ONE SECTOR
3244   D7FB 86 A0        LD7FB  LDA   #$A0                       $A0 IS WRITE FLAG (1793 WRITE SECTOR)
3245   D7FD 34 02               PSHS  A                          SAVE READ/WRITE FLAG ON STACK
3246   D7FF 8E 09 7E            LDX   #DR0TRK                    POINT X TO TRACK NUMBER TABLE IN RAM
3247   D802 D6 EB               LDB   DCDRV                      GET DRIVE NUMBER
3248   D804 3A                  ABX                              POINT X TO CORRECT DRIVE'S TRACK BYTE
3249   D805 E6 84               LDB   ,X                         GET TRACK NUMBER OF CURRENT HEAD POSITION
3250   D807 F7 FF 49            STB   FDCREG+1                   SEND TO 1793 TRACK REGISTER
3251   D80A D1 EC               CMPB  DCTRK                      COMPARE TO DESIRED TRACK
3252   D80C 27 1E               BEQ   LD82C                      BRANCH IF ON CORRECT TRACK
3253   D80E 96 EC               LDA   DCTRK                      GET TRACK DESIRED
3254   D810 B7 FF 4B            STA   FDCREG+3                   SEND TO 1793 DATA REGiSTER
3255   D813 A7 84               STA   ,X                         SAVE IN RAM TRACK IMAGE
3256   D815 86 17               LDA   #$17                       * SEEK COMMAND FOR 1793: DO NOT LOAD THE
3257   D817 B7 FF 48            STA   FDCREG                     * HEAD AT START, VERIFY DESTINATION TRACK,
3258   D81A 1E 88               EXG   A,A                        * 30 MS STEPPING RATE - WAIT FOR
3259   D81C 1E 88               EXG   A,A                        * VALID STATUS FROM 1793
3260   D81E 8D B1               BSR   LD7D1                      WAIT TILL NOT BUSY
3261   D820 26 08               BNE   LD82A                      RETURN IF TIMED OUT
3262   D822 8D CC               BSR   LD7F0                      WAIT SOME MORE
3263   D824 84 18               ANDA  #$18                       KEEP ONLY SEEK ERROR OR CRC ERROR IN ID FIELD
3264   D826 27 04               BEQ   LD82C                      BRANCH IF NO ERRORS - HEAD ON CORRECT TRACK
```

```
3265   D828 97 F0              STA   DCSTA                          SAVE IN DSKCON STATUS
3266   D82A 35 82        LD82A PULS  A,PC
3267                     * HEAD POSITIONED ON CORRECT TRACK
3268   D82C 96 ED        LD82C LDA   DSEC                           GET SECTOR NUMBER DESIRED
3269   D82E B7 FF 4A           STA   FDCREG+2                       SEND TO 1793 SECTOR REGISTER
3270   D831 8E D8 8B           LDX   #LD88B                         * POINT X TO ROUTINE TO BE VECTORED
3271   D834 BF 09 83           STX   DNMIVC                         * TO BY NMI UPON COMPLETION OF DISK I/O AND SAVE VECTOR
3272   D837 9E EE              LDX   DCBPT                          POINT X TO I/O BUFFER
3273   D839 B6 FF 48           LDA   FDCREG                         RESET INTRQ (FDC INTERRUPT REQUEST)
3274   D83C B6 09 86           LDA   DRGRAM                         GET DSKREG IMAGE
3275   D83F 8A 80              ORA   #$80                           SET FLAG TO ENABLE 1793 TO HALT 6809
3276   D841 35 04              PULS  B                              GET READ/WRITE COMMAND FROM STACK
3277   D843 10 9E 8A           LDY   ZERO                           ZERO OUT Y - TIMEOUT INITIAL VALUE
3278   D846 CE FF 48           LDU   #FDCREG                        U POINTS TO 1793 INTERFACE REGISTERS
3279   D849 73 09 82           COM   NMIFLG                         NMI FLAG = $FF: ENABLE NMI VECTOR
3280   D84C 1A 50              ORCC  #$50                           DISABLE FIRQ,IRQ
3281   D84E F7 FF 48           STB   FDCREG                         * SEND READ/WRITE COMMAND TO 1793: SINGLE RECORD, COMPARE
3282   D851 1E 88              EXG   A,A                            * FOR SIDE 0, NO 15 MS DELAY, DISABLE SIDE SELECT
3283   D853 1E 88              EXG   A,A                            * COMPARE, WRITE DATA ADDRESS MARK (FB) - WAIT FOR STATUS
3284   D855 C1 80              CMPB  #$80                           WAS THIS A READ?
3285   D857 27 1C              BEQ   LD875                          IF SO, GO LOOK FOR DATA
3286                     * WAIT FOR THE 1793 TO ACKNOWLEDGE READY TO WRITE DATA
3287   D859 C6 02              LDB   #$02                           DRQ MASK BIT
3288   D85B E5 C4        LD85B BITB  ,U                             IS 1793 READY FOR A BYTE? (DRQ SET IN STATUS BYTE)
3289   D85D 26 0C              BNE   LD86B                          BRANCH IF SO
3290   D85F 31 3F              LEAY  -1,Y                           DECREMENT WAIT TIMER
3291   D861 26 F8              BNE   LD85B                          KEEP WAITING FOR THE 1793 DRQ
3292   D863 7F 09 82     LD863 CLR   NMIFLG                         RESET NMI FLAG
3293   D866 1C AF              ANDCC #$AF                           ENABLE FIRQ,IRQ
3294   D868 7E D7 DF           JMP   LD7DF                          FORCE INTERRUPT, SET DRIVE NOT READY ERROR
3295
3296                     * WRITE A SECTOR
3297   D86B E6 80        LD86B LDB   ,X+                            GET A BYTE FROM RAM
3298   D86D F7 FF 4B           STB   FDCREG+3                       SEND IT TO 1793 DATA REGISTER
3299   D870 B7 FF 40           STA   DSKREG                         REPROGRAM FDC CONTROL REGISTER
3300   D873 20 F6              BRA   LD86B                          SEND MORE DATA
3301                     * WAIT FOR THE 17933 TO ACKNOWLEDGE READY TO READ DATA
3302   D875 C6 02        LD875 LDB   #$02                           DRQ MASK BIT
3303   D877 E5 C4        LD877 BITB  ,U                             DOES THE 1793 HAVE A BYTE? (DRQ SET IN STATUS BYTE)
3304   D879 26 06              BNE   LD881                          YES, GO READ A SECTOR
3305   D87B 31 3F              LEAY  -1,Y                           DECREMENT WAIT TIMER
3306   D87D 26 F8              BNE   LD877                          KEEP WAITING FOR 1793 DRQ
3307   D87F 20 E2              BRA   LD863                          GENERATE DRIVE NOT READY ERROR
3308
3309                     * READ A SECTOR
3310   D881 F6 FF 4B     LD881 LDB   FDCREG+3                       GET DATA BYTE FROM 1793 DATA REGISTER
3311   D884 E7 80              STB   ,X+                            PUT IT IN RAM
3312   D886 B7 FF 40           STA   DSKREG                         REPROGRAM FDC CONTROL REGISTER
3313   D889 20 F6              BRA   LD881                          KEEP GETTING DATA
3314                     * BRANCH HERE ON COMPLETION OF SECTOR READ/WRITE
3315   D88B 1C AF        LD88B ANDCC #$AF                           ENABLE IRQ, FIRO
3316   D88D B6 FF 48           LDA   FDCREG                         * GET STATUS & KEEP WRITE PROTECT, RECORD TYPE/WRITE
3317   D890 84 7C              ANDA  #$7C                           * FAULT, RECORD NOT FOUND, CRC ERROR OR LOST DATA
3318   D892 97 F0              STA   DCSTA                          SAVE IN DSKCON STATUS
3319   D894 39                 RTS
3320
3321                     * DSKCON OPERATION CODE JUMP VECTORS
3322   D895 D7 B8        LD895 FDB   LD7B8                          RESTORE HEAD TO TRACK ZERO
3323   D897 D7 D0              FDB   LD7D0                          NO OP - RETURN
3324   D899 D7 F8              FDB   LD7F8                          READ SECTOR
3325   D89B D7 FB              FDB   LD7FB                          WRITE SECTOR
3326
3327                     * DSKREG MASKS FOR DISK DRIVE SELECT
3328   D89D 01           LD89D FCB   1                              DRIVE SEL 0
3329   D89E 02                 FCB   2                              DRIVE SEL 1
3330   D89F 04                 FCB   4                              DRIVE SEL 2
3331   D8A0 40                 FCB   $40                            DRIVE SEL 3
3332
3333                     * NMI SERVICE
3334   D8A1 B6 09 82     DNMISV LDA  NMIFLG                         GET NMI FLAG
3335   D8A4 27 08              BEQ   LD8AE                          RETURN IF NOT ACTIVE
3336   D8A6 BE 09 83           LDX   DNMIVC                         GET NEW RETURN VECTOR
3337   D8A9 AF 6A              STX   10,S                           STORE AT STACKED PC SLOT ON STACK
3338   D8AB 7F 09 82           CLR   NMIFLG                         RESET NMI FLAG
3339   D8AE 3B                 RTI
3340
3341                     * IRQ SERVICE
3342   D8AF B6 FF 03     DIRQSV LDA  PIA0+3                         63.5 MICRO SECOND OR 60 HZ INTERRUPT?
3343   D8B2 2A FA              BPL   LD8AE                          RETURN IF 63.5 MICROSECOND
3344   D8B4 B6 FF 02           LDA   PIA0+2                         RESET 60 HZ PIA INTERRUPT FLAG
3345   D8B7 B6 09 85           LDA   RDYTMR                         GET TIMER
3346   D8BA 27 11              BEQ   LD8CD                          BRANCH IF NOT ACTIVE
3347   D8BC 4A                 DECA                                 DECREMENT THE TIMER
3348   D8BD B7 09 85           STA   RDYTMR                         SAVE IT
3349   D8C0 26 0B              BNE   LD8CD                          BRANCH IF NOT TIME TO TURN OFF DISK MOTORS
3350   D8C2 B6 09 86           LDA   DRGRAM                         = GET DSKREG IMAGE
3351   D8C5 84 B0              ANDA  #$B0                           = TURN ALL MOTORS AND DRIVE SELECTS OFF
3352   D8C7 B7 09 86           STA   DRGRAM                         = PUT IT BACK IN RAM IMAGE
3353   D8CA B7 FF 40           STA   DSKREG                         SEND TO CONTROL REGISTER (MOTORS OFF)
3354   D8CD 7E 89 55     LD8CD JMP   L8955                          JUMP TO EXTENDED BASIC'S IRQ HANDLER
3355
3356                     * THIS IS THE END OF DISK BASIC (EXCEPT FOR THE DOS COMMAND AT $DF00).
3357                     * THE CODE FROM THIS POINT TO $DF00 IS GARBAGE.
3358                     * DOSBAS 1.1 = 1686 WASTED BYTES
3359
3360                     ** THIS IS THE CODE FOR THE DOS COMMAND
```

```
3361
3362                                    ORG   DOSBAS+$1F00
3363   DF00 11 3F              DOSCOM   SWI3                          DO A SOFTWARE INTERRUPT (#3)
3364   DF02 0F 03                       CLR   TMPLOC                  RESET SECTOR COUNTER
3365   DF04 CC 26 00                    LDD   #DOSBUF                 RAM LOAD ADDRESS FOR SECTOR DATA
3366   DF07 34 06                       PSHS  B,A                     SAVE RAM LOAD ADDRESS
3367   DF09 BE C0 06           LDF09    LDX   DSKVAR                  POINT X TO DSKCON VARIABLES
3368   DF0C 0C 03                       INC   TMPLOC                  INCREMENT SECTOR COUNTER
3369   DF0E 96 03                       LDA   TMPLOC                  GET THE SECTOR COUNTER
3370   DF10 81 12                       CMPA  #SECMAX                 LOADED IN 18 SECTORS? (ONE TRACK)
3371   DF12 22 22                       BHI   LDF36                   YES - EXIT
3372   DF14 A7 03                       STA   $03,X                   NO - SAVE SECTOR NUMBER IN DSEC
3373   DF16 CC 02 00                    LDD   #$0200                  GET FDC OP CODE (READ) AND DRIVE NUMBER (0)
3374   DF19 A7 84                       STA   ,X                      SAVE THEM IN DSKCON VARIABLES (BUG - SHOULD BE STD ,X)
3375   DF1B 86 22                       LDA   #34                     GET TRACK NUMBER (34)
3376   DF1D A7 02                       STA   $02,X                   SAVE IT IN DSKCON VARIABLES TOO
3377   DF1F 35 06                       PULS  A,B                     GET RAM LOAD ADDRESS
3378   DF21 ED 04                       STD   $04,X                   AND SAVE IT IN THE DSKCON VARIABLES
3379   DF23 8B 01                       ADDA  #$01                    ADD 256 (ONE SECTOR) TO RAM LOAD ADDRESS (SHOULD BE INCA)
3380   DF25 34 06                       PSHS  B,A                     SAVE NEW RAM LOAD ADDRESS
3381   DF27 AD 9F C0 04                 JSR   [DCNVEC]                GO READ A SECTOR
3382   DF2B 6D 06                       TST   $06,X                   CHECK FOR ERRORS
3383   DF2D 27 DA                       BEQ   LDF09                   KEEP READING IF NONE
3384   DF2F 35 06              LDF36    PULS  A,B                     PULL LOAD ADDRESS OFF OF THE STACK
3385   DF31 C6 28                       LDB   #2*20                   'IO' ERROR
3386   DF33 7E AC 46                    JMP   LAC46                   JUMP TO ERROR SERVICING ROUTINE
3387   DF36 35 06                       PULS  A,B                     PULL RAM LOAD ADDRESS OFF OF THE STACK
3388   DF38 FC 26 00                    LDD   DOSBUF                  GET FIRST TWO BYTES OF RAM DATA
3389   DF3B 10 83 4F 53                 CMPD  #'OS'                   LOOK FOR 'OS' (OS9) AT START OF BUFFER
3390   DF3F 10 27 46 BF                 LBEQ  DOSBUF+2                IF 'OS' THEN BRANCH TO DATA LOADED IN RAM
3391   DF43 7F 26 00                    CLR   DOSBUF                  * OTHERWISE CLEAR THE FIRST TWO
3392
3393   DF46 7F 26 01           DOSINI   CLR   DOSBUF+1                * BYTES OF RAM DATA
3394   DF49 7E A0 E8                    JMP   BAWMST                  JUMP TO BASIC'S WARM START
3395
3396   DF4C CC 3B 3B                    LDD   #$3B3B                  TWO RTI INSTRUCTIONS
3397   DF4F FD 01 00                    STD   SW3VEC                  *
3398   DF52 FD 01 02                    STD   SW3VEC+2                * LOAD THE SWI2 AND SWI3 JUMP
3399   DF55 FD 01 04                    STD   SW2VEC+1                * VECTORS WITH RTIS
3400   DF58 39                 END      RTS
3401
3402                           * END OF THE DOS AND DOSINI COMMANDS - THE REST OF THE CODE
3403                           * TO THE END OF THE DISK ROM ($DFFF) IS GARBAGE.
3404                           * DOSBAS 1.1 = 167 WASTED BYTES
3405
```

```
0001        00 E1           DHITOK   EQU  $E0                          HIGHEST 1.0 DISK TOKEN
0002        00 32           CYEAR    EQU  '2'
0003                        *
0004                        *
0005                        *
0006                        **
0007                        **** FILE ALLOCATION TABLE FORMAT
0008                        **
0009                        *
0010                        * THE FILE ALLOCATION TABLE (FAT) CONTAINS THE STATUS OF THE GRANULES ON A DISKETTE.
0011                        * THE FAT CONTAINS 6 CONTROL BYTES FOLLOWED BY 68 DATA BYTES (ONE PER GRANULE). ONLY THE
0012                        * FIRST TWO OF THE SIX CONTROL BYTES ARE USED. A VALUE OF $FF IS SAVED IN UNALLOCATED
0013                        * GRANULES. IF BITS 6 & 7 OF THE DATA BYTE ARE SET, THE GRANULE IS THE LAST GRANULE
0014                        * IN A FILE AND BITS 0-5 ARE THE NUMBER OF USED SECTORS IN THAT GRANULE. IF BITS 6 & 7
0015                        * ARE NOT SET, THE DATA BYTE CONTAINS THE NUMBER OF THE NEXT GRANULE IN THE FILE.
0016
0017                        * OFFSETS TO FAT CONTROL BYTES
0018        00 00           FAT0     EQU  0                           ACTIVE FILE COUNTER : DISK TO RAM FAT IMAGE DISABLE
0019        00 01           FAT1     EQU  1                           VALID DATA FLAG: 0=DISK DATA VALID, <> 0 = NEW FAT
0020                        *                                         DATA - DISK DATA INVALID
0021                        *            2 TO 5                       NOT USED
0022        00 06           FATCON   EQU  6                           OFFSET TO START OF FAT DATA (68 BYTES)
0023                        *
0024                        **
0025                        **** DIRECTORY ENTRY FORMAT
0026                        **
0027                        *
0028                        * THE DIRECTORY IS USED TO KEEP TRACK OF HOW MANY FILES ARE STORED ON A DISKETTE
0029                        * AND WHERE THE FILE IS STORED ON THE DISK. THE FIRST GRANULE USED BY THE FILE WILL
0030                        * ALLOW THE FAT TO TRACK DOWN ALL OF THE GRANULES USED BY THE FILE. IF THE FIRST
0031                        * BYTE OF THE DIRECTORY ENTRY IS ZERO, THE FILE HAS BEEN KILLED;
0032                        * IF THE FIRST BYTE IS $FF THEN THE DIRECTORY ENTRY HAS NEVER BEEN USED.
0033                        *
0034                        *            BYTE                         DESCRIPTION
0035
0036        00 00           DIRNAM   EQU  0                           FILE NAME
0037        00 08           DIREXT   EQU  8                           FILE EXTENSION
0038        00 0B           DIRTYP   EQU  11                          FILE TYPE
0039        00 0C           DIRASC   EQU  12                          ASCII FLAG
0040        00 0D           DIRGRN   EQU  13                          FIRST GRANULE IN FILE
0041        00 0E           DIRLST   EQU  14                          NUMBER OF BYTES IN LAST SECTOR
0042                        *            16 TO 31                     UNUSED
0043                        *
0044                        **
0045                        **** FILE CONTROL BLOCK FORMAT
0046                        **
0047                        *
0048                        * THE FILE STRUCTURE OF COLOR TRS DOS IS CONTROLLED BY A FILE CONTROL BLOCK (FCB)
0049                        * THE FCB CONTAINS 25 CONTROL BYTES AND A SECTOR LONG (256 BYTES) DATA BUFFER.
0050                        * THE CONTROL BYTES CONTROL THE ORDERLY FLOW OF DATA FROM THE COMPUTER'S RAM TO
0051                        * THE DISKETTE AND VICE VERSA. THE OPEN COMMAND INITIALIZES THE FCB; THE INPUT,
0052                        * OUTPUT, WRITE, PRINT, GET AND PUT COMMANDS TRANSFER DATA THROUGH THE FCB AND
0053                        * THE CLOSE COMMAND TURNS OFF THE FCB.
0054
0055                        * TABLES OF OFFSETS TO FCB CONTROL BYTES
0056
0057                        ***** RANDOM FILE
0058                        *            BYTE                         DESCRIPTION
0059        00 00           FCBTYP   EQU  0                           FILE TYPE: $40=RANDOM/DIRECT, 0=CLOSED
0060        00 01           FCBDRV   EQU  1                           DRIVE NUMBER
0061        00 02           FCBFGR   EQU  2                           FIRST GRANULE IN FILE
0062        00 03           FCBCGR   EQU  3                           CURRENT GRANULE BEING USED
0063        00 04           FCBSEC   EQU  4                           CURRENT SECTOR BEING USED (1-9)
0064                        *            5                            UNUSED
0065        00 06           FCBPOS   EQU  6                           CURRENT PRINT POSITION - ALWAYS ZERO IN RANDOM FILES
0066        00 07           FCBREC   EQU  7                           CURRENT RECORD NUMBER
0067        00 09           FCBRLN   EQU  9                           RANDOM FILE RECORD LENGTH
0068        00 0B           FCBBUF   EQU  11                          POINTER TO START OF THIS FILE'S RANDOM ACCESS BUFFER
0069        00 0D           FCBSOF   EQU  13                          SECTOR OFFSET TO CURRENT POSITION IN RECORD
0070        00 0F           FCBFLG   EQU  15                          GET/PUT FLAG: 0=PUT, 1=PUT
0071                        *            16,17                        NOT USED
0072        00 12           FCBDIR   EQU  18                          DIRECTORY ENTRY NUMBER (0-71)
0073        00 13           FCBLST   EQU  19                          NUMBER OF BYTES IN LAST SECTOR OF FILE
0074        00 15           FCBGET   EQU  21                          'GET' RECORD COUNTER: HOW MANY CHARACTERS HAVE BEEN
0075                        *                                         PULLED OUT OF THE CURRENT RECORD
0076        00 17           FCBPUT   EQU  23                          'PUT' RECORD COUNTER: POINTER TO WHERE IN THE RECORD THE NEXT
0077                        *                                         BYTE WILL BE 'PUT'
0078        00 19           FCBCON   EQU  25                          OFFSET TO START OF FCB DATA BUFFER (256 BYTES)
0079
0080                        ***** SEQUENTIAL FILE
0081                        *            BYTE                         DESCRIPTION
0082        00 00           FCBTYP   EQU  0                           FILE TYPE: $10=INPUT, $20=OUTPUT, 0=CLOSED
0083        00 01           FCBDRV   EQU  1                           DRIVE NUMBER
0084        00 02           FCBFGR   EQU  2                           FIRST GRANULE IN FILE
0085        00 03           FCBCGR   EQU  3                           CURRENT GRANULE BEING USED
0086        00 04           FCBSEC   EQU  4                           CURRENT SECTOR BEING USED (1-9)
0087        00 05           FCBCPT   EQU  5                           INPUT FILE: CHARACTER POINTER - POINTS TO NEXT CHARACTER IN
0088                        *                                         FILE TO BE PROCESSED.
0089                        *                                         OUTPUT FILE: FULL SECTOR FLAG - IF IT IS 1 WHEN THE FILE IS
0090                        *                                         CLOSED IT MEANS 256 BYTES OF THE LAST SECTOR HAVE BEEN USED.
0091        00 06           FCBPOS   EQU  6                           CURRENT PRINT POSITION
0092        00 07           FCBREC   EQU  7                           CURRENT RECORD NUMBER: HOW MANY WHOLE SECTORS HAVE BEEN
0093                        *                                         INPUT OR OUTPUT TO A FILE.
0094                        *            9 TO 15                      UNUSED
0095        00 10           FCBRLN   EQU  16                          CACHE FLAG: 00=CACHE EMPTY, $FF=CACHE FULL
0096        00 11           FCBCDT   EQU  17                          CACHE DATA BYTE
```

```
0097        00 12            FCBDIR  EQU   18                          DIRECTORY ENTRY NUMBER (0-71)
0098        00 13            FCBLST  EQU   19                          NUMBER OF BYTES IN LAST SECTOR OF FILE
0099                         *               21,22                     UNUSED
0100        00 17            FCBDFL  EQU   23                          INPUT FILE ONLY: DATA LEFT FLAG: 0=DATA LEFT, $FF=NO DATA (EMPTY)
0101        00 18            FCBLFT  EQU   24                          NUMBER OF CHARACTERS LEFT IN BUFFER (INPUT FILE)
0102                         *                                         NUMBER OF CHARS STORED IN BUFFER (OUTPUT FILE)
0103        00 19            FCBCON  EQU   25                          OFFSET TO FCB DATA BUFFER (256 BYTES)
0104
0105                                 ORG   $C000
0106
0107   C000 44 4B            DOSBAS  FCC   'DK'
0108   C002 20 04            LC002   BRA   LC008
0109
0110   C004 D6 6C            DCNVEC  FDB   DSKCON                       DSKCON POINTER
0111   C006 00 EA            DSKVAR  FDB   $00EA                        ADDRESS OF DSKCON VARIABLES
0112
0113                         **** ZERO OUT THE RAM USED BY DISK BASIC
0114   C008 8E 06 00         LC008   LDX   #DBUF0                       POINT X TO START OF DISK RAM
0115   C00B 6F 80            LC00B   CLR   ,X+                          CLEAR A BYTE
0116   C00D 8C 09 89                 CMPX  #DFLBUF                     END OF DISK'S RAM?
0117   C010 26 F9                    BNE   LC00B                       NO - KEEP CLEARING
0118   C012 8E C0 F6                 LDX   #LC0F6                      POINT X TO ROM IMAGE OF COMMAND INTERPRETATION TABLE
0119   C015 CE 01 34                 LDU   #COMVEC+20                  POINT U TO RAM ADDRESS OF SAME
0120   C018 C6 0A                    LDB   #10                         10 BYTES PER TABLE
0121   C01A BD A5 9A                 JSR   LA59A                       MOVE (B) BYTES FROM (X) TO (U)
0122   C01D CC B2 77                 LDD   #LB277                      SYNTAX ERROR ADDRESS
0123   C020 ED 43                    STD   $03,U                       * SET JUMP TABLE ADDRESSES OF THE USER COMMAND
0124   C022 ED 48                    STD   $08,U                       * INTERPRETATION TABLE TO POINT TO SYNTAX ERROR
0125   C024 6F C4                    CLR   ,U                          CLEAR BYTE 0 OF USER TABLE (DOESN'T EXIST FLAG)
0126   C026 6F 45                    CLR   $05,U                       SET NUMBER OF SECONDARY USER TOKENS TO ZERO
0127   C028 CC CE 2E                 LDD   #DXCVEC                     * SAVE NEW
0128   C02B FD 01 2D                 STD   COMVEC+13                   * POINTERS TO EXBAS
0129   C02E CC CE 56                 LDD   #DXIVEC                     * COMMAND AND SECONDARY
0130   C031 FD 01 32                 STD   COMVEC+18                   * COMMAND INTERPRETATION ROUTINES
0131                         **** MOVE THE NEW RAM VECTORS FROM ROM TO RAM
0132   C034 CE 01 5E                 LDU   #RVEC0                      POINT U TO 1ST RAM VECTOR
0133   C037 86 7E            LC037   LDA   #$7E                        OP CODE OF JMP INSTRUCTION
0134   C039 B7 01 A0                 STA   RVEC22                      SET 1ST BYTE OF 'GET'/'PUT' RAM VECTOR TO 'JMP'
0135   C03C A7 C0                    STA   ,U+                         SET 1ST BYTE OF RAM VECTOR TO 'JMP'
0136   C03E EC 81                    LDD   ,X++                        GET RAM VECTOR FROM ROM
0137   C040 ED C1                    STD   ,U++                        STORE IT IN RAM
0138   C042 8C C1 26                 CMPX  #LC126                      COMPARE TO END OF ROM VALUES
0139   C045 26 F0                    BNE   LC037                       BRANCH IF NOT ALL VECTORS MOVED
0140   C047 8E C2 9A                 LDX   #DVEC22                     GET ROM VALUE OF 'GET'/'PUT' RAM VECTOR
0141   C04A BF 01 A1                 STX   RVEC22+1                    SAVE IT IN RAM
0142                         **** INITIALIZE DISK BASIC'S USR VECTORS
0143   C04D 8E 09 5F                 LDX   #DUSRVC                     POINT X TO START OF DISK BASIC USR VECTORS
0144   C050 9F B0                    STX   USRADR                      SAVE START ADDRESS IN USRADR
0145   C052 CE B4 4A                 LDU   #LB44A                      POINT U TO ADDRESS OF 'FUNCTION CALL' ERROR
0146   C055 C6 0A                    LDB   #10                         10 USER VECTORS TO INITIALIZE
0147   C057 EF 81            LC057   STU   ,X++                        SET USR VECTOR TO 'FC' ERROR
0148   C059 5A                       DECB                              DECREMENT USR VECTOR COUNTER
0149   C05A 26 FB                    BNE   LC057                       BRANCH IN NOT DONE WITH ALL 10 VECTORS
0150   C05C 8E D7 AE                 LDX   #DNMISV                     GET ADDRESS OF NMI SERVICING ROUTINE
0151   C05F BF 01 0A                 STX   NMIVEC+1                    SAVE IT IN NMI VECTOR
0152   C062 86 7E                    LDA   #$7E                        OP CODE OF JMP
0153   C064 B7 01 09                 STA   NMIVEC                      MAKE THE NMI VECTOR A JMP
0154   C067 8E D7 BC                 LDX   #DIRQSV                     GET ADDRESS OF DISK BASIC IRQ SERVICING ROUTINE
0155   C06A BF 01 0D                 STX   IRQVEC+1                    SAVE IT IN IRQVEC
0156   C06D 86 13                    LDA   #19                         = INITIALIZE WRITE FAT
0157   C06F B7 09 7A                 STA   WFATVL                      = TO DISK TRIGGER VALUE
0158   C072 7F 08 00                 CLR   FATBL0                      *
0159   C075 7F 08 4A                 CLR   FATBL1                      * INITIALIZE THE ACTIVE FILE COUNTER OF
0160   C078 7F 08 94                 CLR   FATBL2                      * EACH FAT TO ZERO. THIS WILL CAUSE THE FATS
0161   C07B 7F 08 DE                 CLR   FATBL3                      * TO THINK THERE ARE NO ACTIVE FILES
0162   C07E 8E 09 89                 LDX   #DFLBUF                     = GET THE STARTING ADDRESS OF THE
0163   C081 BF 09 48                 STX   RNBFAD                      = RANDOM FILE BUFFER FREE AREA AND DAVE IT AS THE
0164   C084 30 89 01 00              LEAX  256,X                       = START ADDRESS OF FREE RAM FOR RANDOM FILE BUFFERS
0165   C088 BF 09 4A                 STX   FCBADR                      SAVE 256 BYTES FOR RANDOM FILE BUFFERS INITIALLY
0166                         *                                         SAVE START ADDRESS OF FCBS
0167   C08B 30 01                    LEAX  $01,X                       * ADD ONE AND SAVE THE STARTING
0168   C08D BF 09 28                 STX   FCBV1                       * ADDRESS OF FCB1
0169   C090 6F 00                    CLR   FCBTYP,X                    CLEAR THE FIRST BYTE OF FCB 1 (CLOSE FCB)
0170   C092 30 89 01 19              LEAX  FCBLEN,X                    POINT X TO FCB 2
0171   C096 BF 09 2A                 STX   FCBV1+2                     SAVE ITS STARTING ADDRESS IN FCB VECTOR TABLE
0172   C099 6F 00                    CLR   FCBTYP,X                    CLEAR THE FIRST BYTE OF FCB 2 (CLOSE FCB)
0173   C09B 30 89 01 19              LEAX  FCBLEN,X                    * POINT X TO SYSTEM FCB - THIS FCB WILL ONLY
0174                         *                                         * BE USED TO COPY, LOAD, SAVE, MERGE, ETC
0175   C09F BF 09 2C                 STX   FCBV1+4                     SAVE ITS ADDRESS IN THE FCB VECTOR TABLE
0176   C0A2 6F 00                    CLR   FCBTYP,X                    CLEAR THE FIRST BYTE OF SYSTEM FCB (CLOSE FCB)
0177   C0A4 86 02                    LDA   #$02                        * SET THE NUMBER OF ACTIVE RESERVED
0178   C0A6 B7 09 5B                 STA   FCBACT                      * FILE BUFFERS TO 2 (1,2)
0179   C0A9 30 89 01 19              LEAX  FCBLEN,X                    POINT X TO ONE PAST THE END OF SYSTEM FCB
0180   C0AD 1F 10                    TFR   X,D                         SAVE THE ADDRESS IN ACCD
0181   C0AF 5D                       TSTB                              ON AN EVEN 256 BYTE BOUNDARY?
0182   C0B0 27 01                    BEQ   LC0B3                       YES
0183   C0B2 4C                       INCA                              NO - ADD 256 TO ADDRESS
0184   C0B3 1F 89            LC0B3   TFR   A,B                         COPY ACCA TO ACCB
0185   C0B5 CB 18                    ADDB  #24                         SAVE ENOUGH ROOM FOR 4 GRAPHICS PAGES (PCLEAR 4)
0186   C0B7 D7 19                    STB   TXTTAB                      SAVE NEW START OF BASIC ADDRESS
0187   C0B9 BD 96 EC                 JSR   L96EC                       INITIALIZE EXBAS VARIABLES & DO A NEW
0188   C0BC 96 BA                    LDA   BEGGRP                      GET THE START OF CURRENT GRAPHICS PAGE
0189   C0BE 8B 06                    ADDA  #$06                        ADD 1.5K (6 X 256 = ONE GRAPHICS PAGE)
0190   C0C0 97 B7                    STA   ENDGRP                      SAVE NEW END OF GRAPHICS PAGE
0191   C0C2 8D 19                    BSR   LC0DD                       GO INITIALIZE THE FLOPPY DISK CONTROLLER
0192   C0C4 1C AF                    ANDCC #$AF                        TURN ON IRQ AND FIRQ
```

```
0193   C0C6 8E C1 25                LDX   #LC125                          POINT X TO DISK BASIC COPYRIGHT MESSAGE
0194   C0C9 BD B9 9C                JSR   STRINOUT                        PRINT COPYRIGHT MESSAGE TO SCREEN
0195   C0CC 8E C0 D4                LDX   #DKWMST                         GET DISK BASIC WARM START ADDRESS
0196   C0CF 9F 72                   STX   RSTVEC                          SAVE IT IN RESET VECTOR
0197   C0D1 7E A0 E2                JMP   LA0E2                           JUMP BACK TO BASIC
0198
0199   C0D4 12              DKWMST  NOP                                   WARM START INDICATOR
0200   C0D5 8D 06                   BSR   LC0DD                           INITIALIZE THE FLOPPY DISK CONTROLLER
0201   C0D7 BD D1 E5                JSR   LD1E5                           CLOSE FILES AND DO MORE INITIALIZATION
0202   C0DA 7E 80 C0                JMP   XBWMST                          JUMP TO EXBAS' WARM START
0203   C0DD 7F 09 82        LC0DD   CLR   NMIFLG                          RESET NMI FLAG
0204   C0E0 7F 09 85                CLR   RDYTMR                          RESET DRIVE NOT READY TIMER
0205   C0E3 7F 09 86                CLR   DRGRAM                          RESET RAM IMAGE OF DSKREG (MOTORS OFF)
0206   C0E6 7F FF 40                CLR   DSKREG                          RESET DISK CONTROL REGISTER
0207   C0E9 86 D0                   LDA   #$D0                            FORCE INTERRUPT COMMAND OF 1793
0208   C0EB B7 FF 48                STA   FDCREG                          SEND IT TO 1793
0209   C0EE 1E 88                   EXG   A,A                             * DELAY
0210   C0F0 1E 88                   EXG   A,A                             * DELAY SOME MORE
0211   C0F2 B6 FF 48                LDA   FDCREG                          GET 1793 STATUS (CLEAR REGISTER)
0212   C0F5 39                      RTS
0213
0214                        * DISK BASIC COMMAND INTERP TABLES
0215   C0F6 13              LC0F6   FCB   19                              19 DISK BASIC 1.0 COMMANDS
0216   C0F7 C1 7F                   FDB   LC17F                           DISK BASIC'S COMMAND DICTIONARY
0217   C0F9 C2 20                   FDB   LC220                           COMMAND JUMP TABLE
0218   C0FB 06                      FCB   6                               6 DISK BASIC SECONDARY FUNCTIONS
0219   C0FC C2 01                   FDB   LC201                           SECONDARY FUNCTION TABLE
0220   C0FE C2 36                   FDB   LC236                           SECONDARY FUNCTION JUMP TABLE
0221
0222                        * RAM HOOKS FOR DISK BASIC
0223   C100 C4 26 C8 38 C8 43 LC100 FDB   DVEC0,DVEC1,DVEC2
0224   C106 CB 4A C5 8F C8 18       FDB   DVEC3,DVEC4,DVEC5
0225   C10C C8 1B CA 3B CA 4B       FDB   DVEC6,DVEC7,DVEC8
0226   C112 8E 90 CC 5B C8 59       FDB   XVEC9,DVEC10,DVEC11
0227   C118 C6 B7 CA 36 C8 60       FDB   DVEC12,DVEC13,DVEC14
0228   C11E CD F6 C6 B7 C2 4D       FDB   DVEC15,DVEC12,DVEC17
0229   C124 C9 90                   FDB   DVEC18
0230
0231                        * DISK BASIC COPYRIGHT MESSAGE
0232   C126 44 49 53 4B 20 45 LC126 FCC   'DISK EXTENDED COLOR BASIC 1.0'
0233   C12C 58 54 45 4E 44 45
0234   C132 44 20 43 4F 4C 4F
0235   C138 52 20 42 41 53 49
0236   C13E 43 20 31 2E 30
0237   C143 0D                      FCB   CR
0238   C144 43 4F 50 59 52 49       FDB   'COPYRIGHT (C) 198'
0239   C14A 47 48 54 20 28 43
0240   C150 29 20 31 39 38
0241   C155 32                      FCB   CYEAR
0242   C156 20 42 59 20 54 41       FCC   ' BY TANDY'
0243   C15C 4E 44 59
0244   C15F 0D                      FCB   CR
0245   C160 55 4E 44 45 52 20       FCC   'UNDER LICENSE FROM MICROSOFT'
0246   C166 4C 49 43 45 4E 53
0247   C16C 45 20 46 52 4F 4D
0248   C172 20 4D 49 43 52 4F
0249   C178 53 4F 46 54
0250   C17C 0D 0D 00                FDB   CR,CR,0
0251
0252                        * DISK BASIC COMMAND DICTIONARY TABLE
0253                        *                                             TOKEN #
0254   C17F 44 49 D2        LC17F   FCC   'DI',$80+'R'                    CE
0255   C182 44 52 49 56 C5          FCC   'DRIV',$80+'E'                  CF
0256   C187 46 49 45 4C C4          FCC   'FIEL',$80+'D'                  D0
0257   C18C 46 49 4C 45 D3          FCC   'FILE',$80+'S'                  D1
0258   C191 4B 49 4C CC             FCC   'KIL',$80+'L'                   D2
0259   C195 4C 4F 41 C4             FCC   'LOA',$80+'D'                   D3
0260   C199 4C 53 45 D4             FCC   'LSE',$80+'T'                   D4
0261   C19D 4D 45 52 47 C5          FCC   'MERG',$80+'E'                  D5
0262   C1A2 52 45 4E 41 4D C5       FCC   'RENAM',$80+'E'                 D6
0263   C1A8 52 53 45 D4             FCC   'RSE',$80+'T'                   D7
0264   C1AC 53 41 56 C5             FCC   'SAV',$80+'E'                   D8
0265   C1B0 57 52 49 54 C5          FCC   'WRIT',$80+'E'                  D9
0266   C1B5 56 45 52 49 46 D9       FCC   'VERIF',$80+'Y'                 DA
0267   C1BB 55 4E 4C 4F 41 C4       FCC   'UNLOA',$80+'D'                 DB
0268   C1C1 44 53 4B 49 4E C9       FCC   'DSKIN',$80+'I'                 DC
0269   C1C7 42 41 43 4B 55 D0       FCC   'BACKU',$80+'P'                 DD
0270   C1CD 43 4F 50 D9             FCC   'COP',$80+'Y'                   DE
0271   C1D1 44 53 4B 49 A4          FCC   'DSKI',$80+'$'                  DF
0272   C1D6 44 53 4B 4F A4          FCC   'DSKO',$80+'$'                  E0
0273
0274                        * DISK BASIC COMMAND JUMP TABLE
0275                        *                                             COMMAND / TOKEN #
0276   C1DB CC A9           LC1DB   FDB   DIR                             DIR / CE
0277   C1DD CE C5                   FDB   DRIVE                           DRIVE / CF
0278   C1DF D0 BC                   FDB   FIELD                           FIELD / D0
0279   C1E1 D1 5C                   FDB   FILES                           FILES / D1
0280   C1E3 C6 EF                   FDB   KILL                            KILL / D2
0281   C1E5 CA 48                   FDB   LOAD                            LOAD / D3
0282   C1E7 D1 02                   FDB   LSET                            LSET / D4
0283   C1E9 CA 39                   FDB   MERGE                           MERGE / D5
0284   C1EB D0 1B                   FDB   RENAME                          RENAME / D6
0285   C1ED D1 01                   FDB   RSET                            RSET / D7
0286   C1EF C9 E0                   FDB   SAVE                            SAVE / D8
0287   C1F1 D0 66                   FDB   WRITE                           WRITE / D9
0288   C1F3 D7 4E                   FDB   VERIFY                          VERIFY / DA
```

```
0289   C1F5 D2 33                FDB   UNLOAD                        UNLOAD / DB
0290   C1F7 D5 99                FDB   DSKINI                        DSKINI /DC
0291   C1F9 D2 62                FDB   BACKUP                        BACKUP / DD
0292   C1FB D3 B9                FDB   COPY                          COPY / DE
0293   C1FD D4 ED                FDB   DSKI                          DSKI$ / DF
0294   C1FF D5 62                FDB   DSKO                          DSKO$ / E0
0295
0296                        * SECONDARY FUNCTION DICTIONARY TABLE
0297                        *                                         TOKEN #
0298   C201 43 56 CE        LC201  FCC   'CV',$80+'N'                 A2
0299   C204 46 52 45 C5            FCC   'FRE',$80+'E'               A3
0300   C208 4C 4F C3               FCC   'LO',$80+'C'                A4
0301   C20B 4C 4F C6               FCC   'LO',$80+'F'                A5
0302   C20E 4D 4B 4E A4            FCC   'MKN',$80+'$'               A6
0303   C212 41 D3                  FCC   'A',$80+'S'                 A7
0304
0305                        * DISK BASIC SECONDARY FUNCTION JUMP TABLE
0306                        *                                         FUNCTION / TOKEN #
0307   C214 CD F4           LC214  FDB   CVN                         CVN / A2
0308   C216 CE 9C                  FDB   FREE                        FREE / A3
0309   C218 CE 10                  FDB   LOC                         LOC / A4
0310   C21A CE 37                  FDB   LOF                         LOF / A5
0311   C21C CE 02                  FDB   MKN$                        MKN$ / A6
0312   C21E B2 77                  FDB   AS                          AS / A7
0313
0314   C220 81 E0           LC220  CMPA  #DHITOK                     *COMPARE TO HIGHEST DISK BASIC TOKEN
0315   C222 22 08                  BHI   LC22C                       *AND BRANCH IF HIGHER
0316   C224 8E C1 DB               LDX   #LC1DB                      POINT X TO DISK BASIC COMMAND JUMP TABLE
0317   C227 80 CE                  SUBA  #$CE                        SUBTRACT OUT LOWEST DISK BASIC COMMAND TOKEN
0318   C229 7E AD D4               JMP   LADD4                       JUMP TO BASIC'S COMMAND HANDLER
0319   C22C 81 E0           LC22C  CMPA  #DHITOK                     COMPARE TO HIGHEST DISK BASIC TOKEN
0320   C22E 10 23 F0 45            LBLS  LB277                       'SYNTAX' ERROR IF < DISK BASIC COMMAND TOKEN
0321   C232 6E 9F 01 41            JMP   [COMVEC+33]                 PROCESS A USER COMMAND TOKEN
0322                        *DISK BASIC SECONDARY COMMAND INTERPRETATION HANDLER
0323   C236 C1 4E           LC236  CMPB  $($A7-$80)*2                *COMPARE MODIFIED SECONDARY TOKEN TO
0324   C238 23 04                  BLS   LC23E                       *HIGHEST DISK BASIC TOKEN & BRANCH IF HIGHER
0325   C23A 6E 9F 01 46            JMP   [COMVEC+38]                 JUMP TO USER SECONDARY COMMAND HANDLER
0326   C23E C0 44           LC23E  SUBB  $($A2-$80)*2                *SUBTRACT OUT THE SMALLEST SECONDARY
0327   C240 34 04                  PSHS  B                           *DISK TOKEN & SAVE MODIFIED TOKEN ON THE STACK
0328   C242 BD B2 62               JSR   LB262                       SYNTAX CHECK FOR '(' AND EVALUATE EXPRESSION
0329   C245 35 04                  PULS  B                           RESTORE MODIFIED TOKEN
0330   C247 8E C2 14               LDX   #LC214                      POINT X TO SECONDARY COMMAND JUMP TABLE
0331   C24A 7E B2 CE               JMP   LB2CE                       JUMP TO BASIC'S SECONDARY COMMAND HANDLER
0332
0333                        * ERROR DRIVER RAM VECTOR
0334   C24D 35 20           DVEC17 PULS  Y                           PUT THE RETURN ADDRESS INTO Y
0335   C24F BD AD 33               JSR   LAD33                       RESET THE CONT FLAG, ETC
0336   C252 BD D1 E5               JSR   LD1E5                       INITIALIZE SOME DISK VARIABLES AND CLOSE FILES
0337   C255 34 24                  PSHS  Y,B                         PUT RETURN ADDRESS AND ERROR NUMBER ON THE STACK
0338   C257 BD CA 3B               JSR   DVEC7                       CLOSE ALL FILES
0339   C25A 35 04                  PULS  B                           GET THE ERROR NUMBER BACK
0340   C25C C1 36                  CMPB  #2*27                       COMPARE TO THE LOWEST DISK ERROR NUMBER
0341   C25E 10 25 C6 8E            LBCS  XVEC17                      BRANCH TO EXBAS ERROR HANDLER IF NOT DISK ERROR NUMBER
0342   C262 32 62                  LEAS  $02,S                       PURGE RETURN ADDRESS OFF THE STACK
0343   C264 BD A7 E9               JSR   LA7E9                       TURN OFF THE CASSETTE MOTOR
0344   C267 BD A9 74               JSR   LA974                       DISABLE THE ANALOG MULTIPLEXER
0345   C26A 0F 6F                  CLR   DEVNUM                      SET DEVICE NUMBER TO THE SCREEN
0346   C26C BD B9 5C               JSR   LB95C                       SEND A CR TO THE SCREEN
0347   C26F BD B9 AF               JSR   LB9AF                       SEND A '?' TO THE SCREEN
0348   C272 8E C2 42               LDX   #LC278-2*27                 POINT X TO DISK BASIC'S ERROR TABLE
0349   C275 7E AC 60               JMP   LAC60                       JUMP TO BASIC'S ERROR HANDLER
0350
0351                        * DISK BASIC ERROR MESSAGES
0352   C278 42 52           LC290  FCC   'BR'                        27 BAD RECORD
0353   C27A 44 46                  FCC   'DF'                        28 DISK FULL
0354   C27C 4F 42                  FCC   'OB'                        29 OUT OF BUFFER SPACE
0355   C27E 57 50                  FCC   'WP'                        30 WRITE PROTECTED
0356   C280 46 4E                  FCC   'FN'                        31 BAD FILE NAME
0357   C282 46 53                  FCC   'FS'                        32 BAD FILE STRUCTURE
0358   C284 41 45                  FCC   'AE'                        33 FILE ALREADY EXISTS
0359   C286 46 4F                  FCC   'FO'                        34 FIELD OVERFLOW
0360   C288 53 45                  FCC   'SE'                        35 SET TO NON-FIELDED STRING
0361   C28A 56 46                  FCC   'VF'                        36 VERIFICATION ERROR
0362   C28C 45 52                  FCC   'ER'                        37 WRITE OR INPUT PAST END OF RECORD
0363
0364                        * DISK FILE EXTENSIONS
0365   C28E 42 41 53        BASEXT FCC   'BAS'                       BASIC FILE EXTENSION
0366   C291 20 20 20        DEFEXT FCC   '   '                        BLANK (DEFAULT) FILE EXTENSION
0367   C294 44 41 54        DATEXT FCC   'DAT'                       DATA FILE EXTENSION
0368   C297 42 49 4E        BINEXT FCC   'BIN'                       BINARY FILE EXTENSION
0369
0370                        * CLS RAM VECTOR
0371   C29A 34 11           DVEC22 PSHS  X,CC                        SAVE X REG AND STATUS
0372   C29C AE 63                  LDX   $03,S                       LOAD X WITH CALLING ADDRESS
0373   C29E 8C 97 5F               CMPX  #L975F                      COMING FROM EXBAS' GET/PUT?
0374   C2A1 26 04                  BNE   LC2A7                       NO
0375   C2A3 81 23                  CMPA  #'#'                        NUMBER SIGN (GET#, PUT#)?
0376   C2A5 27 02                  BEQ   LC2A9                       BRANCH IF GET OR PUT TO RANDOM FILE
0377   C2A7 35 91           LC2A7  PULS  CC,X,PC                     RESTORE X REG, STATUS AND RETURN
0378
0379                        * GET/PUT TO A DIRECT/RANDOM FILE
0380   C2A9 32 65           LC2A9  LEAS  $05,S                       PURGE RETURN ADDRESS AND REGISTERS OFF OF THE STACK
0381   C2AB BD C7 FE               JSR   LC7FE                       EVALUATE DEVICE NUMBER & SET FCB POINTER
0382   C2AE 9F F1                  STX   FCBTMP                      SAVE FCB POINTER
0383   C2B0 6F 88 15               CLR   FCBGET,X                    * RESET THE GET
0384   C2B3 6F 88 16               CLR   FCBGET+1,X                  * DATA POINTER
```

```
0385   C2B6 6F 88 17              CLR   FCBPUT,X                      = RESET THE PUT
0386   C2B9 6F 88 18              CLR   FCBPUT+1,X                    = DATA POINTER
0387   C2BC 6F 06                 CLR   FCBPOS,X                      RESET PRINT POSITION COUNTER
0388   C2BE A6 01                 LDA   FCBDRV,X                      *GET THE FCB DRIVE NUMBER AND
0389   C2C0 97 EB                 STA   DCDRV                         *SAVE IT IN DSKCON VARIABLE
0390   C2C2 9D A5                 JSR   GETCCH                        GET CURRENT INPUT CHARACTER FROM BASIC
0391   C2C4 27 0A                 BEQ   LC2D0                         BRANCH IF END OF LINE
0392   C2C6 BD B2 6D              JSR   SYNCOMMA                      SYNTAX CHECK FOR COMMA
0393   C2C9 BD B3 E6              JSR   LB3E6                         EVALUATE EXPRESSION - RETURN IN ACCD
0394   C2CC 9E F1          LC2CC  LDX   FCBTMP                        POINT X TO FCB
0395   C2CE ED 07                 STD   FCBREC,X                      SAVE RECORD NUMBER IN FCB
0396   C2D0 BD C6 58       LC2D0  JSR   LC658                         INCREMENT RECORD NUMBER
0397   C2D3 EC 09                 LDD   FCBRLN,X                      * GET RANDOM FILE RECORD LENGTH AND RANDOM FILE
0398   C2D5 AE 0B                 LDX   FCBBUF,X                      * BUFFER POINTER AND SAVE THEM ON THE STACK -
0399   C2D7 34 16                 PSHS  X,B,A                         * THESE ARE THE INITIAL VALUES OF A TEMPORARY
0400                       *                                          * RECORD LENGTH COUNTER AND RANDOM BUFFER
0401                       *                                          * POINTER WHICH ARE MAINTAINED ON THE STACK
0402   C2D9 30 5E                 LEAX  -2,U                          POINT X TO (RECORD NUMBER -1)
0403   C2DB BD 9F B5              JSR   L9FB5                         MULT (UNSIGNED) RECORD LENGTH X (RECORD NUMBER -1)
0404   C2DE 34 60                 PSHS  U,Y                           SAVE PRODUCT ON THE STACK
0405   C2E0 A6 E0                 LDA   ,S+                           CHECK MS BYTE OF PRODUCT
0406   C2E2 26 09                 BNE   LC2ED                         'BR' ERROR IF NOT ZERO (RECORD NUMBER TOO BIG)
0407   C2E4 35 10                 PULS  X                             * PULL THE BOTTOM 3 PRODUCT BYTES OFF THE STACK;
0408   C2E6 35 04                 PULS  B                             * TOP TWO IN X, BOTTOM IN ACCB; ACCB POINTS TO
0409                       *                                          * THE FIRST BYTE OF THE SECTOR USED BY THIS RECORD,
0410                       *                                          * (X) CONTAINS THE SECTOR OFFSET (IN WHICH SECTOR
0411                       *                                          * FROM THE START THE BYTE IS LOCATED)
0412   C2E8 8C 02 64       LC2E8  CMPX  #(TRKMAX-1)                   612 SECTORS MAX IN A RANDOM FILE
0413   C2EB 25 05                 BLO   LC2F2                         BRANCH IF RECORD LENGTH O.K.
0414   C2ED C6 36          LC2ED  LDB   #2*27                         'BAD RECORD' ERROR
0415   C2EF 7E AC 46              JMP   LAC46                         JUMP TO ERROR HANDLER
0416   C2F2 DE F1          LC2F2  LDU   FCBTMP                        POINT U TO FCB
0417   C2F4 AC 4D                 CMPX  FCBSOF,U                      * COMPARE SAVED SECTOR OFFSET TO THE CURRENT SECTOR OFFSET
0418   C2F6 10 27 00 B7           LBEQ  LC3B1                         * BEING PROCESSED - DO NOT PROCESS A NEW SECTOR IF THEY ARE EQUAL
0419   C2FA 34 14                 PSHS  X,B                           SAVE BYTE AND SECTOR OFFSET TO RECORD START ON STACK
0420   C2FC A6 4F                 LDA   FCBFLG,U                      * CHECK FCB GET/PUT FLAG AND
0421   C2FE 27 06                 BEQ   LC306                         * BRANCH IF IT WAS A GET
0422   C300 6F 4F                 CLR   FCBFLG,U                      FORCE GET/PUT TO 'PUT'
0423   C302 C6 03                 LDB   #$03                          DSKCON WRITE OP CODE
0424   C304 8D 33                 BSR   LC339                         GO WRITE A SECTOR - SAVE 'PUT' DATA ON DISK
0425                       * CONVERT THE SECTOR OFFSET TO A GRANULE AND SECTOR NUMBER
0426   C306 EC 61          LC306  LDD   $01,S                         * GET THE NUMBER OF SECTORS TO THE START OF
0427   C308 BD C7 54              JSR   LC754                         * THIS RECORD NUMBER AND CONVERT THEM TO A GRANULE OFFSET
0428   C30B 34 04                 PSHS  B                             SAVE GRANULE OFFSET ON THE STACK
0429   C30D BD C7 49              JSR   LC749                         MULTIPLY GRANULE NUMBER X 9 - CONVERT TO NUMBER OF SECTORS
0430   C310 50                    NEGB                                * NEGATE LS BYTE OF GRANULE OFFSET AND ADD THE
0431   C311 EB 63                 ADDB  $03,S                         * LS BYTE OF SECTOR OFFSET - ACCB = SECTOR
0432                       *                                          * NUMBER (0-8) CORRESPONDING TO THE SECTOR NUMBER WITHIN A
0433                       *                                          * GRANULE OF THE LAST SECTOR OF THE SECTOR OFFSET
0434   C313 5C                    INCB                                = ADD ONE - SECTORS SAVED IN THE FCB; START
0435   C314 E7 44                 STB   FCBSEC,U                      = AT 1 NOT 0 - SAVE IT IN THE FCB
0436   C316 E6 42                 LDB   FCBFGR,U                      GET FIRST GRANULE IN FILE
0437   C318 BD C7 25              JSR   LC725                         POINT X TO FAT
0438   C31B 33 06                 LEAU  FATCON,X                      POINT U TO FAT DATA
0439   C31D A6 E4                 LDA   ,S                            GET NUMBER OF GRANULES OFFSET TO RECORD
0440   C31F 4C                    INCA                                ADD ONE (COMPENSATE FOR DECA BELOW)
0441   C320 30 C4          LC320  LEAX  ,U                            POINT X TO FAT DATA
0442   C322 3A                    ABX                                 POINT X TO CORRECT GRANULE
0443   C323 4A                    DECA                                DECREMENT GRANULE COUNTER
0444   C324 27 37                 BEQ   LC35D                         BRANCH IF CORRECT GRANULE FOUND
0445   C326 E7 E4                 STB   ,S                            SAVE GRANULE ADDRESS ON STACK
0446   C328 E6 84                 LDB   ,X                            GET NEXT GRANULE IN FILE
0447   C32A C1 C0                 CMPB  #$C0                          LAST GRANULE IN FILE?
0448   C32C 25 F2                 BLO   LC320                         NO - KEEP LOOKING
0449
0450                       * THE GRANULE BEING SEARCHED FOR IS NOT PRESENTLY DEFINED IN THIS RANDOM FILE
0451   C32E E6 E4                 LDB   ,S                            GET OFFSET TO LAST GRANULE
0452   C330 0D D8                 TST   VD8                           * CHECK GET/PUT FLAG
0453   C332 26 14                 BNE   LC348                         * AND BRANCH IF PUT
0454   C334 C6 2E          LC334  LDB   #2*23                         'INPUT PAST END OF FILE' ERROR
0455   C336 7E AC 46              JMP   LAC46                         JUMP TO ERROR HANDLER
0456   C339 30 C8 19       LC339  LEAX  FCBCON,U                      POINT X TO FCB DATA BUFFER
0457
0458                       * READ/WRITE A SECTOR. ENTER WITH OP CODE IN ACCB, BUFFER PTR IN X
0459   C33C D7 EA          LC33C  STB   DCOPC                         SAVE DSKCON OPERATION CODE VARIABLE
0460   C33E 9F EE                 STX   DCBPT                         SAVE DSKCON LOAD BUFFER VARIABLE
0461   C340 30 C4                 LEAX  ,U                            POINT X TO FCB
0462   C342 BD C7 33              JSR   LC733                         CONVERT FCB TRACK AND SECTOR TO DSKCON VARIABLES
0463   C345 7E D5 FF              JMP   LD5FF                         READ/WRITE A TRACK OR SECTOR
0464
0465                       * 'PUT' DATA INTO A GRANULE NOT PRESENTLY INCLUDED IN THIS FILE
0466   C348 34 12          LC348  PSHS  X,A                           SAVE GRANULE COUNTER AND POINTER TO LAST USED GRANULE
0467   C34A BD C7 8F              JSR   LC78F                         FIND FIRST FREE GRANULE IN FAT
0468   C34D 1F 89                 TFR   A,B                           SAVE FREE GRANULE NUMBER IN ACCB
0469   C34F 35 42                 PULS  A,U                           PULL LAST GRANULE POINTER AND COUNTER OFF OF STACK
0470   C351 E7 C4                 STB   ,U                            SAVE NEWLY FOUND GRANULE NUMBER IN ADDRESS OF LAST GRANULE
0471   C353 4A                    DECA                                DECREMENT GRANULE COUNTER
0472   C354 26 F2                 BNE   LC348                         GET ANOTHER GRANULE IF NOT DONE
0473   C356 34 14                 PSHS  X,B                           SAVE POINTER TO LAST GRANULE AND OFFSET
0474   C358 BD C6 F1              JSR   LC6F1                         WRITE FAT TO DISK
0475   C35B 35 14                 PULS  B,X                           RESTORE POINTER AND OFFSET
0476
0477                       * WHEN CORRECT GRANULE IS FOUND, FIND THE RIGHT SECTOR
0478   C35D 32 61          LC35D  LEAS  $01,S                         REMOVE GRAN NUMBER FROM STACK
0479   C35F DE F1                 LDU   FCBTMP                        POINT U TO FCB
0480   C361 E7 43                 STB   FCBCGR,U                      SAVE CURRENT GRANULE IN FCB
```

```
0481   C363 86 FF                  LDA   #$FF                        *SET FCBSOF,U TO ILLEGAL SECTOR OFFSET WHICH WILL
0482   C365 A7 4D                  STA   FCBSOF,U                    *FORCE NEW SECTOR DATA TO BE READ IN
0483   C367 A6 84                  LDA   ,X                          GET CURRENT GRANULE
0484   C369 81 C0                  CMPA  #$C0                        IS IT THE LAST GRANULE?
0485   C36B 25 27                  BLO   LC394                       NO
0486   C36D 84 3F                  ANDA  #$3F                        MASK OFF LAST GRANULE FLAG BITS
0487   C36F A1 44                  CMPA  FCBSEC,U                    * COMPARE CALCULATED SECTOR TO CURRENT SECTOR IN FCB
0488   C371 24 21                  BHS   LC394                       * AND BRANCH IF CALCULATED SECTOR IS > LAST SECTOR IN FILE
0489   C373 96 D8                  LDA   VD8                         = CHECK GET/PUT FLAG: IF 'GET' THEN 'INPUT
0490   C375 27 BD                  BEQ   LC334                       = PAST END OF FILE' ERROR
0491   C377 A6 44                  LDA   FCBSEC,U                    * GET CURRENT SECTOR NUMBER FROM FCB,
0492   C379 8A C0                  ORA   #$C0                        * OR IN THE LAST GRANULE FLAG BITS
0493   C37B A7 84                  STA   ,X                          * AND SAVE IN FAT
0494   C37D BD C5 7C               JSR   LC57C                       WRITE FAT TO DISK IF NECESSARY
0495   C380 AE 49                  LDX   FCBRLN,U                    * GET RECORD LENGTH AND CHECK TO
0496   C382 8C 01 00               CMPX  #SECLEN                     * SEE IF IT IS SECLEN (EXACTLY ONE SECTOR)
0497   C385 26 08                  BNE   LC38F                       BRANCH IF IT IS NOT EXACTLY ONE SECTOR
0498   C387 AC C8 13               CMPX  FCBLST,U                    =BRANCH IF THE NUMBER OF BYTES IN THE LAST SECTOR
0499   C38A 27 08                  BEQ   LC394                       =IS SET TO ONE SECTOR (SECLEN)
0500   C38C 86 81                  LDA   #$81                        *SET THE PRESAVED FLAG (BIT15) AND FORCE
0501   C38E 21 4F           LC38E  BRN   LC3DF                       *THE NUMBER OF BYTES IN LAST SECTOR TO 256
0502   C38F 4F              LC38F  CLRA                              SET THE NUMBER OF BYTES IN LAST SECTOR TO ZERO
0503   C390 5F                     CLRB                              CLEAR LS BYTE OF ACCD
0504   C391 ED C8 13               STD   FCBLST,U                    SAVE THE NUMBER OF BYTES IN LAST SECTOR
0505   C394 C6 02           LC394  LDB   #$02                        DSKCON READ OP CODE
0506   C396 AE 49                  LDX   FCBRLN,U                    * GET RECORD LENGTH AND COMPARE
0507   C398 8C 01 00               CMPX  #SECLEN                     * IT TO SECLEN - EXACTLY ONE SECTOR
0508   C39B 26 0D                  BNE   LC3AA                       BRANCH IF NOT EXACTLY ONE SECTOR LONG
0509   C39D 32 67                  LEAS  $07,S                       CLEAN UP STACK
0510   C39F AE 4B                  LDX   FCBBUF,U                    POINT X TO START OF RANDOM FILE BUFFER
0511   C3A1 96 D8                  LDA   VD8                         * CHECK GET/PUT FLAG AND
0512   C3A3 27 02                  BEQ   LC3A7                       * BRANCH IF GET
0513   C3A5 C6 03                  LDB   #$03                        DSKCON WRITE OP CODE
0514 > C3A7 7E C3 3C        LC3A7  JMP   LC33C                       READ/WRITE A SECTOR
0515 > C3AA BD C3 39        LC3AA  JSR   LC339                       READ A SECTOR INTO FCB DATA BUFFER
0516   C3AD 35 14                  PULS  B,X                         * GET BACK THE BYTE OFFSET TO RECORD: X = NUMBER OF
0517                        *                                       * SECTORS; ACCB = BYTE POINTER IN SECTOR
0518   C3AF AF 4D                  STX   FCBSOF,U                    SAVE SECTOR OFFSET IN FCB
0519   C3B1 34 04           LC3B1  PSHS  B                           SAVE BYTE OFFSET ON STACK
0520   C3B3 BD C7 25               JSR   LC725                       POINT X TO FILE ALLOCATION TABLE
0521   C3B6 30 06                  LEAX  FATCON,X                    MOVE X TO FAT DATA
0522   C3B8 E6 43                  LDB   FCBCGR,U                    GET CURRENT GRANULE NUMBER
0523   C3BA 3A                     ABX                               POINT X TO PROPER GRANULE IN FAT
0524   C3BB A6 84                  LDA   ,X                          * GET CURRENT GRANULE AND CHECK TO
0525   C3BD 81 C0                  CMPA  #$C0                        * SEE IF IT IS LAST GRANULE
0526   C3BF 25 24                  BLO   LC3E5                       BRANCH IF THIS GRANULE IS < LAST GRANULE
0527   C3C1 84 3F                  ANDA  #$3F                        MASK OFF LAST GRANULE FLAG BITS
0528   C3C3 A1 44                  CMPA  FCBSEC,U                    * COMPARE LAST SECTOR USED IN GRANULE TO
0529   C3C5 26 1E                  BNE   LC3E5                       * CALCULATED SECTOR; BRANCH IF NOT EQUAL
0530   C3C7 EC C8 13               LDD   FCBLST,U                    GET NUMBER OF BYTES IN LAST SECTOR
0531   C3CA 84 7F                  ANDA  #$7F                        MASK OFF PRESAVED FLAG (BIT 15)
0532   C3CC 34 06                  PSHS  B,A                         SAVE NUMBER OF BYTES IN LAST SECTOR ON STACK
0533   C3CE 4F                     CLRA                              * LOAD ACCB WITH THE BYTE OFFSET TO CURRENT
0534   C3CF E6 62                  LDB   $02,S                       * RECORD AND ADD THE REMAINING RECORD LENGTH
0535   C3D1 E3 63                  ADDD  $03,S                       * TO IT - ACCD = END OF RECORD OFFSET
0536   C3D3 10 A3 E1               CMPD  ,S++                        =COMPARE THE END OF RECORD OFFSET TO THE NUMBER OF
0537   C3D6 23 0D                  BLS   LC3E5                       =BYTES USED IN THE LAST SECTOR
0538   C3D8 0D D8                  TST   VD8                         * CHECK GET/PUT FLAG AND BRANCH IF 'GET'
0539   C3DA 10 27 FF 56            LBEQ  LC334                       * TO 'INPUT PAST END OF FILE' ERROR
0540
0541                        * IF LAST USED SECTOR, CALCULATE HOW MANY BYTES ARE USED
0542                        * IF DATA IS BEING 'PUT' PASTH THE CURRENT END OF FILE
0543   C3DE 84 01                  ANDA  #$01
0544   C3E0 8A 80                  ORA   #$80                        * SET PRE-SAVED FLAG BIT - ALL PUT RECORDS ARE
0545                        *                                       * WRITTEN TO DISK BEFORE LEAVING 'PUT'
0546   C3E2 ED C8 13               STD   FCBLST,U                    SAVE NUMBER OF BYTES USED IN LAST SECTOR
0547   C3E5 35 04           LC3E5  PULS  B                           PULL BYTE OFFSET OFF OF THE STACK
0548   C3E7 30 C8 19               LEAX  FCBCON,U                    POINT X TO FCB DATA BUFFER
0549   C3EA 3A                     ABX                               MOVE X TO START OF RECORD
0550   C3EB EE 62                  LDU   $02,S                       POINT U TO CURRENT POSITION IN RANDOM FILE BUFFER
0551   C3ED 34 04                  PSHS  B                           SAVE BYTE OFFSET ON STACK
0552   C3EF 86 FF                  LDA   #-1                         * CONVERT ACCD INTO A NEGATIVE 2 BYTE NUMBER
0553                        *                                       * REPRESENTING THE REMAINING UNUSED BYTES IN THE SECTOR
0554   C3F1 E3 61                  ADDD  $01,S                       * ADD TEMPORARY RECORD LENGTH COUNTER (SUBTRACT
0555                        *                                       * REMAINING BYTES FROM TEMPORARY RECORD LENGTH)
0556   C3F3 24 07                  BHS   LC3FC                       BRANCH IF THERE ARE ENOUGH UNUSED BYTES TO FINISH THE RECORD
0557   C3F5 ED 61                  STD   $01,S                       SAVE NEW TEMPORARY RECORD LENGTH COUNTER
0558   C3F7 35 04                  PULS  B                           RESTORE BYTE COUNTER
0559   C3F9 50                     NEGB                              * NEGATE IT - ACCB = THE NUMBER OF BYTES
0560                        *                                       * AVAILABLE TO A RECORD IN THIS SECTOR
0561   C3FA 20 08                  BRA   LC404                       MOVE THE DATA
0562
0563                        * BRANCH HERE IF REMAINING RECORD LENGTH WILL FIT IN
0564                        * WHAT'S LEFT OF THE CURRENTLY SELECTED SECTOR
0565   C3FC E6 62           LC3FC  LDB   $02,S                       GET REMAINING RECORD LENGTH
0566   C3FE 6F 61                  CLR   $01,S                       * CLEAR THE TEMPORARY RECORD LENGTH
0567   C400 6F 62                  CLR   $02,S                       * COUNTER ON THE STACK
0568   C402 32 61                  LEAS  $01,S                       PURGE BYTE OFFSET FROM STACK
0569   C404 96 D8           LC404  LDA   VD8                         * CHECK GET/PUT FLAG AND
0570   C406 27 02                  BEQ   LC40A                       * BRANCH IF GET
0571   C408 1E 13                  EXG   X,U                         SWAP SOURCE AND DESTINATION POINTERS
0572   C40A BD A5 9A        LC40A  JSR   LA59A                       TRANSFER DATA FROM SOURCE TO DESTINATION BUFFERS
0573   C40D EF 62                  STU   $02,S                       SAVE NEW TEMP RECORD POINTER ON THE STACK (GET)
0574
0575                        * MOVE DATA FROM FCB DATA BUFFER TO THE RANDOM FILE BUFFER IF 'GET'
0576                        * OR FROM RANDOM FILE BUFFER TO FCB DATA BUFFER IF 'PUT'
```

```
0577   C40F DE F1                 LDU   FCBTMP                  POINT U TO FCB
0578   C411 96 D8                 LDA   VD8                     * CHECK GET/PUT FLAG AND
0579   C413 27 04                 BEQ   LC419                   * BRANCH IF GET
0580   C415 A7 4F                 STA   FCBFLG,U                SAVE 'PUT' FLAG IN THE FCB
0581   C417 AF 62                 STX   $02,S                   SAVE NEW TEMPORARY RECORD POINTER ON STACK (PUT)
0582   C419 AE 4D          LC419  LDX   FCBSOF,U                * GET SECTOR OFFSET COUNTER AND
0583   C41B 30 01                 LEAX  $01,X                   * ADD ONE TO IT
0584   C41D 5F                    CLRB                          SET BYTE OFFSET = 0
0585   C41E EE E4                 LDU   ,S                      * CHECK THE LENGTH OF THE TEMPORARY RECORD LENGTH
0586   C420 10 26 FE C4           LBNE  LC2E8                   * COUNTER AND KEEP MOVING DATA IF <> 0
0587   C424 35 96                 PULS  A,B,X,PC                * PULL TEMPORARY RECORD LENGTH AND
0588                       *                                    * BUFFER ADDRESS OFF STACK AND RETURN
0589
0590                       * OPEN RAM HOOK
0591   C426 32 62          DVEC0  LEAS  $02,S                   PULL RETURN ADDRESS OFF OF THE STACK
0592   C428 BD B1 56              JSR   LB156                   EVALUATE AN EXPRESSION
0593   C42B BD B6 A4              JSR   LB6A4                   *GET MODE(I,O,R) - FIRST BYTE OF STRING EXPRESSION
0594   C42E 34 04                 PSHS  B                       *AND SAVE IT ON STACK
0595   C430 BD A5 A2              JSR   LA5A2                   GET DEVICE NUMBER
0596   C433 5D                    TSTB                          SET FLAGS
0597   C434 10 2F E1 CB           LBLE  LA603                   BRANCH IF NOT A DISK FILE
0598   C438 35 02                 PULS  A                       GET MODE
0599   C43A 34 06                 PSHS  B,A                     SAVE MODE AND DEVICE NUMBER (FILE NUMBER)
0600   C43C 0F 6F                 CLR   DEVNUM                  SET DEVICE NUMBER TO SCREEN
0601   C43E BD B2 6D              JSR   SYNCOMMA                SYNTAX CHECK FOR COMMA
0602   C441 8E C2 94              LDX   #DATEXT                 POINT TO 'DAT' FOR EXTENSION
0603   C444 BD C8 8A              JSR   LC88A                   GET FILENAME FROM BASIC
0604   C447 CC 01 FF              LDD   #$01FF                  DEFAULT DISK FILE TYPE AND ASCII FLAG
0605   C44A FD 09 57              STD   DFLTYP                  SAVE DEFAULT VALUES: DATA, ASCII
0606   C44D 8E 01 00              LDX   #SECLEN                 DEFAULT RECORD LENGTH - 1 PAGE
0607   C450 9D A5                 JSR   GETCCH                  GET CHAR FROM BASIC
0608   C452 27 08                 BEQ   LC45C                   BRANCH IF END OF LINE
0609   C454 BD B2 6D              JSR   SYNCOMMA                SYNTAX CHECK FOR COMMA
0610   C457 BD B3 E6              JSR   LB3E6                   EVALUATE EXPRESSION
0611   C45A 9E 52                 LDX   FPA0+2                  GET EVALUATED EXPRESSION
0612   C45C BF 09 7C       LC45C  STX   DFFLEN                  RECORD LENGTH
0613   C45F 10 27 EF E7           LBEQ  LB44A                   IF = 0, THEN 'ILLEGAL FUNCTION CALL'
0614   C463 BD A5 C7              JSR   LA5C7                   ERROR IF ANY FURTHER CHARACTERS ON LINE
0615   C466 35 06                 PULS  A,B                     GET MODE AND FILE NUMBER
0616
0617                       * OPEN DISK FILE FOR READ OR WRITE
0618   C468 34 02          LC468  PSHS  A                       SAVE MODE ON STACK
0619   C46A BD C7 19              JSR   LC719                   POINT X TO FCB FOR THIS FILE
0620   C46D 10 26 E1 AB           LBNE  LA61C                   'FILE ALREADY OPEN' ERROR IF FILE OPEN
0621   C471 9F F1                 STX   FCBTMP                  SAVE FILE BUFFER POINTER
0622   C473 BD C7 6D              JSR   LC76D                   MAKE SURE FILE ALLOC TABLE IS VALID
0623   C476 BD C6 5F              JSR   LC65F                   SCAN DIRECTORY FOR 'FILENAME.EXT'
0624   C479 35 04                 PULS  B                       GET MODE
0625   C47B 86 10                 LDA   #INPFIL                 INPUT TYPE FILE
0626   C47D 34 02                 PSHS  A                       SAVE FILE TYPE ON STACK
0627   C47F C1 49                 CMPB  #'I'                    INPUT MODE?
0628   C481 26 1F                 BNE   LC4A2                   BRANCH IF NOT
0629
0630                       * OPEN A SEQUENTIAL FILE FOR INPUT
0631   C483 BD C6 B8              JSR   LC6B8                   CHECK TO SEE IF DIRECTORY MATCH IS FOUND
0632   C486 BD C7 D7              JSR   LC7D7                   CHECK TO SEE IF FILE ALREADY OPEN
0633   C489 BE 09 74              LDX   V974                    GET RAM DIRECTORY BUFFER
0634   C48C EC 0B                 LDD   DIRTYP,X                GET FILE TYPE AND ASCII FLAG
0635   C48E FD 09 57              STD   DFLTYP                  SAVE IN RAM IMAGE
0636   C491 8D 6D                 BSR   LC500                   INITIALIZE FILE BUFFER CONTROL BLOCK
0637   C493 BD C5 FA              JSR   LC5FA                   GO FILL DATA BUFFER
0638   C496 BD C7 25       LC496  JSR   LC725                   POINT X TO PROPER FILE ALLOCATION TABLE
0639   C499 6C 00                 INC   FAT0,X                  ADD ONE TO FAT ACTIVE FILE COUNTER
0640   C49B 9E F1                 LDX   FCBTMP                  GET FILE BUFFER POINTER
0641   C49D 35 02                 PULS  A                       GET FILE TYPE
0642   C49F A7 00                 STA   FCBTYP,X                SAVE IT IN FCB
0643   C4A1 39                    RTS
0644   C4A2 68 E4          LC4A2  ASL   ,S                      SET FILE TYPE TO OUTPUT
0645   C4A4 C1 4F                 CMPB  #'O'                    FILE MODE = OUTPUT?
0646   C4A6 26 1A                 BNE   LC4C2                   BRANCH IF NOT
0647
0648                       * OPEN A SEQUENTIAL FILE FOR OUTPUT
0649   C4A8 7D 09 73              TST   V973                    DOES FILE EXIST ON DIRECTORY?
0650   C4AB 27 0F                 BEQ   LC4BC                   BRANCH IF NOT
0651   C4AD BD C6 CF              JSR   LC6CF                   KILL THE OLD FILE
0652   C4B0 B6 09 73              LDA   V973                    * GET DIRECTORY SECTOR NUMBER OF OLD FILE AND
0653   C4B3 B7 09 77              STA   V977                    * SAVE IT AS FIRST FREE DIRECTORY ENTRY
0654   C4B6 BE 09 74              LDX   V974                    =GET RAM DIRECTORY IMAGE OF OLD FILE AND
0655   C4B9 BF 09 78              STX   V978                    =SAVE IT AS FIRST FREE DIRECTORY ENTRY
0656
0657   C4BC 8D 7C          LC4BC  BSR   LC53A                   SET UP NEW DIRECTORY ENTRY ON DISK
0658   C4BE 8D 4B                 BSR   LC50B                   INITIALIZE FILE BUFFER
0659   C4C0 20 D4                 BRA   LC496                   FLAG AND MAP FCB AS BEING USED
0660   C4C2 C1 52          LC4C2  CMPB  #'R'                    FILE MODE = R (RANDOM)?
0661   C4C4 27 06                 BEQ   LC4CC                   BRANCH IF SO
0662   C4C6 C1 44                 CMPB  #'D'                    FILE MODE = D (DIRECT)?
0663   C4C8 10 26 E1 4A           LBNE  LA616                   'BAD FILE MODE' ERROR IF NOT
0664
0665                       * OPEN A RANDOM/DIRECT FILE
0666   C4CC 68 E4          LC4CC  ASL   ,S                      SET FILE TYPE TO DIRECT
0667   C4CE FC 09 48              LDD   RNBFAD                  * GET ADDRESS OF RANDOM FILE BUFFER AREA
0668   C4D1 34 06                 PSHS  B,A                     * AND SAVE IT ON THE STACK
0669   C4D3 F3 09 7C              ADDD  DFFLEN                  ADD THE RECORD LENGTH
0670   C4D6 25 06                 BLO   LC4DE                   'OB' ERROR IF SUM > $FFFF
0671   C4D8 10 B3 09 4A           CMPD  FCBADR                  IS IT > THAN FCB DATA AREA?
0672   C4DC 23 05                 BLS   LC4E3                   BRANCH IF NOT
```

```
0673   C4DE C6 3A          LC4DE   LDB   #2*29                       'OUT OF BUFFER SPACE' ERROR
0674   C4E0 7E AC 46               JMP   LAC46                       JUMP TO ERROR HANDLER
0675   C4E3 34 06          LC4E3   PSHS  B,A                         SAVE END OF RANDOM BUFFER ON STACK
0676   C4E5 7D 09 73               TST   V973                        DID THIS FILE EXIST
0677   C4E8 26 02                  BNE   LC4EC                       BRANCH IF SO
0678   C4EA 8D 4E                  BSR   LC53A                       SET UP NEW FILE IN DIRECTORY
0679   C4EC 8D 12          LC4EC   BSR   LC500                       INITIALIZE FILE BUFFER
0680   C4EE 63 0D                  COM   FCBSOF,X                    * SET FCBSOF,X TO $FF (ILLEGAL SECTOR OFFSET) WHICH WILL
0681                       *                                         * FORCE NEW SECTOR DATA TO BE READ IN DURING GET/PUT
0682   C4F0 6C 08                  INC   FCBREC+1,X                  INITIALIZE RECORD NUMBER = 1
0683   C4F2 35 46                  PULS  A,B,U                       U = START OF RANDOM FILE BUFFER AREA, ACCD = END
0684   C4F4 FD 09 48               STD   RNBFAD                      SAVE NEW START OF RANDOM FILE BUFFER AREA
0685   C4F7 EF 0B                  STU   FCBBUF,X                    SAVE BUFFER START IN FCB
0686   C4F9 FE 09 7C               LDU   DFFLEN                      * GET RANDOM FILE RECORD LENGTH
0687   C4FC EF 09                  STU   FCBRLN,X                    * AND SAVE IT IN FCB
0688   C4FE 20 96                  BRA   LC496                       SET FAT FLAG, SAVE FILE TYPE IN FCB
0689
0690                       * INITIALIZE FCB DATA FOR INPUT
0691   C500 8D 09          LC500   BSR   LC50B                       INITIALIZE FCB
0692   C502 FE 09 74               LDU   V974                        GET RAM DIRECTORY IMAGE
0693   C505 EE 4E                  LDU   DIRLST,U                    *GET NUMBER OF BYTES IN LAST SECTOR OF FILE
0694   C507 EF 88 13               STU   FCBLST,X                    *SAVE IT IN FCB
0695   C50A 39                     RTS
0696                       * INITIALIZE FILE CONTROL BLOCK
0697   C50B 9E F1          LC50B   LDX   FCBTMP                      GET CURRENT FILE BUFFER
0698   C50D C6 19                  LDB   #FCBCON                     CLEAR FCB CONTROL BYTES
0699   C50F 6F 80          LC50F   CLR   ,X+                         CLEAR A BYTE
0700   C511 5A                     DECB                              DECREMENT COUNTER
0701   C512 26 FB                  BNE   LC50F                       BRANCH IF NOT DONE
0702   C514 9E F1                  LDX   FCBTMP                      GET CURRENT FILE BUFFER ADDRESS BACK
0703   C516 96 EB                  LDA   DCDRV                       *GET CURRENT DRIVE NUMBER AND
0704   C518 A7 01                  STA   FCBDRV,X                    *SAVE IT IN FCB
0705   C51A B6 09 76               LDA   V976                        =GET FIRST GRANULE -
0706   C51D A7 02                  STA   FCBFGR,X                    =SAVE IT AS THE STARTING GRANULE NUMBER AND
0707   C51F A7 03                  STA   FCBCGR,X                    =SAVE IT AS CURRENT GRANULE NUMBER
0708   C521 F6 09 73               LDB   V973                        GET DIRECTORY SECTOR NUMBER
0709   C524 C0 03                  SUBB  #$03                        SUBTRACT 3 - DIRECTORY SECTORS START AT 3
0710   C526 58                     ASLB                              * MULTIPLY SECTORS
0711   C527 58                     ASLB                              * BY 8 (8 DIRECTORY
0712   C528 58                     ASLB                              * ENTRIES PER SECTOR)
0713   C529 34 04                  PSHS  B                           SAVE SECTOR OFFSET
0714   C52B FC 09 74               LDD   V974                        GET RAM DIRECTORY IMAGE
0715   C52E 83 06 00               SUBD  #DBUF0                      SUBTRACT RAM OFFSET
0716   C531 86 08                  LDA   #$08                        8 DIRECTORY ENTRIES/SECTOR
0717   C533 3D                     MUL                               NOW ACCA CONTAINS 0-7
0718   C534 AB E0                  ADDA  ,S+                         ACCA CONTAINS DIRECTORY ENTRY (0-71)
0719   C536 A7 88 12               STA   FCBDIR,X                    SAVE DIRECTORY ENTRY NUMBER
0720   C539 39                     RTS
0721
0722                       * SET UP DIRECTORY AND UPDATE FILE ALLOCATION TABLE ENTRY IN FIRST UNUSED SECTOR
0723   C53A C6 38                  LDB   #28*2                       'DISK FULL' ERROR
0724   C53C B6 09 77               LDA   V977                        GET SECTOR NUMBER OF FIRST EMPTY DIRECTORY ENTRY
0725   C53F 10 27 E7 03            LBEQ  LAC46                       'DISK FULL' ERROR IF NO EMPTY DIRECTORY ENTRIES
0726   C543 B7 09 73               STA   V973                        SAVE SECTOR NUMBER OF FIRST EMPTY DIRECTORY ENTRY
0727   C546 97 ED                  STA   DSEC                        SAVE SECTOR NUMBER IN DSKCON REGISTER
0728   C548 C6 02                  LDB   #$02                        READ OP CODE
0729   C54A D7 EA                  STB   DCOPC                       SAVE IN DSKCON REGISTER
0730   C54C BD D5 FF               JSR   LD5FF                       READ SECTOR
0731   C54F BE 09 78               LDX   V978                        * GET ADDRESS OF RAM IMAGE OF UNUSED DIRECTORY
0732   C552 BF 09 74               STX   V974                        * ENTRY AND SAVE AS CURRENT USED RAM IMAGE
0733   C555 33 84                  LEAU  ,X                          (TFR X,U) POINT U TO DIRECTORY RAM IMAGE
0734   C557 C6 20                  LDB   #DIRLEN                     SET COUNTER TO CLEAR 32 BYTES (DIRECTORY ENTRY)
0735   C559 6F 80          LC559   CLR   ,X+                         CLEAR BYTE
0736   C55B 5A                     DECB                              DECREMENT COUNTER
0737   C55C 26 FB                  BNE   LC559                       CONTINUE IF NOT DONE
0738   C55E 8E 09 4C               LDX   #DNAMBF                     POINT TO FILENAME AND EXTENSION RAM IMAGE
0739   C561 C6 0B                  LDB   #11                         11 BYTES IN FILENAME AND EXTENSION
0740   C563 BD A5 9A               JSR   LA59A                       MOVE B BYTES FROM X TO U
0741   C566 FC 09 57               LDD   DFLTYP                      GET FILE TYPE AND ASCII FLAG
0742   C569 ED 40                  STD   ,U                          SAVE IN RAM IMAGE
0743   C56B C6 21                  LDB   #33                         FIRST GRANULE TO CHECK
0744   C56D BD C7 8F               JSR   LC78F                       FIND THE FIRST FREE GRANULE
0745   C570 B7 09 76               STA   V976                        SAVE IN RAM
0746   C573 A7 42                  STA   $02,U                       SAVE IN RAM IMAGE OF DIRECTORY TRACK
0747   C575 C6 03                  LDB   #$03                        * GET WRITE OPERATION CODE AND SAVE
0748   C577 D7 EA                  STB   DCOPC                       * IT IN DSKCON REGISTER
0749   C579 BD D5 FF               JSR   LD5FF                       GO WRITE A SECTOR IN DIRECTORY
0750   C57C 34 56          LC57C   PSHS  U,X,B,A                     SAVE REGISTERS
0751   C57E BD C7 25               JSR   LC725                       POINT X TO FILE ALLOCATION TABLE
0752   C581 6C 01                  INC   FAT1,X                      INDICATE NEW DATA IN FILE ALLOC TABLE
0753   C583 A6 01                  LDA   FAT1,X                      GET NEW DATA FLAG
0754   C585 B1 09 7A               CMPA  WFATVL                      * HAVE ENOUGH GRANULES BEEN REMOVED FROM THE FAT TO
0755                       *                                         * CAUSE THE FAT TO BE WRITTEN TO THE DISK
0756   C588 25 03                  BLO   LC58D                       RETURN IF NO NEED TO WRITE OUT ALLOCATION TABLE
0757   C58A BD C6 F1               JSR   LC6F1                       WRITE FILE ALLOCATION SECTOR TO DISK
0758   C58D 35 D6          LC58D   PULS  A,B,X,U,PC                  RESTORE REGISTERS
0759
0760                       * CONSOLE IN RAM VECTOR
0761   C58F 96 6F          DVEC4   LDA   DEVNUM                       GET DEVICE NUMBER
0762   C591 10 2F C7 5C            LBLE  XVEC4                       BRANCH IF NOT DISK FILE
0763   C595 32 62                  LEAS  $02,S                       GET RID OF RETURN ADDRESS
0764   C597 34 14          LC597   PSHS  X,B                         SAVE REGISTERS
0765   C599 0F 70                  CLR   CINBFL                      CLEAR BUFFER NOT EMPTY FLAG
0766   C59B 8E 09 26               LDX   #FCBV1-2                    POINT TO FILE BUFFER VECTOR TABLE
0767   C59E D6 6F                  LDB   DEVNUM                      GET ACTIVE DISK FILE NUMBER
0768   C5A0 58                     ASLB                              TIMES 2 - TWO BYTES PER FCB ADDRESS
```

```
0769   C5A1 AE 85                    LDX   B,X                           NOW X POINTS TO FILE BUFFER
0770   C5A3 E6 84                    LDB   FCBTYP,X                      GET FILE TYPE
0771   C5A5 C1 40                    CMPB  #RANFIL                       IS THIS A RANDOM (DIRECT) FILE?
0772   C5A7 26 16                    BNE   LC5BF                         BRANCH IF NOT
0773
0774                        * GET A BYTE FROM A RANDOM FILE - RETURN CHAR IN ACCA
0775   C5A9 EC 88 15                 LDD   FCBGET,X                      GET THE RECORD COUNTER
0776   C5AC 10 A3 09                 CMPD  FCBRLN,X                      *COMPARE TO RECORD LENGTH AND
0777   C5AF 24 20                    BHS   LC5D1                         *BRANCH TO BUFFER EMPTY IF >= RECORD LENGTH
0778   C5B1 C3 00 01                 ADDD  #$0001                        = ADD ONE TO RECORD POINTER AND
0779   C5B4 ED 88 15                 STD   FCBGET,X                      = SAVE IT IN FCB
0780   C5B7 AE 0B                    LDX   FCBBUF,X                      * POINT X TO START OF RANDOM FILE BUFFER AND
0781   C5B9 30 8B                    LEAX  D,X                           * ADD THE RECORD COUNTER TO IT
0782   C5BB A6 1F                    LDA   -1,X                          GET A CHARACTER FROM THE BUFFER
0783   C5BD 35 94                    PULS  B,X,PC                        RESTORE REGISTERS AND RETURN
0784                        * GET A BYTE FROM A SEQUENTIAL FILE
0785   C5BF E6 88 10       LC5BF     LDB   FCBCFL,X                      * TEST THE CACHE FLAG AND BRANCH IF AN
0786   C5C2 27 08                    BEQ   LC5CC                         * EXTRA CHARACTER HAS NOT BEEN READ FROM FILE
0787   C5C4 A6 88 11                 LDA   FCBCDT,X                      GET THE CACHE CHARACTER
0788   C5C7 6F 88 10                 CLR   FCBCFL,X                      CLEAR THE CACHE FLAG
0789   C5CA 35 94                    PULS  B,X,PC                        RESTORE REGISTERS AND RETURN
0790
0791   C5CC E6 88 17       LC5CC     LDB   FCBDFL,X                      IS ANY DATA LEFT?
0792   C5CF 27 04                    BEQ   LC5D5                         BRANCH IF SO
0793   C5D1 03 70          LC5D1     COM   CINBFL                        SET FLAG TO BUFFER EMPTY
0794   C5D3 35 94                    PULS  B,X,PC
0795
0796   C5D5 E6 05          LC5D5     LDB   FCBCPT,X                      GET CHARACTER POINTER
0797   C5D7 6C 05                    INC   FCBCPT,X                      ADD ONE TO CHARACTER POINTER
0798   C5D9 6A 88 18                 DEC   FCBLFT,X                      DECREMENT NUMBER OF CHARACTERS LEFT IN FILE BUFFER
0799   C5DC 27 06                    BEQ   LC5E4                         IF LAST CHARACTER, GO GET SOME MORE
0800   C5DE 3A                       ABX                                 ADD CHARACTER COUNTER TO X
0801   C5DF A6 88 19                 LDA   FCBCON,X                      GET DATA CHARACTER (SKIP PAST 25 FCB CONTROL BYTES
0802   C5E2 35 94                    PULS  B,X,PC
0803                        * GET A CHARACTER FROM FCB DATA BUFFER - RETURN CHAR IN ACCA
0804   C5E4 34 60          LC5E4     PSHS  U,Y                           SAVE REGISTERS
0805   C5E6 4F                       CLRA                                *
0806   C5E7 33 8B                    LEAU  D,X                           * POINT U TO CORRECT CHARACTER
0807   C5E9 A6 C8 19                 LDA   FCBCON,U                      =GET DATA CHAR (SKIP PAST 25 CONTROL BYTES)
0808   C5EC 34 02                    PSHS  A                             =AND SAVE DATA CHARACTER ON STACK
0809   C5EE 6F 05                    CLR   FCBCPT,X                      RESET CHAR POINTER TO START OF BUFFER
0810   C5F0 A6 01                    LDA   FCBDRV,X                      GET DRIVE NUMBER AND SAVE IT IN
0811   C5F2 97 EB                    STA   DCDRV                         DSKCON VARIABLE
0812   C5F4 8D 04                    BSR   LC5FA                         GO READ A SECTOR - FILL THE BUFFER
0813   C5F6 35 62                    PULS  A,Y,U                         RESTORE REGISTERS AND DATA CHARACTER
0814   C5F8 35 94                    PULS  B,X,PC                        RESTORE REGISTERS AND RETURN
0815                        * REFILL THE FCB INPUT DATA BUFFER FOR SEQUENTIAL FILES
0816   C5FA A6 04          LC5FA     LDA   FCBSEC,X                      GET CURRENT SECTOR NUMBER
0817   C5FC 4C             LC5FC     INCA                                ADD ONE
0818   C5FD 34 02                    PSHS  A                             SAVE NEW SECTOR NUMBER ON THE STACK
0819   C5FF 81 09                    CMPA  #$09                          NINE SECTORS PER GRANULE
0820   C601 23 01                    BLS   LC604                         BRANCH IF <= 9
0821   C603 4F                       CLRA                                SET TO SECTOR ZERO
0822   C604 A7 04          LC604     STA   FCBSEC,X                      SAVE SECTOR NUMBER
0823   C606 E6 03                    LDB   FCBCGR,X                      GET GRANULE NUMBET TO FAT POINTER
0824   C608 33 84                    LEAU  ,X                            POINT U TO FCB (TFR X,U)
0825   C60A BD C7 25                 JSR   LC725                         POINT X TO PROPER FILE ALLOCATION TABLE
0826   C60D 3A                       ABX                                 ADD OLD GRANULE NUMBER TO FAT POINTER
0827   C60E E6 06                    LDB   FATCON,X                      GET GRANULE NUMBER (6 CONTROL BYTES AT FRONT OF FAT)
0828   C610 30 C4                    LEAX  ,U                            POINT X TO FCB
0829   C612 C1 C0                    CMPB  #$C0                          IS CURRENT GRANULE LAST ONE IN FILE?
0830   C614 24 0A                    BHS   LC620                         YES
0831   C616 35 02                    PULS  A                             GET SECTOR NUMBER
0832   C618 80 0A                    SUBA  #10                           WAS IT 10? - OVERFLOW TO NEXT GRANULE IF SO
0833   C61A 26 15                    BNE   LC631                         BRANCH IF NOT
0834   C61C E7 03                    STB   FCBCGR,X                      SAVE NEW GRANULE NUMBER
0835   C61E 20 DC                    BRA   LC5FC                         SET VARIABLES FOR NEW GRANULE
0836   C620 C4 3F          LC620     ANDB  #$3F                          GET NUMBER OF SECTORS USED IN THIS GRANULE
0837   C622 C1 09                    CMPB  #$09                          9 SECTORS / GRANULE
0838   C624 23 05                    BLS   LC62B                         BRANCH IF OK
0839   C626 C6 40          LC626     LDB   #2*32                         'BAD FILE STRUCTURE' ERROR
0840   C628 7E AC 46                 JMP   LAC46                         ERROR DRIVER
0841   C62B E0 E0          LC62B     SUBB  ,S+                           SUBTRACT CURRENT SECTOR NUMBER AND PULS A
0842   C62D 25 21                    BLO   LC650                         BRANCH IF PAST LAST SECTOR
0843   C62F 1F 98                    TFR   B,A                           SECTOR NUMBER TO ACCA
0844   C631 34 02          LC631     PSHS  A                             SAVE SECTOR NUMBER DIFFERENCE
0845   C633 8D 23                    BSR   LC658                         INCREMENT RECORD NUMBER
0846   C635 86 02                    LDA   #$02                          *GET READ OPERATION CODE
0847   C637 97 EA                    STA   DCOPC                         *AND SAVE IT IN DSKCON VARIABLE
0848   C639 BD C7 33                 JSR   LC733                         GET PROPER TRACK AND SECTOR TO DSKCON VARIABLES
0849   C63C 33 88 19                 LEAU  FCBCON,X                      * POINT U TO START OF FCB DATA BUFFER
0850   C63F DF EE                    STU   DCBPT                         * AND SAVE IT IN DSKCON VARIABLE
0851   C641 BD D5 FF                 JSR   LD5FF                         GO READ A SECTOR INTO FCB BUFFER
0852   C644 6F 88 18                 CLR   FCBLFT,X                      NUMBER OF CHARS LEFT IN BUFFER = 256
0853   C647 E6 E0                    LDB   ,S+                           GET SECTOR NUMBER OFF STACK
0854   C649 26 0C                    BNE   LC657                         RETURN IF DATA LEFT; FALL THRU IF LAST SECTOR
0855   C64B EC 88 13                 LDD   FCBLST,X                      GET NUMBER OF BYTES IN THE LAST SECTOR
0856   C64E 26 04                    BNE   LC654                         BRANCH IF SOME BYTES IN LAST SECTOR
0857   C650 5F             LC650     CLRB                                SET NUMBER OF REMAINING BYTES = 256
0858   C651 63 88 17                 COM   FCBDFL,X                      SET DATA LEFT FLAG TO $FF
0859   C654 E7 88 18       LC654     STB   FCBLFT,X                      SAVE THE NUMBER OF CHARS LEFT IN BUFFER
0860   C657 39             LC657     RTS
0861
0862   C658 EE 07          LC658     LDU   FCBREC,X                      GET CURRENT RECORD NUMBER
0863   C65A 33 41                    LEAU  $01,U                         BUMP IT
0864   C65C EF 07                    STU   FCBREC,X                      PUT IT BACK
```

```
0865   C65E 39                    RTS
0866
0867                      * SCAN DIRECTORY FOR FILENAME.EXT FOUND IN DNAMBF. IF FILENAME FOUND,
0868                      * RETURN WITH SECTOR NUMBER IN V973, GRANULE IN V976 AND RAM BUFFER
0869                      * CONTAINING DIRECTORY DATA IN V974. IF DISK IS FULL THEN V973,
0870                      * V977 = 0. THE FIRST UNUSED SECTOR RETURNED IN V977, RAM IMAGE IN V978
0871   C65F 7F 09 73      LC65F   CLR   V973                        CLEAR SECTOR NUMBER
0872   C662 7F 09 77              CLR   V977                        CLEAR TEMP SECTOR COUNTER
0873   C665 CC 11 02              LDD   #$1102                      TRACK 17 (DIRECTORY), READ OPERATION CODE
0874   C668 97 EC                 STA   DCTRK                       SAVE TRACK NUMBER
0875   C66A D7 EA                 STB   DCOPC                       SAVE OPERATION CODE (READ)
0876   C66C C6 03                 LDB   #$03                        READ SECTOR 3 (FIRST DIRECTORY SECTOR)
0877   C66E D7 ED         LC66E   STB   DSEC                        SAVE SECTOR NUMBER IN DSKCON VARIABLE
0878   C670 CE 06 00              LDU   #DBUF0                      *BUFFER AREA NUMBER 0 AS DATA BUFFER - SAVE
0879   C673 DF EE                 STU   DCBPT                       *IN DSKCON VARIABLE
0880   C675 BD D5 FF              JSR   LD5FF                       GO READ A SECTOR
0881   C678 FF 09 74      LC678   STU   V974                        SAVE RAM DIRECTORY BUFFER ADDRESS
0882   C67B 31 C4                 LEAY  ,U                          POINT Y TO DIRECTORY BUFFER
0883   C67D A6 C4                 LDA   ,U                          GET A BYTE FROM BUFFER
0884   C67F 26 28                 BNE   LC6A9                       BRANCH IF NOT ZERO - FILE IS ACTIVE
0885   C681 8D 29                 BSR   LC6AC                       SET UNUSED FILE POINTERS IF ENTRY HAS BEEN KILLED
0886   C683 8E 09 4C      LC683   LDX   #DNAMBF                      POINT TO DISK FILE NAME BUFFER
0887   C686 A6 80         LC686   LDA   ,X+                         *COMPARE THE FILENAME AND EXTENSION
0888   C688 A1 C0                 CMPA  ,U+                         *STORED IN RAM AT DNAMBF TO THE DIRECTORY
0889   C68A 26 0E                 BNE   LC69A                       *ENTRY STORED AT ,U (BRANCH IF MISMATCH)
0890   C68C 8C 09 57              CMPX  #DNAMBF+11                  AT END OF FILE NAME BUFFER?
0891   C68F 26 F5                 BNE   LC686                       BRANCH IF NOT DONE CHECKING FILENAME
0892   C691 F7 09 73              STB   V973                        SAVE SECTOR NUMBER IN DSKCON VARIABLE
0893   C694 A6 42                 LDA   FCBFGR,U                    *GET NUMBER OF FIRST GRANULE IN FILE
0894   C696 B7 09 76              STA   V976                        *AND SAVE IT IN V976
0895   C699 39                    RTS
0896
0897   C69A 33 A8 20      LC69A   LEAU  DIRLEN,Y                    GET NEXT DIRECTORY ENTRY (DIRLEN BYTES PER ENTRY)
0898   C69D 11 83 07 00           CMPU  #DBUF0+SECLEN               AT END OF BUFFER?
0899   C6A1 26 D5                 BNE   LC678                       CHECK NEXT ENTRY IF NOT AT END
0900   C6A3 5C                    INCB                              NEXT SECTOR
0901   C6A4 C1 0B                 CMPB  #11                         11 SECTORS MAX IN DIRECTORY
0902   C6A6 23 C6                 BLS   LC66E                       BRANCH IF MORE SECTORS
0903   C6A8 39                    RTS
0904
0905   C6A9 43            LC6A9   COMA                              COMPLEMENT FIRST BYTE IN DIRECTORY EMTRY
0906   C6AA 26 D7                 BNE   LC683                       BRANCH IF FILE IS ACTIVE - FALL THRU IF NOT USED
0907
0908                      * SET POINTERS FOR FIRST UNUSED DIRECTORY ENTRY
0909   C6AC B6 09 77      LC6AC   LDA   V977                        UNUSED ENTRY ALREADY FOUND?
0910   C6AF 26 06                 BNE   DVEC12                      RETURN IF UNUSED ENTRY ALREADY FOUND
0911   C6B1 F7 09 77              STB   V977                        SECTOR CONTAINING THIS DIRECTORY ENTRY
0912   C6B4 FF 09 78              STU   V978                        POINTS TO RAM AREA WHERE DIRECTORY DATA IS STORED
0913   C6B7 39            DVEC12  RTS
0914
0915   C6B8 C6 34         LC6B8   LDB   #2*26                       'NE' ERROR
0916   C6BA 7D 09 73              TST   V973                        WAS A DIRECTORY MATCH FOUND?
0917   C6BD 26 F8                 BNE   DVEC12                      RETURN IF FOUND
0918   C6BF 7E AC 46              JMP   LAC46                       JUMP TO ERROR HANDLER IF NOT FOUND
0919
0920                      * KILL COMMAND
0921   C6C2 BD C8 87      KILL    JSR   LC887                       GET FILENAME.EXT FROM BASIC
0922   C6C5 BD A5 C7              JSR   LA5C7                       'SYNTAX' ERROR IF MORE CHARACTERS ON LINE
0923   C6C8 BD C7 6D              JSR   LC76D                       GET VALID FAT DATA
0924   C6CB 8D 92                 BSR   LC65F                       TEST FOR FILE NAME MATCH IN DIRECTORY
0925   C6CD 8D E9                 BSR   LC6B8                       MAKE SURE THE FILE EXISTED
0926   C6CF 86 FF         LC6CF   LDA   #$FF                        * MATCH FILE TYPE = $FF; THIS WILL CAUSE AN 'AO'
0927                      *                                         * ERROR TO BE GENERATED IF ANY FILE TYPE IS OPEN
0928   C6D1 BD C7 D7              JSR   LC7D7                       CHECK TO MAKE SURE FILE IS NOT OPEN
0929   C6D4 BE 09 74              LDX   V974                        *GET RAM IMAGE OF DIRECTORY
0930   C6D7 6F 84                 CLR   DIRNAM,X                    *AND ZERO FIRST BYTE - KILL FILE
0931   C6D9 C6 03                 LDB   #$03                        =WRITE OPERATION CODE - SAVE
0932   C6DB D7 EA                 STB   DCOPC                       =IT IN DSKCON VARIABLE
0933   C6DD BD D5 FF              JSR   LD5FF                       WRITE A SECTOR
0934   C6E0 E6 0D                 LDB   DIRGRN,X                    GET NUMBER OF FIRST GRANULE IN FILE
0935   C6E2 8D 41         LC6E2   BSR   LC725                       POINT X TO PROPER FILE ALLOCATION TABLE
0936   C6E4 30 06                 LEAX  FATCON,X                    SKIP 6 CONTROL BYTES
0937   C6E6 3A                    ABX                               POINT TO CORRECT ENTRY
0938   C6E7 E6 84                 LDB   ,X                          GET NEXT GRANULE
0939   C6E9 86 FF                 LDA   #$FF                        *GET FREE GRANULE FLAG AND
0940   C6EB A7 84                 STA   ,X                          *MARK GRANULE AS FREE
0941   C6ED C1 C0                 CMPB  #$C0                        WAS THIS THE LAST GRANULE?
0942   C6EF 25 F1                 BLO   LC6E2                       * KEEP FREEING GRANULES IF NOT LAST ONE
0943                      *                                         * WRITE FILE ALLOCATION SECTOR TO DIRECTORY - DO NOT WRITE
0944                      *                                         * THE SIX CONTROL BYTES AT THE START OF THE FAT TO THE DISK
0945   C6F1 CE 06 00      LC6F1   LDU   #DBUF0                      =POINT U TO DISK BUFFER 0 AND
0946   C6F4 DF EE                 STU   DCBPT                       =SAVE IT AS DSKCON VARIABLE
0947   C6F6 CC 11 03              LDD   #$1103                      * WRITE DIRECTORY TRACK - SAVE
0948   C6F9 97 EC                 STA   DCTRK                       * TRACK AND WRITE OPERATION CODE IN
0949   C6FB D7 EA                 STB   DCOPC                       * DSKCON VARIABLES
0950   C6FD C6 02                 LDB   #$02                        = GET FILE ALLOCATION SECTOR AND
0951   C6FF D7 ED                 STB   DSEC                        = SAVE IN DSKCON VARIABLE
0952   C701 8D 22                 BSR   LC725                       POINT X TO PROPER FILE ALLOCATION TABLE
0953   C703 6F 01                 CLR   FAT1,X                      RESET FLAG INDICATING VALID FAT DATA HAS BEEN STORED ON DISK
0954   C705 30 06                 LEAX  FATCON,X                    MOVE (X) TO START OF GRANULE DATA
0955   C707 C6 44                 LDB   #GRANMX                     68 BYTES IN FAT
0956   C709 BD A5 9A              JSR   LA59A                       MOVE ACCB BYTES FROM FAT RAM IMAGE TO DBUF0
0957
0958                      * ZERO OUT ALL OF THE BYTES IN THE FAT SECTOR WHICH DO NOT CONTAIN THE GRANULE DATA
0959                      * ZERO OUT THE REMAINDER OF THE SECTOR BUFFER
0960   C70C 6F 80         LC70C   CLR   ,X+                         THIS IS A BUG; SHOULD BE CLR ,U+
```

```
0961   C70E 8C 07 00              CMPX #DBUF0+SECLEN                  MORE OF THE SAME BUG; SHOULD BE CMPU
0962                        * BNE LC70C  THIS INSTRUCTION HAS BEEN LEFT OUT
0963   C711 7E D5 FF              JMP   LD5FF                         WRITE A SECTOR
0964
0965                        * ENTER WITH ACCB CONTAINING FILE NUMBER (1-15); EXIT WITH X POINTING
0966                        * TO CORRECT FILE BUFFER; FLAGS SET ACCORDING TO FILE TYPE.
0967
0968   C714 34 04           LC714 PSHS  B                             SAVE FILE NUMBER ON STACK
0969   C716 D6 6F                 LDB   DEVNUM                        GET DEVICE NUMBER (FILE NUMBER)
0970   C718 8C 34 04        LC718 CMPX  #$3404                        SKIP TWO BYTES
0971   C719 34 04           LC719 PSHS  B                             SAVE FILE NUMBER ON STACK
0972   C71B 58                    ASLB                                X2: 2 BYTES PER POINTER
0973   C71C 8E 09 26              LDX   #FCBV1-2                      POINT X TO START OF FCB POINTERS
0974   C71F AE 85                 LDX   B,X                           POINT X TO PROPER FCB
0975   C721 E6 00                 LDB   FCBTYP,X                      SET FLAGS ACCORDING TO FILE TYPE
0976   C723 35 84                 PULS  B,PC                          RESTORE FILE NUMBER
0977
0978                        * POINT X TO DRIVE ALLOCATION TABLE
0979
0980   C725 34 06           LC725 PSHS  B,A                           SAVE ACCD ON STACK
0981   C727 96 EB                 LDA   DCDRV                         GET DRIVE NUMBER
0982   C729 C6 4A                 LDB   #FATLEN                       GET LENGTH OF FILE ALLOCATION TABLE
0983   C72B 3D                    MUL                                 MULTIPLY BY DRIVE NUMBER TO GET OFFSET
0984   C72C 8E 08 00              LDX   #FATBL0                       START OF FILE ALLOCATION TABLE
0985   C72F 30 8B                 LEAX  D,X                           POINT TO RIGHT TABLE
0986   C731 35 86                 PULS  A,B,PC                        RESTORE ACCD
0987
0988                        * CONVERT GRANULE NUMBER TO TRACK & SECTOR NUMBER - X MUST BE POINTING TO CORRECT
0989                        * FCB; THE TRACK AND SECTOR NUMBER WILL BE STORED IN DSKCON REGISTERS
0990   C733 E6 03           LC733 LDB   FCBCGR,X                      GET GRANULE NUMBER
0991   C735 54                    LSRB                                DIVIDE BY 2 - 2 GRANULES / TRACK
0992   C736 D7 EC                 STB   DCTRK                         TRACK NUMBER
0993   C738 C1 11                 CMPB  #17                           TRACK 17 = DIRECTORY TRACK
0994   C73A 25 02                 BLO   LC73E                         BRANCH IF < DIRECTORY TRACK
0995   C73C 0C EC                 INC   DCTRK                         INCR TRACK NUMBER IF > DIRECTORY TRACK
0996   C73E 58              LC73E ASLB                                MULTIPLY TRACK NUMBER BY 2
0997   C73F 50                    NEGB                                NEGATE GRANULE NUMBER
0998   C740 EB 03                 ADDB  FCBCGR,X                      B=0 IF EVEN GRANULE; 1 IF ODD
0999   C742 8D 05                 BSR   LC749                         RETURN B=0 FOR EVEN GRANULE NUMBER, B=9 FOR ODD GRANULE NUMBER
1000   C744 EB 04                 ADDB  FCBSEC,X                      ADD SECTOR NUMBER
1001   C746 D7 ED                 STB   DSEC                          SAVE SECTOR NUMBER
1002   C748 39                    RTS
1003                        * MULTIPLY ACCD BY 9
1004   C749 34 06           LC749 PSHS  B,A                           TEMP STORE ACCD ON STACK
1005   C74B 58                    ASLB                                *
1006   C74C 49                    ROLA                                * MULTIPLY BY 2
1007   C74D 58                    ASLB                                =
1008   C74E 49                    ROLA                                = MULTIPLY BY FOUR
1009   C74F 58                    ASLB                                *
1010   C750 49                    ROLA                                * MULTIPLY BY EIGHT
1011   C751 E3 E1                 ADDD  ,S++                          ADD ONE = MULTIPLY BY NINE
1012   C753 39                    RTS
1013
1014                        * CONVERT ACCD INTO A GRANULE NUMBER - RETURN RESULT IN ACCB;
1015                        * ENTER WITH ACCD CONTAINING A NUMBER OF SECTORS. RETURN IN ACCB
1016                        * THE NUMBER (0-67) CORRESPONDING TO THE NUMBER OF COMPLETE
1017                        * GRANULES CONTAINED IN THAT MANY SECTORS.
1018                        * DIVIDE BY 90, MULTIPLY BY 10 IS FASTER THAN DIVIDE BY 9
1019   C754 6F E2           LC754 CLR   ,-S                           CLEAR A TEMPORARY SLOT ON THE STACK
1020   C756 6C E4           LC756 INC   ,S                            * DIVIDE ACCD BY 90 - SAVE THE
1021   C758 83 00 5A              SUBD  #9*10                         * QUOTIENT+1 ON THE STACK - REMAINDER
1022   C75B 2A F9                 BPL   LC756                         * IN ACCB
1023   C75D A6 E4                 LDA   ,S                            = PUT THE QUOTIENT+1 IN ACCA AND
1024   C75F E7 E4                 STB   ,S                            = SAVE REMAINDER ON STACK
1025   C761 C6 0A                 LDB   #10                           * MULTIPLY (QUOTIENT+1)
1026   C763 3D                    MUL                                 * BY 10
1027   C764 35 02                 PULS  A                             PUT THE REMAINDER IN ACCA
1028   C766 5A              LC766 DECB                                * DECREMENT THE GRANULE COUNT BY ONE FOR
1029   C767 8B 09                 ADDA  #$09                          * EVERY NINE SECTORS (1 GRANULE) IN THE
1030   C769 2B FB                 BMI   LC766                         * REMAINDER - COMPENSATE FOR THE + 1 IN QUOTIENT+1
1031   C76B 4F                    CLRA                                CLEAR MS BYTE OF ACCD
1032   C76C 39              LC76C RTS
1033
1034                        * MAKE SURE RAM FILE ALLOCATION TABLE DATA IS VALID
1035   C76D 8D B6           LC76D BSR   LC725                         POINT X TO FAT FOR THE CORRECT DRIVE NUMBER
1036   C76F 6D 00                 TST   FAT0,X                        CHECK TO SEE IF ANY FILES ARE ACTIVE
1037   C771 26 F9                 BNE   LC76C                         RETURN IF ANY FILES ACTIVE IN THIS FAT
1038   C773 6F 01                 CLR   FAT1,X                        RESET FAT DATA VALID FLAG
1039   C775 33 06                 LEAU  FATCON,X                      LOAD U WITH START OF GRANULE DATA BUFFER
1040   C777 8E 06 00              LDX   #DBUF0                        BUFFER FOR DISK TRANSFER
1041   C77A 9F EE                 STX   DCBPT                         PUT IN DSKCON PARAMETER
1042   C77C CC 11 02              LDD   #$1102                        DIRECTORY TRACK, READ SECTOR
1043   C77F 97 EC                 STA   DCTRK                         STORE IN DSKCON TRACK NUMBER
1044   C781 D7 EA                 STB   DCOPC                         STORE IN DSKCON OP CODE
1045   C783 C6 02                 LDB   #$02                          GET SECTOR NUMBER 2 (FILE ALLOCATION TABLE)
1046   C785 D7 ED                 STB   DSEC                          STORE IN DSKCON PARAMETER
1047   C787 BD D5 FF              JSR   LD5FF                         GO READ SECTOR
1048   C78A C6 44                 LDB   #GRANMX                       TRANSFER FILE ALLOCATION TABLE TO FILE ALLOC TABLE BUFFER
1049   C78C 7E A5 9A              JMP   LA59A                         MOVE B BYTES FROM (X) TO (U)
1050
1051                        * FIND FIRST FREE GRANULE - ENTER WITH ACCB CONTAINING
1052                        * GRANULE FROM WHICH TO START SEARCHING. THE FOUND GRANULE
1053                        * IS MARKED BY STORING A $C0 IN THE GRANULE'S DATA BYTE
1054                        * TO INDICATE THAT IT IS THE LAST GRANULE IN THE FILE.
1055                        * RETURN WITH FIRST FREE GRANULE FOUND IN ACCA
1056   C78F 8D 94           LC78F BSR   LC725                         POINT X TO FILE ALLOC TABLE
```

```
1057   C791 30 06              LEAX  FATCON,X                  SKIP CONTROL BYTES
1058   C793 4F                 CLRA                            USE ACCA AS GRANULE COUNTER
1059   C794 C4 FE              ANDB  #$FE                      MASK OFF BIT ZERO OF SEARCH GRANULE
1060   C796 6F E2              CLR   ,-S                       INITIALIZE AND SAVE A BYTE ON STACK (DIRECTION FLAG)
1061   C798 63 85        LC798 COM   B,X                       IS THIS GRANULE FREE? ($FF=FREE)
1062   C79A 27 31              BEQ   LC7CD                     BRANCH IF IT IS
1063   C79C 63 85              COM   B,X                       RESTORE GRANULE DATA
1064   C79E 4C                 INCA                            ADD ONE TO GRANULE COUNTER
1065   C79F 81 44              CMPA  #GRANMX                   GRANMX GEANULES PER DISK
1066   C7A1 24 25              BHS   LC7C8                     BRANCH IF ALL GRANULES CHECKED (DISK FULL)
1067   C7A3 5C                 INCB                            INCR TO NEXT GRANULE
1068   C7A4 C5 01              BITB  #$01                      IS BIT 0 SET?
1069   C7A6 26 F0              BNE   LC798                     BRANCH IF ODD GRANULE NUMBER (SAME TRACK)
1070   C7A8 34 06              PSHS  B,A                       SAVE GRANULE COUNTER AND CURRENT GRANULE NUMBER
1071   C7AA C0 02              SUBB  #$02                      SUBTRACT ONE TRACK (2 GRANULES)
1072   C7AC 63 62              COM   $02,S                     COMPLEMENT DIRECTION FLAG
1073   C7AE 26 0C              BNE   LC7BC                     BRANCH EVERY OTHER TIME
1074   C7B0 E0 E0              SUBB  ,S+                       SUBTRACT THE GRANULE COUNTER FROM THE CURRENT GRANULE NUMBER
1075   C7B2 2A 04              BPL   LC7B8                     BRANCH IF LOWER BOUND NOT EXCEEDED
1076   C7B4 E6 E4              LDB   ,S                        RESTORE CURRENT GRANULE NUMBER IF LOWER BOUND EXCEEDED
1077   C7B6 63 61        LC7B6 COM   $01,S                     * COMPLEMENT FLAG - IF GRANULE NUMBER HAS EXCEEDED
1078                     *                                     * BOUNDS ON EITHER THE HI OR LO SIDE, FORCE IT TO GO IN
1079                     *                                     * THE DIRECTION OPPOSITE THE EXCEEDED BOUND
1080   C7B8 32 61        LC7B8 LEAS  $01,S                     CLEAN UP STACK
1081   C7BA 20 DC              BRA   LC798                     CHECK FOR ANOTHER FREE GRANULE
1082
1083   C7BC EB E0        LC7BC ADDB  ,S+                       ADD THE GRANULE COUNTER TO THE CURRENT GRANULE NUMBER
1084   C7BE C1 44              CMPB  #GRANMX                   GRANMX GRANULES PER DISK
1085   C7C0 25 F6              BLO   LC7B8                     BRANCH IF UPPER BOUND NOT EXCEEDED
1086   C7C2 E6 E4              LDB   ,S                        * RESTORE CURRENT GRANULE COUNT AND GO TWICE
1087   C7C4 C0 04              SUBB  #$04                      * AS FAR AS USUAL IN OPPOSITE DIRECTION IF UPPER BOUND EXCEEDED
1088   C7C6 20 EE              BRA   LC7B6                     KEEP SEARCHING
1089   C7C8 C6 38        LC7C8 LDB   #2*28                     'DISK FULL' ERROR
1090   C7CA 7E AC 46           JMP   LAC46                     JUMP TO ERROR HANDLER
1091
1092                     * POINT X TO FIRST FREE GRANULE POSITION IN THE FILE ALLOCATION
1093                     * TABLE AND MARK THE POSITION WITH A LAST GRANULE IN FILE MARKER
1094   C7CD 32 61        LC7CD LEAS  $01,S                     CLEAR UP STACK - REMOVE DIRECTION FLAG
1095   C7CF 1F 98              TFR   B,A                       GRANULE NUMBER TO ACCA
1096   C7D1 3A                 ABX                             POINT X TO FIRST FOUND GRANULE
1097   C7D2 C6 C0              LDB   #$C0                      LAST GRANULE FLAG
1098   C7D4 E7 84              STB   ,X                        MARK THE FIRST FOUND GRANULE AS THE LAST GRANULE
1099   C7D6 39           LC7D6 RTS
1100
1101                     * CHECK ALL ACTIVE FILES TO MAKE SURE A FILE IS NOT ALREADY OPEN - TO BE OPEN
1102                     * A FILE BUFFER MUST MATCH THE DRIVE NUMBER AND FIRST GRANULE NUMBER
1103                     * IN RAM DIRECTORY ENTRY AND THE FCB TYPE MUST NOT MATCH THE FILE TYPE IN ACCA
1104                     * AN 'AO' ERROR WILL NOT BE GENERATED IF A FILE IS BEING OPENED FOR
1105                     * THE SAME MODE THAT IT HAS ALREADY BEEN OPENED UNDER.
1106
1107   C7D7 34 02        LC7D7 PSHS  A                         SAVE FILE TYPE ON STACK
1108   C7D9 F6 09 5B           LDB   FCBACT                    NUMBER OF CURRENTLY OPEN FILES
1109   C7DC 5C                 INCB                            ADD ONE MORE TO FILE COUNTER
1110   C7DD BD C7 19     LC7DD JSR   LC719                     POINT X TO FCB OF THIS FILE
1111   C7E0 27 17              BEQ   LC7F9                     BRANCH IF BUFFER NOT BEING USED
1112   C7E2 96 EB              LDA   DCDRV                     * GET DRIVE NUMBER AND CHECK TO SEE IF IT
1113   C7E4 A1 01              CMPA  FCBDRV,X                  * MATCHES THE DRIVE NUMBER FOR THIS BUFFER
1114   C7E6 26 11              BNE   LC7F9                     FILE EXISTS ON ANOTHER DRIVE
1115   C7E8 FE 09 74           LDU   V974                      GET RAM DIRECTORY AREA
1116   C7EB A6 4D              LDA   DIRGRN,U                  GET FIRST GRANULE IN FILE
1117   C7ED A1 02              CMPA  FCBFGR,X                  DOES IT MATCH THIS FILE BUFFER?
1118   C7EF 26 08              BNE   LC7F9                     NO
1119   C7F1 A6 00              LDA   FCBTYP,X                  GET FILE TYPE OF THIS BUFFER
1120   C7F3 A1 E4              CMPA  ,S                        DOES IT MATCH THE ONE WE ARE LOOKING FOR?
1121   C7F5 10 26 DE 23        LBNE  LA61C                     'FILE ALREADY OPEN' ERROR IF NOT
1122   C7F9 5A           LC7F9 DECB                            DECR FILE COUNTER
1123   C7FA 26 E1              BNE   LC7DD                     BRANCH IF HAVEN'T CHECKED ALL ACTIVE FILES
1124   C7FC 35 82              PULS  A,PC                      RESTORE FILE TYPE AND RETURN
1125
1126   C7FE BD A5 A5     LC7FE JSR   LA5A5                     EVALUATE AN EXPRESSION (DEVICE NUMBER)
1127   C801 0F 6F              CLR   DEVNUM                    SET DEVICE NUMBER TO SCREEN
1128   C803 5D                 TSTB                            TEST NEW DEVICE NUMBER
1129   C804 10 2F EC 42        LBLE  LB44A                     'FC' ERROR IF DEVICE NUMBER NOT A DISK FILE
1130   C808 BD C7 19           JSR   LC719                     POINT X TO FCB
1131   C80B A6 00              LDA   FCBTYP,X                  TEST IF BUFFER IS IN USE
1132   C80D 10 27 DB EA        LBEQ  LA3FB                     'FILE NOT OPEN' ERROR
1133   C811 81 40              CMPA  #RANFIL                   DIRECT/RANDOM FILE?
1134   C813 27 C1              BEQ   LC7D6                     RETURN IF RANDOM
1135   C815 7E A6 16     LC815 JMP   LA616                     BAD FILE MODE ERROR IF NOT RANDOM
1136
1137                     * INPUT DEVICE NUMBER CHECK RAM HOOK
1138   C818 86 10        DVEC5 LDA   #INPFIL                   INPUT FILE TYPE
1139   C81A 8C 86 20     LC81A CMPX  #$8620                    SKIP TWO BYTES
1140
1141                     * PRINT DEVICE NUMBER CHECK RAM HOOK
1142   C81B 86 20        DVEC6 LDA   #OUTFIL                   OUTPUT FILE TYPE
1143   C81D 0D 6F              TST   DEVNUM                    * CHECK DEVICE NUMBER AND RETURN IF
1144   C81F 2F B5              BLE   LC7D6                     * NOT A DISK FILE
1145   C821 AF E4              STX   ,S                        = REPLACE SUBROUTINE RETURN ADDRESS WITH X REGISTER -
1146                     *                                     = THIS IS THE SAME AS LEAS 2,S AND PSHS X
1147   C823 BD C7 14           JSR   LC714                     POINT X TO FCB
1148   C826 34 02              PSHS  A                         SAVE FILE TYPE ON STACK
1149   C828 A6 00              LDA   FCBTYP,X                  GET FILE TYPE
1150   C82A 10 27 DB CD        LBEQ  LA3FB                     'FILE NOT OPEN' ERROR
1151   C82E 81 40              CMPA  #RANFIL                   RANDOM FILE?
1152   C830 27 04              BEQ   LC836                     BRANCH IF RANDOM FILE
```

```
1153   C832 A1 E4                CMPA  ,S                          IS THIS FCB OF THE PROPER TYPE?
1154   C834 26 DF                BNE   LC815                       'FILE MODE' ERROR IF NOT
1155   C836 35 92          LC836 PULS  A,X,PC                      RETURN
1156                       * DEVICE NUMBER VALIDITY CHECK RAM HOOK
1157   C838 2F 9C          DVEC1 BLE   LC7D6                       RETURN IF NOT A DISK FILE
1158   C83A F1 09 5B             CMPB  FCBACT                      COMPARE DEVICE NUMBER TO HIGHEST POSSIBLE
1159   C83D 10 22 DD DE          LBHI  LA61F                       'DEVICE NUMBER' ERROR IF TOO BIG
1160   C841 35 90                PULS  X,PC                        RETURN
1161
1162                       * SET PRINT PARAMETERS RAM HOOK
1163   C843 0D 6F          DVEC2 TST   DEVNUM                      *CHECK DEVICE NUMBER AND
1164   C845 2F 8F                BLE   LC7D6                       *RETURN IF NOT DISK FILE
1165   C847 32 62                LEAS  $02,S                       PURGE RETURN ADDRESS OFF OF THE STACK
1166   C849 34 16                PSHS  X,B,A                       SAVE REGISTERS
1167   C84B 0F 6E                CLR   PRTDEV                      SET PRINT DEVICE NUMBER TO NON-CASSETTE
1168   C84D BD C7 14             JSR   LC714                       POINT X TO FCB
1169   C850 E6 06                LDB   FCBPOS,X                    GET PRINT POSITION
1170   C852 4F                   CLRA                              PRINT WIDTH (256)
1171   C853 8E 10 00             LDX   #$1000                      TAB FIELD WIDTH AND TAB ZONE
1172   C856 7E A3 7C             JMP   LA37C                       SAVE THE PRINT PARAMETERS
1173
1174                       * BREAK CHECK RAM HOOK
1175   C859 0D 6F          DVEC11 TST  DEVNUM                      * CHECK DEVICE NUMBER AND RETURN
1176   C85B 2F 02                BLE   LC85F                       * IF NOT A DISK FILE
1177   C85D 32 62                LEAS  $02,S                       = PURGE RETURN ADDRESS OFF OF THE STACK - DON'T
1178   C85F 39             LC85F RTS                               = DO A BREAK CHECK IF DISK FILE
1179
1180                       * EOF RAM HOOK
1181   C860 32 62          DVEC14 LEAS $02,S                       PURGE RETURN ADDRESS OFF OF THE STACK
1182   C862 96 6F                LDA   DEVNUM                      * GET DEVICE NUMBER AND SAVE
1183   C864 34 02                PSHS  A                           * IT ON THE STACK
1184   C866 BD A5 AE             JSR   LA5AE                       STRIP DEVICE NUMBER OFF OF INPUT LINE
1185   C869 BD A3 ED             JSR   LA3ED                       VERIFY THAT THE FILE TYPE WAS 'INPUT'
1186   C86C 0D 6F                TST   DEVNUM                      * CHECK DEVICE NUMBER AND
1187   C86E 10 2F DD 68          LBLE  LA5DA                       * BRANCH BACK TO  BASIC'S EOF IF NOT DISK FILE
1188   C872 BD C7 14             JSR   LC714                       POINT X TO FCB
1189   C875 E6 00                LDB   FCBTYP,X                    GET FILE TYPE
1190   C877 C1 40                CMPB  #RANFIL                     RANDOM FILE?
1191   C879 27 9A                BEQ   LC815                       'BAD FILE MODE' ERROR IF RANDOM
1192   C87B 5F                   CLRB                              FILE NOT EMPTY FLAG - SET TO NOT EMPTY
1193   C87C A6 88 10             LDA   FCBCFL,X                    *CHECK THE CACHE FLAG - BRANCH IF
1194   C87F 26 03                BNE   LC884                       *THERE IS A CHARACTER WHICH HAS BEEN CACHED
1195   C881 E6 88 17             LDB   FCBDFL,X                    GET SEQUENTIAL INPUT FILE STATUS
1196   C884 7E A5 E4       LC884 JMP   LA5E4                       LINK BACK TO BASIC'S EOF STATEMENT
1197
1198                       * GET FILENAME/EXTENSION: DRIVE NUMBER FROM BASIC
1199   C887 8E C2 91       LC887 LDX   #DEFEXT                     POINT TO '   ' BLANK (DEFAULT) EXTENSION
1200   C88A 6F E2          LC88A CLR   ,-S                         CLEAR A BYTE ON STACK FOR USE AS A DRIVES FLAG
1201   C88C B6 09 5A             LDA   DEFDRV                      * GET DEFAULT DISK NUMBER
1202   C88F 97 EB                STA   DCDRV                       * STORE IN DSKCON PARAMETER
1203   C891 CE 09 4C             LDU   #DNAMBF                     DISK FILENAME BUFFER
1204   C894 CC 20 08             LDD   #$2008                      STORE 8 BLANKS IN RAM (DEFAULT FILE NAME)
1205   C897 A7 C0          LC897 STA   ,U+                         STORE A BLANK IN FILE NAME
1206   C899 5A                   DECB                              DECREMENT COUNTER
1207   C89A 26 FB                BNE   LC897                       BRANCH IF NOT DONE
1208   C89C C6 03                LDB   #$03                        3 BYTES IN EXTENSION
1209   C89E BD A5 9A             JSR   LA59A                       MOVE B BYTES FROM (X) TO (U)
1210   C8A1 BD 87 48             JSR   L8748                       EVALUATE A STRING EXPRESSION
1211   C8A4 33 84                LEAU  ,X                          POINT U TO START OF STRING
1212   C8A6 C1 02                CMPB  #$02                        * CHECK LENGTH OF STRING AND
1213   C8A8 25 12                BLO   LC8BC                       * BRANCH IF < 2
1214   C8AA A6 41                LDA   $01,U                       = GET 2ND CHARACTER IN STRING AND
1215   C8AC 81 3A                CMPA  #':'                        = CHECK FOR COLON
1216   C8AE 26 0C                BNE   LC8BC                       BRANCH IF NO DRIVE NUMBER
1217   C8B0 A6 C4                LDA   ,U                          * GET 1ST CHARACTER
1218   C8B2 81 30                CMPA  #'0'                        * IN STRING AND
1219   C8B4 25 06                BLO   LC8BC                       * CHECK TO SEE
1220   C8B6 81 33                CMPA  #'3'                        * IF IT IS IN
1221   C8B8 22 02                BHI   LC8BC                       * THE RANGE 0-3
1222   C8BA 8D 33                BSR   LC8EF                       GET DRIVE NUMBER
1223   C8BC 8E 09 4C       LC8BC LDX   #DNAMBF                     POINT X TO FILE NAME BUFFER
1224   C8BF 5C                   INCB                              COMPENSATE FOR DECB BELOW
1225   C8C0 5A             LC8C0 DECB                              DECREMENT STRING LENGTH
1226   C8C1 26 0C                BNE   LC8CF                       BRANCH IF MORE CHARACTERS IN STRING
1227   C8C3 32 61                LEAS  $01,S                       CLEAN UP STACK - REMOVE DRIVE FLAG
1228   C8C5 8C 09 4C             CMPX  #DNAMBF                     POINTER STILL AT START OF BUFFER?
1229   C8C8 26 67                BNE   LC931                       RETURN IF NOT
1230   C8CA C6 3E          LC8CA LDB   #2*31                       'BAD FILENAME' ERROR IF NULL FILENAME
1231   C8CC 7E AC 46             JMP   LAC46                       ERROR HANDLER
1232   C8CF A6 C0          LC8CF LDA   ,U+                         GET A CHARACTER FROM STRING
1233   C8D1 81 2E                CMPA  #'.'                        LOOK FOR PERIOD?
1234   C8D3 27 2D                BEQ   LC902                       YES
1235   C8D5 81 2F                CMPA  #'/'                        SLASH?
1236   C8D7 27 29                BEQ   LC902                       YES
1237   C8D9 81 3A                CMPA  #':'                        COLON?
1238   C8DB 27 09                BEQ   LC8E6                       YES
1239   C8DD 8C 09 54             CMPX  #DEXTBF                     COMPARE POINTER TO END OF FILENAME BUFFER
1240   C8E0 27 E8                BEQ   LC8CA                       'BAD FILENAME' ERROR - FILENAME TOO LONG
1241   C8E2 8D 3E                BSR   LC922                       PUT A CHARACTER IN FILENAME
1242   C8E4 20 DA                BRA   LC8C0                       GET ANOTHER CHARACTER FROM STRING
1243   C8E6 8D DD          LC8E6 BSR   LC8C5                       'BAD FILENAME' ERROR IF NO FILENAME YET
1244   C8E8 8D 05                BSR   LC8EF                       GET DRIVE NUMBER
1245   C8EA 5D                   TSTB                              * CHECK LENGTH OF STRING
1246   C8EB 26 DD                BNE   LC8CA                       * 'BAD FILENAME' ERROR IF MORE CHARACTERS LEFT
1247   C8ED 35 82          LC8ED PULS  A,PC                        REMOVE DRIVES FLAG FROM STACK AND RETURN
1248
```

```
1249                            * GRAB DRIVE NUMBER
1250   C8EF 63 62               LC8EF    COM   $Ø2,S                          TOGGLE DRIVE FLAG
1251   C8F1 27 D7                        BEQ   LC8CA                          'BAD FILENAME' ERROR IF DRIVE NUMBER DEFINED TWICE
1252   C8F3 A6 C1                        LDA   ,U++                           ASCII VALUE OF DRIVE NUMBER TO ACCA
1253   C8F5 CØ Ø2                        SUBB  #$Ø2                           DECREMENT STRING LENGTH BY 2 FOR DRIVE (:X)
1254   C8F7 8Ø 3Ø                        SUBA  #'Ø'                           SUBTRACT ASCII BIAS
1255   C8F9 25 CF                        BLO   LC8CA                          DRIVE NUMBER TOO LOW - 'BAD FILENAME' ERROR
1256   C8FB 81 Ø3                        CMPA  #$Ø3                           MAX OF 4 DRIVES
1257   C8FD 22 CB                        BHI   LC8CA                          DRIVE NUMBER TOO HIGH - 'BAD FILENAME' ERROR
1258   C8FF 97 EB                        STA   DCDRV                          STORE IN DSKCON DRIVE NUMBER
1259   C9Ø1 39                           RTS
1260
1261                            * GRAB EXTENSION
1262   C9Ø2 8D C1               LC9Ø2    BSR   LC8C5                          'BAD FILENAME' ERROR IF NO FILENAME YET
1263   C9Ø4 8E Ø9 57                     LDX   #DNAMBF+11                     POINT X TO END OF EXTENSION BUFFER
1264   C9Ø7 86 2Ø                        LDA   #SPACE                         BLANK
1265   C9Ø9 A7 82               LC9Ø9    STA   ,-X                            *
1266   C9ØB 8C Ø9 54                     CMPX  #DEXTBF                        * FILL EXTENSION WITH
1267   C9ØE 26 F9                        BNE   LC9Ø9                          * BLANKS (DEFAULT)
1268   C91Ø 5A                  LC91Ø    DECB                                 DECREMENT STRING COUNTER
1269   C911 27 DA                        BEQ   LC8ED                          RETURN IF ZERO
1270   C913 A6 CØ                        LDA   ,U+                            GET A CHARACTER FROM STRING
1271   C915 81 3A                        CMPA  #':'                           *CHECK FOR DRIVE SEPARATOR
1272   C917 27 CD                        BEQ   LC8E6                          *
1273   C919 8C Ø9 57                     CMPX  #DNAMBF+11                     =CHECK FOR END OF ESTENSION RAM BUFFER &
1274   C91C 27 AC                        BEQ   LC8CA                          ='BAD FILENAME' ERROR IF EXTENSION TOO LONG
1275   C91E 8D Ø2                        BSR   LC922                          PUT A CHARACTER IN EXTENSION BUFFER
1276   C92Ø 2Ø EE                        BRA   LC91Ø                          GET ANOTHER EXTENSION CHARACTER
1277
1278                            * INSERT CHARACTER INTO FILENAME OR EXTENSION
1279   C922 A7 8Ø               LC922    STA   ,X+                            STORE CHARACTER IN FILENAME BUFFER
1280   C924 27 A4                        BEQ   LC8CA                          'BAD FILENAME' ERROR; ZEROES ARE ILLEGAL
1281   C926 81 2E                        CMPA  #'.'                           PERIOD?
1282   C928 27 AØ                        BEQ   LC8CA                          'BAD FILENAME' ERROR IF PERIOD
1283   C92A 81 2F                        CMPA  #'/'                           SLASH?
1284   C92C 27 9C                        BEQ   LC8CA                          'BAD FILENAME' ERROR IF SLASH
1285   C92E 4C                           INCA                                 CHECK FOR $FF
1286   C92F 27 99                        BEQ   LC8CA                          'BAD FILENAME' ERROR IF $FF
1287   C931 39                  LC931    RTS
1288
1289                            * SAVE COMMAND
1290   C932 81 4D               SAVE     CMPA  #'M'                           *
1291   C934 1Ø 27 Ø5 54                  LBEQ  LCE8C                          *BRANCH IF SAVEM
1292   C938 8D 4B                        BSR   LC985                          GO GET FILENAME, ETC. FROM BASIC
1293   C93A 9E 8A                        LDX   ZERO                           ZERO OUT X REG
1294   C93C BF Ø9 57                     STX   DFLTYP                         SET FILE TYPE AND ASCII FLAG TO ZERO
1295   C93F 9D A5                        JSR   GETCCH                         GET CURRENT INPUT CHARACTER FROM BASIC
1296   C941 27 21                        BEQ   LC964                          BRANCH IF END OF LINE
1297   C943 BD B2 6D                     JSR   SYNCOMMA                       SYNTAX CHECK FOR COMMA
1298   C946 C6 41                        LDB   #'A'                           *ASCII FILE?
1299   C948 BD B2 6F                     JSR   LB26F                          *SYNTAX CHECK ON CONTENTS OF ACCB
1300   C94B 26 E4                        BNE   LC931                          RETURN IF NO MORE CHARACTERS ON LINE
1301   C94D 73 Ø9 58                     COM   DASCFL                         SET CRUNCHED/ASCII FLAG TO ASCII
1302   C95Ø 8D Ø4                        BSR   LC956                          OPEN A SEQUENTIAL FILE FOR OUTPUT
1303   C952 4F                           CLRA                                 SET ZERO FLAG - CAUSE ENTIRE FILE TO BE LISTED
1304   C953 7E B7 64                     JMP   LIST                           'LIST' THE FILE TO CONSOLE OUT
1305
1306                            * OPEN A SEQUENTIAL FILE FOR INPUT/OUTPUT - USE THE SYSTEM
1307                            * FCB LOCATED AT THE TOP OF FCBS
1308   C956 86 4F               LC956    LDA   #'O'                           OUTPUT FILE TYPE
1309   C958 8C 86 49            LC958    CMPX  #$8649                         SKIP TWO BYTES
1310   C959 86 49               LC959    LDA   #'I'                           INPUT FILE TYPE
1311   C95B F6 Ø9 5B                     LDB   FCBACT                         GET NUMBER OF RESERVED FILES CURRENTLY RESERVED
1312   C95E 5C                           INCB                                 ADD ONE - USE ONE ABOVE HIGHEST RESERVED FCB
1313   C95F D7 6F                        STB   DEVNUM                         SAVE IT IN DEVICE NUMBER
1314   C961 7E C4 68                     JMP   LC468                          OPEN A FILE & INITIALIZE FCB
1315                            * SAVE A CRUNCHED FILE - A PREAMBLE OF THREE BYTES WILL PRECEED CRUNCHED
1316                            * FILES: BYTE 1 = $FF, 2,3 = LENGTH OF BASIC PROGRAM
1317   C964 8D FØ               LC964    BSR   LC956                          OPEN A SEQUENTIAL FILE FOR OUTPUT
1318   C966 86 FF                        LDA   #$FF                           BASIC FILE FLAG
1319   C968 BD CB 52                     JSR   LCB52                          CONSOLE OUT
1320   C96B DC 1B                        LDD   VARTAB                         LOAD ACCD WITH START OF VARIABLES
1321   C96D 93 19                        SUBD  TXTTAB                         SUBTRACT START OF BASIC
1322   C96F BD CB 52                     JSR   LCB52                          CONSOLE OUT FILE LENGTH MS BYTE
1323   C972 1F 98                        TFR   B,A                            PULL LS BYTE INTO ACCA
1324   C974 BD CB 52                     JSR   LCB52                          CONSOLE OUT FILE LENGTH LS BYTE
1325   C977 9E 19                        LDX   TXTTAB                         POINT X TO START OF BASIC
1326   C979 A6 8Ø               LC979    LDA   ,X+                            GET BYTE FROM BASIC
1327   C97B BD CB 52                     JSR   LCB52                          SEND TO CONSOLE OUT
1328   C97E 9C 1B                        CMPX  VARTAB                         COMPARE TO END OF BASIC
1329   C98Ø 26 F7                        BNE   LC979                          KEEP GOING IF NOT AT END
1330   C982 7E A4 2D                     JMP   LA42D                          CLOSE FILE
1331   C985 8E C2 8E            LC985    LDX   #BASEXT                        POINT TO 'BAS' EXTENSION (DEFAULT)
1332   C988 7E C8 8A                     JMP   LC88A                          GET FILENAME.EXT FROM BASIC
1333
1334                            * MERGE COMMAND
1335   C98B 4F                  MERGE    CLRA                                 RUN FLAG (Ø = DON'T RUN)
1336   C98C C6 FF                        LDB   #$FF                           MERGE FLAG ($FF = MERGE)
1337   C98E 2Ø 12                        BRA   LC9A2                          GO LOAD THE FILE
1338
1339                            * RUN RAM VECTOR
1340   C99Ø 81 22               DVEC18   CMPA  #'"'                           CHECK FOR FILENAME DELIMITER (DOUBLE QUOTE)
1341   C992 1Ø 26 B9 Ø6                  LBNE  XVEC18                         NONE - JUMP TO EXBAS RUN RAM HOOK
1342   C996 86 Ø2                        LDA   #$Ø2                           RUN FLAG - DON'T CLOSE ALL FILES BEFORE RUN
1343   C998 2Ø Ø7                        BRA   LC9A1                          LOAD THE FILE
1344
```

```
1345                            * LOAD COMMAND
1346   C99A 81 4D               LOAD    CMPA  #'M'                        *
1347   C99C 10 27 05 45                 LBEQ  LCEE5                       *BRANCH IF LOADM
1348   C9A0 4F                          CLRA                              RUN FLAG = ZERO (DON'T RUN)
1349   C9A1 5F                  LC9A1   CLRB                              CLEAR MERGE FLAG
1350   C9A2 B7 09 59            LC9A2   STA   DRUNFL                      RUN FLAG (0 = DON'T RUN, 2 = RUN)
1351   C9A5 F7 09 5E                    STB   DMRGFL                      MERGE FLAG (0 = NO MERGE, $FF = MERGE)
1352   C9A8 8D DB                       BSR   LC985                       GO GET FILENAME, ETC. FROM BASIC
1353   C9AA 9D A5                       JSR   GETCCH                      GET CURRENT INPUT CHAR
1354   C9AC 27 10                       BEQ   LC9BE                       BRANCH IF END OF LINE
1355   C9AE BD B2 6D                    JSR   SYNCOMMA                    SYNTAX CHECK FOR COMMA
1356   C9B1 C6 52                       LDB   #'R'                        *
1357   C9B3 BD B2 6F                    JSR   LB26F                       *IS NEXT CHAR 'R'? RUN AFTER LOAD
1358   C9B6 BD A5 C7                    JSR   LA5C7                       SYNTAX ERROR IF ANY MORE CHARS ON LINE
1359   C9B9 86 03                       LDA   #$03                        *SET FLAGS TO RUN AND CLOSE ALL FILES
1360   C9BB B7 09 59                    STA   DRUNFL                      *BEFORE THE FILE IS RUN
1361   C9BE 8D 99               LC9BE   BSR   LC959                       GRAB FCB FOR INPUT FILE
1362   C9C0 B6 09 58                    LDA   DASCFL                      *CHECK ASCII FLAG AND BRANCH
1363   C9C3 27 0B                       BEQ   LC9D0                       *IF CRUNCHED BASIC FILE
1364   C9C5 7D 09 5E                    TST   DMRGFL                      IS THIS A MERGE?
1365   C9C8 26 03                       BNE   LC9CD                       BRANCH IF MERGE
1366   C9CA BD AD 19                    JSR   LAD19                       DO A 'NEW' - ERASE VARIABLES, RESET VARIABLES
1367   C9CD 7E AC 7C            LC9CD   JMP   LAC7C                       GO TO BASIC'S MAIN LOOP, IT WILL LOAD PROGRAM
1368
1369                            * LOAD IN A CRUNCHED BASIC FILE
1370   C9D0 B6 09 57            LC9D0   LDA   DFLTYP                      *CHECK FILE TYPE (MUST BE BASIC:0) & CHECK
1371   C9D3 BA 09 5E                    ORA   DMRGFL                      *MERGE FLAG (MUST BE NO MERGE: 0)
1372   C9D6 10 26 DC 3C                 LBNE  LA616                       'BAD FILE MODE' ERROR IF MERGE OR NON-BASIC
1373   C9DA BD AD 19                    JSR   LAD19                       DO A 'NEW' - RESET POINTERS, ERASE VARIABLES
1374   C9DD 73 09 5D                    COM   DLODFL                      * SET THE LOAD FLAG TO $FF - THIS WILL CAUSE A NEW TO
1375                            *                                         * OCCUR IF AN ERROR OCCURS WHILE THE PROGRAM IS BEING LOADED
1376   C9E0 BD CC E2                    JSR   LCCE2                       GET CHAR FROM BUFFER - SHOULD BE $FF
1377   C9E3 BD CC E2                    JSR   LCCE2                       GET ANOTHER - MS BYTE OF LENGTH
1378   C9E6 34 02                       PSHS  A                           SAVE MS BYTE ON STACK
1379   C9E8 BD CC E2                    JSR   LCCE2                       LS BYTE OF LENGTH OF PROGRAM
1380   C9EB 1F 89                       TFR   A,B                         PUT LS BYTE INTO ACCB
1381   C9ED 35 02                       PULS  A                           NOW ACCD CONTAINS LENGTH OF PROGRAM
1382   C9EF D3 19                       ADDD  TXTTAB                      ADD BEGINNING OF BASIC
1383   C9F1 BD AC 37                    JSR   LAC37                       SEE OF ENOUGH ROOM IN RAM FOR THIS FILE
1384   C9F4 9E 19                       LDX   TXTTAB                      GET START OF BASIC
1385   C9F6 BD C5 97            LC9F6   JSR   LC597                       READ A CHAR FROM CONSOLE IN
1386   C9F9 D6 70                       LDB   CINBFL                      BUFFER EMPTY?
1387   C9FB 26 04                       BNE   LCA01                       BRANCH IF SO
1388   C9FD A7 80                       STA   ,X+                         STORE CHAR
1389   C9FF 20 F5                       BRA   LC9F6                       GET ANOTHER CHARACTER
1390
1391   CA01 7F 09 5D            LCA01   CLR   DLODFL                      CLEAR LOAD FLAG - LOAD WAS ERROR FREE
1392   CA04 9F 1B                       STX   VARTAB                      SAVE NEW START OF VARIABLES
1393                            * MAKE SURE LAST THREE BYTES LOADED WERE ZERO
1394   CA06 C6 03                       LDB   #$03                        CHECK THREE BYTES
1395   CA08 A6 82               LCA08   LDA   ,-X                         CHECK A BYTE
1396   CA0A 26 03                       BNE   LCA0F                       BRANCH IF NON-ZERO
1397   CA0C 5A                          DECB                              DECREMENT COUNTER
1398   CA0D 26 F9                       BNE   LCA08                       KEEP CHECKING IF NOT DONE
1399   CA0F 9E 1B               LCA0F   LDX   VARTAB                      GET START OF VARIABLES
1400   CA11 9F 1B               LCA11   STX   VARTAB                      SAVE START OF VARIABLES
1401   CA13 6F 80                       CLR   ,X+                         CLEAR A BYTE
1402   CA15 5A                          DECB                              DECREMRNT COUNTER
1403   CA16 2A F9                       BPL   LCA11                       KEEP CLEARING BYTES IF NOT DONE
1404   CA18 BD A4 2D            LCA18   JSR   LA42D                       CLOSE SELECTED FILE
1405   CA1B BD AD 21                    JSR   LAD21                       DO PART OF NEW - ERASE VARIABLES, RESET INPUT PTR
1406   CA1E BD 82 9C                    JSR   XVEC18                      INITIALIZE EXBAS GRAPHICS VARIABLES
1407   CA21 BD AC EF                    JSR   LACEF                       RELOCATE ALL THE BASIC NEXT LINE POINTERS
1408   CA24 77 09 59                    ASR   DRUNFL                      CHECK LSB OF RUN FLAG
1409   CA27 25 03                       BLO   LCA2C                       BRANCH IF DON'T CLOSE ALL FILES
1410   CA29 BD A4 26                    JSR   LA426                       CLOSE ALL FILES
1411   CA2C 77 09 59            LCA2C   ASR   DRUNFL                      TEST BIT 1 OF RUN FLAG
1412   CA2F 10 25 E3 6B                 LBCS  LAD9E                       BRANCH TO COMM INTERPRETATION LOOP IF BIT 1 SET
1413   CA33 7E AC 73                    JMP   LAC73                       RETURN TO DIRECT MODE
1414
1415   CA36 0D 6F               DVEC13  TST   DEVNUM                      * CHECK DEVICE NUMBER AND
1416   CA38 2E DE                       BGT   LCA18                       * TRY TO RUN FILE IF IT IS A DISK FILE
1417   CA3A 39                          RTS
1418
1419                            * CLOSE ALL FILE BUFFERS RAM VECTOR
1420   CA3B F6 09 5B            DVEC7   LDB   FCBACT                      GET THE NUMBER OF RESERVED FILE BUFFERS
1421   CA3E 5C                          INCB                              ADD ONE
1422   CA3F 34 04               LCA3F   PSHS  B                           SAVE IT
1423   CA41 D7 6F                       STB   DEVNUM                      STORE IT IN DEVICE NUMBER
1424   CA43 8D 0E                       BSR   LCA53                       CLOSE FILE
1425   CA45 35 04                       PULS  B                           GET BACK NUMBER OF FILE BUFFERS
1426   CA47 5A                          DECB                              DECREMENT FILE BUFFER COUNTER
1427   CA48 26 F5                       BNE   LCA3F                       BRANCH IF ALL FILES NOT CLOSED
1428   CA4A 39                  LCA4A   RTS
1429
1430                            * CLOSE FILE RAM HOOK
1431   CA4B 0D 6F               DVEC8   TST   DEVNUM                      * CHECK DEVICE NUMBER AND RETURN
1432   CA4D 10 2F B8 35                 LBLE  XVEC8                       * IF NOT A DISK FILE
1433   CA51 32 62                       LEAS  $02,S                       PURGE RETURN ADDRESS OFF OF THE STACK
1434   CA53 BD C7 14            LCA53   JSR   LC714                       POINT X TO CORRECT FCB
1435   CA56 0F 6F                       CLR   DEVNUM                      SET DEVICE NUMBER TO SCREEN
1436   CA58 9F F1               LCA58   STX   FCBTMP                      SAVE FILE BUFFER POINTER
1437   CA5A A6 00                       LDA   FCBTYP,X                    GET THE TYPE OF THIS FILE
1438   CA5C 27 EC                       BEQ   LCA4A                       RETURN IF FILE NOT OPEN
1439   CA5E 34 02                       PSHS  A                           SAVE FILE TYPE
1440   CA60 6F 00                       CLR   FCBTYP,X                    CLOSE THE FILE - ZERO OUT THE FILE TYPE
```

```
1441   CA62 E6 Ø1                    LDB   FCBDRV,X                    * GET DRIVE NUMBER AND
1442   CA64 D7 EB                    STB   DCDRV                       * SAVE IT IN DSKCON VARIABLE
1443   CA66 81 2Ø                    CMPA  #OUTFIL                     = CHECK FOR OUTPUT TYPE AND
1444   CA68 26 16                    BNE   LCA8Ø                       = BRANCH IF NOT OUTPUT TYPE FILE
1445
1446                         * CLOSE A SEQUENTIAL OUTPUT FILE
1447   CA6A E6 88 18                 LDB   FCBLFT,X                    GET THE NUMBER OF CHARACTERS IN BUFFER
1448   CA6D 86 8Ø                    LDA   #$8Ø                        * SET THE PRE-SAVED BIT TO INDICATE THAT THE DATA
1449                         *                                         * HAS ALREADY BEEN SAVED ON DISK
1450   CA6F ED 88 13                 STD   FCBLST,X                    SAVE THE NUMBER OF BYTES USED IN THE LAST SECTOR
1451   CA72 6C Ø4                    INC   FCBSEC,X                    INCREMENT THE SECTOR NUMBER
1452   CA74 E6 Ø3                    LDB   FCBCGR,X                    GET THE CURRENT GRANULE NUMBER
1453   CA76 BD C7 25                 JSR   LC725                       POINT X TO FILE ALLOCATION TABLE
1454   CA79 A7 Ø1                    STA   FAT1,X                      SET FAT DATA NOT VALID FLAG (ACCA <> Ø)
1455   CA7B 3A                       ABX                               ADD GRANULE OFFSET TO FAT POINTER
1456   CA7C 6C Ø6                    INC   FATCON,X                    * INCREMENT GRANULE DATA (ADD ONE SECTOR TO LAST
1457                         *                                         * GRANULE) SKIP PAST THE SIX FAT CONTROL BYTES
1458   CA7E 2Ø 71                    BRA   LCAF1                       UPDATE FAT AND DIRECTORY
1459   CA8Ø 81 4Ø            LCA8Ø   CMPA  #RANFIL                     RANDOM FILE?
1460   CA82 26 6D                    BNE   LCAF1                       NO - UPDATE FAT AND DIRECTORY
1461
1462                         * CLOSE A RANDOM FILE
1463   CA84 EC Ø9                    LDD   FCBRLN,X                    GET RECORD LENGTH
1464   CA86 AE ØB                    LDX   FCBBUF,X                    POINT X TO RANDOM FILE BUFFER
1465   CA88 31 8B                    LEAY  D,X                         POINT Y TO END OF RANDOM FILE BUFFER
1466   CA8A 34 36                    PSHS  Y,X,B,A                     SAVE POINTERS ON STACK
1467   CA8C DE 1B                    LDU   VARTAB                      GET START OF VARIABLES
1468   CA8E 11 93 1D         LCA8E   CMPU  ARYTAB                      COMPARE TO START OF ARRAYS
1469   CA91 27 ØE                    BEQ   LCAA1                       BRANCH IF ALL VARIABLES CHECKED
1470   CA93 A6 41                    LDA   $Ø1,U                       GET 2ND BYTE OF VARIABLE NAME
1471   CA95 33 42                    LEAU  $Ø2,U                       MOVE POINTER TO START OF DESCRIPTOR
1472   CA97 2A Ø2                    BPL   LCA9B                       BRANCH IF VARIABLE - NUMERIC
1473   CA99 8D 28                    BSR   LCAC3                       ADJUST STRING VARIABLE IF IN RANDOM FILE BUFFER
1474   CA9B 33 45            LCA9B   LEAU  $Ø5,U                       MOVE POINTER TO NEXT VARIABLE
1475   CA9D 2Ø EF                    BRA   LCA8E                       PROCESS ANOTHER VARIABLE
1476   CA9F 35 4Ø            LCA9F   PULS  U                           GET ADDRESS OF NEXT ARRAY TO U
1477   CAA1 11 93 1F         LCAA1   CMPU  ARYEND                      COMPARE TO END OF ARRAYS
1478   CAA4 27 3A                    BEQ   LCAEØ                       BRANCH IF END OF ARRAYS
1479   CAA6 1F 3Ø                    TFR   U,D                         * SAVE ARRAY START IN ACCD, ADD OFFSET
1480   CAA8 E3 42                    ADDD  $Ø2,U                       * TO NEXT ARRAY AND SAVE ADDRESS OF
1481   CAAA 34 Ø6                    PSHS  B,A                         * NEXT ARRAY ON THE STACK
1482   CAAC A6 41                    LDA   $Ø1,U                       GET 2ND LETTER OF VARIABLE NAME
1483   CAAE 2A EF                    BPL   LCA9F                       BRANCH IF NUMERIC
1484   CABØ E6 44                    LDB   $Ø4,U                       GET THE NUMBER OF DIMENSIONS
1485   CAB2 58                       ASLB                              X2:2 BYTES PER DIMENSION
1486   CAB3 CB Ø5                    ADDB  #$Ø5                        5 BYTES CONSTANT PER ARRAY DESCRIPTOR
1487   CAB5 4F                       CLRA                              CLEAR MSB OF OFFSET - (ONLY 125 DIMENSIONS ALLOWED)
1488   CAB6 33 CB                    LEAU  D,U                         POINT U TO START OF THIS ARRAY'S VARIABLES
1489   CAB8 11 A3 E4         LCAB8   CMPU  ,S                          AT END OF THIS ARRAY?
1490   CABB 27 E2                    BEQ   LCA9F                       YES
1491   CABD 8D Ø4                    BSR   LCAC3                       ADJUST STRING VARIABLE IF IN RANDOM FILE BUFFER
1492   CABF 33 45                    LEAU  $Ø5,U                       MOVE POINTER TO NEXT DESCRIPTOR
1493   CAC1 2Ø F5                    BRA   LCAB8                       CHECK NEXT VARIABLE
1494                         *
1495                         * CHECK TO SEE IF A STRING IS LOCATED IN THE RANDOM FILE BUFFER AREA. IF IT IS
1496                         * THE RANDOM FILE BUFFER IN QUESTION, IT WILL BE DELETED. IF IT IS HIGHER IN THE RANDOM
1497                         * FILE BUFFER SPACE THAN THE BUFFER IN QUESTION, THE LENGTH OF THE CURRENT
1498                         * BUFFER WILL BE SUBTRACTED FROM THE ADDRESS OF THE STRING BECAUSE THE CURRENT
1499                         * BUFFER IS BEING DELETED (CLOSED).
1500   CAC3 AE 42            LCAC3   LDX   $Ø2,U                       POINT X TO START OF STRING
1501   CAC5 BC Ø9 48                 CMPX  RNBFAD                      COMPARE TO START OF FREE RANDOM FILE BUFFER AREA
1502   CAC8 24 ØE                    BHS   LCAD8                       RETURN IF > START OF FREE RANDOM FILE BUFFER AREA
1503   CACA AC 64                    CMPX  $Ø4,S                       COMPARE TO START OF THIS FILE'S RANDOM BUFFER
1504   CACC 25 ØB                    BLO   LCAD9                       BRANCH IF < START OF THIS FILE'S RANDOM BUFFER
1505                         *                                         ** BUG ** THIS SHOULD BE A BRANCH TO A RETURN
1506   CACE AC 66                    CMPX  $Ø6,S                       COMPARE TO END OF THIS FILE'S RANDOM BUFFER
1507   CAD0 25 Ø7                    BLO   LCAD9                       BRANCH IF < END OF THIS FILE'S RANDOM BUFFER
1508   CAD2 1F 1Ø                    TFR   X,D                         SAVE POINTER IN ACCD
1509   CAD4 A3 62                    SUBD  $Ø2,S                       SUBTRACT RECORD LENGTH FROM START OF STRING ADDRESS
1510   CAD6 ED 42                    STD   $Ø2,U                       SAVE NEW START OF STRING ADDRESS
1511   CAD8 39               LCAD8   RTS
1512   CAD9 6F C4            LCAD9   CLR   ,U                          CLEAR THE LENGTH OF THE STRING
1513   CADB 6F 42                    CLR   $Ø2,U                       * CLEAR THE ADDRESS
1514   CADD 6F 43                    CLR   $Ø3,U                       * OF THE STRING
1515   CADF 39                       RTS
1516                         *REMOVE RESERVED SPACE IN RANDOM FILE BUFFER FOR A 'CLOSED' RANDOM FILE
1517   CAEØ 35 56            LCAEØ   PULS  A,B,X,U                     *U = END OF RANDOM FILE BUFFER, X = START OF RANDOM
1518                         *                                         *FILE BUFFER, ACCD = RECORD LENGTH
1519
1520                         ** THIS WOULD PROBABLY BE THE MOST CONVENIENT PLACE TO FIX THE BUG WHICH
1521                         ** CAUSES THE SYSTEM TO HANG IF AN ERROR IS ENCOUNTERED DURING 'COPY'
1522
1523                         *       CMPU  FCBADR                      * IS THE END OF THIS FCB'S BUFFER ABOVE THE END
1524                         *                                         * OF THE START OF THE FCB AREA
1525                         *       BLO   LCAE2                       NO - FREE UP THE SPACE USED BY THIS FILE IN RANDOM BUFFER
1526                         *       LDX   #DFLBUF                     YES - DOING A 'COPY'; RESET START OF RANDOM BUFFER
1527                         *       BRA   LCAEE
1528                         * RANDOM FILE BUFFER AREA
1529
1530                         * REMOVE RESERVED SPACE FOR CLOSED FILE FROM RANDOM FILE BUFFER SPACE
1531   CAE2 11 B3 Ø9 48      LCAE2   CMPU  RNBFAD                      AT THE BOTTOM OF FREE RANDOM BUFFER AREA?
1532   CAE6 27 Ø6                    BEQ   LCAEE                       BRANCH IF THERE
1533   CAE8 A6 CØ                    LDA   ,U+                         = GRAB A SOURCE BYTE AND
1534   CAEA A7 8Ø                    STA   ,X+                         = MOVE IT TO DESTINATION
1535   CAEC 2Ø F4                    BRA   LCAE2                       KEEP MOVING BYTES
1536   CAEE BF Ø9 48         LCAEE   STX   RNBFAD                      SAVE NEW START OF FREE RANDOM BUFFER AREA
```

```
1537   CAF1 BD C7 25          LCAF1   JSR   LC725                          POINT X TO PROPER FILE ALLOCATION TABLE
1538   CAF4 6A 00                     DEC   FAT0,X                         REMOVE ONE ACTIVE FILE
1539   CAF6 6D 01                     TST   FAT1,X                         NEW DATA IN FAT RAM IMAGE?
1540   CAF8 27 03                     BEQ   LCAFD                          NO
1541   CAFA BD C6 F1                  JSR   LC6F1                          WRITE OUT FILE ALLOCATION TABLE TO DISK
1542   CAFD 9E F1             LCAFD   LDX   FCBTMP                        GET FILE BUFFER POINTER
1543   CAFF 35 02                     PULS  A                              GET FILE TYPE
1544   CB01 81 20                     CMPA  #OUTFIL                        IS IT A SEQUENTIAL OUTPUT FILE?
1545   CB03 27 08                     BEQ   LCB0D                          YES
1546   CB05 81 40                     CMPA  #RANFIL                        IS IT A RANDOM FILE?
1547   CB07 26 CF                     BNE   LCAD8                          RETURN IF NOT A RANDOM FILE (SEQUENTIAL INPUT)
1548   CB09 A6 0F                     LDA   FCBFLG,X                       * TEST THE GET/PUT FLAG AND
1549   CB0B 27 0A                     BEQ   LCB17                          * BRANCH IF 'GET'
1550
1551                          * WRITE CONTENTS OF FILE BUFFER TO DISK
1552   CB0D BD C7 33          LCB0D   JSR   LC733                          GET PROPER TRACK & SECTOR NUMBERS
1553   CB10 33 88 19                  LEAU  FCBCON,X                       POINT U TO START OF FCB DATA
1554   CB13 DF EE                     STU   DCBPT                          SET UP FILE BUFFER POINTER FOR DSKCON
1555   CB15 8D 2C                     BSR   LCB43                          GO WRITE A SECTOR
1556   CB17 A6 88 13          LCB17   LDA   FCBLST,X                       CHECK THE PRE-SAVED FLAG
1557   CB1A 2A BC                     BPL   LCAD8                          RETURN IF RECORD HAS ALREADY BEEN SAVED ON DISK
1558   CB1C E6 88 12                  LDB   FCBDIR,X                       GET DIRECTORY NUMBER OF THIS FILE
1559   CB1F C4 07                     ANDB  #$07                           8 ENTRIES PER SECTOR
1560   CB21 86 20                     LDA   #DIRLEN                        DIRLEN BYTES PER DIRECTORY ENTRY
1561   CB23 3D                        MUL                                  GET SECTOR OFFSET FOR THIS ENTRY
1562   CB24 CE 06 00                  LDU   #DBUF0                         * GET READ/WRITE BUFFER 0 AND
1563   CB27 DF EE                     STU   DCBPT                          * SAVE IT IN DSKCON REGISTER
1564   CB29 31 CB                     LEAY  D,U                            Y POINTS TO CORRECT DIRECTORY ENTRY
1565   CB2B E6 88 12                  LDB   FCBDIR,X                       GET DIRECTORY ENTRY NUMBER
1566   CB2E 54                        LSRB                                 *
1567   CB2F 54                        LSRB                                 *
1568   CB30 54                        LSRB                                 * DIVIDE BY 8; EIGHT DIRECTORY ENTRIES PER SECTOR
1569   CB31 CB 03                     ADDB  #$03                           ADD BIAS; FIRST 3 SECTORS NOT DIRECTORY
1570   CB33 D7 ED                     STB   DSEC                           STORE SECTOR NUMBER
1571   CB35 CC 11 02                  LDD   #$1102                         DIRECTORY TRACK - READ OP CODE
1572   CB38 97 EC                     STA   DCTRK                          STORE TRACK NUMBER
1573   CB3A 8D 09                     BSR   LCB45                          GO READ DIRECTORY
1574   CB3C EC 88 13                  LDD   FCBLST,X                       GET NUMBER OF BYTES IN THE LAST SECTOR
1575   CB3F 84 7F                     ANDA  #$7F                           MASK OFF THE PRE-SAVED FLAG
1576   CB41 ED 2E                     STD   DIRLST,Y                       SAVE NUMBER OF BYTES IN LAST SECTOR OF FILE IN DIRECTORY
1577   CB43 C6 03             LCB43   LDB   #$03                           WRITE OP CODE
1578   CB45 D7 EA             LCB45   STB   DCOPC                          SAVE DSKCON OP CODE VARIABLE
1579   CB47 7E D5 FF                  JMP   LD5FF                          GO READ/WRITE SECTOR
1580
1581                          * CONSOLE OUT RAM HOOK
1582   CB4A 0D 6F             DVEC3   TST   DEVNUM                         CHECK DEVICE NUMBER
1583   CB4C 10 2F B7 23               LBLE  XVEC3                          BRANCH TO EX BASIC IF NOT A DISK FILE
1584   CB50 32 62                     LEAS  $02,S                          POP RETURN OFF STACK
1585                          * SEND A CHARACTER IN ACCA TO A DISK FILE. A CARRIAGE RETURN WILL RESET THE
1586                          * PRINT POSITION AND CONTROL CODES WILL NOT INCREMENT THE PRINT POSITION.
1587   CB52 34 16             LCB52   PSHS  X,B,A                          SAVE REGISTERS
1588   CB54 8E 09 26                  LDX   #FCBV1-2                       POINT X TO TABLE OF FILE NUMBER VECTORS
1589   CB57 D6 6F                     LDB   DEVNUM                         GET CURRENT FILE NUMBER
1590   CB59 58                        ASLB                                 2 BYTES PER FCB ADDRESS
1591   CB5A AE 85                     LDX   B,X                            POINT X TO PROPER FCB
1592   CB5C E6 84                     LDB   FCBTYP,X                       GET FILE TYPE
1593   CB5E C1 10                     CMPB  #INPFIL                        IS IT AN INPUT FILE?
1594   CB60 27 36                     BEQ   LCB98                          RETURN IF SO
1595   CB62 81 0D                     CMPA  #CR                            CARRIAGE RETURN (ENTER)
1596   CB64 26 02                     BNE   LCB68                          NO
1597   CB66 6F 06                     CLR   FCBPOS,X                       CLEAR PRINT POSITION IF CARRIAGE RETURN
1598   CB68 81 20             LCB68   CMPA  #SPACE                         *
1599   CB6A 25 02                     BLO   LCB6E                          *BRANCH IF CONTROL CHAR
1600   CB6C 6C 06                     INC   FCBPOS,X                       INCREMENT PRINT POSITION
1601   CB6E C1 40             LCB6E   CMPB  #RANFIL                        IS IT RANDOM FILE?
1602   CB70 26 1A                     BNE   LCB8C                          BRANCH IF NOT RANDOM
1603                          * PUT A BYTE INTO A RANDOM FILE
1604   CB72 EC 88 17                  LDD   FCBPUT,X                       GET 'PUT' BYTE COUNTER
1605   CB75 C3 00 01                  ADDD  #$0001                         ADD ONE
1606   CB78 10 A3 09                  CMPD  FCBRLN,X                       COMPARE TO RECORD LENGTH
1607   CB7B 10 22 01 72               LBHI  LCCF1                          'FR' ERROR IF 'PUT' BYTE COUNTER > RECORD LENGTH
1608   CB7F ED 88 17                  STD   FCBPUT,X                       SAVE NEW 'PUT' BYTE COUNTER
1609   CB82 AE 0B                     LDX   FCBBUF,X                       POINT TO RANDOM FILE BUFFER POINTER
1610   CB84 30 8B                     LEAX  D,X                            POINT TO ONE PAST END OF CURRENT RECORD DATA
1611   CB86 35 02                     PULS  A                              PULL DATA FROM STACK
1612   CB88 A7 1F                     STA   -1,X                           STORE IN DATA BUFFER
1613   CB8A 35 94                     PULS  B,X,PC                         RESTORE REGISTERS AND RETURN
1614
1615                          * WRITE A BYTE TO SEQUENTIAL OUTPUT FILE
1616   CB8C 6C 88 18          LCB8C   INC   FCBLFT,X                       INCREMENT CHARACTER COUNT
1617   CB8F E6 88 18                  LDB   FCBLFT,X                       * GET CHARACTER COUNT AND BRANCH
1618   CB92 27 06                     BEQ   LCB9A                          * IF THE BUFFER IS FULL
1619   CB94 3A                        ABX                                  ADD CHARACTER COUNT TO FCB ADDRESS
1620   CB95 A7 88 18                  STA   FCBCON-1,X                     STORE NEW CHARACTER (SKIP PAST 25 CONTROL BYTES AT FCB START)
1621   CB98 35 96             LCB98   PULS  A,B,X,PC
1622
1623                          * WRITE OUT A FULL BUFFER AND RESET BUFFER
1624   CB9A 34 60             LCB9A   PSHS  U,Y                            SAVE REGISTERS
1625   CB9C A7 89 01 18               STA   SECLEN+FCBCON-1,X              STORE LAST CHARACTER IN BUFFER
1626   CBA0 BD C6 58                  JSR   LC658                          INCREMENT RECORD NUMBER
1627   CBA3 E6 01                     LDB   FCBDRV,X                       * GET DRIVE NUMBER AND SAVE
1628   CBA5 D7 EB                     STB   DCDRV                          * IT IN DSKCON CONTROL TABLE
1629   CBA7 6C 04                     INC   FCBSEC,X                       INCREMENT SECTOR NUMBER
1630   CBA9 BD CB 0D                  JSR   LCB0D                          WRITE THE FILE BUFFER TO DISK
1631   CBAC 31 84                     LEAY  ,X                             SAVE FCB POINTER IN Y
1632   CBAE E6 03                     LDB   FCBCGR,X                       GET GRANULE NUMBER
```

```
1633   CBB0 BD C7 25                JSR   LC725                          POINT X TO PROPER ALLOCATION TABLE
1634   CBB3 3A                      ABX                                  ADD THE GRANULE NUMBER TO FAT POINTER
1635   CBB4 33 06                   LEAU  FATCON,X                       *POINT U TO THE CORRECT GRANULE IN FAT - SKIP PAST
1636                        *                                           *THE SIX FAT CONTROL BYTES
1637   CBB6 A6 24                   LDA   FCBSEC,Y                       GET CURRENT SECTOR FOR THIS GRANULE
1638   CBB8 81 09                   CMPA  #$09                           MAX SECTOR NUMBER (9 SECTORS/GRANULE)
1639   CBBA 25 08                   BLO   LCBC4                          BRANCH IF NOT AT END OF GRANULE
1640   CBBC 6F 24                   CLR   FCBSEC,Y                       CLEAR SECTOR NUMBER
1641   CBBE BD C7 8F                JSR   LC78F                          GET NEXT FREE GRANULE
1642   CBC1 A7 23                   STA   FCBCGR,Y                       SAVE NEW GRANULE IN FCB
1643   CBC3 8C 8A C0         LCBC3  CMPX  #$8AC0                         SKIP TWO BYTES   NO DATA STORED IN NEW SECTOR YET
1644   CBC4 8A C0            LCBC4  ORA   #$C0                           FORCE GRANULE NUMBER TO BE FINAL GRANULE IN FILE
1645   CBC6 A7 C4                   STA   ,U                             STORE IN MAP
1646   CBC8 BD C5 7C                JSR   LC57C                          UPDATE FILE ALLOCATION TABLE
1647   CBCB 35 60                   PULS  Y,U                            RESTORE REGISTERS
1648   CBCD 35 96                   PULS  A,B,X,PC                       RESTORE REGISTERS AND RETURN
1649
1650                        * DIR COMMAND
1651   CBCF BD D1 62         DIR    JSR   LD162                          SCAN DRIVE NUMBER FROM INPUT LINE
1652   CBD2 BD C7 6D                JSR   LC76D                          GET FAT FOR THIS DRIVE
1653   CBD5 BD B9 58                JSR   LB958                          PRINT CARRIAGE RETURN TO CONSOLE OUT
1654   CBD8 CC 11 02                LDD   #$1102                         * GET TRACK 17 AND
1655   CBDB 97 EC                   STA   DCTRK                          * READ OP CODE AND
1656   CBDD D7 EA                   STB   DCOPC                          * SAVE IN DSKCON VARIABLES
1657   CBDF C6 03                   LDB   #$03                           START WITH SECTOR 3 (FIRST DIRECTORY SECTOR)
1658
1659                        * READ A DIRECTORY SECTOR INTO THE I/O BUFFER
1660   CBE1 D7 ED            LCBE1  STB   DSEC                           SAVE SECTOR NUMBER IN DSKCON VARIABLE
1661   CBE3 8E 06 00                LDX   #DBUF0                         * USE I/O BUFFER 0 FOR DATA TRANSFER
1662   CBE6 9F EE                   STX   DCBPT                          * SAVE IN DSKCON VARIABLE
1663   CBE8 BD D5 FF                JSR   LD5FF                          READ A SECTOR
1664
1665                        * SEND DIRECTORY INFORMATION TO CONSOLE OUT
1666   CBEB 35 40            LCBEB  PULS  U                              SAVE TOP OF STACK
1667   CBED BD A5 49                JSR   LA549                          GO DO A BREAK CHECK
1668   CBF0 34 40                   PSHS  U                              RESTORE STACK
1669   CBF2 A6 84                   LDA   DIRNAM,X                       TEST FILE NAME FIRST BYTE
1670   CBF4 27 38                   BEQ   LCC2E                          BRANCH IF KILLED
1671   CBF6 43                      COMA                                 FF = END OF DIRECTORY
1672   CBF7 27 44                   BEQ   LCC3D                          RETURN IF END OF DIRECTORY
1673   CBF9 34 10                   PSHS  X                              SAVE DIRECTORY POINTER ON STACK
1674   CBFB C6 08                   LDB   #$08                           NUMBER CHARACTERS TO PRINT
1675   CBFD BD B9 A2                JSR   LB9A2                          SEND FILENAME TO CONSOLE OUT
1676   CC00 8D 3F                   BSR   LCC41                          SEND BLANK TO CONSOLE OUT
1677   CC02 C6 03                   LDB   #$03                           NUMBER CHARACTERS TO PRINT
1678   CC04 BD B9 A2                JSR   LB9A2                          SEND EXTENSION TO CONSOLE OUT
1679   CC07 8D 38                   BSR   LCC41                          SEND BLANK TO CONSOLE OUT
1680   CC09 E6 00                   LDB   FCBTYP,X                       GET FILE TYPE
1681   CC0B C1 0A                   CMPB  #10                            * CHECK THE NUMBER OF DECIMAL DIGITS IN
1682   CC0D 24 02                   BHS   LCC11                          * ACCB: IF THERE IS ONLY ONE DIGIT,
1683   CC0F 8D 30                   BSR   LCC41                          * SEND BLANK TO CONSOLE OUT
1684   CC11 4F               LCC11  CLRA                                 CLEAR MS BYTE OF ACCD
1685   CC12 BD BD CC                JSR   LBDCC                          PRINT ACCD IN DECIMAL TO CONSOLE OUT
1686   CC15 8D 2A                   BSR   LCC41                          SEND BLANK TO CONSOLE OUT
1687   CC17 AE E4                   LDX   ,S                             X NOW POINTS TO DIRECTORY ENTRY
1688   CC19 86 42                   LDA   #'A'+1                         ASCII BIAS
1689   CC1B AB 0C                   ADDA  DIRASC,X                       ADD TO ASCII FLAG
1690   CC1D 8D 1F                   BSR   LCC3E                          PRINT CHARACTER AND BLANK TO CONSOLE OUT
1691   CC1F E6 0D                   LDB   DIRGRN,X                       GET FIRST GRANULE IN FILE
1692   CC21 8D 21                   BSR   LCC44                          COUNT GRANULES
1693   CC23 1F 89                   TFR   A,B                            SAVE COUNT IN ACCB
1694   CC25 4F                      CLRA                                 CLEAR MS BYTE OF ACCD
1695   CC26 BD BD CC                JSR   LBDCC                          PRINT ACCD IN DECIMAL TO CONSOLE OUT
1696   CC29 BD B9 58                JSR   LB958                          SEND CARRIAGE RETURN TO CONSOLE OUT
1697   CC2C 35 10                   PULS  X                              PULL DIRECTORY POINTER OFF OF THE STACK
1698   CC2E 30 88 20         LCC2E  LEAX  DIRLEN,X                       MOVE X TO NEXT DIRECTORY ENTRY
1699   CC31 8C 07 00                CMPX  #DBUF0+SECLEN                  END OF I/O BUFFER?
1700   CC34 25 B5                   BLO   LCBEB                          BRANCH IF MORE DIRECTORY ENTRIES IN BUFFER
1701   CC36 D6 ED                   LDB   DSEC                           GET CURRENT SECTOR
1702   CC38 5C                      INCB                                 BUMP COUNT
1703   CC39 C1 12                   CMPB  #SECMAX                        SECMAX SECTORS IN DIRECTORY TRACK
1704   CC3B 23 A4                   BLS   LCBE1                          GET NEXT SECTOR
1705   CC3D 39               LCC3D  RTS                                  FINISHED
1706   CC3E BD A2 82                JSR   LA282                          SEND CHARACTER TO CONSOLE OUT
1707   CC41 7E B9 AC                JMP   LB9AC                          SEND BLANK TO CONSOLE OUT
1708
1709                        * ENTER WITH ACCB POINTING TO FIRST GRANULE IN A FILE; RETURN THE NUMBER OF
1710                        * GRANULES IN THE FILE IN ACCA, THE GRANULE DATA FOR THE LAST SECTOR IN ACCB
1711   CC44 BD C7 25         LCC44  JSR   LC725                          POINT X TO FILE ALLOCATION BUFFER
1712   CC47 33 06                   LEAU  FATCON,X                       POINT U TO START OF GRANULE DATA
1713   CC49 4F                      CLRA                                 RESET GRANULE COUNTER
1714   CC4A 4C               LCC4A  INCA                                 INCREMENT GRANULE COUNTER
1715   CC4B 81 44                   CMPA  #GRANMX                        CHECKED ALL 68 GRANULES?
1716   CC4D 10 22 F9 D5             LBHI  LC626                          YES - 'BAD FILE STRUCTURE' ERROR
1717   CC51 30 C4                   LEAX  ,U                             POINT U TO START OF GRANULE DATA
1718   CC53 3A                      ABX                                  ADD POINTER TO FIRST GRANULE
1719   CC54 E6 84                   LDB   ,X                             GET THIS GRANULE'S CONTROL BYTE
1720   CC56 C1 C0                   CMPB  #$C0                           IS THIS THE LAST GRANULE IN FILE?
1721   CC58 25 F0                   BLO   LCC4A                          NO - KEEP GOING
1722   CC5A 39                      RTS
1723
1724                        * INPUT RAM HOOK
1725   CC5B 0D 6F            DVEC10 TST   DEVNUM                         * CHECK DEVICE NUMBER AND RETURN
1726   CC5D 2F 5E                   BLE   LCCBD                          * IF NOT A DISK FILE
1727   CC5F 8E B0 69                LDX   #LB069                         = CHANGE THE RETURN ADDRESS ON THE STACK TO RE-ENTER BASIC'S INPUT
1728   CC62 AF E4                   STX   ,S                             = ROUTINE AT A DIFFERENT PLACE THAN THE CALLING ROUTINE
```

```
1729   CC64 8E Ø2 DD                LDX   #LINBUF+1                      POINT X TO THE LINE INPUT BUFFER
1730   CC67 C6 2C                   LDB   #','                           =
1731   CC69 D7 Ø1                   STB   CHARAC                         =COMMA IS READ ITEM SEPARATOR (TEMPORARY STRING SEARCH FLAG)
1732   CC6B 96 Ø6                   LDA   VALTYP                         * GET VARIABLE TYPE AND BRANCH IF
1733   CC6D 26 Ø2                   BNE   LCC71                          * IT IS A STRING
1734   CC6F C6 2Ø                   LDB   #SPACE                         SPACE = NUMERIC SEARCH DELIMITER
1735   CC71 8D 6F            LCC71  BSR   LCCE2                          GET AN INPUT CHARACTER
1736   CC73 81 2Ø                   CMPA  #SPACE                         SPACE?
1737   CC75 27 FA                   BEQ   LCC71                          YES - GET ANOTHER CHARACTER
1738   CC77 81 22                   CMPA  #'"'                           QUOTE?
1739   CC79 26 ØA                   BNE   LCC85                          NO
1740   CC7B C1 2C                   CMPB  #','                           SEARCH CHARACTER = COMMA?
1741   CC7D 26 Ø6                   BNE   LCC85                          NO - NUMERIC SEARCH
1742   CC7F 1F 89                   TFR   A,B                            * SAVE DOUBLE QUOTE AS
1743   CC81 D7 Ø1                   STB   CHARAC                         * THE SEARCH FLAG
1744   CC83 2Ø 22                   BRA   LCCA7                          SAVE DOUBLE QUOTES AS FIRST ITEM IN BUFFER
1745
1746   CC85 C1 22            LCC85  CMPB  #'"'                           *
1747   CC87 27 11                   BEQ   LCC9A                          *BRANCH IF INPUTTING A STRING VARIABLE
1748   CC89 81 ØD                   CMPA  #CR                            IS THE INPUT CHARACTER A CARRIAGE RETURN
1749   CC8B 26 ØD                   BNE   LCC9A                          NO
1750   CC8D 8C Ø2 DD                CMPX  #LINBUF+1                      *IF AT THE START OF INPUTBUFFER, CHECK FOR A
1751   CC9Ø 27 44                   BEQ   LCCD6                          *FOLLOWING LINE FEED AND EXIT ROUTINE
1752   CC92 A6 1F                   LDA   -1,X                           =IF THE INPUT CHARACTER PRECEEDING THE CR WAS A LINE FEED,
1753   CC94 81 ØA                   CMPA  #LF                            =THEN INSERT THE CR IN THE INPUT STRING, OTHERWISE
1754   CC96 26 3E                   BNE   LCCD6                          =CHECK FOR A FOLLOWING LINE FEED AND EXIT THE ROUTINE
1755   CC98 86 ØD                   LDA   #CR                            RESTORE CARRIAGE RETURN AS THE INPUT CHARACTER
1756   CC9A 4D               LCC9A  TSTA                                 *CHECK FOR A NULL (ZERO) INPUT CHARACTER AND
1757   CC9B 27 17                   BEQ   LCCB4                          *IGNORE IT IF 1T IS A NULL
1758   CC9D 91 Ø1                   CMPA  CHARAC                         =
1759   CC9F 27 1D                   BEQ   LCCBE                          =CHECK TO SEE IF THE INPUT CHARACTER MATCHES
1760   CCA1 34 Ø4                   PSHS  B                              =EITHER ACCB OR CHARAC AND IF IT DOES, THEN
1761   CCA3 A1 EØ                   CMPA  ,S+                            =BRANCH TO CHECK FOR ITEM SEPARATOR OR
1762   CCA5 27 17                   BEQ   LCCBE                          =TERMINATOR SEQUENCE AND EXIT ROUTINE
1763   CCA7 A7 8Ø            LCCA7  STA   ,X+                            STORE NEW CHARACTER IN BUFFER
1764   CCA9 8C Ø3 D6                CMPX  #LINBUF+LBUFMX                 END OF INPUT BUFFER
1765   CCAC 26 Ø6                   BNE   LCCB4                          NO
1766   CCAE 8D 46                   BSR   LCCF6                          GET A CHARACTER FROM CONSOLE IN
1767   CCBØ 26 Ø6                   BNE   LCCB8                          EXIT ROUTINE IF BUFFER EMPTY
1768   CCB2 2Ø 1E                   BRA   LCCD2                          CHECK FOR CR OR CR/LF AND EXIT ROUTINE
1769
1770   CCB4 8D 4Ø            LCCB4  BSR   LCCF6                          GET A CHARACTER FROM CONSOLE IN
1771   CCB6 27 CD                   BEQ   LCC85                          BRANCH IF BUFFER NOT EMPTY
1772   CCB8 6F 84            LCCB8  CLR   ,X                             PUT A ZERO AT END OF BUFFER WHEN DONE
1773   CCBA 8E Ø2 DC                LDX   #LINBUF                        POINT (X) TO LINBUF - RESET POINTER
1774   CCBD 39               LCCBD  RTS
1775
1776                         * CHECK FOR ITEM SEPARATOR OR TERMINATOR AND EXIT THE INPUT ROUTINE
1777   CCBE 81 22            LCCBE  CMPA  #'"'                           QUOTE?
1778   CCCØ 27 Ø4                   BEQ   LCCC6                          YES
1779   CCC2 81 2Ø                   CMPA  #SPACE                         SPACE?
1780   CCC4 26 F2                   BNE   LCCB8                          NO - EXIT ROUTINE
1781   CCC6 8D 2E            LCCC6  BSR   LCCF6                          GET A CHARACTER FROM CONSOLE IN
1782   CCC8 26 EE                   BNE   LCCB8                          EXIT ROUTINE IF BUFFER EMPTY
1783   CCCA 81 2Ø                   CMPA  #SPACE                         SPACE?
1784   CCCC 27 F8                   BEQ   LCCC6                          YES - GET ANOTHER CHARACTER
1785   CCCE 81 2C                   CMPA  #','                           COMMA (ITEM SEPARATOR)?
1786   CCDØ 27 E6                   BEQ   LCCB8                          YES - EXIT ROUTINE
1787   CCD2 81 ØD            LCCD2  CMPA  #CR                            CARRIAGE RETURN?
1788   CCD4 26 Ø8                   BNE   LCCDE                          NO
1789   CCD6 8D 1E            LCCD6  BSR   LCCF6                          GET A CHARACTER FROM CONSOLE IN
1790   CCD8 26 DE                   BNE   LCCB8                          EXIT ROUTINE IF BUFFER EMPTY
1791   CCDA 81 ØA                   CMPA  #LF                            LINE FEED? TREAT CR,LF AS A CR
1792   CCDC 27 DA                   BEQ   LCCB8                          YES - EXIT ROUTINE
1793   CCDE 8D 1C            LCCDE  BSR   LCCFC                          BACK UP PTR INPUT POINTER ONE
1794   CCEØ 2Ø D6                   BRA   LCCB8                          EXIT ROUTINE
1795
1796   CCE2 8D 12            LCCE2  BSR   LCCF6                          GET A CHAR FROM INPUT BUFFER - RETURN IN ACCA
1797   CCE4 27 15                   BEQ   LCCFB                          RETURN IF BUFFER NOT EMPTY
1798   CCE6 BD C7 14                JSR   LC714                          POINT X TO START OF FILE BUFFER
1799   CCE9 E6 ØØ                   LDB   FCBTYP,X                       GET FILE TYPE
1800   CCEB C1 4Ø                   CMPB  #RANFIL                        IS IT RANDOM FILE TYPE?
1801   CCED 1Ø 26 F6 43             LBNE  LC334                          'INPUT PAST END OF FILE  ERROR IF NOT RANDOM
1802   CCF1 C6 4A            LCCF1  LDB   #2*37                          'WRITE/INPUT PAST END OF RECORD  ERROR IF RANDOM
1803   CCF3 7E AC 46                JMP   LAC46                          JUMP TO THE ERROR HANDLER
1804
1805   CCF6 BD A1 76         LCCF6  JSR   LA176                          GET A CHAR FROM INPUT BUFFER
1806   CCF9 ØD 7Ø                   TST   CINBFL                         SET FLAGS ACCORDING TO CONSOLE INPUT FLAG
1807   CCFB 39               LCCFB  RTS
1808
1809                         * MOVE THE INPUT POINTER BACK ONE (DISK FILE)
1810   CCFC 34 14            LCCFC  PSHS  X,B                            SAVE REGISTERS ON STACK
1811   CCFE BD C7 14                JSR   LC714                          POINT X TO PROPER FCB
1812   CDØ1 E6 ØØ                   LDB   FCBTYP,X                       GET FILE TYPE OF THIS FCB
1813   CDØ3 C1 4Ø                   CMPB  #RANFIL                        IS IT A RANDOM FILE?
1814   CDØ5 26 ØB                   BNE   LCD12                          BRANCH IF NOT A RANDOM FILE
1815   CDØ7 EC 88 15                LDD   FCBGET,X                       *GRAB THE RANDOM FILE 'GET' POINTER,
1816   CDØA 83 ØØ Ø1                SUBD  #$ØØØ1                        *MOVE IT BACK ONE AND RESTORE IT
1817   CDØD ED 88 15                STD   FCBGET,X                       *
1818   CD1Ø 35 94                   PULS  B,X,PC                         RESTORE REGISTERS AND RETURN
1819   CD12 A7 88 11         LCD12  STA   FCBCDT,X                       SAVE THE CHARACTER IN THE CACHE
1820   CD15 63 88 1Ø                COM   FCBCFL,X                       SET THE CACHE FLAG TO $FF - DATA IN CACHE
1821   CD18 35 94                   PULS  B,X,PC                         RESTORE REGISTERS AND RETURN
1822
1823                         * CVN COMMAND
1824   CD1A BD B6 54         CVN    JSR   LB654                          GET LENGTH AND ADDRESS OF STRING
```

```
1825   CD1D C1 Ø5                  CMPB  #$Ø5                        FIVE BYTES IN A FLOATING POINT NUMBER
1826   CD1F 1Ø 25 E7 27            LBCS  LB44A                       'FC' ERROR IF <> 5 BYTES
1827   CD23 ØF Ø6                  CLR   VALTYP                      SET VARIABLE TYPE TO NUMERIC
1828   CD25 7E BC 14               JMP   LBC14                       COPY A PACKED FP NUMBER FROM (X) TO FPAØ
1829
1830                       * MKN$ COMMAND
1831   CD28 BD B1 43       MKN     JSR   LB143                       'TM' ERROR IF VALTYP=STRING
1832   CD2B C6 Ø5                  LDB   #$Ø5                        FIVE BYTES IN A FLOATING POINT NUMBER
1833   CD2D BD B5 ØF               JSR   LB5ØF                       RESERVE FIVE BYTES IN STRING SPACE
1834   CD3Ø BD BC 35               JSR   LBC35                       PACK FPAØ AND STORE IT IN STRING SPACE
1835   CD33 7E B6 9B               JMP   LB69B                       SAVE STRING DESCRIPTOR ON STRING STACK
1836
1837                       * LOC COMMAND
1838   CD36 8D Ø5          LOC     BSR   LCD3D                       POINT X TO FILE BUFFER
1839   CD38 EC Ø7                  LDD   FCBREC,X                    GET RECORD NUMBER (RANDOM FILE) OR SECTOR CTR (SEQUENTIAL)
1840   CD3A 7E B4 F4       LCD3A   JMP   GIVABF                      PUT ACCD IN FPAØ
1841
1842                       * STRIP A DEVICE NUMBER FROM A BASIC STATEMENT, SET PRINT
1843                       * PARAMETERS ACCORDING TO IT - ERROR IF FILE NOT
1844                       * OPEN. RETURN WITH (X) POINTING TO THAT FILE'S FCB
1845   CD3D 96 6F          LCD3D   LDA   DEVNUM                       * GET CURRENT DEVICE NUMBER AND
1846   CD3F 34 Ø2                  PSHS  A                           * SAVE IT ON THE STACK
1847   CD41 BD B1 43               JSR   LB143                       'TM' ERROR IF VALTYP=STRING
1848   CD44 BD A5 AE               JSR   LA5AE                       CHECK FOR VALID DEVICE NUMBER/SET PRINT PARAMETERS
1849   CD47 ØD 6F                  TST   DEVNUM                      * CHECK DEVICE NUMBER
1850   CD49 1Ø 2F E6 FD            LBLE  LB44A                       * BRANCH IF NOT DISK FILE 'ILLEGAL FUNCTION CALL'
1851   CD4D BD C7 14               JSR   LC714                       POINT (X) TO FILE BUFFER
1852   CD5Ø 35 Ø2                  PULS  A                           * GET OLD DEVICE NUMBER OFF OF THE STACK AND
1853   CD52 97 6F                  STA   DEVNUM                      * SAVE IT AS DEVICE NUMBER
1854   CD54 6D Ø Ø                 TST   FCBTYP,X                    IS FILE OPEN?
1855   CD56 1Ø 27 D6 A1            LBEQ  LA3FB                       'FILE NOT OPEN' ERROR IF NOT OPEN
1856   CD5A 39                     RTS
1857
1858                       * LOF COMMAND
1859   CD5B 8D EØ          LOF     BSR   LCD3D                       POINT X TO FILE BUFFER
1860   CD5D A6 Ø1                  LDA   FCBDRV,X                    * GET DRIVE NUMBER AND SAVE IT
1861   CD5F 97 EB                  STA   DCDRV                       * IN DSKCON VARIABLE
1862   CD61 E6 Ø2                  LDB   FCBFGR,X                    GET FIRST GRANULE OF FILE
1863   CD63 34 1Ø                  PSHS  X                           SAVE FCB POINTER ON STACK
1864   CD65 BD CC 44               JSR   LCC44                       FIND TOTAL NUMBER OF GRANULES IN THIS FILE
1865   CD68 4A                     DECA                              SUBTRACT THE LAST GRANULE IN THE FILE
1866   CD69 C4 3F                  ANDB  #$3F                        GET NUMBER OF SECTORS USED IN LAST GRANULE
1867   CD6B 34 Ø4                  PSHS  B                           SAVE NUMBER OF SECTORS IN LAST GRANULE ON STACK
1868   CD6D 1F 89                  TFR   A,B                         * CONVERT ACCA TO POSITIVE
1869   CD6F 4F                     CLRA                              * 2 BYTE VALUE IN ACCD
1870   CD7Ø BD C7 49               JSR   LC749                       MULT NUMBER OF FULL GRANULES BY 9
1871   CD73 EB EØ                  ADDB  ,S+                         ADD NUMBER SECTORS IN LAST TRACK
1872   CD75 89 Ø Ø                 ADCA  #$Ø Ø                       PROPAGATE CARRY TO MS BYTE OF ACCD
1873   CD77 35 1Ø                  PULS  X                           GET FCB POINTER BACK
1874   CD79 34 Ø2                  PSHS  A                           SAVE ACCA ON STACK
1875   CD7B A6 Ø Ø                 LDA   FCBTYP,X                    * GET FILE TYPE OF THIS FCB AND
1876   CD7D 81 4Ø                  CMPA  #RANFIL                     * CHECK TO SEE IF IT'S A RANDOM FILE
1877   CD7F 35 Ø2                  PULS  A                           RESTORE ACCA
1878   CD81 26 B7                  BNE   LCD3A                       *IF NOT A RANDOM FILE, THEN THE TOTAL NUMBER OF SECTORS IN THE FILE
1879                       *                                         *IS THE LENGTH OF THE FILE
1880
1881                       * CALCULATE LOF FOR A RANDOM FILE - THE LENGTH OF A RANDOM FILE IS THE
1882                       * NUMBER OF RECORDS IN THE FILE.
1883   CD83 34 1Ø                  PSHS  X                           SAVE FCB POINTER ON STACK
1884   CD85 93 8A                  SUBD  ZERO                        SUBTRACT ZERO FROM ACCD (NUMBER OF SECTORS)
1885   CD87 27 Ø3                  BEQ   LCD8C                       BRANCH IF ZERO SECTORS
1886   CD89 83 Ø Ø Ø1              SUBD  #$Ø Ø Ø1                    SUBTRACT ONE SECTOR - THE LAST SECTOR MAY NOT BE IOOZ USED
1887   CD8C 8D AC          LCD8C   BSR   LCD3A                       PUT ACCD INTO FPAØ
1888   CD8E D6 4F                  LDB   FPØEXP                      GET EXPONENT OF FPAØ
1889   CD9Ø 27 Ø4                  BEQ   LCD96                       BRANCH IF FPAØ = Ø
1890   CD92 CB Ø8                  ADDB  #$Ø8                        * ADD 8 TO EXPONENT (MULTIPLY FPAØ BY
1891   CD94 D7 4F                  STB   FPØEXP                      * 256 BYTES/SECTOR) AND SAVE NEW EXPONENT
1892   CD96 BD BC 5F       LCD96   JSR   LBC5F                       SAVE NUMBER OF BYTES IN FULL SECTORS IN FPA1
1893   CD99 AE E4                  LDX   ,S                          POINT X TO FCB
1894   CD9B EC 88 13               LDD   FCBLST,X                    GET NUMBER OF BYTES IN LAST SECTOR
1895   CD9E 84 7F                  ANDA  #$7F                        MASK OFF THE PRE-SAVED BYTE
1896   CDAØ 8D 98                  BSR   LCD3A                       PUT NUMBER BYTES IN LAST SECTOR INTO FPAØ
1897   CDA2 ØF 62                  CLR   RESSGN                      FORCE SUM SIGN = POSITIVE
1898   CDA4 96 5C                  LDA   FP1EXP                      * GET EXPONENTS OF FPAØ AND
1899   CDA6 D6 4F                  LDB   FPØEXP                      * FPA1 PRIOR TO ADDITION
1900   CDA8 BD B9 C5               JSR   LB9C5                       =ADD NUMBER BYTES IN LAST SECTOR TO NUMBER OF
1901                       *                                         =BYTES IN FULL SECTORS
1902   CDAB BD BC 5F               JSR   LBC5F                       SAVE TOTAL NUMBER OF BYTES IN FPA1
1903   CDAE 35 1Ø                  PULS  X                           POINT X TO FCB
1904   CDBØ EC Ø9                  LDD   FCBRLN,X                    * GET RECORD LENGTH
1905   CDB2 8D 86                  BSR   LCD3A                       * PUT IT INTO FPAØ
1906   CDB4 ØF 62                  CLR   RESSGN                      FORCE QUOTIENT SIGN = POSITIVE
1907   CDB6 96 5C                  LDA   FP1EXP                      * GET EXPONENTS OF FPAØ AND
1908   CDB8 D6 4F                  LDB   FPØEXP                      * FPA1 PRIOR TO DIVISION
1909   CDBA BD BB 91               JSR   LBB91                       DIVIDE TOTAL NUMBER OF BYTES BY NUMBER OF BYTES IN A RECORD
1910   CDBD 7E BC EE               JMP   INT                         CONVERT FPAØ TO AN INTEGER
1911
1912                       * FREE COMMAND
1913   CDCØ BD B1 43       FREE    JSR   LB143                       * NUMBER TYPE CHECK
1914   CDC3 BD B7 ØE               JSR   LB7ØE                       *EVALUATE NUMERIC EXPRESSION AND RETURN VALUE IN ACCB
1915   CDC6 C1 Ø3                  CMPB  #$Ø3                        ONLY 4 LEGAL DRIVES
1916   CDC8 1Ø 22 D8 53            LBHI  LA61F                       'DEVICE NUMBER' ERROR IF DRIVE NUMBER IS > 3
1917   CDCC D7 EB                  STB   DCDRV                       SAVE IN DRIVE NUMBER
1918   CDCE BD C7 6D               JSR   LC76D                       GET FILE ALLOCATION TABLE AND STORE IN BUFFER
1919   CDD1 BD C7 25               JSR   LC725                       POINT X TO START OF FILE ALLOCATION TABLE BUFFER
1920   CDD4 3Ø Ø6                  LEAX  FATCON,X                    MOVE TO FIRST GRANULE DATA BYTE
```

```
1921   CDD6 6F E2                CLR   ,-S                        SPACE FOR FREE GRANULE COUNTER
1922   CDD8 C6 44                LDB   #GRANMX                    GET MAXIMUM NUMBER OF GRANULES
1923   CDDA A6 80          LCDDA LDA   ,X+                        GET GRANULE DATA
1924   CDDC 43                   COMA                             *FREE GRANULES $FF
1925   CDDD 26 02                BNE   LCDE1                      *BRANCH IF NOT FREE
1926   CDDF 6C E4                INC   ,S                         INCREMENT FREE GRANULE COUNTER
1927   CDE1 5A             LCDE1 DECB                             DECREMENT GRANULE COUNTER
1928   CDE2 26 F6                BNE   LCDDA                      BRANCH IF NOT DONE
1929   CDE4 35 04                PULS  B                          GET FREE GRANULE COUNTER TO ACCB
1930   CDE6 7E B4 F3             JMP   LB4F3                      LOAD ACCB INTO FPA0
1931
1932                       * DRIVE COMMAND
1933   CDE9 BD B7 0B       DRIVE JSR   EVALEXPB                   EVALUATE EXPR; RETURN VALUE IN ACCB
1934   CDEC C1 03                CMPB  #$03                       MAX DRIVE NUMBER = 3
1935   CDEE 10 22 D8 2D          LBHI  LA61F                      'DEVICE #' ERROR IF DRIVE NUMBER > 3
1936   CDF2 F7 09 5A             STB   DEFDRV                     SAVE DEFAULT DRIVE NUMBER
1937   CDF5 39                   RTS
1938
1939                       * EVALUATE EXPRESSION RAM VECTOR
1940   CDF6 A6 64          DVEC15 LDA  $04,S                      = CHECK STACKED PRECEDENCE FLAG AND IF IT IS NOT AN END
1941   CDF8 26 13                BNE   LCE0D                      = OF OPERATION, BRANCH TO EXTENDED BASIC'S EXPRESSION
1942                       *                                      = EVALUATION ROUTINE
1943   CDFA AE 65                LDX   $05,S                      *
1944   CDFC 8C AF 9A             CMPX  #LAF9A                     *
1945   CDFF 26 0C                BNE   LCE0D                      * CHECK TWO RETURN ADDRESSES BACK ON THE STACK
1946   CE01 AE 62                LDX   $02,S                      * TO SEE IF THE CALL TO EVALUATE EXPRESSION IS
1947   CE03 8C B1 66             CMPX  #LB166                     * COMING FROM THE 'LET' COMMAND - BRANCH OUT IF
1948   CE06 26 05                BNE   LCE0D                      * NOT COMING FROM 'LET'
1949   CE08 8E CE 10             LDX   #LCE10                     = IF COMING FROM 'LET', REPLACE THE RETURN ADDR
1950   CE0B AF 65                STX   $05,S                      = WITH THE DISK BASIC 'LET' MODIFIER ADDRESS
1951   CE0D 7E 88 46       LCE0D JMP   XVEC15                     EXTENDED BASIC EXPRESSION EVALUATION
1952
1953                       * LET MODIFIER
1954   CE10 35 02          LCE10 PULS  A                          PULL VARIABLE TYPE OFF OF THE STACK
1955   CE12 46                   RORA                             SET CARRY IF SIRING, CLEAR CARRY IF NUMERIC
1956   CE13 BD B1 48             JSR   LB148                      DO A 'TM' CHECK
1957   CE16 10 27 EE 19          LBEQ  LBC33                      IF NUMERIC VARIABLE, PACK FPA0 INTO VARDES
1958   CE1A 9E 52                LDX   FPA0+2                     POINT X TO STRING DESCRIPTOR
1959   CE1C EC 02                LDD   $02,X                      GET ADDRESS OF SIRING
1960   CE1E 10 83 09 89          CMPD  #DFLBUF                    * COMPARE TO START OF RANDOM FILE BUFFERS
1961   CE22 25 07                BLO   LCE2B                      * AND BRANCH IF LOWER
1962   CE24 B3 09 4A             SUBD  FCBADR                     SUBTRACT OUT THE END OF RANDOM FILE BUFFERS
1963   CE27 10 25 E1 86          LBCS  LAFB1                      *BRANCH IF STRING STORED IN RANDOM FILE BUFFER -
1964                       *                                      *MOVE IT INTO THE STRING SPACE
1965   CE2B 7E AF A4       LCE2B JMP   LAFA4                      BRANCH BACK TO BASIC S 'LET' COMMAND
1966
1967                       *MODIFIER FOR EXBAS COMMAND INTERPRETATION HANDLER
1968   CE2E 81 CA          DXCVEC CMPA #$CA                       TOKEN FOR DLOAD?
1969   CE30 27 1C                BEQ   LCE4E                      YES
1970   CE32 81 C8                CMPA  #$C8                       TOKEN FOR PMODE?
1971   CE34 10 26 B3 04          LBNE  L813C                      NO
1972                       * DISK BASIC MODIFIER FOR PMODE - ALLOWS FOR THE RAM THE DOS USES
1973   CE38 9D 9F                JSR   GETNCH                     GET NEXT CHARACTER FROM BASIC
1974   CE3A 81 2C                CMPA  #','                       CHECK FOR COMMA
1975   CE3C 10 27 C8 10          LBEQ  L9650                      BRANCH IF COMMA
1976   CE40 BD B7 0B             JSR   EVALEXPB                   EVALUATE EXPRESSION; RETURN VALUE IN ACCB
1977   CE43 C1 04                CMPB  #$04                       CHECK FOR PMODE 4
1978   CE45 10 22 E6 01          LBHI  LB44A                      'FC' ERROR IF PMODE > 4
1979   CE49 96 BC                LDA   GRPRAM                     NUMBER BLOCKS BEFORE GRAPHICS PAGES
1980   CE4B 7E 96 2E             JMP   L962E                      JUMP TO EXEAS' PMODE COMMAND
1981
1982                       * DISK BASIC DLOAD MODIFIER
1983   CE4E BD A4 29       LCE4E JSR   LA429                      CLOSE FILES
1984   CE51 9D 9F                JSR   GETNCH                     GET NEXT CHARACTER FROM BASIC
1985   CE53 7E 8C 1B             JMP   L8C1B                      JUMP TO EXEAS' DLOAD
1986
1987   CE56 C1 34          DXIVEC CMPB #($9A-$80)*2               MODIFIED TOKEN FOR POS
1988   CE58 10 26 B3 0C          LBNE  L8168                      IF NOT POS, GO TO EXBAS SECONDARY COMM HANDLER
1989   CE5C BD B2 62             JSR   LB262                      SYNTAX CHECK FOR '(' AND EVALUATE EXPRESSION
1990   CE5F 96 6F                LDA   DEVNUM                     * GET DEVICE NUMBER AND
1991   CE61 34 02                PSHS  A                          * SAVE IT ON STACK
1992   CE63 BD A5 AE             JSR   LA5AE                      EVALUATE DEVICE NUMBER
1993   CE66 BD A4 06             JSR   LA406                      TEST DEVICE NUMBER
1994   CE69 0D 6F                TST   DEVNUM                     * CHECK DEVICE NUMBER AND BRANCH
1995   CE6B 2F 13                BLE   LCE80                      * IF NOT A DISK FILE
1996   CE6D BD C7 14             JSR   LC714                      POINT X TO FCB
1997   CE70 E6 00                LDB   FCBTYP,X                   GET FILE TYPE
1998   CE72 C1 40                CMPB  #RANFIL                    DIRECT/RANDOM FILE?
1999   CE74 26 0A                BNE   LCE80                      BRANCH IF NOT A RANDOM FILE
2000   CE76 35 02                PULS  A                          * RESTORE DEVICE NUMBER
2001   CE78 97 6F                STA   DEVNUM                     *
2002   CE7A EC 88 17             LDD   FCBPUT,X                   =GRAB THE 'PUT' DATA ITEM COUNTER AND CONVERT
2003   CE7D 7E B4 F4             JMP   GIVABF                     =IT TO A FLOATING POINT NUMBER
2004   CE80 BD A3 5F       LCE80 JSR   LA35F                      SET PRINT PARAMETERS
2005   CE83 35 02                PULS  A                          * RESTORE DEVICE NUMBER
2006   CE85 97 6F                STA   DEVNUM                     *
2007   CE87 D6 6C                LDB   DEVPOS                     =GET PRINT POSITION AND
2008   CE89 7E B4 F3             JMP   LB4F3                      =CONVERT IT TO FLOATING POINT NUMBER IN FPA0
2009
2010                       * SAVEM COMMAND
2011   CE8C 9D 9F          LCE8C JSR   GETNCH                     GET NEXT INPUT CHARACTER
2012   CE8E 8D 4F                BSR   LCEDF                      GET FILENAME, ETC.
2013   CE90 BD 83 6C             JSR   L836C                      EVALUATE EXPRESSION, PUT II (2 BYTES) ON STACK
2014   CE93 BD 83 6C             JSR   L836C                      DITTO
2015   CE96 AC 62                CMPX  $02,S                      COMPARE END ADDRESS TO START ADDRESS
2016   CE98 10 25 E5 AE          LBCS  LB44A                      IF START > END, THEN 'ILLEGAL FUNCTION CALL'
```

```
2017  CE9C BD 83 6C              JSR   L836C                    EVAL EXPRESSION (TRANSFER ADDRESS), PUT ON STACK
2018  CE9F BD A5 C7              JSR   LA5C7                    SYNTAX ERROR IF ANY MORE CHARS ON THIS LINE
2019  CEA2 CC 02 00              LDD   #$0200                   * FILE TYPE=2, ASCII FLAG = CRUNCHED (0)
2020  CEA5 FD 09 57              STD   DFLTYP                   *
2021  CEA8 BD C9 56              JSR   LC956                    GET NEXT UNOPEN FILE AND INITIALIZE FCB
2022  CEAB 4F                    CLRA                           *ZERO FLAG - FIRST BYTE OF PREAMBLE
2023  CEAC 8D 2B                 BSR   LCED9                    *WRITE A BYTE TO BUFFER
2024  CEAE EC 62                 LDD   $02,S                    GET END ADDRESS
2025  CEB0 A3 64                 SUBD  $04,S                    SUBTRACT THE START ADDRESS
2026  CEB2 C3 00 01              ADDD  #$0001                   THE SAVED DATA BLOCK WILL INCLUDE BOTH THE FIRST AND LAST BYTES
2027  CEB5 1F 02                 TFR   D,Y                      SAVE LENGTH IN Y
2028  CEB7 8D 1E                 BSR   LCED7                    WRITE FILE LENGTH TO BUFFER - FIRST ARGUMENT OF PREAMBLE
2029  CEB9 EC 64                 LDD   $04,S                    GET THE START ADDRESS
2030  CEBB 8D 1A                 BSR   LCED7                    WRITE OUT THE START ADDRESS - SECOND PREAMBLE ARGUMENT
2031  CEBD AE 64                 LDX   $04,S                    GET START ADDRESS
2032  CEBF A6 80          LCEBF  LDA   ,X+                      GRAB A BYTE
2033  CEC1 BD CB 52              JSR   LCB52                    WRITE IT OUT
2034  CEC4 31 3F                 LEAY  -1,Y                     DECREMENT BYTE COUNTER
2035  CEC6 26 F7                 BNE   LCEBF                    BRANCH IF ALL BYTES NOT DONE
2036  CEC8 86 FF                 LDA   #$FF                     FIRST BYTE OF POSTAMBLE
2037  CECA 8D 0D                 BSR   LCED9                    WRITE IT OUT - EOF RECORD
2038  CECC 4F                    CLRA                           * FIRST ARGUMENT OF POSTAMBLE IS
2039  CECD 5F                    CLRB                           * A DUMMY - ZERO VALUE
2040  CECE 8D 07                 BSR   LCED7                    WRITE OUT POSTAMBLE FIRST ARGUMENT
2041  CED0 35 36                 PULS  A,B,X,Y                  GET CONTROL ADDRESSES FROM THE STACK
2042  CED2 8D 03                 BSR   LCED7                    WRITE OUT THE TRANSFER ADDRESS - 2ND ARGUMENT
2043  CED4 7E A4 2D              JMP   LA42D                    GO CLOSE ALL FILES
2044
2045                       * WRITE ACCD TO THE BUFFER
2046  CED7 8D 00          LCED7  BSR   LCED9                    WRITE ACCA TO BUFFER, THEN SWAP ACCA,ACCB
2047  CED9 BD CB 52       LCED9  JSR   LCB52                    WRITE ACCA TO BUFFER
2048  CEDC 1E 89                 EXG   A,B                      SWAP ACCA,ACCB
2049  CEDE 39                    RTS
2050  CEDF 8E C2 97       LCEDF  LDX   #BINEXT                  POINT TO .BIN EXTENSION
2051  CEE2 7E C8 8A              JMP   LC88A                    GET FILENAME, ETC.
2052
2053                       * LOADM COMMAND
2054  CEE5 9D 9F          LCEE5  JSR   GETNCH                   GET NEXT INPUT CHARACTER
2055  CEE7 8D F6                 BSR   LCEDF                    GET FILENAME, ETC.
2056  CEE9 BD C9 59              JSR   LC959                    OPEN NEXT AVAILABLE FILE FOR INPUT
2057  CEEC FC 09 57              LDD   DFLTYP                   GET FILE TYPE AND ASCII FLAG
2058  CEEF 83 02 00              SUBD  #$0200                   FOR LOADM FILE: TYPE=2, ASCII FLAG=0
2059  CEF2 10 26 D7 20           LBNE  LA616                    'BAD FILE MODE' ERROR
2060  CEF6 9E 8A                 LDX   ZERO                     ZERO OUT X REG - DEFAULT VALUE OF OFFSET
2061  CEF8 9D A5                 JSR   GETCCH                   GET CURRENT CHARACTER FROM BASIC
2062  CEFA 27 06                 BEQ   LCF02                    BRANCH IF END OF LINE - NO OFFSET
2063  CEFC BD B2 6D              JSR   SYNCOMMA                 SYNTAX CHECK FOR COMMA
2064  CEFF BD B7 3D              JSR   LB73D                    EVALUATE EXPRESSION
2065  CF02 9F D3          LCF02  STX   VD3                      STORE OFFSET IN VD3
2066  CF04 BD A5 C7              JSR   LA5C7                    SYNTAX ERROR IF OTHER CHARACTERS ON LINE
2067
2068                       * GET PREAMBLE/POSTAMBLE
2069  CF07 BD CC E2       LCF07  JSR   LCCE2                    GET FIRST BYTE
2070  CF0A 34 02                 PSHS  A                        SAVE IT ON THE STACK
2071  CF0C 8D 29                 BSR   LCF37                    GET FIRST ARGUMENT
2072  CF0E 1F 02                 TFR   D,Y                      SAVE IT IN Y
2073  CF10 8D 25                 BSR   LCF37                    GET THE SECOND ARGUMENT
2074  CF12 D3 D3                 ADDD  VD3                      ADD IT TO THE OFFSET
2075  CF14 DD 9D                 STD   EXECJP                   STORE IT IN THE JUMP ADDRESS OF THE EXEC COMMAND
2076  CF16 1F 01                 TFR   D,X                      SAVE IT IN X
2077  CF18 A6 E0                 LDA   ,S+                      GET THE FIRST BYTE OFF OF THE STACK
2078  CF1A 10 26 D5 0F           LBNE  LA42D                    CLOSE FILE IF POSTAMBLE (EOF)
2079
2080                       * GET RECORD BYTE(S)
2081  CF1E BD C5 97       LCF1E  JSR   LC597                    GET BYTE FROM BUFFER
2082  CF21 D6 70                 LDB   CINBFL                   GET STATUS OF CONSOLE IN BUFFER
2083  CF23 27 03                 BEQ   LCF28                    BRANCH IF BUFFER NOT EMPTY
2084  CF25 7E C3 34              JMP   LC334                    'INPUT PAST END OF FILE' ERROR
2085  CF28 A7 84          LCF28  STA   ,X                       STORE BYTE IN MEMORY
2086  CF2A A1 80                 CMPA  ,X+                      *TEST TO SEE IF IT STORED PROPERLY AND
2087  CF2C 27 03                 BEQ   LCF31                    *BRANCH IF PROPER STORE (NOT IN ROM OR BAD RAM)
2088  CF2E 7E D6 16              JMP   LD616                    'I/O ERROR' IF BAD STORE
2089  CF31 31 3F          LCF31  LEAY  -1,Y                     DECREMENT BYTE COUNT
2090  CF33 26 E9                 BNE   LCF1E                    GET NEXT BYTE IF NOT DONE
2091  CF35 20 D0                 BRA   LCF07                    READ ANOTHER PRE/POST AMBLE
2092                       * READ TWO BYTES FROM BUFFER - RETURN THEM IN ACCD
2093  CF37 8D 00          LCF37  BSR   LCF39                    READ A BYTE, SAVE IT IN ACCB
2094  CF39 BD CC E2       LCF39  JSR   LCCE2                    GET A CHARACTER FROM INPUT BUFFER, RETURN IT IN ACCA
2095  CF3C 1E 89                 EXG   A,B                      SWAP ACCA,ACCB
2096  CF3E 39                    RTS
2097
2098                       * RENAME COMMAND
2099  CF3F 9E A6          RENAME LDX   CHARAD                   * SAVE CURRENT INPUT POINTER
2100  CF41 34 10                 PSHS  X                        * ON THE STACK
2101  CF43 8D 35                 BSR   LCF7A                    GET FILENAME OF SOURCE FILE
2102  CF45 96 EB                 LDA   DCDRV                    * SAVE DRIVE NUMBER
2103  CF47 34 02                 PSHS  A                        * ON THE STACK
2104  CF49 8D 2A                 BSR   LCF75                    SYNTAX CHECK FOR 'TO' AND GET NEW FILENAME
2105  CF4B 35 02                 PULS  A                        GET SOURCE DRIVE NUMBER
2106  CF4D 91 EB                 CMPA  DCDRV                    COMPARE TO NEW FILE DRIVE NUMBER
2107  CF4F 10 26 E4 F7           LBNE  LB44A                    'FC' ERROR IF FILES ON DIFFERENT DRIVES
2108  CF53 8D 28                 BSR   LCF7D                    VERIFY THAT NEW FILE DOES NOT ALREADY EXIST
2109  CF55 35 10                 PULS  X                        * RESTORE INPUT POINTER
2110  CF57 9F A6                 STX   CHARAD                   *
2111  CF59 8D 1F                 BSR   LCF7A                    GET SOURCE FILENAME AGAIN
2112  CF5B BD C6 5F              JSR   LC65F                    SCAN DIRECTORY FOR SOURCE FILENAME
```

```
2113   CF5E BD C6 B8                JSR   LC6B8                          'NE' ERROR IF NOT FOUND
2114   CF61 8D 12                   BSR   LCF75                          SYNTAX CHECK FOR 'TO' AND GET NEW FILENAME
2115   CF63 8E Ø9 4C                LDX   #DNAMBF                        POINT X TO FILENAME
2116   CF66 FE Ø9 74                LDU   V974                           POINT U TO DIRECTORY ENTRY OF SOURCE FILE
2117   CF69 C6 ØB                   LDB   #11                            11 CHARACTERS IN FILENAME AND EXTENSION
2118   CF6B BD A5 9A                JSR   LA59A                          COPY NEW FILENAME TO SOURCE FILE DIRECTORY RAM IMAGE
2119   CF6E C6 Ø3                   LDB   #$Ø3                           * GET WRITE OP CODE AND
2120   CF7Ø D7 EA                   STB   DCOPC                          * SAVE IN DSKCON VARIABLE
2121   CF72 7E D5 FF                JMP   LD5FF                          WRITE NEW DIRECTORY SECTOR
2122
2123                        * DO A SYNTAX CHECK FOR 'TO  AND STRIP A FILENAME FROM BASIC
2124   CF75 C6 A5           LCF75   LDB   #$A5                           'TO' TOKEN
2125   CF77 BD B2 6F                JSR   LB26F                          SYNTAX CHECK FOR 'TO'
2126   CF7A 7E C8 87        LCF7A   JMP   LC887                          GET FILENAME FROM BASIC
2127   CF7D BD C6 5F        LCF7D   JSR   LC65F                          SCAN DIRECTORY FOR FILENAME
2128   CF8Ø C6 42                   LDB   #33*2                          'FILE ALREADY EXISTS' ERROR
2129   CF82 7D Ø9 73                TST   V973                           CHECK FOR A MATCH
2130   CF85 1Ø 26 DC BD             LBNE  LAC46                          'AE' ERROR IF FILE IN DIRECTORY
2131   CF89 39                      RTS
2132
2133                        * WRITE COMMAND
2134   CF8A 1Ø 27 E9 CA     WRITE   LBEQ  LB958                          PRINT CARRIAGE RETURN TO CONSOLE OUT IF END OF LINE
2135   CF8E 8D Ø3                   BSR   LCF93                          GO WRITE AN ITEM LIST
2136   CF9Ø ØF 6F                   CLR   DEVNUM                         SET DEVICE NUMBER TO SCREEN
2137   CF92 39              LCF92   RTS
2138   CF93 81 23           LCF93   CMPA  #'#'                           CHECK FOR DEVICE NUMBER FLAG
2139   CF95 26 ØF                   BNE   LCFA6                          DEFAULT TO CURRENT DEVICE NUMBER IF NONE GIVEN
2140   CF97 BD A5 A5                JSR   LA5A5                          SET DEVICE NUMBER; CHECK VALIDITY
2141   CF9A BD A4 Ø6                JSR   LA4Ø6                          MAKE SURE SELECTED FILE IS AN OUTPUT FILE
2142   CF9D 9D A5                   JSR   GETCCH                         GET CURRENT INPUT CHARACTER
2143   CF9F 1Ø 27 E9 B5             LBEQ  LB958                          PRINT CR TO CONSOLE OUT IF END OF LINE
2144   CFA3 BD B2 6D        LCFA3   JSR   SYNCOMMA                       SYNTAX CHECK FOR COMMA
2145   CFA6 BD B1 56        LCFA6   JSR   LB156                          EVALUATE EXPRESSION
2146   CFA9 96 Ø6                   LDA   VALTYP                         GET VARIABLE TYPE
2147   CFAB 26 1E                   BNE   LCFCB                          BRANCH IF STRING
2148   CFAD BD BD D9                JSR   LBDD9                          CONVERT FP NUMBER TO ASCII STRING
2149   CFBØ BD B5 16                JSR   LB516                          PUT ON TEMPORARY STRING STACK
2150   CFB3 BD B9 9F                JSR   LB99F                          PRINT STRING TO CONSOLE OUT
2151
2152                        * PRINT ITEM SEPARATOR TO CONSOLE OUT
2153   CFB6 9D A5           LCFB6   JSR   GETCCH                         GET CURRENT CHARACTER
2154   CFB8 1Ø 27 E9 9C             LBEQ  LB958                          PUT CR TO CONSOLE OUT IF END OF LINE
2155   CFBC 86 2C                   LDA   #','                           COMMA: NON-CASSETTE SEPARATOR
2156   CFBE BD A3 5F                JSR   LA35F                          SET PRINT PARAMETERS
2157   CFC1 ØD 6E                   TST   PRTDEV                         * GET CONSOLE PRINT DEVICE AND
2158   CFC3 27 Ø2                   BEQ   LCFC7                          * BRANCH IF NOT CASSETTE
2159   CFC5 86 ØD                   LDA   #CR                            GET CARRIAGE RETURN - CASSETTE ITEM SEPARATOR
2160   CFC7 8D 14           LCFC7   BSR   LCFDD                          SEND SEPARATOR TO CONSOLE OUT
2161   CFC9 2Ø D8                   BRA   LCFA3                          GET NEXT ITEM
2162
2163                        * PRINT A STRING TO CONSOLE OUT
2164   CFCB 8D Ø7           LCFCB   BSR   LCFD4                          PRINT LEADING STRING DELIMITER (")
2165   CFCD BD B9 9F                JSR   LB99F                          PRINT STRING TO CONSOLE OUT
2166   CFDØ 8D Ø2                   BSR   LCFD4                          PRINT ENDING STRING DELIMITER (")
2167   CFD2 2Ø E2                   BRA   LCFB6                          GO PRINT SEPARATOR
2168
2169                        * PRINT STRING DELIMITER (") TO CONSOLE OUT
2170   CFD4 BD A3 5F        LCFD4   JSR   LA35F                          SET PRINT PARAMETERS
2171   CFD7 ØD 6E                   TST   PRTDEV                         * GET CONSOLE PRINT DEVICE AND
2172   CFD9 26 B7                   BNE   LCF92                          * RETURN IF CASSETTE
2173   CFDB 86 22                   LDA   #'"'                           QUOTE: NON-CASSETTE STRING DELIMITER
2174   CFDD 7E A2 82        LCFDD   JMP   LA282                          SEND TO CONSOLE OUT
2175
2176                        * FIELD COMMAND
2177   CFEØ BD C7 FE        FIELD   JSR   LC7FE                          EVALUATE DEVICE NUMBER & VERIFY RANDOM FILE OPEN
2178   CFE3 4F                      CLRA                                 *
2179   CFE4 5F                      CLRB                                 * CLEAR TOTAL FIELD LENGTH COUNTER
2180   CFE5 34 16                   PSHS  X,B,A                          SAVE FCB POINTER & INITIALIZE TOTAL FIELD LENGTH TO ZERO
2181   CFE7 9D A5           LCFE7   JSR   GETCCH                         GET CURRENT INPUT CHARACTER
2182   CFE9 26 Ø2                   BNE   LCFED                          BRANCH IF NOT END OF LINE
2183   CFEB 35 96                   PULS  A,B,X,PC                       CLEAN UP STACK AND RETURN
2184   CFED BD B7 38        LCFED   JSR   LB738                          SYNTAX CHECK FOR COMMA, EVALUATE EXPRESSION
2185   CFF0 34 14                   PSHS  X,B                            *SAVE FIELD LENGTH (ACCB) ON STACK, X IS A DUMMY WHICH WILL
2186                        *                                            *RESERVE 2 BYTES FOR THE ADDRESS WHICH WILL BE CALCULATED BELOW
2187                        * AT THIS POINT THE STACK WILL HAVE THE FOLLOWING INFORMATION ON IT:
2188                        * ,S = FIELD LENGTH   1 2,S = RANDOM FILE BUFFER ADDRESS
2189                        * 3 4,S = TOTAL FIELD LENGTH  5 6,S = FCD POINTER
2190   CFF2 4F                      CLRA                                 CLEAR MS BYTE
2191   CFF3 E3 63                   ADDD  $Ø3,S                          ADD FIELD LENGTH TO TOTAL FIELD LENGTH COUNTER
2192   CFF5 25 Ø7                   BLO   LCFFE                          'FO' ERROR IF SUM > $FFFF
2193   CFF7 AE 65                   LDX   $Ø5,S                          POINT X TO FCB
2194   CFF9 1Ø A3 Ø9                CMPD  FCBRLN,X                       * COMPARE TO RECORD LENGTH & BRANCH IF
2195   CFFC 23 Ø5                   BLS   LDØØ3                          *TOTAL FIELD LENGTH < RECORD LENGTH
2196   CFFE C6 44           LCFFE   LDB   #34*2                          'FIELD OVERFLOW' ERROR
2197   DØØØ 7E AC 46                JMP   LAC46                          JUMP TO ERROR DRIVER
2198   DØØ3 EE 63           LDØØ3   LDU   $Ø3,S                          LOAD U WITH OLD TOTAL LENGTH OF ALL FIELDS
2199   DØØ5 ED 63                   STD   $Ø3,S                          SAVE NEW TOTAL FIELD LENGTH
2200   DØØ7 EC ØB                   LDD   FCBBUF,X                       POINT ACCD TO START OF RANDOM FILE BUFFER
2201   DØØ9 33 CB                   LEAU  D,U                            *POINT U TO THIS FIELD'S SLOT IN THE RANDOM
2202   DØØB EF 61                   STU   $Ø1,S                          *FILE BUFFER AND SAVE IT ON THE STACK
2203   DØØD C6 FF                   LDB   #$FF                           SECONDARY TOKEN
2204   DØØF BD B2 6F                JSR   LB26F                          SYNTAX CHECK FOR SECONDARY TOKEN
2205   DØ12 C6 A7                   LDB   #$A7                           'AS' TOKEN
2206   DØ14 BD B2 6F                JSR   LB26F                          SYNTAX CHECK FOR 'AS' TOKEN
2207   DØ17 BD B3 57                JSR   LB357                          EVALUATE VARIABLE
2208   DØ1A BD B1 46                JSR   LB146                          'TM' ERROR IF NUMERIC VARIABLE
```

```
2209  D01D 35 44              PULS  B,U                 * PULL STRING ADDRESS AND LENGTH
2210  D01F E7 84              STB   ,X                  * OFF OF THE STACK AND SAVE THEM
2211  D021 EF 02              STU   $02,X               * IN STRING DESCRIPTOR
2212  D023 20 C2              BRA   LCFE7               CHECK FOR ANOTHER FIELD SPECIFICATION
2213
2214                        * RSET COMMAND
2215  D025 86 4F            RSET    LDA   #$4F          SKIP ONE BYTE
2216
2217                        * LSET COMMAND
2218  D026 4F               LSET    CLRA                LSET FLAG = 0
2219  D027 34 02              PSHS  A                   SAVE RSET($4F),LSET(00) FLAG ON THE STACK
2220  D029 BD B3 57           JSR   LB357               EVALUATE FIELD STRING VARIABLE
2221  D02C BD B1 46           JSR   LB146               'TM' ERROR IF NUMERIC VARIABLE
2222  D02F 34 10              PSHS  X                   SAVE STRING DESCRIPTOR ON STACK
2223  D031 AE 02              LDX   $02,X               POINT X TO ADDRESS OF STRING
2224  D033 8C 09 89           CMPX  #DFLBUF             * COMPARE STRING ADDRESS TO START OF RANDOM
2225  D036 25 05              BLO   LD03D               * FILE BUFFER; 'SE' ERROR IF < RANDOM FILE BUFFER
2226  D038 BC 09 4A           CMPX  FCBADR              = COMPARE STRING ADDRESS TO TOP OF RANDOM FILE BUFFER
2227  D03B 25 05              BLO   LD042               = AREA - BRANCH IF STRING IN RANDOM FILE BUFFER
2228  D03D C6 46            LD03D   LDB   #2*35         'SET TO NON-FIELDED STRING' ERROR
2229  D03F 7E AC 46           JMP   LAC46               JUMP TO ERROR HANDLER
2230  D042 C6 B3            LD042   LDB   #$B3          *
2231  D044 BD B2 6F           JSR   LB26F               * SYNTAX CHECK FOR '=' TOKEN
2232  D047 BD 87 48           JSR   L8748               =EVALUATE DATA STRING EXPRESSION; RETURN WITH X
2233                        *                           =POINTING TO STRING; ACCB = LENGTH
2234  D04A 35 20              PULS  Y                   POINT Y TO FIELD STRING DESCRIPTOR
2235  D04C A6 A4              LDA   ,Y                  GET LENGTH OF FIELD STRING
2236  D04E 27 2E              BEQ   LD07E               RETURN IF NULL STRING
2237  D050 34 04              PSHS  B                   SAVE LENGTH OF DATA STRING ON STACK
2238  D052 C6 20              LDB   #SPACE              PREPARE TO FILL DATA STRING WITH BLANKS
2239  D054 EE 22              LDU   $02,Y               POINT U TO FIELD STRING ADDRESS
2240                        * FILL THE FIELDED STRING WITH BLANKS
2241  D056 E7 C0            LD056   STB   ,U+           STORE A SPACE IN FIELDED STRING
2242  D058 4A                 DECA                      DECREMENT LENGTH COUNTER
2243  D059 26 FB              BNE   LD056               KEEP FILLING W/SPACES IF NOT DONE
2244  D05B E6 E0              LDB   ,S+                 *GET THE LENGTH OF THE DATA STRING AND
2245  D05D 27 1F              BEQ   LD07E               *RETURN IF IT IS NULL (ZERO)
2246  D05F E1 A4              CMPB  ,Y                  =COMPARE LENGTH OF DATA STRING TO LENGTH OF FIELD
2247  D061 25 04              BLO   LD067               =STRING, BRANCH IF FIELD STRING > DATA STRING
2248  D063 E6 A4              LDB   ,Y                  *GET THE LENGTH OF THE FIELD STRING AND FORCE THE
2249  D065 6F E4              CLR   ,S                  *RSET/LSET FLAG TO LSET (0) IF DATA STRING LENGTH IS
2250                        *                           *>= THE FIELD STRING LENGTH. THIS WILL CAUSE THE RIGHT
2251                        *                           *SIDE OF THE DATA STRING TO BE TRUNCATED
2252  D067 EE 22            LD067   LDU   $02,Y         LOAD U WITH THE ADDRESS OF THE FIELD STRING
2253  D069 6D E0              TST   ,S+                 * GET THE RSET/LSET FLAG FROM THE STACK
2254  D06B 27 0E              BEQ   LD07B               * AND BRANCH IF LSET
2255                        * RSET ROUTINE
2256  D06D 34 04              PSHS  B                   SAVE THE NUMBER OF BYTES TO MOVE INTO THE FIELD STRING
2257  D06F 4F                 CLRA                      = TAKE THE 2'S COMPLEMENT OF AN UNSIGNED
2258  D070 50                 NEGB                      = NUMBER IN ACCB - LEAVE THE DOUBLE BYTE SIGNED
2259  D071 82 00              SBCA  #$00                = RESULT IN ACCD
2260  D073 EB A4              ADDB  ,Y                  * ADD THE LENGTH OF THE FIELD STRING TO THE INVERSE
2261  D075 89 00              ADCA  #$00                * OF THE NUMBER OF BYTES TO BE MOVED
2262  D077 33 CB              LEAU  D,U                 =ADD RESULT TO START OF FIELD STRING. NOW U
2263                                                    =WILL POINT TO (-NUMBER OF BYTES TO MOVE)
2264                                                    =FROM THE RIGHT SIDE OF THE FIELD STRING
2265  D079 35 04              PULS  B                   GET THE NUMBER OF BYTES TO MOVE
2266  D07B 7E A5 9A         LD07B   JMP   LA59A         MOVE ACCB BYTES FROM X TO U (DATA TO FIELD STRING)
2267  D07E 35 82            LD07E   PULS  A,PC          PULL LSET/RSET FLAG OFF OF STACK AND RETURN
2268
2269                        * FILES COMMAND
2270  D080 BD 95 AC         FILES   JSR   L95AC         RESET SAM DISPLAY PAGE AND VDG MODE
2271  D083 FC 09 4A           LDD   FCBADR              GET START OF FILE BUFFERS
2272  D086 83 09 89           SUBD  #DFLBUF             SUBTRACT THE START OF RANDOM FILE BUFFER SPACE
2273  D089 34 06              PSHS  B,A                 SAVE DEFAULT VALUE OF RANDOM FILE BUFFER SPACE ON STACK
2274  D08B F6 09 5B           LDB   FCBACT              * GET CURRENT NUMBER OF FCBS
2275  D08E 34 04              PSHS  B                   * AND SAVE ON THE STACK (DEFAULT VALUE)
2276  D090 9D A5              JSR   GETCCH              GET CURRENT INPUT CHAR
2277  D092 81 2C              CMPA  #','                CHECK FOR COMMA
2278  D094 27 0F              BEQ   LD0A5               BRANCH IF COMMA - NO BUFFER NUMBER PARAMETER GIVEN
2279  D096 BD B7 0B           JSR   EVALEXPB            EVALUATE EXPRESSION (BUFFER NUMBER)
2280  D099 C1 0F              CMPB  #15                 15 FCBS MAX
2281  D09B 10 22 E3 AB        LBHI  LB44A               BRANCH IF > 15 - 'ILLEGAL FUNCTION CALL'
2282  D09F E7 E4              STB   ,S                  SAVE NUMBER OF FCBS ON STACK
2283  D0A1 9D A5              JSR   GETCCH              CHECK CURRENT INPUT CHAR
2284  D0A3 27 0B              BEQ   LD0B0               BRANCH IF END OF LINE
2285  D0A5 BD B2 6D         LD0A5   JSR   SYNCOMMA      SYNTAX CHECK FOR COMMA
2286  D0A8 BD B3 E6           JSR   LB3E6               EVALUATE EXPRESSION, RETURN VALUE IN ACCD
2287  D0AB C3 00 01           ADDD  #$0001              ADD ONE BYTE
2288  D0AE ED 61              STD   $01,S               SAVE RANDOM FILE BUFFER SIZE ON STACK
2289  D0B0 BD CA 3B         LD0B0   JSR   DVEC7         CLOSE FILES
2290  D0B3 E6 E4              LDB   ,S                  * GET THE NUMBER OF BUFFERS TO MAKE AND
2291  D0B5 34 04              PSHS  B                   * INITIALIZE A BUFFER COUNTER ON THE STACK
2292  D0B7 CC 09 89           LDD   #DFLBUF             GET START OF RANDOM FILE BUFFERS
2293  D0BA E3 62              ADDD  $02,S               ADD THE NEWLY SPECIFIED RANDOM FILE BUFFER SPACE
2294  D0BC 25 5D              BLO   LD11B               'OUT OF MEMORY' ERROR IF > $FFFF
2295  D0BE ED 62              STD   $02,S               SAVE START OF FCBS
2296                        * RESERVE SPACE FOR FCBS
2297  D0C0 C3 01 19         LD0C0   ADDD  #FCBLEN       FCBLEN REQUIRED FOR EACH BUFFER
2298  D0C3 25 56              BLO   LD11B               'OUT OF MEMORY' ERROR IF > $FFFF
2299  D0C5 6A E4              DEC   ,S                  DECREMENT BUFFER COUNTER
2300  D0C7 2A F7              BPL   LD0C0               *BRANCH IF NOT DONE - THE BPL WILL SET UP ONE MORE BUFFER
2301                        *                           *THAN THE NUMBER REQUESTED. THIS EXTRA BUFFER IS THE SYSTEM BUFFER
2302                        *                           *AND IS LOCATED AT THE END OF THE NORMAL FCBS. ONLY SYSTEM ROUTINES
2303                        *                           *(COPY, BACKUP, MERGE ETC.) MAY ACCESS THIS BUFFER.
2304  D0C9 5D                 TSTB                      AT AN EXACT 256 BYTE BOUNDARY?
```

```
2305   D0CA 27 03                BEQ   LD0CF                       YES
2306   D0CC 4C                   INCA                              NO - ADD 256
2307   D0CD 27 4C                BEQ   LD11B                       'OUT OF MEMORY' ERROR IF PAST $FFFF
2308   D0CF A7 E4          LD0CF STA   ,S                          SAVE MS BYTE OF NEW GRAPHIC RAM START
2309   D0D1 DC 1B                LDD   VARTAB                      GET START OF VARIABLES
2310   D0D3 90 BC                SUBA  GRPRAM                      *SUBTRACT THE OLD GRAPHIC RAM START - ACCD CONTAINS LENGTH
2311                       *                                       *OF PROGRAM PLUS RESERVED GRAPHIC RAM
2312   D0D5 AB E4                ADDA  ,S                          ADD IN THE AMOUNT OF RAM CALCULATED ABOVE
2313   D0D7 25 42                BLO   LD11B                       'OUT OF MEMORY' ERROR IF > $FFFF
2314   D0D9 1F 01                TFR   D,X                         SAVE NEW VARTAB IN X
2315   D0DB 4C                   INCA                              *ADD 256 - TO GUARANTEE ENOUGH ROOM SINCE ALL CALCULATIONS USE
2316                       *                                       *ONLY THE MSB OF THE ADDRESS
2317   D0DC 27 3D                BEQ   LD11B                       'OUT OF MEMORY' ERROR IF PAST $FFFF
2318   D0DE 10 93 21             CMPD  FRETOP                      IS IT GREATER THAN THE START OF STRING SPACE
2319   D0E1 24 38                BHS   LD11B                       'OUT OF MEMORY' IF > START OF STRING SPACE
2320   D0E3 4A                   DECA                              SUBTRACT 256 - COMPENSATE FOR INCA ABOVE
2321   D0E4 93 1B                SUBD  VARTAB                      SUBTRACT START OF VARIABLES
2322   D0E6 D3 19                ADDD  TXTTAB                      ADD START OF BASIC
2323   D0E8 1F 02                TFR   D,Y                         Y HAS NEW START OF BASIC
2324   D0EA A6 E4                LDA   ,S                          * GET THE GRAPHIC RAM START, SUBTRACT
2325   D0EC 90 BC                SUBA  GRPRAM                      * THE OLD GRAPHIC RAN START AND SAVE
2326   D0EE 1F 89                TFR   A,B                         * THE DIFFERENCE IN ACCA AND ACCB
2327   D0F0 9B BA                ADDA  BEGGRP                      = ADD THE OLD GRAPHIC PAGE START AND
2328   D0F2 97 BA                STA   BEGGRP                      = STORE THE NEW START OF GRAPHICS RAM
2329   D0F4 DB B7                ADDB  ENDGRP                      * ADD THE OLD GRAPHIC RAM END ADDRESS AND
2330   D0F6 D7 B7                STB   ENDGRP                      * STORE THE NEW END OF GRAPHICS RAM
2331   D0F8 35 46                PULS  A,B,U                       = ACCA=MSB OF START OF GRAPHIC RAM; ACCB=NUMBER OF FILE BUFFERS
2332                       *                                       = U=START OF FILE BUFFERS
2333   D0FA 97 BC                STA   GRPRAM                      SAVE NEW START OF GRAPHIC RAM
2334   D0FC F7 09 5B             STB   FCBACT                      NUMBER OF FILE BUFFERS
2335   D0FF FF 09 4A             STU   FCBADR                      START OF FILE BUFFERS
2336   D102 DE 1B          LD102 LDU   VARTAB                      POINT U TO OLD START OF VARIABLES
2337   D104 9F 1B                STX   VARTAB                      SAVE NEW START OF VARIBLES
2338   D106 11 93 1B             CMPU  VARTAB                      * COMPARE OLD START OF VARIABLES TO NEW START OF
2339   D109 22 13                BHI   LD11E                       * VARIABLES & BRANCH IF OLD > NEW
2340                       * MOVE BASIC PROGRAM IF OLD START ADDRESS <= NEW START ADDRESS
2341   D10B A6 C2          LD10B LDA   ,-U                         GET A BYTE
2342   D10D A7 82                STA   ,-X                         MOVE lT
2343   D10F 11 93 19             CMPU  TXTTAB                      AT START OF BASIC PROGRAM?
2344   D112 26 F7                BNE   LD10B                       NO
2345   D114 10 9F 19             STY   TXTTAB                      STORE NEW START OF BASIC PROGRAM
2346   D117 6F 3F                CLR   -1,Y                        RESET START OF PROGRAM FLAG
2347   D119 20 13                BRA   LD12E                       CLOSE ALL FILES
2348   D11B 7E AC 44       LD11B JMP   LAC44                       'OUT OF MEMORY' ERROR
2349                       * MOVE BASIC PROGRAM IF OLD START ADDRESS > NEW START ADDRESS
2350   D11E DE 19          LD11E LDU   TXTTAB                      POINT U TO OLD START OF BASIC
2351   D120 10 9F 19             STY   TXTTAB                      SAVE NEW START OF BASIC
2352   D123 6F 3F                CLR   -1,Y                        RESET START OF BASIC FLAG
2353   D125 A6 C0          LD125 LDA   ,U+                         GET A BYTE
2354   D127 A7 A0                STA   ,Y+                         MOVE IT
2355   D129 10 9C 1B             CMPY  VARTAB                      AT START OF VARIABLES
2356   D12C 26 F7                BNE   LD125                       NO - MOVE ANOTHER BYTE
2357
2358                       * CLOSE ALL FCBS AND RECALCULATE FCB START ADDRESSES
2359   D12E CE 09 28       LD12E LDU   #FCBV1                      POINT U TO FILE BUFFER POINTERS
2360   D131 BE 09 4A             LDX   FCBADR                      POINT X TO START OF BUFFERS
2361   D134 5F                   CLRB                              RESET FILE COUNTER
2362   D135 AF C1          LD135 STX   ,U++                        STORE FILE ADDRESS IN VECTOR TABLE
2363   D137 6F 00                CLR   FCBTYP,X                    RESET FILE TYPE TO CLOSED
2364   D139 30 89 01 19          LEAX  FCBLEN,X                    GO TO NEXT FCB
2365   D13D 5C                   INCB                              INCREMENT FILE COUNTER
2366   D13E F1 09 5B             CMPB  FCBACT                      CLOSE ALL ACTIVE BUFFERS AND SYSTEM FCB
2367   D141 23 F2                BLS   LD135                       BRANCH IF NOT DONE
2368   D143 7E 96 CB             JMP   L96CB                       READJUST LINE NUMBERS, ETC.
2369
2370                       * UNLOAD COMMAND
2371   D146 8D 1A          UNLOAD BSR  LD162                       GET DRIVE NUMBER
2372   D148 5F                   CLRB                              CLEAR FILE COUNTER
2373   D149 5C             LD149 INCB                              INCREMENT FILE COUNTER
2374   D14A BD C7 19             JSR   LC719                       POINT X TO FCB
2375   D14D 27 0D                BEQ   LD15C                       BRANCH IF FILE NOT OPEN
2376   D14F A6 01                LDA   FCBDRV,X                    CHECK DRIVE NUMBER
2377   D151 91 EB                CMPA  DCDRV                       DOES IT MATCH THE 'UNLOAD' DRIVE NUMBER?
2378   D153 26 07                BNE   LD15C                       NO MATCH - DO NOT CLOSE THE FILE
2379   D155 34 04                PSHS  B                           SAVE FILE COUNTER ON THE STACK
2380   D157 BD CA 58             JSR   LCA58                       CLOSE FCB
2381   D15A 35 04                PULS  B                           RESTORE FILE COUNTER
2382   D15C F1 09 5B       LD15C CMPB  FCBACT                      CHECKED ALL FILES?
2383   D15F 23 E8                BLS   LD149                       NO
2384   D161 39                   RTS
2385                       * GET DRIVE NUMBER FROM BASIC - USE THE DEFAULT DRIVE IF NONE GIVEN
2386   D162 F6 09 5A       LD162 LDB   DEFDRV                      GET DEFAULT DRIVE NUMBER
2387   D165 9D A5                JSR   GETCCH                      GET NEXT INPUT CHAR
2388   D167 27 09                BEQ   LD172                       USE DEFAULT DRIVE NUMBER IF NONE GIVEN
2389   D169 BD B7 0B       LD169 JSR   EVALEXPB                    EVALUATE EXPRESSION
2390   D16C C1 03                CMPB  #$03                        4 DRIVES MAX
2391   D16E 10 22 D4 AD          LBHI  LA61F                       'DEVICE NUMBER ERROR' IF > 3
2392   D172 D7 EB          LD172 STB   DCDRV                       STORE IN DSKCON VARIABLE
2393   D174 39                   RTS
2394
2395                       * BACKUP COMMAND
2396   D175 10 27 D4 A6    BACKUP LBEQ LA61F                       DEVICE NUMBER ERROR IF NO DRIVE NUMBERS GIVEN
2397   D179 BD 95 AC             JSR   L95AC                       RESET SAM DISPLAY PAGE AND VOG MODE
2398 > D17C BD D1 69             JSR   LD169                       * GET SOURCE DRIVE NUMBER AND SAVE
2399   D17F F7 06 FF             STB   DBUF0+255                   * IT AT TOP OF DBUF0 (TOP OF NEW STACK)
2400   D182 9D A5                JSR   GETCCH                      GET A CHARACTER FROM BASIC
```

```
2401   D184 27 08                BEQ   LD18E                       BRANCH IF END OF LINE
2402   D186 C6 A5                LDB   #$A5                        TOKEN FOR 'TO'
2403   D188 BD B2 6F             JSR   LB26F                       SYNTAX CHECK FOR 'TO'
2404 > D18B BD D1 69             JSR   LD169                       GET DESTINATION DRIVE NUMBER
2405   D18E 10 CE 06 FF  LD18E   LDS   #DBUF0+255                  PUT STACK AT TOP OF DBUF0
2406   D192 34 04                PSHS  B                           SAVE DESTINATION DRIVE NUMBER ON STACK
2407   D194 BD A5 C7             JSR   LA5C7                       SYNTAX ERROR IF NOT END OF LINE
2408   D197 BD CA 3B             JSR   DVEC7                       CLOSE ALL FILES
2409   D19A 6F E2                CLR   ,-S                         CLEAR A TRACK COUNTER ON STACK
2410   D19C 8E 09 88             LDX   #DFLBUF-1                   POINT X TO TOP OF DISK RAM VARIABLES
2411   D19F 6C E4        LD19F   INC   ,S                          INCREMENT TRACK COUNTER
2412   D1A1 30 89 12 00          LEAX  SECMAX*SECLEN,X             INCREMENT X BY ONE TRACK
2413   D1A5 9C 27                CMPX  MEMSIZ                      COMPARE TO TOP OF NON RESERVED RAN
2414   D1A7 23 F6                BLS   LD19F                       KEEP GOING IF MORE FREE RAM LEFT
2415   D1A9 6A E4                DEC   ,S                          DECREMENT TRACK COUNTER
2416   D1AB 10 27 DA 95          LBEQ  LAC44                       'OM' ERROR IF < 1 TRACK OF FREE RAM
2417   D1AF 86 23                LDA   #TRKMAX                     GET MAXIMUM NUMBER OF TRACKS   INITIALIZE REMAINING TRACKS CTR
2418   D1B1 5F                   CLRB                              INITIALIZE TRACKS WRITTEN COUNTER TO ZERO
2419   D1B2 34 06                PSHS  B,A                         SAVE TRACKS WRITTEN AND REMAINING COUNTERS ON STACK
2420
2421                     * AT THIS POINT THE STACK HAS THE FOLLOWING DATA ON IT:
2422                     * ,S = TRACKS REMAINING COUNTER; 1,S = TRACKS WRITTEN COUNTER
2423                     * 2,S = NUMBER OF TRACKS WHICH FIT IN RAM; 3,S = DESTINATION DRIVE NUMBER
2424                     * 4,S = SOURCE DRIVE NUMBER
2425   D1B4 73 09 5C             COM   DRESFL                      SET THE DISK RESET FLAG TO CAUSE A RESET
2426   D1B7 5F           LD1B7   CLRB                              INITIALIZE WRITE TRACK COUNTER TO ZERO
2427   D1B8 5C           LD1B8   INCB                              ADD ONE TO WRITE TRACK COUNTER
2428   D1B9 6A E4                DEC   ,S                          * DECREMENT REMAINING TRACKS COUNTER
2429   D1BB 27 04                BEQ   LD1C1                       * AND BRANCH IF NO TRACKS LEFT
2430   D1BD E1 62                CMPB  $02,S                       = COMPARE WRITE TRACK COUNTER TO NUMBER OF TRACKS THAT
2431   D1BF 26 F7                BNE   LD1B8                       = WILL FIT IN RAM AND BRANCH IF ROOM FOR MORE TRACKS IN RAM
2432   D1C1 D7 03        LD1C1   STB   TMPLOC                      SAVE THE NUMBER OF TRACKS TO BE TRANSFERRED
2433   D1C3 E6 64                LDB   $04,S                       GET SOURCE DRIVE NUMBER
2434   D1C5 8D 48                BSR   LD20F                       FILL RAM BUFFER WITH TMPLOC TRACKS OF DATA
2435   D1C7 86 FF                LDA   #$FF                        SET SOURCE/DESTINATION FLAG TO DESTINATION
2436 > D1C9 BD D2 35             JSR   LD235                       PRINT PROMPT MESSAGE IF NEEDED
2437   D1CC E6 63                LDB   $03,S                       GET DESTINATION DRIVE NUMBER
2438   D1CE 8D 42                BSR   LD212                       WRITE TMPLOC TRACKS FROM BUFFER
2439   D1D0 6D E4                TST   ,S                          TEST TRACKS REMAINING FLAG
2440   D1D2 27 0C                BEQ   LD1E0                       BRANCH IF BACKUP DONE
2441   D1D4 4F                   CLRA                              SET SOURCE/DESTINATION FLAG TO SOURCE
2442 > D1D5 BD D2 35             JSR   LD235                       PRINT PROMPT MESSAGE IF NEEDED
2443   D1D8 E6 61                LDB   $01,S                       * GET THE TRACKS WRITTEN COUNTER, ADD THE NUMBER OF
2444   D1DA DB 03                ADDB  TMPLOC                      * TRACKS MOVED THIS TIME THROUGH LOOP AND
2445   D1DC E7 61                STB   $01,S                       * SAVE THE NEW TRACKS WRITTEN COUNTER
2446   D1DE 20 D7                BRA   LD1B7                       COPY SOME MORE TRACKS
2447
2448   D1E0 8D 03        LD1E0   BSR   LD1E5                       CHECK FOR DOS INITIALIZATION
2449   D1E2 7E AC 73             JMP   LAC73                       JUMP BACK TO BASIC S MAIN LOOP
2450
2451   D1E5 35 40        LD1E5   PULS  U                           PUT THE RETURN ADDRESS IN U
2452   D1E7 B6 09 5C             LDA   DRESFL                      TEST DISK RESET FLAG
2453   D1EA 27 16                BEQ   LD202                       DON T RESET THE DOS IF FLAG NOT SET
2454   D1EC 8E 09 28             LDX   #FCBV1                      POINT X TO TABLE OF FCB ADDRESSES
2455   D1EF 4F                   CLRA                              SET FILE COUNTER TO ZERO
2456   D1F0 6F 91        LD1F0   CLR   [,X++]                      MARK FCB AS CLOSED
2457   D1F2 4C                   INCA                              ADD ONE TO FILE COUNTER
2458   D1F3 B1 09 5B             CMPA  FCBACT                      COMPARE TO NUMBER OF RESERVED FILES
2459   D1F6 23 F8                BLS   LD1F0                       BRANCH IF ANY FILES NOT SHUT DOWN
2460   D1F8 9E 19                LDX   TXTTAB                      LOAD X WITH THE START OF BASIC
2461   D1FA 6F 1F                CLR   -1,X                        SET FIRST BYTE OF BASIC PROGRAM TO ZERO
2462   D1FC BD AD 19             JSR   LAD19                       GO DO A 'NEW'
2463   D1FF 7F 09 5C             CLR   DRESFL                      RESET THE DOS RESET FLAG
2464   D202 B6 09 5D     LD202   LDA   DLODFL                      * CHECK THE LOAD RESET FLAG AND
2465   D205 27 06                BEQ   LD20D                       * BRANCH IF NOT SET
2466   D207 7F 09 5D             CLR   DLODFL                      CLEAR THE LOAD RESET FLAG
2467   D20A BD AD 19             JSR   LAD19                       GO DO A 'NEW'
2468   D20D 6E C4        LD20D   JMP   ,U                          JUMP BACK TO RETURN ADDRESS SAVED IN U ABOVE
2469
2470   D20F 86 02                LDA   #$02                        READ OP CODE
2471   D211 8C 86 03     LD211   CMPX  #$8603                      SKIP TWO BYTES
2472   D212 86 33        LD212   LDA   #$03                        WRITE OP CODE
2473   D214 DD EA                STD   DCOPC                       SAVE IN DSKCON VARIABLE
2474   D216 A6 63                LDA   $03,S                       * GET THE NUMBER OF THE TRACK BEING CURRENTLY
2475   D218 97 EC                STA   DCTRK                       * WRITTEN AND SAVE IT IN DSKCON VARIABLE
2476   D21A 8E 09 89             LDX   #DFLBUF                     = TRACK BUFFER STARTS AT DFLBUF
2477   D21D 9F EE                STX   DCBPT                       = SAVE IT IN DSKCON VARIABLE
2478   D21F 96 03                LDA   TMPLOC                      GET NUMBER OF TRACKS TO MOVE
2479   D221 C6 01        LD221   LDB   #$01                        INITIALIZE SECTOR COUNTER TO ONE
2480   D223 D7 ED        LD223   STB   DSEC                        SAVE DSKCON SECTOR VARIABLE
2481   D225 BD D5 FF             JSR   LD5FF                       READ/WRITE A SECTOR
2482   D228 0C EE                INC   DCBPT                       MOVE BUFFER POINTER UP ONE SECTOR (256 BYTES)
2483   D22A 5C                   INCB                              INCREMENT SECTOR COUNTER
2484   D22B C1 12                CMPB  #SECMAX                     COMPARE TO MAXIMUM NUMBER OF SECTORS PER TRACK
2485   D22D 23 F4                BLS   LD223                       BRANCH IF ANY SECTORS LEFT
2486   D22F 0C EC                INC   DCTRK                       INCREMENT TRACK COUNTER VARIABLE TO NEXT TRACK
2487   D231 4A                   DECA                              DECREMENT TRACKS TO MOVE COUNTER
2488   D232 26 ED                BNE   LD221                       READ MORE TRACKS IF ANY LEFT
2489   D234 39                   RTS
2490
2491   D235 E6 65        LD235   LDB   $05,S                       * GET THE DESTINATlON DRIVE NUMBER AND
2492   D237 E1 66                CMPB  $06,S                       * COMPARE IT TO THE SOURCE DRIVE NUMBER
2493
2494                     * PRINT SOURCE/DESTINATION DISK SWITCH PROMPT MESSAGE
2495   D239 26 36        LD239   BNE   LD271                       RETURN IF DRIVE NUMBERS NOT EQUAL
2496   D23B 7F 09 85             CLR   RDYTMR                      RESET THE READY TIMER
```

```
2497   D23E 7F FF 40                  CLR   DSKREG                          CLEAR DSKREG - TURN OFF ALL DISK MOTORS
2498   D241 7F 09 86                  CLR   DRGRAM                          CLEAR DSKREG RAM IMAGE
2499   D244 34 02                     PSHS  A                               SAVE SOURCE/DESTINATION FLAG ON STACK
2500   D246 BD A9 28                  JSR   LA928                           CLEAR SCREEN
2501   D249 8E D2 72                  LDX   #LD272                          POINT X TO 'INSERT SOURCE' MESSAGE
2502   D24C C6 0D                     LDB   #13                             13 BYTES IN MESSAGE
2503   D24E A6 E0                     LDA   ,S+                             GET SOURCE/DESTINATION FLAG FROM THE STACK
2504   D250 27 05                     BEQ   LD257                           BRANCH IF SOURCE
2505   D252 8E D2 7F                  LDX   #LD27F                          POINT X TO 'INSERT DESTINATION' MESSAGE
2506   D255 C6 12                     LDB   #18                             18 BYTES IN MESSAGE
2507   D257 BD B9 A2          LD257   JSR   LB9A2                           SEND MESSAGE TO CONSOLE OUT
2508   D25A 8E D2 91                  LDX   #LD291                          POINT X TO 'DISKETTE AND' MESSAGE
2509   D25D C6 1B                     LDB   #27                             27 BYTES IN MESSAGE
2510   D25F BD B9 A2                  JSR   LB9A2                           SEND MESSAGE TO CONSOLE OUT
2511   D262 CC 64 05                  LDD   #$6405                          * SET UP 'SOUND' PARAMETERS
2512   D265 97 8C                     STA   SNDTON                          * FOR A BEEP
2513   D267 BD A9 51                  JSR   LA951                           JUMP TO 'SOUND' - DO A BEEP
2514   D26A BD A1 71          LD26A   JSR   LA171                           GET A CHARACTER FROM CONSOLE IN
2515   D26D 81 0D                     CMPA  #CR                             * KEEP LOOKING AT CONSOLE IN UNTIL
2516   D26F 26 F9                     BNE   LD26A                           * YOU GET A CARRIAGE RETURN
2517   D271 39                LD271   RTS
2518
2519   D272 49 4E 53 45 52 54 LD272   FCC   'INSERT SOURCE'
2520   D278 20 53 4F 55 52 43
2521   D27E 45
2522   D27F 49 4E 53 45 52 54 LD27F   FCC   'INSERT DESTINATION'
2523   D285 20 44 45 53 54 49
2524   D28B 4E 41 54 49 4F 4E
2525   D291 20 44 49 53 4B 45 LD291   FCC   ' DISKETTE AND'
2526   D297 54 54 45 20 41 4E
2527   D29D 44
2528   D29E 0D                        FCB   CR
2529   D29F 50 52 45 53 53 20         FCC   'PRESS 'ENTER''
2530   D2A5 27 45 4E 54 45 52
2531   D2AB 27
2532
2533                          * PUSH FILENAME.EXT AND DRIVE NUMBER ONTO THE STACK
2534   D2AC 35 20             LD2AC   PULS  Y                               SAVE RETURN ADDRESS IN Y
2535   D2AE C6 0B                     LDB   #11                             11 CHARACTERS IN FILENAME AND EXTENSION
2536   D2B0 8E 09 57                  LDX   #DNAMBF+11                      POINT X TO TOP OF DISK NAME/EXT BUFFER
2537   D2B3 A6 82             LD2B3   LDA   ,-X                             * GET A CHARACTER FROM FILENAME.
2538   D2B5 34 02                     PSHS  A                               * EXT BUFFER AND PUSH IT ONTO THE
2539   D2B7 5A                        DECB                                  * STACK - DECREMENT COUNTER AND
2540   D2B8 26 F9                     BNE   LD2B3                           * KEEP LOOPING UNTIL DONE
2541   D2BA 96 EB                     LDA   DCDRV                           = GET DRIVE NUMBER AND PUSH
2542   D2BC 34 02                     PSHS  A                               = IT ONTO THE STACK
2543   D2BE 6E A4                     JMP   ,Y                              PSEUDO - RETURN TO CALLING ROUTINE
2544
2545                          * PULL FILENAME.EXT AND DRIVE NUMBER FROM (X) TO RAM
2546   D2C0 A6 80             LD2C0   LDA   ,X+                             * GET DRIVE NUMBER AND SAVE
2547   D2C2 97 EB                     STA   DCDRV                           * IT IN DSKCON VARIABLE
2548   D2C4 C6 0B                     LDB   #11                             11 BYTES IN FILENAME AND EXTENSION
2549   D2C6 CE 09 4C                  LDU   #DNAMBF                         POINT U TO DISK NAME BUFFER
2550   D2C9 7E A5 9A                  JMP   LA59A                           MOVE FILENANE.EXT FROM (X) TO DNAMBF
2551
2552                          * COPY
2553                          * THE COPY PROCESS IS PERFORMED BY COPYING DATA FROM THE SOURCE FILE
2554                          * TO RAM AND THEN COPYING IT TO THE DESTINATION FILE. THE SOURCE AND
2555                          * DESTINATION FILES ARE OPENED AS RANDOM FILES AND BOTH USE THE SYSTEM
2556                          * FCB ABOVE THE RESERVED FCBS. ALL OF AVAILABLE FREE RAM ABOVE THE
2557                          * VARIABLES IS USED AS A COPY BUFFER WHICH SPEEDS UP THE COPYING PROCESS
2558                          * BUT UNFORTUNATELY THE METHOD USED WILL ALLOW AN ERROR ENCOUNTERED DURING
2559                          * THE COPY PROCESS TO 'HANG' THE SYSTEM. THIS IS CAUSED BY POINTING THE FCB'S
2560                          * RANDOM FILE BUFFER POINTER (FCBBUF,X) TO THE FREE RAM BUFFER. AN ERROR
2561                          * WILL THEN CAUSE THE OPEN FILE TO BE CLOSED WITH FCBBUF,X POINTING TO AN
2562                          * AREA IN RAM WHERE THE RANDOM FILE BUFFER CLOSE ROUTINE (LCAE2) WILL NEVER
2563                          * LOOK FOR IT
2564   D2CC BD C8 87          COPY    JSR   LC887                           * GET SOURCE FILENAME.EXT & DRIVE NUMBER FROM BASIC
2565   D2CF 8D DB                     BSR   LD2AC                           * AND SAVE THEM ON THE STACK
2566   D2D1 6F E2                     CLR   ,-S                             CLEAR A BYTE ON STACK - SINGLE DISK COPY (SDC) FLAG
2567   D2D3 9D A5                     JSR   GETCCH                          GET CURRENT INPUT CHARACTER
2568   D2D5 27 0A                     BEQ   LD2E1                           BRANCH IF END OF LINE - SINGLE DISK COPY
2569   D2D7 63 E4                     COM   ,S                              SET SOC FLAG TO $FF (NO SINGLE DISK COPY)
2570   D2D9 C6 A5                     LDB   #$A5                            TOKEN FOR 'TO'
2571   D2DB BD B2 6F                  JSR   LB26F                           SYNTAX CHECK FOR 'TO'
2572   D2DE BD C8 87                  JSR   LC887                           GET DESTINATION FILENAME.EXT AND DRIVE NUMBER
2573   D2E1 8D C9             LD2E1   BSR   LD2AC                           SAVE DESTINATION FILENAME.EXT & DRIVE NUMBER ON STACK
2574   D2E3 BD A5 C7                  JSR   LA5C7                           SYNTAX ERROR IF MORE CHARACTERS ON LINE
2575   D2E6 BD CA 3B                  JSR   DVEC7                           CLOSE ALL FILES
2576
2577                          * COUNT THE NUMBER OF SECTORS WORTH OF FREE RAM AVAILABLE
2578   D2E9 6F E2                     CLR   ,-S                             CLEAR A SECTOR COUNTER ON THE STACK
2579   D2EB 30 E8 9C                  LEAX  -100,S                          ** BUG ** THIS SHOULD BE $100 TO POINT X ONE SECTOR LENGTH BELOW STACK
2580   D2EE 6C E4             LD2EE   INC   ,S                              INCREMENT SECTOR COUNTER
2581   D2F0 30 89 FF 00               LEAX  -SECLEN,X                       DECREMENT X BY ONE SECTOR
2582   D2F4 9C 1F                     CMPX  ARYEND                          COMPARE TO TOP OF ARRAYS
2583   D2F6 24 F6                     BHS   LD2EE                           BRANCH IF NOT AT BOTTOM OF FREE RAM
2584   D2F8 6A E4                     DEC   ,S                              DECREMENT SECTOR COUNTER
2585   D2FA 10 27 D9 46               LBEQ  LAC44                           'OM' ERROR IF NOT AT LEAST ONE FULL SECTOR OF FREE RAM
2586   D2FE 30 6E                     LEAX  14,S                            POINT X TO START OF SOURCE DATA
2587   D300 8D BE                     BSR   LD2C0                           PUT SOURCE DATA INTO DNAMBF AND DSKCON
2588   D302 BD C6 5F                  JSR   LC65F                           SCAN DIRECTORY FOR A MATCH
2589   D305 BD C6 B8                  JSR   LC6B8                           'NE' ERROR IF MATCH NOT FOUND
2590   D308 BE 09 74                  LDX   V974                            POINT X TO DIRECTORY RAM IMAGE OF FOUND FILE
2591   D30B EE 0E                     LDU   DIRLST,X                        * GET NUMBER OF BYTES IN LAST SECTOR AND
2592   D30D AE 0B                     LDX   DIRTYP,X                        * SOURCE FILE TYPE AND ASCII FLAG
```

```
2593   D30F 34 50                 PSHS  U,X                         * AND SAVE THEM ON THE STACK
2594   D311 BD C7 6D              JSR   LC76D                       GET VALID FAT DATA
2595   D314 F6 09 76              LDB   V976                        GET NUMBER OF FIRST GRANULE IN FILE
2596   D317 BD CC 44              JSR   LCC44                       * GET THE NUMBER OF GRANULES IN FILE
2597   D31A 34 02                 PSHS  A                           * AND SAVE IT ON THE STACK
2598   D31C 4A                    DECA                              SUBTRACT OFF THE LAST GRANULE
2599   D31D C4 3F                 ANDB  #$3F                        * MASK OFF LAST GRANULE FLAG BITS AND SAVE THE
2600   D31F 34 04                 PSHS  B                           * NUMBER OF SECTORS IN LAST GRANULE ON STACK
2601   D321 1F 89                 TFR   A,B                         SAVE THE NUMBER OF GRANULES IN ACCB
2602   D323 4F                    CLRA                              CLEAR THE MS BYTE OF ACCD
2603   D324 BD C7 49              JSR   LC749                       MULTIPLY ACCD BY NINE
2604   D327 EB E4                 ADDB  ,S                          * ADD THE NUMBER OF SECTORS IN THE LAST
2605   D329 89 00                 ADCA  #$00                        * GRANULE TO ACCD
2606   D32B 8E 00 01              LDX   #$0001                      INITIALIZE RECORD COUNTER TO ONE
2607   D32E 34 16                 PSHS  X,B,A                       INITIALIZE SECTOR AND RECORD COUNTERS ON THE STACK
2608
2609                         * AT THIS POINT THE CONTROL VARIABLES FOR COPY ARE STORED ON THE STACK.
2610                         *   0 1,S = REMAINING SECTORS COUNTER; 2 3,S = RECORD COUNTER
2611                         *     4,S = NUMBER OF SECTORS TO BE COPIED. INITIALLY SET TO NUMBER OF
2612                         *           SECTORS IN THE LAST GRANULE.
2613                         *     5,S = GRAN TEST FLAG. INITIALLY SET TO NUMBER OF GRANS IN FILE
2614                         *     6,S = FILE TYPE; 7,S = ASCII FLAG; 8 9,S = NUMBER OF BYTES IN LAST SECTOR
2615                         *    10,S = NUMBER OF SECTORS WHICH WILL FIT IN THE CURRENTLY AVAILABLE FREE RAM
2616                         * 11-22,S = DESTINATION FILENAME.EXT AND DRIVE NUMBER
2617                         *    23,S = SINGLE DISK COPY FLAG; 24-35,S = SOURCE FILENAME.EXT AND DRIVE NUMBER
2618   D330 5F              LD330    CLRB                              SET SECTOR COUNTER TO ZERO
2619   D331 AE E4                    LDX   ,S                          GET THE NUMBER OF SECTORS REMAINING IN THE FILE
2620   D333 27 09                    BEQ   LD33E                       BRANCH IF NO SECTORS LEFT
2621   D335 5C              LD335    INCB                              ADD A SECTOR TO TEMPORARY SECTOR COUNTER
2622   D336 30 1F                    LEAX  -1,X                        DECREMENT REMAINING SECTORS COUNTER
2623   D338 27 04                    BEQ   LD33E                       BRANCH IF NO SECTORS LEFT
2624   D33A E1 6A                    CMPB  10,S                        *COMPARE TEMPORARY COUNTER TO NUMBER OF SECTORS WHICH MAY
2625                         *                                         *BE STORED IN FREE RAM
2626   D33C 26 F7                    BNE   LD335                       BRANCH IF STILL ROOM FOR MORE SECTORS
2627   D33E AF E4           LD33E    STX   ,S                          SAVE THE NUMBER OF UNCOPIED SECTORS REMAINING IN THE FILE
2628   D340 E7 64                    STB   $04,S                       SAVE THE NUMBER OF SECTORS TO BE COPIED THIS TIME THROUGH LOOP
2629   D342 8D 50                    BSR   $D394                       'GET' ACCB SECTORS TO RAM BUFFER
2630   D344 86 FF                    LDA   #$FF                        SET SOURCE/DESTINATION FLAG TO DESTINATION
2631   D346 8D 40                    BSR   LD388                       PRINT PROMPT MESSAGE IF REQUIRED
2632   D348 6D 65                    TST   $05,S                       * CHECK THE GRAN TEST FLAG. IF <> 0, IT CONTAINS THE
2633   D34A 27 25                    BEQ   LD371                       * NUMBER OF GRANS IN THE FILE AND THE DESTINATION DISK
2634                         *                                         * MUST BE CHECKED FOR ENOUGH ROOM. IF IT IS = 0
2635                         *                                         * THEN THE CHECK HAS ALREADY BEEN DONE
2636   D34C 30 6B                    LEAX  11,S                        POINT TO DESTINATION FILE PARAMETERS
2637   D34E BD D2 C0                 JSR   LD2C0                       GET DESTINATION FILE PARAMETERS FROM STACK
2638   D351 BD CF 7D                 JSR   LCF7D                       SCAN DIRECTORY FOR FILE - 'AE' ERROR IF IT EXISTS
2639   D354 BD C7 6D                 JSR   LC76D                       GET VALID FAT DATA
2640
2641                         * MAKE SURE THERE ARE ENOUGH FREE GRANULES ON THE DESTINATION DISK
2642   D357 BD C7 25                 JSR   LC725                       POINT X TO FAT
2643   D35A 30 06                    LEAX  FATCON,X                    SKIP PAST THE FAT CONTROL BYTES
2644   D35C A6 65                    LDA   $05,S                       GET THE NUMBER OF GRANS IN THE FILE
2645   D35E C6 44                    LDB   #GRANMX                     SET GRAN COUNTER TO MAXIMUM
2646   D360 63 84           LD360    COM   ,X                          * CHECK TO SEE IF A BRAN IS FREE
2647   D362 26 03                    BNE   LD367                       * AND BRANCH IF IT IS NOT FREE
2648   D364 4A                       DECA                              = DECREMENT COUNTER AND BRANCH IF
2649   D365 27 08                    BEQ   LD36F                       = THERE ARE ENOUGH FREE GRANULES
2650   D367 63 80           LD367    COM   ,X+                         RESTORE FAT BYTE AND INCREMENT POINTER
2651   D369 5A                       DECB                              DECREMENT GRAN COUNTER
2652   D36A 26 F4                    BNE   LD360                       BRANCH IF ALL GRANS NOT CHECKED
2653   D36C 7E C7 C8                 JMP   LC7C8                       'DISK FULL' ERROR
2654   D36F 63 84           LD36F    COM   ,X                          RESTORE FAT BYTE
2655   D371 8D 1B           LD371    BSR   LD38E                       'PUT' DATA FROM RAM BUFFER TO DESTINATION FILE
2656   D373 AE E4                    LDX   ,S                          GET THE NUMBER OF REMAINING SECTORS
2657   D375 27 0D                    BEQ   LD384                       EXIT ROUTINE IF NO SECTORS LEFT
2658   D377 EC 62                    LDD   $02,S                       *
2659   D379 EB 64                    ADDB  $04,S                       * GET THE CURRENT RECORD COUNTER, ADD
2660   D37B 89 00                    ADCA  #$00                        * THE NUMBER OF SECTORS (RECORDS) MOVED
2661   D37D ED 62                    STD   $02,S                       * AND SAVE THE NEW RECORD COUNTER
2662   D37F 4F                       CLRA                              SET SOURCE/DESTINATION FLAG TO SOURCE
2663   D380 8D 06                    BSR   LD388                       PRINT PROMPT MESSAGE IF REQUIRED
2664   D382 20 AC                    BRA   LD330                       KEEP COPYING SECTORS
2665
2666   D384 32 E8 24        LD384    LEAS  36,S                        REMOVE TEMPORARY STORAGE VARIABLES FROM STACK
2667   D387 39                       RTS                               **** COPY DONE ****
2668
2669   D388 6D E8 19        LD388    TST   25,S                        *CHECK SINGLE DISK COPY FLAG - IF <> ZERO, THEN DON'T
2670                         *                                         *PRINT THE PROMPT MESSAGE
2671   D38B 7E D2 39                 JMP   LD239                       PRINT THE PROMPT MESSAGE IF REQUIRED
2672   D38E 86 FF           LD38E    LDA   #$FF
2673
2674                         * 'PUT'.'GET' DATA FROM THE DESTINATION/SOURCE I'PUT' FLAG
2675   D390 30 6D                    LEAX  13,S                        POINT X TO DESTINATION FILENAME DATA
2676   D392 20 04                    BRA   LD398                       GO 'PUT' SOME DATA
2677   D394 4F              LD394    CLRA                              ZERO IS THE 'GET' FLAG
2678   D395 30 E8 1A                 LEAX  26,S                        POINT X TO THE SOURCE FILENAME DATA
2679   D398 97 D8           LD398    STA   VD8                         SAVE THE 'GET'/'PUT' FLAG
2680   D39A BD D2 C0                 JSR   LD2C0                       GET FILENAME AND DRIVE DATA FROM THE STACK
2681   D39D AE 68                    LDX   8,S                         * GET ASCII FLAG AND FILE TYPE AND SAVE
2682   D39F BF 09 57                 STX   DFLTYP                      * THEM IN THE DISK RAM VARIABLES
2683   D3A2 8E 01 00                 LDX   #SECLEN                     = SAVE ONE SECTOR LENGTH IN
2684   D3A5 BF 09 7C                 STX   DFFLEN                      = RAM RECORD LENGTH VARIABLE
2685   D3A8 86 52                    LDA   #'R'                        RANDOM FILE TYPE FLAG
2686   D3AA F6 09 5B                 LDB   FCBACT                      * GET THE HIGHEST RESERVED FCB NUMBER, ADD ONE
2687   D3AD 5C                       INCB                              * AND OPEN A RANDOM FILE WHOSE FCB WILL BE ONE ABOVE
2688   D3AE BD C4 68                 JSR   LC468                       * THE HIGHEST RESERVED FCB (THE SYSTEM FCB)
```

```
2689   D3B1 9E F1                 LDX   FCBTMP                    POINT X TO THE 'SYSTEM' FCB
2690   D3B3 CC 01 00              LDD   #SECLEN                   * SET THE NUMBER OF BYTES IN THE LAST SECTOR
2691   D3B6 ED 88 13              STD   FCBLST,X                  * OF THE FILE EQUAL TO ONE SECTOR LENGTH
2692   D3B9 E6 66                 LDB   $06,S                     =GET THE NUMBER OF SECTORS TO MOVE AND
2693   D3BB 27 29                 BEQ   LD3E6                     =BRANCH IF NONE LEFT
2694   D3BD D6 D8                 LDB   VD8                       *GRAB THE 'GET'/'PUT' FLAG, 'AND' IT WITH THE
2695   D3BF E4 67                 ANDB  $07,S                     *GRAN TEST FLAG - BRANCH IF 'GET'ING DATA OR THIS IS
2696   D3C1 27 09                 BEQ   LD3CC                     *NOT THE FIRST TIME THROUGH THE LOOP
2697   D3C3 EC 62                 LDD   $02,S                     =GET THE NUMBER OF SECTORS REMAINING TO BE COPIED AND
2698   D3C5 EB 66                 ADDB  $06,S                     =ADD THE NUMBER TO BE COPIED THIS TIME THROUGH LOOP
2699   D3C7 89 00                 ADCA  #$00                      =
2700   D3C9 BD C2 CC              JSR   LC2CC                     *'PUT' THE LAST RECORD IN THE FILE TO THE SYSTEM FCB.
2701   D3CC 9E F1          LD3CC  LDX   FCBTMP                    *THE RECORD NUMBER IS IN ACCD.
2702                       *                                      POINT X TO THE SYSTEM FCB
2703   D3CE EE 64                 LDU   $04,S                     * GET THE CURRENT RECORD NUMBER
2704   D3D0 EF 07                 STU   FCBREC,X                  * AND SAVE IT IN THE FCB
2705   D3D2 E6 66                 LDB   $06,S                     GET THE NUMBER OF THE RECORD (SECTOR) TO MOVE
2706   D3D4 DE 1F                 LDU   ARYEND                    END OF ARRAYS IS THE START OF THE COPY FREE RAM BUFFER
2707   D3D6 34 44          LD3D6  PSHS  U,B                       SAVE SECTOR COUNTER AND BUFFER POINTER ON THE STACK
2708   D3D8 9E F1                 LDX   FCBTMP                    POINT X TO SYSTEM FCB
2709   D3DA EF 0B                 STU   FCBBUF,X                  *SET THE RANDOM FILE BUFFER POINTER TO THE 'COPY' RAM BUFFER
2710   D3DC BD C2 D0              JSR   LC2D0                     *THIS WILL CAUSE THE SYSTEM TO 'HANG' IF AN ERROR OCCURS DURING COPY.
2711                       *                                      GO 'GET' OR 'PUT' DATA TO THE SYSTEM FCB
2712   D3DF 6C 61                 INC   $01,S                     ADD 256 (ONE SECTOR) TO THE BUFFER POINTER
2713   D3E1 35 44                 PULS  B,U                       GET THE SECTOR COUNTER AND BUFFER POINER
2714   D3E3 5A                    DECB                            DECREMENT SECTOR COUNTER
2715   D3E4 26 F0                 BNE   LD3D6                     BRANCH IF ALL SECTORS NOT DONE
2716   D3E6 9E F1          LD3E6  LDX   FCBTMP                    POINT X TO SYSTEM FCB
2717   D3E8 CE 09 89              LDU   #DFLBUF                   * RESET THE RANDOM FILE BUFFER POINTER FOR THE SYSTEM
2718   D3EB EF 0B                 STU   FCBBUF,X                  * FCB TO THE BOTTOM OF RANDOM FILE BUFFER AREA
2719   D3ED D6 D8                 LDB   VD8                       =GRAB THE 'GET'/'PUT' FLAG, 'AND' IT WITH THE GRAN
2720   D3EF E4 67                 ANDB  $07,S                     =TEST FLAG - CLOSE THE FILE IF 'GET'ING DATA AND
2721   D3F1 27 09                 BEQ   LD3FC                     =THIS IS NOT THE FIRST TIME THROUGH THE LOOP
2722   D3F3 6F 67                 CLR   $07,S                     RESET THE GRAN TEST FLAG IF FIRST TIME THROUGH LOOP
2723   D3F5 EC 6A                 LDD   10,S                      *GET THE NUMBER OF BYTES IN THE LAST SECTOR,
2724   D3F7 8A 80                 ORA   #$80                      *'OR' IN THE PRE-SAVED FLAG AND
2725   D3F9 ED 88 13              STD   FCBLST,X                  *SAVE THE NUMBER OF BYTES IN THE LAST SECTOR IN THE FCB
2726   D3FC 7E CA 58       LD3FC  JMP   LCA58                     CLOSE THE FILE
2727
2728                       * DSKI$ COMMAND
2729   D3FF 8D 38          DSKI   BSR   LD439                     GET THE DRIVE, TRACK AND SECTOR NUMBERS
2730   D401 8D 2B                 BSR   LD42E                     * EVALUATE STRING VARIABLE 1 AND SAVE
2731   D403 34 10                 PSHS  X                         * THE DESCRIPTOR ADDRESS ON THE STACK
2732   D405 8D 27                 BSR   LD42E                     = EVALUATE STRING VARIABLE 2 AND SAVE
2733   D407 34 10                 PSHS  X                         = THE DESCRiPTOR ADDRESS ON THE STACK
2734   D409 C6 02                 LDB   #$02                      DSKCON READ OP CODE
2735   D40B BD D4 A1              JSR   LD4A1                     REAO A SECTOR INTO DBUF0
2736   D40E CE 06 80              LDU   #DBUF0+128                POINT U TO TOP HALF OF DBUF0
2737   D411 35 10                 PULS  X                         GET STRING 2 DESCRIPTOR ADDRESS
2738   D413 8D 05                 BSR   LD41A                     PUT STRING 2 INTO STRING SPACE
2739   D415 CE 06 00              LDU   #DBUF0                    POINT U TO BOTTOM HALF OF DBUF0
2740   D418 35 10                 PULS  X                         GET STRING 1 DESCRIPTOR ADDRESS
2741   D41A 34 50          LD41A  PSHS  U,X                       PUT STRING DESCRIPTOR & SOURCE POINTER ON THE STACK
2742   D41C C6 80                 LDB   #128                      *
2743   D41E BD B5 0F              JSR   LB50F                     * RESERVE 128 BYTES IN STRING SPACE
2744   D421 33 84                 LEAU  ,X                        POINT U TO RESERVED STRING SPACE
2745   D423 35 10                 PULS  X                         GET STRING DESCRIPTOR ADDRESS
2746   D425 E7 84                 STB   ,X                        * SAVE DESCRIPTOR DATA (LENGTH AND ADDRESS)
2747   D427 EF 02                 STU   $02,X                     * OF THE NEW STRING
2748   D429 35 10                 PULS  X                         GET THE SOURCE (DBUF0) POINTER
2749   D42B 7E A5 9A       LD42B  JMP   LA59A                     MOVE SECTOR DATA FROM DBUF0 TO STRING SPACE
2750
2751   D42E BD B2 6D       LD42E  JSR   SYNCOMMA                  SYNTAX CHECK FOR A COMMA
2752   D431 8E B3 57              LDX   #LB357                    POINT X TO EVALUATE VARIABLE ROUTINE
2753   D434 8D 2F                 BSR   LD465                     EVALUATE A VARIABLE
2754   D436 7E B1 46       LD436  JMP   LB146                     'TM' ERROR IF NUMERIC VARIABLE
2755
2756                       * EVALUATE DRIVE, TRACK AND SECTOR NUMBERS
2757   D439 BD B7 0B       LD439  JSR   EVALEXPB                  EVALUATE EXPRESSION, RETURN VALUE IN ACCB
2758   D43C C1 03                 CMPB  #$03                      * COMPARE TO 3 (HIGHEST DRIVE NUMBER) -
2759   D43E 22 1C                 BHI   LD45C                     * 'FC' ERROR IF IT S > 3
2760   D440 34 04                 PSHS  B                         SAVE DRIVE NUMBER ON THE STACK
2761   D442 BD B7 38              JSR   LB738                     SYNTAX CHECK FOR COMMA. EVALUATE EXPRESSION (TRACK NUMBER)
2762   D445 C1 22                 CMPB  #TRKMAX-1                 * CHECK FOR MAXIMUM TRACK NUMBER
2763   D447 22 13                 BHI   LD45C                     * 'FC' ERROR IF TRACK NUMBER > 34
2764   D449 34 04                 PSHS  B                         SAVE TRACK NUMBER ON THE STACK
2765   D44B BD B7 38              JSR   LB738                     SYNTAX CHECK FOR COMMA, EVALUATE EXPRESSION (SECTOR NUMBER)
2766   D44E D7 ED                 STB   DSEC                      SAVE SECTOR NUMBER IN DSKCON VARIABLE
2767   D450 5A                    DECB                            *USELESS INSTRUCTION. NEXT INSTRUCTION SHOULD JUST
2768   D451 C1 11                 CMPB  #SECMAX-1                 *CHECK FOR MAXIMUM SECTOR NUMBER (SECMAX)
2769   D453 22 07                 BHI   LD45C                     'FC' ERROR IF SECTOR NUMBER TOO BIG
2770   D455 35 06                 PULS  A,B                       * GET TRACK AND DRIVE NUMBER OFF OF
2771   D457 97 EC                 STA   DCTRK                     * THE STACK AND SAVE IN DSKCON
2772   D459 D7 EB                 STB   DCDRV                     * VARIABLES
2773   D45B 39                    RTS
2774   D45C 7E B4 4A       LD45C  JMP   LB44A                     JUMP TO 'FC' ERROR
2775
2776   D45F BD B2 6D       LD45F  JSR   SYNCOMMA                  SYNTAX CHECK FOR COMMA
2777   D462 8E B1 56              LDX   #LB156                    POINT X TO 'EVALUATE EXPRESSION' ROUTINE ADDRESS
2778   D465 D6 EB          LD465  LDB   DCDRV                     * GET THE DSKCON DRIVE, TRACK AND
2779   D467 DE EC                 LDU   DCTRK                     * SECTOR VALUES AND SAVE THEM ON THE STACK
2780   D469 34 44                 PSHS  U,B                       *
2781   D46B AD 84                 JSR   ,X                        GO EVALUATE AN EXPRESSION OR A VARIABLE
2782   D46D 35 44                 PULS  B,U                       * GET THE DRIVE, TRACK AND SECTOR
2783   D46F D7 EB                 STB   DCDRV                     * NUMBERS OFF OF THE STACK AND PUT
2784   D471 DF EC                 STU   DCTRK                     * THEM BACK INTO THE DSKCON VARIABLES
```

```
2785   D473 39                    RTS
2786
2787                       * DSKO$ COMMAND
2788   D474 8D C3          DSKO    BSR   LD439                GET THE DRIVE, TRACK AND SECTOR NUMBERS
2789   D476 8D E7                  BSR   LD45F                GET THE DESCRIPTOR OF STRING 1
2790   D478 8D BC                  BSR   LD436                'TM' ERROR IF NUMERIC EXPRESSION
2791   D47A 9E 52                  LDX   FPA0+2               * GET STRING 1 DESCRIPTOR ADDRESS
2792   D47C 34 10                  PSHS  X                    * AND SAVE IT ON THE STACK
2793   D47E 8D DF                  BSR   LD45F                GET THE DESCRIPTOR OF STRING 2
2794   D480 BD B6 54               JSR   LB654                *GET LENGTH AND ADDRESS OF STRING 2 AND
2795   D483 34 14                  PSHS  X,B                  *SAVE THEM ON THE STACK
2796   D485 5F                     CLRB                       SET CLEAR COUNTER TO 256 (FULL SECTOR BUFFER)
2797   D486 8E 06 00               LDX   #DBUF0               USE DBUF0 AS THE DSKO$ I/O BUFFER
2798   D489 6F 80          LD489   CLR   ,X+                  CLEAR A BYTE IN I/O BUFFER
2799   D48B 5A                     DECB                       DECREMENT CLEAR COUNTER
2800   D48C 26 FB                  BNE   LD489                BRANCH IF ALL 256 BYTES NOT CLEARED
2801   D48E 35 14                  PULS  B,X                  GET THE LENGTH AND ADDRESS OF STRING 2
2802   D490 CE 06 80               LDU   #DBUF0+128           POINT X TO STRING 2 DESTINATION
2803   D493 8D 96                  BSR   LD42B                MOVE STRING 2 DATA INTO DBUF0
2804   D495 35 10                  PULS  X                    POINT X TO STRING 1 DESCRIPTOR
2805   D497 BD B6 59               JSR   LB659                GET THE LENGTH AND ADDRESS OF STRING 1
2806   D49A CE 06 00               LDU   #DBUF0               POINT U TO STRING 1 DESTINATION
2807   D49D 8D 8C                  BSR   LD42B                MOVE STRING 1 DATA INTO DBUF0
2808   D49F C6 03                  LDB   #$03                 DSKCON WRITE OP CODE
2809   D4A1 8E 06 00       LD4A1   LDX   #DBUF0               POINT X TO I/O BUFFER (DBUF0)
2810   D4A4 9F EE                  STX   DCBPT                *
2811   D4A6 D7 EA                  STB   DCOPC                * SAVE NEW DSKCON BUFFER POINTER AND OP CODE VARIABLES
2812   D4A8 7E D5 FF               JMP   LD5FF                GO WRITE OUT A SECTOR
2813
2814                       * DSKINI COMMAND
2815   D4AB 10 27 D1 70    DSKINI  LBEQ  LA61F                BRANCH TO 'DN' ERROR IF NO DRIVE NUMBER SPECIFIED
2816   D4AF BD D1 69               JSR   LD169                CALCULATE DRIVE NUMBER
2817   D4B2 C6 04                  LDB   #$04                 SKIP FACTOR DEFAULT VALUE
2818   D4B4 9D A5                  JSR   GETCCH               GET CURRENT INPUT CHAR FROM BASIC
2819   D4B6 27 0C                  BEQ   LD4C4                BRANCH IF END OF LINE
2820   D4B8 BD B7 38               JSR   LB738                SYNTAX CHECK FOR COMMA AND EVALUATE EXPRESSION
2821   D4BB C1 11                  CMPB  #17                  MAX VALUE OF SKIP FACTOR = 16
2822   D4BD 10 24 DF 89            LBHS  LB44A                'ILLEGAL FUNCTION CALL' IF BAD SKIP FACTOR
2823   D4C1 BD A5 C7               JSR   LA5C7                SYNTAX ERROR IF MORE CHARACTERS ON THE LINE
2824   D4C4 34 04          LD4C4   PSHS  B                    SAVE SKIP FACTOR ON THE STACK
2825   D4C6 8E 07 12               LDX   #DBUF1+SECMAX        POINT TO END OF LOGICAL SECTOR NUMBER STORAGE AREA
2826   D4C9 C6 12                  LDB   #SECMAX              18 SECTORS PER TRACK
2827   D4CB 6F 82          LD4CB   CLR   ,-X                  CLEAR A BYTE IN THE BUFFER
2828   D4CD 5A                     DECB                       CLEARED ALL 18?
2829   D4CE 26 FB                  BNE   LD4CB                KEEP GOING IF NOT
2830   D4D0 4F                     CLRA                       RESET PHYSICAL SECTOR COUNTER
2831   D4D1 20 0D                  BRA   LD4E0                START WITH FIRST PHYSICAL SECTOR = 1
2832
2833                       * CALCULATE LOGICAL SECTOR NUMBERS
2834   D4D3 EB E4          LD4D3   ADDB  ,S                   ADD SKIP FACTOR TO LOGICAL SECTOR COUNTER
2835   D4D5 5C             LD4D5   INCB                       ADD ONE TO LOGICAL SECTOR COUNTER
2836   D4D6 C0 12          LD4D6   SUBB  #SECMAX              SUBTRACT MAX NUMBER OF SECTORS
2837   D4D8 24 FC                  BHS   LD4D6                BRANCH UNTIL 0 > ACCB >= -18
2838   D4DA CB 12                  ADDB  #SECMAX              ADD 18, NOW ACCB IS 0-17
2839   D4DC 6D 85                  TST   B,X                  IS ANYTHING STORED HERE ALREADY?
2840   D4DE 26 F5                  BNE   LD4D5                YES - GET ANOTHER SECTOR
2841   D4E0 4C             LD4E0   INCA                       * INCREMENT PHYSICAL SECTOR NUMBER AND
2842   D4E1 A7 85                  STA   B,X                  * SAVE IT IN THE RAM BUFFER
2843   D4E3 81 12                  CMPA  #SECMAX              FINISHED WITH ALL SECTORS?
2844   D4E5 25 EC                  BLO   LD4D3                NO - KEEP GOING
2845   D4E7 32 61                  LEAS  $01,S                REMOVE SKIP FACTOR FROM STACK
2846   D4E9 8E 22 0F               LDX   #DFLBUF+$1888-2      GET TOP OF RAM USED BY DSKINI
2847   D4EC 9C 27                  CMPX  MEMSIZ               IS IT > CLEARED AREA?
2848   D4EE 10 22 D7 52            LBHI  LAC44                'OUT OF MEMORY' ERROR IF > CLEARED AREA
2849   D4F2 BD CA 3B               JSR   DVEC7                CLOSE ALL FILES
2850   D4F5 73 09 5C               COM   DRESFL               SET RESET FLAG TO $FF - THIS WILL CAUSE A DOS RESET
2851   D4F8 10 CE 08 00            LDS   #DBUF1+SECLEN        SET STACK TO TOP OF DBUF1
2852   D4FC BD 95 AC               JSR   L95AC                RESET SAM TO DISPLAY PAGE ZERO AND ALPHA GRAPHICS
2853   D4FF 86 00                  LDA   #$00                 YOU COULD DELETE THIS INSTRUCTION AND CHANGE FOLLOWING STA TO CLR
2854   D501 97 EA                  STA   DCOPC                RESTORE HEAD TO TRACK ZERO DSKCON OP CODE
2855   D503 BD D5 FF               JSR   LD5FF                RESTORE HEAD TO TRACK ZERO
2856   D506 7F 09 85               CLR   RDYTMR               RESET THE READY TIMER
2857   D509 86 C0                  LDA   #$C0                 * FOC READ ADDRESS CODE
2858   D50B B7 FF 48               STA   FDCREG               *
2859   D50E BD D6 DE               JSR   LD6DE                CHECK DRIVE READY - WAIT UNTIL READY
2860   D511 10 26 00 86            LBNE  LD59B                BRANCH IF NOT READY - ISSUE AN ERROR
2861   D515 0F EC                  CLR   DCTRK                RESET TRACK NUMBER
2862   D517 20 1A                  BRA   LD533                START THE FORMATTING PROCESS
2863   D519 81 16          LD519   CMPA  #22                   = CHECK FOR TRACK 22 (PRECOMPENSATION)
2864   D51B 25 08                  BLO   LD525                = AND BRANCH IF < TRACK 22 - NO PRECOMP
2865   D51D B6 09 86               LDA   DRGRAM               * GET THE RAM IMAGE OF DSKREG, 'OR'
2866   D520 8A 10                  ORA   #$10                 * IN THE PRECOMPENSATION FLAG AND
2867   D522 B7 FF 40               STA   DSKREG               * SEND IT TO DSKREG
2868   D525 86 53          LD525   LDA   #$53                  = GET STEP IN COMMAND
2869   D527 B7 FF 48               STA   FDCREG               = AND SEND IT TO THE 1793
2870   D52A 1E 88                  EXG   A,A                  * DELAY AFTER ISSUING COMMAND TO 1793
2871   D52C 1E 88                  EXG   A,A                  *
2872   D52E BD D6 DE               JSR   LD6DE                CHECK DRIVE READY
2873   D531 26 68                  BNE   LD59B                BRANCH IF NOT READY - ISSUE AN ERROR
2874   D533 BD D6 FD       LD533   JSR   LD6FD                WAIT A WHILE
2875   D536 8D 6C                  BSR   $D5A4                BUILD A FORMATTED TRACK IN RAM
2876   D538 10 8E FF 4B            LDY   #FDCREG+3            Y POINTS TO 1793 DATA REGISTER
2877   D53C 1A 50                  ORCC  #$50                 DISABLE INTERRUPTS
2878   D53E 8E D5 62               LDX   #LD562               * GET RETURN ADDRESS AND STORE
2879   D541 BF 09 83               STX   DNMISV               * IT IN THE NON MASKABLE INTERRUPT VECTOR
2880   D544 8E 09 89               LDX   #DFLBUF              POINT X TO THE FORMATTED TRACK RAM IMAGE
```

```
2881   D547 B6 FF 48            LDA   FDCREG                    RESET STATUS OF THE 1793
2882   D54A 86 FF               LDA   #$FF                      * ENABLE THE NMI FLAG TO VECTOR
2883   D54C B7 09 82            STA   NMIFLG                    * OUT OF AN I/O LOOP UPON AN NMI INTERRUPT
2884   D54F C6 F4               LDB   #$F4                      = GET WRITE TRACK COMMAND AND
2885   D551 F7 FF 48            STB   FDCREG                    = SEND TO 1793
2886   D554 B6 09 86            LDA   DRGRAM                    * GET THE DSKREG RAM IMAGE AND 'OR' IN THE
2887   D557 8A 80               ORA   #$80                      * FLAG WHICH WILL ENABLE THE 1793 TO HALT
2888   D559 B7 FF 40            STA   DSKREG                    * THE 6809. SEND RESULT TO DSKREG
2889   D55C E6 80        LD55C  LDB   ,X+                       = GET A BYTE FROM THE FORMATTED TRACK
2890   D55E E7 A4               STB   ,Y                        = RAM IMAGE, SEND IT TO THE 1793 AND
2891   D560 20 FA               BRA   LD55C                     = LOOP BACK TO GET ANOTHER BYTE
2892
2893   D562 B6 FF 48     LD562  LDA   FDCREG                    GET STATUS
2894   D565 1C AF               ANDCC #$AF                      ENABLE INTERRUPTS
2895   D567 84 44               ANDA  #$44                      * KEEP ONLY WRITE PROTECT & LOST DATA
2896   D569 97 F0               STA   DCSTA                     * AND SAVE IT IN THE DSKCON STATUS BYTE
2897   D56B 26 2E               BNE   LD59B                     BRANCH IF ERROR
2898   D56D 0C EC               INC   DCTRK                     SKIP TO THE NEXT TRACK
2899   D56F 96 EC               LDA   DCTRK                     GET THE TRACK NUMBER
2900   D571 81 23               CMPA  #TRKMAX                   WAS IT THE LAST TRACK
2901   D573 26 A4               BNE   LD519                     NO - KEEP GOING
2902
2903                     * VERIFY THAT ALL SECTORS ARE READABLE
2904   D575 86 02               LDA   #$02                      = GET THE DSKCON READ OP CODE
2905   D577 97 EA               STA   DCOPC                     = AND SAVE IT IN THE DSKCON VARIABLE
2906   D579 8E 06 00            LDX   #DBUF0                    * POINT THE DSKCON BUFFER POINTER
2907   D57C 9F EE               STX   DCBPT                     * TO DBUF0
2908   D57E CE 07 00            LDU   #DBUF1                    POINT U TO THE LOGICAL SECTOR NUMBERS
2909   D581 4F                  CLRA                            RESET THE TRACK COUNTER TO ZERO
2910   D582 97 EC        LD582  STA   DCTRK                     SET THE DSKCON TRACK VARIABLE
2911   D584 5F                  CLRB                            RESET THE SECTOR COUNTER
2912   D585 A6 C5        LD585  LDA   B,U                       GET THE PHYSICAL SECTOR NUMBER
2913   D587 97 ED               STA   DSEC                      SAVE DSKCON SECTOR VARIABLE
2914 > D589 BD D5 FF            JSR   LD5FF                     READ A SECTOR
2915   D58C 5C                  INCB                            * INCREMENT THE SECTOR COUNTER
2916   D58D C1 12               CMPB  #SECMAX                   * AND COMPARE IT TO MAXIMUM SECTOR NUMBER
2917   D58F 25 F4               BLO   LD585                     * AND KEEP LOOPING IF MORE SECTORS LEFT
2918   D591 96 EC               LDA   DCTRK                     = GET THE CURRENT TRACK NUMBER
2919   D593 4C                  INCA                            = ADD ONE TO IT, COMPARE TO THE MAXIMUM TRACK
2920   D594 81 23               CMPA  #TRKMAX                   = NUMBER AND KEEP LOOPING IF
2921   D596 25 EA               BLO   LD582                     = THERE ARE STILL TRACKS TO DO
2922   D598 7E D1 E0            JMP   LD1E0                     GO CHECK FOR A DOS RESET
2923   D59B 7F 09 86     LD59B  CLR   DRGRAM                    CLEAR RAM IMAGE OF DSKREG
2924   D59E 7F FF 40            CLR   DSKREG                    CLEAR DSKREG - TURN DISK MOTORS OFF
2925 > D5A1 7E D6 0E            JMP   LD60E                     PROCESS DRIVES NOT READY ERROR
2926
2927                     * BUILD A FORMATTED TRACK OF DATA IN RAM STARTING AT DFLBUF.
2928
2929   D5A4 8E 09 89            LDX   #DFLBUF                   START TRACK BUFFER AT DFLBUF
2930   D5A7 CC 20 4E            LDD   #$204E                    GET SET TO WRITE 32 BYTES OF $4E
2931   D5AA 8D 29               BSR   LD5D5                     GO WRITE GAP IV
2932   D5AC 5F                  CLRB                            RESET SECTOR COUNTER
2933   D5AD 34 04        LD5AD  PSHS  B                         SAVE SECTOR COUNTER
2934   D5AF CE 07 00            LDU   #DBUF1                    POINT U TO THE TABLE OF LOGICAL SECTORS
2935   D5B2 E6 C5               LDB   B,U                       * GET LOGICAL SECTOR NUMBER FROM TABLE AND
2936   D5B4 D7 ED               STB   DSEC                      * SAVE IT IN THE DSKCON VARIABLE
2937   D5B6 CE D5 E7            LDU   #LD5E7                    POINT U TO TABLE OF SECTOR FORMATTING DATA
2938   D5B9 C6 03               LDB   #$03                      * GET FIRST 3 DATA BLOCKS AND
2939   D5BB 8D 1E               BSR   LD5DB                     * WRITE THEM TO BUFFER
2940   D5BD 96 EC               LDA   DCTRK                     = GET TRACK NUMBER AND STORE 1T
2941   D5BF A7 80               STA   ,X+                       = IN THE RAM BUFFER
2942   D5C1 6F 80               CLR   ,X+                       CLEAR A BYTE (SIDE NUMBER) IN BUFFER
2943   D5C3 96 ED               LDA   DSEC                      * GET SECTOR NUMBER AND
2944   D5C5 A7 80               STA   ,X+                       * STORE IT IN THE BUFFER
2945   D5C7 C6 09               LDB   #$09                      = GET THE LAST NINE DATA BLOCKS AND
2946   D5C9 8D 10               BSR   LD5DB                     = WRITE THEM TO THE BUFFER
2947   D5CB 35 04               PULS  B                         GET SECTOR COUNTER
2948   D5CD 5C                  INCB                            NEXT SECTOR
2949   D5CE C1 12               CMPB  #SECMAX                   18 SECTORS PER TRACK
2950   D5D0 25 DB               BLO   LD5AD                     BRANCH IF ALL SECTORS NOT DONE
2951   D5D2 CC C8 4E            LDD   #$C84E                    WRITE 200 BYTES OF $4E AT END OF TRACK
2952
2953                     * WRITE ACCA BYTES OF ACCB INTO BUFFER
2954   D5D5 E7 80        LD5D5  STB   ,X+                       STORE A BYTE IN THE BUFFER
2955   D5D7 4A                  DECA                            DECREMENT COUNTER
2956   D5D8 26 FB               BNE   LD5D5                     BRANCH IF ALL BYTES NOT MOVED
2957   D5DA 39                  RTS
2958   D5DB 34 04        LD5DB  PSHS  B                         SAVE THE COUNTER ON THE STACK
2959   D5DD EC C1               LDD   ,U++                      GET TWO BYTES OF DATA FROM THE TABLE
2960   D5DF 8D F4               BSR   LD5D5                     WRITE ACCA BYTES OF ACCB INTO THE BUFFER
2961   D5E1 35 04               PULS  B                         * GET THE COUNTER BACK, DECREMENT
2962   D5E3 5A                  DECB                            * IT AND BRANCH IF ALL DATA BLOCKS
2963   D5E4 26 F5               BNE   LD5DB                     * NOT DONE
2964   D5E6 39                  RTS
2965
2966                     * DATA USED TO FORMAT A SECTOR ON THE DISK
2967
2968                     * THESE DATA ARE CLOSE TO THE IBM SYSTEM 34 FORMAT FOR 256 BYTE SECTORS.
2969                     * DOUBLE DENSITY. THE FORMAT GENERALLY CONFORMS TO THAT SPECIFIED ON THE
2970                     * 1793 DATA SHEET. THE GAP SIZES HAVE BEEN REDUCED TO THE MINIMUM
2971                     * ALLOWABLE. THE IBM FORMAT USES $40 AS THE FILL CHARACTER FOR THE DATA
2972                     * BLOCKS WHILE COLOR DOS USES AN $FF AS THE FILL CHARACTER.
2973   D5E7 08 00        LD6D4  FCB   8,0                       SYNC FIELD
2974   D5E9 03 F5               FCB   3,$F5
2975   D5EB 01 FE               FCB   1,$FE                    ID ADDRESS MARK (AM1)
2976                     * TRACK, SIDE, AND SECTOR NUMBERS ARE INSERTED HERE
```

```
2977   D5ED 01 01                  FCB   1,1                       SECTOR SIZE (256 BYTE SECTORS)
2978   D5EF 01 F7                  FCB   1,$F7                     CRC REQUEST
2979   D5F1 16 4E                  FCB   22,$4E                    GAP II (POST-ID GAP)
2980   D5F3 0C 00                  FCB   12,0                      SYNC FIELD
2981   D5F5 03 F5                  FCB   3,$F5
2982   D5F7 01 FB                  FCB   1,$FB                     DATA ADDRESS MARK (AM2)
2983   D5F9 00 FF                  FCB   0,$FF                     DATA FIELD (256 BYTES)
2984   D5FB 01 F7                  FCB   1,$F7                     CRC REQUEST
2985   D5FD 18 4E                  FCB   24,$4E                    GAP III (POST DATA GAP)
2986
2987
2988   D5FF 34 04          LD5FF   PSHS  B                         SAVE ACCB
2989   D601 C6 05                  LDB   #$05                      5 RETRIES
2990   D603 F7 09 88               STB   ATTCTR                    SAVE RETRY COUNT
2991   D606 35 04                  PULS  B                         RESTORE ACCB
2992   D608 8D 62          LD608   BSR   DSKCON                    GO EXECUTE COMMAND
2993   D60A 0D F0                  TST   DCSTA                     CHECK STATUS
2994   D60C 27 0D                  BEQ   LD61B                     BRANCH IF NO ERRORS
2995   D60E 96 F0          LD60E   LDA   DCSTA                     GET DSKCON ERROR STATUS
2996   D610 C6 3C                  LDB   #2*30                     'WRITE PROTECTED' ERROR
2997   D612 85 40                  BITA  #$40                      CHECK BIT 6 OF STATUS
2998   D614 26 02                  BNE   LD618                     BRANCH IF WRITE PROTECT ERROR
2999   D616 C6 28          LD616   LDB   #2*20                     'I/O ERROR'
3000   D618 7E AC 46       LD618   JMP   LAC46                     JUMP TO ERROR DRIVER
3001   D61B 34 02          LD61B   PSHS  A                         SAVE ACCA
3002   D61D 96 EA                  LDA   DCOPC                     GET OPERATION CODE
3003   D61F 81 03                  CMPA  #$03                      CHECK FOR WRITE SECTOR COMMAND
3004   D621 35 02                  PULS  A                         RESTORE ACCA
3005   D623 26 2A                  BNE   LD64F                     RETURN IF NOT WRITE SECTOR
3006   D625 7D 09 87               TST   DVERVL                    CHECK VERIFY FLAG
3007   D628 27 25                  BEQ   LD64F                     RETURN IF NO VERIFY
3008   D62A 34 56                  PSHS  U,X,B,A                   SAVE REGISTERS
3009   D62C 86 02                  LDA   #$02                      READ OPERATION CODE
3010   D62E 97 EA                  STA   DCOPC                     STORE TO DSKCON PARAMETER
3011   D630 DE EE                  LDU   DCBPT                     POINT U TO WRITE BUFFER ADDRESS
3012   D632 8E 07 00               LDX   #DBUF1                    * ADDRESS OF VERIFY BUFFER
3013   D635 9F EE                  STX   DCBPT                     * TO DSKCON VARIABLE
3014   D637 8D 33                  BSR   DSKCON                    GO READ SECTOR
3015   D639 DF EE                  STU   DCBPT                     RESTORE WRITE BUFFER
3016   D63B 86 03                  LDA   #$03                      WRITE OP CODE
3017   D63D 97 EA                  STA   DCOPC                     SAVE IN DSKCON VARIABLE
3018   D63F 96 F0                  LDA   DCSTA                     CHECK STATUS FOR THE READ OPERATION
3019   D641 26 0D                  BNE   LD650                     BRANCH IF ERROR
3020   D643 5F                     CLRB                            CHECK 256 BYTES
3021   D644 A6 80          LD644   LDA   ,X+                       GET BYTE FROM WRITE BUFFER
3022   D646 A1 C0                  CMPA  ,U+                       COMPARE TO READ BUFFER
3023   D648 26 06                  BNE   LD650                     BRANCH IF NOT EQUAL
3024   D64A 5A                     DECB                            * DECREMENT BYTE COUNTER AND
3025   D64B 26 F7                  BNE   LD644                     * BRANCH IF NOT DONE
3026   D64D 35 56                  PULS  A,B,X,U                   RESTORE REGISTERS
3027   D64F 39             LD64F   RTS
3028   D650 35 56          LD650   PULS  A,B,X,U                   RESTORE REGISTERS
3029   D652 7A 09 88               DEC   ATTCTR                    DECREMENT THE VERIFY COUNTER
3030   D655 26 B1                  BNE   LD608                     BRANCH IF MORE TRIES LEFT
3031   D657 C6 48                  LDB   #2*336                    'VERIFY ERROR'
3032   D659 20 BD                  BRA   LD618                     JUMP TO ERROR HANDLER
3033
3034                       * VERIFY COMMAND
3035   D65B 5F             VERIFY  CLRB                            OFF FLAG = 0
3036   D65C 81 AA                  CMPA  #$AA                      OFF TOKEN ?
3037   D65E 27 07                  BEQ   LD667                     YES
3038   D660 53                     COMB                            ON FLAG = $FF
3039   D661 81 88                  CMPA  #$88                      ON TOKEN
3040   D663 10 26 DC 10            LBNE  LB277                     BRANCH TO 'SYNTAX ERROR' IF NOT ON OR OFF
3041   D667 F7 09 87       LD667   STB   DVERVL                    SET VERIFY FLAG
3042   D66A 0E 9F                  JMP   GETNCH                    GET NEXT CHARACTER FROM BASIC
3043
3044                       * DSKCON ROUTINE
3045   D66C 34 76          DSKCON  PSHS  U,Y,X,B,A                 SAVE REGISTERS
3046   D66E 86 05                  LDA   #$05                      * GET RETRY COUNT AND
3047   D670 34 02                  PSHS  A                         * SAVE IT ON THE STACK
3048   D672 7F 09 85       LD672   CLR   RDYTMR                    RESET DRIVE NOT READY TIMER
3049   D675 D6 EB                  LDB   DCDRV                     GET DRIVE NUMBER
3050   D677 8E D7 AA               LDX   #LD7AA                    POINT X TO DRIVE ENABLE MASKS
3051   D67A B6 09 86               LDA   DRGRAM                    GET DSKREG IMAGE
3052   D67D 84 A8                  ANDA  #$A8                      KEEP MOTOR STATUS, DOUBLE DENSITY. HALT ENABLE
3053   D67F AA 85                  ORA   B,X                       'OR' IN DRIVE SELECT DATA
3054   D681 8A 20                  ORA   #$20                      'OR' IN DOUBLE DENSITY
3055   D683 D6 EC                  LDB   DCTRK                     GET TRACK NUMBER
3056   D685 C1 16                  CMPB  #22                       PRECOMPENSATION STARTS AT TRACK 22
3057   D687 25 02                  BLO   LD68B                     BRANCH IF LESS THAN 22
3058   D689 8A 10                  ORA   #$10                      TURN ON WRITE PRECOMPENSATION IF >= 22
3059   D68B 1F 89          LD68B   TFR   A,B                       SAVE PARTIAL IMAGE IN ACCB
3060   D68D 8A 08                  ORA   #$08                      'OR' IN MOTOR ON CONTROL BIT
3061   D68F B7 09 86               STA   DRGRAM                    SAVE IMAGE IN RAM
3062   D692 B7 FF 40               STA   DSKREG                    PROGRAM THE 1793 CONTROL REGISTER
3063   D695 C5 08                  BITB  #$08                      = WERE MOTORS ALREADY ON?
3064   D697 26 06                  BNE   LD69F                     = DON'T WAIT FOR IT TO COME UP TO SPEED IF ALREADY ON
3065   D699 BD A7 D1               JSR   LA7D1                     * WAIT A WHILE
3066   D69C BD A7 D1               JSR   LA7D1                     * WAIT SOME MORE FOR MOTOR TO COME UP TO SPEED
3067   D69F 8D 3D          LD69F   BSR   LD6DE                     WAIT UNTIL NOT BUSY OR TIME OUT
3068   D6A1 26 0A                  BNE   LD6AD                     BRANCH IF TIMED OUT (DOOR OPEN. NO DISK, NO POWER. ETC.)
3069   D6A3 0F F0                  CLR   DCSTA                     CLEAR STATUS REGISTER
3070   D6A5 8E D7 A2               LDX   #LD7A2                    POINT TO COMMAND JUMP VECTORS
3071   D6A8 D6 EA                  LDB   DCOPC                     GET COMMAND
3072   D6AA 58                     ASLB                            2 BYTES PER COMMAND JUMP ADDRESS
```

```
3073   D6AB AD 95              JSR   [B,X]                          GO DO IT
3074   D6AD 35 02        LD6AD PULS  A                              GET RETRY COUNT
3075   D6AF D6 F0              LDB   DCSTA                          GET STATUS
3076   D6B1 27 0B              BEQ   LD6BE                          BRANCH IF NO ERRORS
3077   D6B3 4A                 DECA                                 DECREMENT RETRIES COUNTER
3078   D6B4 27 08              BEQ   LD6BE                          BRANCH IF NO RETRIES LEFT
3079   D6B6 34 02              PSHS  A                              SAVE RETRY COUNT ON STACK
3080   D6B8 8D 0B              BSR   LD6C5                          RESTORE HEAD TO TRACK 0
3081   D6BA 26 F1              BNE   LD6AD                          BRANCH IF SEEK ERROR
3082   D6BC 20 B4              BRA   LD672                          GO TRY COMMAND AGAIN IF NO ERROR
3083   D6BE 86 78        LD6BE LDA   #120                           120*1/60 = 2 SECONDS (1/60 SECOND FOR EACH IRQ INTERRUPT)
3084   D6C0 B7 09 85           STA   RDYTMR                         WAIT 2 SECONDS BEFORE TURNING OFF MOTOR
3085   D6C3 35 F6              PULS  A,B,X,Y,U,PC                   RESTORE REGISTERS - EXIT DSKCON
3086                     * RESTORE HEAD TO TRACK 0
3087   D6C5 8E 09 7E     LD6C5 LDX   #DR0TRK                        POINT TO TRACK TABLE
3088   D6C8 D6 EB              LDB   DCDRV                          GET DRIVE NUMBER
3089   D6CA 6F 85              CLR   B,X                            ZERO TRACK NUMBER
3090   D6CC 86 03              LDA   #$03                           * RESTORE HEAD TO TRACK 0, UNLOAD THE HEAD
3091   D6CE B7 FF 48           STA   FDCREG                         * AT START, 30 MS STEPPING RATE
3092   D6D1 1E 88              EXG   A,A                            =
3093   D6D3 1E 88              EXG   A,A                            = WAIT FOR 1793 TO RESPOND TO COMMAND
3094   D6D5 8D 07              BSR   LD6DE                          WAIT TILL DRIVE NOT BUSY
3095   D6D7 8D 24              BSR   LD6FD                          WAIT SOME MORE
3096   D6D9 84 10              ANDA  #$10                           1793 STATUS : KEEP ONLY SEEK ERROR
3097   D6DB 97 F0              STA   DCSTA                          SAVE IN DSKCON STATUS
3098   D6DD 39           LD6DD RTS
3099                     * WAIT FOR THE 1793 TO BECOME UNBUSY. IF IT DOES NOT BECOME UNBUSY,
3100                     * FORCE AN INTERRUPT AND ISSUE A  DRIVE NOT READY  1793 ERROR.
3101   D6DE 9E 8A        LD6DE LDX   ZERO                           GET ZERO TO X REGISTER - LONG WAIT
3102   D6E0 30 1F        LD6E0 LEAX  -1,X                           DECREMENT LONG WAIT COUNTER
3103   D6E2 27 08              BEQ   LD6EC                          IF NOT READY BY NOW, FORCE INTERRUPT
3104   D6E4 B6 FF 48           LDA   FDCREG                         * GET 1793 STATUS AND TEST
3105   D6E7 85 01              BITA  #$01                           * BUSY STATUS BIT
3106   D6E9 26 F5              BNE   LD6E0                          BRANCH IF BUSY
3107   D6EB 39                 RTS
3108   D6EC 86 D0        LD6EC LDA   #$D0                           * FORCE INTERRUPT COMMAND - TERMINATE ANY COMMAND
3109   D6EE B7 FF 48           STA   FDCREG                         * IN PROCESS. DO NOT GENERATE A 1793 INTERRUPT REQUEST
3110   D6F1 1E 88              EXG   A,A                            * WAIT BEFORE READING 1793
3111   D6F3 1E 88              EXG   A,A                            *
3112   D6F5 B6 FF 48           LDA   FDCREG                         RESET INTRQ (FDC INTERRUPT REQUEST)
3113   D6F8 86 80              LDA   #$80                           RETURN DRIVE NOT READY STATUS IF THE DRIVE DID NOT BECOME UNBUSY
3114   D6FA 97 F0              STA   DCSTA                          SAVE DSKCON STATUS BYTE
3115   D6FC 39                 RTS
3116                     * MEDIUM DELAY
3117   D6FD 8E 22 2E     LD6FD LDX   #8750                          DELAY FOR A WHILE
3118   D700 30 1F        LD700 LEAX  -1,X                           * DECREMENT DELAY COUNTER AND
3119   D702 26 FC              BNE   LD700                          * BRANCH IF NOT DONE
3120   D704 39                 RTS
3121                     * READ ONE SECTOR
3122   D705 86 80        LD705 LDA   #$80                           $80 IS READ FLAG (1793 READ SECTOR)
3123   D707 8C           LD707 CMPX  #$86A0                         SKIP TWO BYTES
3124                     * WRITE ONE SECTOR
3125   D708 86 A0        LD708 LDA   #$A0                           $A0 IS WRITE FLAG (1793 WRITE SECTOR)
3126   D70A 34 02              PSHS  A                              SAVE READ/WRITE FLAG ON STACK
3127   D70C 8E 09 7E           LDX   #DR0TRK                        POINT X TO TRACK NUMBER TABLE IN RAM
3128   D70F D6 EB              LDB   DCDRV                          GET DRIVE NUMBER
3129   D711 3A                 ABX                                  POINT X TO CORRECT DRIVE'S TRACK BYTE
3130   D712 E6 84              LDB   ,X                             GET TRACK NUMBER OF CURRENT HEAD POSITION
3131   D714 F7 FF 49           STB   FDCREG+1                       SEND TO 1793 TRACK REGISTER
3132   D717 D1 EC              CMPB  DCTRK                          COMPARE TO DESIRED TRACK
3133   D719 27 1E              BEQ   LD739                          BRANCH IF ON CORRECT TRACK
3134   D71B 96 EC              LDA   DCTRK                          GET TRACK DESIRED
3135   D71D B7 FF 4B           STA   FDCREG+3                       SEND TO 1793 DATA REGiSTER
3136   D720 A7 84              STA   ,X                             SAVE IN RAM TRACK IMAGE
3137   D722 86 17              LDA   #$17                           * SEEK COMMAND FOR 1793: DO NOT LOAD THE
3138   D724 B7 FF 48           STA   FDCREG                         * HEAD AT START, VERIFY DESTINATION TRACK,
3139   D727 1E 88              EXG   A,A                            * 30 MS STEPPING RATE - WAIT FOR
3140   D729 1E 88              EXG   A,A                            * VALID STATUS FROM 1793
3141   D72B 8D B1              BSR   LD6DE                          WAIT TILL NOT BUSY
3142   D72D 26 08              BNE   LD737                          RETURN IF TIMED OUT
3143   D72F 8D CC              BSR   LD6FD                          WAIT SOME MORE
3144   D731 84 18              ANDA  #$18                           KEEP ONLY SEEK ERROR OR CRC ERROR IN ID FIELD
3145   D733 27 04              BEQ   LD739                          BRANCH IF NO ERRORS - HEAD ON CORRECT TRACK
3146   D735 97 F0              STA   DCSTA                          SAVE IN DSKCON STATUS
3147   D737 35 82        LD737 PULS  A,PC
3148                     * HEAD POSITIONED ON CORRECT TRACK
3149   D739 96 ED        LD739 LDA   DSEC                           GET SECTOR NUMBER DESIRED
3150   D73B B7 FF 4A           STA   FDCREG+2                       SEND TO 1793 SECTOR REGISTER
3151   D73E 8E D7 98           LDX   #LD798                         * POINT X TO ROUTINE TO BE VECTORED
3152   D741 BF 09 83           STX   DNMISV                         * TO BY NMI UPON COMPLETION OF DISK I/O AND SAVE VECTOR
3153   D744 9E EE              LDX   DCBPT                          POINT X TO I/O BUFFER
3154   D746 B6 FF 48           LDA   FDCREG                         RESET INTRQ (FDC INTERRUPT REQUEST)
3155   D749 B6 09 86           LDA   DRGRAM                         GET DSKREG IMAGE
3156   D74C 8A 80              ORA   #$80                           SET FLAG TO ENABLE 1793 TO HALT 6809
3157   D74E 35 04              PULS  B                              GET READ/WRITE COMMAND FROM STACK
3158   D750 10 9E 8A           LDY   ZERO                           ZERO OUT Y - TIMEOUT INITIAL VALUE
3159   D753 CE FF 48           LDU   #FDCREG                        U POINTS TO 1793 INTERFACE REGISTERS
3160   D756 73 09 82           COM   NMIFLG                         NMI FLAG = $FF: ENABLE NMI VECTOR
3161   D759 1A 50              ORCC  #$50                           DISABLE FIRQ,IRQ
3162   D75B F7 FF 48           STB   FDCREG                         * SEND READ/WRITE COMMAND TO 1793: SINGLE RECORD, COMPARE
3163   D75E 1E 88              EXG   A,A                            * FOR SIDE 0, NO 15 MS DELAY, DISABLE SIDE SELECT
3164   D760 1E 88              EXG   A,A                            * COMPARE, WRITE DATA ADDRESS MARK (FB) - WAIT FOR STATUS
3165   D762 C1 80              CMPB  #$80                           WAS THIS A READ?
3166   D764 27 1C              BEQ   LD782                          IF SO, GO LOOK FOR DATA
3167                     * WAIT FOR THE 1793 TO ACKNOWLEDGE READY TO WRITE DATA
3168   D766 C6 02              LDB   #$02                           DRQ MASK BIT
```

```
3169   D768 E5 C4           LD768   BITB  ,U                          IS 1793 READY FOR A BYTE? (DRQ SET IN STATUS BYTE)
3170   D76A 26 0C                   BNE   LD778                       BRANCH IF SO
3171   D76C 31 3F                   LEAY  -1,Y                        DECREMENT WAIT TIMER
3172   D76E 26 F8                   BNE   LD768                       KEEP WAITING FOR THE 1793 DRQ
3173   D770 7F 09 82        LD770   CLR   NMIFLG                      RESET NMI FLAG
3174   D773 1C AF                   ANDCC #$AF                        ENABLE FIRQ,IRQ
3175   D775 7E D6 EC                JMP   LD6EC                       FORCE INTERRUPT, SET DRIVE NOT READY ERROR
3176
3177                        * WRITE A SECTOR
3178   D778 E6 80           LD778   LDB   ,X+                         GET A BYTE FROM RAM
3179   D77A F7 FF 4B                STB   FDCREG+3                    SEND IT TO 1793 DATA REGISTER
3180   D77D B7 FF 40                STA   DSKREG                      REPROGRAM FDC CONTROL REGISTER
3181   D780 20 F6                   BRA   LD778                       SEND MORE DATA
3182                        * WAIT FOR THE 17933 TO ACKNOWLEDGE READY TO READ DATA
3183   D782 C6 02           LD782   LDB   #$02                        DRQ MASK BIT
3184   D784 E5 C4           LD784   BITB  ,U                          DOES THE 1793 HAVE A BYTE? (DRQ SET IN STATUS BYTE)
3185   D786 26 06                   BNE   LD78E                       YES, GO READ A SECTOR
3186   D788 31 3F                   LEAY  -1,Y                        DECREMENT WAIT TIMER
3187   D78A 26 F8                   BNE   LD784                       KEEP WAITING FOR 1793 DRQ
3188   D78C 20 E2                   BRA   LD770                       GENERATE DRIVE NOT READY ERROR
3189
3190                        * READ A SECTOR
3191   D78E F6 FF 4B        LD78E   LDB   FDCREG+3                    GET DATA BYTE FROM 1793 DATA REGISTER
3192   D791 E7 80                   STB   ,X+                         PUT IT IN RAM
3193   D793 B7 FF 40                STA   DSKREG                      REPROGRAM FDC CONTROL REGISTER
3194   D796 20 F6                   BRA   LD78E                       KEEP GETTING DATA
3195                        * BRANCH HERE ON COMPLETION OF SECTOR READ/WRITE
3196   D798 1C AF           LD798   ANDCC #$AF                        ENABLE IRQ, FIRO
3197   D79A B6 FF 48                LDA   FDCREG                      * GET STATUS & KEEP WRITE PROTECT, RECORD TYPE/WRITE
3198   D79D 84 7C                   ANDA  #$7C                        * FAULT, RECORD NOT FOUND, CRC ERROR OR LOST DATA
3199   D79F 97 F0                   STA   DCSTA                       SAVE IN DSKCON STATUS
3200   D7A1 39                      RTS
3201
3202                        * DSKCON OPERATION CODE JUMP VECTORS
3203   D7A2 D6 C5           LD7A2   FDB   LD6C5                       RESTORE HEAD TO TRACK ZERO
3204   D7A4 D6 DD                   FDB   LD6DD                       NO OP - RETURN
3205   D7A6 D7 05                   FDB   LD705                       READ SECTOR
3206   D7A8 D7 08                   FDB   $D708                       WRITE SECTOR
3207
3208                        * DSKREG MASKS FOR DISK DRIVE SELECT
3209   D7AA 01              LD7AA   FCB   1                           DRIVE SEL 0
3210   D7AB 02                      FDB   2                           DRIVE SEL 1
3211   D7AC 04                      FDB   4                           DRIVE SEL 2
3212   D7AD 40                      FCB   $40                         DRIVE SEL 3
3213
3214                        * NMI SERVICE
3215   D7AE B6 09 82        DNMISV  LDA   NMIFLG                      GET NMI FLAG
3216   D7B1 27 08                   BEQ   LD7BB                       RETURN IF NOT ACTIVE
3217   D7B3 BE 09 83                LDX   DNMISV                      GET NEW RETURN VECTOR
3218   D7B6 AF 6A                   STX   10,S                        STORE AT STACKED PC SLOT ON STACK
3219   D7B8 7F 09 82                CLR   NMIFLG                      RESET NMI FLAG
3220   D7BB 3B              LD7BB   RTI
3221
3222                        * IRQ SERVICE
3223   D7BC B6 FF 03        DIRQSV  LDA   PIA0+3                      63.5 MICRO SECOND OR 60 HZ INTERRUPT?
3224   D7BF 2A FA                   BPL   LD7BB                       RETURN IF 63.5 MICROSECOND
3225   D7C1 B6 FF 02                LDA   PIA0+2                      RESET 60 HZ PIA INTERRUPT FLAG
3226   D7C4 B6 09 85                LDA   RDYTMR                      GET TIMER
3227   D7C7 27 11                   BEQ   LD7DA                       BRANCH IF NOT ACTIVE
3228   D7C9 4A                      DECA                              DECREMENT THE TIMER
3229   D7CA B7 09 85                STA   RDYTMR                      SAVE IT
3230   D7CD 26 0B                   BNE   LD7DA                       BRANCH IF NOT TIME TO TURN OFF DISK MOTORS
3231   D7CF B6 09 86                LDA   DRGRAM                      = GET DSKREG IMAGE
3232   D7D2 84 B0                   ANDA  #$B0                        = TURN ALL MOTORS AND DRIVE SELECTS OFF
3233   D7D4 B7 09 86                STA   DRGRAM                      = PUT IT BACK IN RAM IMAGE
3234   D7D7 B7 FF 40                STA   DSKREG                      SEND TO CONTROL REGISTER (MOTORS OFF)
3235   D7DA 7E 89 55        LD7DA   JMP   L8955                       JUMP TO EXTENDED BASIC'S IRQ HANDLER
3236
3237                        * THIS IS THE END OF DISK BASIC.
3238                        * THE CODE FROM THIS POINT TO $DFFF IS GARBAGE.
3239                        * DOSBAS 1.0 = 2083 WASTED BYTES
3240
```

ALLCOL	ØØB5	DCTRK	ØØEC	DVEC2Ø	C8BØ	IRQ	FFF8	LAD33	AD33
ANGLE	ØØE8	DEBVAL	Ø11B	DVEC22	C2B2	IRQVEC	Ø1ØC	LAD9E	AD9E
ARYDIS	ØØØ8	DEFDRV	Ø95A	DVEC3	CC1C	KEYBUF	Ø152	LADC6	ADC6
ARYEND	ØØ1F	DEFEXT	C2A9	DVEC4	C5BC	KILL	C6EF	LADD4	ADD4
ARYTAB	ØØ1D	DEVCFW	ØØ6A	DVEC5	C848	L813C	813C	LADEB	ADEB
ATTCTR	Ø988	DEVLCF	ØØ6B	DVEC6	C84B	L8168	8168	LAE15	AE15
BACKUP	D262	DEVNUM	ØØ6F	DXCVEC	CF0A	L8311	8311	LAF9A	AF9A
BAKCOL	ØØB3	DEVPOS	ØØ6C	DXIVEC	CF32	L8316	8316	LAFA4	AFA4
BASEXT	C2A6	DEVWID	ØØ6D	ENDCHR	ØØØ2	L836C	836C	LAFB1	AFB1
BAWMST	A0E8	DEXTBF	Ø954	ENDFLG	ØØØØ	L8C1B	8C1B	LASTPT	ØØØD
BEGGRP	ØØBA	DFFLEN	Ø97C	ENDGRP	ØØB7	L95AC	95AC	LBØØC	BØØC
BINEXT	C2AF	DFLBUF	Ø989	EVALEXPB	B70B	L962E	962E	LBØ1E	BØ1E
BINVAL	ØØ2B	DFLTYP	Ø957	EXECJP	ØØ9D	L9650	9650	LB148	B148
BLKCNT	Ø094	DIMFLG	ØØØ5	EXPJMP	Ø11D	L96CB	96CB	LB156	B156
BLKLEN	ØØ7D	DIR	CCA9	FATBLØ	Ø8ØØ	L96EC	96EC	LB166	B166
BLKTYP	ØØ7C	DIRQSV	D8AF	FATBL1	Ø84A	L975F	975F	LB244	B244
BOTSTK	Ø017	DKWMST	CØE7	FATBL2	Ø894	L9FB5	9FB5	LB262	B262
BROMHK	AA1A	DLBAUD	ØØE6	FATBL3	Ø8DE	LAØE2	AØE2	LB26F	B26F
CASBUF	Ø1DA	DLODFL	Ø95D	FCBACT	Ø95B	LA171	A171	LB277	B277
CASFLG	Ø11A	DMRGFL	Ø95E	FCBADR	Ø94A	LA176	A176	LB2CE	B2CE
CBTPHA	Ø084	DNAMBF	Ø94C	FCBTMP	ØØF1	LA282	A282	LB357	B357
CBUFAD	ØØ7E	DNMISV	D8A1	FCBV1	Ø928	LA35F	A35F	LB3E6	B3E6
CCKSUM	Ø08Ø	DNMIVC	Ø983	FDCREG	FF48	LA37C	A37C	LB44A	B44A
CFNBUF	Ø1D1	DOS	D6EC	FIELD	DØBC	LA3ED	A3ED	LB4F3	B4F3
CHARAC	ØØØ1	DOSBAS	CØØØ	FILES	D15C	LA3FB	A3FB	LB50F	B50F
CHARAD	ØØA6	DOSBUF	26ØØ	FILSTA	ØØ78	LA4Ø6	A4Ø6	LB516	B516
CHGFLG	ØØDB	DOSCOM	DFØØ	FIRQ	FFF6	LA426	A426	LB654	B654
CINBFL	ØØ7Ø	DOSINI	DF4C	FORCOL	ØØB2	LA429	A429	LB657	B657
CINCTR	Ø079	DOSVEC	CØØA	FPØEXP	Ø04F	LA42D	A42D	LB659	B659
CINPTR	ØØ7A	DOTVAL	ØØE5	FPØSGN	Ø054	LA549	A549	LB69B	B69B
CLSTSN	Ø085	DRØTRK	Ø97E	FP1EXP	Ø05C	LA59A	A59A	LB6A4	B6A4
CMPØ	Ø09Ø	DRESFL	Ø95C	FP1SGN	Ø061	LA5A2	A5A2	LB70E	B70E
CMP1	Ø091	DRGRAM	Ø986	FPAØ	Ø05Ø	LA5A5	A5A5	LB738	B738
CMPMID	Ø08F	DRIVE	CEC5	FPA1	Ø05D	LA5AE	A5AE	LB73D	B73D
COEFCT	Ø055	DRUNFL	Ø959	FPA2	Ø013	LA5C7	A5C7	LB958	B958
COEFPT	Ø064	DSEC	ØØED	FPCARY	Ø05B	LA5DA	A5DA	LB95C	B95C
COMVEC	Ø12Ø	DSINIT	CØØ8	FPSBYT	Ø063	LA5E4	A5E4	LB99F	B99F
COPY	D3B9	DSKCON	D75F	FREE	CE9C	LA6Ø3	A6Ø3	LB9A2	B9A2
CPERTM	Ø083	DSKI	D4ED	FRESPC	Ø025	LA616	A616	LB9AC	B9AC
CPULWD	Ø082	DSKINI	D599	FRETOP	Ø021	LA61C	A61C	LB9AF	B9AF
CSRERR	Ø081	DSKO	D562	FRQVEC	Ø10F	LA61F	A61F	LB9C5	B9C5
CSSVAL	ØØC1	DSKREG	FF4Ø	GARBFL	ØØØ7	LA7D1	A7D1	LBB91	BB91
CURLIN	Ø068	DSKVAR	CØØ6	GETCCH	ØØA5	LA7E9	A7E9	LBC14	BC14
CURPOS	Ø088	DUSRVC	Ø95F	GETNCH	Ø09F	LA928	A928	LBC33	BC33
CVN	CDF4	DVECØ	C44B	GIVABF	B4F4	LA951	A951	LBC35	BC35
DA	FF2Ø	DVEC1	C888	GRBLOK	Ø086	LA974	A974	LBC5F	BC5F
DASCFL	Ø958	DVEC1Ø	CD35	GRPRAM	ØØBC	LAC37	AC37	LBDCC	BDCC
DATEXT	C2AC	DVEC11	C8A9	HORBEG	ØØBD	LAC44	AC44	LBDD9	BDD9
DATPTR	Ø033	DVEC12	C6E4	HORBYT	ØØB9	LAC46	AC46	LC002	C002
DATTMP	Ø035	DVEC13	CAE4	HORDEF	ØØC7	LAC6Ø	AC6Ø	LC00C	C00C
DATTXT	Ø031	DVEC14	C9ØC	HOREND	ØØC3	LAC73	AC73	LC00F	C00F
DBUFØ	Ø6ØØ	DVEC15	CED2	IFCTR	ØØØ4	LAC7C	AC7C	LC03B	C03B
DBUF1	Ø7ØØ	DVEC17	C265	IKEYIM	Ø087	LACEF	ACEF	LC061	C061
DCBPT	ØØEE	DVEC18	CA3E	INPFLG	Ø009	LAD19	AD19	LC0BD	C0BD
DCDRV	ØØEB	DVEC2	C893	INT	BCEE	LAD21	AD21	LC0C2	C0C2
DCNVEC	CØØ4	DVEC7	CAE9	L8748	8748	LBØ69	BØ69	LC2EA	C2EA
DCOPC	ØØEA	DVEC8	CAF9	L880E	880E	LB143	B143	LC3Ø6	C3Ø6
DCSTA	ØØFØ	DVERFL	Ø987	L8955	8955	LB146	B146	LC3ØB	C3ØB

LC0F0	C0F0	LC627	C627	LC905	C905	LCC15	CC15	LD013	D013
LC109	C109	LC629	C629	LC932	C932	LCC17	CC17	LD015	D015
LC139	C139	LC631	C631	LC935	C935	LCC24	CC24	LD051	D051
LC192	C192	LC64D	C64D	LC938	C938	LCC3A	CC3A	LD056	D056
LC1F1	C1F1	LC653	C653	LC945	C945	LCC40	CC40	LD059	D059
LC219	C219	LC658	C658	LC96A	C96A	LCC5E	CC5E	LD06E	D06E
LC22C	C22C	LC65E	C65E	LC96E	C96E	LCC6A	CC6A	LD06F	D06F
LC238	C238	LC67D	C67D	LC973	C973	LCC6C	CC6C	LD07F	D07F
LC244	C244	LC681	C681	LC978	C978	LCC99	CC99	LD082	D082
LC24E	C24E	LC684	C684	LC97D	C97D	LCCBB	CCBB	LD092	D092
LC256	C256	LC685	C685	LC994	C994	LCCC5	CCC5	LD0A3	D0A3
LC290	C290	LC68C	C68C	LC99B	C99B	LCCEB	CCEB	LD0A7	D0A7
LC2BF	C2BF	LC69B	C69B	LC99D	C99D	LCD08	CD08	LD0B0	D0B0
LC2C1	C2C1	LC6A5	C6A5	LC9B0	C9B0	LCD17	CD17	LD0B9	D0B9
LC2E6	C2E6	LC6B0	C6B0	LC9B7	C9B7	LCD18	CD18	LD0C3	D0C3
LC310	C310	LC6B3	C6B3	LC9BE	C9BE	LCD1B	CD1B	LD0C9	D0C9
LC324	C324	LC6C7	C6C7	LC9D0	C9D0	LCD1E	CD1E	LD0DA	D0DA
LC33E	C33E	LC6D6	C6D6	LC9DF	C9DF	LCD24	CD24	LD0DF	D0DF
LC352	C352	LC6D9	C6D9	LCA04	CA04	LCD4B	CD4B	LD119	D119
LC357	C357	LC6E5	C6E5	LCA07	CA07	LCD5F	CD5F	LD11E	D11E
LC35A	C35A	LC6FC	C6FC	LCA12	CA12	LCD74	CD74	LD132	D132
LC366	C366	LC70F	C70F	LCA27	CA27	LCD81	CD81	LD143	D143
LC37B	C37B	LC71E	C71E	LCA33	CA33	LCD8E	CD8E	LD157	D157
LC3AD	C3AD	LC739	C739	LCA4F	CA4F	LCD92	CD92	LD15A	D15A
LC3B2	C3B2	LC744	C744	LCA50	CA50	LCD97	CD97	LD181	D181
LC3C5	C3C5	LC749	C749	LCA6C	CA6C	LCD98	CD98	LD189	D189
LC3C8	C3C8	LC755	C755	LCA7B	CA7B	LCDA0	CDA0	LD199	D199
LC3CF	C3CF	LC763	C763	LCA7E	CA7E	LCDAC	CDAC	LD1A8	D1A8
LC405	C405	LC76E	C76E	LCAA4	CAA4	LCDB0	CDB0	LD1AF	D1AF
LC40A	C40A	LC779	C779	LCAAF	CAAF	LCDB8	CDB8	LD1EF	D1EF
LC421	C421	LC784	C784	LCAB6	CAB6	LCDBC	CDBC	LD1F8	D1F8
LC429	C429	LC786	C786	LCABD	CABD	LCDCB	CDCB	LD208	D208
LC42F	C42F	LC796	C796	LCABF	CABF	LCDD0	CDD0	LD20B	D20B
LC43E	C43E	LC79C	C79C	LCAC6	CAC6	LCDD5	CDD5	LD212	D212
LC481	C481	LC79D	C79D	LCADA	CADA	LCDD6	CDD6	LD21B	D21B
LC48D	C48D	LC7BF	C7BF	LCAED	CAED	LCDEC	CDEC	LD222	D222
LC4BB	C4BB	LC7C8	C7C8	LCAF8	CAF8	LCE14	CE14	LD236	D236
LC4C7	C4C7	LC7E6	C7E6	LCB01	CB01	LCE19	CE19	LD249	D249
LC4E1	C4E1	LC7E8	C7E8	LCB06	CB06	LCE68	CE68	LD24F	D24F
LC4E8	C4E8	LC7EC	C7EC	LCB2E	CB2E	LCE72	CE72	LD256	D256
LC4F2	C4F2	LC7F8	C7F8	LCB31	CB31	LCEB6	CEB6	LD25F	D25F
LC504	C504	LC7FD	C7FD	LCB41	CB41	LCEBD	CEBD	LD27B	D27B
LC509	C509	LC806	C806	LCB4E	CB4E	LCEE9	CEE9	LD28C	D28C
LC514	C514	LC807	C807	LCB52	CB52	LCEEC	CEEC	LD2A4	D2A4
LC519	C519	LC80D	C80D	LCB54	CB54	LCF07	CF07	LD2A5	D2A5
LC52D	C52D	LC829	C829	LCB6B	CB6B	LCF2A	CF2A	LD2AE	D2AE
LC538	C538	LC82E	C82E	LCB76	CB76	LCF5C	CF5C	LD2CD	D2CD
LC53C	C53C	LC845	C845	LCB8B	CB8B	LCF68	CF68	LD2D2	D2D2
LC567	C567	LC866	C866	LCB8C	CB8C	LCF9B	CF9B	LD2DD	D2DD
LC586	C586	LC868	C868	LCB93	CB93	LCFB3	CFB3	LD2EF	D2EF
LC5A9	C5A9	LC881	C881	LCB97	CB97	LCFB5	CFB5	LD2FA	D2FA
LC5BA	C5BA	LC8AF	C8AF	LCBAD	CBAD	LCFBB	CFBB	LD2FC	D2FC
LC5C4	C5C4	LC8B2	C8B2	LCBB4	CBB4	LCFC1	CFC1	LD2FF	D2FF
LC5EC	C5EC	LC8C2	C8C2	LCBC0	CBC0	LCFDE	CFDE	LD30E	D30E
LC5F9	C5F9	LC8D1	C8D1	LCBC3	CBC3	LCFE3	CFE3	LD310	D310
LC5FE	C5FE	LC8DA	C8DA	LCBCF	CBCF	LCFFA	CFFA	LD322	D322
LC602	C602	LC8F3	C8F3	LCBDF	CBDF	LD004	D004	LD326	D326
LC611	C611	LC8FE	C8FE	LCBE9	CBE9	LD00D	D00D	LD344	D344

LD357	D357	LD7Ø9	D7Ø9	OLDPTR	Ø02D	SW3VEC	Ø1ØØ	VD8	ØØD8
LD35E	D35E	LD7ØB	D7ØB	OLDTXT	Ø029	SWI	FFFA	VD9	ØØD9
LD35F	D35F	LD7ØE	D7ØE	PIAØ	FFØØ	SWI2	FFF4	VDA	ØØDA
LD36C	D36C	LD737	D737	PIA1	FF2Ø	SWI3	FFF2	VERBEG	ØØBF
LD37E	D37E	LD742	D742	PIA2	FF4Ø	SWIVEC	Ø1Ø6	VERDEF	ØØC9
LD399	D399	LD743	D743	PLYTMR	ØØE3	SYNCLN	Ø092	VEREND	ØØC5
LD3AØ	D3AØ	LD75A	D75A	PMODE	ØØB6	SYNCOMMA	B26D	VERIFY	D74E
LD3AD	D3AD	LD765	D765	POTVAL	Ø15A	TEMPO	ØØE2	VIDRAM	Ø4ØØ
LD3CE	D3CE	LD77E	D77E	PRTDEV	Ø06E	TEMPPT	ØØØB	VOLHI	ØØDF
LD3DC	D3DC	LD792	D792	RDYTMR	Ø985	TEMPTR	ØØØF	VOLLOW	ØØEØ
LD41E	D41E	LD7AØ	D7AØ	RELFLG	ØØØA	TIMOUT	ØØE7	WCOLOR	ØØB4
LD423	D423	LD7B1	D7B1	RELPTR	Ø03D	TIMVAL	Ø112	WFATVL	Ø97A
LD42C	D42C	LD7B8	D7B8	RENAME	DØ1B	TINPTR	Ø02F	WRITE	D066
LD44E	D44E	LD7DØ	D7DØ	RESETV	FFFE	TMPLOC	ØØØ3	XBWMST	8ØCØ
LD455	D455	LD7D1	D7D1	RESSGN	Ø062	TMPSTK	ØØDC	XVEC15	8846
LD45D	D45D	LD7D3	D7D3	RNBFAD	Ø948	TMPTR1	Ø011	XVEC17	88FØ
LD45F	D45F	LD7DF	D7DF	ROMPAK	CØØØ	TOPRAM	Ø074	XVEC18	829C
LD472	D472	LD7FØ	D7FØ	RSET	D1Ø1	TRCFLG	ØØAF	XVEC3	8273
LD476	D476	LD7F3	D7F3	RSTFLG	Ø071	TRELFL	Ø03F	XVEC4	8CF1
LD47C	D47C	LD7F8	D7F8	RSTVEC	Ø072	TXTTAB	Ø019	XVEC8	8286
LD482	D482	LD7FB	D7FB	RVECØ	Ø15E	UNLOAD	D233	XVEC9	8E9Ø
LD486	D486	LD82A	D82A	RVEC1	Ø161	USRADR	ØØBØ	ZERO	Ø08A
LD4BA	D4BA	LD82C	D82C	RVEC1Ø	Ø17C	USRJMP	Ø112		
LD4C4	D4C4	LD85B	D85B	RVEC11	Ø17F	V4Ø	Ø04Ø		
LD4D4	D4D4	LD863	D863	RVEC12	Ø182	V41	Ø041		
LD4EA	D4EA	LD86B	D86B	RVEC13	Ø185	V42	Ø042		
LD5Ø8	D5Ø8	LD875	D875	RVEC14	Ø188	V43	Ø043		
LD519	D519	LD877	D877	RVEC15	Ø18B	V44	Ø044		
LD51C	D51C	LD881	D881	RVEC16	Ø18E	V45	Ø045		
LD524	D524	LD88B	D88B	RVEC17	Ø191	V46	Ø046		
LD527	D527	LD895	D895	RVEC18	Ø194	V47	Ø047		
LD54A	D54A	LD89D	D89D	RVEC19	Ø197	V48	Ø048		
LD54D	D54D	LD8AE	D8AE	RVEC2	Ø164	V4A	Ø04A		
LD553	D553	LD8CD	D8CD	RVEC2Ø	Ø19A	V4B	Ø04B		
LD577	D577	LDF9Ø	DF9Ø	RVEC21	Ø19D	V4D	Ø04D		
LD58F	D58F	LDF36	DF36	RVEC22	Ø1AØ	V973	Ø973		
LD5B2	D5B2	LINBUF	Ø2DC	RVEC23	Ø1A3	V974	Ø974		
LD5B9	D5B9	LINHDR	Ø2DA	RVEC24	Ø1A6	V976	Ø976		
LD5C1	D5C1	LIST	B764	RVEC3	Ø167	V977	Ø977		
LD5C3	D5C3	LOAD	CA48	RVEC4	Ø16A	V978	Ø978		
LD5C4	D5C4	LOC	CE1Ø	RVEC5	Ø16D	VAB	ØØAB		
LD5CE	D5CE	LOF	CE37	RVEC6	Ø17Ø	VAC	ØØAC		
LD6Ø6	D6Ø6	LPTBTD	Ø095	RVEC7	Ø173	VAD	ØØAD		
LD612	D612	LPTCFW	Ø099	RVEC8	Ø176	VAE	ØØAE		
LD62Ø	D62Ø	LPTLCF	Ø09A	RVEC9	Ø179	VALTMP	ØØØ6		
LD649	D649	LPTLND	Ø097	RVSEED	Ø115	VARDES	Ø03B		
LD64F	D64F	LPTPOS	Ø09C	SAMREG	FFCØ	VARNAM	Ø037		
LD66F	D66F	LPTWID	Ø09B	SAVE	C9EØ	VARPTR	Ø039		
LD672	D672	LSET	D1Ø2	SCALE	ØØE9	VARTAB	Ø01B		
LD688	D688	LSTTXT	Ø066	SETFLG	ØØC2	VCB	ØØCB		
LD691	D691	MEMSIZ	Ø027	SNDDUR	Ø08D	VCD	ØØCD		
LD69A	D69A	MERGE	CA39	SNDTON	Ø08C	VCF	ØØCF		
LD6C2	D6C2	MKN	CEØ2	STRBUF	Ø3D7	VD1	ØØD1		
LD6C8	D6C8	NMI	FFFC	STRDES	Ø056	VD3	ØØD3		
LD6D4	D6D4	NMIFLG	Ø982	STRINOUT	B99C	VD4	ØØD4		
LD6F2	D6F2	NMIVEC	Ø1Ø9	STRSTK	Ø1A9	VD5	ØØD5		
LD6FB	D6FB	NOTELN	ØØE1	STRTAB	Ø023	VD6	ØØD6		
LD7Ø1	D7Ø1	OCTAVE	ØØDE	SW2VEC	Ø1Ø3	VD7	ØØD7		

ALLCOL	ØØB5	DCTRK	ØØEC	DXCVEC	CE2E	L8748	8748	LASTPT	ØØØD
ANGLE	ØØE8	DEBVAL	Ø11B	DXIVEC	CE56	L88ØE	88ØE	LBØØC	BØØC
ARYDIS	ØØØ8	DEFDRV	Ø95A	ENDCHR	ØØØ2	L8955	8955	LBØ1E	BØ1E
ARYEND	ØØ1F	DEFEXT	C291	ENDFLG	ØØØØ	L8C1B	8C1B	LBØ69	BØ69
ARYTAB	ØØ1D	DEVCFW	ØØ6A	ENDGRP	ØØB7	L95AC	95AC	LB143	B143
ATTCTR	Ø988	DEVLCF	ØØ6B	EVALEXPB	B7ØB	L962E	962E	LB146	B146
BACKUP	D175	DEVNUM	ØØ6F	EXECJP	ØØ9D	L965Ø	965Ø	LB148	B148
BAKCOL	ØØB3	DEVPOS	ØØ6C	EXPJMP	Ø11D	L96CB	96CB	LB156	B156
BASEXT	C28E	DEVWID	ØØ6D	FATBLØ	Ø8ØØ	L96EC	96EC	LB166	B166
BAWMST	AØE8	DEXTBF	Ø954	FATBL1	Ø84A	L975F	975F	LB244	B244
BEGGRP	ØØBA	DFFLEN	Ø97C	FATBL2	Ø894	L9FB5	9FB5	LB262	B262
BINEXT	C297	DFLBUF	Ø989	FATBL3	Ø8DE	LAØE2	AØE2	LB26F	B26F
BINVAL	ØØ2B	DFLTYP	Ø957	FCBACT	Ø95B	LA171	A171	LB277	B277
BLKCNT	ØØ94	DIMFLG	ØØØ5	FCBADR	Ø94A	LA176	A176	LB2CE	B2CE
BLKLEN	ØØ7D	DIR	CBCF	FCBTMP	ØØF1	LA282	A282	LB357	B357
BLKTYP	ØØ7C	DIRQSV	D7BC	FCBV1	Ø928	LA35F	A35F	LB3E6	B3E6
BOTSTK	ØØ17	DKWMST	CØD4	FDCREG	FF48	LA37C	A37C	LB44A	B44A
BROMHK	AA1A	DLBAUD	ØØE6	FIELD	CFEØ	LA3ED	A3ED	LB4F3	B4F3
CASBUF	Ø1DA	DLODFL	Ø95D	FILES	DØ8Ø	LA3FB	A3FB	LB5ØF	B5ØF
CASFLG	Ø11A	DMRGFL	Ø95E	FILSTA	ØØ78	LA4Ø6	A4Ø6	LB516	B516
CBTPHA	ØØ84	DNAMBF	Ø94C	FIRQ	FFF6	LA426	A426	LB654	B654
CBUFAD	ØØ7E	DNMISV	D7AE	FORCOL	ØØB2	LA429	A429	LB657	B657
CCKSUM	ØØ8Ø	DNMIVC	Ø983	FPØEXP	Ø04F	LA42D	A42D	LB659	B659
CFNBUF	Ø1D1	DOSBAS	C ØØØ	FPØSGN	Ø054	LA549	A549	LB69B	B69B
CHARAC	ØØØ1	DOSBUF	26ØØ	FP1EXP	Ø05C	LA59A	A59A	LB6A4	B6A4
CHARAD	ØØA6	DOTVAL	ØØE5	FP1SGN	Ø061	LA5A2	A5A2	LB7ØE	B7ØE
CHGFLG	ØØDB	DRØTRK	Ø97E	FPAØ	Ø05Ø	LA5A5	A5A5	LB738	B738
CINBFL	Ø07Ø	DRESFL	Ø95C	FPA1	Ø05D	LA5AE	A5AE	LB73D	B73D
CINCTR	Ø079	DRGRAM	Ø986	FPA2	Ø013	LA5C7	A5C7	LB958	B958
CINPTR	Ø07A	DRIVE	CDE9	FPCARY	Ø05B	LA5DA	A5DA	LB95C	B95C
CLSTSN	Ø085	DRUNFL	Ø959	FPSBYT	Ø063	LA5E4	A5E4	LB99F	B99F
CMPØ	Ø09Ø	DSEC	ØØED	FREE	CDCØ	LA6Ø3	A6Ø3	LB9A2	B9A2
CMP1	Ø091	DSKCON	D66C	FRESPC	Ø025	LA616	A616	LB9AC	B9AC
CMPMID	Ø08F	DSKI	D3FF	FRETOP	Ø021	LA61C	A61C	LB9AF	B9AF
COEFCT	Ø055	DSKINI	D4AB	FRQVEC	Ø1ØF	LA61F	A61F	LB9C5	B9C5
COEFPT	Ø064	DSKO	D474	GARBFL	Ø007	LA7D1	A7D1	LBB91	BB91
COMVEC	Ø12Ø	DSKREG	FF4Ø	GETCCH	ØØA5	LA7E9	A7E9	LBC14	BC14
COPY	D2CC	DSKVAR	CØØ6	GETNCH	Ø09F	LA928	A928	LBC33	BC33
CPERTM	Ø083	DUSRVC	Ø95F	GIVABF	B4F4	LA951	A951	LBC35	BC35
CPULWD	Ø082	DVECØ	C426	GRBLOK	Ø086	LA974	A974	LBC5F	BC5F
CSRERR	Ø081	DVEC1	C838	GRPRAM	ØØBC	LAC37	AC37	LBDCC	BDCC
CSSVAL	ØØC1	DVEC1Ø	CC5B	HORBEG	ØØBD	LAC44	AC44	LBDD9	BDD9
CURLIN	Ø068	DVEC11	C859	HORBYT	ØØB9	LAC46	AC46	LCØØ2	CØØ2
CURPOS	Ø088	DVEC12	C6B7	HORDEF	ØØC7	LAC6Ø	AC6Ø	LCØØ8	CØØ8
CVN	CD1A	DVEC13	CA36	HOREND	ØØC3	LAC73	AC73	LCØØB	CØØB
DA	FF2Ø	DVEC14	C86Ø	IFCTR	Ø004	LAC7C	AC7C	LCØ37	CØ37
DASCFL	Ø958	DVEC15	CDF6	IKEYIM	Ø087	LACEF	ACEF	LCØ57	CØ57
DATEXT	C294	DVEC17	C24D	INPFLG	Ø009	LAD19	AD19	LCØB3	CØB3
DATPTR	Ø033	DVEC18	C99Ø	INT	BCEE	LAD21	AD21	LCØDD	CØDD
DATTMP	Ø035	DVEC2	C843	IRQ	FFF8	LAD33	AD33	LCØF6	CØF6
DATTXT	Ø031	DVEC22	C29A	IRQVEC	Ø1ØC	LAD9E	AD9E	LC126	C126
DBUFØ	Ø6ØØ	DVEC3	CB4A	KEYBUF	Ø152	LADC6	ADC6	LC17F	C17F
DBUF1	Ø7ØØ	DVEC4	C58F	KILL	C6C2	LADD4	ADD4	LC1DB	C1DB
DCBPT	ØØEE	DVEC5	C818	L813C	813C	LADEB	ADEB	LC2Ø1	C2Ø1
DCDRV	ØØEB	DVEC6	C81B	L8168	8168	LAE15	AE15	LC214	C214
DCNVEC	CØØ4	DVEC7	CA3B	L8311	8311	LAF9A	AF9A	LC22Ø	C22Ø
DCOPC	ØØEA	DVEC8	CA4B	L8316	8316	LAFA4	AFA4	LC22C	C22C
DCSTA	ØØFØ	DVERFL	Ø987	L836C	836C	LAFB1	AFB1	LC236	C236

LC23E	C23E	LC657	C657	LC956	C956	LCCB4	CCB4	LDØ7B	DØ7B
LC278	C278	LC658	C658	LC959	C959	LCCB8	CCB8	LDØ7E	DØ7E
LC2A7	C2A7	LC65F	C65F	LC964	C964	LCCBD	CCBD	LDØA5	DØA5
LC2A9	C2A9	LC66E	C66E	LC979	C979	LCCBE	CCBE	LDØBØ	DØBØ
LC2CC	C2CC	LC678	C678	LC985	C985	LCCC6	CCC6	LDØCØ	DØCØ
LC2DØ	C2DØ	LC683	C683	LC9A1	C9A1	LCCD2	CCD2	LDØCF	DØCF
LC2E8	C2E8	LC686	C686	LC9A2	C9A2	LCCD6	CCD6	LD1Ø2	D1Ø2
LC2ED	C2ED	LC69A	C69A	LC9BE	C9BE	LCCDE	CCDE	LD1ØB	D1ØB
LC2F2	C2F2	LC6A9	C6A9	LC9CD	C9CD	LCCE2	CCE2	LD11B	D11B
LC3Ø6	C3Ø6	LC6AC	C6AC	LC9DØ	C9DØ	LCCF1	CCF1	LD11E	D11E
LC32Ø	C32Ø	LC6B8	C6B8	LC9F6	C9F6	LCCF6	CCF6	LD125	D125
LC334	C334	LC6CF	C6CF	LCAØ1	CAØ1	LCCFB	CCFB	LD12E	D12E
LC339	C339	LC6E2	C6E2	LCAØ8	CAØ8	LCCFC	CCFC	LD135	D135
LC33C	C33C	LC6F1	C6F1	LCAØF	CAØF	LCD12	CD12	LD149	D149
LC348	C348	LC7ØC	C7ØC	LCA11	CA11	LCD3A	CD3A	LD15C	D15C
LC35D	C35D	LC714	C714	LCA18	CA18	LCD3D	CD3D	LD162	D162
LC38F	C38F	LC719	C719	LCA2C	CA2C	LCD8C	CD8C	LD169	D169
LC394	C394	LC725	C725	LCA3F	CA3F	LCD96	CD96	LD172	D172
LC3A7	C3A7	LC733	C733	LCA4A	CA4A	LCDDA	CDDA	LD18E	D18E
LC3AA	C3AA	LC73E	C73E	LCA53	CA53	LCDE1	CDE1	LD19F	D19F
LC3B1	C3B1	LC749	C749	LCA58	CA58	LCEØD	CEØD	LD1B7	D1B7
LC3EØ	C3EØ	LC754	C754	LCA8Ø	CA8Ø	LCE1Ø	CE1Ø	LD1B8	D1B8
LC3E5	C3E5	LC756	C756	LCA8E	CA8E	LCE2B	CE2B	LD1C1	D1C1
LC3FC	C3FC	LC766	C766	LCA9B	CA9B	LCE4E	CE4E	LD1EØ	D1EØ
LC4Ø4	C4Ø4	LC76C	C76C	LCA9F	CA9F	LCE8Ø	CE8Ø	LD1E5	D1E5
LC4ØA	C4ØA	LC76D	C76D	LCAA1	CAA1	LCE8C	CE8C	LD1FØ	D1FØ
LC419	C419	LC78F	C78F	LCAB8	CAB8	LCEBF	CEBF	LD2Ø2	D2Ø2
LC45C	C45C	LC798	C798	LCAC3	CAC3	LCED7	CED7	LD2ØD	D2ØD
LC468	C468	LC7B6	C7B6	LCAD8	CAD8	LCED9	CED9	LD211	D211
LC496	C496	LC7B8	C7B8	LCAD9	CAD9	LCEDF	CEDF	LD212	D212
LC4A2	C4A2	LC7BC	C7BC	LCAEØ	CAEØ	LCEE5	CEE5	LD221	D221
LC4BC	C4BC	LC7C8	C7C8	LCAE2	CAE2	LCFØ2	CFØ2	LD223	D223
LC4C2	C4C2	LC7CD	C7CD	LCAEE	CAEE	LCFØ7	CFØ7	LD235	D235
LC4CC	C4CC	LC7D6	C7D6	LCAF1	CAF1	LCF1E	CF1E	LD239	D239
LC4DE	C4DE	LC7D7	C7D7	LCAFD	CAFD	LCF28	CF28	LD257	D257
LC4E3	C4E3	LC7DD	C7DD	LCBØD	CBØD	LCF31	CF31	LD26A	D26A
LC4EC	C4EC	LC7F9	C7F9	LCB17	CB17	LCF37	CF37	LD271	D271
LC5ØØ	C5ØØ	LC7FE	C7FE	LCB43	CB43	LCF39	CF39	LD272	D272
LC5ØB	C5ØB	LC815	C815	LCB45	CB45	LCF75	CF75	LD27F	D27F
LC5ØF	C5ØF	LC836	C836	LCB52	CB52	LCF7A	CF7A	LD291	D291
LC53A	C53A	LC85F	C85F	LCB68	CB68	LCF7D	CF7D	LD2AC	D2AC
LC559	C559	LC884	C884	LCB6E	CB6E	LCF92	CF92	LD2B3	D2B3
LC57C	C57C	LC887	C887	LCB8C	CB8C	LCF93	CF93	LD2CØ	D2CØ
LC58D	C58D	LC88A	C88A	LCB98	CB98	LCFA3	CFA3	LD2E1	D2E1
LC597	C597	LC897	C897	LCB9A	CB9A	LCFA6	CFA6	LD2EE	D2EE
LC5BF	C5BF	LC8BC	C8BC	LCBC3	CBC3	LCFB6	CFB6	LD33Ø	D33Ø
LC5CC	C5CC	LC8CØ	C8CØ	LCBC4	CBC4	LCFC7	CFC7	LD335	D335
LC5D1	C5D1	LC8C5	C8C5	LCBE1	CBE1	LCFCB	CFCB	LD33E	D33E
LC5D5	C5D5	LC8CA	C8CA	LCBEB	CBEB	LCFD4	CFD4	LD36Ø	D36Ø
LC5E4	C5E4	LC8CF	C8CF	LCC11	CC11	LCFDD	CFDD	LD367	D367
LC5FA	C5FA	LC8E6	C8E6	LCC2E	CC2E	LCFE7	CFE7	LD36F	D36F
LC5FC	C5FC	LC8ED	C8ED	LCC3D	CC3D	LCFED	CFED	LD371	D371
LC6Ø4	C6Ø4	LC8EF	C8EF	LCC44	CC44	LCFFE	CFFE	LD384	D384
LC626	C626	LC9Ø2	C9Ø2	LCC4A	CC4A	LDØØ3	DØØ3	LD388	D388
LC62B	C62B	LC9Ø9	C9Ø9	LCC71	CC71	LDØ3D	DØ3D	LD38E	D38E
LC631	C631	LC91Ø	C91Ø	LCC85	CC85	LDØ42	DØ42	LD394	D394
LC65Ø	C65Ø	LC922	C922	LCC9A	CC9A	LDØ56	DØ56	LD398	D398
LC654	C654	LC931	C931	LCCA7	CCA7	LDØ67	DØ67	LD3CC	D3CC

LD3D6	D3D6	LD768	D768	RVEC13	Ø185	V42	Ø042
LD3E6	D3E6	LD77Ø	D77Ø	RVEC14	Ø188	V43	Ø043
LD3FC	D3FC	LD778	D778	RVEC15	Ø18B	V44	Ø044
LD41A	D41A	LD782	D782	RVEC16	Ø18E	V45	Ø045
LD42B	D42B	LD784	D784	RVEC17	Ø191	V46	Ø046
LD42E	D42E	LD78E	D78E	RVEC18	Ø194	V47	Ø047
LD436	D436	LD798	D798	RVEC19	Ø197	V48	Ø048
LD439	D439	LD7A2	D7A2	RVEC2	Ø164	V4A	Ø04A
LD45C	D45C	LD7AA	D7AA	RVEC2Ø	Ø19A	V4B	Ø04B
LD45F	D45F	LD7BB	D7BB	RVEC21	Ø19D	V4D	Ø04D
LD465	D465	LD7DA	D7DA	RVEC22	Ø1AØ	V973	Ø973
LD489	D489	LINBUF	Ø2DC	RVEC23	Ø1A3	V974	Ø974
LD4A1	D4A1	LINHDR	Ø2DA	RVEC24	Ø1A6	V976	Ø976
LD4C4	D4C4	LIST	B764	RVEC3	Ø167	V977	Ø977
LD4CB	D4CB	LOAD	C99A	RVEC4	Ø16A	V978	Ø978
LD4D3	D4D3	LOC	CD36	RVEC5	Ø16D	VAB	ØØAB
LD4D5	D4D5	LOF	CD5B	RVEC6	Ø17Ø	VAC	ØØAC
LD4D6	D4D6	LPTBTD	Ø095	RVEC7	Ø173	VAD	ØØAD
LD4EØ	D4EØ	LPTCFW	Ø099	RVEC8	Ø176	VAE	ØØAE
LD519	D519	LPTLCF	Ø09A	RVEC9	Ø179	VALTMP	Ø006
LD525	D525	LPTLND	Ø097	RVSEED	Ø115	VARDES	Ø03B
LD533	D533	LPTPOS	Ø09C	SAMREG	FFCØ	VARNAM	Ø037
LD55C	D55C	LPTWID	Ø09B	SAVE	C932	VARPTR	Ø039
LD562	D562	LSET	D026	SCALE	ØØE9	VARTAB	Ø01B
LD582	D582	LSTTXT	Ø066	SETFLG	ØØC2	VCB	ØØCB
LD585	D585	MEMSIZ	Ø027	SNDDUR	Ø08D	VCD	ØØCD
LD59B	D59B	MERGE	C98B	SNDTON	Ø08C	VCF	ØØCF
LD5AD	D5AD	MKN	CD28	STRBUF	Ø3D7	VD1	ØØD1
LD5D5	D5D5	NMI	FFFC	STRDES	Ø056	VD3	ØØD3
LD5DB	D5DB	NMIFLG	Ø982	STRINOUT	B99C	VD4	ØØD4
LD5FF	D5FF	NMIVEC	Ø109	STRSTK	Ø1A9	VD5	ØØD5
LD6Ø8	D6Ø8	NOTELN	ØØE1	STRTAB	Ø023	VD6	ØØD6
LD6ØE	D6ØE	OCTAVE	ØØDE	SW2VEC	Ø103	VD7	ØØD7
LD616	D616	OLDPTR	Ø02D	SW3VEC	Ø1ØØ	VD8	ØØD8
LD618	D618	OLDTXT	Ø029	SWI	FFFA	VD9	ØØD9
LD61B	D61B	PIAØ	FFØØ	SWI2	FFF4	VDA	ØØDA
LD644	D644	PIA1	FF2Ø	SWI3	FFF2	VERBEG	ØØBF
LD64F	D64F	PIA2	FF4Ø	SWIVEC	Ø106	VERDEF	ØØC9
LD65Ø	D65Ø	PLYTMR	ØØE3	SYNCLN	Ø092	VEREND	ØØC5
LD667	D667	PMODE	ØØB6	SYNCOMMA	B26D	VERIFY	D65B
LD672	D672	POTVAL	Ø15A	TEMPO	ØØE2	VIDRAM	Ø4ØØ
LD68B	D68B	PRTDEV	Ø06E	TEMPPT	ØØØB	VOLHI	ØØDF
LD69F	D69F	RDYTMR	Ø985	TEMPTR	ØØØF	VOLLOW	ØØEØ
LD6AD	D6AD	RELFLG	ØØØA	TIMOUT	ØØE7	WCOLOR	ØØB4
LD6BE	D6BE	RELPTR	Ø03D	TIMVAL	Ø112	WFATVL	Ø97A
LD6C5	D6C5	RENAME	CF3F	TINPTR	Ø02F	WRITE	CF8A
LD6D4	D6D4	RESETV	FFFE	TMPLOC	ØØØ3	XBWMST	80CØ
LD6DD	D6DD	RESSGN	Ø062	TMPSTK	ØØDC	XVEC15	8846
LD6DE	D6DE	RNBFAD	Ø948	TMPTR1	Ø011	XVEC17	88FØ
LD6EØ	D6EØ	ROMPAK	CØØØ	TOPRAM	Ø074	XVEC18	829C
LD6EC	D6EC	RSET	D025	TRCFLG	ØØAF	XVEC3	8273
LD6FD	D6FD	RSTFLG	Ø071	TRELFL	Ø03F	XVEC4	8CF1
LD7ØØ	D7ØØ	RSTVEC	Ø072	TXTTAB	Ø019	XVEC8	8286
LD7Ø5	D7Ø5	RVECØ	Ø15E	UNLOAD	D146	XVEC9	8E9Ø
LD7Ø7	D7Ø7	RVEC1	Ø161	USRADR	ØØBØ	ZERO	Ø08A
LD7Ø8	D7Ø8	RVEC1Ø	Ø17C	USRJMP	Ø112		
LD737	D737	RVEC11	Ø17F	V4Ø	Ø04Ø		
LD739	D739	RVEC12	Ø182	V41	Ø041		

MODIFIED REGISTERS	1.1 ADDRESS	1.Ø ADDRESS	DESCRIPTION
B,X,Y,U	C48D	C468	OPEN DISK FILE - enter with the mode (I,O,D,R) in ACCA, file number in ACCB, filename.ext in DNAMBF, and DFLTYP, DASCFL & DFFLEN initialized. An FCB will be opened and initialized but any errors will cause control to be returned to BASIC.
A,B,X,U	C52D	C5ØØ	INITIALIZE FCB FOR INPUT - enter with the address of the desired FCB in FCBTMP and valid directory information in V973-V978. The FCB will be properly initialized and the directory will be loaded with the number of bytes in the last sector of the file.
A,B,X,U	C538	C5ØB	INITITALIZE FCB - this is the same as initialize FCB for input except that the number of bytes in the last sector is not written into the directory.
A,B,X,U	C567	C53A	SET UP DIRECTORY ENTRY - scan the directory for the first unused entry and open a file for the information contained in DNAMBF, DFLTYP, and DASCFL. The first unused directory entry will be allocated to the file as will the first unused granule.
A	C5C4	C597	DISK CONSOLE IN - get a byte from the already OPENed disk file specified by DEVNUM. Return the byte in ACCA.
ALL	C68C	C65F	SEARCH DIRECTORY - search the directory for the filename and extension located in DNAMBF. Set the variables V973-V978 accordingly.
ALL	C6F5	C6C8	KILL FILE - kill the file whose name is in DNAMBF.
A,B,X,U	C7ØF	C6E2	FREE FILE GRANULES - enter with the number of the first granule in a file in ACCB. Free (set to $FF) all of the granules in that file and save the new FAT on the disk.
X	C744	C714	SET X TO FILE BUFFER - enter with ACCB containing the file number. Return with X pointing to the correct FCB and the flags set according to the file type.
X	C755	C725	SET X TO FAT - point X to the FAT RAM image for thE drive number stored in DCDRV.
A,B	C763	C733	CONVERT GRANULE TO TRACK & SECTOR - enter with X pointing to an FCB. The current granule number (FCBCGR) will be converted to the equivalent track and sector numbers in DCTRK & DSEC.
A,B	C779	C749	MULTIPLY ACCD BY NINE - multiply the value in ACCD by nine.
A,B	C784	C754	CONVERT SECTORS TO GRANULE - enter with a total number of sectors in ACCD. Convert this number into

			the number of complete granules (Ø-67) contained in that many sectors and return the count in ACCD.
A,B,X	C79D	C76D	READ FAT DATA - load the RAM image of the FAT with data from the disk. Data will not be loaded into the RAM image if any disk files are OPEN.
A,B,X	C7BF	C78F	FIND FREE GRANULE - find the first free granule. Enter with the granule at which to start the search in ACCB. The found granule is marked with a $CØ to indicate that it is the last granule in the file and the number of the granule is returned in ACCA.
B,X	C8Ø7	C7D7	FILE OPEN CHECK - check all active files to make sure a file is not already OPEN. Enter with ACCA containing a file type to disable the AO error for that file type.
A,B,X,U	C935	C887	GET FILENAME.EXT:DRIVE FROM BASIC - get the file-name extension and drive number from a BASIC input line.
A,B,X,U	CD1E	CC44	GET GRANULE COUNT - enter with the granule number of the first granule in a file. The number of whole granules in that file will be returned in ACCA. ACCB will contain the data from the last granule in the file.
A,B,X	CEA8	CDCC	GET FREE GRANULE COUNT - enter with a drive number (Ø-3) in ACCB, return the number of free granules in floating point accumulator Ø.
NONE	D75F	D66C	DSKCON - universal disk I/O routine. A detailed explanation is available in the Color Computer disk user's manual.
A,B,X	D7B8	D6C5	RESTORE HEAD TO TRACK ZERO - restore the head for the drive in DCDRV to track zero. Return DCSTA = $1Ø if there is a SEEK error.
ALL	D7F8	D7Ø5	READ ONE SECTOR - read one sector as specified by the DSKCON parameters (DSEC,DCTRK,DCDRV) and store the data at the address in DCBPT.
ALL	D7FB	D7Ø8	WRITE ONE SECTOR - write one sector as specified by the DSKCON parameters (DSEC,DCTRK,DCDRV) and get the data to go on the disk from the address in DCBPT.

1.1		1.0		
START	END	START	END	DESCRIPTION
C000	C001	C000	C001	DISK BASIC ROM IDENTIFIER
C004	C00B	C004	C007	INDIRECT JUMP TABLE
C109	C112	C0F6	C0FF	COMMAND INTERPRETATION TABLE ROM IMAGE
C113	C138	C100	C125	RAM HOOKS ROM IMAGE
C139	C191	C126	C17E	COPYRIGHT MESSAGES
C192	C1F0	C17F	C1DA	PRIMARY RESERVED WORD TABLE
C1F1	C218	C1DB	C200	PRIMARY RESERVED WORD DISPATCH TABLE
C219	C22B	C201	C213	SECONDARY RESERVED WORD TABLE
C22C	C237	C214	C21F	SECONDARY RESERVED WORD DISPATCH TABLE
C290	C2A5	C278	C28D	ERROR MESSAGES
C2A6	C2B1	C28E	C299	DISK FILE EXTENSION MESSAGES
D35F	D398	D272	D2AB	INSERT SOURCE/DESTINATION MESSAGES
D6D4	D6EB	D5E7	D5FE	DISK FORMATTING DATA TABLE
D895	D89C	D7A2	D7A9	DSKCON OPERATION CODE JUMP VECTORS
D89D	D8A0	D7AA	D7AD	DSKREG MASKS FOR DRIVE SELECT

There are times when it is useful to cause an error message to be printed to the screen in the same manner that BASIC prints its error messages. The following table is provided to give the user the DISK BASIC entry points which will cause error messages to be printed to the screen. A JMP to one of these error message routines will cause the two letter short form error message to be printed on the screen and a pseudo warm start into BASIC will be taken. The pseudo warm start will reset the stack, the string stack and the continue pointer and jump to BASIC s direct mode (OK).

DISK BASIC ERROR JUMPS

NAME	NBR	1.1 LABEL	1.Ø LABEL	1.1 ADDR	1.Ø ADDR	DESCRIPTION
IO	2Ø	LD7Ø9	LD616	D7Ø9	D616	INPUT/OUTPUT
IE	23	LC352	LC334	C352	C334	INPUT PAST END OF FILE
NE	26	LC6E5	LC6B8	C6E5	C6B8	FILE NOT FOUND
BR	27	LC3ØB	LC2ED	C3ØB	C2ED	BAD RECORD
DF	28	LC7F8	LC7C8	C7F8	C7C8	DISK FULL
OB	29	LC5Ø4	LC4DE	C5Ø4	C4DE	OUT OF BUFFER SPACE
FN	31	LC978	LC8CA	C978	C8CA	BAD FILE NAME
FS	32	LC653	LC626	C653	C626	BAD FILE STRUCTURE
FO	34	LDØDA	LCFFE	DØDA	CFFE	FIELD OVERFLOW
SE	35	LD119	LDØ3D	D119	DØ3D	SET TO NON-FIELDED STRING
VF	36	LD74A	LD657	D74A	D657	VERIFICATION ERROR
ER	37	LCDCB	LCFF1	CDCB	CFF1	WRITE OR INPUT PAST END OF RECORD

There are no unconditional jump entry points for error #3Ø (WP - Write Proctected) or error #33 (AE - file Already Exists). These errors may be generated by loading a value equal to 2*(error number) into ACCB and then JMPing to AC46.

DISPLAY CHARACTER SET

HEX VALUE		CHARACTER	HEX VALUE		CHARACTER	HEX VALUE		CHARACTER
Non-Inverted	Inverted		Non-Inverted	Inverted		Non-Inverted	Inverted	
00	40	@	18	58	X	30	40	0
01	41	A	19	59	Y	31	41	1
02	42	B	1A	5A	Z	32	42	2
03	43	C	1B	5B	[	33	43	3
04	44	D	1C	5C	\	34	44	4
05	45	E	1D	5D	]	35	45	5
06	46	F	1E	5E	↑	36	46	6
07	47	G	1F	5F	←	37	47	7
08	48	H	20	60		38	48	8
09	49	I	21	61	!	39	49	9
0A	4A	J	22	62	"	3A	4A	:
0B	4B	K	23	63	#	3B	4B	;
0C	4C	L	24	64	$	3C	4C	<
0D	4D	M	25	65	%	3D	4D	=
0E	4E	N	26	66	&	3E	4E	>
0F	4F	O	27	67	'	3F	4F	?
10	50	P	28	68	(			
11	51	Q	29	69	)			
12	52	R	2A	6A	*			
13	53	S	2B	6B	+			
14	54	T	2C	6C	,			
15	55	U	2D	6D	-			
16	56	V	2E	6E	.			
17	57	W	2F	6F	/			