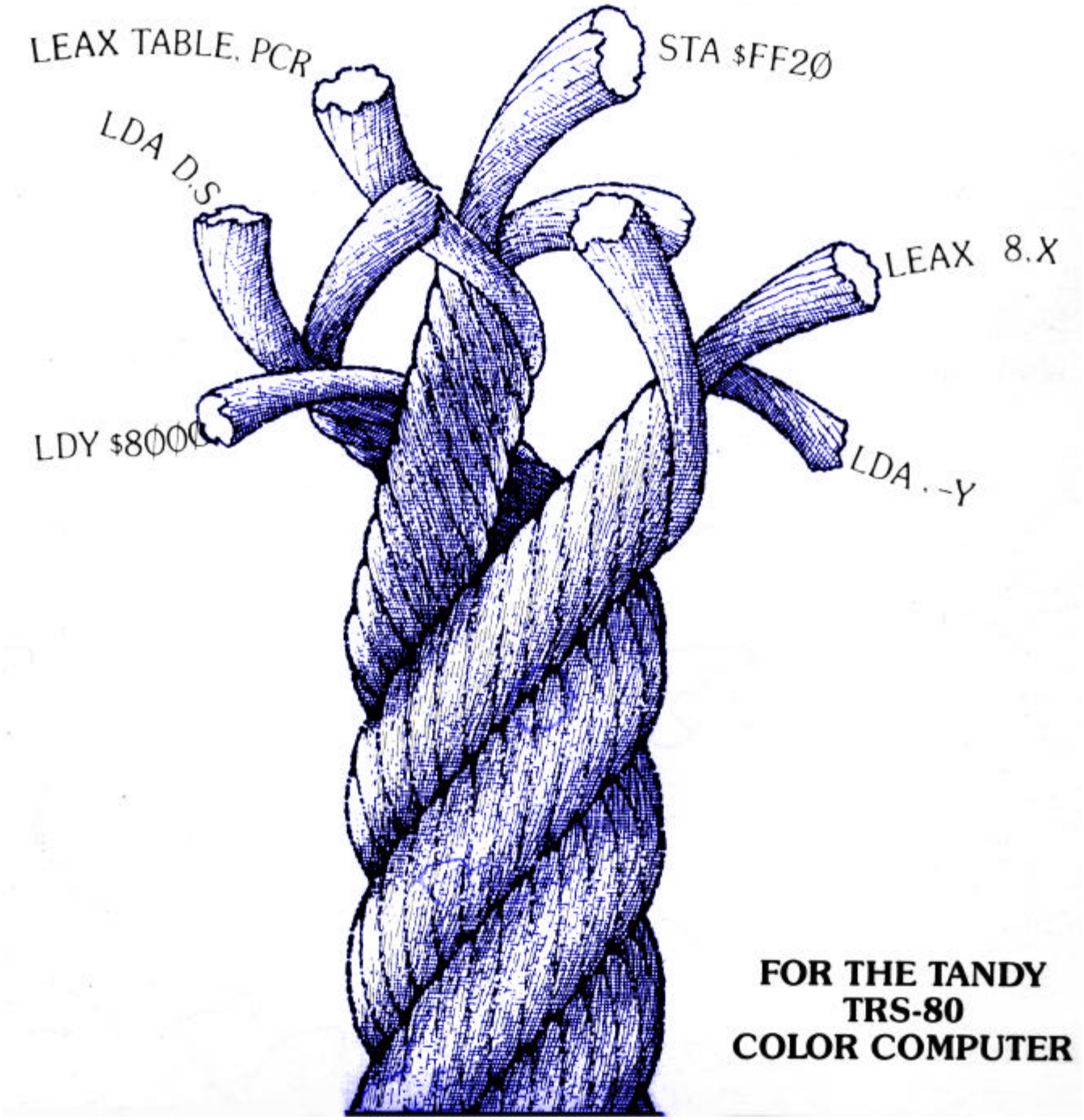
EXTENDED BASIC UNRAVELLED II



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 0001.
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 0 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

Extended Basic Unravelled will provide the reader with a complete, detailed and fully commented assembly listing of the graphics package of Radio Shack s COLOR BASIC. It is not within the scope of this book to teach the neophyte how to develop his own color graphics or high-level arithmetic function routines. 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 Basic Users manual which contains a general description of the overall operation of Basic and much useful information concerning the manner in which the high resolution graphics information is processed and put on the screen. The information and routines explained in this book will allow the user to understand how the Color Computer s routines alter the graphics screens and even allow the user to build his own routines to interface with the graphics routines in the Extended Basic ROMs.

No attempt will be made to re-explain the functions of BASIC or any routines, which were explained in the first book of the Color BASIC Unravelled series. The reader should be aware of the fact that Extended Basic is not a stand-alone system. There are many direct calls into the Basic ROMs. These calls are not explained in this book and it will be necessary for the reader to refer to the other Color 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 in Appendix A (Memory Map), and a symbol table showing the location of the variables is also given.

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 ROM (Revision 1.Ø) used only 6K of the available 8K ROM space, and the second disk ROM (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 Extended Basic ROMs the changes are relatively minor and Appendix G details the differences between the Version 1.Ø and 1.1 Extended Basic ROMs. The op code of each instruction in the disassembly listing has been removed, however the object code value of the instruction s address field has been retained in order to assist the reader to locate variables and subroutines referred to by the instruction.

HOW TO USE THIS BOOK

Extended BASIC Unravelled is a commented, disassembled listing of the Color Computer Extended 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 addresses 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 C 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 a small group of EQUates at the beginning of the Extended Basic disassembly listing (Appendix B).

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 is more efficient in terms of processor time (one cycle shorter).

The reader will find a few places in the disassembly where an instruction such as LDA #0 is found. These instructions usually stem from an original source code instruction, which is like LDA #LABEL with LABEL equal to zero. The original programmer did not go back and change those instructions to a CLRA. In some instances an LDA #0 may be necessary, as the programmer did not wish the instruction to modify the CARRY flag.

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 EXTENDED BASIC

Extended Basic provides several enhancements to the original Color Basic ROM. These enhancements are primarily the new graphics commands with major space devoted to the DLOAD, PRINT USING and complex mathematical commands. There is a significant amount of space used to interface Color and Extended Basic through the RAM vectors (hooks), which also allow the addition of some features (&H and &O number types, CLOADing binary blocks, etc.). Extended Basic does not modify the overall BASIC operating system as established by Color Basic. No new variable types (integer, double precision) are introduced and the variable evaluation and storage procedures are identical. Color Basic s floating point and expression evaluation routines are used.

All of the complex mathematical functions are generated in the same manner. Any mathematical function, which is continuous within a certain set of bounds, may be represented by an infinite polynomial of the form:

a+bX+cX^2+dX^3+eX^4+...

A series of this form is referred to as a Taylor Series and the values a,b,c,d,e... are referred to as the coefficients of the series. This is the type of polynomial used in the Color Computer to evaluate its complex mathematical functions such as LN, SIN, COS, ATN, EXP etc. A computer may be powerful but it still cannot evaluate an infinite series in a finite amount of time. Therefore, the computer truncates the Taylor series after a certain number of terms of the polynomial have been evaluated. This truncation will obviously induce an error and the number of terms kept will determine how large the error is. The error of a Taylor series expansion is not constant over the entire range of the particular mathematical function being evaluated. For some functions the error may be negligible at one end of the range and blow up to an unacceptable value at the opposite end of the range. In order to reduce this wide range of error values, the Taylor series coefficients have had the Tchebyshev correction factor applied to them. This causes the error to be much more uniform over the entire range of the function. The error will not be allowed to blow up to an unacceptable value at any point within the function.

PRINT USING is a complex print formatting command, which consumes over 1/8 of the space in the Extended Basic ROM. There is a good description of PRINT USING and EDIT, another large Extended Basic command in the Extended Basic users manual so they will not be explained here. DLOAD is the most obscure command in the Color Computer and absorbs a substantial amount of space in the ROM. DLOAD is so poorly understood because Tandy has never made the necessary companion routine, DSEND. DLOAD will DOWNLOAD a file over the RS 232 line from another system, however there is no companion routine, which will transmit a file over the RS 232 line to another Color Computer. Once a DSEND routine is built and made available to the masses, DLOAD will be much better understood.

The graphics commands have been developed to use several of the different graphics modes, which are available in the 6847 Video Display Generator (VDG). Only the higher resolution modes are used and both two and four color modes are used. Using all of these modes causes some difficulty in how the pixels (graphic data

points) are accessed. Since the different graphic modes have varying numbers of pixels per horizontal and vertical coordinates, all of the different PMODEs (VDG graphic modes) will allow a horizontal coordinate from Ø-255 and a vertical coordinate from Ø-191. The horizontal and vertical coordinates are normalized for the different PMODEs. The normalization process will scale the horizontal and vertical coordinates to fit whichever PMODE has been selected.

The VDG does not organize the display data in terms of X (horizontal) and Y (vertical) coordinates. It expects the data to be a continuous stream from left to right, top to bottom. Accordingly, a method must be devised which will translate the X and Y coordinates used by BASIC into the absolute RAM address (screen position) of the particular pixel in question and which position inside the byte that the pixel occupies. The pixel position is determined and kept track of by maintaining a "mask" in ACCA. The mask is a byte with the bit positions corresponding to the correct pixel set to "1". The routine which calculates the screen position and mask for a certain X and Y coordinate is called a CALPOS (CALculate POSition) routine.

All of the BASIC graphics routines require their parameters to be given in terms of X and Y coordinates. Any data manipulation, which is required, is performed on the coordinates, which are then translated into a screen position by CALPOS in order to turn on the appropriate pixel on the screen.

Listed below is a brief description of all of the graphics routines including some little known features of some of the routines:

COMMONLY USED TERMS

NORMALIZING	A routine which takes the current X,Y coordinates (which are entered from BASIC as 0-255 for the X coord and 0-191 for the Y coord) and converts them into X,Y coordinates for the current PMODE.
CALPOS	A routine which calculates an absolute screen address from the X,Y coordinates. This is accomplished by multiplying the vertical (Y) coordinate by the number of bytes per horizontal row and adding to that the start address of the current graphics page. Next, the horizontal (X) coordinate is divided by the number of pixels per byte (8 in the two color mode and 4 in the four color mode) and is added to the result of the vertical computations.
PIXEL	A dot on the graphics screen which may be turned on or off. It will either consist of a single bit for the 2 color mode or a bit pair for the 4 color mode.
PIXEL MASK	A data mask which, if ANDed with a graphic byte from the video screen will leave only the information for one pixel.

GRAPHICS ROUTINES

PUT/GET

PUT and GET graphics have one relatively unexplained option. The G option, as you may know if you have used it much can cause great problems if you don t use it in exactly the right manner.

If the G option is not used, BASIC figures out which byte the GET starts in and stores the entire byte into the array (even if the start point is in the middle of the byte) which means that information that was not actually within the limits of the GET will be stored at the new location when PUT is used. Refer to FIGURE 1A for a better understanding of what happens when the G option is not used.

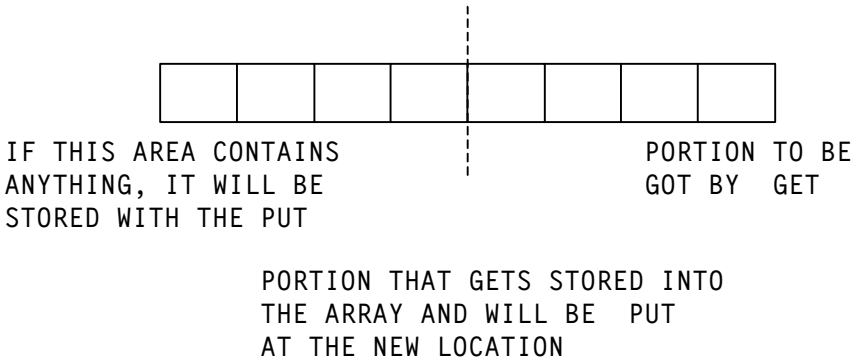


FIGURE 1A

When the G option is used, BASIC figures out which byte the GET starts in, but starts at the exact pixel within that byte and stores the information bit by bit into the array, which means that only the information within the boundaries of the GET area will be stored at the new location when PUT is used. Using the G option makes for a more accurate transfer, but because of the method used to move the information, it is about 1Ø times slower than if the G option were not used. Also, when using the G option you must insure that the array you PUT is exactly the same size as the array you originally used GET on. This is not as critical when the G option is not used. Refer to figure 1B for a better understanding of what happens when the G option is used.

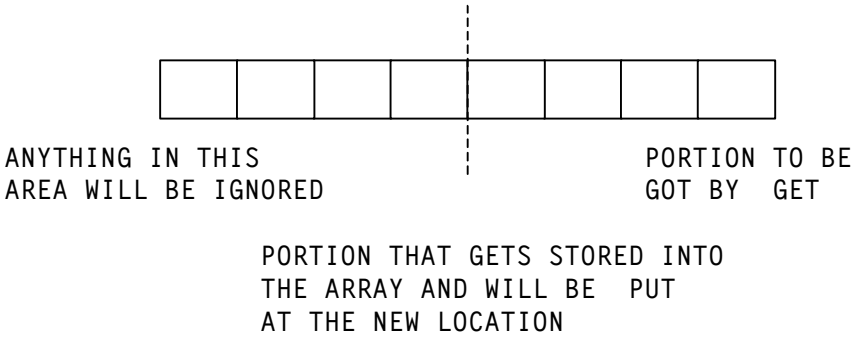


FIGURE 1B

PCLEAR

PCLEAR is used to reserve the number of 1.5K graphics pages that you need. It is also the culprit responsible for the strange behavior of a BASIC program when a PCLEAR statement is encountered within the program in Extended Basic 1.Ø (PCLEAR BUG). This occurs because of the way that PCLEAR works. The BASIC program normally lives immediately after the of the reserved graphics pages, therefore if the number of graphics pages reserved changes, the memory location of the BASIC program must be adjusted to the correct place. If your computer was running under EXTENDED BASIC V 1.Ø strange things could occur when this happened. The BASIC program would get moved as it should, but the pointer in the direct page would not get told that it moved and as such the input line pointer would be pointing to the wrong line after the move. Most of the time this would result in an error of some sort and would require that you simply run the program again, but once in a while strange and bizarre things would occur as the program would somehow manage to continue... in the middle of the wrong line! This bug was corrected in version 1.1 of EXTENDED BASIC by adding the code that would re-adjust the BASIC input pointer necessary for proper operation.

PMODE

PMODE is a routine which sets up the graphics mode and graphics page, it also sets up the background and foreground default colors and stores the number of bytes per horizontal row for the selected mode into the direct page. It is interesting to note that both arguments are not necessary when using this command from BASIC, for example; "PMODE 4" will set the graphics mode but will not alter the viewing page, whereas "PMODE ,2" will leave the graphics mode as is and will only alter the viewing page and "PMODE 4,2" will do both.

SCREEN

SCREEN is the routine that actually turns on the viewing screen. You can use it to select the graphics screen, which was set up by PMODE, or you can use it to elect the text screen; it also allows you to select the colorset. Both arguments are not required when using this command from BASIC, for example "SCREEN1" will just select the graphics screen but will not change the colorset, "SCREEN,1" will change the colorset but will leave the viewing screen as it is. It is interesting to note that anytime a PRINT is executed, the screen and colorset will be reset to default values (text screen, colorset Ø).

PCLS

PCLS is a routine that clears the graphics page starting at BEGGRP and ending at ENDGRP (which were set by PMODE). If no argument is specified, the current background color is used as a default value otherwise the ASCII Ø TO 3 which was parsed from the BASIC program line gets converted to a binary Ø to 3 and is multiplied by $55, this will leave ACCB containing the proper bit pattern for the particular color.

Some very interesting things can be made to happen by altering BEGGRP and/or ENDGRP since these memory locations contain the absolute address of the start and

end of the current graphics page. For example, a partial PCLS of the screen may be accomplished from BASIC by saving the original values of BEGGRP and ENDGRP, altering them, doing a PCLS and restoring the original values. Extreme caution must be exercised when doing something like this, POKEing the wrong values could cause your BASIC program to be erased, and if you forget to restore the original values, the graphics commands will not work properly.

COLOR

COLOR is a routine that sets up the foreground and background colors and stores them in FORCOL and BAKCOL. It is not necessary to specify both arguments when using this from BASIC. For example, "COLOR 1" sets only the foreground color, "COLOR ,1" sets only the background color, and "COLOR 1,1" sets both.

PPOINT

PPOINT is a routine that checks to see if the pixel at the specified X,Y coordinate is a color or turned off. The first thing that this routine does is to parse off the horizontal (X) and the vertical (Y) coordinates from the BASIC program line. These coordinates are then normalized for the current PMODE and converted to an absolute screen address and a pixel mask by a calpos routine. The pixel is then tested to see if it is set to a color (0 to 8, 0 = pixel off) and the result is returned as a floating-point number.

LINE

The LINE routine sort of serves a dual purpose: it is the first step of the LINE INPUT command and also is used to draw graphic lines. As the routine is entered, a check is made to see if the token for INPUT follows the LINE token. If this is the case, the program branches and a LINE INPUT is performed, otherwise the line routine continues. From this point, the LINE routine checks for one of three characters, the "(", the "-" or the "@" symbols and if none of these are found, a syntax error is generated. If one of these symbols is found, the routine parses the start and end coordinates, which are normalized for the current PMODE and placed in HORBEG, VERBEG, HOREND, and VEREND. Next, the B ox and F ill options are looked for and flags are set accordingly. The line is then drawn and appropriate actions are taken depending on the status of the Box and Fill flags. It is interesting to note that the "@" symbol does not do anything! It is there to make the command syntax consistent with the "PRINT @" concept and to make it compatible with other versions of Microsoft BASIC.

PSET/PRESET

PSET is a routine that sets or turns on a single pixel for the current PMODE and is the exact routine used by PRESET. SETFLAG is used to indicate what action to take, if it is set, the routine was called by a PSET and a pixel will be turned on, if it is clear, the routine was called by PRESET and a pixel will be turned off. The main routine takes the specified X,Y coordinates, normalizes them for the current PMODE, calculates the absolute screen address by a calpos routine and performs the appropriate action on the pixel.

DRAW

DRAW is a routine that has the ability to draw lines of a specified length in any one of 8 angles, Ø, 45, 9Ø, 135, 18Ø, 225, 27Ø, and 315 degrees. The directions and lengths are parsed from the BASIC program line and flags are set to indicate which direction (or directions in the case of diagonal lines) that the line will be drawn. What actually happens is this: The X,Y coordinates are figured and normalized for the current PMODE, the absolute screen address is calculated by a calpos routine, and a portion of the PSET routine is called to turn on the pixel. The X,Y coordinates are adjusted in the proper direction and the process is repeated LENGTH number of times. This continues until the end of the DRAW command string. Something not generally known about the DRAW routine is its ability to use variables to indicate parameters like length, color, and scale! There is a certain syntax that must be followed, which to my knowledge has not yet been published anywhere until now. Following is a short example of how to do this.

```
1Ø A=1Ø:B=13
2Ø DRAW "BM=A,A;U=B;R=B;D=B;L=B"
```

The above program will draw a box that has sides equal to the variable B , as the program stands, line 2Ø is equivalent to the following:

```
2Ø DRAW "BM1Ø,1Ø;U13R13D13L13"
```

There are many good possibilities for using variables with DRAW; it s too bad that nobody has outlined how to do it until now.

CIRCLE

CIRCLE, believe it or not, is not really drawn as a circle; it is instead drawn as a 64-sided polygon using a formula and a sine/cosine table to calculate the coordinates before drawing the individual lines. The routine has provisions so that partial circles can be drawn, color can be specified, and height to width ratio can also be specified.

PAINT

PAINT is a routine, which starts at a specified X,Y coordinate and draws horizontal lines until either a border of specified color is encountered or the edge of the screen is reached. The process continues until all borders have been reached. As it PAINTs, the routine keeps track of places where a line of equal length has not encountered a border or a screen edge so that it can paint odd shaped areas.

PCOPY

PCOPY is a routine, which copies a 1.5K block of memory from one graphics page to another. There are a total of 8 graphics pages for use which may be reserved by the PCLEAR command, PCOPY was designed to allow copying from page to page within the reserved area, however due to a little known bug in the routine that checks for this, it is possible to PCOPY to page 5 even if only 4 pages were reserved (PCLEAR4). This can be very hazardous to the health of your BASIC program (remember, your BASIC program starts immediately after the end of the reserved

graphics pages). Imagine what would happen to your program if you were to write the following program:

```
10 PCLEAR 4
20 PCOPY 1 TO 5
```

If all were as should be you would be greeted with an FC error, but unfortunately the routine which should detect such an error does not work properly. Microsoft did not catch the error in time to correct it in the 1.1 revision Extended Basic, but did manage to fix it for the DRAGON computer (a color computer clone from England).

PLAY

PLAY is a routine, which allows you to create complex sounds with much greater efficiency than the SOUND routine. Values are parsed from the BASIC line and are used to set such things as volume, octave, note and duration. These values are used in conjunction with delay routines and a waveform table to create music or sound effects. A little known fact about the PLAY routine is its ability to allow the use of variables within the program line in a way similar to that described in the section about DRAW. In fact PLAY and DRAW both use the same string interpretation routine when variables are involved.

```
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
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
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
0034
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
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 0=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: 0=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 0-3
0286 00EB           DCDRV       RMB   1               *PV DSKCON DRIVE NUMBER 0 3
0287 00EC           DCTRK       RMB   1               *PV DSKCON TRACK NUMBER 0 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 FEED          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          INITØ       RMB   1              INITIALIZATION REGISTER Ø
0715
0716 FF9Ø          BITØ        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          BITØ        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
0814                              640x2         1      0      1      0      0
```

```
                              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
                         BIT4           PAØ, 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
0970                  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
0980                                  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                                  X0000 - X1FFF   Ø      Ø      Ø      Ø      FFAØ
0987                                  X2000 - X3FFF   Ø      Ø      Ø      1      FFA1
0988                                  X4000 - X5FFF   Ø      Ø      1      Ø      FFA2
0989                                  X6000 - X7FFF   Ø      Ø      1      1      FFA3
0990                                  X8000 - X9FFF   Ø      1      Ø      Ø      FFA4
0991                                  XA000 - XBFFF   Ø      1      Ø      1      FFA5
0992                                  XC000 - XDFFF   Ø      1      1      Ø      FFA6
0993                                  XE000 - XFFFF   Ø      1      1      1      FFA7
0994
0995                                  X0000 - X1FFF   1      Ø      Ø      Ø      FFA8
0996                                  X2000 - X3FFF   1      Ø      Ø      1      FFA9
0997                                  X4000 - X5FFF   1      Ø      1      Ø      FFAA
0998                                  X6000 - X7FFF   1      Ø      1      1      FFAB
0999                                  X8000 - X9FFF   1      1      Ø      Ø      FFAC
1000                                  XA000 - XBFFF   1      1      Ø      1      FFAD
1001                                  XC000 - XDFFF   1      1      1      Ø      FFAE
1002                                  XE000 - XFFFF   1      1      1      1      FFAF
1003
1004 FFBØ          PALETREG  RMB   16              COLOR PALETTE REGISTERS (6 BITS)
1005
1006                                  DATA BIT        D5     D4     D3     D2     D1     DØ
1007                                  RGB OUTPUT      R1     G1     B1     RØ     GØ     BØ
1008                                  COMP. OUTPUT    I1     IØ     P3     P2     P1     PØ
1009
1010                * FOR COCO COMPATIBILITY, THE FOLLOWING SHOULD BE LOADED ON INITIALIZATION
1011                  (RGB VALUES) FOR PAL VERSION, IGNORE TABLE FOR COMPOSITE
1012                          FFBØ  GREEN           $12
1013                          FFB1  YELLOW          $36
1014                          FFB2  BLUE            $Ø9
1015                          FFB3  RED             $24
1016                          FFB4  BUFF            $3F
1017                          FFB5  CYAN            $1Ø
1018                          FFB6  MAGENTA         $2D
1019                          FFB7  ORANGE          $26
1020                          FFB8  BLACK           $ØØ
1021                          FFB9  GREEN           $12
1022                          FFBA  BLACK           $ØØ
1023                          FFBB  BUFF            $3F
1024                          FFBC  BLACK           $ØØ
1025                          FFBD  GREEN           $12
1026                          FFBE  BLACK           $ØØ
1027                          FFBF  ORANGE          $26
1028
1029 FFCØ          SAMREG    EQU   *               SAM CONTROL REGISTERS
1030
1031 FFCØ          VØCLR     RMB   1               CLEAR COCO GRAPHICS MODE VØ
1032 FFC1          VØSET     RMB   1               SET COCO GRAPHICS MODE VØ
1033 FFC2          V1CLR     RMB   1               CLEAR COCO GRAPHICS MODE V1
1034 FFC3          V1SET     RMB   1               SET COCO GRAPHICS MODE V1
1035 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   8000                        ORG   $8000
0002   8000 45 58           EXBAS  FCC   'EX'
0003                        *
0004                        * MOVE EXTENDED BASIC'S COMMAND INTERPRETATION TABLE FROM ROM TO RAM
0005   8002 8E 80 DE        L8002  LDX   #L80DE                    ROM ADDRESS
0006   8005 CE 01 2A               LDU   #COMVEC+10                RAM ADDRESS
0007   8008 C6 0A                  LDB   #10                       10 BYTES TO MOVE
0008   800A BD A5 9A               JSR   LA59A                     MOVE B BYTES FROM (X) TO (U)
0009   800D 8E B2 77               LDX   #LB277                    ADDRESS OF SYNTAX ERROR
0010   8010 AF 43                  STX   $03,U                     PUT SYNTAX ERROR IN ADDRESS OF DISK
0011   8012 AF 48                  STX   $08,U                     BASIC S COMMAND INTERPRETATION LOOPS
0012   8014 8E 89 4C               LDX   #XIRQSV                   *PUT EXBASIC S IRQ SERVICING ROUTINE
0013   8017 BF 01 0D               STX   IRQVEC+1                  *ADDRESS IN THE IRQ VECTOR
0014   801A 9E 8A                  LDX   ZERO                      GET X=0
0015   801C BF 01 12               STX   TIMVAL                    INITIALIZE TIMER = 0
0016   801F BD 82 9C               JSR   XVEC18                    INITIALIZE A BUNCH OF VARIABLES
0017   8022 CC 2C 05               LDD   #$2C05                    *INITIALIZE DLOAD TO 1200 BAUD AND
0018   8025 DD E6                  STD   DLBAUD                    *TIMEOUT CONSTANT TO 5
0019   8027 8E 01 3E               LDX   #USR0                     =INITIALIZE ADDRESS OF START OF
0020   802A 9F B0                  STX   USRADR                    =USR JUMP TABLE
0021                        * INITIALIZE THE USR CALLS TO  FC ERROR
0022   802C CE B4 4A               LDU   #LB44A                    ADDRESS OF  FC ERROR  ROUTINE
0023   802F C6 0A                  LDB   #10                       10 USR CALLS IN EX BASIC
0024   8031 EF 81           L8031  STU   ,X++                      STORE  FC  ERROR AT USR ADDRESSES
0025   8033 5A                     DECB                            FINISHED ALL 10?
0026   8034 26 FB                  BNE   L8031                     NO
0027
0028                        * MODIFY THE RAM HOOKS FOR THE NEW ROUTINES CONTAINED IN EXT BASIC
0029   8036 86 7E                  LDA   #$7E                      OP CODE OF  JMP
0030   8038 B7 01 9A               STA   RVEC20                    =
0031   803B 8E 82 B9               LDX   #XVEC20                   =
0032   803E BF 01 9B               STX   RVEC20+1                  = COMMAND INTERPRETATION LOOP
0033   8041 B7 01 8B               STA   RVEC15                    *
0034   8044 8E 88 46               LDX   #XVEC15                   *
0035   8047 BF 01 8C               STX   RVEC15+1                  * EXPRESSION EVALUATION
0036   804A B7 01 97               STA   RVEC19                    =
0037   804D 8E 87 E5               LDX   #XVEC19                   =
0038   8050 BF 01 98               STX   RVEC19+1                  = ASCII TO FLOATING POINT CONVERSION
0039   8053 B7 01 79               STA   RVEC9                     *
0040   8056 8E 8E 90               LDX   #XVEC9                    *
0041   8059 BF 01 7A               STX   RVEC9+1                   * PRINT
0042   805C B7 01 91               STA   RVEC17                    =
0043   805F 8E 88 F0               LDX   #XVEC17                   =
0044   8062 BF 01 92               STX   RVEC17+1                  = ERROR DRIVER
0045   8065 B7 01 6A               STA   RVEC4                     *
0046   8068 8E 8C F1               LDX   #XVEC4                    *
0047   806B BF 01 6B               STX   RVEC4+1                   * CONSOLE IN
0048   806E B7 01 67               STA   RVEC3                     =
0049   8071 8E 82 73               LDX   #XVEC3                    =
0050   8074 BF 01 68               STX   RVEC3+1                   = CONSOLE OUT
0051   8077 B7 01 76               STA   RVEC8                     *
0052   807A 8E 82 86               LDX   #XVEC8                    *
0053   807D BF 01 77               STX   RVEC8+1                   * CLOSE A FILE
0054   8080 B7 01 A3               STA   RVEC23                    =
0055   8083 8E 83 04               LDX   #XVEC23                   =
0056   8086 BF 01 A4               STX   RVEC23+1                  = CRUNCH A BASIC LINE
0057   8089 B7 01 94               STA   RVEC18                    *
0058   808C 8E 82 9C               LDX   #XVEC18                   *
0059   808F BF 01 95               STX   RVEC18+1                  * RUN
0060   8092 B7 01 1D               STA   EXPJMP                    STORE OP CODE OF JMP
0061   8095 8E 84 89               LDX   #L8489                    GET EXPONENTIATION ADDRESS
0062   8098 BF 01 1E               STX   EXPJMP+1                  SAVE IT
0063   809B BD 96 E6               JSR   L96E6                     GO INITIALIZE EXBAS GRAPHICS VARIABLES
0064   809E B6 FF 03               LDA   PIA0+3                    * ENABLE PIA0 TO
0065   80A1 8A 01                  ORA   #$01                      * PASS 60HZ
0066   80A3 B7 FF 03               STA   PIA0+3                    * INTERRUPT TO MPU
0067   80A6 8E 44 4B               LDX   #'DK'                     FIRST TWO BYTES OF DISK ROM
0068   80A9 BC C0 00               CMPX  DOSBAS                    COMPARE TO DISK ROM ADDRESS
0069   80AC 10 27 3F 52            LBEQ  DOSBAS+2                  BRANCH IF DISK BASIC EXISTS
0070   80B0 1C AF                  ANDCC #$AF                      ENABLE INTERRUPTS
0071   80B2 8E 80 E7               LDX   #L80E6+1                  POINT TO SIGN ON MESSAGE
0072   80B5 BD B9 9C               JSR   STRINOUT                  DISPLAY IT
0073   80B8 8E 80 C0               LDX   #XBWMST                   GET EXBAS WARM START (RESET) VECTOR
0074   80BB 9F 72                  STX   RSTVEC                    SAVE IT
0075   80BD 7E A0 E2               JMP   LA0E2                     SET WARM START FLAG, ENTER BASIC
0076                        * EXBAS WARM START ENTRY POINT
0077   80C0 12              XBWMST NOP                             WARM START ENABLE
0078   80C1 0F E3                  CLR   PLYTMR                    =
0079   80C3 0F E4                  CLR   PLYTMR+1                  = CLEAR PLAY TIMER
0080   80C5 B6 FF 03               LDA   PIA0+3                    * ENABLE PIA0 TO
0081   80C8 8A 01                  ORA   #$01                      * PASS 60HZ
0082   80CA B7 FF 03               STA   PIA0+3                    * INTERRUPT TO MPU
0083   80CD 7E A0 E8               JMP   BAWMST                    JUMP TO BASIC S WARM START
0084                        *
0085                        * THIS CODE IS A PATCH TO FIX THE PCLEAR BUG
0086   80D0 96 68           L80D0  LDA   CURLIN                    GET THE CURRENT LINE NUMBER
0087   80D2 4C                     INCA                            TEST FOR DIRECT MODE
0088   80D3 27 08                  BEQ   L80DD                     RETURN IF DIRECT MODE
0089   80D5 1F 20                  TFR   Y,D                       SAVE OFFSET IN ACCD
0090   80D7 93 19                  SUBD  TXTTAB                    SUBTRACT OUT START OF BASIC
0091   80D9 D3 A6                  ADDD  CHARAD                    ADD THE CURRENT BASIC INPUT POINTER
0092   80DB DD A6                  STD   CHARAD                    SAVE NEW BASIC INPUT POINTER
0093   80DD 39              L80DD  RTS
0094
0095   80DE 19              L80DE  FCB   25                         25 EXBAS COMMANDS
0096   80DF 81 83           L80DF  FDB   L8183                      EXBAS RESERVED WORD DICTIONARY TABLE
```

```
0097   80E1 81 3C             L80E1   FDB   L813C                          EXBAS RESERVED WORD HANDLER
0098   80E3 0E                L80E3   FCB   14                             14 EXBAS SECONDARY COMMANDS
0099   80E4 82 1E             L80E4   FDB   L821E                          EXBAS SECONDARY RESERVED WORD TABLE
0100   80E6 81 68             L80E6   FDB   L8168                          EXBAS SECONDARY RESERVED WORD HANDLER
0101
0102   80E8 45 58 54 45 4E 44 L80E8   FCC   'EXTENDED COLOR BASIC 1.1'
0103   80EE 45 44 20 43 4F 4C
0104   80F4 4F 52 20 42 41 53
0105   80FA 49 43 20 31 2E 31
0106   8100 0D                L8100   FCB   CR
0107   8101 43 4F 50 59 52 49 L8101   FCC   'COPYRIGHT (C) 1982'
0108   8107 47 48 54 20 28 43
0109   810D 29 20 31 39 38 32
0110   8113 20 42 59 20 54 41         ' BY TANDY'
0111   8119 4E 44 59
0112   811C 0D                L811C   FCB   CR
0113   811D 55 4E 44 45 52 20 L811D   FCC   'UNDER LICENSE FROM MICROSOFT'
0114   8123 4C 49 43 45 4E 53
0115   8129 45 20 46 52 4F 4D
0116   812F 20 4D 49 43 52 4F
0117   8135 53 4F 46 54
0118   8139 0D 0D 00          L8139   FCB   CR,CR,0
0119
0120                          * EXBAS COMMAND INTERPRETATION LOOP
0121   813C 81 CB             L813C   CMPA  #$CB                          $CB IS LARGEST EX BASIC COMMAND TOKEN
0122   813E 22 08                     BHI   L8148                          BRANCH IF > LARGEST TOKEN
0123   8140 8E 81 F0                  LDX   #L81F0                         POINT X TO EXBAS DISPATCH TABLE FOR COMMANDS
0124   8143 80 B5                     SUBA  #$B5                           $B5 IS SMALLEST EXBAS TOKEN
0125   8145 7E AD D4                  JMP   LADD4                          INTERPRET BASIC TOKEN HANDLER
0126   8148 81 FF             L8148   CMPA  #$FF                          CHECK FOR SECONDARY TOKEN
0127   814A 27 08                     BEQ   L8154                          BRANCH IF IT IS SECONDARY
0128   814C 81 CD                     CMPA  #$CD                           LARGEST EXBAS TOKEN
0129   814E 23 15                     BLS   L8165                           SYNTAX  ERROR FOR USING & FN
0130   8150 6E 9F 01 37               JMP   [COMVEC+23]                    GO TO DISK BASIC RESERVED WORD HANDLER
0131   8154 9D 9F             L8154   JSR   GETNCH                         GET AN INPUT CHARACTER FROM BASIC
0132   8156 81 90                     CMPA  #$90                           TOKEN FOR MID$
0133   8158 10 27 05 7A               LBEQ  L86D6                          BRANCH IF MID$
0134   815C 81 9F                     CMPA  #$9F                           TOKEN FOR TIMER
0135   815E 10 27 07 FE               LBEQ  L8960                          BRANCH IF TIMER
0136   8162 BD 01 A0                  JSR   RVEC22                         HOOK INTO RAM
0137   8165 7E B2 77          L8165   JMP   LB277                           SYNTAX  ERROR
0138
0139                          * EXBAS SECONDARY COMMAND HANDLER
0140   8168 C1 42             L8168   CMPB  #2*33                          *80+33 IS LARGEST EXBAS SECONDARY COMMAND
0141   816A 23 04                     BLS   L8170                          BRANCH IF LEGITIMATE EXBAS SECONDARY TOKEN
0142   816C 6E 9F 01 3C               JMP   [COMVEC+28]                    GO TO DISK BASIC SECONDARY COMMAND HANDLER
0143   8170 C0 28             L8170   SUBB  #2*20                          SUBTRACT OUT 20 BASIC SECONDARY COMMANDS
0144   8172 C1 10                     CMPB  #2*8                           HEX$ TOKEN
0145   8174 22 07                     BHI   L817D                          BRANCH IF > HEX$
0146   8176 34 04                     PSHS  B                              SAVE TOKEN OFFSET
0147   8178 BD B2 62                  JSR   LB262                          EVALUATE EXPRESSION IN PARENTHESES
0148   817B 35 04                     PULS  B                              GET TOKEN OFFSET BACK
0149   817D 8E 82 57          L817D   LDX   #L8257                         EXBAS SECONDARY COMMAND JUMP TABLE
0150   8180 7E B2 CE                  JMP   LB2CE                          JUMP TO SECONDARY FUNCT1ON HANDLER
0151
0152                          * RESERVED WORD TABLE FOR EXTENDED BASIC
0153                          *                                            TOKEN #
0154   8183 44 45 CC          L8183   FCC   'DE',$80+'L'                   B5
0155   8186 45 44 49 D4       L8186   FCC   'EDI',$80+'T'                  B6
0156   818A 54 52 4F CE       L818A   FCC   'TRO',$80+'N'                  B7
0157   818E 54 52 4F 46 C6    L818E   FCC   'TROF',$80+'F'                 B8
0158   8193 44 45 C6          L8193   FCC   'DE',$80+'F'                   B9
0159   8196 4C 45 D4          L8196   FCC   'LE',$80+'T'                   BA
0160   8199 4C 49 4E C5       L8199   FCC   'LIN',$80+'E'                  BB
0161   819D 50 43 4C D3       L819D   FCC   'PCL',$80+'S'                  BC
0162   81A1 50 53 45 D4       L81A1   FCC   'PSE',$80+'T'                  BD
0163   81A5 50 52 45 53 45 D4 L81A5   FCC   'PRESE',$80+'T'                BE
0164   81AB 53 43 52 45 45 CE L81AB   FCC   'SCREE',$80+'N'                BF
0165   81B1 50 43 4C 45 41 D2 L81B1   FCC   'PCLEA',$80+'R'                C0
0166   81B7 43 4F 4C 4F D2    L81B7   FCC   'COLO',$80+'R'                 C1
0167   81BC 43 49 52 43 4C C5 L81BC   FCC   'CIRCL',$80+'E'                C2
0168   81C2 50 41 49 4E D4    L81C2   FCC   'PAIN',$80+'T'                 C3
0169   81C7 47 45 D4          L81C7   FCC   'GE',$80+'T'                   C4
0170   81CA 50 55 D4          L81CA   FCC   'PU',$80+'T'                   C5
0171   81CD 44 52 41 D7       L81CD   FCC   'DRA',$80+'W'                  C6
0172   81D1 50 43 4F 50 D9    L81D1   FCC   'PCOP',$80+'Y'                 C7
0173   81D6 50 4D 4F 44 C5    L81D6   FCC   'PMOD',$80+'E'                 C8
0174   81DB 50 4C 41 D9       L81DB   FCC   'PLA',$80+'Y'                  C9
0175   81DF 44 4C 4F 41 C4    L81DF   FCC   'DLOA',$80+'D'                 CA
0176   81E4 52 45 4E 55 CD    L81E4   FCC   'RENU',$80+'M'                 CB
0177   81E9 46 CE             L81E9   FCC   'F',$80+'N'                    CC
0178   81EB 55 53 49 4E C7    L81EB   FCC   'USIN',$80+'G'                 CD
0179
0180                          * DISPATCH TABLE FOR EXTENDED BASIC COMMANDS
0181                          *                                                  TOKEN #
0182   81F0 89 70             L81F0   FDB   DEL                            DEL     B5
0183   81F2 85 33             L81F2   FDB   EDIT                           EDIT    B6
0184   81F4 86 A7             L81F4   FDB   TRON                           TRON    B7
0185   81F6 86 A8             L81F6   FDB   TROFF                          TROFF   B8
0186   81F8 88 71             L81F8   FDB   DEF                            DEF     B9
0187   81FA AF 89             L81FA   FDB   LET                            LET     BA
0188   81FC 93 BB             L81FC   FDB   LINE                           LINE    BB
0189   81FE 95 32             L81FE   FDB   PCLS                           PCLS    BC
0190   8200 93 61             L8200   FDB   PSET                           PSET    BD
0191   8202 93 65             L8202   FDB   PRESET                         PRESET  BE
0192   8204 96 70             L8204   FDB   SCREEN                         SCREEN  BF
```

```
0193   8206 96 8B             L8206    FDB   PCLEAR                          PCLEAR  C0
0194   8208 95 46             L8208    FDB   COLOR                           COLOR   C1
0195   820A 9E 9D             L820A    FDB   CIRCLE                          CIRCLE  C2
0196   820C 98 EC             L820C    FDB   PAINT                           PAINT   C3
0197   820E 97 55             L820E    FDB   GET                             GET     C4
0198   8210 97 58             L8210    FDB   PUT                             PUT     C5
0199   8212 9C B6             L8212    FDB   DRAW                            DRAW    C6
0200   8214 97 23             L8214    FDB   PCOPY                           PCOPY   C7
0201   8216 96 21             L8216    FDB   PMOD                            PMODE   C7
0202   8218 9A 22             L8218    FDB   PLAY                            PLAY    C9
0203   821A 8C 18             L821A    FDB   DLOAD                           DLOAD   CA
0204   821C 8A 09             L821C    FDB   RENUM                           RENUM   CB
0205
0206                          * SECONDARY FUNCTION FOR EXTENDED BASIC
0207                          * TOKENS ARE PRECEEDED WITH AN $FF BYTE
0208                          *                                               TOKEN #
0209   821E 41 54 CE          L821E    FCC   'AT',$80+'N'                    94
0210   8221 43 4F D3          L8221    FCC   'CO',$80+'S'                    95
0211   8224 54 41 CE          L8224    FCC   'TA',$80+'N'                    96
0212   8227 45 58 D0          L8227    FCC   'EX',$80+'P'                    97
0213   822A 46 49 D8          L822A    FCC   'FI',$80+'X'                    98
0214   822D 4C 4F C7          L822D    FCC   'LO',$80+'G'                    99
0215   8230 50 4F D3          L8230    FCC   'PO',$80+'S'                    9A
0216   8233 53 51 D2          L8233    FCC   'SQ',$80+'R'                    9B
0217   8236 48 45 58 A4       L8236    FCC   'HEX',$80+'$'                   9C
0218   823A 56 41 52 50 54 D2 L823A    FCC   'VARPT',$80+'R'                 9D
0219   8240 49 4E 53 54 D2    L8240    FCC   'INST',$80+'R'                  9E
0220   8245 54 49 4D 45 D2    L8245    FCC   'TIME',$80+'R'                  9F
0221   824A 50 50 4F 49 4E D4 L824A    FCC   'PPOIN',$80+'T'                 A0
0222   8250 53 54 52 49 4E 47 L8250    FCC   'STRING',$80+'$'                A1
0223   8256 A4
0224                          * JUMP TABLE FOR EXTENDED BASIC SECONDARY FUNCTIONS
0225                          *                                                     TOKEN #
0226   8257 83 B0             L8257    FDB   ATN                             ATN     94
0227   8259 83 78             L8259    FDB   COS                             COS     95
0228   825B 83 81             L825B    FDB   TAN                             TAN     96
0229   825D 84 F2             L825D    FDB   EXP                             EXP     97
0230   825F 85 24             L825F    FDB   FIX                             FIX     98
0231   8261 84 46             L8261    FDB   LOG                             LOG     99
0232   8263 86 AC             L8263    FDB   POS                             POS     9A
0233   8265 84 80             L8265    FDB   SQR                             SQR     9B
0234   8267 8B DD             L8267    FDB   HEXDOL                          HEXDOL  9C
0235   8269 86 BE             L8269    FDB   VARPTR                          VARPT   9D
0236   826B 87 7E             L826B    FDB   INSTR                           INSTR   9E
0237   826D 89 68             L826D    FDB   TIMER                           TIMER   9F
0238   826F 93 39             L826F    FDB   PPOINT                          PPOINT  A0
0239   8271 87 4E             L8271    FDB   STRING                          STRING  A1
0240
0241                          * CONSOLE OUT RAM HOOK
0242   8273 0D 6F             XVEC3    TST   DEVNUM                          CHECK DEVICE NUMBER
0243   8275 10 27 13 33               LBEQ  L95AC                           BRANCH IF SCREEN
0244   8279 34 04                     PSHS  B                               SAVE CHARACTER
0245   827B D6 6F                     LDB   DEVNUM                          *GET DEVICE NUMBER AND
0246   827D C1 FD                     CMPB  #-3                             *CHECK FOR DLOAD
0247   827F 35 04                     PULS  B                               GET CHARACTER BACK
0248   8281 26 02                     BNE   L8285                           RETURN IF NOT DLOAD
0249   8283 32 62                     LEAS  $02,S                           *TAKE RETURN OFF STACK & GO BACK TO ROUTINE
0250                          *                                             *THAT CALLED CONSOLE OUT
0251   8285 39                L8285    RTS
0252
0253                          * CLOSE FILES RAM HOOK - THIS CODE CORRECTS A
0254                          * BUG IN 1.0 BASIC WHICH WAS FIXED BY 1.1 BASIC
0255   8286 96 6F             XVEC8    LDA   DEVNUM                          GET DEVICE NUMBER
0256   8288 4C                         INCA                                  CHECK FOR CASSETTE
0257   8289 26 FA                     BNE   L8285                           RETURN IF NOT CASSETTE
0258   828B 96 78                     LDA   FILSTA                          GET FILE STATUS
0259   828D 81 02                     CMPA  #$02                            OPEN FOR OUTPUT?
0260   828F 26 F4                     BNE   L8285                           RETURN IF NOT OPEN FOR OUTPUT
0261   8291 96 79                     LDA   CINCTR                          GET CHARACTER BUFFER COUNTER
0262   8293 26 F0                     BNE   L8285                           RETURN IF NOT EMPTY
0263   8295 0F 6F                     CLR   DEVNUM                          SET DEVICE NUMBER TO SCREEN
0264   8297 32 62                     LEAS  $02,S                           GET RETURN ADDRESS OFF OF STACK
0265   8299 7E A4 44                  JMP   LA444                           WRITE END OF FILE TAPE BLOCK
0266                          * RUN RAM HOOK
0267   829C CC BA 42          XVEC18   LDD   #$BA42                         MID HIGH VALUE + MID LOW VALUE
0268   829F DD DF                     STD   VOLHI                           INITIALIZE PLAY VOLUME
0269   82A1 86 02                     LDA   #$02
0270   82A3 97 E2                     STA   TEMPO                           INITIALIZE TEMPO TO 2
0271   82A5 97 DE                     STA   OCTAVE                          INITIALIZE OCTAVE TO 3
0272   82A7 48                        ASLA                                  X2
0273   82A8 97 E1                     STA   NOTELN                          INITIALIZE NOTE LENGTH TO 5
0274   82AA 0F E5                     CLR   DOTVAL                          LEAR NOTE TIMER SCALE FACTOR
0275   82AC DC 8A                     LDD   ZERO                            ZERO ACCD
0276   82AE DD E8                     STD   ANGLE                           INITIALIZE DRAW ANGLE AND SCALE TO 1
0277   82B0 C6 80                     LDB   #128                            * INITIALIZE HORIZONTAL DEFAULT
0278   82B2 DD C7                     STD   HORDEF                          * COORDINATE TO MID POSITION
0279   82B4 C6 60                     LDB   #96                             = INITIALIZE VERTICAL DEFAULT
0280   82B6 DD C9                     STD   VERDEF                          = COORDINATE TO MID POSITION
0281   82B8 39                        RTS
0282                          * COMMAND INTERPRETATION LOOP RAM HOOK
0283   82B9 32 62             XVEC20   LEAS  $02,S                           PURGE RETURN ADDRESS FROM STACK
0284   82BB 1C AF             L82BB    ANDCC #$AF                            ENABLE INTERRUPTS
0285   82BD BD AD EB                   JSR   LADEB                           CHECK FOR KEYBOARD BREAK
0286   82C0 9E A6                     LDX   CHARAD                          * GET CURRENT BASIC LINE
0287   82C2 9F 2F                     STX   TINPTR                          * POINTER AND SAVE IT
0288   82C4 A6 80                     LDA   ,X+                             GET CURRENT INPUT CHARACTER AND ADVANCE POINTER
```

```
0289   82C6 27 07              BEQ   L82CF                    BRANCH IF END OF LINE
0290   82C8 81 3A              CMPA  #':'                     CHECK FOR COLON
0291   82CA 27 25              BEQ   L82F1                    CONTINUE INTERPRETING IF COLON
0292   82CC 7E B2 77           JMP   LB277                    SNYTAX ERROR - COLON ONLY LEGAL LINE SEPARATOR
0293   82CF A6 81        L82CF LDA   ,X++                     * GET 1ST BYTE OF ADDRESS OF NEXT
0294   82D1 97 00              STA   ENDFLG                   * BASIC LINE AND SAVE IT
0295   82D3 26 03              BNE   L82D8                    BRANCH IF NOT END OF PROGRAM
0296   82D5 7E AE 15           JMP   LAE15                    RETURN TO DIRECT MODE - PRINT OK
0297   82D8 EC 80        L82D8 LDD   ,X+                      GET LINE NUMBER OF NEXT LINE
0298   82DA DD 68              STD   CURLIN                   SAVE LINE NUMBER
0299   82DC 9F A6              STX   CHARAD                   SAVE ADDRESS NEXT BYTE TO INTERPRET
0300   82DE 96 AF              LDA   TRCFLG                   TEST THE TRACE FLAG
0301   82E0 27 0F              BEQ   L82F1                    BRANCH IF TRACE OFF
0302   82E2 86 5B              LDA   #$5B                     <LEFT HAND MARKER FOR TRON LINE NUMBER
0303   82E4 BD A2 82           JSR   LA282                    OUTPUT A CHARACTER
0304   82E7 96 68              LDA   CURLIN                   GET MS BYTE OF LINE NUMBER
0305   82E9 BD BD CC           JSR   LBDCC                    CONVERT ACCD TO DECIMAL AND PRINT ON SCREEN
0306   82EC 86 5D              LDA   #$5D                     > RIGHT HAND MARKER FOR TRON LINE NUMBER
0307   82EE BD A2 82           JSR   LA282                    OUTPUT A CHARACTER
0308   82F1 9D 9F        L82F1 JSR   GETNCH                   GET A CHARACTER FROM BASIC
0309   82F3 1F A9              TFR   CC,B                     SAVE STATUS IN ACCB
0310   82F5 81 98              CMPA  #$98                     CSAVE TOKEN
0311   82F7 27 1D              BEQ   L8316                    GO DO A CSAVE
0312   82F9 81 97              CMPA  #$97                     CLOAD TOKEN
0313   82FB 27 14              BEQ   L8311                    PROCESS CLOAD
0314   82FD 1F 9A              TFR   B,CC                     GET STATUS REG BACK
0315   82FF BD AD C6           JSR   LADC6                    LINK BACK TO BASIC S INTERPRETATION LOOP
0316   8302 20 B7              BRA   L82BB                    GO TO MAIN INTERPRETATION LOOP
0317                     * CRUNCH RAM HOOK
0318   8304 AE 62        XVEC23 LDX   $02,S                   *CHECK TO SEE IF THE ROUTINE CALLING  CRUNCH
0319   8306 8C AC 9D           CMPX  #LAC9D                   *IS COMING FROM THE MAIN LOOP IN BASIC
0320   8309 26 05              BNE   L8310                    *AND BRANCH IF NOT
0321   830B 8E 82 F1           LDX   #L82F1                   =IF IT IS, DO NOT RETURN TO COLOR BASIC
0322   830E AF 62              STX   $02,S                    =BUT TO THE EXBAS PATCH INSTEAD
0323   8310 39           L8310 RTS
0324   8311 BD 8C 62     L8311 JSR   L8C62                    CHECK EXBAS CLOAD HANDLER
0325   8314 20 A5              BRA   L82BB                    GO TO MAIN INTERPRETATION LOOP
0326   8316 8D 02        L8316 BSR   L831A                    DO A CSAVE
0327   8318 20 A1              BRA   L82BB                    GO TO MAIN INTERPRETATION LOOP
0328   831A 9D 9F        L831A JSR   GETNCH                   GET A CHAR FROM BASIC
0329   831C 81 4D              CMPA  #'M'                     CHECK FOR CSAVEM
0330   831E 10 26 21 2A        LBNE  LA44C                    BRANCH IF IT S NOT CSAVEM
0331                     * CSAVEM
0332   8322 9D 9F              JSR   GETNCH                   GET A CHAR FROM BASIC
0333   8324 BD A5 78           JSR   LA578                    GET NAME OF FILE FROM BASIC
0334   8327 8D 43              BSR   L836C                    GO GET THE START ADDRESS
0335   8329 BF 01 E7           STX   CASBUF+13                PUT IT IN HEADER BUFFER
0336   832C 8D 3E              BSR   L836C                    GO GET END ADDRESS
0337   832E AC 62              CMPX  $02,S                    COMPARE TO START ADDRESS
0338   8330 10 25 31 16        LBCS  LB44A                     FC  ERROR IF START > END
0339   8334 8D 36              BSR   L836C                    GO GET XFER ADDRESS
0340   8336 BF 01 E5           STX   CASBUF+11                PUT IT IN HEADER BUFFER
0341   8339 9D A5              JSR   GETCCH                   GET NEW CHARACTER
0342   833B 26 D3              BNE   L8310                    RETURN IF NOT END OF LINE
0343   833D 86 02              LDA   #$02                     FILE TYPE (MACHINE LANGUAGE)
0344   833F 9E 8A              LDX   ZERO                     X = 0000 FILE MODE AND ASCII FLAG
0345   8341 BD A6 5F           JSR   LA65F                    WRITE HEADER BLOCK
0346   8344 0F 78              CLR   FILSTA                   CLOSE CASSETTE FILES
0347   8346 0C 7C              INC   BLKTYP                   BLOCK TYPE = 1
0348   8348 BD A7 D8           JSR   LA7D8                    GO WRITE LEADER
0349   834B AE 64              LDX   $04,S                    GET STARTING ADDRESS
0350   834D 9F 7E        L834D STX   CBUFAD                   STORE BUFFER START ADOR
0351   834F 86 FF              LDA   #255                     BLOCK SIZE = 2SS
0352   8351 97 7D              STA   BLKLEN                   STORE IN BLOCK SIZE
0353   8353 EC 62              LDD   $02,S                    GET ENDING ADDRESS
0354   8355 93 7E              SUBD  CBUFAD                   SUBTRACT START ADDRESS
0355   8357 24 05              BCC   L835E                    BRANCH IF MORE TO BE WRITTEN
0356   8359 32 66              LEAS  $06,S                    CLEAN UP STACK
0357   835B 7E A4 91           JMP   LA491                    WRITE FINAL BLOCK
0358   835E 10 83 00 FF  L835E CMPD  #$00FF                   AT LEAST 1 FULL BLK LEFT?
0359   8362 24 03              BCC   L8367                    YES
0360   8364 5C                 INCB                           NO - PUT WHAT S LEFT IN BLKLEN
0361   8365 D7 7D              STB   BLKLEN                   BUFFER SIZE
0362   8367 BD A7 F4     L8367 JSR   LA7F4                    WRITE A BLOCK
0363   836A 20 E1              BRA   L834D                    GO DO SOME MORE
0364   836C BD B2 6D     L836C JSR   SYNCOMMA                 SYNTAX CHECK FOR COMMA
0365   836F BD B7 3D           JSR   LB73D                    EVAL EXPR - RETURN VALUE IN X
0366   8372 EE E4              LDU   ,S                       SAVE RETURN ADDRESS IN U
0367   8374 AF E4              STX   ,S                       PUT THE EXPRESSION ON THE STACK
0368   8376 1F 35              TFR   U,PC                      RETURN  TO CALLING ADDRESS
0369
0370                     * COS
0371                     * THE VALUE OF COS(X) IS DETERMINED BY THE TRIG IDENTITY COS(X)=SIN((PI/2)+X)
0372   8378 8E 83 AB     COS   LDX   #L83AB                   POINT X TO FP CONSTANT (P1/2)
0373   837B BD B9 C2           JSR   LB9C2                    ADD FPA0 TO (X)
0374   837E 7E BF 78     L837E JMP   LBF78                    JUMP TO SIN ROUTINE
0375
0376                     * TAN
0377                     * THE VALUE OF TAN(X) IS DETERMINED BY THE TRIG IDENTITY TAN(X)=SIN(X)/COS(X)
0378   8381 BD BC 2F     TAN   JSR   LBC2F                    PACK FPA0 AND MOVE IT TO FPA3
0379   8384 0F 0A              CLR   RELFLG                   RESET QUADRANT FLAG
0380   8386 8D F6              BSR   L837E                    CALCULATE SIN OF ARGUMENT
0381   8388 8E 00 4A           LDX   #V4A                     POINT X TO FPA5
0382   838B BD BC 35           JSR   LBC35                    PACK FPA0 AND MOVE IT TO FPA5
0383   838E 8E 00 40           LDX   #V40                     POINT X TO FPA3
0384   8391 BD BC 14           JSR   LBC14                    MOVE FPA3 TO FPA0
```

```
0385   8394 0F 54                  CLR   FP0SGN                        FORCE FPA0 MANTISSA TO BE POSITIVE
0386   8396 96 0A                  LDA   RELFLG                        GET THE QUADRANT FLAG - COS NEGATIVE IN QUADS 2,3
0387   8398 8D 0C                  BSR   L83A6                         CALCULATE VALUE OF COS(FPA0)
0388   839A 0D 4F                  TST   FP0EXP                        CHECK EXPONENT OF FPA0
0389   839C 10 27 36 F2            LBEQ  LBA92                          OV  ERROR IF COS(X)=0
0390   83A0 8E 00 4A               LDX   #V4A                          POINT X TO FPA5
0391   83A3 7E BB 8F       L83A3   JMP   LBB8F                         DIVIDE (X) BY FPA0 - SIN(X)/COS(X)
0392   83A6 34 02          L83A6   PSHS  A                             SAVE SIGN FLAG ON STACK
0393   83A8 7E BF A6               JMP   LBFA6                         EXPAND POLYNOMIAL
0394
0395   83AB 81 49 0F DA A2 L83AB   FCB   $81,$49,$0F,$DA,$A2           1.57079633 (PI/2)
0396
0397                       * ATN
0398                       * A 12 TERM TAYLOR SERIES IS USED TO EVALUATE THE
0399                       * ARCTAN EXPRESSION.  TWO DIFFERENT FORMULI ARE USED
0400                       * TO EVALUATE THE EXPRESSION DEPENDING UPON
0401                       * WHETHER OR NOT THE ARGUMENT SQUARED IS > OR < 1.0
0402
0403                       * IF X**2<1 THEN ATN=X-(X**3)/3+(X**5)/5-(X**7)/7   ..
0404                       * IF X**2>=1 THEN ATN=PI/2-(1/X-1/((X**3)*3)+(1/((X**5)*5)-   )
0405
0406   83B0 96 54          ATN     LDA   FP0SGN                        * GET THE SIGN OF THE MANTISSA AND
0407   83B2 34 02                  PSHS  A                             * SAVE IT ON THE STACK
0408   83B4 2A 02                  BPL   L83B8                         BRANCH IF POSITIVE MANTISSA
0409   83B6 8D 24                  BSR   L83DC                         CHANGE SIGN OF FPA0
0410   83B8 96 4F          L83B8   LDA   FP0EXP                        * GET EXPONENT OF FPA0 AND
0411   83BA 34 02                  PSHS  A                             * SAVE IT ON THE STACK
0412   83BC 81 81                  CMPA  #$81                          IS FPAO < 1.0?
0413   83BE 25 05                  BLO   L83C5                         YES
0414   83C0 8E BA C5               LDX   #LBAC5                        POINT X TO FP CONSTANT 1.0
0415   83C3 8D DE                  BSR   L83A3                         GET RECIPROCAL OF FPA0
0416   83C5 8E 83 E0       L83C5   LDX   #L83E0                        POINT (X) TO TAYLOR SERIES COEFFICIENTS
0417   83C8 BD BE F0               JSR   LBEF0                         EXPAND POLYNOMIAL
0418   83CB 35 02                  PULS  A                             GET EXPONENT OF ARGUMENT
0419   83CD 81 81                  CMPA  #$81                          WAS ARGUMENT < 1.0?
0420   83CF 25 06                  BLO   L83D7                         YES
0421   83D1 8E 83 AB               LDX   #L83AB                        POINT (X) TO FP NUMBER (PI/2)
0422   83D4 BD B9 B9               JSR   LB9B9                         SUBTRACT FPA0 FROM (PI/2)
0423   83D7 35 02          L83D7   PULS  A                             * GET SIGN OF INITIAL ARGUMENT MANTISSA
0424   83D9 4D                     TSTA                                * AND SET FLAGS ACCORDING TO IT
0425   83DA 2A 03                  BPL   L83DF                         RETURN IF ARGUMENT WAS POSITIVE
0426   83DC 7E BE E9       L83DC   JMP   LBEE9                         CHANGE MANTISSA SIGN OF FPA0
0427   83DF 39             L83DF   RTS
0428                       *
0429                       * TCHEBYSHEV MODIFIED TAYLOR SERIES COEFFICIENTS FOR ARCTANGENT
0430   83E0 0B             L83E0   FCB   $0B                           TWELVE COEFFICIENTS
0431   83E1 76 B3 83 BD D3 L83E1   FCB   $76,$B3,$83,$BD,$D3           -6.84793912E-04 1/23
0432   83E6 79 1E F4 A6 F5 L83E6   FCB   $79,$1E,$F4,$A6,$F5           +4.85094216E-03 1/21
0433   83EB 7B 83 FC B0 10 L83EB   FCB   $7B,$83,$FC,$B0,$10           -0.0161117018   1/19
0434   83F0 7C 0C 1F 67 CA L83F0   FCB   $7C,$0C,$1F,$67,$CA           +0.0342096381   1/17
0435   83F5 7C DE 53 CB C1 L83F5   FCB   $7C,$DE,$53,$CB,$C1           -0.0542791328   1/15
0436   83FA 7D 14 64 70 4C L83FA   FCB   $7D,$14,$64,$70,$4C           +0.0724571965   1/13
0437   83FF 7D B7 EA 51 7A L83FF   FCB   $7D,$B7,$EA,$51,$7A           -0.0898023954   1/11
0438   8404 7D 63 30 88 7E L8404   FCB   $7D,$63,$30,$88,$7E           +0.110932413    1/9
0439   8409 7E 92 44 99 3A L8409   FCB   $7E,$92,$44,$99,$3A           -0.142839808    1/7
0440   840E 7E 4C CC 91 C7 L840E   FCB   $7E,$4C,$CC,$91,$C7           +0.199999121    1/5
0441   8413 7F AA AA AA 13 L8413   FCB   $7F,$AA,$AA,$AA,$13           -0.333333316    1/3
0442   8418 81 00 00 00 00 L8418   FCB   $81,$00,$00,$00,$00           +1.000000000    1/1
0443                       *
0444                       *** TCHEBYSHEV MODIFIED TAYLOR SERIES COEFFICIENTS FOR LN(X)
0445                       *
0446   841D 03             L841D   FCB   $03                           FOUR COEFFICIENTS
0447   841E 7F 5E 56 CB 79 L841E   FCB   $7F,$5E,$56,$CB,$79           0.434255942  (2/7)*(1/LN(2))
0448   8423 80 13 9B 0B 64 L8423   FCB   $80,$13,$9B,$0B,$64           0.576584541  (2/5)*(1/LN(2))
0449   8428 80 76 38 93 16 L8428   FCB   $80,$76,$38,$93,$16           0.961800759  (2/3)*(1/LN(2))
0450   842D 82 38 AA 3B 20 L842D   FCB   $82,$38,$AA,$3B,$20           2.88539007   (2/1)*(1/LN(2))
0451
0452   8432 80 35 04 F3 34 L8432   FCB   $80,$35,$04,$F3,$34           1/SQR(2)
0453
0454   8437 81 35 04 F3 34 L8437   FCB   $81,$35,$04,$F3,$34           SQR(2)
0455
0456   843C 80 80 00 00 00 L843C   FCB   $80,$80,$00,$00,$00           -.5
0457
0458   8441 80 31 72 17 F8 L8441   FCB   $80,$31,$72,$17,$F8           LN(2)
0459                       *
0460                       * LOG - NATURAL LOGARITHM (LN)
0461
0462                       * THE NATURAL OR NAPERIAN LOGARITHM IS CALCULATED USING
0463                       * MATHEMATICAL IDENTITIES. FPA0 IS OF THE FORM FPA0=A*(2**B) (SCIENTIFIC
0464                       * NOTATION). THEREFORE, THE LOG ROUTINE DETERMINES THE VALUE OF
0465                       * LN(A*(2**B)). A SERIES OF MATHEMATICAL IDENTITIES WILL EXPAND THIS
0466                       * TERM: LN(A*(2**B))=(-1/2+(1/LN(2))*(LN(A*SQR(2)))+B)*LN(2). ALL OF
0467                       * THE TERMS OF THE LATTER EXPRESSION ARE CONSTANTS EXCEPT FOR THE
0468                       * LN(A*SQR(2)) TERM WHICH IS EVALUATED USING THE TAYLOR SERIES EXPANSION
0469   8446 BD BC 6D       LOG     JSR   LBC6D                         CHECK STATUS OF FPA0
0470   8449 10 2F 2F FD            LBLE  LB44A                          FC  ERROR IF NEGATIVE OR ZERO
0471   844D 8E 84 32               LDX   #L8432                        POINT (X) TO FP NUMBER (1/SQR(2))
0472   8450 96 4F                  LDA   FP0EXP                        *GET EXPONENT OF ARGUMENT
0473   8452 80 80                  SUBA  #$80                          *SUBTRACT OFF THE BIAS AND
0474   8454 34 02                  PSHS  A                             *SAVE IT ON THE STACK
0475   8456 86 80                  LDA   #$80                          =FORCE EXPONENT OF FPA0
0476   8458 97 4F                  STA   FP0EXP                        =TO BE ZERO
0477   845A BD B9 C2               JSR   LB9C2                         ADD FPA0 TO (X)
0478   845D 8E 84 37               LDX   #L8437                        POINT X TO SQR(2)
0479   8460 BD BB 8F               JSR   LBB8F                         DIVIDE SQR(2) BY FPA0
0480   8463 8E BA C5               LDX   #LBAC5                        POINT X TO FP VALUE OF 1.00
```

```
0481   8466 BD B9 B9                  JSR   LB9B9                          SUBTRACT FPA0 FROM (X)
0482                          * NOW FPA0 = (1-SQR(2)*X)/(1+SQR(2)*X) WHERE X IS ARGUMENT
0483   8469 8E 84 1D                  LDX   #L841D                         POINT X TO TABLE OF COEFFICIENTS
0484   846C BD BE F0                  JSR   LBEF0                          EXPAND POLYNOMIAL
0485   846F 8E 84 3C                  LDX   #L843C                         POINT X TO FP VALUE OF (-.5)
0486   8472 BD B9 C2                  JSR   LB9C2                          ADD FPA0 TO X
0487   8475 35 04                     PULS  B                              GET EXPONENT OF ARGUMENT BACK (WITHOUT BIAS)
0488   8477 BD BD 99                  JSR   LBD99                          ADD ACCB TO FPA0
0489   847A 8E 84 41                  LDX   #L8441                         POINT X TO LN(2)
0490   847D 7E BA CA                  JMP   LBACA                          MULTIPLY FPA0 * LN(2)
0491
0492                          * SQR
0493   8480 BD BC 5F          SQR     JSR   LBC5F                          MOVE FPA0 TO FPA1
0494   8483 8E BE C0                  LDX   #LBEC0                         POINT (X) TO FP NUMBER (.5)
0495   8486 BD BC 14                  JSR   LBC14                          COPY A PACKED NUMBER FROM (X) TO FPA0
0496
0497                          * ARITHMETIC OPERATOR FOR EXPONENTIATION JUMPS
0498                          * HERE. THE FORMULA USED TO EVALUATE EXPONENTIATION
0499                          * IS A**X=E**(X LN A) = E**(FPA0*LN(FPA1)), E=2.7182818
0500   8489 27 67             L8489   BEQ   EXP                            DO A NATURAL EXPONENTIATION IF EXPONENT = 0
0501   848B 4D                        TSTA                                 *CHECK VALUE BEING EXPONENTIATED
0502   848C 26 03                     BNE   L8491                          *AND BRANCH IF IT IS <> 0
0503   848E 7E BA 3A                  JMP   LBA3A                          FPA0=0 IF RAISING ZERO TO A POWER
0504   8491 8E 00 4A          L8491   LDX   #V4A                           * PACK FPA0 AND SAVE
0505   8494 BD BC 35                  JSR   LBC35                          * IT IN FPA5 (ARGUMENT S EXPONENT)
0506   8497 5F                        CLRB                                 ACCB=DEFAULT RESULT SIGN FLAG; 0=POSITIVE
0507   8498 96 61                     LDA   FP1SGN                         *CHECK THE SIGN OF ARGUMENT
0508   849A 2A 10                     BPL   L84AC                          *BRANCH IF POSITIVE
0509   849C BD BC EE                  JSR   INT                            CONVERT EXPONENT INTO AN INTEGER
0510   849F 8E 00 4A                  LDX   #V4A                           POINT X TO FPA5 (ORIGINAL EXPONENT)
0511   84A2 96 61                     LDA   FP1SGN                         GET MANTISSA SIGN OF FPA1 (ARGUMENT)
0512   84A4 BD BC A0                  JSR   LBCA0                          *COMPARE FPA0 TO (X) AND
0513   84A7 26 03                     BNE   L84AC                          *BRANCH IF NOT EQUAL
0514   84A9 43                        COMA                                 TOGGLE FPA1 MANTISSA SIGN - FORCE POSITIVE
0515   84AA D6 01                     LDB   CHARAC                         GET LS BYTE OF INTEGER VALUE OF EXPONENT (RESULT SIGN FLAG)
0516   84AC BD BC 4C          L84AC   JSR   LBC4C                          COPY FPA1 TO FPA0; ACCA = MANTISSA SIGN
0517   84AF 34 04                     PSHS  B                              PUT RESULT SIGN FLAG ON THE STACK
0518 > 84B1 BD 84 46                  JSR   LOG                            GET NATURAL LOGARITHM OF FPA0
0519   84B4 8E 00 4A                  LDX   #V4A                           POINT (X) TO FPA5
0520   84B7 BD BA CA                  JSR   LBACA                          MULTIPLY FPA0 BY FPA5
0521   84BA 8D 36                     BSR   EXP                            CALCULATE E**(FPA0)
0522   84BC 35 02                     PULS  A                              * GET RESULT SIGN FLAG FROM THE STACK
0523   84BE 46                        RORA                                 * AND BRANCH IF NEGATIVE
0524   84BF 10 25 3A 26               LBCS  LBEE9                          CHANGE SIGN OF FPA0 MANTISSA
0525   84C3 39                        RTS
0526
0527                          * CORRECTION FACTOR FOR EXPONENTIAL FUNCTION
0528   84C4 81 38 AA 3B 29    L84C4   FCB   $81,$38,$AA,$3B,$29             1.44269504 ( CF )
0529                          *
0530                          * TCHEBYSHEV MODIFIED TAYLOR SERIES COEFFICIENTS FOR E**X
0531                          *
0532   84C9 07                L84C9   FCB   $07                            EIGHT COEFFICIENTS
0533   84CA 71 34 58 3E 56    L84CA   FCB   $71,$34,$58,$3E,$56             2.14987637E-05: 1/(7!*(CF**7))
0534   84CF 74 16 7E B3 1B    L84CF   FCB   $74,$16,$7E,$B3,$1B             1.4352314E-04 : 1/(6!*(CF**6))
0535   84D4 77 2F EE E3 85    L84D4   FCB   $77,$2F,$EE,$E3,$85             1.34226348E-03: 1/(5!*(CF**5))
0536   84D9 7A 1D 84 1C 2A    L84D9   FCB   $7A,$1D,$84,$1C,$2A             9.61401701E-03: 1/(4!*(CF**4))
0537   84DE 7C 63 59 58 0A    L84DE   FCB   $7C,$63,$59,$58,$0A             0.0555051269  : 1/(3!*(CF**3))
0538   84E3 7E 75 FD E7 C6    L84E3   FCB   $7E,$75,$FD,$E7,$C6             0.240226385   : 1/(2!*(CF**2))
0539   84E8 80 31 72 18 10    L84E8   FCB   $80,$31,$72,$18,$10             0.693147186   : 1/(1!*(CF**1))
0540   84ED 81 00 00 00 00    L84ED   FCB   $81,$00,$00,$00,$00             1.
0541                          *
0542                          * EXP ( E**X)
0543                          * THE EXPONENTIAL FUNCTION IS EVALUATED BY FIRST MULTIPLYING THE
0544                          * ARGUMENT BY A CORRECTION FACTOR (CF). AFTER THIS IS DONE, AN
0545                          * ARGUMENT >= 127 WILL YIELD A ZERO RESULT (NO UNDERFLOW) FOR A
0546                          * NEGATIVE ARGUMENT OR AN 'OV' (OVERFLOW) ERROR FOR A POSITIVE
0547                          * ARGUMENT. THE POLYNOMIAL COEFFICIENTS ARE MODIFIED TO REFLECT
0548                          * THE CF MULTIPLICATION AT THE START OF THE EVALUATION PROCESS.
0549
0550   84F2 8E 84 C4          EXP     LDX   #L84C4                         POINT X TO THE CORRECTION FACTOR
0551   84F5 BD BA CA                  JSR   LBACA                          MULTIPLY FPA0 BY (X)
0552   84F8 BD BC 2F                  JSR   LBC2F                          PACK FPA0 AND STORE IT IN FPA3
0553   84FB 96 4F                     LDA   FP0EXP                         *GET EXPONENT OF FPA0 AND
0554   84FD 81 88                     CMPA  #$88                           *COMPARE TO THE MAXIMUM VALUE
0555   84FF 25 03                     BLO   L8504                          BRANCH IF FPA0 < 128
0556   8501 7E BB 5C          L8501   JMP   LBB5C                          SET FPA0 = 0 OR  OV  ERROR
0557   8504 BD BC EE          L8504   JSR   INT                            CONVERT FPA0 TO INTEGER
0558   8507 96 01                     LDA   CHARAC                         GET LS BYTE OF INTEGER
0559   8509 8B 81                     ADDA  #$81                           * WAS THE ARGUMENT =127, IF SO
0560   850B 27 F4                     BEQ   L8501                          * THEN  OV  ERROR; THIS WILL ALSO ADD THE $80 BIAS
0561                          *                                            * REQUIRED WHEN THE NEW EXPONENT IS CALCULATED BELOW
0562   850D 4A                        DECA                                 DECREMENT ONE FROM THE EXPONENT, BECAUSE $81, NOT $80 WAS USED ABOVE
0563   850E 34 02                     PSHS  A                              SAVE EXPONENT OF INTEGER PORTION ON STACK
0564   8510 8E 00 40                  LDX   #V40                           POINT (X) TO FPA3
0565   8513 BD B9 B9                  JSR   LB9B9                          SUBTRACT FPA0 FROM (X) - GET FRACTIONAL PART OF ARGUMENT
0566   8516 8E 84 C9                  LDX   #L84C9                         POINT X TO COEFFICIENTS
0567   8519 BD BE FF                  JSR   LBEFF                          EVALUATE POLYNOMIAL FOR FRACTIONAL PART
0568   851C 0F 62                     CLR   RESSGN                         FORCE THE MANTISSA TO BE POSITIVE
0569   851E 35 02                     PULS  A                              GET INTEGER EXPONENT FROM STACK
0570   8520 BD BB 48                  JSR   LBB48                          * CALCULATE EXPONENT OF NEW FPA0 BY ADDING THE EXPONENTS OF THE
0571                          *                                            * INTEGER AND FRACTIONAL PARTS
0572   8523 39                        RTS
0573
0574                          * FIX
0575   8524 BD BC 6D          FIX     JSR   LBC6D                          CHECK STATUS OF FPA0
0576   8527 2B 03                     BMI   L852C                          BRANCH IF FPA0 = NEGATIVE
```

```
0577   8529 7E BC EE          L8529    JMP   INT                          CONVERT FPA0 TO INTEGER
0578   852C 03 54             L852C    COM   FP0SGN                       TOGGLE SIGN OF FPA0 MANTISSA
0579   852E 8D F9                      BSR   L8529                        CONVERT FPA0 TO INTEGER
0580   8530 7E BE E9                   JMP   LBEE9                        TOGGLE SIGN OF FPA0
0581
0582                          * EDIT
0583   8533 BD 89 AE          EDIT     JSR   L89AE                        GET LINE NUMBER FROM BASIC
0584   8536 32 62                      LEAS  $02,S                        PURGE RETURN ADDRESS OFF OF THE STACK
0585   8538 86 01             L8538    LDA   #$01                          LIST  FLAG
0586   853A 97 D8                      STA   VD8                          SET FLAG TO LIST LINE
0587   853C BD AD 01                   JSR   LAD01                        GO FIND THE LINE NUMBER IN PROGRAM
0588   853F 10 25 29 8F                LBCS  LAED2                        ERROR #7  UNDEFINED LINE #'
0589   8543 BD B7 C2                   JSR   LB7C2                        GO UNCRUNCH LINE INTO BUFFER AT LINBUF+1
0590   8546 1F 20                      TFR   Y,D                          PUT ABSOLUTE ADDRESS OF END OF LINE TO ACCD
0591   8548 83 02 DE                   SUBD  #LINBUF+2                    SUBTRACT OUT THE START OF LINE
0592   854B D7 D7                      STB   VD7                          SAVE LENGTH OF LINE
0593   854D DC 2B             L854D    LDD   BINVAL                       GET THE HEX VALUE OF LINE NUMBER
0594   854F BD BD CC                   JSR   LBDCC                        LIST THE LINE NUMBER ON THE SCREEN
0595   8552 BD B9 AC                   JSR   LB9AC                        PRINT A SPACE
0596   8555 8E 02 DD                   LDX   #LINBUF+1                    POINT X TO BUFFER
0597   8558 D6 D8                      LDB   VD8                          * CHECK TO SEE IF LINE IS TO BE
0598   855A 26 25                      BNE   L8581                        * LISTED TO SCREEN - BRANCH IF IT IS
0599   855C 5F                L855C    CLRB                               RESET DIGIT ACCUMULATOR - DEFAULT VALUE
0600   855D BD 86 87          L855D    JSR   L8687                        GET KEY STROKE
0601   8560 BD 90 AA                   JSR   L90AA                        SET CARRY IF NOT NUMERIC
0602   8563 25 0B                      BLO   L8570                        BRANCH IF NOT NUMERIC
0603   8565 80 30                      SUBA  #'0'                         MASK OFF ASCII
0604   8567 34 02                      PSHS  A                            SAVE IT ON STACK
0605   8569 86 0A                      LDA   #10                          NUMBER BEING CONVERTED IS BASE 10
0606   856B 3D                         MUL                                MULTIPLY ACCUMULATED VALUE BY BASE (10)
0607   856C EB E0                      ADDB  ,S+                          ADD DIGIT TO ACCUMULATED VALUE
0608   856E 20 ED                      BRA   L855D                        CHECK FOR ANOTHER DIGIT
0609   8570 C0 01             L8570    SUBB  #$01                         * REPEAT PARAMETER IN ACCB; IF IT
0610   8572 C9 01                      ADCB  #$01                         *IS 0, THEN MAKE IT  1
0611   8574 81 41                      CMPA  #'A'                         ABORT?
0612   8576 26 05                      BNE   L857D                        NO
0613   8578 BD B9 58                   JSR   LB958                        PRINT CARRIAGE RETURN TO SCREEN
0614   857B 20 BB                      BRA   L8538                        RESTART EDIT PROCESS - CANCEL ALL CHANGES
0615   857D 81 4C             L857D    CMPA  #'L'                         LIST?
0616   857F 26 0B                      BNE   L858C                        NO
0617   8581 8D 31             L8581    BSR   L85B4                        LIST THE LINE
0618   8583 0F D8                      CLR   VD8                          RESET THE LIST FLAG TO  NO LIST
0619   8585 BD B9 58                   JSR   LB958                        PRINT CARRIAGE RETURN
0620   8588 20 C3                      BRA   L854D                        GO INTERPRET ANOTHER EDIT COMMAND
0621   858A 32 62             L858A    LEAS  $02,S                        PURGE RETURN ADDRESS OFF OF THE STACK
0622   858C 81 0D             L858C    CMPA  #CR                          ENTER KEY?
0623   858E 26 0D                      BNE   L859D                        NO
0624   8590 8D 22                      BSR   L85B4                        ECHO THE LINE TO THE SCREEN
0625   8592 BD B9 58          L8592    JSR   LB958                        PRINT CARRIAGE RETURN
0626   8595 8E 02 DD                   LDX   #LINBUF+1                    * RESET BASIC S INPUT POINTER
0627   8598 9F A6                      STX   CHARAD                       * TO THE LINE INPUT BUFFER
0628   859A 7E AC A8                   JMP   LACA8                        GO PUT LINE BACK IN PROGRAM
0629   859D 81 45             L859D    CMPA  #'E'                         EXIT?
0630   859F 27 F1                      BEQ   L8592                        YES - SAME AS ENTER EXCEPT NO ECHO
0631   85A1 81 51                      CMPA  #'Q'                         QUIT?
0632   85A3 26 06                      BNE   L85AB                        NO
0633   85A5 BD B9 58                   JSR   LB958                        PRINT CARRIAGE RETURN TO SCREEN
0634   85A8 7E AC 73                   JMP   LAC73                        GO TO COMMAND LEVEL - MAKE NO CHANGES
0635   85AB 8D 02             L85AB    BSR   L85AF                        INTERPRET THE REMAINING COMMANDS AS SUBROUTINES
0636   85AD 20 AD                      BRA   L855C                        GO INTERPRET ANOTHER EDIT COMMAND
0637   85AF 81 20             L85AF    CMPA  #SPACE                       SPACE BAR?
0638   85B1 26 10                      BNE   L85C3                        NO
0639   85B3 8C                L85B3    FCB   SKP2                         SKIP TWO BYTES
0640                          * DISPLAY THE NEXT ACCB BYTES OF THE LINE IN THE BUFFER TO THE SCREEN
0641                          *
0642   85B4 C6 F9             L85B4    LDB   #LBUFMX-1                    250 BYTES MAX IN BUFFER
0643   85B6 A6 84             L85B6    LDA   ,X                           GET A CHARACTER FROM BUFFER
0644   85B8 27 08                      BEQ   L85C2                        EXIT IF IT S A 0
0645   85BA BD A2 82                   JSR   LA282                        SEND CHAR TO CONSOLE OUT
0646   85BD 30 01                      LEAX  $01,X                        MOVE POINTER UP ONE
0647   85BF 5A                         DECB                               DECREMENT CHARACTER COUNTER
0648   85C0 26 F4                      BNE   L85B6                        LOOP IF NOT DONE
0649   85C2 39                L85C2    RTS
0650   85C3 81 44             L85C3    CMPA  #'D'                         DELETE?
0651   85C5 26 48                      BNE   L860F                        NO
0652   85C7 6D 84             L85C7    TST   ,X                           * CHECK FOR END OF LINE
0653   85C9 27 F7                      BEQ   L85C2                        * AND BRANCH IF SO
0654   85CB 8D 04                      BSR   L85D1                        REMOVE A CHARACTER
0655   85CD 5A                         DECB                               DECREMENT REPEAT PARAMETER
0656   85CE 26 F7                      BNE   L85C7                        BRANCH IF NOT DONE
0657   85D0 39                         RTS
0658                          * REMOVE ONE CHARACTER FROM BUFFER
0659   85D1 0A D7             L85D1    DEC   VD7                          DECREMENT LENGTH OF BUFFER
0660   85D3 31 1F                      LEAY  $-01,X                       POINT Y TO ONE BEFORE CURRENT BUFFER POINTER
0661   85D5 31 21             L85D5    LEAY  $01,Y                        INCREMENT TEMPORARY BUFFER POINTER
0662   85D7 A6 21                      LDA   $01,Y                        GET NEXT CHARACTER
0663   85D9 A7 A4                      STA   ,Y                           PUT IT IN CURRENT POSITION
0664   85DB 26 F8                      BNE   L85D5                        BRANCH IF NOT END OF LINE
0665   85DD 39                         RTS
0666   85DE 81 49             L85DE    CMPA  #'I'                         INSERT?
0667   85E0 27 13                      BEQ   L85F5                        YES
0668   85E2 81 58                      CMPA  #'X'                         EXTEND?
0669   85E4 27 0D                      BEQ   L85F3                        YES
0670   85E6 81 48                      CMPA  #'H'                         HACK?
0671   85E8 26 5C                      BNE   L8646                        NO
0672   85EA 6F 84                      CLR   ,X                           TURN CURRENT BUFFER POINTER INTO END OF LINE FLAG
```

```
0673   85EC 1F 10                  TFR   X,D                    PUT CURRENT BUFFER POINTER IN ACCD
0674   85EE 83 02 DE               SUBD  #LINBUF+2              SUBTRACT INITIAL POINTER POSITION
0675   85F1 D7 D7                  STB   VD7                    SAVE NEW BUFFER LENGTH
0676   85F3 8D BF          L85F3   BSR   L85B4                  DISPLAY THE LINE ON THE SCREEN
0677   85F5 BD 86 87       L85F5   JSR   L8687                  GET A KEYSTROKE
0678   85F8 81 0D                  CMPA  #CR                    ENTER KEY?
0679   85FA 27 8E                  BEQ   L858A                  YES - INTERPRET ANOTHER COMMAND - PRINT LINE
0680   85FC 81 1B                  CMPA  #ESC                   ESCAPE?
0681   85FE 27 25                  BEQ   L8625                  YES - RETURN TO COMMAND LEVEL - DON T PRINT LINE
0682   8600 81 08                  CMPA  #BS                    BACK SPACE?
0683   8602 26 22                  BNE   L8626                  NO
0684   8604 8C 02 DD               CMPX  #LINBUF+1              COMPARE POINTER TO START OF BUFFER
0685   8607 27 EC                  BEQ   L85F5                  DO NOT ALLOW BS IF AT START
0686   8609 8D 45                  BSR   L8650                  MOVE POINTER BACK ONE, BS TO SCREEN
0687   860B 8D C4                  BSR   L85D1                  REMOVE ONE CHARACTER FROM BUFFER
0688   860D 20 E6                  BRA   L85F5                  GET INSERT SUB COMMAND
0689   860F 81 43          L860F   CMPA  #'C'                   CHANGE?
0690   8611 26 CB                  BNE   L85DE                  NO
0691   8613 6D 84          L8613   TST   ,X                     CHECK CURRENT BUFFER CHARACTER
0692   8615 27 0E                  BEQ   L8625                  BRANCH IF END OF LINE
0693   8617 BD 86 87               JSR   L8687                  GET A KEYSTROKE
0694   861A 25 02                  BLO   L861E                  BRANCH IF LEGITIMATE KEY
0695   861C 20 F5                  BRA   L8613                  TRY AGAIN IF ILLEGAL KEY
0696   861E A7 80          L861E   STA   ,X+                    INSERT NEW CHARACTER INTO BUFFER
0697   8620 8D 37                  BSR   L8659                  SEND NEW CHARACTER TO SCREEN
0698   8622 5A                     DECB                         DECREMENT REPEAT PARAMETER
0699   8623 26 EE                  BNE   L8613                  BRANCH IF NOT DONE
0700   8625 39             L8625   RTS
0701   8626 D6 D7          L8626   LDB   VD7                    GET LENGTH OF LINE
0702   8628 C1 F9                  CMPB  #LBUFMX-1              COMPARE TO MAXIMUM LENGTH
0703   862A 26 02                  BNE   L862E                  BRANCH IF NOT AT MAXIMUM
0704   862C 20 C7                  BRA   L85F5                  IGNORE INPUT IF LINE AT MAXIMUM LENGTH
0705   862E 34 10          L862E   PSHS  X                      SAVE CURRENT BUFFER POINTER
0706   8630 6D 80          L8630   TST   ,X+                    * SCAN THE LINE UNTIL END OF
0707   8632 26 FC                  BNE   L8630                  * LINE (0) IS FOUND
0708   8634 E6 82          L8634   LDB   ,-X                    DECR TEMP LINE POINTER AND GET A CHARACTER
0709   8636 E7 01                  STB   $01,X                  PUT CHARACTER BACK DOWN ONE SPOT
0710   8638 AC E4                  CMPX  ,S                     HAVE WE REACHED STARTING POINT?
0711   863A 26 F8                  BNE   L8634                  NO - KEEP GOING
0712   863C 32 62                  LEAS  $02,S                  PURGE BUFFER POINTER FROM STACK
0713   863E A7 80                  STA   ,X+                    INSERT NEW CHARACTER INTO THE LINE
0714   8640 8D 17                  BSR   L8659                  SEND A CHARACTER TO CONSOLE OUT
0715   8642 0C D7                  INC   VD7                    ADD ONE TO BUFFER LENGTH
0716   8644 20 AF                  BRA   L85F5                  GET INSERT SUB COMMAND
0717   8646 81 08          L8646   CMPA  #BS                    BACKSPACE?
0718   8648 26 12                  BNE   L865C                  NO
0719   864A 8D 04          L864A   BSR   L8650                  MOVE POINTER BACK 1, SEND BS TO SCREEN
0720   864C 5A                     DECB                         DECREMENT REPEAT PARAMETER
0721   864D 26 FB                  BNE   L864A                  LOOP UNTIL DONE
0722   864F 39                     RTS
0723   8650 8C 02 DD       L8650   CMPX  #LINBUF+1              COMPARE POINTER TO START OF BUFFER
0724   8653 27 D0                  BEQ   L8625                  DO NOT ALLOW BS IF AT START
0725   8655 30 1F                  LEAX  $-01,X                 MOVE POINTER BACK ONE
0726   8657 86 08                  LDA   #BS                    BACK SPACE
0727   8659 7E A2 82       L8659   JMP   LA282                  SEND TO CONSOLE OUT
0728   865C 81 4B          L865C   CMPA  #'K'                   KILL?
0729   865E 27 05                  BEQ   L8665                  YES
0730   8660 80 53                  SUBA  #'S'                   SEARCH?
0731   8662 27 01                  BEQ   L8665                  YES
0732   8664 39                     RTS
0733   8665 34 02          L8665   PSHS  A                      SAVE KILL/SEARCH FLAG ON STACK
0734   8667 8D 1E                  BSR   L8687                  * GET A KEYSTROKE (TARGET CHARACTER)
0735   8669 34 02                  PSHS  A                      * AND SAVE IT ON STACK
0736   866B A6 84          L866B   LDA   ,X                     = GET CURRENT BUFFER CHARACTER
0737   866D 27 16                  BEQ   L8685                  = AND RETURN IF END OF LINE
0738   866F 6D 61                  TST   $01,S                  CHECK KILL/SEARCH FLAG
0739   8671 26 06                  BNE   L8679                  BRANCH IF KILL
0740   8673 8D E4                  BSR   L8659                  SEND A CHARACTER TO CONSOLE OUT
0741   8675 30 01                  LEAX  $01,X                  INCREMENT BUFFER POINTER
0742   8677 20 03                  BRA   L867C                  CHECK NEXT INPUT CHARACTER
0743   8679 BD 85 D1       L8679   JSR   L85D1                  REMOVE ONE CHARACTER FROM BUFFER
0744   867C A6 84          L867C   LDA   ,X                     GET CURRENT INPUT CHARACTER
0745   867E A1 E4                  CMPA  ,S                     COMPARE TO TARGET CHARACTER
0746   8680 26 E9                  BNE   L866B                  BRANCH IF NO MATCH
0747   8682 5A                     DECB                         DECREMENT REPEAT PARAMETER
0748   8683 26 E6                  BNE   L866B                  BRANCH IF NOT DONE
0749   8685 35 A0          L8685   PULS  Y,PC                   THE Y PULL WILL CLEAN UP THE STACK FOR THE 2 PSHS A
0750                       *
0751                       * GET A KEYSTRKE
0752   8687 BD A1 71       L8687   JSR   LA171                  CALL CONSOLE IN : DEV NBR=SCREEN
0753   868A 81 7F                  CMPA  #$7F                   GRAPHIC CHARACTER?
0754   868C 24 F9                  BCC   L8687                  YES - GET ANOTHER CHAR
0755   868E 81 5F                  CMPA  #$5F                   SHIFT UP ARROW (QUIT INSERT)
0756   8690 26 02                  BNE   L8694                  NO
0757   8692 86 1B                  LDA   #ESC                   REPLACE W/ESCAPE CODE
0758   8694 81 0D          L8694   CMPA  #CR                    ENTER KEY
0759   8696 27 0E                  BEQ   L86A6                  YES
0760   8698 81 1B                  CMPA  #ESC                   ESCAPE?
0761   869A 27 0A                  BEQ   L86A6                  YES
0762   869C 81 08                  CMPA  #BS                    BACKSPACE?
0763   869E 27 06                  BEQ   L86A6                  YES
0764   86A0 81 20                  CMPA  #SPACE                 SPACE
0765   86A2 25 E3                  BLO   L8687                  GET ANOTHER CHAR IF CONTROL CHAR
0766   86A4 1A 01                  ORCC  #$01                   SET CARRY
0767   86A6 39             L86A6   RTS
0768
```

```
0769                        * TRON
0770   86A7 86              TRON     FCB   SKP1LD                    SKIP ONE BYTE AND LDA #$4F
0771
0772                        * TROFF
0773   86A8 4F              TROFF    CLRA                            TROFF FLAG
0774   86A9 97 AF                    STA   TRCFLG                    TRON/TROFF FLAG:0=TROFF, <> 0=TRON
0775   86AB 39                       RTS
0776
0777                        * POS
0778   86AC 96 6F           POS      LDA   DEVNUM                    GET DEVICE NUMBER
0779   86AE 34 02                    PSHS  A                         SAVE IT ON STACK
0780   86B0 BD A5 AE                 JSR   LA5AE                     GET DEVICE NUMBER
0781   86B3 BD A4 06                 JSR   LA406                     FILE STATUS CHECK
0782   86B6 BD A3 5F                 JSR   LA35F                     SET UP TAB FIELD WIDTH
0783   86B9 D6 6C                    LDB   DEVPOS                    GET PRINT POSITION
0784   86BB 7E A5 E4                 JMP   LA5E4                     CONVERT PRINT POSITION TO FLOATING POINT
0785
0786                        * VARPTR
0787   86BE BD B2 6A        VARPTR   JSR   LB26A                     SYNTAX CHECK FOR  (
0788   86C1 DC 1F                    LDD   ARYEND                    GET ADDR OF END OF ARRAYS
0789   86C3 34 06                    PSHS  B,A                       SAVE IT ON STACK
0790   86C5 BD B3 57                 JSR   LB357                     GET VARIABLE DESCRIPTOR
0791   86C8 BD B2 67                 JSR   LB267                     SYNTAX CHECK FOR  )
0792   86CB 35 06                    PULS  A,B                       GET END OF ARRAYS ADDR BACK
0793   86CD 1E 10                    EXG   X,D                       SWAP END OF ARRAYS AND VARIABLE DESCRIPTOR
0794   86CF 9C 1F                    CMPX  ARYEND                    COMPARE TO NEW END OF ARRAYS
0795   86D1 26 51                    BNE   L8724                      FC  ERROR IF VARIABLE WAS NOT DEFINED PRIOR TO CALLING VARPTR
0796   86D3 7E B4 F4                 JMP   GIVABF                    CONVERT VARIABLE DESCRIPTOR INTO A FP NUMBER
0797
0798                        * MID$(OLDSTRING,POSITION,LENGTH)=REPLACEMENT
0799   86D6 9D 9F           L86D6    JSR   GETNCH                    GET INPUT CHAR FROM BASIC
0800   86D8 BD B2 6A                 JSR   LB26A                     SYNTAX CHECK FOR  (
0801   86DB BD B3 57                 JSR   LB357                     * GET VARIABLE DESCRIPTOR ADDRESS AND
0802   86DE 34 10                    PSHS  X                         * SAVE IT ON THE STACK
0803   86E0 EC 02                    LDD   $02,X                     POINT ACCD TO START OF OLDSTRING
0804   86E2 10 93 21                 CMPD  FRETOP                    COMPARE TO START OF CLEARED SPACE
0805   86E5 23 04                    BLS   L86EB                     BRANCH IF <=
0806   86E7 93 27                    SUBD  MEMSIZ                    SUBTRACT OUT TOP OF CLEARED SPACE
0807   86E9 23 12                    BLS   L86FD                     BRANCH IF STRING IN STRING SPACE
0808   86EB E6 84           L86EB    LDB   ,X                        GET LENGTH OF OLDSTRING
0809   86ED BD B5 6D                 JSR   LB56D                     RESERVE ACCB BYTES IN STRING SPACE
0810   86F0 34 10                    PSHS  X                         SAVE RESERVED SPACE STRING ADDRESS ON STACK
0811   86F2 AE 62                    LDX   $02,S                     POINT X TO OLDSTRING DESCRIPTOR
0812   86F4 BD B6 43                 JSR   LB643                     MOVE OLDSTRING INTO STRING SPACE
0813   86F7 35 50                    PULS  X,U                       * GET OLDSTRING DESCRIPTOR ADDRESS AND RESERVED STRING
0814   86F9 AF 42                    STX   $02,U                     * ADDRESS AND SAVE RESERVED ADDRESS AS OLDSTRING ADDRESS
0815   86FB 34 40                    PSHS  U                         SAVE OLDSTRING DESCRIPTOR ADDRESS
0816   86FD BD B7 38        L86FD    JSR   LB738                     SYNTAX CHECK FOR COMMA AND EVALUATE LENGTH EXPRESSION
0817   8700 34 04                    PSHS  B                         SAVE POSITION PARAMETER ON STACK
0818   8702 5D                       TSTB                            * CHECK POSITION PARAMETER AND BRANCH
0819   8703 27 1F                    BEQ   L8724                     * IF START OF STRING
0820   8705 C6 FF                    LDB   #$FF                      DEFAULT REPLACEMENT LENGTH = $FF
0821   8707 81 29                    CMPA  #')'                      * CHECK FOR END OF MID$ STATEMENT AND
0822   8709 27 03                    BEQ   L870E                     * BRANCH IF AT END OF STATEMENT
0823   870B BD B7 38                 JSR   LB738                     SYNTAX CHECK FOR COMMA AND EVALUATE LENGTH EXPRESSION
0824   870E 34 04           L870E    PSHS  B                         SAVE LENGTH PARAMETER ON STACK
0825   8710 BD B2 67                 JSR   LB267                     SYNTAX CHECK FOR  )
0826   8713 C6 B3                    LDB   #$B3                      TOKEN FOR =
0827   8715 BD B2 6F                 JSR   LB26F                     SYNTAX CHECK FOR  =
0828   8718 8D 2E                    BSR   L8748                     EVALUATE REPLACEMENT STRING
0829   871A 1F 13                    TFR   X,U                       SAVE REPLACEMENT STRING ADDRESS IN U
0830   871C AE 62                    LDX   $02,S                     POINT X TO OLOSTRING DESCRIPTOR ADDRESS
0831   871E A6 84                    LDA   ,X                        GET LENGTH OF OLDSTRING
0832   8720 A0 61                    SUBA  $01,S                     SUBTRACT POSITION PARAMETER
0833   8722 24 03                    BCC   L8727                     INSERT REPLACEMENT STRING INTO OLDSTRING
0834   8724 7E B4 4A        L8724    JMP   LB44A                      FC  ERROR IF POSITION > LENGTH OF OLDSTRING
0835   8727 4C              L8727    INCA                            * NOW ACCA = NUMBER OF CHARACTERS TO THE RIGHT
0836                        *                                       * (INCLUSIVE) OF THE POSITION PARAMETER
0837   8728 A1 E4                    CMPA  ,S                        COMPARE TO LENGTH PARAMETER
0838   872A 24 02                    BCC   L872E                     BRANCH IF NEW STRING WILL FIT IN OLDSTRING
0839   872C A7 E4                    STA   ,S                        IF NOT, USE AS MUCH OF LENGTH PARAMETER AS WILL FIT
0840   872E A6 61           L872E    LDA   $01,S                     GET POSITION PARAMETER
0841   8730 1E 89                    EXG   A,B                       ACCA=LENGTH OF REPL STRING, ACCB=POSITION PARAMETER
0842   8732 AE 02                    LDX   $02,X                     POINT X TO OLDSTRING ADDRESS
0843   8734 5A                       DECB                            * BASIC S POSITION PARAMETER STARTS AT 1; THIS ROUTINE
0844                        *                                       * WANTS IT TO START AT ZERO
0845   8735 3A                       ABX                             POINT X TO POSITION IN OLDSTRING WHERE THE REPLACEMENT WILL GO
0846   8736 4D                       TSTA                            * IF THE LENGTH OF THE REPLACEMENT STRING IS ZERO
0847   8737 27 0D                    BEQ   L8746                     * THEN RETURN
0848   8739 A1 E4                    CMPA  ,S                        = IF THE LENGTH OF REPLACEMENT STRING IS <= THE
0849   873B 23 02                    BLS   L873F                       ADJUSTED LENGTH PARAMETER, THEN BRANCH
0850   873D A6 E4                    LDA   ,S                        OTHERWISE USE AS MUCH ROOM AS IS AVAILABLE
0851   873F 1F 89           L873F    TFR   A,B                       SAVE NUMBER OF BYTES TO MOVE IN ACCB
0852   8741 1E 31                    EXG   U,X                       SWAP SOURCE AND DESTINATION POINTERS
0853   8743 BD A5 9A                 JSR   LA59A                     MOVE (B) BYTES FROM (X) TO (U)
0854   8746 35 96           L8746    PULS  A,B,X,PC                  CLEAN UP THE STACK AND RETURN
0855   8748 BD B1 56        L8748    JSR   LB156                     EVALUATE EXPRESSION
0856   874B 7E B6 54                 JMP   LB654                     * TM  ERROR IF NUMERIC; RETURN WITH X POINTING
0857                        *                                       *TO STRING, ACCB = LENGTH
0858
0859                        * STRING
0860   874E BD B2 6A        STRING   JSR   LB26A                     SYNTAX CHECK FOR  (
0861   8751 BD B7 0B                 JSR   EVALEXPB                  EVALUATE EXPRESSION; ERROR IF > 255
0862   8754 34 04                    PSHS  B                         SAVE LENGTH OF STRING
0863   8756 BD B2 6D                 JSR   SYNCOMMA                  SYNTAX CHECK FOR COMMA
0864   8759 BD B1 56                 JSR   LB156                     EVALUATE EXPRESSION
```

```
0865   875C BD B2 67                JSR   LB267                          SYNTAX CHECK FOR  )
0866   875F 96 06                   LDA   VALTYP                         GET VARIABLE TYPE
0867   8761 26 05                   BNE   L8768                          BRANCH IF STRING
0868   8763 BD B7 0E                JSR   LB70E                          CONVERT FPA0 INTO AN INTEGER IN ACCB
0869   8766 20 03                   BRA   L876B                          SAVE THE STRING IN STRING SPACE
0870   8768 BD B6 A4          L8768 JSR   LB6A4                         GET FIRST BYTE OF STRING
0871   876B 34 04             L876B PSHS  B                              SAVE FIRST BYTE OF EXPRESSION
0872   876D E6 61                   LDB   $01,S                          GET LENGTH OF STRING
0873   876F BD B5 0F                JSR   LB50F                          RESERVE ACCB BYTES IN STRING SPACE
0874   8772 35 06                   PULS  A,B                            GET LENGTH OF STRING AND CHARACTER
0875   8774 27 05                   BEQ   L877B                          BRANCH IF NULL STRING
0876   8776 A7 80             L8776 STA   ,X+                            SAVE A CHARACTER IN STRING SPACE
0877   8778 5A                      DECB                                 DECREMENT LENGTH
0878   8779 26 FB                   BNE   L8776                          BRANCH IF NOT DONE
0879   877B 7E B6 9B          L877B JMP   LB69B                          PUT STRING DESCRIPTOR ONTO STRING STACK
0880
0881                          * INSTR
0882   877E BD B2 6A          INSTR JSR   LB26A                          SYNTAX CHECK FOR  (
0883   8781 BD B1 56                JSR   LB156                          EVALUATE EXPRESSION
0884   8784 C6 01                   LDB   #$01                           DEFAULT POSITION = 1 (SEARCH START)
0885   8786 34 04                   PSHS  B                              SAVE START
0886   8788 96 06                   LDA   VALTYP                         GET VARIABLE TYPE
0887   878A 26 10                   BNE   L879C                          BRANCH IF STRING
0888   878C BD B7 0E                JSR   LB70E                          CONVERT FPA0 TO INTEGER IN ACCB
0889   878F E7 E4                   STB   ,S                             SAVE START SEARCH VALUE
0890   8791 27 91                   BEQ   L8724                          BRANCH IF START SEARCH AT ZERO
0891   8793 BD B2 6D                JSR   SYNCOMMA                       SYNTAX CHECK FOR COMMA
0892   8796 BD B1 56                JSR   LB156                          EVALUATE EXPRESSION - SEARCH STRING
0893   8799 BD B1 46                JSR   LB146                           TM  ERROR IF NUMERIC
0894   879C 9E 52             L879C LDX   FPA0+2                         SEARCH STRING DESCRIPTOR ADDRESS
0895   879E 34 10                   PSHS  X                              SAVE ON THE STACK
0896   87A0 BD B2 6D                JSR   SYNCOMMA                       SYNTAX CHECK FOR COMMA
0897 > 87A3 BD 87 48                JSR   L8748                          EVALUATE TARGET STRING EXPRESSION
0898   87A6 34 14                   PSHS  X,B                            SAVE ADDRESS AND LENGTH ON STACK
0899   87A8 BD B2 67                JSR   LB267                          SYNTAX CHECK FOR ')'
0900   87AB AE 63                   LDX   $03,S                          * LOAD X WITH SEARCH STRING DESCRIPTOR ADDRESS
0901   87AD BD B6 59                JSR   LB659                          * AND GET THE LENGTH ANDADDRESS OF SEARCH STRING
0902   87B0 34 04                   PSHS  B                              SAVE LENGTH ON STACK
0903                          *
0904                          * AT THIS POINT THE STACK HAS THE FOLLOWING INFORMATION
0905                          * ON IT: 0,S-SEARCH LENGTH; 1,S-TARGET LENGTH; 2 3,S-TARGET
0906                          * ADDRESS; 4 5,S-SEARCH DESCRIPTOR ADDRESS; 6,S-SEARCH POSITION
0907   87B2 E1 66                   CMPB  $06,S                          COMPARE LENGTH OF SEARCH STRING TO START
0908   87B4 25 23                   BLO   L87D9                          POSITION; RETURN 0 IF LENGTH < START
0909   87B6 A6 61                   LDA   $01,S                          GET LENGTH OF TARGET STRING
0910   87B8 27 1C                   BEQ   L87D6                          BRANCH IF TARGET STRING = NULL
0911   87BA E6 66                   LDB   $06,S                          GET START POSITION
0912   87BC 5A                      DECB                                 MOVE BACK ONE
0913   87BD 3A                      ABX                                  POINT X TO POSITION IN SEARCH STRING WHERE SEARCHING WILL START
0914   87BE 31 84             L87BE LEAY  ,X                             POINT Y TO SEARCH POSITION
0915   87C0 EE 62                   LDU   $02,S                          POINT U TO START OF TARGET
0916   87C2 E6 61                   LDB   $01,S                          LOAD ACCB WITH LENGTH OF TARGET
0917   87C4 A6 E4                   LDA   ,S                             LOAD ACCA WITH LENGTH OF SEARCH
0918   87C6 A0 66                   SUBA  $06,S                          SUBTRACT SEARCH POSITION FROM SEARCH LENGTH
0919   87C8 4C                      INCA                                 ADD ONE
0920   87C9 A1 61                   CMPA  $01,S                          COMPARE TO TARGET LENGTH
0921   87CB 25 0C                   BLO   L87D9                          RETURN 0 IF TARGET LENGTH > WHAT S LEFT OF SEARCH STRING
0922   87CD A6 80             L87CD LDA   ,X+                            GET A CHARACTER FROM SEARCH STRING
0923   87CF A1 C0                   CMPA  ,U+                            COMPARE IT TO TARGET STRING
0924   87D1 26 0C                   BNE   L87DF                          BRANCH IF NO MATCH
0925   87D3 5A                      DECB                                 DECREMENT TARGET LENGTH
0926   87D4 26 F7                   BNE   L87CD                          CHECK ANOTHER CHARACTER
0927   87D6 E6 66             L87D6 LDB   $06,S                          GET MATCH POSITION
0928   87D8 21                L87D8 FCB   SKP1                           SKIP NEXT BYTE
0929   87D9 5F                L87D9 CLRB                                 MATCH ADDRESS = 0
0930   87DA 32 67                   LEAS  $07,S                          CLEAN UP THE STACK
0931   87DC 7E B4 F3                JMP   LB4F3                          CONVERT ACCB TO FP NUMBER
0932   87DF 6C 66             L87DF INC   $06,S                          INCREMENT SEARCH POSITION
0933   87E1 30 21                   LEAX  $01,Y                          MOVE X TO NEXT SEARCH POSITION
0934   87E3 20 D9                   BRA   L87BE                          KEEP LOOKING FOR A MATCH
0935
0936                          * ASCII TO FLOATING POINT CONVERSION RAM HOOK
0937   87E5 81 26             XVEC19 CMPA  #'&'                          *
0938   87E7 26 5C                   BNE   L8845                          * RETURN IF NOT HEX OR OCTAL VARIABLE
0939   87E9 32 62                   LEAS  $02,S                          PURGE RETURN ADDRESS FROM STACK
0940                          * PROCESS A VARIABLE PRECEEDED BY A  &  (&H,&O)
0941   87EB 0F 52             L87EB CLR   FPA0+2                         * CLEAR BOTTOM TWO
0942   87ED 0F 53                   CLR   FPA0+3                         * BYTES OF FPA0
0943   87EF 8E 00 52                LDX   #FPA0+2                        BYTES 2,3 OF FPA0 = (TEMPORARY ACCUMULATOR)
0944   87F2 9D 9F                   JSR   GETNCH                         GET A CHARACTER FROM BASIC
0945   87F4 81 4F                   CMPA  #'O'                           OCTAL VALUE?
0946   87F6 27 12                   BEQ   L880A                          YES
0947   87F8 81 48                   CMPA  #'H'                           HEX VALUE?
0948   87FA 27 23                   BEQ   L881F                          YES
0949   87FC 9D A5                   JSR   GETCCH                         GET CURRENT INPUT CHARACTER
0950   87FE 20 0C                   BRA   L880C                          DEFAULT TO OCTAL (&O)
0951   8800 81 38             L8800 CMPA  #'8'                           *
0952   8802 10 22 2A 71             LBHI  LB277                          *  SYNTAX  ERROR IF
0953   8806 C6 03                   LDB   #$03                           BASE 8 MULTIPLIER
0954   8808 8D 2A                   BSR   L8834                          ADD DIGIT TO TEMPORARY ACCUMULATOR
0955                          * EVALUATE AN &O VARIABLE
0956   880A 9D 9F             L880A JSR   GETNCH                         GET A CHARACTER FROM BASIC
0957   880C 25 F2             L880C BLO   L8800                          BRANCH IF NUMERIC
0958   880E 0F 50             L880E CLR   FPA0                           * CLEAR 2 HIGH ORDER
0959   8810 0F 51                   CLR   FPA0+1                         * BYTES OF FPA0
0960   8812 0F 06                   CLR   VALTYP                         SET VARXABLE TYPE TO NUMERIC
```

```
0961   8814 0F 63                CLR   FPSBYT                    ZERO OUT SUB BYTE OF FPA0
0962   8816 0F 54                CLR   FP0SGN                    ZERO OUT MANTISSA SIGN OF FPA0
0963   8818 C6 A0                LDB   #$A0                      * SET EXPONENT OF FPA0
0964   881A D7 4F                STB   FP0EXP                    *
0965   881C 7E BA 1C             JMP   LBA1C                     GO NORMALIZE FPA0
0966                     * EVALUATE AN &H VARIABLE
0967   881F 9D 9F       L881F    JSR   GETNCH                    GET A CHARACTER FROM BASIC
0968   8821 25 0B                BLO   L882E                     BRANCH IF NUMERIC
0969   8823 BD B3 A2             JSR   LB3A2                     SET CARRY IF NOT ALPHA
0970   8826 25 E6                BLO   L880E                     BRANCH IF NOT ALPHA OR NUMERIC
0971   8828 81 47                CMPA  #'G'                      CHECK FOR LETTERS A-F
0972   882A 24 E2                BCC   L880E                     BRANCH IF >= G (ILLEGAL HEX LETTER)
0973   882C 80 07                SUBA  #'A'-('9'+1)              SUBTRACT ASCII DIFFERENCE BETWEEN A AND 9
0974   882E C6 04       L882E    LDB   #$04                      BASE 16 DIGIT MULTIPLIER = 2**4
0975   8830 8D 02                BSR   L8834                     ADD DIGIT TO TEMPORARY ACCUMULATOR
0976   8832 20 EB                BRA   L881F                     KEEP EVALUATING VARIABLE
0977   8834 68 01       L8834    ASL   $01,X                     * MULTIPLY TEMPORARY
0978   8836 69 84                ROL   ,X                        * ACCUMULATOR BY TWO
0979   8838 10 25 32 56          LBCS  LBA92                      OV' OVERFLOW ERROR
0980   883C 5A                   DECB                            DECREMENT SHIFT COUNTER
0981   883D 26 F5                BNE   L8834                     MULTIPLY TEMPORARY ACCUMULATOR AGAIN
0982   883F 80 30                SUBA  #'0'                      MASK OFF ASCII
0983   8841 AB 01                ADDA  $01,X                     * ADD DIGIT TO TEMPORARY
0984   8843 A7 01                STA   $01,X                     * ACCUMULATOR AND SAVE IT
0985   8845 39          L8845    RTS
0986
0987                     * EXPRESSION EVALUATION RAM HOOK
0988   8846 35 40       XVEC15   PULS  U                         PULL RETURN ADDRESS AND SAVE IN U REGISTER
0989   8848 0F 06                CLR   VALTYP                    SET VARIABLE TYPE TO NUMERIC
0990   884A 9E A6                LDX   CHARAD                    CURRENT INPUT POINTER TO X
0991   884C 9D 9F                JSR   GETNCH                    GET CHARACTER FROM BASIC
0992   884E 81 26                CMPA  #'&'                      HEX AND OCTAL VARIABLES ARE PRECEEDED BY &
0993   8850 27 99                BEQ   L87EB                     PROCESS A  &  VARIABLE
0994   8852 81 CC                CMPA  #$CC                      TOKEN FOR FN
0995   8854 27 5E                BEQ   L88B4                     PROCESS FN CALL
0996   8856 81 FF                CMPA  #$FF                      CHECK FOR SECONDARY TOKEN
0997   8858 26 08                BNE   L8862                     NOT SECONDARY
0998   885A 9D 9F                JSR   GETNCH                    GET CHARACTER FROM BASIC
0999   885C 81 83                CMPA  #$83                      TOKEN FOR USR
1000   885E 10 27 00 CA          LBEQ  L892C                     PROCESS USR CALL
1001   8862 9F A6       L8862    STX   CHARAD                    RESTORE BASIC S INPUT POINTER
1002   8864 6E C4                JMP   ,U                        RETURN TO CALLING ROUTINE
1003   8866 9E 68       L8866    LDX   CURLIN                    GET CURRENT LINE NUMBER
1004   8868 30 01                LEAX  $01,X                     IN DIRECT MODE?
1005   886A 26 D9                BNE   L8845                     RETURN IF NOT IN DIRECT MODE
1006   886C C6 16                LDB   #2*11                      ILLEGAL DIRECT STATEMENT  ERROR
1007   886E 7E AC 46    L886E    JMP   LAC46                     PROCESS ERROR
1008
1009                     * DEF
1010   8871 AE 9F 00 A6 DEF      LDX   [CHARAD]                  GET TWO INPUT CHARS
1011   8875 8C FF 83             CMPX  #$FF83                    TOKEN FOR USR
1012   8878 10 27 00 93          LBEQ  L890F                     BRANCH IF DEF USR
1013   887C 8D 23                BSR   L88A1                     GET DESCRIPTOR ADDRESS FOR FN VARIABLE NAME
1014   887E 8D E6                BSR   L8866                     DON T ALLOW DEF FN IF IN DIRECT MODE
1015   8880 BD B2 6A             JSR   LB26A                     SYNTAX CHECK FOR  (
1016   8883 C6 80                LDB   #$80                      * GET THE FLAG TO INDICATE ARRAY VARIABLE SEARCH DISABLE
1017   8885 D7 08                STB   ARYDIS                    * AND SAVE IT IN THE ARRAY DISABLE FLAG
1018   8887 BD B3 57             JSR   LB357                     GET VARIABLE DESCRIPTOR
1019   888A 8D 25                BSR   L88B1                      TM  ERROR IF STRING
1020   888C BD B2 67             JSR   LB267                     SYNTAX CHECK FOR  )
1021   888F C6 B3                LDB   #$B3                      TOKEN FOR  =
1022   8891 BD B2 6F             JSR   LB26F                     DO A SYNTAX CHECK FOR =
1023   8894 9E 4B                LDX   V4B                       GET THE ADDRESS OF THE FN NAME DESCRIPTOR
1024   8896 DC A6                LDD   CHARAD                    * GET THE CURRENT INPUT POINTER ADDRESS AND
1025   8898 ED 84                STD   ,X                        * SAVE IT IN FIRST 2 BYTES OF THE DESCRIPTOR
1026   889A DC 39                LDD   VARPTR                    = GET THE DESCRIPTOR ADDRESS OF THE ARGUMENT
1027   889C ED 02                STD   $02,X                     = VARIABLE AND SAVE IT IN THE DESCRIPTOR OF THE FN NAME
1028   889E 7E AE E0             JMP   LAEE0                     MOVE INPUT POINTER TO END OF LINE OR SUBLINE
1029   88A1 C6 CC       L88A1    LDB   #$CC                      TOKEN FOR FN
1030   88A3 BD B2 6F             JSR   LB26F                     DO A SYNTAX CHECK FOR FN
1031   88A6 C6 80                LDB   #$80                      * GET THE FLAG TO INDICATE ARRAY VARIABLE SEARCH DISABLE FLAG
1032   88A8 D7 08                STB   ARYDIS                    * AND SAVE IT IN ARRAY VARIABLE FLAG
1033   88AA 8A 80                ORA   #$80                      SET BIT 7 OF CURRENT INPUT CHARACTER TO INDICATE AN FN VARIABLE
1034   88AC BD B3 5C             JSR   LB35C                     * GET THE DESCRIPTOR ADDRESS OF THIS
1035   88AF 9F 4B                STX   V4B                       * VARIABLE AND SAVE IT IN V4B
1036   88B1 7E B1 43    L88B1    JMP   LB143                      TM  ERROR IF STRING VARIABLE
1037                     * EVALUATE AN FN CALL
1038   88B4 8D EB       L88B4    BSR   L88A1                     * GET THE DESCRIPTOR OF THE FN NAME
1039   88B6 34 10                PSHS  X                         * VARIABLE AND SAVE IT ON THE STACK
1040   88B8 BD B2 62             JSR   LB262                     SYNTAX CHECK FOR  (  & EVALUATE EXPR
1041   88BB 8D F4                BSR   L88B1                      TM  ERROR IF STRING VARIABLE
1042   88BD 35 40                PULS  U                         POINT U TO FN NAME DESCRIPTOR
1043   88BF C6 32                LDB   #2*25                      UNDEFINED FUNCTION CALL  ERROR
1044   88C1 AE 42                LDX   $02,U                     POINT X TO ARGUMENT VARIABLE DESCRIPTOR
1045   88C3 27 A9                BEQ   L886E                     BRANCH TO ERROR HANDLER
1046   88C5 10 9E A6             LDY   CHARAD                    SAVE CURRENT INPUT POINTER IN Y
1047   88C8 EE C4                LDU   ,U                        * POINT U TO START OF FN FORMULA AND
1048   88CA DF A6                STU   CHARAD                    * SAVE IT IN INPUT POINTER
1049   88CC A6 04                LDA   $04,X                     = GET FP VALUE OF
1050   88CE 34 02                PSHS  A                         = ARGUMENT VARIABLE, CURRENT INPUT
1051   88D0 EC 84                LDD   ,X                        = POINTER, AND ADDRESS OF START
1052   88D2 EE 02                LDU   $02,X                     = OF FN FORMULA AND SAVE
1053   88D4 34 76                PSHS  U,Y,X,B,A                 = THEM ON THE STACK
1054   88D6 BD BC 35             JSR   LBC35                     PACK FPA0 AND SAVE IT IN (X)
1055   88D9 BD B1 41    L88D9    JSR   LB141                     EVALUATE FN EXPRESSION
1056   88DC 35 76                PULS  A,B,X,Y,U                 RESTORE REGISTERS
```

```
1057   88DE ED 84              STD   ,X                        * GET THE FP
1058   88E0 EF 02              STU   $02,X                     * VALUE OF THE ARGUMENT
1059   88E2 35 02              PULS  A                         * VARIABLE OFF OF THE
1060   88E4 A7 04              STA   $04,X                     * STACK AND RE-SAVE IT
1061   88E6 9D A5              JSR   GETCCH                    GET FINAL CHARACTER OF THE FN FORMULA
1062   88E8 10 26 29 8B        LBNE  LB277                      SYNTAX  ERROR IF NOT END OF LINE
1063   88EC 10 9F A6           STY   CHARAD                    RESTORE INPUT POINTER
1064   88EF 39           L88EF RTS
1065
1066                     * ERROR DRIVER RAM HOOK
1067   88F0 C1 32        XVEC17 CMPB #2*25                     CHECK FOR EXBAS ERROR NUMBER
1068   88F2 25 FB              BLO   L88EF                     BRANCH IF < EXBAS ERROR
1069   88F4 BD A7 E9           JSR   LA7E9                     TURN CASSETTE MOTOR OFF
1070   88F7 BD A9 74           JSR   LA974                     DISABLE ANALOG MULTIPLEXER
1071   88FA BD AD 33           JSR   LAD33                     DO PART OF A NEW
1072   88FD 0F 6F              CLR   DEVNUM                    SET DEVICE NUMBER TO SCREEN
1073   88FF BD B9 5C           JSR   LB95C                     MOVE CURSOR TO START OF NEXT LINE
1074   8902 BD B9 AF           JSR   LB9AF                     SEND A  ?  TO CONSOLE OUT
1075   8905 8E 88 D9           LDX   #L890B-25*2               POINT X TO EXBAS ERRORS
1076   8908 7E AC 60           JMP   LAC60                     PROCESS ERROR
1077
1078                     * ADDITIONAL ERROR MESSAGES ADDED BY EXTENDED BASIC
1079
1080   890B 55 46        L890B FCC   'UF'                      25 UNDEFINED FUNCTION (FN) CALL
1081   890D 4E 45        L890D FCC   'NE'                      26 FILE NOT FOUND
1082
1083                     * DEF USR
1084   890F 9D 9F        L890F JSR   GETNCH                    SKIP PAST SECOND BYTE OF DEF USR TOKEN
1085   8911 8D 09              BSR   L891C                     GET FN NUMBER
1086   8913 34 10              PSHS  X                         SAVE FN EXEC ADDRESS STORAGE LOC
1087   8915 8D 2D              BSR   L8944                     CALCULATE EXEC ADDRESS
1088   8917 35 40              PULS  U                         GET FN EXEC ADDRESS STORAGE LOC
1089   8919 AF C4              STX   ,U                        SAVE EXEC ADDRESS
1090   891B 39                 RTS
1091   891C 5F           L891C CLRB                            DEFAULT TO USR0 IF NO ARGUMENT
1092   891D 9D 9F              JSR   GETNCH                    GET A CHARACTER FROM BASIC
1093   891F 24 06              BCC   L8927                     BRANCH IF NOT NUMERIC
1094   8921 80 30              SUBA  #'0'                      MASK OFF ASCII
1095   8923 1F 89              TFR   A,B                       SAVE USR NUMBER IN ACCB
1096   8925 9D 9F              JSR   GETNCH                    GET A CHARACTER FROM BASIC
1097   8927 9E B0        L8927 LDX   USRADR                    GET ADDRESS OF STORAGE LOCs FOR USR ADDRESS
1098   8929 58                 ASLB                            X2 - 2 BYTES/USR ADDRESS
1099   892A 3A                 ABX                             ADD OFFSET TO START ADDRESS OF STORAGE LOCs
1100   892B 39                 RTS
1101                     * PROCESS A USR CALL
1102   892C 8D EE        L892C BSR   L891C                     GET STORAGE LOC OF EXEC ADDRESS FOR USR N
1103   892E AE 84              LDX   ,X                        * GET EXEC ADDRESS AND
1104   8930 34 10              PSHS  X                         * PUSH IT ONTO STACK
1105   8932 BD B2 62           JSR   LB262                     SYNTAX CHECK FOR  (  & EVALUATE EXPR
1106   8935 8E 00 4F           LDX   #FP0EXP                   POINT X TO FPA0
1107   8938 96 06              LDA   VALTYP                    GET VARIABLE TYPE
1108   893A 27 07              BEQ   L8943                     BRANCH IF NUMERIC, STRING IF <> 0
1109   893C BD B6 57           JSR   LB657                     GET LENGTH & ADDRESS OF STRING VARIABLE
1110   893F 9E 52              LDX   FPA0+2                    GET POINTER TO STRING DESCRIPTOR
1111   8941 96 06              LDA   VALTYP                    GET VARIABLE TYPE
1112   8943 39           L8943 RTS                             JUMP TO USR ROUTINE (PSHS X ABOVE)
1113   8944 C6 B3        L8944 LDB   #$B3                      TOKEN FOR  =
1114   8946 BD B2 6F           JSR   LB26F                     DO A SYNTAX CHECK FOR =
1115   8949 7E B7 3D           JMP   LB73D                     EVALUATE EXPRESSION, RETURN VALUE IN X
1116
1117                     * EXTENDED BASIC S IRQ ROUTINE
1118
1119   894C B6 FF 03     XIRQSV LDA  PIA0+3                    GET PIA0, PORT B CONTROL REGISTER
1120   894F 2B 01              BMI   L8952                     BRANCH IF 60 HZ INTERRUPT
1121   8951 3B                 RTI                             RETURN IF 63.5 MICROSECOND INTERRUPT
1122   8952 B6 FF 02     L8952 LDA   PIA0+2                    RESET PIA INTERRUPT FLAG
1123   8955 BE 01 12     L8955 LDX   TIMVAL                    GET REAL TIME CLOCK
1124   8958 30 01              LEAX  $01,X                     INCREMENT IT
1125   895A BF 01 12           STX   TIMVAL                    SAVE IT
1126   895D 7E 9C 3E           JMP   L9C3E                     GO CHECK SOME MORE STUFF
1127
1128   8960 9D 9F        L8960 JSR   GETNCH                    GET A CHARACTER FROM BASIC
1129   8962 8D E0              BSR   L8944                     GET NEW TIMER VALUE
1130   8964 BF 01 12           STX   TIMVAL                    SET TIMER COUNTER
1131   8967 39                 RTS
1132
1133                     * TIMER
1134   8968 BE 01 12     TIMER LDX   TIMVAL                    GET TIMER VALUE
1135   896B 9F 52              STX   FPA0+2                    SAVE TIMER VALUE IN BOTTOM OF FPA0
1136   896D 7E 88 0E           JMP   L880E                     CONVERT BALANCE OF FPA0 TO POSITIVE INTEGER
1137
1138                     * DEL
1139   8970 10 27 2A D6  DEL   LBEQ  LB44A                     FC  ERROR IF NO ARGUMENT
1140   8974 BD AF 67           JSR   LAF67                     CONVERT A DECIMAL BASIC NUMBER TO BINARY
1141   8977 BD AD 01           JSR   LAD01                     FIND RAM ADDRESS OF START OF A BASIC LINE
1142   897A 9F D3              STX   VD3                       SAVE RAM ADDRESS OF STARTING LINE NUMBER
1143   897C 9D A5              JSR   GETCCH                    GET CURRENT INPUT CHARACTER
1144   897E 27 10              BEQ   L8990                     BRANCH IF END OF LINE
1145   8980 81 AC              CMPA  #$AC                      TOKEN FOR  -'
1146   8982 26 3B              BNE   L89BF                     TERMINATE COMMAND IF LINE NUMBER NOT FOLLOWED BY  -
1147   8984 9D 9F              JSR   GETNCH                    GET A CHARACTER FROM BASIC
1148   8986 27 04              BEQ   L898C                     IF END OF LINE, USE DEFAULT ENDING LINE NUMBER
1149   8988 8D 24              BSR   L89AE                     * CONVERT ENDING LINE NUMBER TO BINARY
1150   898A 20 04              BRA   L8990                     * AND SAVE IT IN BINVAL
1151   898C 86 FF        L898C LDA   #$FF                      = USE $FFXX AS DEFAULT ENDING
1152   898E 97 2B              STA   BINVAL                    = LINE NUMBER - SAVE IT IN BINVAL
```

```
1153   8990 DE D3          L8990   LDU   VD3                 POINT U TO STARTING LINE NUMBER ADDRESS
1154   8992 8C             L8992   FCB   SKP2                SKIP TWO BYTES
1155   8993 EE C4          L8993   LDU   ,U                  POINT U TO START OF NEXT LINE
1156   8995 EC C4                  LDD   ,U                  CHECK FOR END OF PROGRAM
1157   8997 27 06                  BEQ   L899F               BRANCH IF END OF PROGRAM
1158   8999 EC 42                  LDD   $02,U               LOAD ACCD WITH THIS LINE S NUMBER
1159   899B 93 2B                  SUBD  BINVAL              SUBTRACT ENDING LINE NUMBER ADDRESS
1160   899D 23 F4                  BLS   L8993               BRANCH IF = < ENDING LINE NUMBER
1161   899F 9E D3          L899F   LDX   VD3                 GET STARTING LINE NUMBER
1162   89A1 8D 15                  BSR   L89B8               MOVE (U) TO (X) UNTIL END OF PROGRAM
1163   89A3 BD AD 21               JSR   LAD21               RESET BASIC S INPUT POINTER AND ERASE VARIABLES
1164   89A6 9E D3                  LDX   VD3                 GET STARTING LINE NUMBER ADDRESS
1165   89A8 BD AC F1               JSR   LACF1               RECOMPUTE START OF NEXT LINE ADDRESSES
1166   89AB 7E AC 73               JMP   LAC73               JUMP TO BASIC S MAIN COMMAND LOOP
1167   89AE BD AF 67       L89AE   JSR   LAF67               GO GET LINE NUMBER CONVERTED TO BINARY
1168   89B1 7E A5 C7               JMP   LA5C7               MAKE SURE THERE S NO MORE ON THIS LINE
1169   89B4 A6 C0          L89B4   LDA   ,U+                GET A BYTE FROM (U)
1170   89B6 A7 80                  STA   ,X+                 MOVE THE BYTE TO (X)
1171   89B8 11 93 1B       L89B8   CMPU  VARTAB              COMPARE TO END OF BASIC
1172   89BB 26 F7                  BNE   L89B4               BRANCH IF NOT AT END
1173   89BD 9F 1B                  STX   VARTAB              SAVE (X) AS NEW END OF BASIC
1174   89BF 39             L89BF   RTS
1175
1176                       * LINE INPUT
1177   89C0 BD 88 66       L89C0   JSR   L8866                BS  ERROR IF IN DIRECT MODE
1178   89C3 9D 9F                  JSR   GETNCH              GET A CHAR FROM BASIC
1179   89C5 81 23                  CMPA  #'#'                * CHECK FOR DEVICE NUMBER FLAG AND
1180   89C7 26 09                  BNE   L89D2               * BRANCH IF NOT THERE
1181   89C9 BD A5 A5               JSR   LA5A5               CHECK FOR VALID DEVICE NUMBER
1182   89CC BD A3 ED               JSR   LA3ED               CHECK FOR OPEN FILE
1183   89CF BD B2 6D               JSR   SYNCOMMA            SYNTAX CHECK FOR COMMA
1184   89D2 81 22          L89D2   CMPA  #'"'               CHECK FOR PROMPT STRING
1185   89D4 26 0B                  BNE   L89E1               BRANCH IF NO PROMPT STRING
1186   89D6 BD B2 44               JSR   LB244               STRIP OFF PROMPT STRING & PUT IT ON STRING STACK
1187   89D9 C6 3B                  LDB   #';'                *
1188   89DB BD B2 6F               JSR   LB26F               * DO A SYNTAX CHECK FOR;
1189   89DE BD B9 9F               JSR   LB99F               REMOVE PROMPT STRING FROM STRING STACK & SEND TO CONSOLE OUT
1190   89E1 32 7E          L89E1   LEAS  $-02,S              RESERVE TWO STORAGE SLOTS ON STACK
1191   89E3 BD B0 35               JSR   LB035               INPUT A LINE FROM CURRENT INPUT DEVICE
1192   89E6 32 62                  LEAS  $02,S               CLEAN UP THE STACK
1193   89E8 0F 6F                  CLR   DEVNUM              SET DEVICE NUMBER TO SCREEN
1194   89EA BD B3 57               JSR   LB357               SEARCH FOR A VARIABLE
1195   89ED 9F 3B                  STX   VARDES              SAVE POINTER TO VARIABLE DESCRIPTOR
1196   89EF BD B1 46               JSR   LB146                 TM  ERROR IF VARIABLE TYPE = NUMERIC
1197   89F2 8E 02 DC               LDX   #LINBUF             POINT X TO THE STRING BUFFER WHERE THE INPUT STRING WAS STORED
1198   89F5 4F                     CLRA                      TERMINATOR CHARACTER 0 (END OF LINE)
1199   89F6 BD B5 1A               JSR   LB51A               PARSE THE INPUT STRING AND STORE IT IN THE STRING SPACE
1200   89F9 7E AF A4               JMP   LAFA4               REMOVE DESCRIPTOR FROM STRING STACK
1201   89FC BD AF 67       L89FC   JSR   LAF67               STRIP A DECIMAL NUMBER FROM BASIC INPUT LINE
1202   89FF 9E 2B                  LDX   BINVAL              GET BINARY VALUE
1203   8A01 39                     RTS
1204   8A02 9E D1          L8A02   LDX   VD1                 GET CURRENT OLD NUMBER BEING RENUMBERED
1205   8A04 9F 2B          L8A04   STX   BINVAL              SAVE THE LINE NUMBER BEING SEARCHED FOR
1206   8A06 7E AD 01               JMP   LAD01               GO FIND THE LINE NUMBER IN BASIC PROGRAM
1207
1208                       * RENUM
1209   8A09 BD AD 26       RENUM   JSR   LAD26               ERASE VARIABLES
1210   8A0C CC 00 0A               LDD   #10                 DEFAULT LINE NUMBER INTERVAL
1211   8A0F DD D5                  STD   VD5                 SAVE DEFAULT RENUMBER START LINE NUMBER
1212   8A11 DD CF                  STD   VCF                 SAVE DEFAULT INTERVAL
1213   8A13 5F                     CLRB                      NOW ACCD = 0
1214   8A14 DD D1                  STD   VD1                 DEFAULT LINE NUMBER OF WHERE TO START RENUMBERING
1215   8A16 9D A5                  JSR   GETCCH              GET CURRENT INPUT CHARACTER
1216   8A18 24 06                  BCC   L8A20               BRANCH IF NOT NUMERIC
1217   8A1A 8D E0                  BSR   L89FC               CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY
1218   8A1C 9F D5                  STX   VD5                 SAVE LINE NUMBER WHERE RENUMBERING STARTS
1219   8A1E 9D A5                  JSR   GETCCH              GET CURRENT INPUT CHARACTER
1220   8A20 27 1B          L8A20   BEQ   L8A3D               BRANCH IF END OF LINE
1221   8A22 BD B2 6D               JSR   SYNCOMMA            SYNTAX CHECK FOR COMMA
1222   8A25 24 06                  BCC   L8A2D               BRANCH IF NEXT CHARACTER NOT NUMERIC
1223   8A27 8D D3                  BSR   L89FC               CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY
1224   8A29 9F D1                  STX   VD1                 SAVE NEW RENUMBER LINE
1225   8A2B 9D A5                  JSR   GETCCH              GET CURRENT INPUT CHARACTER
1226   8A2D 27 0E          L8A2D   BEQ   L8A3D               BRANCH IF END OF LINE
1227   8A2F BD B2 6D               JSR   SYNCOMMA            SYNTAX CHECK FOR COMMA
1228   8A32 24 06                  BCC   L8A3A               BRANCH IF NEXT CHARACTER NOT NUMERIC
1229   8A34 8D C6                  BSR   L89FC               CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY
1230   8A36 9F CF                  STX   VCF                 SAVE NEW INTERVAL
1231   8A38 27 49                  BEQ   L8A83                FC' ERROR
1232   8A3A BD A5 C7       L8A3A   JSR   LA5C7               CHECK FOR MORE CHARACTERS ON LINE -  SYNTAX  ERROR IF ANY
1233   8A3D 8D C3          L8A3D   BSR   L8A02               GO GET ADDRESS OF OLD NUMBER BEING RENUMBERED
1234   8A3F 9F D3                  STX   VD3                 SAVE ADDRESS
1235   8A41 9E D5                  LDX   VD5                 GET NEXT RENUMBERED LINE NUMBER TO USE
1236   8A43 8D BF                  BSR   L8A04               FIND THE LINE NUMBER IN THE BASIC PROGRAM
1237   8A45 9C D3                  CMPX  VD3                 COMPARE TO ADDRESS OF OLD LINE NUMBER
1238   8A47 25 3A                  BLO   L8A83                FC  ERROR IF NEW ADDRESS < OLD ADDRESS
1239   8A49 8D 1C                  BSR   L8A67               MAKE SURE RENUMBERED LINE NUMBERS WILL BE IN RANGE
1240   8A4B BD 8A DD               JSR   L8ADD               CONVERT ASCII LINE NUMBERS TO  EXPANDED  BINARY
1241   8A4E BD AC EF               JSR   LACEF               RECALCULATE NEXT LINE RAM ADDRESSES
1242   8A51 8D AF                  BSR   L8A02               GET RAM ADDRESS OF FIRST LINE TO BE RENUMBERED
1243   8A53 9F D3                  STX   VD3                 SAVE IT
1244   8A55 8D 3A                  BSR   L8A91               MAKE SURE LINE NUMBERS EXIST
1245   8A57 8D 0F                  BSR   L8A68               INSERT NEW LINE NUMBERS IN LINE HEADERS
1246   8A59 8D 36                  BSR   L8A91               INSERT NEW LINE NUMBERS IN PROGRAM STATEMENTS
1247   8A5B BD 8B 7B               JSR   L8B7B               CONVERT PACKED BINARY LINE NUMBERS TO ASCII
1248   8A5E BD AD 26               JSR   LAD26               ERASE VARIABLES
```

```
1249   8A61 BD AC EF              JSR   LACEF                     RECALCULATE NEXT LINE RAM ADDRESS
125Ø   8A64 7E AC 73              JMP   LAC73                     GO BACK TO BASIC S MAIN LOOP
1251   8A67 86            L8A67   FCB   SKP1LD                   SKIP ONE BYTE - LDA #$4F
1252   8A68 4F            L8A68   CLRA                            NEW LINE NUMBER FLAG - Ø; INSERT NEW LINE NUMBERS
1253   8A69 97 D8                 STA   VD8                       SAVE NEW LINE NUMBER FLAG; Ø = INSERT NEW NUMBERS
1254   8A6B 9E D3                 LDX   VD3                       GET ADDRESS OF OLD LINE NUMBER BEING RENUMBERED
1255   8A6D DC D5                 LDD   VD5                       GET THE CURRENT RENUMBERED LINE NUMBER
1256   8A6F 8D 15                 BSR   L8A86                     RETURN IF END OF PROGRAM
1257   8A71 ØD D8         L8A71   TST   VD8                       CHECK NEW LINE NUMBER FLAG
1258   8A73 26 Ø2                 BNE   L8A77                     BRANCH IF NOT INSERTING NEW LINE NUMBERS
1259   8A75 ED Ø2                 STD   $Ø2,X                     STORE THE NEW LINE NUMBER IN THE BASIC PROGRAM
126Ø   8A77 AE 84         L8A77   LDX   ,X                        POINT X TO THE NEXT LINE IN BASIC
1261   8A79 8D ØB                 BSR   L8A86                     RETURN IF END OF PROGRAM
1262   8A7B D3 CF                 ADDD  VCF                       ADD INTERVAL TO CURRENT RENUMBERED LINE NUMBER
1263   8A7D 25 Ø4                 BLO   L8A83                      FC  ERROR IF LINE NUMBER > $FFFF
1264   8A7F 81 FA                 CMPA  #MAXLIN                   LARGEST LINE NUMBER = $F9FF
1265   8A81 25 EE                 BLO   L8A71                     BRANCH IF LEGAL LINE NUMBER
1266   8A83 7E B4 4A      L8A83   JMP   LB44A                      FC  ERROR IF LINE NUMBER MS BYTE > $F9
1267                      * TEST THE TWO BYTES POINTED TO BY (X).
1268                      * NORMAL RETURN IF <> Ø. IF = Ø (END OF
1269                      * PROGRAM) RETURN IS PULLED OFF STACK AND
127Ø                      * YOU RETURN TO PREVIOUS SUBROUTINE CALL.
1271   8A86 34 Ø6         L8A86   PSHS  B,A                       SAVE ACCD
1272   8A88 EC 84                 LDD   ,X                        TEST THE 2 BYTES POINTED TO BY X
1273   8A8A 35 Ø6                 PULS  A,B                       RESTORE ACCD
1274   8A8C 26 Ø2                 BNE   L8A9Ø                     BRANCH IF NOT END OF PROGRAM
1275   8A8E 32 62                 LEAS  $Ø2,S                     PURGE RETURN ADDRESS FROM STACK
1276   8A9Ø 39            L8A9Ø   RTS
1277   8A91 9E 19         L8A91   LDX   TXTTAB                    GET START OF BASIC PROGRAM
1278   8A93 3Ø 1F                 LEAX  $-Ø1,X                    MOVE POINTER BACK ONE
1279   8A95 3Ø Ø1         L8A95   LEAX  $Ø1,X                     MOVE POINTER UP ONE
128Ø   8A97 8D ED                 BSR   L8A86                     RETURN IF END OF PROGRAM
1281   8A99 3Ø Ø3         L8A99   LEAX  $Ø3,X                     SKIP OVER NEXT LINE ADDRESS AND LINE NUMBER
1282   8A9B 3Ø Ø1         L8A9B   LEAX  $Ø1,X                     MOVE POINTER TO NEXT CHARACTER
1283   8A9D A6 84                 LDA   ,X                        CHECK CURRENT CHARACTER
1284   8A9F 27 F4                 BEQ   L8A95                     BRANCH IF END OF LINE
1285   8AA1 9F ØF                 STX   TEMPTR                    SAVE CURRENT POINTER
1286   8AA3 4A                    DECA                            =
1287   8AA4 27 ØC                 BEQ   L8AB2                     =BRANCH IF START OF PACKED NUMERIC LINE
1288   8AA6 4A                    DECA                            *
1289   8AA7 27 2A                 BEQ   L8AD3                     *BRANCH IF LINE NUMBER EXISTS
1290   8AA9 4A                    DECA                            =
1291   8AAA 26 EF                 BNE   L8A9B                     =MOVE TO NEXT CHARACTER IF > 3
1292   8AAC 86 Ø3         L8AAC   LDA   #$Ø3                      * SET 1ST BYTE = 3 TO INDICATE LINE
1293   8AAE A7 8Ø                 STA   ,X+                       * NUMBER DOESN T CURRENTLY EXIST
1294   8ABØ 2Ø E7                 BRA   L8A99                     GO GET ANOTHER CHARACTER
1295   8AB2 EC Ø1         L8AB2   LDD   $Ø1,X                     GET MS BYTE OF LINE NUMBER
1296   8AB4 6A Ø2                 DEC   $Ø2,X                     DECREMENT ZERO CHECK BYTE
1297   8AB6 27 Ø1                 BEQ   L8AB9                     BRANCH IF MS BYTE <> Ø
1298   8AB8 4F                    CLRA                            CLEAR MS BYTE
1299   8AB9 E6 Ø3         L8AB9   LDB   $Ø3,X                     GET LS BYTE OF LINE NUMBER
13ØØ   8ABB 6A Ø4                 DEC   $Ø4,X                     DECREMENT ZERO CHECK FLAG
13Ø1   8ABD 27 Ø1                 BEQ   L8ACØ                     BRANCH IF IS BYTE <> Ø
13Ø2   8ABF 5F                    CLRB                            CLEAR LS BYTE
13Ø3   8ACØ ED Ø1         L8ACØ   STD   $Ø1,X                     SAVE BINARY LINE NUMBER
13Ø4   8AC2 DD 2B                 STD   BINVAL                    SAVE TRIAL LINE NUMBER
13Ø5   8AC4 BD AD Ø1              JSR   LADØ1                     FIND RAM ADDRESS OF A BASIC LINE NUMBER
13Ø6   8AC7 9E ØF         L8AC7   LDX   TEMPTR                    GET BACK POINTER TO START OF PACKED LINE NUMBER
13Ø7   8AC9 25 E1                 BLO   L8AAC                     BRANCH IF NO LINE NUMBER MATCH FOUND
13Ø8   8ACB DC 47                 LDD   V47                       GET START ADDRESS OF LINE NUMBER
13Ø9   8ACD 6C 8Ø                 INC   ,X+                       * SET 1ST BYTE = 2, TO INDICATE LINE NUMBER EXISTS IF CHECKING FOR
131Ø                      *                                       * EXISTENCE OF LINE NUMBER, SET IT = 1 IF INSERTING LINE NUMBERS
1311
1312   8ACF ED 84                 STD   ,X                        SAVE RAM ADDRESS OF CORRECT LINE NUMBER
1313   8AD1 2Ø C6                 BRA   L8A99                     GO GET ANOTHER CHARACTER
1314   8AD3 6F 84         L8AD3   CLR   ,X                        CLEAR CARRY FLAG AND 1ST BYTE
1315   8AD5 AE Ø1                 LDX   $Ø1,X                     POINT X TO RAM ADDRESS OF CORRECT LINE NUMBER
1316   8AD7 AE Ø2                 LDX   $Ø2,X                     PUT CORRECT LINE NUMBER INTO (X)
1317   8AD9 9F 47                 STX   V47                       SAVE IT TEMPORARILY
1318   8ADB 2Ø EA                 BRA   L8AC7                     GO INSERT IT INTO BASIC LINE
1319   8ADD 9E 19         L8ADD   LDX   TXTTAB                    GET BEGINNING OF BASIC PROGRAM
132Ø   8ADF 2Ø Ø4                 BRA   L8AE5
1321   8AE1 9E A6         L8AE1   LDX   CHARAD                    *GET CURRENT INPUT POINTER
1322   8AE3 3Ø Ø1                 LEAX  $Ø1,X                     *AND BUMP IT ONE
1323   8AE5 8D 9F         L8AE5   BSR   L8A86                     RETURN IF END OF PROGRAM
1324   8AE7 3Ø Ø2                 LEAX  $Ø2,X                     SKIP PAST NEXT LINE ADDRESS
1325   8AE9 3Ø Ø1         L8AE9   LEAX  $Ø1,X                     ADVANCE POINTER BY ONE
1326   8AEB 9F A6         L8AEB   STX   CHARAD                    SAVE NEW BASIC INPUT POINTER
1327   8AED 9D 9F         L8AED   JSR   GETNCH                    GET NEXT CHARACTER FROM BASIC
1328   8AEF 4D            L8AEF   TSTA                            CHECK THE CHARACTER
1329   8AFØ 27 EF                 BEQ   L8AE1                     BRANCH IF END OF LINE
133Ø   8AF2 2A F9                 BPL   L8AED                     BRANCH IF NOT A TOKEN
1331   8AF4 9E A6                 LDX   CHARAD                    GET CURRENT INPUT POINTER
1332   8AF6 81 FF                 CMPA  #$FF                      IS THIS A SECONDARY TOKEN?
1333   8AF8 27 EF                 BEQ   L8AE9                     YES - IGNORE IT
1334   8AFA BD Ø1 AØ              JSR   RVEC22                    HOOK INTO RAM AND CHECK FOR USER ADDED TOKENS
1335   8AFD 81 A7                 CMPA  #$A7                      TOKEN FOR THEN?
1336   8AFF 27 12                 BEQ   L8B13                     YES
1337   8BØ1 81 84                 CMPA  #$84                      TOKEN FOR ELSE?
1338   8BØ3 27 ØE                 BEQ   L8B13                     YES
1339   8BØ5 81 81                 CMPA  #$81                      TOKEN FOR GO?
134Ø   8BØ7 26 E4                 BNE   L8AED                     NO
1341   8BØ9 9D 9F                 JSR   GETNCH                    GET A CHARACTER FROM BASIC
1342   8BØB 81 A5                 CMPA  #$A5                      TOKEN FOR TO?
1343   8BØD 27 Ø4                 BEQ   L8B13                     YES
1344   8BØF 81 A6                 CMPA  #$A6                      TOKEN FOR SUB?
```

```
1345   8B11 26 D8                BNE   L8AEB                     NO
1346   8B13 9D 9F       L8B13    JSR   GETNCH                    GET A CHARACTER FROM BASIC
1347   8B15 25 04                BLO   L8B1B                     BRANCH IF NUMERIC
1348   8B17 9D A5       L8B17    JSR   GETCCH                    GET CURRENT BASIC INPUT CHARRACTER
1349   8B19 20 D4                BRA   L8AEF                     KEEP CHECKING THE LINE
1350   8B1B 9E A6       L8B1B    LDX   CHARAD                    GET CURRENT INPUT ADDRESS
1351   8B1D 34 10                PSHS  X                         SAVE IT ON THE STACK
1352   8B1F BD AF 67             JSR   LAF67                     CONVERT DECIMAL BASIC NUMBER TO BINARY
1353   8B22 9E A6                LDX   CHARAD                    GET CURRENT INPUT POINTER
1354   8B24 A6 82       L8B24    LDA   ,-X                       GET PREVIOUS INPUT CHARACTER
1355   8B26 BD 90 AA             JSR   L90AA                     CLEAR CARRY IF NUMERIC INPUT VALUE
1356   8B29 25 F9                BLO   L8B24                     BRANCH IF NON-NUMERIC
1357   8B2B 30 01                LEAX  $01,X                     MOVE POINTER UP ONE
1358   8B2D 1F 10                TFR   X,D                       NOW ACCD POINTS TO ONE PAST END OF LINE NUMBER
1359   8B2F E0 61                SUBB  $01,S                     SUBTRACT PRE-NUMERIC POINTER LS BYTE
1360   8B31 C0 05                SUBB  #$05                      MAKE SURE THERE ARE AT LEAST 5 CHARACTERS IN THE NUMERIC LINE
1361                     *
1362   8B33 27 20                BEQ   L8B55                     BRANCH IF EXACTLY 5
1363   8B35 25 0A                BLO   L8B41                     BRANCH IF < 5
1364   8B37 33 84                LEAU  ,X                        TRANSFER X TO U
1365   8B39 50                   NEGB                            NEGATE B
1366   8B3A 30 85                LEAX  B,X                       MOVE X BACK B BYTES
1367   8B3C BD 89 B8             JSR   L89B8                     *MOVE BYTES FROM (U) TO (X) UNTIL
1368                     *                                       *U = END OF BASIC; (I) = NEW END OF BASIC
1369   8B3F 20 14                BRA   L8B55
1370                     * FORCE FIVE BYTES OF SPACE FOR THE LINE NUMBER
1371   8B41 9F 47       L8B41    STX   V47                       SAVE END OF NUMERIC VALUE
1372   8B43 9E 1B                LDX   VARTAB                    GET END OF BASIC PROGRAM
1373   8B45 9F 43                STX   V43                       SAVE IT
1374   8B47 50                   NEGB                            NEGATE B
1375   8B48 30 85                LEAX  B,X                       ADD IT TO END OF NUMERIC POiNTER
1376   8B4A 9F 41                STX   V41                       SAVE POINTER
1377   8B4C 9F 1B                STX   VARTAB                    STORE END OF BASIC PROGRAM
1378   8B4E BD AC 1E             JSR   LAC1E                     ACCD = TOP OF ARRAYS - CHECK FOR ENOUGH ROOM
1379   8B51 9E 45                LDX   V45                       * GET AND SAVE THE
1380   8B53 9F A6                STX   CHARAD                    * NEW CURRENT INPUT POINTER
1381   8B55 35 10       L8B55    PULS  X                         RESTORE POINTER TO START OF NUMERIC VALUE
1382   8B57 86 01                LDA   #$01                      NEW LINE NUMBER FLAG
1383   8B59 A7 84                STA   ,X                        * SAVE NEW LINE FLAG
1384   8B5B A7 02                STA   $02,X                     *
1385   8B5D A7 04                STA   $04,X                     *
1386   8B5F D6 2B                LDB   BINVAL                    GET MS BYTE OF BINARY LINE NUMBER
1387   8B61 26 04                BNE   L8B67                     BRANCH IF IT IS NOT ZERO
1388   8B63 C6 01                LDB   #$01                      SAVE A 1 IF BYTE IS 0; OTHERWISE, BASIC WILL
1389                                                             THINK IT IS THE END OF A LINE
1390   8B65 6C 02                INC   $02,X                     IF 2,X = 2, THEN PREVIOUS BYTE WAS A ZERO
1391   8B67 E7 01       L8B67    STB   $01,X                     SAVE MS BYTE OF BINARY LINE NUMBER
1392   8B69 D6 2C                LDB   BINVAL+1                  GET IS BYTE OF BINARY LINE NUMBER
1393   8B6B 26 04                BNE   L8B71                     BRANCH IF NOT A ZERO BYTE
1394   8B6D C6 01                LDB   #$01                      SAVE A 1 IF BYTE IS A 0
1395   8B6F 6C 04                INC   $04,X                     IF 4,X = 2, THEN PREVIOUS BYTE WAS A 0
1396   8B71 E7 03       L8B71    STB   $03,X                     SAVE LS BYTE OF BINARY LINE NUMBER
1397   8B73 9D A5                JSR   GETCCH                    GET CURRENT INPUT CHARACTER
1398   8B75 81 2C                CMPA  #','                      IS IT A COMMA?
1399   8B77 27 9A                BEQ   L8B13                     YES - PROCESS ANOTHER NUMERIC VALUE
1400   8B79 20 9C                BRA   L8B17                     NO - GO GET AND PROCESS AN INPUT CHARACTER
1401   8B7B 9E 19       L8B7B    LDX   TXTTAB                    POINT X TO START OF BASIC PROGRAM
1402   8B7D 30 1F                LEAX  $-01,X                    MOVE POINTER BACK ONE
1403   8B7F 30 01       L8B7F    LEAX  $01,X                     MOVE POINTER UP ONE
1404   8B81 EC 02                LDD   $02,X                     GET ADDRESS OF NEXT LINE
1405   8B83 DD 68                STD   CURLIN                    SAVE IT IN CURLIN
1406   8B85 BD 8A 86             JSR   L8A86                     RETURN IF END OF PROGRAM
1407   8B88 30 03                LEAX  $03,X                     SKIP OVER ADDRESS OF NEXT LINE AND 1ST BYTE OF LINE NUMBER
1408   8B8A 30 01       L8B8A    LEAX  $01,X                     MOVE POINTER UP ONE
1409   8B8C A6 84       L8B8C    LDA   ,X                        GET CURRENT CHARACTER
1410   8B8E 27 EF                BEQ   L8B7F                     BRANCH IF END OF LINE
1411   8B90 4A                   DECA                            INPUT CHARACTER = 1? - VALID LINE NUMBER
1412   8B91 27 1B                BEQ   L8BAE                     YES
1413   8B93 80 02                SUBA  #$02                      INPUT CHARACTER 3? - UL LINE NUMBER
1414   8B95 26 F3                BNE   L8B8A                     NO
1415   8B97 34 10                PSHS  X                         SAVE CURRENT POSITION OF INPUT POINTER
1416   8B99 8E 8B D8             LDX   #L8BD9-1                  POINT X TO  UL  MESSAGE
1417   8B9C BD B9 9C             JSR   STRINOUT                  PRINT STRING TO THE SCREEN
1418   8B9F AE E4                LDX   ,S                        GET INPUT POINTER
1419   8BA1 EC 01                LDD   $01,X                     GET THE UNDEFINED LINE NUMBER
1420   8BA3 BD BD CC             JSR   LBDCC                     CONVERT NUMBER IN ACCD TO DECIMAL AND DISPLAY IT
1421   8BA6 BD BD C5             JSR   LBDC5                     PRINT  IN XXXX  XXXX = CURRENT LINE NUMBER
1422   8BA9 BD B9 58             JSR   LB958                     SEND A CR TO CONSOLE OUT
1423   8BAC 35 10                PULS  X                         GET INPUT POINTER BACK
1424   8BAE 34 10       L8BAE    PSHS  X                         SAVE CURRENT POSITION OF INPUT POINTER
1425   8BB0 EC 01                LDD   $01,X                     LOAD ACCD WITH BINARY VALUE OF LINE NUMBER
1426   8BB2 DD 52                STD   FPA0+2                    SAVE IN BOTTOM 2 BYTES OF FPA0
1427   8BB4 BD 88 0E             JSR   L880E                     ADJUST REST OF FPA0 AS AN INTEGER
1428   8BB7 BD BD D9             JSR   LBDD9                     CONVERT FPA0 TO ASCII, STORE IN LINE NUMBER
1429   8BBA 35 40                PULS  U                         LOAD U WITH PREVIOUS ADDRESS OF INPUT POINTER
1430   8BBC C6 05                LDB   #$05                      EACH EXPANDED LINE NUMBER USES 5 BYTES
1431   8BBE 30 01       L8BBE    LEAX  $01,X                     MOVE POINTER FORWARD ONE
1432   8BC0 A6 84                LDA   ,X                        GET AN ASCII BYTE
1433   8BC2 27 05                BEQ   L8BC9                     BRANCH IF END OF NUMBER
1434   8BC4 5A                   DECB                            DECREMENT BYTE COUNTER
1435   8BC5 A7 C0                STA   ,U+                       STORE ASCII NUMBER IN BASIC LINE
1436   8BC7 20 F5                BRA   L8BBE                     CHECK FOR ANOTHER DIGIT
1437   8BC9 30 C4       L8BC9    LEAX  ,U                        TRANSFER NEW LINE POINTER TO (X)
1438   8BCB 5D                   TSTB                            DOES THE NEW LINE NUMBER REQUIRE 5 BYTES?
1439   8BCC 27 BE                BEQ   L8B8C                     YES - GO GET ANOTHER INPUT CHARACTER
1440   8BCE 31 C4                LEAY  ,U                        SAVE NEW LINE POINTER IN Y
```

```
1441   8BD0 33 C5                LEAU  B,U                          POINT U TO END OF 5 BYTE PACKED LINE NUMBER BLOCK
1442   8BD2 BD 89 B8             JSR   L89B8                        MOVE BYTES FROM (U) TO (X) UNTIL END OF PROGRAM
1443   8BD5 30 A4                LEAX  ,Y                           LOAD (X) WITH NEW LINE POINTER
1444   8BD7 20 B3                BRA   L8B8C                        GO GET ANOTHER INPUT CHARACTER
1445
1446   8BD9 55 4C 20          L8BD9  FCC   'UL '                    UNKNOWN LINE NUMBER MESSAGE
1447   8BDC 00                   FCB   0
1448
1449                          * CONVERT AN INTEGER INTO AN ASCII STRING AND PRINT IT ON THE SCREEN
1450   8BDD BD B7 40          HEXDOL JSR   LB740                    CONVERT FPA0 INTO A POSITIVE 2 BYTE INTEGER
1451   8BE0 8E 03 D9             LDX   #STRBUF+2                    POINT TO TEMPORARY BUFFER
1452   8BE3 C6 04                LDB   #$04                         CONVERT 4 NIBBLES
1453   8BE5 34 04             L8BE5  PSHS  B                        SAVE NIBBLE COUNTER
1454   8BE7 5F                   CLRB                               CLEAR CARRY FLAG
1455   8BE8 86 04                LDA   #$04                         4 SHIFTS
1456   8BEA 08 53             L8BEA  ASL   FPA0+3                   * SHIFT BOTTOM TWO BYTES OF
1457   8BEC 09 52                ROL   FPA0+2                       * FPA0 LEFT ONE BIT (X2)
1458   8BEE 59                   ROLB                               IF OVERFLOW, ACCB <> 0
1459   8BEF 4A                   DECA                               * DECREMENT SHIFT COUNTER AND
1460   8BF0 26 F8                BNE   L8BEA                        * BRANCH IF NOT DONE
1461   8BF2 5D                   TSTB                               CHECK FOR OVERFLOW
1462   8BF3 26 0A                BNE   L8BFF                        BRANCH IF OVERFLOW
1463   8BF5 A6 E4                LDA   ,S                           * GET NIBBLE COUNTER,
1464   8BF7 4A                   DECA                               * DECREMENT IT AND
1465   8BF8 27 05                BEQ   L8BFF                        * BRANCH IF DONE
1466   8BFA 8C 03 D9             CMPX  #STRBUF+2                    = DO NOT DO A CONVERSION UNTIL A NON-ZERO
1467   8BFD 27 0C                BEQ   L8C0B                        = BYTE IS FOUND - LEADING ZERO SUPPRESSION
1468   8BFF CB 30             L8BFF  ADDB  #'0'                     ADD IN ASCII ZERO
1469   8C01 C1 39                CMPB  #'9'                         COMPARE TO ASCII 9
1470   8C03 23 02                BLS   L8C07                        BRANCH IF < 9
1471   8C05 CB 07                ADDB  #'A'-('9'+1)                 ADD ASCII OFFSET IF HEX LETTER
1472   8C07 E7 80             L8C07  STB   ,X+                      STORE HEX VALUE AND ADVANCE POINTER
1473   8C09 6F 84                CLR   ,X                           CLEAR NEXT BYTE - END OF STRING FLAG
1474   8C0B 35 04             L8C0B  PULS  B                        * GET NIBBLE COUNTER,
1475   8C0D 5A                   DECB                               * DECREMENT IT AND
1476   8C0E 26 D5                BNE   L8BE5                        * BRANCH IF NOT DONE
1477   8C10 32 62                LEAS  $02,S                        PURGE RETURN ADDRESS OFF OF STACK
1478   8C12 8E 03 D8             LDX   #STRBUF+1                    RESET POINTER
1479   8C15 7E B5 18             JMP   LB518                        SAVE STRING ON STRING STACK
1480
1481                          * DLOAD
1482   8C18 BD A4 29          DLOAD  JSR   LA429                    CLOSE FILES
1483   8C1B 9D A5             L8C1B  JSR   GETCCH                   GET THE CURRENT INPUT CHARACTER
1484   8C1D 80 4D                SUBA  #'M'                         CHECK FOR DLOADM
1485   8C1F 34 02                PSHS  A                            SAVE DLOADM (=0), OLOAD (<>0) FLAG
1486   8C21 26 02                BNE   L8C25                        BRANCH IF OLOAD
1487   8C23 9D 9F                JSR   GETNCH                       GET AN INPUT CHAR FROM BASIC
1488   8C25 BD A5 78          L8C25  JSR   LA578                    GET THE NAME OF FILE FROM BASIC
1489   8C28 9D A5                JSR   GETCCH                       GET CURRENT INPUT CHAR FROM BASIC
1490   8C2A 27 18                BEQ   L8C44                        BRANCH IF END OF LINE
1491   8C2C BD B2 6D             JSR   SYNCOMMA                     SYNTAX CHECK FOR COMMA
1492   8C2F 81 2C                CMPA  #','                         CHECK FOR TWO CONSECUTIVE COMMAS
1493   8C31 27 11                BEQ   L8C44                        *BRANCH IF,, - IF THIS CASE IS SELECTED
1494                          *                                    *THE BAUD DELAY MUST HAVE BEEN PREVIOUSLY STORED IN DIBAUD
1495   8C33 BD B7 0B             JSR   EVALEXPB                     EVAL EXPR, RETURN VALUE IN ACCB
1496   8C36 86 B0                LDA   #44*4                        DELAY VALUE FOR 300 BAUD
1497   8C38 5D                   TSTB                               WAS ARGUMENT = 0?
1498   8C39 27 07                BEQ   L8C42                        YES - 300 BAUD
1499   8C3B 86 2C                LDA   #44                          DELAY VALUE FOR 1200 BAUD
1500   8C3D 5A                   DECB                               CHECK FOR ARGUMENT OF 1
1501   8C3E 10 26 28 08          LBNE  LB44A                         FC  ERROR IF NOT ZERO OR ONE OR COMMA
1502   8C42 97 E6             L8C42  STA   DLBAUD                   SAVE DELAY VALUE
1503   8C44 BD 8C D0          L8C44  JSR   L8CD0                    TRANSMIT FILE NAME AND READ IN FILE STATUS
1504   8C47 34 02                PSHS  A                            SAVE ACCA
1505   8C49 86 FD                LDA   #-3                          DLOAD DEVICE NUMBER TO -3
1506   8C4B 97 6F                STA   DEVNUM                       SET DEVICE NUMBER TO DLOAD
1507   8C4D 35 02                PULS  A                            RESTORE ACCA
1508   8C4F 6D E0                TST   ,S+                          DLOAD OR DLOADM?
1509   8C51 27 32                BEQ   L8C85                        DLOADM
1510
1511                          * READ IN A DLOAD FILE
1512   8C53 BD A5 C7             JSR   LA5C7                        CHECK FOR END OF LINE - SYNTAX ERROR IF NOT
1513   8C56 5D                   TSTB                               CHECK ASCII FLAG
1514   8C57 27 06                BEQ   L8C5F                         FM  ERROR IF NOT ASCII
1515   8C59 BD AD 19             JSR   LAD19                        GO DO A  NEW
1516   8C5C 7E AC 7C             JMP   LAC7C                        *JUMP BACK TO BASIC S MAIN INPUT LOOP;
1517                          *                                    *DLOAD FILES MUST BE ASCII FILES
1518   8C5F 7E A6 16          L8C5F  JMP   LA616                     BAD FILE MODE  ERROR
1519
1520                          * EXBAS CLOAD PROCESSOR
1521   8C62 9D 9F             L8C62  JSR   GETNCH                   GET A CHAR FROM BASIC
1522   8C64 81 4D                CMPA  #'M'                         CHECK FOR CLOADM
1523   8C66 10 26 18 2E          LBNE  LA498                        GO DO A CLOAD
1524   8C6A 0F 78                CLR   FILSTA                       CLOSE FILES
1525   8C6C 9D 9F                JSR   GETNCH                       GET A CHAR FROM BASIC
1526   8C6E BD A5 78             JSR   LA578                        STRIP A FILENAME OFF OF THE BASIC LINE
1527   8C71 BD A6 48             JSR   LA648                        SEARCH FOR FILE
1528   8C74 7D 01 E4             TST   CASBUF+10                    CHECK FILE MODE
1529   8C77 10 27 18 8A          LBEQ  LA505                        BRANCH TO CLOADM IF NOT BLOCK LOAD
1530   8C7B FE 01 E2             LDU   CASBUF+8                     SAVE FILE TYPE AND ASCII FLAG IN U
1531   8C7E 0A 6F                DEC   DEVNUM                       SET DEVICE NUMBER TO -1 (CASSETTE)
1532   8C80 BD A6 35             JSR   LA635                        GO READ IN A DATA BLOCK
1533   8C83 1F 30                TFR   U,D                          PUT FILE TYPE & ASCII FLAG BACK IN ACCD
1534
1535                          * STRIP A LOAD OFFSET FROM THE BASIC LINE, THEN LOAD IN BLOCKS OF
1536                          * DATA (CLOADM,DLOADM) WHICH ARE PRECEEDED BY A 5 BYTE PRE OR POST-
```

```
1537                        $ AMBLE. THE PREAMBLE CONTAINS A BLOCK LENGTH AND A LOAD ADDRESS SO
1538                        * THAT ANY NUMBER OF NON-CONTIGUOUS BLOCKS MAY BE LOADED. THE POST-
1539                        * AMBLE WILL TERMINATE THE LOADING PROCESS AND PROVIDE A TRANSFER ADDRESS
1540  8C85 83 02 00         L8C85   SUBD  #$200                    * CHECK FILE STATUS;
1541  8C88 26 D5                    BNE   L8C5F                    *  FM  ERROR IF MODE <> 2 OR TYPE <> 0
1542  8C8A 9E 8A                    LDX   ZERO                     ZERO THE X REG - DEFAULT OFFSET
1543  8C8C 9D A5                    JSR   GETCCH                   GET CURRENT INPUT CHARACTER
1544  8C8E 27 06                    BEQ   L8C96                    BRANCH IF END OF LINE
1545  8C90 BD B2 6D                 JSR   SYNCOMMA                 SYNTAX CHECK FOR COMMA
1546  8C93 BD B7 3D                 JSR   LB73D                    EVAL INTEGER EXPR - RETURN VALUE IN X
1547  8C96 9F D3            L8C96   STX   VD3                      SAVE OFFSET
1548  8C98 BD A5 C7                 JSR   LA5C7                    SYNTAX ERROR IF MORE CHARS ON LINE
1549  8C9B 8D 29            L8C9B   BSR   L8CC6                    GO GET EOF FLAG FROM CONSOLE IN
1550  8C9D 34 02                    PSHS  A                        SAVE IT ON THE STACK
1551  8C9F 8D 1E                    BSR   L8CBF                    * READ IN BLOCK LENGTH FROM CONSOLE IN
1552  8CA1 1F 02                    TFR   D,Y                      * AND SAVE IT IN Y
1553  8CA3 8D 1A                    BSR   L8CBF                    GET LOAD ADDRESS FROM CONSOLE IN
1554  8CA5 D3 D3                    ADDD  VD3                      ADD OFFSET TO LOAD ADDRESS
1555  8CA7 DD 9D                    STD   EXECJP                   SAVE IN EXEC ADDRESS
1556  8CA9 1F 01                    TFR   D,X                      SAVE LOAD ADDRESS IN X
1557  8CAB A6 E0                    LDA   ,S+                      GET EOF FLAG FROM STACK
1558  8CAD 10 26 17 7C              LBNE  LA42D                    CLOSE FILES IF POSTAMBLE BLOCK
1559  8CB1 8D 13            L8CB1   BSR   L8CC6                    GET A CHARACTER FROM CONSOLE IN
1560  8CB3 A7 84                    STA   ,X                       SAVE IT IN RAM
1561  8CB5 A1 80                    CMPA  ,X+                      COMPARE SAVED BYTE TO ACTUAL BYTE
1562  8CB7 26 14                    BNE   L8CCD                    'IO  ERROR IF NOT = (SAVED IN ROM OR BAD RAM)
1563  8CB9 31 3F                    LEAY  $-01,Y                   DECREMENT BYTE COUNT
1564  8CBB 26 F4                    BNE   L8CB1                    READ MORE CHARACTERS
1565  8CBD 20 DC                    BRA   L8C9B                    LOOK FOR ANOTHER BLOCK OF DATA
1566                        * GET 2 CHARACTERS - RETURN THEM IN ACCD
1567  8CBF 8D 00            L8CBF   BSR   L8CC1                    GET A CHARACTER IN ACCB
1568  8CC1 8D 03            L8CC1   BSR   L8CC6                    GET A CHARACTER IN ACCA
1569  8CC3 1E 89                    EXG   A,B                      SAVE IT IN ACCB
1570  8CC5 39               L8CC5   RTS
1571
1572  8CC6 BD A1 76         L8CC6   JSR   LA176                    GET A CHARACTER FROM CONSOLE IN
1573  8CC9 0D 70                    TST   CINBFL                   IS FILE EMPTY?
1574  8CCB 27 F8                    BEQ   L8CC5                    RETURN IF NOT EMPTY
1575  8CCD 7E A6 19         L8CCD   JMP   LA619                     IO  ERROR IF EMPTY
1576  8CD0 8D 42            L8CD0   BSR   L8D14                    TRANSMIT FILE NAME, RETURN FILE STATUS
1577  8CD2 34 06                    PSHS  B,A                      SAVE FILE STATUS ON STACK
1578  8CD4 4C                       INCA                           CHECK FILE TYPE
1579  8CD5 27 06                    BEQ   L8CDD                     NE  ERROR IF FILE NOT FOUND
1580  8CD7 DE 8A                    LDU   ZERO                     ZERO U REG :FIRST BLOCK NUMBER
1581  8CD9 8D 09                    BSR   L8CE4                    READ IN 128 CHARACTERS
1582  8CDB 35 86                    PULS  A,B,PC                   GET FILE STATUS BACK AND RETURN
1583  8CDD C6 34            L8CDD   LDB   #2*26                     NE  ERROR
1584  8CDF 7E AC 46                 JMP   LAC46                    GO TO ERROR SERVICING ROUTINE
1585                        * REFILL CONSOLE IN CHARACTER BUFFER FROM DLOAD
1586  8CE2 DE 7E            L8CE2   LDU   CBUFAD                   GET BLOCK NUMBER
1587  8CE4 30 41            L8CE4   LEAX  $01,U                    * INCREMENT BLOCK NUMBER
1588  8CE6 9F 7E                    STX   CBUFAD                   * AND SAVE IT
1589  8CE8 8E 01 DA                 LDX   #CASBUF                  USE CASBUF AS DLOAD INPUT BUFFER
1590  8CEB BD 8D 7C                 JSR   L8D7C                    READ 128 CHARACTERS (ONE BLOCK) INTO BUFFER
1591  8CEE 7E A6 44                 JMP   LA644                    RESET CONSOLE IN BUFFER
1592
1593                        * CONSOLE IN RAM HOOK
1594  8CF1 96 6F            XVEC4   LDA   DEVNUM                   GET DEVICE NUMBER
1595  8CF3 81 FD                    CMPA  #-3                      DLOAD DEVICE NUMBER
1596  8CF5 26 0A                    BNE   L8D01                    BRANCH IF NOT OLOAD
1597  8CF7 32 62                    LEAS  $02,S                    PURGE 1ST RETURN ADDR FROM STACK
1598  8CF9 0F 70                    CLR   CINBFL                   RESET EMPTY/FULL FLAG
1599  8CFB 0D 79                    TST   CINCTR                   ANY CHARACTERS LEFT IN BUFFER?
1600  8CFD 26 03                    BNE   L8D02                    YES, GO GET ONE
1601  8CFF 03 70                    COM   CINBFL                   SET EMPTY/FULL FLAG TO EOF
1602  8D01 39               L8D01   RTS
1603  8D02 34 74            L8D02   PSHS  U,Y,X,B                  SAVE REGISTERS
1604  8D04 9E 7A                    LDX   CINPTR                   GET CONSOLE IN CHARACTER BUFFER
1605  8D06 A6 80                    LDA   ,X+                      GET A CHARACTER
1606  8D08 34 02                    PSHS  A                        SAVE IT ON THE STACK
1607  8D0A 9F 7A                    STX   CINPTR                   SAVE NEW CHARACTER BUFFER
1608  8D0C 0A 79                    DEC   CINCTR                   DECREMENT CHARACTER COUNTER
1609  8D0E 26 02                    BNE   L8D12                    RETURN IF BUFFER NOT EMPTY
1610  8D10 8D D0                    BSR   L8CE2                    GO REFILL THE CHARACTER BUFFER
1611  8D12 35 F6            L8D12   PULS  A,B,X,Y,U,PC             RESTORE REGISTERS AND RETURN
1612                        * TRANSMIT FILE NAME - READ FILE STATUS FROM SENDER
1613  8D14 4F               L8D14   CLRA                           RESET ATTEMPT COUNTER
1614  8D15 34 16                    PSHS  X,B,A                    SAVE SPACE ON STACK FOR TEMP VARIABLES
1615  8D17 31 E4                    LEAY  ,S                       STACK TO Y (TFR S,Y) - SAVE VARIABLE POINTER
1616  8D19 20 02                    BRA   L8D1D
1617  8D1B 8D 2B            L8D1B   BSR   L8D48                    INCREMENT ATTEMPT COUNTER
1618  8D1D 86 8A            L8D1D   LDA   #$8A                     * GET FILE REQUEST CONTROL CODE
1619  8D1F 8D 37                    BSR   L8D58                    * AND TRANSMIT IT
1620  8D21 26 F8                    BNE   L8D1B                    BRANCH IF NO ECHO OR ERROR
1621  8D23 8E 01 D2                 LDX   #CFNBUF+1                POINT TO CASS FILE NAME BUFFER
1622  8D26 A6 80            L8D26   LDA   ,X+                      GET CHARACTER FROM NAME BUFFER
1623  8D28 BD 8E 04                 JSR   L8E04                    OUTPUT IT TO RS 232 PORT
1624  8D2B 8C 01 DA                 CMPX  #CFNBUF+9                COMPARE TO END OF BUFFER
1625  8D2E 26 F6                    BNE   L8D26                    LOOP UNTIL DONE
1626  8D30 8D 30                    BSR   L8D62                    OUTPUT CHECK BYTE AND LOOK FOR ACKNOWLEDGE
1627  8D32 26 E7                    BNE   L8D1B                    TRANSMIT NAME AGAIN IF NO ACKNOWLEDGE
1628  8D34 8D 3C                    BSR   L8D72                    GET FILE TYPE $FF = NOT FOUND
1629  8D36 26 E3                    BNE   L8D1B                    BRANCH IF ERROR
1630  8D38 A7 22                    STA   $02,Y                    SAVE FILE TYPE
1631  8D3A 8D 36                    BSR   L8D72                    READ ASCII FLAG
1632  8D3C 26 DD                    BNE   L8D1B                    BRANCH IF ERROR
```

```
1633   8D3E A7 23                STA   $Ø3,Y                          SAVE ASCII FLAG
1634   8D4Ø 8D 29                BSR   L8D6B                          READ CHECK BYTE FROM SENDER
1635   8D42 26 D7                BNE   L8D1B                          BRANCH IF NO CHECKBYTE MATCH
1636   8D44 32 62                LEAS  $Ø2,S                          PURGE ATTEMPT COUNTER & CHECK BYTE FROM STACK
1637   8D46 35 86                PULS  A,B,PC                         RETURN FILE STATUS IN ACCD
1638                      * INCREMENT ATTEMPT COUNTER - AFTER 5 TRIES, GIVE UP (IO ERROR)
1639   8D48 6C A4          L8D48 INC   ,Y                             INCREMENT ATTEMPT COUNTER
1640   8D4A A6 A4                LDA   ,Y                             GET ATTEMPT COUNTER
1641   8D4C 81 Ø5                CMPA  #$Ø5                           IS THIS THE FIFTH TRY?
1642   8D4E 25 1A                BLO   L8D6A                          NO
1643   8D5Ø 86 BC                LDA   #$BC                           YES ; TIME TO QUIT-GET ABORT CODE
1644   8D52 BD 8E ØC             JSR   L8E0C                          OUTPUT ABORT CODE OVER THE RS 232 PORT
1645   8D55 7E A6 19             JMP   LA619                           IO  ERROR
1646                      * ECHO CHECK - OUTPUT A CHARACTER, READ A CHARACTER AND
1647                      * COMPARE IT TO THE OUTPUT CHARACTER. Z=Ø IF NO MATCH OR ERROR
1648   8D58 34 Ø2          L8D58 PSHS  A                              SAVE COMPARE CHARACTER ON STACK
1649   8D5A 8D 5C                BSR   L8DB8                          SEND A CHARACTER OUT
1650   8D5C 26 Ø2                BNE   L8D6Ø                          BRANCH IF READ ERROR
1651   8D5E A1 E4                CMPA  ,S                             COMPARE RECEIVED CHARACTER TO TRANSMITTED CHARACTER
1652   8D6Ø 35 82          L8D6Ø PULS  A,PC                           RESTORE COMPARE CHARACTER AND RETURN
1653                      * TRANSMIT XOR CHECKBYTE AND READ ACKNOWLEGE ($C8)
1654                      * RETURN ZERO FLAG SET IF NO ERROR AND ACKNOWLEGE
1655   8D62 A6 21          L8D62 LDA   $Ø1,Y                          GET XOR CHECKBYTE
1656   8D64 8D 52                BSR   L8DB8                          OUTPUT XOR CHECKBYTE AND READ ONE BYTE
1657   8D66 26 Ø2                BNE   L8D6A                          BRANCH IF READ ERROR
1658   8D68 81 C8                CMPA  #$C8                           COMPARE INPUT BYTE TO ACKNOWLEDGE CODE
1659   8D6A 39             L8D6A RTS
1660                      * READ XOR CHECKBYTE THEN LOAD ACCUMULATED XOR CHECKBYTE.
1661                      * SET ZERO FLAG IF ACCUMULATED CHECK BYTE = Ø
1662   8D6B 8D Ø5          L8D6B BSR   L8D72                          INPUT A CHARACTER FROM RS 232
1663   8D6D 26 FB                BNE   L8D6A                          BRANCH IF TIMEOUT
1664   8D6F A6 21                LDA   $Ø1,Y                          GET CHECK BYTE
1665   8D71 39                   RTS
1666   8D72 8D 48          L8D72 BSR   L8DBC                          INPUT A CHARACTER FROM RS 232
1667   8D74 34 Ø3                PSHS  A,CC                           SAVE CHARACTER AND ZERO FLAG ON STACK
1668   8D76 A8 21                EORA  $Ø1,Y                          * EXCLUSIVE OR INPUT
1669   8D78 A7 21                STA   $Ø1,Y                          * CHARACTER WITH CHECK BYTE
1670   8D7A 35 83                PULS  CC,A,PC                        RESTORE CHARACTER AND ZERO FLAG
1671                      * REQUEST A BLOCK FROM RS 232 INPUT -
1672                      * LOAD THE RECEIVED DATA INTO THE BUFFER POINTED TO BY X
1673                      * U REGFISTER CONTAINS THE BLOCK NUMBER; RETURN Z=1 IF NO
1674                      * ERRORS, CHARACTER COUNT IN ACCA; ACCA = Ø IF FILE EMPTY
1675   8D7C 4F             L8D7C CLRA                                 RESET ATTEMPT COUNTER
1676   8D7D 34 76                PSHS  U,Y,X,B,A                      SAVE SPACE FOR STACK BUFFER
1677   8D7F 68 67                ASL   $Ø7,S                          * 6,7 S (U REG) CONTAIN THE 14 BIT BLOCK NUMBER -
1678   8D81 69 66                ROL   $Ø6,S                          * PUT THE BOTTOM 7 BITS IN 7,S AND THE
1679   8D83 64 67                LSR   $Ø7,S                          * TOP SEVEN BITS IN 6,S
1680   8D85 31 E4                LEAY  ,S                             STACK POINTER TO Y (TFR S,Y)
1681   8D87 2Ø Ø2                BRA   L8D8B
1682   8D89 8D BD          L8D89 BSR   L8D48                          INCREMENT ATTEMPT COUNTER
1683   8D8B 86 97          L8D8B LDA   #$97                           * TRANSMIT A BLOCK REQUEST CODE, ECHO
1684   8D8D 8D C9                BSR   L8D58                          * CHECK AND RESET CHECK BYTE
1685   8D8F 26 F8                BNE   L8D89                          BRANCH IF NO MATCH OR ERROR
1686   8D91 A6 26                LDA   $Ø6,Y                          * SEND OUT HIGH ORDER SEVEN BITS
1687   8D93 8D 6F                BSR   L8E04                          * OF BLOCK NUMBER
1688   8D95 A6 27                LDA   $Ø7,Y                          = SEND OUT LOW ORDER SEVEN BITS
1689   8D97 8D 6B                BSR   L8E04                          = OF BLOCK NUMBER
1690   8D99 8D C7                BSR   L8D62                          TRANSMIT CHECK BYTE AND GET ACKNOWLEDGE
1691   8D9B 26 EC                BNE   L8D89                          BRANCH IF ERROR OR NO ACKNOWLEDGE
1692   8D9D 8D D3                BSR   L8D72                          READ CHARACTER COUNT
1693   8D9F 26 E8                BNE   L8D89                          BRANCH IF READ ERROR
1694   8DA1 A7 24                STA   $Ø4,Y                          SAVE CHARACTER COUNT IN STACK VARIABLES
1695   8DA3 AE 22                LDX   $Ø2,Y                          GET VARIABLES POINTER FROM STACK BUFFER
1696                      * READ IN A BLOCK OF 128 CHARACTERS - THE HOST WILL TRANSMIT 128
1697                      * CHARACTERS REGARDLESS OF HOW MANY ARE VALID. OF HOW MANY ARE VALID.
1698   8DA5 C6 8Ø                LDB   #128                           128 CHARACTERS/BUFFER
1699   8DA7 8D C9          L8DA7 BSR   L8D72                          READ A CHARACTER
1700   8DA9 26 DE                BNE   L8D89                          RESTART PROCESS IF READ ERROR
1701   8DAB A7 8Ø                STA   ,X+                            SAVE THE CHARACTER IN BUFFER
1702   8DAD 5A                   DECB                                 DECREMENT CHARACTER COUNTER
1703   8DAE 26 F7                BNE   L8DA7                          BRANCH IF NOT DONE
1704   8DBØ 8D B9                BSR   L8D6B                          INPUT XOR CHECKBYTE
1705   8DB2 26 D5                BNE   L8D89                          RESTART PROCESS IF READ ERROR OR BAD CHECKBYTE
1706   8DB4 32 64                LEAS  $Ø4,S                          PURGE ATTEMPT COUNTER, CHECK BYTE AND LOAD ADDRESS FROM STACK
1707   8DB6 35 96                PULS  A,B,X,PC                       RETURN CHARACTER COUNT IN ACCA
1708   8DB8 6F 21          L8DB8 CLR   $Ø1,Y                          CLEAR CHECK BYTE
1709   8DBA 8D 5Ø                BSR   L8E0C                          OUTPUT A CHARACTER OVER RS 232 PORT
1710                      $ READ A CHARACTER FROM THE RS 232 INPUT PORT.
1711                      * RETURN CHARACTER IN ACCA.  EXIT WITH Z=Ø
1712                      * FOR TIMEOUT ERROR, Z = 1 FOR VALID BYTE INPUT.
1713   8DBC 4F             L8DBC CLRA                                 CLEAR ATTEMPT COUNTER
1714   8DBD 34 15                PSHS  X,B,CC                         SAVE REGISTERS AND INTERRUPT STATUS
1715   8DBF 1A 5Ø                ORCC  #$5Ø                           DISABLE INTERRUPTS
1716   8DC1 96 E7                LDA   TIMOUT                         GET TIMEOUT VARIABLE DELAY
1717   8DC3 9E 8A                LDX   ZERO                           X=Ø: TIMEOUT CONSTANT DELAY
1718   8DC5 8D 1F          L8DC5 BSR   L8DE6                          GO GET RS 232 STATUS
1719   8DC7 24 FC                BCC   L8DC5                          LOOP IF SPACING
1720   8DC9 8D 1B          L8DC9 BSR   L8DE6                          GET RS 232 STATUS
1721   8DCB 25 FC                BLO   L8DC9                          LOOP IF MARKING
1722   8DCD 8D 2A                BSR   L8DF9                          DELAY 1/2 BIT TIME
1723   8DCF C6 Ø1                LDB   #$Ø1                           * GET BIT SHIFT COUNTER AND BIT
1724   8DD1 34 Ø4                PSHS  B                              * MASK AND SAVE IT ON STACK
1725   8DD3 4F                   CLRA                                 RESET DATA BYTE
1726   8DD4 8D 21          L8DD4 BSR   L8DF7                          GO DELAY ONE BIT TIME
1727   8DD6 F6 FF 22             LDB   PIA1+2                         * RS 232 INPUT TO
1728   8DD9 56                   RORB                                 * CARRY FLAG
```

```
1729   8DDA 24 02                BCC   L8DDE                       BRANCH IF RS 232 INPUT = 0 (SPACING)
1730   8DDC AA E4                ORA   ,S                          IF MARKING,  OR  A 1 BIT INTO DATA BYTE
1731   8DDE 68 E4          L8DDE ASL   ,S                          SHIFT BIT COUNTER ONE BIT TO LEFT
1732   8DE0 24 F2                BCC   L8DD4                       CARRY WILL BE SET AFTER 8 SHIFTS
1733   8DE2 32 61                LEAS  $01,S                       PULL BIT COUNTER OFF THE STACK
1734   8DE4 35 95                PULS  CC,B,X,PC                   RESTORE INTERRUPT STATUS & RETURN
1735
1736                       * PUT RS 232 STATUS INTO THE CARRY FLAG AND CHECK FOR TIMEOUT
1737   8DE6 F6 FF 22       L8DE6 LDB   PIA1+2                      * RS 232 INPUT TO
1738   8DE9 56                   RORB                              * CARRY FLAG
1739   8DEA 30 01                LEAX  $01,X                       = INCREMENT CONSTANT TIMEOUT
1740   8DEC 26 08                BNE   L8DF6                       = DELAY, RETURN IF <> 0
1741   8DEE 4A                   DECA                              * DECREMENT VARIABLE TIMEOUT
1742   8DEF 26 05                BNE   L8DF6                       * DELAY: RETURN IF <> 0
1743                      * DLOAD HAS TIMED OUT HERE
1744   8DF1 32 62                LEAS  $02,S                       PURGE RETURN ADDRESS OFF STACK
1745   8DF3 35 15                PULS  CC,B,X                      CLEAN UP STACK/RESTORE INTERRUPTS
1746   8DF5 4C                   INCA                              SET ACCA = 1; ZERO FLAG = 0
1747   8DF6 39             L8DF6 RTS
1748                      * DELAY LOOP -- COUNT DOWN DLBAUD
1749   8DF7 8D 00          L8DF7 BSR   L8DF9                       CALL DELAY ROUTINE
1750   8DF9 34 02          L8DF9 PSHS  A                           SAVE ACCA
1751   8DFB 96 E6                LDA   DLBAUD                      GET DLOAD DELAY - 1/2 BIT TIME DELAY
1752   8DFD 21 FE          L8DFD BRN   L8DFD                       DUMMY INST - JUST ADD TO DELAY
1753   8DFF 4A                   DECA                              DEC DELAY TIMER
1754   8E00 26 FB                BNE   L8DFD                       NOT DONE
1755   8E02 35 82                PULS  A,PC                        RESTORE ACCA AND RETURN
1756                      *
1757   8E04 34 02          L8E04 PSHS  A                           SAVE CHARACTER ON STACK
1758   8E06 A8 21                EORA  $01,Y                       * EOR CHARACTER WITH 1,Y AND
1759   8E08 A7 21                STA   $01,Y                       * SAVE RESULT IN 1,Y
1760   8E0A 35 02                PULS  A                           GET CHARACTER BACK
1761
1762                       * SEND CHAR IN ACCA OUT OVER RS232 OUTPUT
1763   8E0C 34 07          L8E0C PSHS  B,A,CC                      SAVE ACCD AND INTERRUPT STATUS
1764   8E0E 1A 50                ORCC  #$50                        DISABLE INTERRUPTS
1765   8E10 8D E5                BSR   L8DF7                       DELAY AWHILE
1766   8E12 8D E3                BSR   L8DF7                       DELAY SOME MORE
1767   8E14 7F FF 20             CLR   PIA1                        SET R5232 OUTPUT TO SPACING
1768   8E17 8D DE                BSR   L8DF7                       DELAY SOME MORE - START BIT
1769   8E19 C6 01                LDB   #$01                        BIT CTR - SEND 8 BITS
1770   8E1B 34 04                PSHS  B                           SAVE BIT CTR ON STACK
1771   8E1D A6 62          L8E1D LDA   $02,S                       GET OUTPUT BYTE
1772   8E1F A4 E4                ANDA  ,S                          AND IT W/THE BIT CTR
1773   8E21 27 02                BEQ   L8E25                       THIS BIT IN OUTPUT BYTE = 0
1774   8E23 86 02                LDA   #$02                        OUTPUT BIT = 1; SET R5232 TO MARKING
1775   8E25 B7 FF 20       L8E25 STA   PIA1                        BET R5232 TO VALUE IN ACCA
1776   8E28 8D CD                BSR   L8DF7                       DELAY FOR AWHILE
1777   8E2A 68 E4                ASL   ,S                          SHIFT BIT CTR
1778   8E2C 24 EF                BCC   L8E1D                       WHEN CARRY SET, 8 BITS DONE
1779   8E2E 86 02                LDA   #$02                        WHEN DONE, SET R5232 TO MARKING
1780   8E30 B7 FF 20             STA   PIA1                        SET R5232 OUTPUT
1781   8E33 32 61                LEAS  $01,S                       PULL BIT CTR OFF THE STACK
1782   8E35 35 87                PULS  CC,A,B,PC                   RESTORE ACCD, INTERRUPTS & RETURN
1783
1784                       * PROCESS EXCLAMATION POINT
1785   8E37 86 01          L8E37 LDA   #$01                        * SET SPACES
1786   8E39 97 D9                STA   VD9                         * COUNTER = 1
1787                       * PROCESS STRING ITEM - LIST
1788   8E3B 5A             L8E3B DECB                              DECREMENT FORMAT STRING LENGTH COUNTER
1789   8E3C BD 8F D8             JSR   L8FD8                       SEND A '+' TO CONSOLE OUT IF VDA <>0
1790   8E3F 9D A5                JSR   GETCCH                      GET CURRENT INPUT CHARACTER
1791   8E41 10 27 00 93          LBEQ  L8ED8                       EXIT PRINT USING IF END OF LINE
1792   8E45 D7 D3                STB   VD3                         SAVE REMAINDER FORMAT STRING LENGTH
1793   8E47 BD B1 56             JSR   LB156                       EVALUATE EXPRESSION
1794   8E4A BD B1 46             JSR   LB146                        TM  ERROR IF NUMERIC VARIABLE
1795   8E4D 9E 52                LDX   FPA0+2                      * GET ITEM - LIST DESCRIPTOR ADDRESS
1796   8E4F 9F 4D                STX   V4D                         * AND SAVE IT IN V4D
1797   8E51 D6 D9                LDB   VD9                         GET SPACES COUNTER
1798   8E53 BD B6 AD             JSR   LB6AD                       PUT ACCB BYTES INTO STRING SPACE & PUT DESCRIPTOR ON STRING STACK
1799   8E56 BD B9 9F             JSR   LB99F                       PRINT THE FORMATTED STRING TO CONSOLE OUT
1800                       * PAD FORMAT STRING WITH SPACES IF ITEM - LIST STRING < FORMAT STRING LENGTH
1801   8E59 9E 52                LDX   FPA0+2                      POINT X TO FORMATTED STRING DESCRIPTOR ADDRESS
1802   8E5B D6 D9                LDB   VD9                         GET SPACES COUNTER
1803   8E5D E0 84                SUBB  ,X                          SUBTRACT LENGTH OF FORMATTED STRING
1804   8E5F 5A             L8E5F DECB                              DECREMENT DIFFERENCE
1805   8E60 10 2B 01 4F          LBMI  L8FB3                       GO INTERPRET ANOTHER ITEM - LIST
1806   8E64 BD B9 AC             JSR   LB9AC                       PAD FORMAT STRING WITH A SPACE
1807   8E67 20 F6                BRA   L8E5F                       KEEP PADDING
1808                       * PERCENT SIGN - PROCESS A %SPACES% COMMAND
1809   8E69 D7 D3          L8E69 STB   VD3                         * SAVE THE CURRENT FORMAT STRING
1810   8E6B 9F 0F                STX   TEMPTR                      * COUNTER AND POINTER
1811   8E6D 86 02                LDA   #$02                        INITIAL SPACES COUNTER = 2
1812   8E6F 97 D9                STA   VD9                         SAVE IN SPACES COUNTER
1813   8E71 A6 84          L8E71 LDA   ,X                          GET A CHARACTER FROM FORMAT STRING
1814   8E73 81 25                CMPA  #'%'                        COMPARE TO TERMINATOR CHARACTER
1815   8E75 27 C4                BEQ   L8E3B                       BRANCH IF END OF SPACES COMMAND
1816   8E77 81 20                CMPA  #' '                        BLANK
1817   8E79 26 07                BNE   L8E82                       BRANCH IF ILLEGAL CHARACTER
1818   8E7B 0C D9                INC   VD9                         ADD ONE TO SPACES COUNTER
1819   8E7D 30 01                LEAX  $01,X                       MOVE FORMAT POINTER UP ONE
1820   8E7F 5A                   DECB                              DECREMENT LENGTH COUNTER
1821   8E80 26 EF                BNE   L8E71                       BRANCH IF NOT END OF FORMAT STRING
1822   8E82 9E 0F          L8E82 LDX   TEMPTR                      * RESTORE CURRENT FORMAT STRING COUNTER
1823   8E84 D6 D3                LDB   VD3                         * AND POINTER TO POSITION BEFORE SPACES COMMAND
1824   8E86 86 25                LDA   #'%'                        SEND A  %  TO CONSOLE OUT AS A DEBUGGING AID
```

```
1825                        * ERROR PROCESSOR - ILLEGAL CHARACTER OR BAD SYNTAX IN FORMAT STRING
1826   8E88 BD 8F D8        L8E88   JSR  L8FD8                        SEND A  +' TO CONSOLE OUT IF VDA <> Ø
1827   8E8B BD A2 82                JSR  LA282                       SEND CHARACTER TO CONSOLE OUT
1828   8E8E 2Ø 29                   BRA  L8EB9                       GET NEXT CHARACTER IN FORMAT STRING
1829
1830                        * PRINT RAM HOOK
1831   8E9Ø 81 CD           XVEC9   CMPA #$CD                        USING TOKEN
1832   8E92 27 Ø1                   BEQ  L8E95                       BRANCH IF PRINT USING
1833   8E94 39                      RTS
1834
1835                        * PRINT USING
1836                        * VDA IS USED AS A STATUS BYTE: BIT 6 = COMMA FORCE
1837                        * BIT 5=LEADING ASTERISK FORCE; BIT 4 = FLOATING $ FORCE
1838                        * BIT 3 = PRE SIGN FORCE; BIT 2 = POST SIGN FORCE; BIT Ø = EXPONENTIAL FORCE
1839   8E95 32 62           L8E95   LEAS $Ø2,S                       PURGE RETURN ADDRESS OFF THE STACK
1840   8E97 BD B1 58                JSR  LB158                       EVALUATE FORMAT STRING
1841   8E9A BD B1 46                JSR  LB146                        TM  ERROR IF VARIABLE TYPE = NUMERIC
1842   8E9D C6 3B                   LDB  #';'                        CHECK FOR ITEM LIST SEPARATOR
1843   8E9F BD B2 6F                JSR  LB26F                       SYNTAX CHECK FOR ;
1844   8EA2 9E 52                   LDX  FPA0+2                      * GET FORMAT STRING DESCRIPTOR ADDRESS
1845   8EA4 9F D5                   STX  VD5                         * AND SAVE IT IN VD5
1846   8EA6 2Ø Ø6                   BRA  L8EAE                       GO PROCESS FORMAT STRING
1847   8EA8 96 D7           L8EA8   LDA  VD7                         *CHECK NEXT PRINT ITEM FLAG AND
1848   8EAA 27 Ø8                   BEQ  L8EB4                       * FC  ERROR IF NO FURTHER PRINT ITEMS
1849   8EAC 9E D5                   LDX  VD5                         RESET FORMAT STRING POINTER TO START OF STRING
1850   8EAE ØF D7           L8EAE   CLR  VD7                         RESET NEXT PRINT ITEM FLAG
1851   8EBØ E6 84                   LDB  ,X                          GET LENGTH OF FORMAT STRING
1852   8EB2 26 Ø3                   BNE  L8EB7                       INTERPRET FORMAT STRING IF LENGTH > Ø
1853   8EB4 7E B4 4A        L8EB4   JMP  LB44A                        FC  ERROR IF FORMAT STRING = NULL
1854   8EB7 AE Ø2           L8EB7   LDX  $Ø2,X                       POINT X TO START OF FORMAT STRING
1855                        * INTERPRET THE FORMAT STRING
1856   8EB9 ØF DA           L8EB9   CLR  VDA                         CLEAR THE STATUS BYTE
1857   8EBB ØF D9           L8EBB   CLR  VD9                         CLEAR LEFT DIGIT COUNTER
1858   8EBD A6 8Ø                   LDA  ,X+                         GET A CHARACTER FROM FORMAT STRING
1859   8EBF 81 21                   CMPA #'!'                        EXCLAMATION POINT?
1860   8EC1 1Ø 27 FF 72             LBEQ L8E37                       YES - STRING TYPE FORMAT
1861   8EC5 81 23                   CMPA #'#'                        NUMBER SIGN? (DIGIT LOCATOR)
1862   8EC7 27 5B                   BEQ  L8F24                       YES - NUMERIC TYPE FORMAT
1863   8EC9 5A                      DECB                             DECREMENT FORMAT STRING LENGTH
1864   8ECA 26 16                   BNE  L8EE2                       BRANCH IF NOT DONE
1865   8ECC BD 8F D8                JSR  L8FD8                       SEND A  +  TO CONSOLE OUT IF VDA <> Ø
1866   8ECF BD A2 82                JSR  LA282                       SEND CHARACTER TO CONSOLE OUT
1867   8ED2 9D A5           L8ED2   JSR  GETCCH                      GET CURRENT CHARACTER FROM BASIC
1868   8ED4 26 D2                   BNE  L8EA8                       BRANCH IF NOT END OF LINE
1869   8ED6 96 D7                   LDA  VD7                         GET NEXT PRINT ITEM FLAG
1870   8ED8 26 Ø3           L8ED8   BNE  L8EDD                       BRANCH IF MORE PRINT ITEMS
1871   8EDA BD B9 58                JSR  LB958                       SEND A CARRIAGE RETURN TO CONSOLE OUT
1872   8EDD 9E D5           L8EDD   LDX  VD5                         POINT X TO FORMAT STRING DESCRIPTOR
1873   8EDF 7E B6 59                JMP  LB659                       RETURN ADDRESS AND LENGTH OF FORMAT STRING - EXIT PRINT USING
1874   8EE2 81 2B           L8EE2   CMPA #'+'                        CHECK FOR  +  (PRE-SIGN FORCE)
1875   8EE4 26 Ø9                   BNE  L8EEF                       NO PLUS
1876   8EE6 BD 8F D8                JSR  L8FD8                       SEND A  +' TO CONSOLE OUT IF VDA <> Ø
1877   8EE9 86 Ø8                   LDA  #$Ø8                        * LOAD THE STATUS BYTE WITH 8;
1878   8EEB 97 DA                   STA  VDA                         * PRE-SIGN FORCE FLAG
1879   8EED 2Ø CC                   BRA  L8EBB                       INTERPRET THE REST OF THE FORMAT STRING
1880   8EEF 81 2E           L8EEF   CMPA #'.'                        DECIMAL POINT?
1881   8EF1 27 4E                   BEQ  L8F41                       YES
1882   8EF3 81 25                   CMPA #'%'                        PERCENT SIGN?
1883   8EF5 1Ø 27 FF 7Ø             LBEQ L8E69                       YES
1884   8EF9 A1 84                   CMPA ,X                          COMPARE THE PRESENT FORMAT STRING INPUT
1885                        *                                       CHARACTER TO THE NEXT ONE IN THE STRING
1886   8EFB 26 8B           L8EFB   BNE  L8E88                       NO MATCH - ILLEGAL CHARACTER
1887                        * TWO CONSECUTIVE EQUAL CHARACTERS IN FORMAT STRING
1888   8EFD 81 24                   CMPA #'$'                        DOLLAR SIGN?
1889   8EFF 27 19                   BEQ  L8F1A                       YES - MAKE THE DOLLAR SIGN FLOAT
1890   8F01 81 2A                   CMPA #'*'                        ASTERISK?
1891   8F03 26 F6                   BNE  L8EFB                       NO - ILLEGAL CHARACTER
1892   8F05 96 DA                   LDA  VDA                         * GRAB THE STATUS BYTE AND BET BIT 5
1893   8F07 8A 2Ø                   ORA  #$2Ø                        * TO INDICATE THAT THE OUTPUT WILL
1894   8F09 97 DA                   STA  VDA                         * BE LEFT PADDED WITH ASTERISKS
1895   8F0B C1 Ø2                   CMPB #2                          = CHECK TO SEE IF THE $$ ARE THE LAST TWO
1896   8F0D 25 11                   BLO  L8F2Ø                       = CHARACTERS IN THE FORMAT STRING AND BRANCH IF SO
1897   8F0F A6 Ø1                   LDA  $Ø1,X                       GET THE NEXT CHARACTER AFTER **
1898   8F11 81 24                   CMPA #'$'                        CHECK FOR **$
1899   8F13 26 ØB                   BNE  L8F2Ø                       CHECK FOR MORE CHARACTERS
1900   8F15 5A                      DECB                             DECREMENT STRING LENGTH COUNTER
1901   8F16 3Ø Ø1                   LEAX $Ø1,X                       MOVE FORMAT STRING POINTER UP ONE
1902   8F18 ØC D9                   INC  VD9                         ADD ONE TO LEFT DIGIT COUNTER - FOR ASTERISK PAD AND
1903                        *                                       FLOATING DOLLAR SIGN COMBINATION
1904   8F1A 96 DA           L8F1A   LDA  VDA                         * GET THE STATUS BYTE AND SET
1905   8F1C 8A 1Ø                   ORA  #$1Ø                        * BIT 4 TO INDICATE A
1906   8F1E 97 DA                   STA  VDA                         * FLOATING DOLLAR SIGN
1907   8F2Ø 3Ø Ø1           L8F2Ø   LEAX $Ø1,X                       MOVE FORMAT STRING POINTER UP ONE
1908   8F22 ØC D9                   INC  VD9                         ADD ONE TO LEFT DIGIT (FLOATING $ OR ASTERISK PAD)
1909                        * PROCESS CHARACTERS TO THE LEFT OF THE DECIMAL POINT IN THE FORMAT STRING
1910   8F24 ØF D8           L8F24   CLR  VD8                         CLEAR THE RIGHT DIGIT COUNTER
1911   8F26 ØC D9           L8F26   INC  VD9                         ADD ONE TO LEFT DIGIT COUNTER
1912   8F28 5A                      DECB                             DECREMENT FORMAT STRING LENGTH COUNTER
1913   8F29 27 49                   BEQ  L8F74                       BRANCH IF END OF FORMAT STRING
1914   8F2B A6 8Ø                   LDA  ,X+                         GET THE NEXT FORMAT CHARACTER
1915   8F2D 81 2E                   CMPA #'.'                        DECIMAL POINT?
1916   8F2F 27 1E                   BEQ  L8F4F                       YES
1917   8F31 81 23                   CMPA #'#'                        NUMBER SIGN?
1918   8F33 27 F1                   BEQ  L8F26                       YES
1919   8F35 81 2C                   CMPA #','                        COMMA?
1920   8F37 26 21                   BNE  L8F5A                       NO
```

```
1921   8F39 96 DA                      LDA   VDA                         * GET THE STATUS BYTE
1922   8F3B 8A 40                      ORA   #$40                        * AND SET BIT 6 WHICH IS THE
1923   8F3D 97 DA                      STA   VDA                         * COMMA SEPARATOR FLAG
1924   8F3F 20 E5                      BRA   L8F26                       PROCESS MORE CHARACTERS TO LEFT OF DECIMAL POINT
1925                          * PROCESS DECIMAL POINT IF NO DIGITS TO LEFT OF IT
1926   8F41 A6 84             L8F41    LDA   ,X                          GET NEXT FORMAT CHARACTER
1927   8F43 81 23                      CMPA  #'#'                        IS IT A NUMBER SIGN?
1928   8F45 10 26 FF 3F                LBNE  L8E88                       NO
1929   8F49 86 01                      LDA   #1                          * SET THE RIGHT DIGIT COUNTER TO 1 -
1930   8F4B 97 D8                      STA   VD8                         * ALLOW ONE SPOT FOR DECIMAL POINT
1931   8F4D 30 01                      LEAX  $01,X                       MOVE FORMAT POINTER UP ONE
1932                          * PROCESS DIGITS TO RIGHT OF DECIMAL POINT
1933   8F4F 0C D8             L8F4F    INC   VD8                         ADD ONE TO RIGHT DIGIT COUNTER
1934   8F51 5A                         DECB                              DECREMENT FORMAT LENGTH COUNTER
1935   8F52 27 20                      BEQ   L8F74                       BRANCH IF END OF FORMAT STRING
1936   8F54 A6 80                      LDA   ,X+                         GET A CHARACTER FROM FORMAT STRING
1937   8F56 81 23                      CMPA  #'#'                        IS IT NUMBER SIGN?
1938   8F58 27 F5                      BEQ   L8F4F                       YES - KEEP CHECKING
1939                          * CHECK FOR EXPONENTIAL FORCE
1940   8F5A 81 5E             L8F5A    CMPA  #$5E                        CHECK FOR UP ARROW
1941   8F5C 26 16                      BNE   L8F74                       NO UP ARROW
1942   8F5E A1 84                      CMPA  ,X                          IS THE NEXT CHARACTER AN UP ARROW?
1943   8F60 26 12                      BNE   L8F74                       NO
1944   8F62 A1 01                      CMPA  $01,X                       AND THE NEXT CHARACTER?
1945   8F64 26 0E                      BNE   L8F74                       NO
1946   8F66 A1 02                      CMPA  $02,X                       HOW ABOUT THE 4TH CHARACTER?
1947   8F68 26 0A                      BNE   L8F74                       NO, ALSO
1948   8F6A C1 04                      CMPB  #4                          * CHECK TO SEE IF THE 4 UP ARROWS ARE IN THE
1949   8F6C 25 06                      BLO   L8F74                       * FORMAT STRING AND BRANCH IF NOT
1950   8F6E C0 04                      SUBB  #4                          = MOVE POINTER UP 4 AND SUBTRACT
1951   8F70 30 04                      LEAX  $04,X                       = FOUR FROM LENGTH
1952   8F72 0C DA                      INC   VDA                         INCREMENT STATUS BYTE - EXPONENTIAL FORM
1953
1954                          * CHECK FOR A PRE OR POST - SIGN FORCE AT END OF FORMAT STRING
1955   8F74 30 1F             L8F74    LEAX  $-01,X                      MOVE POINTER BACK ONE
1956   8F76 0C D9                      INC   VD9                         ADD ONE TO LEFT DIGIT COUNTER FOR PRE-SIGN FORCE
1957   8F78 96 DA                      LDA   VDA                         * PRE-SIGN
1958   8F7A 85 08                      BITA  #$08                        * FORCE AND
1959   8F7C 26 18                      BNE   L8F96                       * BRANCH IF SET
1960   8F7E 0A D9                      DEC   VD9                         DECREMENT LEFT DIGIT   NO PRE-SIGN FORCE
1961   8F80 5D                         TSTB                              * CHECK LENGTH COUNTER AND BRANCH
1962   8F81 27 13                      BEQ   L8F96                       * IF END OF FORMAT STRING
1963   8F83 A6 84                      LDA   ,X                          GET NEXT FORMAT STRING CHARACTER
1964   8F85 80 2D                      SUBA  #'-'                        CHECK FOR MINUS SIGN
1965   8F87 27 06                      BEQ   L8F8F                       BRANCH IF MINUS SIGN
1966   8F89 81 FE                      CMPA  #('+')-('-')                CHECK FOR PLUS SIGN
1967   8F8B 26 09                      BNE   L8F96                       BRANCH IF NO PLUS SIGN
1968   8F8D 86 08                      LDA   #$08                        GET THE PRE-SIGN FORCE FLAG
1969   8F8F 8A 04             L8F8F    ORA   #$04                         OR  IN POST-SIGN FORCE FLAG
1970   8F91 9A DA                      ORA   VDA                         OR  IN THE STATUS BYTE
1971   8F93 97 DA                      STA   VDA                         SAVE THE STATUS BYTE
1972   8F95 5A                         DECB                              DECREMENT FORMAT STRING LENGTH
1973
1974                          * EVALUATE NUMERIC ITEM-LIST
1975   8F96 9D A5             L8F96    JSR   GETCCH                      GET CURRENT CHARACTER
1976   8F98 10 27 FF 3C                LBEQ  L8ED8                       BRANCH IF END OF LINE
1977   8F9C D7 D3                      STB   VD3                         SAVE FORMAT STRING LENGTH WHEN FORMAT EVALUATION ENDED
1978   8F9E BD B1 41                   JSR   LB141                       EVALUATE EXPRESSION
1979   8FA1 96 D9                      LDA   VD9                         GET THE LEFT DIGIT COUNTER
1980   8FA3 9B D8                      ADDA  VD8                         ADD IT TO THE RIGHT DIGIT COUNTER
1981   8FA5 81 11                      CMPA  #17                         *
1982   8FA7 10 22 24 9F                LBHI  LB44A                       * FC  ERROR IF MORE THAN 16 DIGITS AND DECIMAL POiNT
1983 > 8FAB BD 8F E5                   JSR   L8FE5                       CONVERT ITEM-LIST TO FORMATTED ASCII STRING
1984   8FAE 30 1F                      LEAX  $-01,X                      MOVE BUFFER POINTER BACK ONE
1985   8FB0 BD B9 9C                   JSR   STRINOUT                    DISPLAY THE FORMATTED STRING TO CONSOLE OUT
1986   8FB3 0F D7             L8FB3    CLR   VD7                         RESET NEXT PRINT ITEM FLAG
1987   8FB5 9D A5                      JSR   GETCCH                      GET CURRENT INPUT CHARACTER
1988   8FB7 27 0D                      BEQ   L8FC6                       BRANCH IF END OF LINE
1989   8FB9 97 D7                      STA   VD7                         SAVE CURRENT CHARACTER (<>0) IN NEXT PRINT ITEM FLAG
1990   8FBB 81 3B                      CMPA  #';'                        * CHECK FOR ; - ITEM-LIST SEPARATOR AND
1991   8FBD 27 05                      BEQ   L8FC4                       * BRANCH IF SEMICOLON
1992   8FBF BD B2 6D                   JSR   SYNCOMMA                    SYNTAX CHECK FOR COMMA
1993   8FC2 20 02                      BRA   L8FC6                       PROCESS NEXT PRINT ITEM
1994   8FC4 9D 9F             L8FC4    JSR   GETNCH                      GET NEXT INPUT CHARACTER
1995   8FC6 9E D5             L8FC6    LDX   VD5                         GET FORMAT STRING DESCRIPTOR ADDRESS
1996   8FC8 E6 84                      LDB   ,X                          GET LENGTH OF FORMAT STRING
1997   8FCA D0 D3                      SUBB  VD3                         SUBTRACT AMOUNT OF FORMAT STRING LEFT AFTER LAST PRINT ITEM
1998   8FCC AE 02                      LDX   $02,X                       *GET FORMAT STRING START ADDRESS AND ADVANCE
1999   8FCE 3A                         ABX                               *POINTER TO START OF UNUSED FORMAT STRING
2000   8FCF D6 D3                      LDB   VD3                         =GET AMOUNT OF UNUSED FORMAT STRING
2001   8FD1 10 26 FE E4                LBNE  L8EB9                       =REINTERPRET FORMAT STRING FROM THAT POINT
2002   8FD5 7E 8E D2                   JMP   L8ED2                       REINTERPRET FORMAT STRING FROM THE START IF ENTIRELY
2003                          *                                         USED ON LAST PRINT ITEM
2004
2005                          * PRINT A  +  TO CONSOLE OUT IF THE STATUS BYTE <> 0
2006   8FD8 34 02             L8FD8    PSHS  A                           RESTORE ACCA AND RETURN
2007   8FDA 86 2B                      LDA   #'+'                        GET ASCII PLUS SIGN
2008   8FDC 0D DA                      TST   VDA                         * CHECK THE STATUS BYTE AND
2009   8FDE 27 03                      BEQ   L8FE3                       * RETURN IF = 0
2010   8FE0 BD A2 82                   JSR   LA282                       SEND A CHARACTER TO CONSOLE OUT
2011   8FE3 35 82             L8FE3    PULS  A,PC                        RETURN ACCA AND RETURN
2012
2013                          * CONVERT ITEM-LIST TO DECIMAL ASCII STRING
2014   8FE5 CE 03 DB          L8FE5    LDU   #STRBUF+4                   POINT U TO STRING BUFFER
2015   8FE8 C6 20                      LDB   #SPACE                      BLANK
2016   8FEA 96 DA                      LDA   VDA                         * GET THE STATUS FLAG AND
```

```
2017   8FEC 85 08                BITA #$08                 * CHECK FOR A PRE-SIGN FORCE
2018   8FEE 27 02                BEQ  L8FF2                * BRANCH IF NO PRE-SIGN FORCE
2019   8FF0 C6 2B                LDB  #'+'                 PLUS SIGN
2020   8FF2 0D 54          L8FF2 TST  FP0SGN               CHECK THE SIGN OF FPA0
2021   8FF4 2A 04                BPL  L8FFA                BRANCH IF POSITIVE
2022   8FF6 0F 54                CLR  FP0SGN               FORCE FPA0 SIGN TO BE POSITIVE
2023   8FF8 C6 2D                LDB  #'-'                 MINUS SIGN
2024   8FFA E7 C0          L8FFA STB  ,U+                  SAVE THE SIGN IN BUFFER
2025   8FFC C6 30                LDB  #'0'                 * PUT A ZERO INTO THE BUFFER
2026   8FFE E7 C0                STB  ,U+                  *
2027   9000 84 01                ANDA #$01                 = CHECK THE EXPONENTIAL FORCE FLAG IN
2028   9002 10 26 01 07          LBNE L910D                = THE STATUS BYTE - BRANCH IF ACTIVE
2029   9006 8E BD C0             LDX  #LBDC0               POINT X TO FLOATING POINT 1E + 09
2030   9009 BD BC A0             JSR  LBCA0                COMPARE FPA0 TO (X)
2031   900C 2B 15                BMI  L9023                BRANCH IF FPA0 < 1E+09
2032   900E BD BD D9             JSR  LBDD9                CONVERT FP NUMBER TO ASCII STRING
2033   9011 A6 80          L9011 LDA  ,X+                  * ADVANCE POINTER TO END OF
2034   9013 26 FC                BNE  L9011                * ASCII STRING (ZERO BYTE)
2035   9015 A6 82          L9015 LDA  ,-X                  = MOVE THE
2036   9017 A7 01                STA  $01,X                = ENTIRE STRING
2037   9019 8C 03 DA             CMPX #STRBUF+3            = UP ONE
2038   901C 26 F7                BNE  L9015                = BYTE
2039   901E 86 25                LDA  #'%'                 * INSERT A % SIGN AT START OF
2040   9020 A7 84                STA  ,X                   * STRING - OVERFLOW ERROR
2041   9022 39                   RTS
2042
2043   9023 96 4F          L9023 LDA  FP0EXP               = GET EXPONENT OF FPA0
2044   9025 97 47                STA  V47                  = AND SAVE IT IN V47
2045   9027 27 03                BEQ  L902C                BRANCH IF FPA0 = 0
2046   9029 BD 91 CD             JSR  L91CD                CONVERT FPA0 TO NUMBER WITH 9 SIGNIFICANT
2047                       *                               PLACES TO LEFT OF DECIMAL POINT
2048   902C 96 47          L902C LDA  V47                  GET BASE 10 EXPONENT OFFSET
2049   902E 10 2B 00 81          LBMI L90B3                BRANCH IF FPA0 < 100,000,000
2050   9032 40                   NEGA                      * CALCULATE THE NUMBER OF LEADING ZEROES TO INSERT -
2051   9033 9B D9                ADDA VD9                  * SUBTRACT BASE 10 EXPONENT OFFSET AND 9 (FPA0 HAS
2052   9035 80 09                SUBA #$09                 * 9 PLACES TO LEFT OF EXPONENT) FROM LEFT DIGIT COUNTER
2053   9037 BD 90 EA             JSR  L90EA                PUT ACCA ZEROES IN STRING BUFFER
2054   903A BD 92 63             JSR  L9263                INITIALIZE DECIMAL POINT AND COMMA COUNTERS
2055   903D BD 92 02             JSR  L9202                CONVERT FPA0 TO DECIMAL ASCII IN THE STRING BUFFER
2056   9040 96 47                LDA  V47                  * GET BASE 10 EXPONENT AND PUT THAT MANY
2057   9042 BD 92 81             JSR  L9281                * ZEROES IN STRING BUFFER - STOP AT DECIMAL POINT
2058   9045 96 47                LDA  V47                  WASTED INSTRUCTION - SERVES NO PURPOSE
2059   9047 BD 92 49             JSR  L9249                CHECK FOR DECIMAL POINT
2060   904A 96 D8                LDA  VD8                  GET THE RIGHT DIGIT COUNTER
2061   904C 26 02                BNE  L9050                BRANCH IF RIGHT DIGIT COUNTER <> 0
2062   904E 33 5F                LEAU $-01,U               * MOVE BUFFER POINTER BACK ONE - DELETE
2063                       *                               * DECIMAL POINT IF NO RIGHT DIGITS SPECIFIED
2064   9050 4A             L9050 DECA                      SUBTRACT ONE (DECIMAL POINT)
2065   9051 BD 90 EA             JSR  L90EA                PUT ACCA ZEROES INTO BUFFER (TRAILING ZEROES)
2066   9054 BD 91 85       L9054 JSR  L9185                INSERT ASTERISK PADDING, FLOATING $, AND POST-SIGN
2067   9057 4D                   TSTA                      WAS THERE A POST-SIGN?
2068   9058 27 06                BEQ  L9060                NO
2069   905A C1 2A                CMPB #'*'                 IS THE FIRST CHARACTER AN $?
2070   905C 27 02                BEQ  L9060                YES
2071   905E E7 C0                STB  ,U+                  STORE THE POST-SIGN
2072   9060 6F C4          L9060 CLR  ,U                   CLEAR THE LAST CHARACTER IN THE BUFFER
2073                       *
2074                       * REMOVE ANY EXTRA BLANKS OR ASTERISKS FROM THE
2075                       * STRING BUFFER TO THE LEFT OF THE DECIMAL POINT
2076   9062 8E 03 DA             LDX  #STRBUF+3            POINT X TO THE START OF THE BUFFER
2077   9065 30 01          L9065 LEAX $01,X                MOVE BUFFER POINTER UP ONE
2078   9067 9F 0F                STX  TEMPTR               SAVE BUFFER POINTER IN TEMPTR
2079   9069 96 3A                LDA  VARPTR+1             * GET ADDRESS OF DECIMAL POINT IN BUFFER, SUBTRACT
2080   906B 90 10                SUBA TEMPTR+1             * CURRENT POSITION AND SUBTRACT LEFT DIGIT COUNTER -
2081   906D 90 D9                SUBA VD9                  * THE RESULT WILL BE ZERO WHEN TEMPTR+1 IS POINTING
2082                             *                         * TO THE FIRST DIGIT OF THE FORMAT STRING
2083   906F 27 38                BEQ  L90A9                RETURN IF NO DIGITS TO LEFT OF THE DECIMAL POINT
2084   9071 A6 84                LDA  ,X                   GET THE CURRENT BUFFER CHARACTER
2085   9073 81 20                CMPA #SPACE               SPACE?
2086   9075 27 EE                BEQ  L9065                YES - ADVANCE POINTER
2087   9077 81 2A                CMPA #'*'                 ASTERISK?
2088   9079 27 EA                BEQ  L9065                YES - ADVANCE POINTER
2089   907B 4F                   CLRA                      A ZERO ON THE STACK IS END OF DATA POINTER
2090   907C 34 02          L907C PSHS A                    PUSH A CHARACTER ONTO THE STACK
2091   907E A6 80                LDA  ,X+                  GET NEXT CHARACTER FROM BUFFER
2092   9080 81 2D                CMPA #'-'                 MINUS SIGN?
2093   9082 27 F8                BEQ  L907C                YES
2094   9084 81 2B                CMPA #'+'                 PLUS SIGN?
2095   9086 27 F4                BEQ  L907C                YES
2096   9088 81 24                CMPA $'$'                 DOLLAR SIGN?
2097   908A 27 F0                BEQ  L907C                YES
2098   908C 81 30                CMPA #'0'                 ZERO?
2099   908E 26 0E                BNE  L909E                NO - ERROR
2100   9090 A6 01                LDA  $01,X                GET CHARACTER FOLLOWING ZERO
2101   9092 8D 16                BSR  L90AA                CLEAR CARRY IF NUMERIC
2102   9094 25 08                BLO  L909E                BRANCH IF NOT A NUMERIC CHARACTER - ERROR
2103   9096 35 02          L9096 PULS A                    * PULL A CHARACTER OFF OF THE STACK
2104   9098 A7 82                STA  ,-X                  * AND PUT IT BACK IN THE STRING BUFFER
2105   909A 26 FA                BNE  L9096                * KEEP GOING UNTIL ZERO FLAG
2106   909C 20 C7                BRA  L9065                KEEP CLEANING UP THE INPUT BUFFER
2107   909E 35 02          L909E PULS A                    = REMOVE THE CHARACTERS ON
2108   90A0 4D                   TSTA                      = THE STACK AND EXIT WHEN
2109   90A1 26 FB                BNE  L909E                = ZERO FLAG FOUND
2110   90A3 9E 0F                LDX  TEMPTR               GET THE STRING BUFFER START POINTER
2111   90A5 86 25                LDA  #'%'                 * PUT A % SIGN BEFORE THE ERROR POSITION TO
2112   90A7 A7 82                STA  ,-X                  * INDICATE AN ERROR
```

```
2113  90A9 39           L90A9   RTS
2114                    *
2115                    * CLEAR CARRY IF NUMERIC
2116  90AA 81 30        L90AA   CMPA #'0'                        ASCII ZERO
2117  90AC 25 04                BLO  L90B2                      RETURN IF ACCA < ASCII 0
2118  90AE 80 3A                SUBA #'9'+1                     *
2119  90B0 80 C6                SUBA #-('9'+1)                  *CARRY CLEAR IF NUMERIC
2120  90B2 39           L90B2   RTS
2121                    *
2122                    * PROCESS AN ITEM-LIST WHICH IS < 100,000,000
2123  90B3 96 D8        L90B3   LDA  VD8                        GET RIGHT DIGIT COUNTER
2124  90B5 27 01                BEQ  L90B8                      BRANCH IF NO FORMATTED DIGITS TO THE RIGHT OF DECIMAL PT
2125  90B7 4A                   DECA                            SUBTRACT ONE FOR DECIMAL POINT
2126  90B8 9B 47        L90B8   ADDA V47                        *ADD THE BASE 10 EXPONENT OFFSET - ACCA CONTAINS THE
2127                    *                                       *NUMBER OF SHIFTS REQUIRED TO ADJUST FPA0 TO THE SPECIFIED
2128                    *                                       *NUMBER OF DIGITS TO THE RIGHT OF THE DECIMAL POINT
2129  90BA 2B 01                BMI  L90BD                      IF ACCA >= 0 THEN NO SHIFTS ARE REQUIRED
2130  90BC 4F                   CLRA                            FORCE SHIFT COUNTER = 0
2131  90BD 34 02        L90BD   PSHS A                          SAVE INITIAL SHIFT COUNTER ON THE STACK
2132  90BF 2A 0A        L90BF   BPL  L90CB                      EXIT ROUTINE IF POSITIVE
2133  90C1 34 02                PSHS A                          SAVE SHIFT COUNTER ON STACK
2134  90C3 BD BB 82             JSR  LBB82                      DIVIDE FPA0 BY 10 - SHIFT ONE DIGIT TO RIGHT
2135  90C6 35 02                PULS A                          GET SHIFT COUNTER FROM THE STACK
2136  90C8 4C                   INCA                            BUMP SHIFT COUNTER UP BY ONE
2137  90C9 20 F4                BRA  L90BF                      CHECK FOR FURTHER DIVISION
2138  90CB 96 47        L90CB   LDA  V47                        * GET BASE 10 EXPONENT OFFSET, ADD INITIAL SHIFT COUNTER
2139  90CD A0 E0                SUBA ,S+                        * AND SAVE NEW BASE 10 EXPONENT OFFSET - BECAUSE
2140  90CF 97 47                STA  V47                        * FPA0 WAS SHIFTED ABOVE
2141  90D1 8B 09                ADDA #$09                       =ADD NINE (SIGNIFICANT PLACES) AND BRANCH IF THERE ARE NO
2142  90D3 2B 19                BMI  L90EE                      =ZEROES TO THE LEFT OF THE DECIMAL POINT IN THIS PRINT ITEM
2143  90D5 96 D9                LDA  VD9                        *DETERMINE HOW MANY FILLER ZEROES TO THE LEFT OF THE DECIMAL
2144  90D7 80 09                SUBA #$09                       *POINT. GET THE NUMBER OF FORMAT PLACES TO LEFT OF DECIMAL
2145  90D9 90 47                SUBA V47                        *POINT, SUBTRACT THE BASE 10 EXPONENT OFFSET AND THE CONSTANT 9
2146  90DB 8D 0D                BSR  L90EA                      *(UNNORMALIZATION)-THEN OUTPUT THAT MANY ZEROES TO THE BUFFER
2147  90DD BD 92 63             JSR  L9263                      INITIALIZE DECIMAL POINT AND COMMA COUNTERS
2148  90E0 20 1D                BRA  L90FF                      PROCESS THE REMAINDER OF THE PRINT ITEM
2149                    *
2150                    * PUT (ACCA+1) ASCII ZEROES IN BUFFER
2151  90E2 34 02        L90E2   PSHS A                          SAVE ZERO COUNTER
2152  90E4 86 30                LDA  #'0'                       * INSERT A ZERO INTO
2153  90E6 A7 C0                STA  ,U+                        * THE BUFFER
2154  90E8 35 02                PULS A                          RESTORE ZERO COUNTER
2155
2156                    * PUT ACCA ASCII ZEROES INTO THE BUFFER
2157  90EA 4A           L90EA   DECA                            DECREMENT ZERO COUNTER
2158  90EB 2A F5                BPL  L90E2                      BRANCH IF NOT DONE
2159  90ED 39                   RTS
2160
2161  90EE 96 D9        L90EE   LDA  VD9                        * GET THE LEFT DIGIT COUNTER AND PUT
2162  90F0 8D F8                BSR  L90EA                      * THAT MANY ZEROES IN THE STRING BUFFER
2163  90F2 BD 92 4D             JSR  L924D                      PUT THE DECIMAL POINT IN THE STRING BUFFER
2164  90F5 86 F7                LDA  #-9                        *DETERMINE HOW MANY FILLER ZEROES BETWEEN THE DECIMAL POINT
2165  90F7 90 47                SUBA V47                        *AND SIGNIFICANT DATA. SUBTRACT BASE 10 EXPONENT FROM -9
2166  90F9 8D EF                BSR  L90EA                      *(UNNORMALIZATION) AND OUTPUT THAT MANY ZEROES TO BUFFER
2167  90FB 0F 45                CLR  V45                        CLEAR THE DECIMAL POINT COUNTER - SUPPRESS THE DECIMAL POINT
2168  90FD 0F D7                CLR  VD7                        CLEAR THE COMMA COUNTER - SUPPRESS COMMAS
2169  90FF BD 92 02     L90FF   JSR  L9202                      DECODE FPA0 INTO A DECIMAL ASCII STRING
2170  9102 96 D8                LDA  VD8                        GET THE RIGHT DIGIT COUNTER
2171  9104 26 02                BNE  L9108                      BRANCH IF RIGHT DIGIT COUNTER <> 0
2172  9106 DE 39                LDU  VARPTR                     RESET BUFFER PTR TO THE DECIMAL POINT IF NO DIGITS TO RIGHT
2173  9108 9B 47        L9108   ADDA V47                        *ADD BASE 10 EXPONENT - A POSITIVE ACCA WILL CAUSE THAT MANY
2174                    *                                       *FILLER ZEROES TO BE OUTPUT TO THE RIGHT OF LAST SIGNIFICANT DATA
2175                    *                                       *SIGNIFICANT DATA
2176  910A 16 FF 43             LBRA L9050                      INSERT LEADING ASTERISKS, FLOATING DOLLAR SIGN, ETC
2177                    *
2178                    * FORCE THE NUMERIC OUTPUT FORMAT TO BE EXPONENTIAL FORMAT
2179  910D 96 4F        L910D   LDA  FP0EXP                      * GET EXPONENT OF FPA0 AND
2180  910F 34 02                PSHS A                          * SAVE IT ON THE STACK
2181  9111 27 03                BEQ  L9116                      BRANCH IF FPA0 = 0
2182  9113 BD 91 CD             JSR  L91CD                      *CONVERT FPA0 INTO A NUMBER WITH 9 SIGNIFICANT
2183                    *                                       *DIGITS TO THE LEFT OF THE DECIMAL POINT
2184  9116 96 D8        L9116   LDA  VD8                        GET THE RIGHT DIGIT COUNTER
2185  9118 27 01                BEQ  L911B                      BRANCH IF NO FORMATTED DIGITS TO THE RIGHT
2186  911A 4A                   DECA                            SUBTRACT ONE FOR THE DECIMAL POINT
2187  911B 9B D9        L911B   ADDA VD9                        ADD TO THE LEFT DIGIT COUNTER
2188  911D 7F 03 DA             CLR  STRBUF+3                   CLEAR BUFFER BYTE AS TEMPORARY STORAGE LOCATION
2189  9120 D6 DA                LDB  VDA                        * GET THE STATUS BYTE FOR A
2190  9122 C4 04                ANDB #$04                       * POST-BYTE FORCE; BRANCH IF
2191  9124 26 03                BNE  L9129                      * A POST-BYTE FORCE
2192  9126 73 03 DA             COM  STRBUF+3                   TOGGLE BUFFER BYTE TO -1 IF NO POST-BYTE FORCE
2193  9129 BB 03 DA     L9129   ADDA STRBUF+3                   SUBTRACT 1 IF NO POST BYTE FORCE
2194  912C 80 09                SUBA #$09                       *SUBTRACT 9 (DUE TO THE CONVERSION TO 9
2195                    *                                       *SIGNIFICANT DIGITS TO LEFT OF DECIMAL POINT)
2196  912E 34 02                PSHS A                          =SAVE SHIFT COUNTER ON THE STACK - ACCA CONTAINS THE NUMBER
2197                    *                                       =OF SHIFTS REQUIRED TO ADJUST FPA0 FOR THE NUMBER OF
2198                    *                                       =FORMATTED PLACES TO THE RIGHT OF THE DECIMAL POINT.
2199  9130 2A 0A        L9130   BPL  L913C                      NO MORE SHIFTS WHEN ACCA >= 0
2200  9132 34 02                PSHS A                          SAVE SHIFT COUNTER
2201  9134 BD BB 82             JSR  LBB82                      DIVIDE FPA0 BY 10 - SHIFT TO RIGHT ONE
2202  9137 35 02                PULS A                          RESTORE THE SHIFT COUNTER
2203  9139 4C                   INCA                            ADD 1 TO SHIFT COUNTER
2204  913A 20 F4                BRA  L9130                      CHECK FOR FURTHER SHIFTING (DIVISION)
2205  913C A6 E4        L913C   LDA  ,S                         *GET THE INITIAL VALUE OF THE SHIFT COUNTER
2206  913E 2B 01                BMI  L9141                      *AND BRANCH IF SHIFTING HAS TAKEN PLACE
2207  9140 4F                   CLRA                            RESET ACCA IF NO SHIFTING HAS TAKEN PLACE
2208  9141 40           L9141   NEGA                            *CALCULATE THE POSITION OF THE DECIMAL POINT BY
```

```
2209  9142 9B D9              ADDA  VD9                       *NEGATING SHIFT COUNTER, ADDING THE LEFT DIGIT COUNTER
2210  9144 4C                 INCA                            *PLUS ONE AND THE POST-BYTE POS1TION, IF USED
2211  9145 BB 03 DA           ADDA  STRBUF+3                  *
2212  9148 97 45              STA   V45                       SAVE DECIMAL POINT COUNTER
2213  914A 0F D7              CLR   VD7                       CLEAR COMMA COUNTER - NO COMMAS INSERTED
2214  914C BD 92 02           JSR   L9202                     CONVERT FPA0 INTO ASCII DECIMAL STRING
2215  914F 35 02              PULS  A                         =GET THE INITIAL VALUE OF SHIFT COUNTER AND
2216  9151 BD 92 81           JSR   L9281                     =INSERT THAT MANY ZEROES INTO THE BUFFER
2217  9154 96 D8              LDA   VD8                       *GET THE RIGHT DIGIT COUNTER AND BRANCH
2218  9156 26 02              BNE   L915A                     *IF NOT ZERO
2219  9158 33 5F              LEAU  $-01,U                    MOVE BUFFER POINTER BACK ONE
2220
2221                        * CALCULATE VALUE OF EXPONENT AND PUT IN STRING BUFFER
2222  915A E6 E0        L915A LDB   ,S+                       GET ORIGINAL EXPONENT OF FPA0
2223  915C 27 09              BEQ   L9167                     BRANCH IF EXPONENT = 0
2224  915E D6 47              LDB   V47                       GET BASE 10 EXPONENT
2225  9160 CB 09              ADDB  #$09                      ADD 9 FOR 9 SIGNIFICANT DIGIT CONVERSION
2226  9162 D0 D9              SUBB  VD9                       SUBTRACT LEFT DIGIT COUNTER
2227  9164 F0 03 DA           SUBB  STRBUF+3                  ADD ONE TO EXPONENT IF POST-SIGN FORCE
2228  9167 86 2B        L9167 LDA   #'+'                      PLUS SIGN
2229  9169 5D                 TSTB                            TEST EXPONENT
2230  916A 2A 03              BPL   L916F                     BRANCH IF POSITIVE EXPONENT
2231  916C 86 2D              LDA   #'-'                      MINUS SIGN
2232  916E 50                 NEGB                            CONVERT EXPONENT TO POSITIVE NUMBER
2233  916F A7 41        L916F STA   $01,U                     PUT SIGN OF EXPONENT IN STRING BUFFER
2234  9171 86 45              LDA   #'E'                      * PUT AN  E  (EXPONENTIATION FLAG) IN
2235  9173 A7 C1              STA   ,U++                      * BUFFER AND SKIP OVER THE SIGN
2236  9175 86 2F              LDA   #'0'-1                    INITIALIZE TENS DIGIT TO ASCII ZERO MINUS ONE
2237                        *CONVERT BINARY EXPONENT IN ACCB TO ASCII VALUE IN ACCA
2238  9177 4C           L9177 INCA                            ADD ONE TO TENS DIGIT COUNTER
2239  9178 C0 0A              SUBB  #10                       *SUBTRACT 10 FROM EXPONENT AND ADD ONE TO TENS
2240  917A 24 FB              BCC   L9177                     * DIGIT IF NO CARRY. TENS DIGIT DONE IF THERE IS A CARRY
2241  917C CB 3A              ADDB  #'9'+1                    ADD ASCII BIAS TO UNITS DIGIT
2242  917E ED C1              STD   ,U++                      SAVE EXPONENT IN BUFFER
2243  9180 6F C4              CLR   ,U                        CLEAR FINAL BYTE IN BUFFER - PRINT TERMINATOR
2244  9182 7E 90 54           JMP   L9054                     INSERT ASTERISK PADDING, FLOATING DOLLAR SIGN, ETC.
2245
2246                        * INSERT ASTERISK PADDING, FLOATING $ AND PRE-SIGN
2247  9185 8E 03 DB     L9185 LDX   #STRBUF+4                 POINT X TO START OF PRINT ITEM BUFFER
2248  9188 E6 84              LDB   ,X                        * GET SIGN BYTE OF ITEM-LIST BUFFER
2249  918A 34 04              PSHS  B                         * AND SAVE IT ON THE STACK
2250  918C 86 20              LDA   #SPACE                    DEFAULT PAD WITH BLANKS
2251  918E D6 DA              LDB   VDA                       * GET STATUS BYTE AND CHECK FOR
2252  9190 C5 20              BITB  #$20                      * ASTERISK LEFT PADDING
2253  9192 35 04              PULS  B                         GET SIGN BYTE AGAIN
2254  9194 27 08              BEQ   L919E                     BRANCH IF NO PADDING
2255  9196 86 2A              LDA   #'*'                      PAD WITH ASTERISK
2256  9198 C1 20              CMPB  #SPACE                    WAS THE FIRST BYTE A BLANK (POSITIVE)?
2257  919A 26 02              BNE   L919E                     NO
2258  919C 1F 89              TFR   A,B                       TRANSFER PAD CHARACTER TO ACCB
2259  919E 34 04        L919E PSHS  B                         SAVE FIRST CHARACTER ON STACK
2260  91A0 A7 80        L91A0 STA   ,X+                      STORE PAD CHARACTER IN BUFFER
2261  91A2 E6 84              LDB   ,X                        GET NEXT CHARACTER IN BUFFER
2262  91A4 27 10              BEQ   L91B6                     INSERT A ZERO IF END OF BUFFER
2263  91A6 C1 45              CMPB  #'E'                      * CHECK FOR AN  E  AND
2264  91A8 27 0C              BEQ   L91B6                     * PUT A ZERO BEFORE IT
2265  91AA C1 30              CMPB  #'0'                      = REPLACE LEADING ZEROES WITH
2266  91AC 27 F2              BEQ   L91A0                     = PAD CHARACTERS
2267  91AE C1 2C              CMPB  #','                      * REPLACE LEADING COMMAS
2268  91B0 27 EE              BEQ   L91A0                     * WITH PAD CHARACTERS
2269  91B2 C1 2E              CMPB  #'.'                      = CHECK FOR DECIMAL POINT
2270  91B4 26 04              BNE   L91BA                     = AND DON T PUT A ZERO BEFORE IT
2271  91B6 86 30        L91B6 LDA   #'0'                      * REPLACE PREVIOUS CHARACTER
2272  91B8 A7 82              STA   ,-X                       * WITH A ZERO
2273  91BA 96 DA        L91BA LDA   VDA                       = GET STATUS BYTE, CHECK
2274  91BC 85 10              BITA  #$10                      = FOR FLOATING $
2275  91BE 27 04              BEQ   L91C4                     = BRANCH IF NO FLOATING $
2276  91C0 C6 24              LDB   #'$'                      * STORE A $ IN
2277  91C2 E7 82              STB   ,-X                       * BUFFER
2278  91C4 84 04        L91C4 ANDA  #$04                      CHECK PRE-SIGN FLAG
2279  91C6 35 04              PULS  B                         GET SIGN CHARACTER
2280  91C8 26 02              BNE   L91CC                     RETURN IF POST-SIGN REQUIRED
2281  91CA E7 82              STB   ,-X                       STORE FIRST CHARACTER
2282  91CC 39           L91CC RTS
2283                        *
2284                        * CONVERT FPA0 INTO A NUMBER OF THE FORM - NNN,NNN,NNN X 10**M.
2285                        * THE EXPONENT M WILL BE RETURNED IN V47 (BASE 10 EXPONENT).
2286  91CD 34 40        L91CD PSHS  U                         SAVE BUFFER POINTER
2287  91CF 4F                 CLRA                            INITIAL EXPONENT OFFSET = 0
2288  91D0 97 47        L91D0 STA   V47                       SAVE EXPONENT OFFSET
2289  91D2 D6 4F              LDB   FP0EXP                    GET EXPONENT OF FPA0
2290  91D4 C1 80              CMPB  #$80                      * COMPARE TO EXPONENT OF .5
2291  91D6 22 11              BHI   L91E9                     * AND BRANCH IF FPA0 > = 1.0
2292
2293                        * IF FPA0 < 1.0, MULTIPLY IT BY 1E+09 UNTIL IT IS >= 1
2294  91D8 8E BD C0           LDX   #LBDC0                    POINT X TO FP NUMBER (1E+09)
2295  91DB BD BA CA           JSR   LBACA                     MULTIPLY FPA0 BY 1E+09
2296  91DE 96 47              LDA   V47                       GET EXPONENT OFFSET
2297  91E0 80 09              SUBA  #$09                      SUBTRACT 9 (BECAUSE WE MULTIPLIED BY 1E+09 ABOVE)
2298  91E2 20 EC              BRA   L91D0                     CHECK TO SEE IF > 1.0
2299  91E4 BD BB 82     L91E4 JSR   LBB82                     DIVIDE FPA0 BY 10
2300  91E7 0C 47              INC   V47                       INCREMENT EXPONENT OFFSET
2301  91E9 8E BD BB     L91E9 LDX   #LBDBB                    POINT X TO FP NUMBER (999,999,999)
2302  91EC BD BC A0           JSR   LBCA0                     COMPARE FPA0 TO X
2303  91EF 2E F3              BGT   L91E4                     BRANCH IF FPA0 > 999,999,999
2304  91F1 8E BD B6     L91F1 LDX   #LBDB6                    POINT X TO FP NUMBER (99,999,999.9)
```

```
2305   91F4 BD BC A0                JSR   LBCA0                          COMPARE FPA0 TO X
2306   91F7 2E 07                   BGT   L9200                          RETURN IF 999,999,999 > FPA0 > 99,999,999.9
2307   91F9 BD BB 6A                JSR   LBB6A                          MULTIPLY FPA0 BY 10
2308   91FC 0A 47                   DEC   V47                            DECREMENT EXPONENT OFFSET
2309   91FE 20 F1                   BRA   L91F1                          KEEP UNNORMALIZING
2310   9200 35 C0            L9200  PULS  U,PC                           RESTORE BUFFER POINTER AND RETURN
2311                         *
2312                         * CONVERT FPA0 INTO AN INTEGER, THEN DECODE IT
2313                         * INTO A DECIMAL ASCII STRING IN THE BUFFER
2314   9202 34 40            L9202  PSHS  U                              SAVE BUFFER POINTER
2315   9204 BD B9 B4                JSR   LB9B4                          ADD .5 TO FPA0 (ROUND OFF)
2316   9207 BD BC C8                JSR   LBCC8                          CONVERT FPA0 TO INTEGER FORMAT
2317   920A 35 40                   PULS  U                              RESTORE BUFFER POINTER
2318                         *
2319                         * CONVERT FPA0 INTO A DECIMAL ASCII STRING
2320   920C 8E BE C5                LDX   #LBEC5                         POINT X TO UNNORMALIZED POWERS OF 10
2321   920F C6 80                   LDB   #$80                           INITIALIZE DIGIT COUNTER TO 0 + $80.
2322                         * BIT 7 SET IS USED TO INDICATE THAT THE POWER OF 10 MANTISSA
2323                         * IS NEGATIVE. WHEN YOU  ADD  A NEGATIVE MANTISSA, IT IS
2324                         * THE SAME AS SUBTRACTING A POSITIVE ONE AND BIT 7 OF ACCB
2325                         * IS HOW THIS ROUTINE KNOWS THAT A  SUBTRACTION  IS OCCURRING.
2326   9211 8D 36            L9211  BSR   L9249                           CHECK FOR COMMA INSERTION
2327   9213 96 53            L9213  LDA   FPA0+3                         *  ADD  A POWER OF 10 MANTISSA TO FPA0.
2328   9215 AB 03                   ADDA  $03,X                          * IF THE MANTISSA IS NEGATIVE, A SUBTRACTION
2329   9217 97 53                   STA   FPA0+3                         * WILL BE WHAT REALLY TAKES PLACE.
2330   9219 96 52                   LDA   FPA0+2                         *
2331   921B A9 02                   ADCA  $02,X                          *
2332   921D 97 52                   STA   FPA0+2                         *
2333   921F 96 51                   LDA   FPA0+1                         *
2334   9221 A9 01                   ADCA  $01,X                          *
2335   9223 97 51                   STA   FPA0+1                         *
2336   9225 96 50                   LDA   FPA0                           *
2337   9227 A9 84                   ADCA  ,X                             *
2338   9229 97 50                   STA   FPA0                           *
2339   922B 5C                      INCB                                 ADD ONE TO DIGIT COUNTER
2340   922C 56                      RORB                                 ROTATE CARRY INTO BIT 7
2341   922D 59                      ROLB                                 * SET OVERFLOW FLAG - BRANCH IF CARRY SET AND
2342   922E 28 E3                   BVC   L9213                          * ADDING MANTISSA OR CARRY CLEAR AND SUBTRACTING MANTISSA
2343   9230 24 03                   BCC   L9235                          BRANCH IF SUBTRACTING MANTISSA
2344   9232 C0 0B                   SUBB  #10+1                          * TAKE THE 9 S COMPLEMENT
2345   9234 50                      NEGB                                 * IF ADDING MANTISSA
2346   9235 CB 2F            L9235  ADDB  #'0'-1                         ADD IN ASCII OFFSET
2347   9237 30 04                   LEAX  $04,X                          MOVE TO NEXT POWER OF 10 MANTISSA
2348   9239 1F 98                   TFR   B,A                            SAVE DIGIT IN ACCA
2349   923B 84 7F                   ANDA  #$7F                           MASK OFF ADD/SUBTRACT FLAG (BIT 7)
2350   923D A7 C0                   STA   ,U+                            STORE DIGIT IN BUFFER
2351   923F 53                      COMB                                 TOGGLE ADD/SUBTRACT FLAG
2352   9240 C4 80                   ANDB  #$80                           MASK OFF EVERYTHING BUT ADD/SUB FLAG
2353   9242 8C BE E9                CMPX  #LBEE9                         COMPARE TO END OF UNNORMALIZED POWERS OF 10
2354   9245 26 CA                   BNE   L9211                          BRANCH IF NOT DONE
2355   9247 6F C4                   CLR   ,U                             PUT A ZERO AT END OF INTEGER
2356
2357                         * DECREMENT DECIMAL POINT COUNTER AND CHECK FOR COMMA INSERTION
2358   9249 0A 45            L9249  DEC   V45                            DECREMENT DECIMAL POINT COUNTER
2359   924B 26 09                   BNE   L9256                          NOT TIME FOR DECIMAL POINT
2360   924D DF 39            L924D  STU   VARPTR                         SAVE BUFFER POINTER-POSITION OF THE DECIMAL POINT
2361   924F 86 2E                   LDA   #'.'                           * STORE A DECIMAL
2362   9251 A7 C0                   STA   ,U+                            * POINT IN THE OUTPUT BUFFER
2363   9253 0F D7                   CLR   VD7                            =CLEAR COMMA COUNTER - NOW IT WILL TAKE 255
2364                         *                                           =DECREMENTS BEFORE ANOTHER COMMA WILL BE INSERTED
2365   9255 39                      RTS
2366   9256 0A D7            L9256  DEC   VD7                            DECREMENT COMMA COUNTER
2367   9258 26 08                   BNE   L9262                          RETURN IF NOT TIME FOR COMMA
2368   925A 86 03                   LDA   #$03                           * RESET COMMA COUNTER TO 3; THREE
2369   925C 97 D7                   STA   VD7                            * DIGITS BETWEEN COMMAS
2370   925E 86 2C                   LDA   #','                           = PUT A COMMA INTO
2371   9260 A7 C0                   STA   ,U+                            = THE BUFFER
2372   9262 39               L9262  RTS
2373
2374                         * INITIALIZE DECIMAL POINT AND COMMA COUNTERS
2375   9263 96 47            L9263  LDA   V47                            GET THE BASE 10 EXPONENT OFFSET
2376   9265 8B 0A                   ADDA  #10                            * ADD 10 (FPA0 WAS  NORMALIZED  TO 9 PLACES LEFT
2377   9267 97 45                   STA   V45                            * OF DECIMAL POINT) - SAVE IN DECIMAL POINT COUNTER
2378   9269 4C                      INCA                                 ADD ONE FOR THE DECIMAL POINT
2379   926A 80 03            L926A  SUBA  #$03                           = DIVIDE DECIMAL POINT COUNTER BY 3; LEAVE
2380   926C 24 FC                   BCC   L926A                          = THE REMAINDER IN ACCA
2381   926E 8B 05                   ADDA  #$05                           CONVERT REMAINDER INTO A NUMBER FROM 1-3
2382   9270 97 D7                   STA   VD7                            SAVE COMMA COUNTER
2383   9272 96 DA                   LDA   VDA                            GET STATUS BYTE
2384   9274 84 40                   ANDA  #$40                           CHECK FOR COMMA FLAG
2385   9276 26 02                   BNE   L927A                          BRANCH IF COMMA FLAG ACTIVE
2386   9278 97 D7                   STA   VD7                            CLEAR COMMA COUNTER - 255 DIGITS OUTPUT BEFORE A COMMA
2387   927A 39               L927A  RTS
2388                         *
2389                         * INSERT ACCA ZEROES INTO THE BUFFER
2390   927B 34 02            L927B  PSHS  A                              SAVE ZEROES COUNTER
2391   927D 8D CA                   BSR   L9249                          CHECK FOR DECIMAL POINT
2392   927F 35 02                   PULS  A                              RESTORE ZEROES COUNTER
2393   9281 4A               L9281  DECA                                 * DECREMENT ZEROES COUNTER AND
2394   9282 2B 0A                   BMI   L928E                          * RETURN IF < 0
2395   9284 34 02                   PSHS  A                              SAVE ZEROES COUNTER
2396   9286 86 30                   LDA   #'0'                           * PUT A ZERO INTO
2397   9288 A7 C0                   STA   ,U+                            * THE BUFFER
2398   928A A6 E0                   LDA   ,S+                            RESTORE THE ZEROES COUNTER
2399   928C 26 ED                   BNE   L927B                          BRANCH IF NOT DONE
2400   928E 39               L928E  RTS
```

```
2401
2402                          ******* GRAPHICS PACKAGE ********
2403
2404                          * GET THE ADDRESS OF THE ROUTINE WHICH
2405                          * WILL CONVERT HOR & VER COORDINATES INTO
2406                          * AN ABSOLUTE RAM ADDRESS AND PIXEL MASK
2407                          * DEPENDING UPON THE CURRENT PMODE AND
2408                          * RETURN THE ADDRESS IN U.
2409                          *
2410  928F CE 92 9C           L928F    LDU   #L929C                      JUMP TABLE ADDRESS TO U
2411  9292 96 B6                       LDA   PMODE                       GET PMODE VALUE
2412  9294 48                          ASLA                              MUL ACCA X2 - 2 BYTES PER ADDRESS
2413  9295 EE C6                       LDU   A,U                         GET JUMP ADDRESS
2414  9297 39                          RTS
2415                          *
2416                          * CONVERT VER COORD (VERBEG) & NOR COORD (HORBEG) INTO
2417                          * ABSOLUTE SCREEN ADDR IN X AND PIXEL MASK IN ACCA.
2418  9298 8D F5              L9298    BSR   L928F                       GO GET JUMP ADDRESS
2419  929A 6E C4                       JMP   ,U                          GO TO IT
2420                          *
2421                          * JUMP TABLE -- HOR, VER COORD CONVERSION
2422  929C 92 A6              L929C    FDB   L92A6                       PMODE Ø
2423  929E 92 C2              L929E    FDB   L92C2                       PMODE 1
2424  92A0 92 A6              L92A0    FDB   L92A6                       PMODE 2
2425  92A2 92 C2              L92A2    FDB   L92C2                       PMODE 3
2426  92A4 92 A6              L92A4    FDB   L92A6                       PMODE 4
2427                          *
2428                          * HOR, VER COORD CONVERSION ROUTINE FOR 2
2429                          * COLOR HIRES GRAPHICS MODES
2430  92A6 34 44              L92A6    PSHS  U,B                         SAVE REGISTERS
2431  92A8 D6 B9                       LDB   HORBYT                      GET NUMBER BYTES/HOR GRAPHIC ROW
2432  92AA 96 CØ                       LDA   VERBEG+1                    GET VERTICAL COORDINATE
2433  92AC 3D                          MUL                               CALCULATE VERTICAL BYTE OFFSET
2434  92AD D3 BA                       ADDD  BEGGRP                      ADD IN START OF GRAPHIC PAGE
2435  92AF 1F Ø1                       TFR   D,X                         SAVE TEMP VALUE IN X REG
2436  92B1 D6 BE                       LDB   HORBEG+1                    GET HORIZONTAL COORDINATE
2437  92B3 54                          LSRB                              * THREE LSRBS EQUALS DIVIDE BY 8 -
2438  92B4 54                          LSRB                              * IN THE TWO COLOR MODE THERE ARE
2439  92B5 54                          LSRB                              * 8 PIXELS/BYTE
2440  92B6 3A                          ABX                               ADD HOR BYTE OFFSET
2441  92B7 96 BE                       LDA   HORBEG+1                    GET HORIZONTAL COORDINATE
2442  92B9 84 Ø7                       ANDA  #$Ø7                        *KEEP ONLY BITS Ø-2, WHICH CONTAIN THE NUMBER
2443                          *                                          *OF THE PIXEL IN THE BYTE
2444  92BB CE 92 DD                    LDU   #L92DD                      POINT U TO MASK LOOKUP TABLE
2445  92BE A6 C6                       LDA   A,U                         *GET PIXEL MASK - THE MASK WILL HAVE ONE BIT SET WHICH
2446                          *                                          *CORRESPONDS TO THE PIXEL SELECTED.
2447  92CØ 35 C4                       PULS  B,U,PC                      RESTORE REGISTERS
2448                          *
2449                          * HOR, VER COORDINATE CONVERSION ROUTINE
2450                          * FOR 4 COLOR HI RES GRAPHICS MODES
2451  92C2 34 44              L92C2    PSHS  U,B                         SAVE REGISTERS
2452  92C4 D6 B9                       LDB   HORBYT                      GET NUMBER BYTES/HOR GRAPHIC ROW
2453  92C6 96 CØ                       LDA   VERBEG+1                    GET VERTICAL COORDINATE
2454  92C8 3D                          MUL                               CALCULATE VERTICAL OFFSET
2455  92C9 D3 BA                       ADDD  BEGGRP                      ADD THE START OF GRAPHIC PAGE
2456  92CB 1F Ø1                       TFR   D,X                         SAVE IN X REGISTER
2457  92CD D6 BE                       LDB   HORBEG+1                    GET HORIZONTAL COORDINATE
2458  92CF 54                          LSRB                              TWO LSRBS = DIVIDE BY 4; IN THE 4
2459  92DØ 54                          LSRB                              COLOR MODE THERE ARE 4 PIXELS/BYTE
2460  92D1 3A                          ABX                               ADD HORIZONTAL BYTE OFFSET
2461  92D2 96 BE                       LDA   HORBEG+1                    GET HORIZONTAL COORDINATE
2462  92D4 84 Ø3                       ANDA  #$Ø3                        *KEEP ONLY BITS Ø,1 WHICH CONTAIN THE NUMBER OF THE PIXEL
2463                          *                                          *TO CHANGE (4 COLOR)
2464  92D6 CE 92 E5                    LDU   #L92E5                      POINT U TO MASK LOOKUP TABLE
2465  92D9 A6 C6                       LDA   A,U                         GET THE MASK FOR THE PROPER PIXEL
2466  92DB 35 C4                       PULS  B,U,PC                      RESTORE REGISTERS AND RETURN
2467
2468                          * 2 COLOR MODE PIXEL MASKS
2469  92DD 8Ø 4Ø 2Ø 1Ø Ø8 Ø4 L92DD    FCB   $8Ø,$4Ø,$2Ø,$1Ø,$Ø8,$Ø4
2470  92E3 Ø2 Ø1                       FCB   $Ø2,$Ø1
2471
2472                          * 4 COLOR MODE PIXEL MASKS
2473  92E5 CØ 3Ø ØC Ø3        L92E5    FCB   $CØ,$3Ø,$ØC,$Ø3
2474
2475                          * MOVE X REG DOWN ONE GRAPHIC ROW
2476  92E9 D6 B9              L92E9    LDB   HORBYT                      GET NUMBER BYTES/HOR ROW
2477  92EB 3A                          ABX                               ADD TO ABSOLUTE SCREEN POSITION
2478  92EC 39                          RTS
2479                          *
2480                          * ENTER W/ABSOLUTE SCREEN POSITION IN X, THE PIXEL
2481                          * MASK IN ACCA - ADJUST X AND ACCA TO THE NEXT
2482                          * PIXEL TO THE RIGHT IN THE TWO COLOR MODE.
2483  92ED 44                 L92ED    LSRA                              SHIFT ONE BIT TO RIGHT
2484  92EE 24 Ø3                       BCC   L92F3                       BRANCH IF IN SAME BYTE
2485  92FØ 46                          RORA                              IF YOU HAVE MOVED TO NEXT BYTE, SET BIT 7 IN ACCA
2486  92F1 3Ø Ø1                       LEAX  $Ø1,X                       AND ADD ONE TO X.
2487  92F3 39                 L92F3    RTS
2488                          *
2489                          *
2490                          * MOVE ABSOLUTE SCREEN ADDRESS OF CURRENT
2491                          * HOR, VER COORD ONE TO RIGHT AND ADJUST
2492                          * THE PIXEL MASK FOR THE 4 COLOR MODE
2493  92F4 44                 L92F4    LSRA                              SHIFT MASK ONE BIT TO RIGHT
2494  92F5 24 F6                       BCC   L92ED                       SHIFT RIGHT AGAIN IF SAME BYTE
2495  92F7 86 CØ                       LDA   #$CØ                        SET PIXEL #3 IF NEW BYTE
2496  92F9 3Ø Ø1                       LEAX  $Ø1,X                       ADD ONE TO ABS SCREEN POSITION
```

```
2497   92FB 39                    RTS
2498                      *
2499                      * EVALUATE TWO EXPRESSIONS - PUT THE FIRST
2500                      * VALUE (HOR COORD) IN HORBEG AND THE
2501                      * SECOND (VER COORD) IN VERBEG.
2502   92FC BD B7 34      L92FC   JSR  LB734                *EVALUATE TWO EXPRESSIONS FROM THE BASIC LINE -
2503                      *                                 *RETURN WITH THE 1ST VALUE IN BINVAL AND THE 2ND IN ACCB
2504   92FF 10 8E 00 BD           LDY  #HORBEG               POINT Y TO TEMP STORAGE LOC
2505   9303 C1 C0         L9303   CMPB #192                  IS VERT COORD > 191?
2506   9305 25 02                 BLO  L9309                 NO
2507   9307 C6 BF                 LDB  #191                  FORCE VER COORD TO 191
2508   9309 4F            L9309   CLRA                       HIGH ORDER BYTE OF VER COORD
2509   930A ED 22                 STD  $02,Y                 SAVE VERTICAL COORDINATE
2510   930C DC 2B                 LDD  BINVAL                GET RAW HORIZONTAL COORDINATE
2511   930E 10 83 01 00           CMPD #256                  IS IT WITHIN RANGE?
2512   9312 25 03                 BLO  L9317                 YES
2513   9314 CC 00 FF              LDD  #255                  FORCE IT TO 255 IF NOT IN RANGE
2514   9317 ED A4         L9317   STD  ,Y                    SAVE HORIZONTAL COORDINATE
2515   9319 39                    RTS
2516
2517                      * NORMALIZE HORIZONTAL AND VERTICAL COORDINATES FOR THE PROPER PHODE
2518                      * RETURN NORMALIZED VALUES IN (HORBEG,VERBEG)
2519 > 931A BD 92 FC      L931A   JSR  L92FC                 GO GET HOR & VER COORDINATES
2520   931D CE 00 BD      L931D   LDU  #HORBEG              POINT U TO HOR & VER COORDS
2521   9320 96 B6         L9320   LDA  PMODE                 GET PHODE
2522   9322 81 02                 CMPA #$02                  CHECK MODE
2523   9324 24 06                 BCC  L932C                 BRANCH IF > 1
2524   9326 EC 42                 LDD  $02,U                 GET THE VERT COORD
2525   9328 44                    LSRA                       *DIVIDE ACCD BY TWO SINCE
2526   9329 56                    RORB                       *PMODES 0&1 HAVE ONLY 96 VERT BLOCKS
2527   932A ED 42                 STD  $02,U                 SAVE NEW VERT COORD
2528   932C 96 B6         L932C   LDA  PMODE                 GET PMODE
2529   932E 81 04                 CMPA #$04                  CHECK PMODE
2530   9330 24 06                 BCC  L9338                 BRANCH IF PMODE = 4
2531   9332 EC C4                 LDD  ,U                    GET HOR COORD
2532   9334 44                    LSRA                       * DIVIDE HORIZONTAL COORDINATE
2533   9335 56                    RORB                       * BY 2-PMODES 0,1,2,3 HAVE ONLY 128 HOR BLOCKS
2534   9336 ED C4                 STD  ,U                    SAVE NEW HOR COORD
2535   9338 39            L9338   RTS
2536
2537                      * PPOINT
2538 > 9339 BD 93 B2      PPOINT  JSR  L93B2                 EVAL TWO EXPRESSIONS, RETURN VALUES IN (HORBEG,VERBEG)
2539 > 933C BD 93 1D             JSR  L931D                 NORMALIZE EXPRESSIONS FOR PROPER PMODE
2540   933F BD 92 98             JSR  L9298                 CONVERT COORDS INTO ABS SCREEN POSITION & PIXEL MASK
2541   9342 A4 84                ANDA ,X                    AND PIXEL MASK WITH CONTENTS OF SCREEN
2542   9344 D6 B6                LDB  PMODE                 GET CURRENT PMODE
2543   9346 56                   RORB                       SHIFT RIGHT
2544   9347 24 12                BCC  L935B                 BRANCH IF PMODE 0,2,4 (2 COLOR)
2545   9349 81 04         L9349  CMPA #$04                  * IS THE  ON  PIXEL IN THE 2 RIGHTMOST BITS?
2546   934B 25 04                BLO  L9351                 BRANCH IF IT IS
2547   934D 46                   RORA                       =SHIFT RIGHT
2548   934E 46                   RORA                       =ONE PIXEL
2549   934F 20 F8                BRA  L9349                 KEEP CHECKING
2550   9351 4C            L9351  INCA                       ADD 1 TO COLOR - BASIC USES 1 TO 4, NOT 0 TO 3
2551   9352 48                   ASLA                       TIMES 2
2552   9353 9B C1                ADDA CSSVAL                ADD COLOR SET (0 OR 8)
2553   9355 44                   LSRA                       DIVIDE ACCB BY TWO - COLORS RANGE FROM 0 - 8
2554   9356 1F 89         L9356  TFR  A,B                   *TRANSFER COLOR INFO INTO ACCB
2555   9358 7E B4 F3             JMP  LB4F3                 *CONVERT ACCB TO FP NUMBER
2556   935B 4D            L935B  TSTA                       IS 2 COLOR PIXEL ON?
2557   935C 27 F8                BEQ  L9356                 NO
2558   935E 4F                   CLRA                       * FORCE THE TWO-COLOR  ON  VALUE TO BE
2559   935F 20 F0                BRA  L9351                 * ONE OR FIVE (DEPENDING ON CSS).
2560
2561                      * PSET
2562   9361 86 01         PSET   LDA  #$01                   PSET FLAG
2563   9363 20 01                BRA  L9366
2564
2565                      * PRESET
2566   9365 4F            PRESET CLRA                       PRESET FLAG
2567   9366 97 C2         L9366  STA  SETFLG                 STORE FLAG 0 = PRESET, 1 = PSET
2568   9368 BD B2 6A             JSR  LB26A                 SYNTAX CHECK FOR  (
2569 > 936B BD 93 1A             JSR  L931A                 EVAL HOR & VER COORDS AND NORMALIZE
2570   936E BD 95 81             JSR  L9581                 *EVALUATE COLOR - RETURN IN WCOLOR; ALLCOL
2571                      *                                 *WILL BE ONE BYTE WITH ALL PIXELS SET TO THAT COLOR
2572   9371 BD B2 67             JSR  LB267                 SYNTAX CHECK FOR  )
2573   9374 BD 92 98             JSR  L9298                 *CALCULATE THE ABSOLUTE ADDRESS OF THE
2574                      *                                 *BYTE TO PSET/PRESET - RETURN ADDRESS IN X - THE MASK
2575                      *                                 *OF PIXEL TO CHANGE RETURNED IN ACCA SET A PIXEL ON
2576                      *                                 *SCREEN - ABS POSIT IN X, MASK IN ACCA, COLOR IN ALLCOL
2577
2578                      * TURN ON THE PIXEL (POINTED TO BY X, PIXEL MASK IN ACCA) TO THE COLOR
2579                      * IN ALLCOL.  SET CHGFLG <> 0 IF THE PIXEL COLOR IS CHANGED.
2580   9377 E6 84         L9377  LDB  ,X                    GET BYTE FROM THE SCREEN
2581   9379 34 04                PSHS B                     SAVE IT ON STACK
2582   937B 1F 89                TFR  A,B                   PUT PIXEL MASK IN ACCB
2583   937D 43                   COMA                       INVERT PIXEL MASK
2584   937E A4 84                ANDA ,X                    *  AND  WITH SCREEN DATA - KEEP ALL PIXELS
2585                                                        * EXCEPT THE ONE TO MODIFY
2586   9380 D4 B5                ANDB ALLCOL                CONVERT PIXEL IN THE PIXEL MASK TO THE PROPER COLOR
2587   9382 34 04                PSHS B                     SAVE IT ON STACK
2588   9384 AA E0                ORA  ,S+                    OR  IT INTO THE REST OF THE PIXELS
2589   9386 A7 84                STA  ,X                    PUT IT ON SCREEN
2590   9388 A0 E0                SUBA ,S+                   SUBTRACT OLD BYTE FROM NEW BYTE; ACCA=0 IF NEW BYTE = OLD BYTE
2591   938A 9A DB                ORA  CHGFLG                 OR  DIFFERENCE WITH CHANGE FLAG
2592   938C 97 DB                STA  CHGFLG                SAVE IT - CHGFLG WILL BE = 0 IF THE GRAPHIC BYTE IS UNCHANGED
```

```
2593   938E 39                  RTS
2594                    * EVALUATE TWO SETS OF COORDINATES SEPARATED BY A MINUS
2595                    * SIGN.  PUT 1ST SET OF COORDS AT (HORBEG,VERBEG), SECOND
2596                    * SET AT (HOREND,VEREND). IF NOTHING BEFORE MINUS SIGN, PUT
2597                    * (HORDEF,VERDEF) AT (HORBEG, VERBEG)
2598   938F 9E C7       L938F   LDX   HORDEF                    GET LAST HORIZ END POINT
2599   9391 9F BD               STX   HORBEG                    PUT IN START POINT STORAGE LOC
2600   9393 9E C9               LDX   VERDEF                    GET LAST VERT END POINT
2601   9395 9F BF               STX   VERBEG                    PUT IN START POINT VERT STORAGE LOC
2602   9397 81 AC               CMPA  #$AC                      TOKEN FOR MINUS SIGN
2603   9399 27 03               BEQ   L939E                     BRANCH IF NO STARTING COORDINATES GIVEN
2604 > 939B BD 93 B2            JSR   L93B2                     GO GET STARTING COORDINATES
2605   939E C6 AC       L939E   LDB   #$AC                      TOKEN FOR MINUS SIGN
2606   93A0 BD B2 6F            JSR   LB26F                     GO DO A SYNTAX CHECK
2607   93A3 BD B2 6A            JSR   LB26A                     SYNTAX CHECK FOR A  (
2608   93A6 BD B7 34            JSR   LB734                     EVALUATE 2 EXPRESSIONS
2609   93A9 10 8E 00 C3         LDY   #HOREND                   TEMP STORAGE LOCATION FOR END COORDINATES OF LINE COMMAND
2610   93AD BD 93 03            JSR   L9303                     GET END POINT COORDS
2611   93B0 20 06               BRA   L93B8                     CHECK SYNTAX FOR  )
2612   93B2 BD B2 6A    L93B2   JSR   LB26A                     SYNTAX CHECK FOR  (
2613   93B5 BD 92 FC            JSR   L92FC                     EVALUATE 2 EXPRESSIONS
2614   93B8 7E B2 67    L93B8   JMP   LB267                     SYNTAX CHECK FOR   )  AND RETURN
2615
2616                    * LINE
2617   93BB 81 89       LINE    CMPA  #$89                       INPUT  TOKEN
2618   93BD 10 27 F5 FF         LBEQ  L89C0                     GO DO  LINE INPUT  COMMAND
2619   93C1 81 28               CMPA  #'('                      CHECK FOR  (
2620   93C3 27 09               BEQ   L93CE                     GO LOOK FOR START AND END POINTS
2621   93C5 81 AC               CMPA  #$AC                      CHECK TOKEN FOR MINUS SIGN
2622   93C7 27 05               BEQ   L93CE                     GO GET START AND END POINTS
2623   93C9 C6 40               LDB   #'@'                      CHECK FOR @ SIGN
2624   93CB BD B2 6F            JSR   LB26F                     DO A SYNTAX CHECK
2625 > 93CE BD 93 8F    L93CE   JSR   L938F                     GET STARTING AND ENDING COORDINATES
2626   93D1 9E C3               LDX   HOREND                    GET ENDING HORIZ COORDINATE
2627   93D3 9F C7               STX   HORDEF                    PUT IN LAST USED HOR END POINT
2628   93D5 9E C5               LDX   VEREND                    GET ENDING VER COORD
2629   93D7 9F C9               STX   VERDEF                    PUT IN LAST USED VER END POINT
2630   93D9 BD B2 6D            JSR   SYNCOMMA                  SYNTAX CHECK FOR COMMA
2631   93DC 81 BE               CMPA  #$BE                      PRESET TOKEN?
2632   93DE 27 09               BEQ   L93E9                     YES
2633   93E0 81 BD               CMPA  #$BD                      PSET TOKEN?
2634   93E2 10 26 1E 91         LBNE  LB277                      SYNTAX ERROR  IF NOT PSET OR PRESET
2635   93E6 C6 01               LDB   #$01                      PSET FLAG
2636   93E8 86          L93E8   FCB   SKP1LD                    SKIP ONE BYTE, LOAD ACCA WITH $5F
2637   93E9 5F          L93E9   CLRB                            PRESET FLAG
2638   93EA 34 04               PSHS  B                         SAVE PSET/PRESET FLAG
2639   93EC 9D 9F               JSR   GETNCH                    GET ANOTHER CHAR
2640 > 93EE BD 94 20            JSR   L9420                     NORMALIZE START/END COORDS
2641   93F1 35 04               PULS  B                         GET PSET/PRESET FLAG
2642   93F3 D7 C2               STB   SETFLG                    SAVE IT
2643   93F5 BD 95 9A            JSR   L959A                     SET ACTIVE COLOR BYTE
2644   93F8 9D A5               JSR   GETCCH                    GET ANOTHER CHARACTER
2645   93FA 10 27 00 A3         LBEQ  L94A1                     BRANCH IF NO BOX TO BE DRAWN
2646   93FE BD B2 6D            JSR   SYNCOMMA                  SYNTAX CHECK FOR COMMA
2647   9401 C6 42               LDB   #'B'                       BOX ?
2648   9403 BD B2 6F            JSR   LB26F                     GO DO A SYNTAX CHECK FOR A  B
2649   9406 26 21               BNE   L9429                     FOUND A  B  AND SOMETHING FOLLOWS
2650   9408 8D 3A               BSR   L9444                     DRAW A HORIZ LINE
2651   940A 8D 62               BSR   L946E                     DRAW A VERTICAL LINE
2652   940C 9E BD               LDX   HORBEG                    GET HOR START COORD
2653   940E 34 10               PSHS  X                         SAVE IT ON STACK
2654   9410 9E C3               LDX   HOREND                    * GET HORIZONTAL END COORDINATE AND
2655   9412 9F BD               STX   HORBEG                    * PUT THEM IN HORIZONTAL START COORDINATE
2656   9414 8D 58               BSR   L946E                     DRAW VERTICAL LINE
2657   9416 35 10               PULS  X                         GET THE PREVIOUS HORIZONTAL START COORDINATE
2658   9418 9F BD               STX   HORBEG                    RESTORE IT
2659   941A 9E C5               LDX   VEREND                    GET VER END COORD
2660   941C 9F BF               STX   VERBEG                    PUT INTO START
2661   941E 20 24               BRA   L9444                     DRAW HORIZ LINE
2662                    * NORMALIZE START COORDS IN (HORBEG,VERBEG) & END COORDS IN (HOREND,VEREND)
2663   9420 BD 93 1D    L9420   JSR   L931D                     NORMALIZE COORDS IN (HORBEG,VERBEG)
2664   9423 CE 00 C3            LDU   #HOREND                   =
2665   9426 7E 93 20            JMP   L9320                     = NORMALIZE COORDS IN (HOREND,VEREND)
2666   9429 C6 46       L9429   LDB   #'F'                      *
2667   942B BD B2 6F            JSR   LB26F                     *GO DO A SYNTAX CHECK FOR AN  F
2668   942E 20 04               BRA   L9434                      FILL  THE BOX
2669   9430 30 1F       L9430   LEAX  $-01,X                    MOVE VER COORD UP ONE
2670   9432 9F BF       L9432   STX   VERBEG                    SAVE NEW VERTICAL START COORDINATE
2671
2672                    * DRAW A SERIES OF HORIZONTAL LINES FROM VER START TO VER END
2673 > 9434 BD 94 44    L9434   JSR   L9444                     DRAW A HORIZ LINE
2674   9437 9E BF               LDX   VERBEG                    GET START VER COORD
2675   9439 9C C5               CMPX  VEREND                    COMPARE TO END VER COORD
2676   943B 27 06               BEQ   L9443                     RETURN IF EQUAL
2677   943D 24 F1               BCC   L9430                     BRANCH IF START HOR > END HOR
2678   943F 30 01               LEAX  $01,X                     MOVE HOR COORD DOWN ONE
2679   9441 20 EF               BRA   L9432                     KEEP DRAWING LINES
2680   9443 39          L9443   RTS                             WASTED BYTE - SHOULD USE L946B INSTEAD
2681                    * DRAW A HORIZONTAL LINE FROM HOREND TO HORBEG
2682                    * AT VER COORD VERBEG; COLOR IN ALLCOL
2683   9444 9E BD       L9444   LDX   HORBEG                    GET STARTING COORDS
2684   9446 34 10               PSHS  X                         SAVE  EM
2685   9448 BD 97 1D            JSR   L971D                     GET ABSOLUTE VALUE OF HOREND - HORBEG (HORIZONTAL COORDINATE)
2686   944B 24 04               BCC   L9451                     BRANCH IF END > START
2687   944D 9E C3               LDX   HOREND                    * TRANSFER END COORD TO START
2688   944F 9F BD               STX   HORBEG                    *
```

```
2689   9451 1F Ø2            L9451    TFR   D,Y                         SAVE DIFFERENCE IN Y
2690   9453 31 21                     LEAY  $Ø1,Y                       ADD ONE TO DIFFERENCE - TURN ON STARTING & ENDING COORDS
2691   9455 BD 92 98                  JSR   L9298                       GET ABS SCREEN POS TO X AND PIXEL MASK TO ACCA
2692   9458 35 4Ø                     PULS  U                           GET START COORDS
2693   945A DF BD                     STU   HORBEG                      RESTORE THEM
2694   945C 8D 36                     BSR   L9494                       POINT U TO ROUTINE TO MOVE PIXEL POINTERS TO RIGHT
2695   945E 97 D7            L945E    STA   VD7                         SAVE PIXEL MASK
2696   9460 BD 93 77                  JSR   L9377                       TURN ON PIXEL
2697   9463 96 D7                     LDA   VD7                         GET OLD PIXEL MASK
2698   9465 AD C4                     JSR   ,U                          MOVE TO NEXT ONE TO RIGHT
2699   9467 31 3F                     LEAY  $-Ø1,Y                      DEC COUNTER
2700   9469 26 F3                     BNE   L945E                       LOOP IF NOT DONE
2701   946B 39               L946B    RTS
2702   946C 35 Ø6            L946C    PULS  A,B                         CLEAN UP STACK
2703
2704                         * DRAW A VERTICAL LINE FROM VEREND TO VERBEG AT HOR COORD HORBEG
2705   946E DC BF            L946E    LDD   VERBEG                      GET END COORDS
2706   9470 34 Ø6                     PSHS  B,A                         SAVE THEM
2707   9472 BD 97 1Ø                  JSR   L971Ø                       CALCULATE ABSOLUTE VALUE OF VEREND-VERBEG
2708   9475 24 Ø4                     BCC   L947B                       BRANCH IF END COORD > START COORD
2709   9477 9E C5                     LDX   VEREND                      *
2710   9479 9F BF                     STX   VERBEG                      *SWITCH VER COORDS IF END COORD IS TO RIGHT OF START
2711   947B 1F Ø2            L947B    TFR   D,Y                         LENGTH OF LINE TO Y
2712   947D 31 21                     LEAY  $Ø1,Y                       SET BOTH START AND END COORDS
2713   947F BD 92 98                  JSR   L9298                       GET ABSOLUTE SCREEN POS TO X, MASK TO ACCA
2714   9482 35 4Ø                     PULS  U                           GET END COORD
2715   9484 DF BF                     STU   VERBEG                      RESTORE THEM
2716   9486 8D 15                     BSR   L949D                       POINT U TO ROUTINE TO MOVE DOWN ONE ROW
2717   9488 2Ø D4                     BRA   L945E                       DRAW VERT LINE
2718                         *
2719                         * JUMP TABLE OF ADDRESSES OF ROUTINES WHICH WILL MOVE THE
2720                         * ABSOLUTE SCREEN ADDRESS POINTER ONE PIXEL TO THE RIGHT.
2721   948A 92 ED            L948A    FDB   L92ED                       PMODE Ø
2722   948C 92 F4            L948C    FDB   L92F4                       PMODE 1
2723   948E 92 ED            L948E    FDB   L92ED                       PMODE 2
2724   9490 92 F4            L9490    FDB   L92F4                       PMODE 3
2725   9492 92 ED            L9492    FDB   L92ED                       PMODE 4
2726
2727                         * POINT U TO ROUTINE WHICH WILL MOVE PIXEL ONE TO RIGHT
2728   9494 CE 94 8A         L9494    LDU   #L948A                      POINT TO JUMP TABLE
2729   9497 D6 B6                     LDB   PMODE                       GET PMODE VALUE
2730   9499 58                        ASLB                              X2
2731   949A EE C5                     LDU   B,U                         GET JUMP ADDRESS
2732   949C 39                        RTS
2733   949D CE 92 E9         L949D    LDU   #L92E9                      POINT U TO ROUTINE TO MOVE ABS POS DOWN ONE ROW
2734   94A0 39                        RTS
2735
2736                         * DRAW LINE FROM (HORBEG.VERBEG) TO (HOREND,VEREND)
2737   94A1 1Ø 8E 95 ØD      L94A1    LDY   #L95ØD                      POINT Y TO INCR VERBEG
2738   94A5 BD 97 1Ø                  JSR   L971Ø                       CALCULATE VEREND - VERBEG (VERTICAL DIFFERENCE)
2739 > 94A8 1Ø 27 FF 98               LBEQ  L9444                       DRAW A HORIZONTAL LINE IF DELTA V = Ø
2740   94AC 24 Ø4                     BCC   L94B2                       BRANCH IF VER END COORD > VER START COORD
2741   94AE 1Ø 8E 95 1B               LDY   #L951B                      POINT Y TO DECR VER COORD (VERBEG)
2742   94B2 34 Ø6            L94B2    PSHS  B,A                         SAVE DELTA V
2743   94B4 CE 95 Ø6                  LDU   #L95Ø6                      POINT U TO INCR HOR COORD
2744   94B7 BD 97 1D                  JSR   L971D                       CALCULATE HOREND-HORBEG (HOR DIFFERENCE)
2745   94BA 27 BØ                     BEQ   L946C                       DRAW A VERTICAL LINE IF DELTA H = Ø
2746   94BC 24 Ø3                     BCC   L94C1                       BRANCH IF HOR END COORD > HOR START COORD
2747   94BE CE 95 14                  LDU   #L9514                      POINT U TO DECR HOR COORD
2748   94C1 1Ø A3 E4         L94C1    CMPD  ,S                          COMPARE DELTA H TO DELTA V
2749   94C4 35 1Ø                     PULS  X                           PUT DELTA V IN X
2750   94C6 24 Ø4                     BCC   L94CC                       BRANCH IF DELTA H > DELTA V
2751   94C8 1E 32                     EXG   U,Y                         SWAP CHANGE HOR & CHANGE VER ADDRESS
2752   94CA 1E Ø1                     EXG   D,X                         EXCHANGE DELTA HOR & DELTA VER
2753   94CC 34 46            L94CC    PSHS  U,B,A                       *SAVE THE LARGER OF DELTA V, DELTA H
2754                         *                                          *AND THE INCREMENT/DECREMENT ADDRESS
2755   94CE 34 Ø6                     PSHS  B,A                         SAVE WHICHEVER IS LARGER OF DELTA V, DELTA H
2756   94DØ 44                        LSRA                              *
2757   94D1 56                        RORB                              * DIVIDE BY 2, SHIFT ACCD RIGHT ONE BIT
2758   94D2 25 Ø9                     BLO   L94DD                       BRANCH IF ODD NUMBER
2759   94D4 11 83 95 ØE               CMPU  #L95ØD+1                    SEE IF INCR OR DECR
2760   94D8 25 Ø3                     BLO   L94DD                       BRANCH IF INCR
2761   94DA 83 ØØ Ø1                  SUBD  #1                          SUBTRACT 1 IF DECREMENT
2762   94DD 34 16            L94DD    PSHS  X,B,A                       *SAVE SMALLEST DELTA (X) AND INITIAL MINOR COORDINATE
2763                                                                     *INCREMENT COUNTER WHICH IS 1/2 OF LARGEST DELTA
2764   94DF BD 92 8F                  JSR   L928F                       POINT U TO PROPER COORDINATE TO SCREEN CONVERSION ROUTINE
2765
2766                         * DRAW THE LINE HERE - AT THIS POINT THE STACK HAS THE DRAW DATA ON IT
2767
2768                         * Ø 1,S=MINOR COORDINATE INCREMENT COUNTER
2769                         * 2 3,S=ASSOLUTE VALUE OF THE SMALLEST DELTA COORDINATE
2770                         * 4 5,S=ABSOLUTE VALUE OF THE LARGEST DELTA COORDINATE
2771                         * 6 7,S=LARGEST COORDINATE COUNTER (HOW MANY TIMES THROUGH THE DRAW
2772                         *       LOOP. INITIALLY SET TO ABSOLUTE VALUE OF LARGEST DELTA COORD
2773                         * 8 9,S=ADDRESS OF THE ROUTINE WHICH WILL INCREMENT OR DECREMENT
2774                         *       THE LARGEST DELTA COORDINATE
2775
2776   94E2 AD C4            L94E2    JSR   ,U                          CONVERT (X,Y) COORDINATES TO ABSOLUTE SCREEN ADDRESS
2777   94E4 BD 93 77                  JSR   L9377                       TURN ON A PIXEL
2778   94E7 AE 66                     LDX   $Ø6,S                       GET DISTANCE COUNTER
2779   94E9 27 17                     BEQ   L95Ø2                       BRANCH IF LINE IS COMPLETELY DRAWN
2780   94EB 3Ø 1F                     LEAX  $-Ø1,X                      DECR ONE
2781   94ED AF 66                     STX   $Ø6,S                       SAVE IT
2782   94EF AD F8 Ø8                  JSR   [$Ø8,S]                     INCR/DECR COORDINATE-WHICH HAS THE LARGEST DELTA
2783   94F2 EC E4                     LDD   ,S                          GET THE MIHOR COORDINATE INCREMENT COUNTER
2784   94F4 E3 62                     ADDD  $Ø2,S                       ADD THE SMALLEST DIFFERENCE
```

```
2785   94F6 ED E4                STD   ,S                          SAVE NEW MINOR COORDINATE INCREMENT COUNTER
2786   94F8 A3 64                SUBD  $Ø4,S                       *SUBTR OUT THE LARGEST DIFFERENCE AND
2787   94FA 25 E6                BLO   L94E2                       *BRANCH IF RESULT NOT > LARGEST DIFFERENCE
2788   94FC ED E4                STD   ,S                          IF >=, THEN STORE NEW MIHOR COORDINATE INCREMENT
2789   94FE AD A4                JSR   ,Y                          INCREMENT/DECREMENT COORDINATE WHICH HAS THE SMALLEST DELTA
2790   95ØØ 2Ø E0                BRA   L94E2                       KEEP GOING
2791   95Ø2 35 1Ø          L95Ø2 PULS  X                           *
2792   95Ø4 35 F6                PULS  A,B,X,Y,U,PC                *CLEAN UP THE STACK AND RETURN
2793                       *
2794                       * THESE ROUTINES ARE USED TO INCREMENT OR DECREMENT THE
2795                       * HORIZONTAL & VERTICAL COORDINATES. THEY NEED TO BE KEPT
2796                       * IN THIS ORDER (INCR,INCR,DECR,DECR).
2797
2798                       * INCR HORBEG (HOR COORD)
2799   95Ø6 9E BD          L95Ø6 LDX   HORBEG                      GET COORDINATE
2800   95Ø8 3Ø Ø1                LEAX  $Ø1,X                       ADD ONE
2801   95ØA 9F BD                STX   HORBEG                      SAVE COORDINATE
2802   95ØC 39                   RTS
2803                       * INCR VERBEG (VER COORD)
2804   95ØD 9E BF          L95ØD LDX   VERBEG                      GET COORDINATE
2805   95ØF 3Ø Ø1                LEAX  $Ø1,X                       ADD ONE
2806   9511 9F BF                STX   VERBEG                      SAVE COORDINATE
2807   9513 39                   RTS
2808                       * DECR HORSES (HOR COORD)
2809   9514 9E BD          L9514 LDX   HORBEG                      GET COORDINATE
2810   9516 3Ø 1F                LEAX  $-Ø1,X                      SUBTRACT ONE
2811   9518 9F BD                STX   HORBEG                      SAVE COORDINATE
2812   951A 39                   RTS
2813                       * DECR VERBEG (VER COORD)
2814   951B 9E BF          L951B LDX   VERBEG                      GET COORDINATE
2815   951D 3Ø 1F                LEAX  $-Ø1,X                      SUBTRACT ONE
2816   951F 9F BF                STX   VERBEG                      SAVE COORDINATE
2817   9521 39                   RTS
2818                       *
2819                       * GET MAXIMUM VALUE OF HOR/VER COORDINATES
2820                       * NORMALIZED FOR PROPER PMODE.  RETURN VALUES
2821                       * HOR = VD3  VER = VD5
2822   9522 CE ØØ D3       L9522 LDU   #VD3                        POINT U TO TEMP STORAGE AREA (VD3)
2823   9525 8E ØØ FF             LDX   #255                        MAXIMUM VALUE HORIZONTAL COORD (255)
2824   9528 AF C4                STX   ,U                          SAVE IT
2825   952A 8E ØØ BF             LDX   #191                        MAXIMUM VALUE VERTICAL COORD (191)
2826   952D AF 42                STX   $Ø2,U                       SAVE IT
2827   952F 7E 93 2Ø             JMP   L932Ø                       GO CONVERT THEM TO PROPER PMODE
2828
2829                       * PCLS
2830   9532 27 ØE          PCLS  BEQ   L9542                       CLEAR TO BACKGROUND COLOR IF NO ARGUMENT
2831   9534 8D 24                BSR   L955A                       EVALUATE EXPRESSION, CONVERT TO PROPER COLOR CODE
2832   9536 86 55          L9536 LDA   #$55                        CONSIDER EACH BYTE AS 4 GROUPS OF 2 BIT SUB-NIBBLES
2833   9538 3D                   MUL                               MULT BY COLOR
2834   9539 9E BA                LDX   BEGGRP                      GET STARTING ADOR
2835   953B E7 8Ø          L953B STB   ,X+                         SET BYTE TO PROPER COLOR
2836   953D 9C B7                CMPX  ENDGRP                      AT END OF GRAPHIC PAGE?
2837   953F 26 FA                BNE   L953B                       NO
2838   9541 39                   RTS
2839   9542 D6 B3          L9542 LDB   BAKCOL                      GET BACKGROUND COLOR
2840   9544 2Ø F0                BRA   L9536
2841
2842                       * COLOR
2843   9546 81 2C          COLOR CMPA  #','                        *CHECK FOR COMMA AND
2844   9548 27 Ø8                BEQ   L9552                       *BRANCH IF FOREGROUND COLOR ARGUMENT MISSING
2845   954A 8D ØE                BSR   L955A                       EVALUATE FIRST ARGUMENT
2846   954C D7 B2                STB   FORCOL                      STORE IN FOREGROUND LOCATION
2847   954E 9D A5                JSR   GETCCH                      GET NEXT INPUT CHARACTER
2848   9550 27 Ø7                BEQ   L9559                       RETURN IF NONE
2849   9552 BD B2 6D       L9552 JSR   SYNCOMMA                    SYNTAX CHECK FOR COMMMA
2850   9555 8D Ø3                BSR   L955A                       EVALUATE LAST ARGUMENT
2851   9557 D7 B3                STB   BAKCOL                      STORE IN BACKGROUND COLOR
2852   9559 39             L9559 RTS
2853
2854                       * EVALUATE AN EXPRESSION AND CONVERT IT TO A PROPER COLOR CODE
2855                       * DEPENDING ON THE PMODE AND CSS; ILLEGAL FUNCTION CALL IF > 8 -
2856                       * RETURN COLOR VALUE IN ACCB; CSS VALUE IN ACCA
2857   955A BD B7 ØB       L955A JSR   EVALEXPB                    EVALUATE EXPRESSION
2858   955D C1 Ø9          L955D CMPB  #$Ø9                        ONLY ALLOW Ø-8
2859   955F 1Ø 24 1E E7          LBCC  LB44A                        ILLEGAL FUNCTION CALL  IF BAD COLOR
2860   9563 4F                   CLRA                              VDG CSS VALUE FOR FIRST COLOR SET
2861   9564 C1 Ø5                CMPB  #$Ø5                        FIRST OR SECOND COLOR SET?
2862   9566 25 Ø4                BLO   L956C                       BRANCH IF FIRST SET
2863   9568 86 Ø8                LDA   #$Ø8                        VDG CSS VALUE FOR SECOND COLOR SET
2864   956A C0 Ø4                SUBB  #$Ø4                        MAKE 5-8 BECOME 1-4
2865   956C 34 Ø2          L956C PSHS  A                           SAVE VDG CSS VALUE ON THE STACK
2866   956E 96 B6                LDA   PMODE                       GET PMODE
2867   9570 46                   RORA                              4 COLOR OR 2 COLOR
2868   9571 24 Ø8                BCC   L957B                       2 COLOR
2869   9573 5D                   TSTB                              WAS COLOR = Ø
2870   9574 26 Ø2                BNE   L9578                       NO
2871   9576 C6 Ø4          L9576 LDB   #$Ø4                        IF SO, MAKE IT 4
2872   9578 5A             L9578 DECB                            CONVERT 1-4 TO Ø-3
2873   9579 35 82          L9579 PULS  A,PC                        PUT VDG CSS VALUE IN ACCA AND RETURN
2874   957B 56             L957B RORB                            CHECK ONLY THE LSB OF COLOR IF IN 2 COLOR MODE
2875   957C 25 F8                BLO   L9576                       BRANCH IF ODD - FORCE ACCB TO 3
2876   957E 5F                   CLRB                              FORCE ACCB = Ø IF EVEN
2877   957F 2Ø F8                BRA   L9579                       RETURN
2878
2879                       * SET THE CURRENT ACTIVE COLOR AND ALL PIXEL BYTE
2880                       * TO FOREGROUND/BACKGROUND COLOR DEPENDING ON
```

```
2881                        * PSET, PRESET IF NO EXPRESSION ,  )  OR
2882                        *  , .  OTHERWISE EVALUATE THE EXPRESSION
2883 > 9581 BD 95 9A        L9581   JSR   L959A                       GET THE COLOR OF A BYTE
2884   9584 9D A5                   JSR   GETCCH                      CHECK CURRENT INPUT CHARACTER
2885   9586 27 1Ø                   BEQ   L9598                       BRANCH IF NONE
2886   9588 81 29                   CMPA  #')'                        * CHECK FOR  )  AND BRANCH IF
2887   958A 27 ØC                   BEQ   L9598                       * NO MORE ARGUMENTS
2888   958C BD B2 6D                JSR   SYNCOMMA                    SYNTAX CHECK FOR COMMA
2889   958F 81 2C                   CMPA  #','                        WAS NEXT CHARACTER A COMMA?
2890   9591 27 Ø5                   BEQ   L9598                       YES
2891 > 9593 BD 95 5A                JSR   L955A                       EVALUATE EXPRESSION, RETURN COLOR IN ACCB
2892   9596 8D ØA                   BSR   L95A2                       TEMP STORE COLOR AND ALL PIXEL BYTE
2893   9598 ØE A5                   JMP   GETCCH                      CHECK INPUT CHARACTER AND RETURN
2894
2895                        * SET THE ACTIVE COLOR BYTE AND THE ALL ACTIVE COLOR BYTE
2896   959A D6 B2           L959A   LDB   FORCOL                       GET FOREGROUND COLOR
2897   959C ØD C2                   TST   SETFLG                      CHECK PSET/PRESET FLAG
2898   959E 26 Ø2                   BNE   L95A2                       BRANCH IF PSET
2899   95AØ D6 B3                   LDB   BAKCOL                      GET BACKGROUND COLOR
2900   95A2 D7 B4           L95A2   STB   WCOLOR                      TEMP STORE COLOR
2901   95A4 86 55                   LDA   #$55                        CONSIDER A BYTE AS 4 PIXELS
2902   95A6 3D                      MUL                               SET COLOR ON ALL 4 PIXE1S
2903   95A7 D7 B5                   STB   ALLCOL                      SAVE BYTE WITH ALL PIXELS TURNED ON
2904   95A9 39                      RTS
2905   95AA 26 23           L95AA   BNE   L95CF                       BRANCH IF GRAPHIC MODE, OTHERWISE SET UP ALPHA GRAPHIC MODE
2906
2907                        * THIS CODE WILL RESET THE DISPLAY PAGE REGISTER IN THE
2908                        * SAM CHIP TO 2 ($4ØØ) AND RESET THE SAM S VDG CONTROL
2909                        * REGISTER TO Ø (ALPHA-NUMERICS). IN ADDITION, IT WILL
2910                        * RESET THE VDG CONTROL PINS TO ALPHA-GRAPHICS MODE.
2911
2912                        * SET UP THE SAM AND VDG TO GRAPHICS MODE
2913   95AC 34 16           L95AC   PSHS  X,B,A                       SAVE REGISTERS
2914   95AE 8E FF C8                LDX   #SAM+8                      POINT X TO THE MIDDLE OF THE SAM CNTL REG
2915   95B1 A7 ØA                   STA   1Ø,X                        **
2916   95B3 A7 Ø8                   STA   $Ø8,X                       ***
2917   95B5 A7 Ø6                   STA   $Ø6,X                       ****
2918   95B7 A7 Ø4                   STA   $Ø4,X                       ***** RESET SAM DISPLAY PAGE TO $4ØØ
2919   95B9 A7 Ø2                   STA   $Ø2,X                       ****
2920   95BB A7 Ø1                   STA   $Ø1,X                       ***
2921   95BD A7 1E                   STA   $-Ø2,X                      **
2922   95BF A7 1C                   STA   $-Ø4,X                      ***
2923   95C1 A7 1A                   STA   $-Ø6,X                      **** RESET SAM S VDG TO ALPHA-NUMERIC MODE
2924   95C3 A7 18                   STA   $-Ø8,X                      ***
2925   95C5 B6 FF 22                LDA   PIA1+2                      GET DATA FROM PIA1, PORT B
2926   95C8 84 Ø7                   ANDA  #$Ø7                        FORCE ALL BITS TO ZERO, KEEP ONLY CSS DATA
2927   95CA B7 FF 22                STA   PIA1+2                      PUT THE VDG INTO ALPHA-GRAPHICS MODE
2928   95CD 35 96                   PULS  A,B,X,PC                    RETURN
2929
2930   95CF 34 16           L95CF   PSHS  X,B,A
2931   95D1 96 B6                   LDA   PMODE                       GET CURRENT PMODE VALUE
2932   95D3 8B Ø3                   ADDA  #$Ø3                        ADD 3 - NOW 3-7 ONLY 5 OF 8 POSSIBLE MODES USED
2933   95D5 C6 1Ø                   LDB   #$1Ø                        $1Ø OFFSET BETWEEN PMODES
2934   95D7 3D                      MUL                               GET PMODE VALUES FOR VDG GMØ, GM1, GM2
2935   95D8 CA 8Ø                   ORB   #$8Ø                        FORCE BIT 7 HIGH (VDG A/G CONTROL)
2936   95DA DA C1                   ORB   CSSVAL                       OR  IN THE VDG CSS DATA
2937   95DC B6 FF 22                LDA   PIA1+2                      GET PIA1, PORT B
2938   95DF 84 Ø7                   ANDA  #$Ø7                        MASK OFF THE VDG CONTROL DATA
2939   95E1 34 Ø2                   PSHS  A                           SAVE IT
2940   95E3 EA E0                   ORB   ,S+                          OR  IT WITH THE VDG VALUES CALCULATED ABOVE
2941   95E5 F7 FF 22                STB   PIA1+2                      STORE IT INTO THE PIA
2942   95E8 96 BA                   LDA   BEGGRP                      GET MSB OF START OF GRAPHIC PAGE
2943   95EA 44                      LSRA                              *DIVIDE BY 2 - ACCA CONTAINS HOW MANY 512 BYTE
2944                        *                                         *BLOCKS IN STARTING ADDR
2945 > 95EB BD 96 ØF                JSR   L960F                       GO SET SAM CONTROL REGISTER
2946   95EE 96 B6                   LDA   PMODE                       GET PMODE VALUE
2947   95FØ 8B Ø3                   ADDA  #$Ø3                        ADD IN BIAS TO ADJUST TO PMODE THE SAM REGISTER WANTS
2948   95F2 81 Ø7                   CMPA  #$Ø7                        WAS PMODE 4?
2949   95F4 26 Ø1                   BNE   L95F7                       NO
2950   95F6 4A                      DECA                              DECREMENT ACCA IF PMODE 4 (SAME VDG AS PMODE3)
2951   95F7 8D Ø2           L95F7   BSR   L95FB                       SET THE SAM S VDG REGISTER
2952   95F9 35 96                   PULS  A,B,X,PC                    RESTORE REGISTERS AND RETURN
2953
2954   95FB C6 Ø3           L95FB   LDB   #$Ø3                        3 BITS IN SAM VDG CONTROL REGISTER
2955                        * ENTER WITH DATA TO GO IN VDG REGISTER IN BOTTOM 3 BITS OF ACCA
2956   95FD 8E FF CØ                LDX   #SAM                        POINT X TO SAM CONTROL REGISTER
2957   96ØØ 46              L96ØØ   RORA                              PUT A BIT INTO CARRY FLAG
2958   96Ø1 24 Ø4                   BCC   L96Ø7                       BRANCH IF BIT WAS A ZERO
2959   96Ø3 A7 Ø1                   STA   $Ø1,X                       SET SAM REGISTER BIT
2960   96Ø5 2Ø Ø2                   BRA   L96Ø9                       DO NEXT BIT
2961   96Ø7 A7 84           L96Ø7   STA   ,X                          CLEAR SAM REGISTER
2962   96Ø9 3Ø Ø2           L96Ø9   LEAX  $Ø2,X                       NEXT BIT IN REGISTER
2963   96ØB 5A                      DECB                              DONE ALL BITS?
2964   96ØC 26 F2                   BNE   L96ØØ                       NO
2965   96ØE 39                      RTS
2966   96ØF C6 Ø7           L96ØF   LDB   #$Ø7                        7 BITS IN SAM DISPLAY PAGE REGISTER
2967   9611 8E FF C6                LDX   #SAM+6                      POINT X TO SAM DISPLAY PAGE REGISTER
2968   9614 2Ø EA                   BRA   L96ØØ                       GO SET THE REGISTER
2969   9616 B6 FF 22        L9616   LDA   PIA1+2                      GET PIA1, PORT B
2970   9619 84 F7                   ANDA  #$F7                        MASK OFF VDG CSS CONTROL BIT
2971   961B 9A C1                   ORA   CSSVAL                      OR IN CSS COLOR DATA
2972   961D B7 FF 22                STA   PIA1+2                      RESTORE IT IN PIA1
2973   9620 39                      RTS
2974
2975                        * PMODE
2976   9621 81 2C           PMOD    CMPA  #','                        CHECK FOR COMMA - FIRST ARGUMENT MAY BE MISSING
```

```
2977   9623 27 2B                BEQ   L965Ø                     IT IS A COMMA
2978   9625 BD B7 ØB             JSR   EVALEXPB                  EVALUATE EXPRESSION
2979   9628 C1 Ø5                CMPB  #$Ø5                      > 4?
2980   962A 24 41                BCC   L966D                     YES, ILLEGAL FUNCTION CALL
2981   962C 96 BC                LDA   GRPRAM                    GET THE START OF GRAPHIC RAM
2982   962E 97 BA          L962E STA   BEGGRP                    SET START GRAPHIC PAGE
2983   9630 58                   ASLB                            MULT MODE BY 2 - TABLE HAS 2 BYTES PER ENTRY
2984   9631 CE 97 Ø7             LDU   #L97Ø6+1                  LOOKUP TABLE
2985   9634 AB C5                ADDA  B,U                       ADD THE AMOUNT OF MEMORY REQUIRED FOR ONE GRAPHIC PAGE
2986   9636 91 19                CMPA  TXTTAB                    COMPARE TO MSB OF START OF BASIC PROGRAM
2987   9638 22 33                BHI   L966D                      FC  ERROR IF END OF GRAPHIC PAGE > START OF BASIC PROGRAM
2988   963A 97 B7                STA   ENDGRP                    STORE THE END OF GRAPHIC PAGE
2989   963C 33 5F                LEAU  $-Ø1,U                    POINT U TO PREVIOUS BYTE IN TABLE
2990   963E A6 C5                LDA   B,U                       *GET THE NUMBER OF BYTES/HORIZONTAL LINE
2991   9640 97 B9                STA   HORBYT                    *AND SAVE IT IN HORBYT
2992   9642 54                   LSRB                            RESTORE PMODE VALUE
2993   9643 D7 B6                STB   PMODE                     SAVE IT
2994   9645 4F                   CLRA                            BACKGROUND COLOR
2995   9646 97 B3                STA   BAKCOL                    SET BACKGROUND COLOR TO ZERO
2996   9648 86 Ø3                LDA   #$Ø3                      FOREGROUND COLOR
2997   964A 97 B2                STA   FORCOL                    SET FOREGROUND COLOR
2998   964C 9D A5                JSR   GETCCH                    IS THERE A STARTING PAGE NUMBER?
2999   964E 27 1C                BEQ   L966C                     NO
3000   9650 BD B7 38       L965Ø JSR   LB738                     EVALUATE EXPRESSION
3001   9653 5D                   TSTB                            SET FLAGS
3002   9654 27 17                BEQ   L966D                      ILLEGAL FUNCTION CALL  - CAN T START ON PAGE ZERO
3003   9656 5A                   DECB                            BUMP ONE; BASIC STARTS ON PAGE 1, THIS ROUTINE AT Ø
3004   9657 86 Ø6                LDA   #$Ø6                      EACH GRAPHIC PAGE = 6 X 256 (1.5K)
3005   9659 3D                   MUL                             MULT BY PAGE NUMBER
3006   965A DB BC                ADDB  GRPRAM                    ADD IN START OF GRAPHIC RAM
3007   965C 34 Ø4                PSHS  B                         SAVE TEMP START ADDR
3008   965E DB B7                ADDB  ENDGRP                    ADD CURRENT END ADDR
3009   9660 DØ BA                SUBB  BEGGRP                    SUB OUT CURRENT START ADDR - (ADDS THE SIZE OF ONE GRAPHIC PAGE)
3010   9662 D1 19                CMPB  TXTTAB                    IS IT > CURRENT START OF BASIC PROGRAM
3011   9664 22 Ø7                BHI   L966D                     YES! ILLEGAL FUNCTION CALL
3012   9666 D7 B7                STB   ENDGRP                    SAVE AS END OF GRAPHIC PAGE
3013   9668 35 Ø4                PULS  B                         GET TEMP START ADOR
3014   966A D7 BA                STB   BEGGRP                    SAVE AS START OF GRAPHIC PAGE
3015   966C 39             L966C RTS
3016   966D 7E B4 4A       L966D JMP   LB44A                      ILLEGAL FUNCTION CALL'
3017
3018                       * SCREEN
3019   9670 81 2C          SCREEN CMPA #','                      CHECK FOR A COMMA
3020   9672 27 ØB                BEQ   L967F                     BRANCH IF COMMA - FIRST ARGUMENT MISSING
3021   9674 BD B7 ØB             JSR   EVALEXPB                  EVALUATE EXPRESSION
3022   9677 5D                   TSTB                            ZERO FLAG SET IF ALPHA, NOT SET IF GRAPHIC SCREEN
3023   9678 BD 95 AA             JSR   L95AA                     SET UP THE SAM & VDG FOR PROPER GRAPHIC MODE
3024   967B 9D A5                JSR   GETCCH                    GET NEXT CHARACTER
3025   967D 27 ED                BEQ   L966C                     RETURN IF NOTHING ELSE ON LINE
3026   967F BD B7 38       L967F JSR   LB738                     CHECK FOR COMMA AND EVALUATE EXPRESSION
3027   9682 5D                   TSTB                            SET FLAGS
3028   9683 27 Ø2                BEQ   L9687                     BRANCH IF COLOR SET ZERO
3029   9685 C6 Ø8                LDB   #$Ø8                      VALUE FOR COLOR SET ONE
3030   9687 D7 C1          L9687 STB   CSSVAL                    SAVE IN VDG CSS RAM IMAGE
3031   9689 2Ø 8B                BRA   L9616                     GO SET IT INTO PIA
3032
3033                       * PCLEAR
3034   968B BD B7 ØB       PCLEAR JSR  EVALEXPB                  EVALUATE EXPRESSION, RETURN VALUE IN ACCB
3035   968E 5D                   TSTB                            SET FLAGS
3036   968F 27 DC                BEQ   L966D                     BRANCH IF PCLEARØ -  FC  ERROR
3037   9691 C1 Ø9                CMPB  #$Ø9                      TRYING TO CLEAR MORE THAN 8 PAGES?
3038   9693 24 D8                BCC   L966D                     YES  ILLEGAL FUNCTION CALL
3039   9695 86 Ø6                LDA   #$Ø6                      6 X 256 (1.5K) PER GRAPHIC PAGE
3040   9697 3D                   MUL                             MULT BY NUMBER OF PAGES
3041   9698 DB BC                ADDB  GRPRAM                    ADD IN START OF GRAPHIC RAM
3042   969A 1F 98                TFR   B,A                       MOVE B TO MSB OF REG ACCD
3043   969C C6 Ø1                LDB   #$Ø1                      REG D NOW CONTAINS TOP OF PCLEARED SPACE +1
3044   969E 1F Ø2                TFR   D,Y                       SAVE IN Y
3045   96AØ 1Ø 93 B7             CMPD  ENDGRP                    COMPARE TOP OF PCLEARED SPACE TO END OF CURRENT GRAPHIC PAGE
3046                       * THIS CODE REFLECTS THE INFAMOUS PCLEAR BUG
3047   96A3 25 C8                BLO   L966D                      FC  ERROR IF TRYING TO CLEAR < END OF GRAPHIC RAM
3048   96A5 93 19                SUBD  TXTTAB                    SUBTRACT START OF BASIC PROGRAM
3049   96A7 D3 1B                ADDD  VARTAB                    ADD END OF BASIC PROGRAM
3050   96A9 1F Ø1                TFR   D,X                       X=TOP OF PCLEARED SPACE + LENGTH OF BASIC PROGRAM
3051   96AB 4C                   INCA                            ADD 256 - LEAVE SOME ROOM FOR THE STACK
3052   96AC 93 21                SUBD  FRETOP                    SUBTRACT OUT TOP OF CLEARED SPACE
3053   96AE 24 BD                BCC   L966D                      FC  ERROR - NO ROOM LEFT
3054   96BØ BD 8Ø DØ             JSR   L8ØDØ                     ADJUST BASIC S INPUT POINTER
3055   96B3 12                   NOP                             SPACE FILLER FOR EXBAS 1.1
3056   96B4 DE 1B                LDU   VARTAB                    GET END OF BASIC PROGRAM
3057   96B6 9F 1B                STX   VARTAB                    STORE NEW END OF BASIC PROGRAM
3058   96B8 11 93 1B             CMPU  VARTAB                    COMPARE OLD END TO NEW END
3059   96BB 24 17                BCC   L96D4                     BRANCH IF OLD END > NEW END
3060   96BD A6 C2          L96BD LDA   ,-U                       GET BYTE FROM OLD PROGRAM
3061   96BF A7 82                STA   ,-X                       MOVE TO NEW PROGRAM LOCATION
3062   96C1 11 93 19             CMPU  TXTTAB                    AT THE BEGINNING OF OLD PROGRAM?
3063   96C4 26 F7                BNE   L96BD                     NO
3064   96C6 1Ø 9F 19             STY   TXTTAB                    SAVE NEW STARTING ADDRESS
3065   96C9 6F 3F                CLR   $-Ø1,Y                    CLEAR BYTE JUST BEFORE PROGRAM
3066   96CB BD AC EF       L96CB JSR   LACEF                     PUT CORRECT ADDRESSES IN FIRST 2 BYTES OF EACH LINE
3067   96CE BD AD 26             JSR   LAD26                     DO PART OF A NEW
3068   96D1 7E AD 9E             JMP   LAD9E                     GO BACK TO BASIC S MAIN LOOP
3069   96D4 DE 19          L96D4 LDU   TXTTAB                    GET START OF BASIC PROGRAM
3070   96D6 1Ø 9F 19             STY   TXTTAB                    STORE NEW STARTING ADDR
3071   96D9 6F 3F                CLR   $-Ø1,Y                    CLEAR THE BYTE JUST BEFORE PROGRAM
3072   96DB A6 CØ          L96DB LDA   ,U+                       GET BYTE FROM OLD PROGRAM
```

```
3073   96DD A7 A0                    STA   ,Y+                          MOVE TO NEW PROG LOCATION
3074   96DF 10 9C 1B                 CMPY  VARTAB                       AT END OF OLD PROGRAM?
3075   96E2 26 F7                    BNE   L96DB                        NO
3076   96E4 20 E5                    BRA   L96CB                        GO RESET SOME POINTERS
3077
3078                        * INITIALIZATION ROUTINE FOR EXBAS GRAPHICS VARIABLES
3079   96E6 C6 1E             L96E6  LDB   #$1E                         *
3080   96E8 D7 19                    STB   TXTTAB                       * SET START OF BASIC PROG TO 1E00
3081   96EA 86 06                    LDA   #$06                         =
3082   96EC 97 BC             L96EC  STA   GRPRAM                       =CONSTANT OFFSET OF $600
3083   96EE 97 BA                    STA   BEGGRP                       START OF GRAPHICS PAGE TO $600
3084   96F0 4F                       CLRA                               PMODE = 0
3085   96F1 97 B6                    STA   PMODE                        SET PMODE TO 0
3086   96F3 86 10                    LDA   #$10                         16 BYTES/HOR GRAPHIC ROW
3087   96F5 97 B9                    STA   HORBYT                       SAVE IT
3088   96F7 86 03                    LDA   #$03                         SET FOREGROUND COLOR TO 3
3089   96F9 97 B2                    STA   FORCOL                       SET FOREGROUND COLOR TO 3
3090   96FB 86 0C                    LDA   #$0C                         *
3091   96FD 97 B7                    STA   ENDGRP                       * SET END OF GRAPHICS PAGE TO $C00
3092   96FF 9E 19                    LDX   TXTTAB                       GET START OF PROGRAM
3093   9701 6F 1F                    CLR   $-01,X                       CLEAR ONE BYTE JUST BEFORE PROGRAM
3094   9703 7E AD 19                 JMP   LAD19                        GO DO A  NEW
3095
3096                        * TABLE OF HOW MANY BYTES/GRAPHIC ROW AND HOW MUCH RAM
3097                        * FOR ONE HI RES SCREEN FOR THE PMODES. ROWS FIRST,
3098                        * BYTES (IN 256 BYTE BLOCKS) SECOND.
3099   9706 10 06             L9706  FCB   $10,$06                      PMODE 0
3100   9708 20 0C             L9708  FCB   $20,$0C                      PMODE 1
3101   970A 10 0C             L970A  FCB   $10,$0C                      PMODE 2
3102   970C 20 18             L970C  FCB   $20,$18                      PMODE 3
3103   970E 20 18             L970E  FCB   $20,$18                      PMODE 4
3104
3105                        * CALC ABS(VEREND - VERBEG)
3106   9710 DC C5             L9710  LDD   VEREND                        GET VERTICAL ENDING ADDRESS
3107   9712 93 BF                    SUBD  VERBEG                       SUBTRACT OUT VERTICAL BEGINNING ADDRESS
3108   9714 24 3B             L9714  BCC   L9751                         RETURN IF END >= START
3109   9716 34 01                    PSHS  CC                           SAVE STATUS (WHICH COORDINATE IS GREATER)
3110   9718 BD 9D C3                 JSR   L9DC3                        NEGATE ACCD IF START COORD > END COORD
3111   971B 35 81                    PULS  CC,PC                        RESTORE STATUS AND RETURN
3112                        * CALC ABS(HOREND - HORBEG)
3113   971D DC C3             L971D  LDD   HOREND                        GET HORIZONTAL END COORD
3114   971F 93 BD                    SUBD  HORBEG                       SUB OUT HORIZONTAL START COORD
3115   9721 20 F1                    BRA   L9714                        GET ABSOLUTE VALUE
3116
3117                        * PCOPY
3118   9723 8D 1A             PCOPY  BSR   L973F                         *EVALUATE SOURCE PAGE NUMBER AND RETURN MSB OF
3119                        *                                           *ADDRESS OF START OF PAGE IN ACCD
3120   9725 34 06                    PSHS  B,A                          SAVE PAGE 1 OFFSET
3121   9727 C6 A5                    LDB   #$A5                         TOKEN FOR  TO
3122   9729 BD B2 6F                 JSR   LB26F                        SYNTAX CHECK FOR  TO
3123   972C 8D 11                    BSR   L973F                        EVALUATE PAGE NUMBER
3124   972E 35 10                    PULS  X                            SET ADDRESS OF SOURCE PAGE TO X
3125   9730 1F 03                    TFR   D,U                          ADDRESS OF DESTINATION PAGE TO U
3126   9732 10 8E 03 00              LDY   #$300                        MOVE $300 PAIRS OF BYTES (ONE GRAPHIC PAGE)
3127   9736 EC 81             L9736  LDD   ,X++                         GET TWO BYTES FROM SOURCE
3128   9738 ED C1                    STD   ,U++                         PUT INTO DESTINATION PAGE
3129   973A 31 3F                    LEAY  $-01,Y                       DECREMENT COUNTER
3130   973C 26 F8                    BNE   L9736                        NOT DONE YET
3131   973E 39                       RTS
3132
3133   973F BD B7 0B          L973F  JSR   EVALEXPB                     EVALUATE EXPRESSION
3134   9742 5D                       TSTB                               PAGE ZERO?
3135   9743 27 0D                    BEQ   L9752                        YES -  ILLEGAL FUNCTION CALL
3136
3137                        * THIS IS A FLAKEY ERROR CHECK - IT WILL LET YOU PCOPY OVER
3138                        * THE TOP OF THE BASIC PROGRAM IN SOME INSTANCES.
3139   9745 D1 19                    CMPB  TXTTAB                       IS PAGE NUMBER > MSB OF START ADDR OF BASIC PROG?
3140   9747 22 09                    BHI   L9752                          FC  ERROR IF SO - BAD ERROR CHECK
3141   9749 5A                       DECB                               *BUMP PAGE NUMBER DOWN 1, BASIC STARTS AT 1, THIS
3142                        *                                           *ROUTINE STARTS AT ZERO
3143   974A 86 06                    LDA   #$06                         6*256 (1.5K) PER GRAPHIC PAGE
3144   974C 3D                       MUL                                GET OFFSET OF THIS PAGE NUMBER
3145   974D DB BC                    ADDB  GRPRAM                       *GET START OF GRAPHIC RAM- ACCB NOW CONTAINS
3146                        *                                           *MSB OF ADDRESS OF THIS PAGE
3147   974F 1E 89                    EXG   A,B                          NON ACCD HAS ADDRESS OF PAGE START
3148   9751 39                L9751  RTS
3149   9752 7E B4 4A          L9752  JMP   LB44A                         ILLEGAL FUNCTION CALL
3150
3151                        * GET
3152   9755 5F                GET    CLRB                                GET  FLAG
3153   9756 20 02                    BRA   L975A                        THIS SHOULD BE FCB SKP2 - IT WOULD SAVE A BYTE
3154
3155                        * PUT
3156   9758 C6 01             PUT    LDB   #$01                          PUT  FLAG
3157   975A D7 D8             L975A  STB   VD8                          SAVE GET/PUT FLAG
3158   975C BD 01 A0                 JSR   RVEC22                       HOOK INTO RAM
3159   975F 81 40                    CMPA  #'@'                         CHECK FOR @ SIGN
3160   9761 26 02                    BNE   L9765                        NO @ SIGN
3161   9763 9D 9F                    JSR   GETNCH                       GO GET NEXT INPUT CHARACTER
3162   9765 BD 93 8F          L9765  JSR   L938F                         GO EVALUATE START AND END POINTS - STORE START AT
3163                                                                     (HORBEG,VERSEG), END AT (HOREND,VEREND)
3164   9768 BD B2 6D                 JSR   SYNCOMMA                     SYNTAX CHECK FOR COMMA
3165   976B BD 98 CC                 JSR   L98CC                        GET POINTER TO ARRAY DESCRIPTOR
3166   976E 1F 10                    TFR   X,D                          SAVE DESCRIPTOR + 2 IN ACCD
3167   9770 EE 84                    LDU   ,X                           SAVE OFFSET TO NEXT ARRAY IN U
3168   9772 33 5E                    LEAU  $-02,U                       POINT U TO START OF DESCRIPTOR
```

```
3169   9774 33 CB                 LEAU  D,U                        POINT U TO END OF ARRAY
3170   9776 DF D1                 STU   VD1                        SAVE END OF DATA (END OF ARRAY)
3171   9778 30 02                 LEAX  $02,X                      POINT X TO NUMBER OF DIMENSIONS AND
3172   977A E6 84                 LDB   ,X                         GET NUMBER DIMENSIONS IN ACCB
3173   977C 58                    ASLB                             TIMES 2 - 2 BYTES/DIMENSION
3174   977D 3A                    ABX                              POINT X TO START OF ARRAY DATA
3175   977E 9F CF                 STX   VCF                        SAVE START OF DATA (START OF ARRAY DATA)
3176   9780 96 06                 LDA   VALTYP                     CHECK VARIABLE TYPE
3177   9782 26 CE                 BNE   L9752                       FC  ERROR IF STRING VARIABLE
3178   9784 0F D4                 CLR   VD4                        GET/PUT GRAPHIC/ACTION FLAG
3179   9786 9D A5                 JSR   GETCCH                     GET CURRENT INPUT CHAR
3180   9788 27 2D                 BEQ   L97B7                      BRANCH IF END OF LINE
3181   978A 03 D4                 COM   VD4                        TOGGLE GET/PUT GRAPHIC/ACTION FLAG
3182   978C BD B2 6D              JSR   SYNCOMMA                   SYNTAX CHECK FOR COMMA
3183   978F 0D D8                 TST   VD8                        CHECK GET/PUT FLAG
3184   9791 26 07                 BNE   L979A                      BRANCH IF PUT
3185   9793 C6 47                 LDB   #'G'                       CHECK FOR FULL GRAPHIC OPTION
3186   9795 BD B2 6F              JSR   LB26F                      DO A SYNTAX CHECK FOR A  G
3187   9798 20 30                 BRA   L97CA                      SKIP AROUND THE  NO G  OPTION CODE
3188   979A C6 05          L979A  LDB   #$05                      FIVE LEGAL TOKENS AT END OF PUT
3189   979C 8E 98 39              LDX   #L9839                     POINT X TO LOOK UP TABLE
3190   979F EE 81          L979F  LDU   ,X++                       GET  CLEAR BIT  ACTION ROUTINE ADDRESS
3191   97A1 10 AE 81              LDY   ,X++                       GET  SET BIT  ACTION ROUTINE ADDRESS
3192   97A4 A1 80                 CMPA  ,X+                        CHECK FOR ONE OF THE FIVE LEGAL TOKENS
3193   97A6 27 06                 BEQ   L97AE                      FOUND ONE
3194   97A8 5A                    DECB                             CHECKED ALL FIVE?
3195   97A9 26 F4                 BNE   L979F                      NO - KEEP GOING
3196   97AB 7E B2 77              JMP   LB277                       SYNTAX  ERROR IF TOKEN NOT FOUND
3197   97AE 10 9F D5       L97AE  STY   VD5                        ARRAY  SET BIT  ACTION ROUTINE ADDRESS
3198   97B1 DF D9                 STU   VD9                        ARRAY  CLEAR BIT  ACTION ROUTINE ADDRESS
3199   97B3 9D 9F                 JSR   GETNCH                     GET INPUT CHAR FROM BASIC
3200   97B5 20 13                 BRA   L97CA                      SKIP AROUND THE  NO G  OPTION CODE
3201
3202                       * NO  G  OPTION OR ACTION SPECIFIED BY BASIC INPUT LINE
3203   97B7 C6 F8          L97B7  LDB   #$F8                       BOTTOM 3 BITS MASK (8 PIXELS/BYTE)
3204   97B9 96 B6                 LDA   PMODE                      GET CURRENT PMODE
3205   97BB 46                    RORA                             BIT 0 TO CARRY
3206   97BC 24 02                 BCC   L97C0                      BRANCH IF PMODE 0,2,4 (2 COLOR)
3207   97BE C6 FC                 LDB   #$FC                       BOTTOM 2 BITS MASK (4 COLOR MODE - 4 PIXELS/BYTE)
3208   97C0 1F 98          L97C0  TFR   B,A                        SAVE ACCB IN ACCA
3209   97C2 D4 BE                 ANDB  HORBEG+1                   *
3210   97C4 D7 BE                 STB   HORBEG+1                   * MASK THE PIXEL COUNTER (BITS 0,1=
3211   97C6 94 C4                 ANDA  HOREND+1                   * 4 COLOR, BITS 0-2=2 COLOR) BITS OFF
3212   97C8 97 C4                 STA   HOREND+1                   * THE HORIZONTAL DIFFERENCE
3213   97CA BD 97 1D       L97CA  JSR   L971D                      CALC HORIZ DIFFERENCE ABS(HOREND-HORBEG)
3214   97CD 24 04                 BCC   L97D3                      BRANCH IF END > START
3215   97CF 9E C3                 LDX   HOREND                     * MAKE START = END IF
3216   97D1 9F BD                 STX   HORBEG                     * START > END
3217   97D3 DD C3          L97D3  STD   HOREND                     SAVE HORIZ DIFFERENCE
3218   97D5 BD 97 10              JSR   L9710                      CALC VERT DIFFERENCE ABS (VEREND-VERBEG)
3219   97D8 24 04                 BCC   L97DE                      BRANCH IF END > START
3220   97DA 9E C5                 LDX   VEREND                     * MAKE START = END IF
3221   97DC 9F BF                 STX   VERBEG                     * START > END
3222   97DE DD C5          L97DE  STD   VEREND                     SAVE VERT DIFFERENCE
3223   97E0 96 B6                 LDA   PMODE                      * GET PMODE BIT 0
3224   97E2 46                    RORA                             * TO THE CARRY FLAG
3225   97E3 DC C3                 LDD   HOREND                     GET HORIZ DIFFERENCE
3226   97E5 24 04                 BCC   L97EB                      BRANCH IF PMODE = 0,2,4(2 COLOR)
3227   97E7 D3 C3                 ADDD  HOREND                     * DOUBLE HORIZ DIFF - 2X AS MANY BYTES FOR
3228   97E9 DD C3                 STD   HOREND                     * NUMBER OF PIXELS IN PMODES 1,3
3229   97EB BD 94 20       L97EB  JSR   L9420                      NORMALIZE DIFFERENCES
3230   97EE DC C3                 LDD   HOREND                     GET HORIZ DIFFERENCE
3231   97F0 9E C5                 LDX   VEREND                     *
3232   97F2 30 01                 LEAX  $01,X                      *
3233   97F4 9F C5                 STX   VEREND                     * ADD 1 TO VERT DIFFERENCE
3234   97F6 0D D4                 TST   VD4                        = CHECK FOR  G  OPTION OR GET ACTION
3235   97F8 26 58                 BNE   L9852                      = AND BRANCH IF GIVEN
3236   97FA 44                    LSRA                             *
3237   97FB 56                    RORB                             *
3238   97FC 44                    LSRA                             *
3239   97FD 56                    RORB                             *
3240   97FE 44                    LSRA                             *
3241   97FF 56                    RORB                             * DIVIDE HORIZONTAL DIFFERENCE BY 8
3242   9800 C3 00 01              ADDD  #1                         ADD ONE TO QUOTIENT
3243   9803 DD C3                 STD   HOREND                     SAVE NEW HOR DIFFERENCE
3244   9805 BD 92 98              JSR   L9298                      *CONVERT (HORBEG,VERSEG) INTO ABSOLUTE SCREEN
3245                                                               *POS (X) AND PIXEL MASK (ACCA)
3246   9808 D6 C4          L9808  LDB   HOREND+1                   GET HORIZ DIFFERENCE
3247   980A 34 10                 PSHS  X                          SAVE SCREEN POSITION
3248   980C 0D D8          L980C  TST   VD8                        CHECK THE GET/PUT FLAG
3249   980E 27 21                 BEQ   L9831                      BRANCH IF  GET
3250   9810 8D 11                 BSR   L9823                      INCREMENT ARRAY DATA POINTER
3251   9812 A6 C4                 LDA   ,U                         GET DATA FROM ARRAY
3252   9814 A7 80                 STA   ,X+                        PUT IT ON THE SCREEN
3253   9816 5A             L9816  DECB                             DECREMENT HORIZ DIFFERENCE
3254   9817 26 F3                 BNE   L980C                      BRANCH IF NOT AT END OF HORIZ LINE
3255   9819 35 10                 PULS  X                          GET SCREEN POSITION BACK
3256   981B BD 92 E9              JSR   L92E9                      MOVE ABS POSITION DOWN ONE ROW
3257   981E 0A C6                 DEC   VEREND+1                   DECREMENT VERTICAL DIFFERENCE
3258   9820 26 E6                 BNE   L9808                      BRANCH IF NOT DONE
3259   9822 39             L9822  RTS
3260
3261   9823 DE CF          L9823  LDU   VCF                        *
3262   9825 33 41                 LEAU  $01,U                      *
3263   9827 DF CF                 STU   VCF                        * ADD ONE TO CURRENT ARRAY DATA POINTER
3264   9829 11 93 D1              CMPU  VD1                        COMPARE TO END OF DATA
```

```
3265   982C 26 F4                BNE   L9822                    RETURN IF NOT AT END
3266   982E 7E B4 4A       L982E JMP   LB44A                     ILLEGAL FUNCTION CALL
3267
3268   9831 A6 80          L9831 LDA   ,X+                       GET DATA FROM SCREEN
3269   9833 8D EE                BSR   L9823                     INCREMENT ARRAY DATA POINTER
3270   9835 A7 C4                STA   ,U                        STORE IN ARRAY
3271   9837 20 DD                BRA   L9816                     KEEP LOOPING TILL DONE
3272                            *
3273   9839 98 94 98 9B    L9839 FDB   L9894,L989B
3274   983D BD             L983D FCB   $BD                       TOKEN FOR PSET
3275   983E 98 9B 98 94    L983E FDB   L989B,L9894
3276   9842 BE             L9842 FCB   $BE                       TOKEN FOR PRESET
3277   9843 98 B1 98 9B    L9843 FDB   L98B1,L989B
3278   9847 B1             L9847 FCB   $B1                       TOKEN FOR OR
3279   9848 98 94 98 B1    L9848 FDB   L9894,L98B1
3280   984C B0             L984C FCB   $B0                       TOKEN FOR AND
3281   984D 98 A1 98 A1    L984D FDB   L98A1,L98A1
3282   9851 A8             L9851 FCB   $A8                       TOKEN FOR NOT
3283
3284                            * GET/PUT WITH 'G' OPTION SPECIFIED
3285   9852 C3 00 01       L9852 ADDD  #1                        ADD ONE TO HORIZ DIFFERENCE
3286   9855 DD C3                STD   HOREND                    AND SAVE IT
3287   9857 96 D8                LDA   VD8                       *CHECK GET/PUT FLAG AND
3288   9859 26 09                BNE   L9864                     *BRANCH IF PUT
3289   985B DE D1                LDU   VD1                       GET END OF ARRAYS
3290   985D A7 C2          L985D STA   ,-U                       *THIS CODE WILL
3291   985F 11 93 CF             CMPU  VCF                       *ZERO OUT THE ENTIRE
3292   9862 22 F9                BHI   L985D                     *'GET' ARRAY
3293   9864 BD 92 98       L9864 JSR   L9298                     =CONVERT (HORBEG,VERBEG) INTO ABSOLUTE SCREEN POSITION
3294                            *                               =(X) AND PIXEL MASK (ACCA)
3295   9867 D6 B6                LDB   PMODE                     GET CURRENT PMODE
3296   9869 56                   RORB                            BIT 0 TO CARRY
3297   986A 24 02                BCC   L986E                     BRANCH IF PMODE 0,2,4 (2 COLOR)
3298   986C 84 AA                ANDA  #$AA                      USE $AA AS THE PIXEL MASK IN 4 COLOR MODE
3299   986E C6 01          L986E LDB   #$01                      INITIALIZE SHIFT CTR
3300   9870 10 9E CF             LDY   VCF                       POINT Y TO ARRAY DATA
3301   9873 34 12          L9873 PSHS  X,A                       SAVE PIXEL MASK (ACCA) AND ABS SCRN POS (X) ON STACK
3302   9875 DE C3                LDU   HOREND                    GET THE HORIZONTAL DIFFERENCE
3303   9877 34 42          L9877 PSHS  U,A                       SAVE PIXEL MASK AND HORIZ DIFF
3304   9879 54                   LSRB                            SHIFT BIT CTR RIGHT
3305   987A 24 08                BCC   L9884                     BRANCH IF ALL 8 SHIFTS NOT DONE
3306   987C 56                   RORB                            SHIFT CARRY BACK INTO ACCB
3307   987D 31 21                LEAY  $01,Y                     INCREMENT ARRAY DATA POINTER
3308   987F 10 9C D1             CMPY  VD1                       COMPARE TO END OF ARRAY
3309   9882 27 AA                BEQ   L982E                      FC  ERROR IF AT END
3310   9884 0D D8          L9884 TST   VD8                       CHECK THE GET/PUT FLAG AND
3311   9886 27 1F                BEQ   L98A7                     BRANCH IF GET
3312   9888 E5 A4                BITB  ,Y                        TEST A BIT IN ARRAY DATA
3313   988A 27 04                BEQ   L9890                     BRANCH IF ZERO
3314   988C 6E 9F 00 D5          JMP   [VD5]                     JUMP TO ACTION ROUTINE FOR ARRAY BIT SET
3315   9890 6E 9F 00 D9    L9890 JMP   [VD9]                     JUMP TO ACTION ROUTINE FOR ARRAY BIT CLEAR
3316   9894 43             L9894 COMA                            *MASK SOURCE DATA
3317   9895 A4 84                ANDA  ,X                        *OFF OF SCREEN DATA
3318   9897 A7 84                STA   ,X                        SAVE TO SCREEN
3319   9899 20 16                BRA   L98B1
3320   989B AA 84          L989B ORA   ,X                         OR  SOURCE DATA WITH SCREEN
3321   989D A7 84                STA   ,X                        SAVE TO SCREEN
3322   989F 20 10                BRA   L98B1
3323   98A1 A8 84          L98A1 EORA  ,X                        INVERT THE PIXEL
3324   98A3 A7 84                STA   ,X                        SAVE TO SCREEN
3325   98A5 20 0A                BRA   L98B1
3326   98A7 A5 84          L98A7 BITA  ,X                        TEST THE PIXEL
3327   98A9 27 06                BEQ   L98B1                     BRANCH IF IT IS OFF
3328   98AB 1F 98                TFR   B,A                       PUT SHIFT CTR IN ACCA
3329   98AD AA A4                ORA   ,Y                        TURN ON PROPER BIT IN
3330   98AF A7 A4                STA   ,Y                        THE ARRAY DATA
3331   98B1 35 42          L98B1 PULS  A,U                       RESTORE PIXEL MASK AND HOR DIFF
3332   98B3 BD 92 ED             JSR   L92ED                     MOVE SCRN POS & PIXEL MASK ONE TO RIGHT (TWO COLOR MODE)
3333   98B6 33 5F                LEAU  $-01,U                    *
3334   98B8 11 93 8A             CMPU  ZERO                      * DECR HORIZ DIFFERENCE AND
3335   98BB 26 BA                BNE   L9877                     * BRANCH IF NOT ZERO
3336   98BD AE 61                LDX   $01,S                     GET ABS SCRN POS FROM STACK
3337   98BF 96 B9                LDA   HORBYT                    GET NUMBER BYTES/GRAPHIC ROW
3338   98C1 30 86                LEAX  A,X                       MOVE SCRN POS DOWN ONE ROW
3339   98C3 35 02                PULS  A                         PULL PIXEL MASK OFF THE STACK
3340   98C5 32 62                LEAS  $02,S                     GET X OFF THE STACK
3341   98C7 0A C6                DEC   VEREND+1                  DECR VERT ROW CTR
3342   98C9 26 A8                BNE   L9873                     BRANCH IF NOT DONE
3343   98CB 39                   RTS                             RETURN FROM GET/PUT COMMAND
3344
3345   98CC BD B3 57       L98CC JSR   LB357                     EVAL ALPHA EXPR, RETURN DESCRIPTOR PTR IN X
3346   98CF E6 82                LDB   ,-X                       *STRIP OFF THE VARIABLE
3347   98D1 A6 82                LDA   ,-X                       *NAME (2 ALPHA-NUMERIC CHARACTERS) AND
3348   98D3 1F 03                TFR   D,U                       *STORE THEM IN U
3349   98D5 9E 1D                LDX   ARYTAB                    GET START OF ARRAYS
3350   98D7 9C 1F          L98D7 CMPX  ARYEND                    COMPARE TO END OF ARRAYS
3351   98D9 10 27 1B 6D          LBEQ  LB44A                      FC  ERROR IF UNDEFINED ARRAY
3352   98DD 11 A3 84             CMPU  ,X                        COMPARE TARGET NAME TO ARRAY NAME
3353   98E0 27 06                BEQ   L98E8                     RETURN IF CORRECT ARRAY FOUND
3354   98E2 EC 02                LDD   $02,X                     * GET OFFSET TO NEXT ARRAY AND
3355   98E4 30 8B                LEAX  D,X                       * ADD TO POINTER
3356   98E6 20 EF                BRA   L98D7                     KEEP SEARCHING FOR MATCH
3357   98E8 30 02          L98E8 LEAX  $02,X                     MOVE POINTER TO OFFSET TO NEXT ARRAY
3358   98EA 39                   RTS                             WASTED BYTE
3359   98EB 39             L98EB RTS
3360
```

```
3361                          * PAINT
3362   98EC 81 40             PAINT   CMPA  #'@'                      CHECK FOR @ SIGN
3363   98EE 26 02                     BNE   L98F2                      SKIP IF NOT
3364   98F0 9D 9F                     JSR   GETNCH                     READ A CHARACTER FROM BASIC INPUT LINE
3365   98F2 BD 93 B2          L98F2   JSR   L93B2                      *SYNTAX CHECK FOR  ( , TWO EXPRESSION, AND  ) .
3366                          *                                        *SAVE HOR COORD IN HORSES, VER COORD IN VERSES
3367   98F5 BD 93 1D                  JSR   L931D                      NORMALIZE THE HOR, VER COORDINATES
3368   98F8 86 01                     LDA   #$01                       PSET VALUE
3369   98FA 97 C2                     STA   SETFLG                     SET PSET/PRESET FLAG TO PSET
3370   98FC BD 95 81                  JSR   L9581                      GET PAINT COLOR CODE & SET THE ACTIVE COLOR AND ALL PIXEL BYTES
3371   98FF DC B4                     LDD   WCOLOR                     GET THEM
3372   9901 34 06                     PSHS  B,A                        SAVE THEM ON STACK
3373   9903 9D A5                     JSR   GETCCH                     GET CURRENT CHARACTER FROM INPUT LINE
3374   9905 27 03                     BEQ   L990A                      BRANCH IF NONE LEFT - DEFAULT BORDER COLOR TO FOREGROUND,
3375                                                                   PAINT COLOR TO BACKGROUND
3376   9907 BD 95 81                  JSR   L9581                      EVALUATE THE BORDER COLOR
3377   990A 96 B5             L990A   LDA   ALLCOL                     GET BORDER COLOR ALL PIXEL BYTE
3378   990C 97 D8                     STA   VD8                        TEMP SAVE IT
3379   990E 35 06                     PULS  A,B                        GET PAINT ACTIVE COLORS BACK
3380   9910 DD B4                     STD   WCOLOR                     RESAVE IT
3381   9912 4F                        CLRA                             * STORE A BLOCK OF  PAINT  DATA ON THE STACK WHICH
3382   9913 34 56                     PSHS  U,X,B,A                    * WILL ACT AS AN END OF  PAINT  DATA FLAG.
3383                          *                                        *THE CLRA WILL CAUSE THE UP/DN FLAG TO BE ZERO WHICH IS
3384                          *                                        *USED AS A FLAG TO EXIT THE  PAINT  ROUTINE
3385   9915 BD 95 22                  JSR   L9522                      GET NORMALIZED MAX HOR/VER VALUES - RETURN RESULT IN VD3, VD5
3386   9918 BD 92 8F                  JSR   L928F                      POINT U TO THE ROUTINE WHICH WILL SELECT A PIXEL
3387                          *
3388                          * 'PAINT' THE FIRST HORIZONTAL LINE FROM THE START COORDINATES
3389   991B DF D9                     STU   VD9                        SAVE IT
3390   991D BD 99 DF                  JSR   L99DF                       PAINT  FROM CURRENT HOR COORD TO ZERO
3391   9920 27 0F                     BEQ   L9931                      BRANCH IF NO PAINTING DONE - HIT BORDER INSTANTLY
3392   9922 BD 99 CB                  JSR   L99CB                       PAINT  TOWARD MAX HOR COORD
3393   9925 86 01                     LDA   #$01                       *
3394   9927 97 D7                     STA   VD7                        * UP/DN FLAG UP=1; DOWN=$FF
3395   9929 BD 99 BA                  JSR   L99BA                      SAVE POSITIVE GOING LINE INFO ON STACK
3396   992C 00 D7                     NEG   VD7                        SET UP/ON FLAG = FF
3397   992E BD 99 BA                  JSR   L99BA                      SAVE NEGATIVE GOING LINE INFO ON STACK
3398   9931 10 DF DC          L9931   STS   TMPSTK                     TEMP STORE STACK POINTER
3399   9934 0D DB             L9934   TST   CHGFLG                     SEE IF  PAINTED  COLOR IS DIFFERENT THAN ORIGINAL COLOR
3400   9936 26 03                     BNE   L993B                      BRANCH IF DATA HAS BEEN MODIFIED
3401   9938 10 DE DC                  LDS   TMPSTK                     GET STACK POINTER BACK
3402   993B 35 56             L993B   PULS  A,B,X,U                    GET DATA FOR THE NEXT LINE SEGMENT TO CHECK FROM THE STACK
3403   993D 0F DB                     CLR   CHGFLG                     CLEAR CHANGE FLAG
3404   993F 10 DF DC                  STS   TMPSTK                     TEMP SAVE STACK
3405   9942 30 01                     LEAX  $01,X                      ADD ONE TO  START HOR COORD - 1
3406   9944 9F BD                     STX   HORBEG                     PUT IT AT  CURRENT HOR COORD ADDR
3407   9946 DF D1                     STU   VD1                         LENGTH OF PARENT LINE
3408   9948 97 D7                     STA   VD7                        SAVE  UP/DN FLAG
3409   994A 27 9F                     BEQ   L98EB                      EXIT ROUTINE IF UP/DN FLAG = 0
3410   994C 2B 06                     BMI   L9954                      BRANCH IF UP/DN FLAG = DOWN
3411                          * CHECK LINE BELOW CURRENT DATA
3412   994E 5C                        INCB                             INCREMENT VER COORD
3413   994F D1 D6                     CMPB  VD6                        COMPARE TO MAXIMUM VER COORD
3414   9951 23 05                     BLS   L9958                      BRANCH IF NOT GREATER - PROCESS LINE
3415   9953 5F                        CLRB                             SET VER COORD TO ZERO TO FORCE WRAP AROUND
3416   9954 5D                L9954   TSTB                             TEST VER COORD
3417   9955 27 DD                     BEQ   L9934                      *PROCESS ANOTHER BLOCK OF  PAINT  DATA IF
3418                          *                                        *WRAP AROUND - DISCARD ANY LINE BELOW
3419                          *                                        *VER COORD = 0 OR ABOVE MAX VERTICAL COORD
3420   9957 5A                        DECB                             DEC VER COORD - WE ARE TESTING UP/DN FLAG = UP IF THERE
3421                          * PROCESS A HOR LINE THAT WAS STORED ON STACK - LIMIT CHECKS HAVE BEEN DONE
3422   9958 D7 C0             L9958   STB   VERBEG+1                   SAVE CURRENT VER COORD
3423   995A BD 99 DF                  JSR   L99DF                      PAINT FROM HOR COORD TO ZERO OR BORDER
3424   995D 27 0F                     BEQ   L996E                      IF NUMBER OF PAINTED PIXELS = 0, COMPLEMENT LENGTH
3425   995F 10 83 00 03               CMPD  #3                         *SEE IF < 3 PIXELS WERE PAINTED - IF FEWER THAN
3426                          *                                        *THREE PIXELS WERE PAINTED THEN THERE IS NO NEED TO
3427                          *                                        *CHECK FOR MORE DATA TO  PAINT  ON THE LINE TO THE
3428                          *                                        *LEFT OF THE CURRENT POSITION IN THE OPPOSITE
3429                          *                                        *DIRECTION THAT THE UP/DN FLAG IS CURRENTLY SET TO
3430   9963 25 04                     BLO   L9969                      BRANCH IF NO NEED TO CHECK FOR  PAINTABLE  DATA
3431   9965 30 1E                     LEAX  $-02,X                     MOVE THE HORIZONTAL COORDINATE TWO PIXELS TO THE LEFT
3432   9967 8D 38                     BSR   L99A1                      *SAVE A BLOCK OF  PAINT  DATA IN THE DIRECTION
3433                          *                                        *OPPOSITE TO THE  UP/DN  FLAG
3434 > 9969 BD 99 CB          L9969   JSR   L99CB                      CONTINUE PAINTING LINE TO THE RIGHT
3435   996C 8D 4C             L996C   BSR   L99BA                      *SAVE A BLOCK OF PAINT DATA IN THE SAME
3436                          *                                        *DIRECTION AS THE  UP/DN  FLAG
3437                          *
3438                          * THIS CODE WILL INSURE THAT THE CURRENT LINE IS
3439                          * EXAMINED TO THE RIGHT FOR 'PAINTABLE' PIXELS FOR
3440                          * A LENGTH EQUAL TO THE LENGTH OF THE 'PARENT' LINE
3441   996E 43                L996E   COMA                             *
3442   996F 53                        COMB                             * COMPLEMENT LENGTH OF LINE JUST PAINTED
3443   9970 D3 D1             L9970   ADDD  VD1                        ADD TO LENGTH OF  PARENT LINE
3444   9972 DD D1                     STD   VD1                        SAVE DIFFERENCE OF LINE JUST PAINTED AND PARENT LINE
3445   9974 2F 16                     BLE   L998C                      BRANCH IF PARENT LINE IS SHORTER
3446   9976 BD 95 06                  JSR   L9506                      GO INC HOR COORD
3447   9979 BD 9A 12                  JSR   L9A12                      CHECK FOR BORDER COLOR
3448   997C 26 05                     BNE   L9983                      NOT BORDER COLOR -
3449   997E CC FF FF                  LDD   #-1                        * GO DECREMENT ONE FROM LENGTH OF DIFFERENCE
3450   9981 20 ED                     BRA   L9970                      * LINE AND KEEP LOOKING FOR NON BORDER COLOR
3451   9983 BD 95 14          L9983   JSR   L9514                      GO DEC HOR COORD
3452   9986 8D 3E                     BSR   L99C6                      GET AND SAVE HOR COORD
3453   9988 8D 5E                     BSR   L99E8                      PAINT FORWARD TO MAX HOR COORD OR BORDER
3454   998A 20 E0                     BRA   L996C                      SAVE A BLOCK OF  PAINT  DATA AND KEEP CHECKING
3455                          *
3456                          * CHECK TO SEE IF THE CURRENT LINE EXTENDS FURTHER TO
```

```
3457                          * THE RIGHT THAN THE PARENT LINE AND PUT A BLOCK OF
3458                          * PAINT DATA ON THE STACK IF IT IS MORE THAN 2 PIXELS
3459                          * PAST THE END OF THE PARENT LINE
3460   998C BD 95 Ø6          L998C    JSR   L95Ø6                      INC CURRENT HOR COORD
3461   998F 3Ø 8B                     LEAX  D,X                       * POINT X TO THE RIGHT END OF THE  PARENT  LINE
3462   9991 9F BD                     STX   HORBEG                    * AND SAVE IT AS THE CURRENT HORIZ COORDINATE
3463   9993 43                        COMA                            = ACCD CONTAINS A NEGATIVE NUMBER CORRESPONDING TO
3464   9994 53                        COMB                            = THE NUMBER OF PIXELS THE CURRENT LINE EXTENDS
3465   9995 83 ØØ Ø1                  SUBD  #1                        = PAST THE RIGHT END OF THE  PARENT  LINE. CONVERT
3466   9998 2F Ø4                     BLE   L999E                     = TO A POSITIVE NUMBER AND BRANCH IF THE LINE DOESN T EXTEND
3467   999A 1F Ø1                     TFR   D,X                       *SAVE THE PORTION OF THE LINE TO THE RIGHT OF THE PARENT LINE
3468                          *                                       *AS THE LENGTH
3469   999C 8D Ø3                     BSR   L99A1                     =SAVE A BLOCK OF  PAINT  DATA IN THE DIRECTION OPPOSITE THE
3470                          *                                       =CURRENT UP/DN FLAG
3471 > 999E 7E 99 34          L999E    JMP   L9934                     PROCESS MORE  PAINT  DATA BLOCKS
3472                          *
3473                          * BLOCKS OF PAINT DATA ARE STORED ON THE STACK SO THAT PAINT
3474                          * CAN  REMEMBER  WHERE IT SHOULD GO BACK TO PAINT UP OR DOWN
3475                          * FROM THE CURRENT LINE IT IS PAINTING. THESE BLOCKS OF DATA
3476                          * REPRESENT HORIZONTAL LINES ABOVE OR BELOW THE CURRENT LINE
3477                          * BEING PAINTED AND REQUIRE SIX BYTES OF STORAGE ON THE STACK.
3478                          * THE DATA ARE STORED AS FOLLOWS: ,S=UP/DOWN FLAG; 1,S=VER COORD
3479                          * OF LINE; 2 3,S=LEFTMOST HOR COORD OF LINE; 4 5,S=LENGTH OF LINE
3480
3481                          * SAVE A BLOCK OF  PAINT  DATA FOR A LINE IN THE
3482                          * OPPOSITE DIRECTION OF THE CURRENT UP/DN FLAG
3483   99A1 DD CB             L99A1    STD   VCB                       SAVE NUMBER PIXELS PAINTED
3484   99A3 35 2Ø                      PULS  Y                         PUT RETURN ADDR IN Y
3485   99A5 DC BD                      LDD   HORBEG                    GET HORIZONTAL START COORDINATE
3486   99A7 34 16                      PSHS  X,B,A                     PUT ON STACK
3487   99A9 96 D7                      LDA   VD7                       GET U/D FLAG
3488   99AB 4Ø                         NEGA                            REVERSE THE UP/DN FLAG
3489   99AC D6 CØ             L99AC    LDB   VERBEG+1                  GET VERTICAL START COORDINATE
3490   99AE 34 Ø6                      PSHS  B,A                       SAVE VERTICAL START COORDINATE AND U/D FLAG
3491   99BØ 34 2Ø                      PSHS  Y                         PUT RETURN ADDR BACK ON STACK
3492
3493                          * CODE BELOW CHECKS FOR ABILITY TO STORE FOUR BYTES IN
3494                          * FREE RAM, HOWEVER THE PAINT ROUTINE WILL STORE SIX
3495                          * BYTES IN FREE RAM - FIRST INSTRUCTION SHOULD BE LDB #3
3496   99B2 C6 Ø2                      LDB   #$Ø2                      * CHECK TO SEE IF THERE S ENOUGH FREE
3497   99B4 BD AC 33                   JSR   LAC33                     * RAM FOR 4 BYTES TEMP STORAGE
3498   99B7 DC CB                      LDD   VCB                       GET LENGTH OF RIGHT PAINTED LINE
3499   99B9 39                         RTS
3500                          *
3501                          * SAVE A BLOCK OF  PAINT  DATA FOR A LINE IN
3502                          * THE SAME DIRECTION AS THE CURRENT UP/DN FLAG
3503   99BA DD CB             L99BA    STD   VCB                       SAVE LENGTH OF RIGHT HOR PAINTED LINE
3504   99BC 35 2Ø                      PULS  Y                         SAVE RETURN ADDRESS IN Y
3505   99BE DC C3                      LDD   HOREND                    HORIZONTAL START COORDINATE
3506   99CØ 34 16                      PSHS  X,B,A                     SAVE HORIZONTAL START COORDINATE AND LENGTH
3507   99C2 96 D7                      LDA   VD7                       GET UP/DOWN FLAG (1 OR -1)
3508   99C4 2Ø E6                      BRA   L99AC                     SAVE THE  PAINT  DATA ON THE STACK
3509   99C6 9E BD             L99C6    LDX   HORBEG                    GET CURRENT HOR COORD
3510   99C8 9F C3                      STX   HOREND                    SAVE IT
3511   99CA 39                         RTS
3512                          * GO HERE TO FINISH PAINTING TO RIGHT AFTER YOU HAVE PAINTED TO THE LEFT
3513   99CB DD CD             L99CB    STD   VCD                       SAVE COUNT OF THE NUMBER OF PIXELS  PAINTED
3514   99CD 1Ø 9E C3                   LDY   HOREND                    GET LAST HOR START COORD
3515   99DØ 8D F4                      BSR   L99C6                     *SAVE CURRENT HOR COORD - NOW HOREND CONTAINS COORDINATE
3516                          *                                       *THE LEFT BORDER OF THIS HORIZONTAL LINE
3517   99D2 1Ø 9F BD                   STY   HORBEG                    START PAINTING TO RIGHT FROM THE LEFT PAINT START COORD
3518   99D5 8D 11                      BSR   L99E8                     PAINT TOWARDS THE RIGHT
3519   99D7 9E CD                      LDX   VCD                       GET THE NUMBER OF PIXELS PAINTED WHEN GOING TOWARDS LEFT PIXELS
3520   99D9 3Ø 8B                      LEAX  D,X                       ADD TO NUMBER PAINTED GOING TOWARD RIGHT
3521   99DB C3 ØØ Ø1                  ADDD  #1                        ADD 1 TO PAINT COUNT TOWARD RIGHT - ACCD = LENGTH OF PAINTED LINE
3522                          *
3523   99DE 39                         RTS
3524                          * PAINT FROM HOR COORD TO ZERO OR HIT BORDER
3525                          * RETURN WITH Z = 1 IF NO PAINTING DONE
3526 > 99DF BD 99 C6          L99DF    JSR   L99C6                     PUT STARTING HOR COORD IN HOREND
3527   99E2 1Ø 8E 95 14                LDY   #L9514                    (DECR HOR COORD ADDRESS) TO Y
3528   99E6 2Ø Ø6                      BRA   L99EE                     GO PAINT THE LINE
3529                          * PAINT FROM HOR COORD TO MAX HOR COORD OR HIT
3530                          * BORDER-RETURN Z=1 IF NO PAINTING DONE
3531   99E8 1Ø 8E 95 Ø6       L99E8    LDY   #L95Ø6                    PUT INCR HOR COORD ADDR IN Y
3532   99EC AD A4                      JSR   ,Y                        INCR HOR COORD - THE LEFT PAINT ROUTINE PAINTED THE FIRST COORD
3533   99EE DE 8A             L99EE    LDU   ZERO                      ZERO U REG - INITIAL PIXEL PAINT COUNTER
3534   99FØ 9E BD                      LDX   HORBEG                    GET HOR COORD
3535   99F2 2B 17             L99F2    BMI   L9AØB                     BRANCH IF HORIZONTAL COORDINATE IS > $7F OR < Ø
3536   99F4 9C D3                      CMPX  VD3                       COMPARE CURRENT HOR COORD TO MAX VALUE
3537   99F6 22 13                      BHI   L9AØB                     BRANCH IF > MAX
3538   99F8 34 6Ø                      PSHS  U,Y                       SAVE PAINT COUNTER, INC/DEC POINTER
3539   99FA 8D 16                      BSR   L9A12                     CHECK FOR BORDER PIXEL
3540   99FC 27 ØB                      BEQ   L9AØ9                     HIT BOROER
3541   99FE BD 93 77                   JSR   L9377                     SET PIXEL TO  PAINT  COLOR - PAINTING IS DONE HERE
3542   9A01 35 6Ø                      PULS  Y,U                       RESTORE PAINT COUNTER AND INC/DEC POINTER
3543   9A03 33 41                      LEAU  $Ø1,U                     ADD ONE TO PAINT COUNTER
3544   9A05 AD A4                      JSR   ,Y                        INCR OR DECR HOR COORD DEPENDING ON CONTENTS OF Y
3545   9A07 2Ø E9                      BRA   L99F2                     KEEP PAINTING THE LINE
3546   9A09 35 6Ø             L9AØ9    PULS  Y,U                       RESTORE PAINT COUNTER AND INC/DEC POINTER
3547   9A0B 1F 3Ø             L9AØB    TFR   U,D                       SAVE PAINT COUNTER IN ACCD
3548   9A0D 1F Ø1                      TFR   D,X                       SAVE PAINT COUNTER IN X
3549   9A0F 93 8A                      SUBD  ZERO                      SET FLAGS ACCDRDING TO CONDITION OF PAINT COUNTER
3550   9A11 39                         RTS
3551                          * CHECK FOR BORDER COLOR - ENTER W/VD9 CONTAINING
3552                          * ADDRESS OF ROUTINE TO GET ABS SCREEN ADDRESS
```

```
3553                        * AND PIXEL MASK - EXIT WITH Z = 1 IF HIT BORDER COLOR PIXEL
3554   9A12 AD 9F ØØ D9     L9A12   JSR   [VD9]                    GET THE ADDR AND PIXEL MASK
3555   9A16 1F 89                    TFR   A,B                      COPY PIXEL MASK TO ACCB
3556   9A18 D4 D8                    ANDB  VD8                      AND PIXEL MASK W/BORDER COLOR; ACCB = ONE PIXEL OF BORDER COLOR
3557   9A1A 34 Ø6                    PSHS  B,A                      PUSH MASK AND BORDER PIXEL
3558   9A1C A4 84                    ANDA  ,X                       * PUT CURRENT PIXEL DATA INTO ACCB AND
3559   9A1E A1 61                    CMPA  $Ø1,S                    * COMPARE IT TO BORDER COLOR; Z FLAG = 1 IF MATCH
3560   9A2Ø 35 86                    PULS  A,B,PC                   RESTORE MASK AND BORDER PIXEL - THEN RETURN
3561
3562                        * PLAY
3563   9A22 9E 8A            PLAY    LDX   ZERO                     *DEFAULT VALUES FOR LENGTH OF  PLAY  COMMAND AND ADDRESS
3564   9A24 C6 Ø1                    LDB   #$Ø1                     *OF START OF  PLAY  STRING IF USED FOR PLAY  (NULL STRING)
3565   9A26 34 14                    PSHS  X,B                      SAVE DEFAULT VALUES
3566   9A28 BD B1 56                 JSR   LB156                    EVALUATE EXPRESSION
3567   9A2B 5F                       CLRB                           *
3568   9A2C BD A9 A2                 JSR   LA9A2                    * SET UP DA TO PASS THROUGH ANA MUX
3569   9A2F BD A9 76                 JSR   LA976                    ENABLE ANA MUX
3570   9A32 BD B6 54         L9A32   JSR   LB654                    *POINT X TO START OF  PLAY  STRING AND PUT LENGTH
3571                        *                                      *OF STRING INTO ACCB
3572   9A35 2Ø Ø2                    BRA   L9A39                    INEFFICIENT - SHOULD BE FCB SKP2
3573   9A37 35 14            L9A37   PULS  B,X                      GET PLAY STRING START AND LENGTH
3574   9A39 D7 D8            L9A39   STB   VD8                      LENGTH OF  PLAY  COMMAND
3575   9A3B 27 FA                    BEQ   L9A37                    GET NEW STRING DATA IF LENGTH = Ø
3576   9A3D 9F D9                    STX   VD9                      START OF  PLAY  STRING
3577   9A3F 1Ø 27 ØF 31              LBEQ  LA974                    DISABLE ANA MUX AND RETURN IF X = Ø
3578   9A43 ØD D8            L9A43   TST   VD8                      SEE IF LENGTH OF STRING = Ø
3579   9A45 27 FØ                    BEQ   L9A37                    GET NEW DATA IF SO
3580   9A47 BD 9B 98                 JSR   L9B98                    GET A COMMAND CHARACTER IF NOT
3581   9A4A 81 3B                    CMPA  #';'                     SUB COMMAND TERMINATED
3582   9A4C 27 F5                    BEQ   L9A43                    IGNORE SEMICOLONS
3583   9A4E 81 27                    CMPA  #'''                     CHECK FOR APOSTROPHE
3584   9A5Ø 27 F1                    BEQ   L9A43                    IGNORE THEM TOO
3585   9A52 81 58                    CMPA  #'X'                     CHECK FOR AN EXECUTABLE SUBSTRING
3586   9A54 1Ø 27 Ø1 B2              LBEQ  L9CØA                    GO PROCESS SUB COMMAND
3587   9A58 8D Ø2                    BSR   L9A5C                    CHECK FOR OTHER COMMANDS
3588   9A5A 2Ø E7                    BRA   L9A43                    KEEP GOING THROUGH INTERPRETATION LOOP
3589                        * OCTAVE
3590   9A5C 81 4F            L9A5C   CMPA  #'O'                     ADJUST OCTAVE?
3591   9A5E 26 ØD                    BNE   L9A6D                    NO
3592   9A6Ø D6 DE                    LDB   OCTAVE                   GET CURRENT OCTAVE
3593   9A62 5C                       INCB                           LEGAL VALUES ARE 1-5 BUT INTERNALLY THE COMPUTER USES Ø-4
3594   9A63 8D 5B                    BSR   L9ACØ                    MODIFIER CHECK
3595   9A65 5A                       DECB                           COMPENSATE FOR INCB ABOVE
3596   9A66 C1 Ø4                    CMPB  #$Ø4                     MAXIMUM VALUE OF 4
3597   9A68 22 63                    BHI   L9ACD                     FC  ERROR
3598   9A6A D7 DE                    STB   OCTAVE                   SAVE NEW VALUE OF OCTAVE
3599   9A6C 39                       RTS
3600                        * VOLUME
3601   9A6D 81 56            L9A6D   CMPA  #'V'                     ADJUST VOLUME?
3602   9A6F 26 1A                    BNE   L9A8B                    NO
3603   9A71 D6 DF                    LDB   VOLHI                    GET CURRENT HIGH VOLUME LIMIT
3604   9A73 54                       LSRB                           *SHIFT 2 BITS TO RIGHT; DA IS ONLY 6 BITS (BIT 2 - BIT 7) -
3605   9A74 54                       LSRB                           *TO MANIPULATE THE DATA IT MUST BE IN BITS Ø-5
3606   9A75 CØ 1F                    SUBB  #31                      SUBTRACT OUT MID VALUE OFFSET
3607   9A77 8D 47                    BSR   L9ACØ                    MODIFIER CHECK
3608   9A79 C1 1F                    CMPB  #31                      MAXIMUM ALLOWED RANGE IS 31
3609   9A7B 22 5Ø                    BHI   L9ACD                     FC  ERROR
3610   9A7D 58                       ASLB                           *
3611   9A7E 58                       ASLB                           *MOVE NEW VALUE BACK TO BITS 2-7
3612   9A7F 34 Ø4                    PSHS  B                        SAVE NEW VOLUME ON THE STACK
3613   9A81 CC 7E 7E                 LDD   #$7E7E                   PUT MID VALUE IN HIGH AND LOW LIMIT
3614   9A84 AB E4                    ADDA  ,S                       ADD NEW VOLUME TO HIGH LIMIT
3615   9A86 E0 E0                    SUBB  ,S+                      SUBTR NEW VOLUME FROM LOW LIMIT
3616   9A88 DD DF                    STD   VOLHI                    SAVE NEW VOLUME LIMITS
3617   9A8A 39                       RTS
3618                        * NOTE LENGTH
3619   9A8B 81 4C            L9A8B   CMPA  #'L'                     SET NOTE LENGTH?
3620   9A8D 26 23                    BNE   L9AB2                    NO
3621   9A8F D6 E1                    LDB   NOTELN                   GET CURRENT LENGTH
3622   9A91 8D 2D                    BSR   L9AC0                    MODIFIER CHECK
3623   9A93 5D                       TSTB                           *
3624   9A94 27 37                    BEQ   L9ACD                    *  FC  ERROR IF LENGTH = Ø
3625   9A96 D7 E1                    STB   NOTELN                   SAVE NEW NOTE LENGTH
3626   9A98 0F E5                    CLR   DOTVAL                   RESET NOTE TIMER SCALE FACTOR
3627   9A9A 8D Ø3            L9A9A   BSR   L9A9F                    CHECK FOR A DOTTED NOTE
3628   9A9C 24 FC                    BCC   L9A9A                    BRANCH IF DOTTED NOTE
3629   9A9E 39                       RTS
3630                        * SCALE FACTOR -  DOTTED NOTE
3631   9A9F ØD D8            L9A9F   TST   VD8                      CHECK COMMAND LENGTH
3632   9AA1 27 ØA                    BEQ   L9AAD                    IT S EMPTY
3633   9AA3 BD 9B 98                 JSR   L9B98                    GET COMMAND CHARACTER
3634   9AA6 81 2E                    CMPA  #'.'                     CHECK FOR DOTTED NOTE
3635   9AA8 27 Ø5                    BEQ   L9AAF                    BRANCH ON DOTTED NOTE AND CLEAR CARRY FLAG
3636   9AAA BD 9B E2                 JSR   L9BE2                    *MOVE COMMAND STRING POINTER BACK ONE AND ADD ONE TO
3637                                                                *COMMAND LENGTH
3638   9AAD 43               L9AAD   COMA                           SET CARRY FLAG
3639   9AAE 39                       RTS
3640   9AAF ØC E5            L9AAF   INC   DOTVAL                   ADD ONE TO NOTE TIMER SCALE FACTOR
3641   9AB1 39                       RTS
3642                        * TEMPO
3643   9AB2 81 54            L9AB2   CMPA  #'T'                     MODIFY TEMPO?
3644   9AB4 26 ØD                    BNE   L9AC3                    NO
3645   9AB6 D6 E2                    LDB   TEMPO                    GET CURRENT TEMPO
3646   9AB8 8D Ø6                    BSR   L9AC0                    EVALUATE MODIFIER
3647   9ABA 5D                       TSTB                           SET FLAGS
3648   9ABB 27 1Ø                    BEQ   L9ACD                     FC  ERROR IF IT S Ø
```

```
3649   9ABD D7 E2                STB   TEMPO                          SAVE NEW TEMPO
3650   9ABF 39                   RTS
3651   9AC0 7E 9B AC     L9AC0   JMP   L9BAC                          EVALUATE THE >,<,+,-,= OPERATORS
3652                     * PAUSE
3653   9AC3 81 50        L9AC3   CMPA  #'P'                           PAUSE COMMAND?
3654   9AC5 26 24                BNE   L9AEB                          NO
3655   9AC7 BD 9C CB             JSR   L9CCB                          EVALUATE A DECIMAL COMMAND STRING VALUE
3656   9ACA 5D                   TSTB                                 * CHECK FOR LEGAL EXPRESSION AND
3657   9ACB 26 03                BNE   L9AD0                          * BRANCH IF PAUSE VALUE <> 0
3658   9ACD 7E B4 4A     L9ACD   JMP   LB44A                           FC  ERROR IF PAUSE <> 0
3659   9AD0 96 E5        L9AD0   LDA   DOTVAL                         *SAVE CURRENT VALUE OF VOLUME AND NOTE
3660   9AD2 9E DF                LDX   VOLHI                          *TIMER SCALE
3661   9AD4 34 12                PSHS  X,A                            *
3662   9AD6 86 7E                LDA   #$7E                           MID VALUE OF DA CONVERTER
3663   9AD8 97 DF                STA   VOLHI                          *SET VOLUME = 0
3664   9ADA 97 E0                STA   VOLLOW                         *
3665   9ADC 0F E5                CLR   DOTVAL                         RESET NOTE TIMER SCALE FACTOR
3666   9ADE 8D 07                BSR   L9AE7                          GO  PLAY  A NOTE OF 0 VOLUME
3667   9AE0 35 12                PULS  A,X                            *
3668   9AE2 97 E5                STA   DOTVAL                         *RESTORE VALUE OF VOLUME
3669   9AE4 9F DF                STX   VOLHI                          *AND NOTE TIMER SCALE
3670   9AE6 39                   RTS
3671   9AE7 6F E2        L9AE7   CLR   ,-S                            PUSH NOTE NUMBER 0 ONTO STACK
3672   9AE9 20 40                BRA   L9B2B                          GO PLAY IT
3673                     * NOTE
3674   9AEB 81 4E        L9AEB   CMPA  #'N'                           LETTER N BEFORE THE NUMBER OF A NOTE?
3675   9AED 26 03                BNE   L9AF2                          NO - IT S OPTIONAL
3676   9AEF BD 9B 98             JSR   L9B98                          GET NEXT COMMAND CHARACTER
3677   9AF2 81 41        L9AF2   CMPA  #'A'                           CHECK FOR NOTE A
3678   9AF4 25 04                BLO   L9AFA                          BELOW
3679   9AF6 81 47                CMPA  #'G'                           CHECK FOR NOTE B
3680   9AF8 23 05                BLS   L9AFF                          FOUND NOTE A-G
3681   9AFA BD 9B BE     L9AFA   JSR   L9BBE                          EVALUATE DECIMAL NUMERIC EXPRESSION IN COMMAND STRING
3682   9AFD 20 23                BRA   L9B22                          PROCESS NOTE VALUE
3683                     * PROCESS A NOTE HERE
3684   9AFF 80 41        L9AFF   SUBA  #'A'                           MASK OFF ASCII
3685   9B01 8E 9C 5B             LDX   #L9C5B                         LOAD X WITH NOTE JUMP TABLE
3686   9B04 E6 86                LDB   A,X                            GET NOTE
3687   9B06 0D D8                TST   VD8                            ANY COMMAND CHARACTERS LEFT?
3688   9B08 27 18                BEQ   L9B22                          NO
3689   9B0A BD 9B 98             JSR   L9B98                          GET COMMAND CHARACTER
3690   9B0D 81 23                CMPA  #'#'                            SHARP  NOTE?
3691   9B0F 27 04                BEQ   L9B15                          YES
3692   9B11 81 2B                CMPA  #'+'                            SHARP  NOTE?
3693   9B13 26 03                BNE   L9B18                          NO
3694   9B15 5C           L9B15   INCB                                 ADD 1 TO NOTE NUMBER (SHARP)
3695   9B16 20 0A                BRA   L9B22                          PROCESS NOTE
3696   9B18 81 2D        L9B18   CMPA  #'-'                            FLAT  NOTE?
3697   9B1A 26 03                BNE   L9B1F                          NO
3698   9B1C 5A                   DECB                                 SUBTR 1 FROM NOTE NUMBER (FLAT)
3699   9B1D 20 03                BRA   L9B22                          PROCESS NOTE
3700   9B1F BD 9B E2     L9B1F   JSR   L9BE2                          *MOVE COMMAND STRING PTR BACK ONE AND ADD ONE
3701                     *                                            *TO COMMAND LENGTH CTR
3702   9B22 5A           L9B22   DECB                                 =ADJUST NOTE NUMBER, BASIC USES NOTE NUMBERS 1-12, INTERNALLY
3703                     *                                            =COMPUTER USES 0-11
3704   9B23 C1 0B                CMPB  #12-1                          MAXIMUM NOTE VALUE
3705   9B25 22 A6                BHI   L9ACD                           FC  ERROR IF > 11
3706   9B27 34 04                PSHS  B                              SAVE NOTE VALUE
3707   9B29 D6 E1                LDB   NOTELN                         GET NOTE LENGTH
3708   9B2B 96 E2        L9B2B   LDA   TEMPO                          GET TEMPO
3709   9B2D 3D                   MUL                                  CALCULATE NOTE DURATION
3710   9B2E DD D5                STD   VD5                            SAVE NOTE DURATION
3711                     * THE IRQ INTERRUPT IS USED TO PROVIDE A MASTER TIMING REFERENCE FOR
3712                     * THE  PLAY  COMMAND.  WHEN A NOTE IS DONE, THE IRQ SERVICING
3713                     * ROUTINE WILL RETURN CONTROL TO THE MAIN  PLAY  COMMAND INTERPRETATION LOOP
3714   9B30 33 61                LEAU  $01,S                          *LOAD U W/CURRENT VALUE OF (STACK POINTER+1) SO THAT THE STACK
3715                     *                                            *POINTER WILL BE PROPERLY RESET WHEN IRQ VECTORS
3716                     *                                            *YOU OUT OF THE  PLAY  TIMING ROUTINES BELOW
3717   9B32 96 DE                LDA   OCTAVE                         GET CURRENT OCTAVE
3718   9B34 81 01                CMPA  #$01                           *
3719   9B36 22 2C                BHI   L9B64                          * BRANCH IF OCTAVE > 1
3720                     * OCTAVES 1 AND 2 USE A TWO BYTE DELAY TO SET THE PROPER FREQUENCY
3721   9B38 8E 9C 62             LDX   #L9C62                         POINT TO DELAY TABLE
3722   9B3B C6 18                LDB   #2*12                          24 BYTES DATA/OCTAVE
3723   9B3D 3D                   MUL                                  CALC OCTAVE TABLE OFFSET
3724   9B3E 3A                   ABX                                  POINT TO CORRECT OCTAVE TABLE
3725   9B3F 35 04                PULS  B                              GET NOTE VALUE BACK
3726   9B41 58                   ASLB                                 X 2 - 2 BYTES/NOTE
3727   9B42 3A                   ABX                                  POINT TO CORRECT NOTE
3728   9B43 31 84                LEAY  ,X                             GET POINTER TO Y REG (TFR X,Y)
3729   9B45 8D 45                BSR   L9B8C                          CALCULATE NOTE TIMER VALUE
3730   9B47 DD E3                STD   PLYTMR                         SAVE IT
3731                     * MAIN SOUND GENERATION LOOP - ONLY THE IRQ SERVICE WILL GET YOU OUT
3732                     * OF THIS LOOP (OCTAVES 1 AND 2)
3733   9B49 8D 0C        L9B49   BSR   L9B57                          MID VALUE TO DA AND WAIT
3734   9B4B 96 DF                LDA   VOLHI                          GET HIGH VALUE
3735   9B4D 8D 0B                BSR   L9B5A                          STORE TO DA AND WAIT
3736   9B4F 8D 06                BSR   L9B57                          MID VALUE TO DA AND WAIT
3737   9B51 96 E0                LDA   VOLLOW                         GET LOW VALUE
3738   9B53 8D 05                BSR   L9B5A                          STORE
3739   9B55 20 F2                BRA   L9B49                          KEEP LOOPING
3740   9B57 86 7E        L9B57   LDA   #$7E                           DA MID VALUE AND RS 232 MARKING
3741   9B59 12                   NOP                                  DELAY SOME - FINE TUNE PLAY FREQUENCY
3742   9B5A B7 FF 20     L9B5A   STA   PIA1                          STORE TO DA CONVERTER
3743   9B5D AE A4                LDX   ,Y                            GET DELAY FROM OCTAVE TABLE
3744   9B5F 30 1F        L9B5F   LEAX  $-01,X                        *
```

```
3745   9B61 26 FC                 BNE   L9B5F                       *COUNT X TO ZERO - PROGRAMMABLE DELAY
3746   9B63 39                    RTS
3747                        * OCTAVES 3,4 AND 5 USE A ONE BYTE DELAY TO SET THE PROPER FREQUENCY
3748   9B64 8E 9C 7A        L9B64 LDX   #L9C92-2*12                   POINT TO DELAY TABLE
3749   9B67 C6 ØC                 LDB   #12                           12 BYTES DATA PER OCTAVE
3750   9B69 3D                    MUL                                 CAIC OCTAVE TABLE OFFSET
3751   9B6A 3A                    ABX                                 POINT TO CORRECT OCTAVE TABLE
3752   9B6B 35 Ø4                 PULS  B                             GET NOTE VALUE BACK
3753   9B6D 3A                    ABX                                 POINT TO CORRECT NOTE
3754   9B6E 8D 1C                 BSR   L9B8C                         CALCULATE NOTE TIMER VALUE
3755   9B7Ø DD E3                 STD   PLYTMR                        SAVE IT
3756   9B72 8D ØC           L9B72 BSR   L9B8Ø                         MID VALUE TO DA AND WAIT
3757   9B74 96 DF                 LDA   VOLHI                         GET HIGH VALUE
3758   9B76 8D ØB                 BSR   L9B83                         STORE TO DA AND WAIT
3759   9B78 8D Ø6                 BSR   L9B8Ø                         MID VALUE TO DA AND WAIT
3760   9B7A 96 EØ                 LDA   VOLLOW                        GET LOW VALUE
3761   9B7C 8D Ø5                 BSR   L9B83                         STORE TO DA AND WAIT
3762   9B7E 2Ø F2                 BRA   L9B72                         KEEP GOING
3763                        * PUT MID VALUE TO DA CONVERTER AND WAIT A WHILE
3764   9B8Ø 86 7E           L9B8Ø LDA   #$7E                          DA CONVERTER MID VALUE AND KEEP RS 232 OUTPUT MARKING
3765   9B82 12                    NOP                                 DELAY SOME - FINE TUNE PLAY FREQUENCY
3766   9B83 B7 FF 2Ø        L9B83 STA   PIA1                          STORE IN DA CONVERTER
3767   9B86 A6 84                 LDA   ,X                            GET DELAY VALUE FROM OCTAVE TABLE
3768   9B88 4A              L9B88 DECA                                COUNT ACCA TO ZERO - TIME DELAY
3769   9B89 26 FD                 BNE   L9B88                         COUNT ACCA TO ZERO - TIME DELAY
3770   9B8B 39                    RTS
3771                        * CALCULATE NOTE TIMER VALUE - RETURN WITH VALUE IN ACCD -
3772                        * THE LARGER ACCD IS, THE LONGER THE NOTE WILL PLAY
3773   9B8C C6 FF           L9B8C LDB   #$FF                          NOTE TIMER BASE VALUE
3774   9B8E 96 E5                 LDA   DOTVAL                        GET NOTE TIMER SCALE FACTOR
3775   9B9Ø 27 Ø5                 BEQ   L9B97                         USE DEFAULT VALUE IF Ø
3776   9B92 8B Ø2                 ADDA  #$Ø2                          ADD IN CONSTANT TIMER SCALE FACTOR
3777   9B94 3D                    MUL                                 MULTIPLY SCALE FACTOR BY BASE VALUE
3778   9B95 44                    LSRA                                *DIVIDE ACCD BY TWO - EACH INCREMENT OF DOTVAL
3779   9B96 56                    RORB                                *WILL INCREASE NOTE TIMER BY 128
3780   9B97 39              L9B97 RTS
3781
3782                        * GET NEXT COMMAND - RETURN VALUE IN ACCA
3783   9B98 34 1Ø           L9B98 PSHS  X                             SAVE X REGISTER
3784   9B9A ØD D8           L9B9A TST   VD8                           CHECK COMMAND COUNTER
3785   9B9C 27 4D                 BEQ   L9BEB                          FC  ERROR IF NO COMMAND DATA LEFT
3786   9B9E 9E D9                 LDX   VD9                           GET COMMAND ADDR
3787   9BAØ A6 8Ø                 LDA   ,X+                           GET COMMAND
3788   9BA2 9F D9                 STX   VD9                           SAVE NEW ADDRESS
3789   9BA4 ØA D8                 DEC   VD8                           DECREMENT COMMAND CTR
3790   9BA6 81 2Ø                 CMPA  #SPACE                        CHECK FOR BLANK
3791   9BA8 27 FØ                 BEQ   L9B9A                         IGNORE BLANKS
3792   9BAA 35 9Ø                 PULS  X,PC                          RESTORE X RESISTER AND RETURN
3793
3794                        * EVALUATE THE >,<,+,-,= OPERATORS - ENTER WITH THE VALUE TO
3795                        * BE OPERATED ON IN ACCB, RETURN NEW VALUE IN SAME
3796   9BAC 8D EA           L9BAC BSR   L9B98                         GET A COMMAND CHARACTER
3797   9BAE 81 2B                 CMPA  #'+'                          ADD ONE?
3798   9BB Ø 27 3C                BEQ   L9BEE                         YES
3799   9BB2 81 2D                 CMPA  #'-'                          SUBTRACT ONE?
3800   9BB4 27 3C                 BEQ   L9BF2                         YES
3801   9BB6 81 3E                 CMPA  #'>'                          MULTIPLY BY TWO?
3802   9BB8 27 42                 BEQ   L9BFC                         YES
3803   9BBA 81 3C                 CMPA  #'<'                          DIVIDE BY TWO?
3804   9BBC 27 39                 BEQ   L9BF7                         YES
3805   9BBE 81 3D           L9BBE CMPA  #'='                          *CHECK FOR VARIABLE EQUATE - BRANCH IF SO; ACCB WILL BE
3806   9BC Ø 27 3F                BEQ   L9C01                         *SET TO THE VALUE OF THE BASIC VARIABLE IN THE COMMAND
3807                        *                                         *STRING WHICH MUST BE NUMERIC, LESS THAN 256
3808                        *                                         *AND THE VARIABLE MUST BE FOLLOWED BY A SEMICOLON.
3809   9BC2 BD 9Ø AA              JSR   L9ØAA                         CLEAR CARRY IF NUMERIC
3810   9BC5 25 24                 BLO   L9BEB                          FC  ERROR IF NON NUMERIC
3811   9BC7 5F                    CLRB                                UNITS DIGIT = Ø
3812                        * STRIP A DECIMAL ASCII VALUE OFF OF THE COMMAND STRING
3813                        * AND RETURN BINARY VALUE IN ACCB
3814   9BC8 8Ø 3Ø           L9BC8 SUBA  #'Ø'                          MASK OFF ASCII
3815   9BCA 97 D7                 STA   VD7                           SAVE VALUE TEMPORARILY
3816   9BCC 86 ØA                 LDA   #1Ø                           BASE 1Ø
3817   9BCE 3D                    MUL                                 MULT BY DIGIT
3818   9BCF 4D                    TSTA                                *
3819   9BDØ 26 19                 BNE   L9BEB                         * FC  ERROR IF RESULT > 255
3820   9BD2 DB D7                 ADDB  VD7                           GET TEMPORARY VALUE
3821   9BD4 25 15                 BLO   L9BEB                          FC  ERROR IF RESULT > 255
3822   9BD6 ØD D8                 TST   VD8                           *
3823   9BD8 27 17                 BEQ   L9BF1                         * RETURN IF NO COMMANDS LEFT
3824 > 9BDA BD 9B 98              JSR   L9B98                         GET ANOTHER COMMAND
3825   9BDD BD 9Ø AA              JSR   L9ØAA                         CLEAR CARRY IF NUMERIC
3826   9BE Ø 24 E6                BCC   L9BC8                         BRANCH IF MORE NUMERIC DATA
3827   9BE2 ØC D8           L9BE2 INC   VD8                           *ADD ONE TO COMMAND COUNTER AND
3828   9BE4 9E D9                 LDX   VD9                           *MOVE COMMAND STRING BACK ONE
3829   9BE6 3Ø 1F                 LEAX  $-Ø1,X                        *
3830   9BE8 9F D9                 STX   VD9                           *
3831   9BEA 39                    RTS
3832   9BEB 7E B4 4A        L9BEB JMP   LB44A                          FC  ERROR
3833   9BEE 5C              L9BEE INCB                                ADD ONE TO PARAMETER
3834   9BEF 27 FA                 BEQ   L9BEB                          FC  ERROR IF ADDING 1 TO 255
3835   9BF1 39              L9BF1 RTS
3836   9BF2 5D              L9BF2 TSTB                                *
3837   9BF3 27 F6                 BEQ   L9BEB                         * FC  ERROR IF TRYING TO DECREMENT Ø
3838   9BF5 5A                    DECB                                SUBTRACT ONE FROM PARAMETER
3839   9BF6 39                    RTS
3840   9BF7 5D              L9BF7 TSTB                                *
```

```
3841   9BF8 27 F1                    BEQ   L9BEB                       * FC  ERROR IF DIVIDING BY ZERO
3842   9BFA 54                       LSRB                              DIVIDE BY TWO
3843   9BFB 39                       RTS
3844   9BFC 5D              L9BFC    TSTB                              *
3845   9BFD 2B EC                    BMI   L9BEB                       * FC  ERROR IF RESULT WOULD BE > 255
3846   9BFF 58                       ASLB                              MULTIPLY BY TWO
3847   9C00 39                       RTS
3848   9C01 34 60           L9C01    PSHS  U,Y                         SAVE U,Y REGISTERS
3849   9C03 8D 16                    BSR   L9C1B                       INTERPRET COMMAND STRING AS IF IT WERE A BASIC VARIABLE
3850   9C05 BD B7 0E                 JSR   LB70E                       CONVERT FPA0 TO AN INTEGER VALUE IN ACCB
3851   9C08 35 E0                    PULS  Y,U,PC                      RESTORE U,Y REGISTERS AND RETURN
3852 > 9C0A BD 9C 1B        L9C0A    JSR   L9C1B                       EVALUATE AN EXPRESSION IN THE COMMAND STRING
3853   9C0D C6 02                    LDB   #2                          =
3854   9C0F BD AC 33                 JSR   LAC33                       =ROOM FOR 4 BYTES ON STACK?
3855   9C12 D6 D8                    LDB   VD8                         * GET THE CURRENT COMMAND LENGTH AND POINTER AND
3856   9C14 9E D9                    LDX   VD9                         * SAVE THEM ON THE STACK
3857   9C16 34 14                    PSHS  X,B                         *
3858   9C18 7E 9A 32                 JMP   L9A32                       GO INTERPRET AND PROCESS THE NEW  PLAY  SUB COMMAND
3859
3860                         * INTERPRET THE PRESENT COMMAND STRING AS IF IT WERE A BASIC VARIABLE
3861   9C1B 9E D9           L9C1B    LDX   VD9                         GET COMMAND POINTER
3862   9C1D 34 10                    PSHS  X                           SAVE IT
3863   9C1F BD 9B 98                 JSR   L9B98                       GET A COMMAND CHARACTER
3864   9C22 BD B3 A2                 JSR   LB3A2                       SET CARRY IF NOT ALPHA
3865   9C25 25 C4                    BLO   L9BEB                        FC  ERROR IF NOT ALPHA - ILLEGAL VARIABLE NAME
3866   9C27 BD 9B 98        L9C27    JSR   L9B98                       GET A COMMAND CHARACTER
3867   9C2A 81 3B                    CMPA  #';'                        CHECK FOR SEMICOLON - COMMAND SEPARATOR
3868   9C2C 26 F9                    BNE   L9C27                       BRANCH UNTIL FOUND
3869   9C2E 35 10                    PULS  X                           GET SAVED COMMAND POINTER
3870   9C30 DE A6                    LDU   CHARAD                      GET BASIC S INPUT POINTER
3871   9C32 34 40                    PSHS  U                           SAVE IT
3872   9C34 9F A6                    STX   CHARAD                      PUT PLAY COMMAND POINTER IN PLACE OF BASIC S INPUT POINTER
3873   9C36 BD B2 84                 JSR   LB284                       EVALUATE AN ALPHA EXPRESSION P GET NEW STRING DESCRIPTOR
3874   9C39 35 10                    PULS  X                           * RESTORE BASIC S INPUT POINTER
3875   9C3B 9F A6                    STX   CHARAD                      *
3876   9C3D 39                       RTS
3877
3878                         * MORE OF EXTENDED BASIC S IRQ ROUTINE
3879
3880   9C3E 4F              L9C3E    CLRA                              CLEAR ACCA
3881   9C3F 1F 8B                    TFR   A,DP                        SET THE DIRECT PAGE TO ZERO
3882   9C41 DC E3                    LDD   PLYTMR                      GET THE  PLAY  TIMER
3883   9C43 10 27 0D 74              LBEQ  LA9BB                       BRANCH TO COLOR BASIC S IRQ ROUTINE IF ZERO
3884   9C47 93 D5                    SUBD  VD5                         SUBTRACT OUT  PLAY  INTERVAL
3885   9C49 DD E3                    STD   PLYTMR                      SAVE THE NEW TIMER VALUE
3886   9C4B 22 0D                    BHI   L9C5A                       BRANCH IF  PLAY  COMMAND NOT DONE
3887   9C4D 0F E3                    CLR   PLYTMR                      RESET MSB OF  PLAY  TIMER IF DONE
3888   9C4F 0F E4                    CLR   PLYTMR+1                    RESET LSB OF  PLAY  TIMER
3889   9C51 35 02                    PULS  A                           GET THE CONDITION CODE REG
3890   9C53 10 EE 67                 LDS   $07,S                       *LOAD THE STACK POINTER WITH THE CONTENTS OF THE U REGISTER
3891                         *                                         *WHICH WAS STACKED WHEN THE INTERRUPT WAS HONORED.
3892   9C56 84 7F                    ANDA  #$7F                        CLEAR E FLAG - MAKE COMPUTER THINK THIS WAS AN FIRQ
3893   9C58 34 02                    PSHS  A                           SAVE CONDITION CODE
3894
3895                         *                                         THE RTI WILL NOW NOT RETURN TO WHERE IT WAS
3896                         *                                         INTERRUPTED FROM - IT WILL RETURN TO THE MAIN  PLAY
3897                         *                                         COMMAND INTERPRETATION LOOP.
3898
3899   9C5A 3B              L9C5A    RTI                               RETURN
3900
3901                         * TABLE OF NUMERICAL NOTE VALUES FOR LETTER NOTES
3902   9C5B 0A 0C 01 03 05 06 L9C5B   FCB   10,12,1,3,5,6,8             NOTES A,B,C,D,E,F,G
3903   9C61 08
3904
3905                         * TABLE OF DELAYS FOR OCTAVE 1
3906   9C62 01 A8 01 90 01 7A L9C62   FDB   $01A8,$0190,$017A           DELAYS FOR OCTAVE 1
3907   9C68 01 64 01 50 01 3D         FDB   $0164,$0150,$013D
3908   9C6E 01 2B 01 1A 01 0A         FDB   $012B,$011A,$010A
3909   9C74 00 FB 00 ED 00 DF         FDB   $00FB,$00ED,$00DF
3910
3911                         * TABLE OF DELAYS FOR OCTAVE 2
3912   9C7A 00 D3 00 C7 00 BB L9C7A   FDB   $00D3,$00C7,$00BB           DELAYS FOR OCTAVE 2
3913   9C80 00 B1 00 A6 00 9D         FDB   $00B1,$00A6,$009D
3914   9C86 00 94 00 8B 00 83         FDB   $0094,$008B,$0083
3915   9C8C 00 7C 00 75 00 6E         FDB   $007C,$0075,$006E
3916
3917                         * TABLE OF DELAYS FOR OCTAVES 3,4,5
3918   9C92 A6 9C 93 8B 83 7B L9C92   FCB   $A6,$9C,$93,$8B,$83,$7B     DELAYS FOR OCTAVES 3,4,5
3919   9C98 74 6D 67 61 5B 56         FCB   $74,$6D,$67,$61,$5B,$56
3920   9C9E 51 4C 47 43 3F 3B         FCB   $51,$4C,$47,$43,$3F,$3B
3921   9CA4 37 34 31 2E 2B 28         FCB   $37,$34,$31,$2E,$2B,$28
3922   9CAA 26 23 21 1F 1D 1B         FCB   $26,$23,$21,$1F,$1D,$1B
3923   9CB0 19 18 16 14 13 12         FCB   $19,$18,$16,$14,$13,$12
3924
3925
3926   9CB6 9E 8A           DRAW      LDX   ZERO                        * X=0. ACCB=1; END OF DRAW COMMAND LINE VALUES -
3927   9CB8 C6 01                     LDB   #$01                        * WHEN THESE VALUES ARE PULLED OFF THE
3928   9CBA 34 14                     PSHS  X,B                         * STACK, THE DRAW COMMAND WILL END
3929   9CBC D7 C2                     STB   SETFLG                      SET PSET/PRESET FLAG TO PSET
3930   9CBE 9F D5                     STX   VD5                         CLEAR UPDATE FLAG AND DRAW FLAG
3931   9CC0 BD 95 9A                  JSR   L959A                       SET ACTIVE COLOR BYTE
3932   9CC3 BD B1 56                  JSR   LB156                       EVALUATE EXPRESSION
3933   9CC6 BD B6 54        L9CC6     JSR   LB654                       GET THE LENGTH AND ADDRESS OF THE COMMAND STRING
3934   9CC9 20 08                     BRA   L9CD3                       INTERPRET THE COMMAND STRING
3935   9CCB BD 9B 98        L9CCB     JSR   L9B98                       GET NEXT CHARACTER FROM COMMAND LINE
3936   9CCE 7E 9B BE                  JMP   L9BBE                       EVALUATE A DECIMAL VALUE IN COMMAND LINE
```

```
3937   9CD1 35 14            L9CD1    PULS  B,X                 GET NEXT COMMAND LINE TO BE INTERPRETED OFF THE STACK
3938   9CD3 D7 D8            L9CD3    STB   VD8                 SET COMMAND LENGTH CTR
3939   9CD5 27 FA                     BEQ   L9CD1               GET NEW COMMAND LINE IF Ø
3940   9CD7 9F D9                     STX   VD9                 SET COMMAND LINE ADDRESS
3941   9CD9 10 27 00 EA               LBEQ  L9DC7               EXIT ROUTINE IF ADDRESS = Ø
3942   9CDD 0D D8            L9CDD    TST   VD8                 TEST COMMAND LENGTH CTR
3943   9CDF 27 F0                     BEQ   L9CD1               GET NEW LINE IF Ø
3944   9CE1 BD 9B 98                  JSR   L9B98               GET A COMMAND CHAR
3945   9CE4 81 3B                     CMPA  #';'                CHECK FOR SEMICOLON
3946   9CE6 27 F5                     BEQ   L9CDD               IGNORE SEMICOLONS
3947   9CE8 81 27                     CMPA  #''                 CHECK FOR APOSTROPHES
3948   9CEA 27 F1                     BEQ   L9CDD               IGNORE APOSTROPHES
3949   9CEC 81 4E                     CMPA  #'N'                UPDATE CHECK?
3950   9CEE 26 04                     BNE   L9CF4               NO
3951   9CF0 03 D5                     COM   VD5                 TOGGLE UPDATE FLAG Ø = UPDATE, FF = NO UPDATE
3952   9CF2 20 E9                     BRA   L9CDD               GET NEXT COMMAND
3953   9CF4 81 42            L9CF4    CMPA  #'B'                CHECK DRAW FLAG?
3954   9CF6 26 04                     BNE   L9CFC               NO
3955   9CF8 03 D6                     COM   VD6                 TOGGLE DRAW FLAG Ø = DRAW LINE, FF = DON T DRAW LINE
3956   9CFA 20 E1                     BRA   L9CDD               GET NEXT COMMAND
3957   9CFC 81 58            L9CFC    CMPA  #'X'                SUBSTRING?
3958   9CFE 10 27 00 96               LBEQ  L9D98               GO EXECUTE A COMMAND
3959   9D02 81 4D                     CMPA  #'M'                MOVE THE DRAW POSITION ?
3960   9D04 10 27 01 2A               LBEQ  L9E32               YES; GO MOVE IT
3961   9D08 34 02                     PSHS  A                   SAVE CURRENT COMMAND
3962   9D0A C6 01                     LDB   #$01                DEFAULT VALUE IF NO NUMBER FOLLOWS COMMAND
3963   9D0C 0D D8                     TST   VD8                 CHECK COMMAND LENGTH CTR
3964   9D0E 27 11                     BEQ   L9D21               BRANCH IF NO COMMANDS LEFT
3965   9D10 BD 9B 98                  JSR   L9B98               GET A COMMAND CHAR
3966   9D13 BD B3 A2                  JSR   LB3A2               SET CARRY IF NOT ALPHA
3967   9D16 34 01                     PSHS  CC                  SAVE CARRY FLAG
3968   9D18 BD 9B E2                  JSR   L9BE2               MOVE COMMAND POINTER BACK ONE
3969   9D1B 35 01                     PULS  CC                  RESTORE CARRY FLAG
3970   9D1D 24 02                     BCC   L9D21               BRANCH IF NEXT COMMAND IS ALPHA
3971   9D1F 8D AA                     BSR   L9CCB               EVALUATE A DECIMAL COMMAND LINE VALUE - RETURN VALUE IN ACCB
3972   9D21 35 02            L9D21    PULS  A                   GET CURRENT COMMAND BACK
3973   9D23 81 43                     CMPA  #'C'                CHANGE COLOR?
3974   9D25 27 28                     BEQ   L9D4F               YES
3975   9D27 81 41                     CMPA  #'A'                CHANGE ANGLE?
3976   9D29 27 2E                     BEQ   L9D59               YES
3977   9D2B 81 53                     CMPA  #'S'                CHANGE SCALE?
3978   9D2D 27 32                     BEQ   L9D61               YES
3979   9D2F 81 55                     CMPA  #'U'                GO UP?
3980   9D31 27 5C                     BEQ   L9D8F               YES
3981   9D33 81 44                     CMPA  #'D'                GO DOWN?
3982   9D35 27 55                     BEQ   L9D8C               YES
3983   9D37 81 4C                     CMPA  #'L'                GO LEFT?
3984   9D39 27 4C                     BEQ   L9D87               YES
3985   9D3B 81 52                     CMPA  #'R'                GO RIGHT?
3986   9D3D 27 43                     BEQ   L9D82               YES
3987   9D3F 80 45                     SUBA  #'E'                MASK OFF ASCII FOR LETTER E-H COMMAND CHECKS
3988   9D41 27 2F                     BEQ   L9D72               BRANCH IF E (45 DEGREES)
3989   9D43 4A                        DECA                      *
3990   9D44 27 27                     BEQ   L9D6D               *BRANCH IF F (135 DEGREES)
3991   9D46 4A                        DECA                      =
3992   9D47 27 32                     BEQ   L9D7B               =BRANCH IF G (225 DEGREES)
3993   9D49 4A                        DECA                      *
3994   9D4A 27 1D                     BEQ   L9D69               *BRANCH IF H (31S DEGREES)
3995   9D4C 7E B4 4A         L9D4C    JMP   LB44A                FC  ERROR IF ILLEGAL COMMAND
3996
3997                         * CHANGE COLOR
3998   9D4F BD 95 5D         L9D4F    JSR   L955D               ADJUST COLOR CODE FOR PROPER PMODE
3999   9D52 D7 B2                     STB   FORCOL              SAVE NEW FOREGROUND COLOR
4000   9D54 BD 95 9A                  JSR   L959A               SET COLOR BYTES (WCOLOR,ALLCOL)
4001   9D57 20 84            L9D57    BRA   L9CDD               GO PROCESS ANOTHER COMMAND
4002                         * CHANGE ANGLE
4003   9D59 C1 04            L9D59    CMPB  #$04                ONLY Ø-3 ARE LEGAL
4004   9D5B 24 EF                     BCC   L9D4C                FC  ERROR IF ANGLE NUMBER > 3
4005   9D5D D7 E8                     STB   ANGLE               SAVE DRAW ANGLE
4006   9D5F 20 F6                     BRA   L9D57               GO PROCESS ANOTHER COMMAND
4007                         * CHANGE SCALE
4008   9D61 C1 3F            L9D61    CMPB  #63                 ONLY Ø-63 ARE LEGAL
4009   9D63 24 E7                     BCC   L9D4C                FC  ERROR IF SCALE > 63
4010   9D65 D7 E9                     STB   SCALE               SAVE DRAW SCALE
4011   9D67 20 EE                     BRA   L9D57               GO PROCESS ANOTHER COMMAND
4012
4013                         * 315 DEGREES
4014   9D69 4F               L9D69    CLRA                      *NEGATE ACCD - MAKE HORIZONTAL
4015   9D6A 8D 58                     BSR   L9DC4               *DIFFERENCE NEGATIVE
4016   9D6C 21               L9D6C    FCB   SKP1                SKIP ONE BYTE - KEEP HORIZONTAL DIFFERENCE NEGATIVE
4017                         * 135 DEGREES
4018   9D6D 4F               L9D6D    CLRA                      CLEAR MS BYTE OF HORIZONTAL DIFFERENCE
4019   9D6E 1F 01                     TFR   D,X                 COPY HORIZONTAL DIFFERENCE TO VERTICAL DIFFERENCE
4020   9D70 20 59                     BRA   L9DCB               GO MOVE THE DRAW POSITION
4021                         * 45 DEGREES
4022   9D72 4F               L9D72    CLRA                      CLEAR MS BYTE OF HORIZONTAL DIFFERENCE
4023   9D73 1F 01                     TFR   D,X                 COPY HORIZONTAL DIFFERENCE TO VERTICAL DIFFERENCE
4024   9D75 8D 4D                     BSR   L9DC4               NEGATE ACCD - MAKE HORIZONTAL DIFFERENCE NEGATIVE
4025   9D77 1E 01                     EXG   D,X                 EXCHANGE HORIZONTAL AND VERTICAL DIFFERENCES
4026   9D79 20 50                     BRA   L9DCB               GO MOVE THE DRAW POSITION
4027                         * 225 DEGREES
4028   9D7B 4F               L9D7B    CLRA                      CLEAR MS BYTE OF HORIZONTAL DIFFERENCE
4029   9D7C 1F 01                     TFR   D,X                 COPY HORIZONTAL DIFFERENCE TO VERTICAL DIFFERENCE
4030   9D7E 8D 44                     BSR   L9DC4               NEGATE ACCD - MAKE HORIZONTAL DIFFERENCE NEGATIVE
4031   9D80 20 49                     BRA   L9DCB               GO MOVE THE DRAW POSITION
4032                         * GO RIGHT
```

```
4033   9D82 4F              L9D82   CLRA                            CLEAR MS BYTE OF HORIZONTAL DIFFERENCE
4034   9D83 9E 8A           L9D83   LDX   ZERO                      X = Ø; VERT DIFF = Ø
4035   9D85 2Ø 44                   BRA   L9DCB                     GO MOVE THE DRAW POSITION
4036                        * GO LEFT
4037   9D87 4F              L9D87   CLRA                            *NEGATE ACCD - MAKE THE HORIZONTAL
4038   9D88 8D 3A                   BSR   L9DC4                     *DIFFERENCE NEGATIVE
4039   9D8A 2Ø F7                   BRA   L9D83                     MAKE VERTICAL DIFFERENCE ZERO AND MOVE THE DRAW POSITION
4040                        * GO DOWN
4041   9D8C 4F              L9D8C   CLRA                            CLEAR MS BYTE OF HORIZONTAL DIFFERENCE
4042   9D8D 2Ø Ø3                   BRA   L9D92                     *MAKE VERTICAL DIFFERENCE = Ø, EXCHANGE HORIZONTAL AND
4043                        *                                       *VERTICAL DIFFERENCES AND MOVE THE DRAW POSTION
4044                        * GO UP
4045   9D8F 4F              L9D8F   CLRA                            *NEGATE ACCD - MAKE THE HORIZONTAL
4046   9D9Ø 8D 32                   BSR   L9DC4                     *DIFFERENCE NEGATIVE
4047   9D92 9E 8A           L9D92   LDX   ZERO                      X = Ø; HORZ DIFF = Ø
4048   9D94 1E 1Ø                   EXG   X,D                       EXCHANGE THE HORIZONTAL AND VERTICAL DIFFERENCES
4049   9D96 2Ø 33                   BRA   L9DCB                     GO MOVE THE DRAW POSITION
4050                        * EXECUTE A COMMAND SUB STRING
4051   9D98 BD 9C 1B        L9D98   JSR   L9C1B                     INTERPRET CURRENT COMMAND AS IF IT WERE A BASIC VARIABLE
4052   9D9B C6 Ø2                   LDB   #$Ø2                      =
4053   9D9D BD AC 33                JSR   LAC33                     =FOUR BYTES OF FREE RAM LEFT?
4054   9DAØ D6 D8                   LDB   VD8                       *
4055   9DA2 9E D9                   LDX   VD9                       * GET CURRENT COMMAND LENGTH AND POINTER
4056   9DA4 34 14                   PSHS  X,B                       * AND SAVE THEM ON THE STACK
4057   9DA6 7E 9C C6                JMP   L9CC6                     EVALUATE NUMERICAL VALUE IN COMMAND LINE
4058                        * MULTIPLY HOR OR VER DIFFERENCE BY SCALE FACTOR.
4059                        * DIVIDE PRODUCT BY 4 AND RETURN VALUE IN ACCD
4060   9DA9 D6 E9           L9DA9   LDB   SCALE                     GET DRAW SCALE AND BRANCH IF ZERO - THIS WILL CAUSE A
4061   9DAB 27 1B                   BEQ   L9DC8                     ZERO DEFAULT TO FULL SCALE
4062   9DAD 4F                      CLRA                            CLEAR MS BYTE
4063   9DAE 1E Ø1                   EXG   D,X                       EXCHANGE DIFFERENCE AND SCALE FACTOR
4064   9DBØ A7 E2                   STA   ,-S                       SAVE MS BYTE OF DIFFERENCE ON STACK (SIGN INFORMATION)
4065   9DB2 2A Ø2                   BPL   L9DB6                     BRANCH IF POSITIVE DIFFERENCE
4066   9DB4 8D ØD                   BSR   L9DC3                     NEGATE ACCD
4067   9DB6 BD 9F B5        L9DB6   JSR   L9FB5                     MULT DIFFERENCE BY SCALE FACTOR
4068   9DB9 1F 3Ø                   TFR   U,D                       SAVE 2 MS BYTES IN ACCD
4069   9DBB 44                      LSRA                            *
4070   9DBC 56                      RORB                            *
4071   9DBD 44              L9DBD   LSRA                            *
4072   9DBE 56                      RORB                            *DIVIDE ACCD BY 4 - EACH SCALE INCREMENT IS 1/4 FULL SCALE
4073   9DBF 6D EØ                   TST   ,S+                       =CHECK SIGN OF ORIGINAL DIFFERENCE AND
4074   9DC1 2A Ø4                   BPL   L9DC7                     =RETURN IF POSITIVE
4075                        * NEGATE ACCUMULATOR D
4076   9DC3 4Ø              L9DC3   NEGA
4077   9DC4 5Ø              L9DC4   NEGB                            *
4078   9DC5 82 ØØ                   SBCA  #$ØØ                      * NEGATE ACCUMULATOR D IF ACCA=Ø
4079   9DC7 39              L9DC7   RTS
4080   9DC8 1F 1Ø           L9DC8   TFR   X,D                       TRANSFER UNCHANGED DIFFERENCE TO ACCD
4081   9DCA 39                      RTS
4082                        * MOVE THE DRAW POSITION - ADD THE ORTHOGONAL DIFFERENCES
4083                        * IN ACCD (HORIZONTAL) AND X (VERTICAL) TO
4084                        * THE CURRENT POSITION; DRAW A LINE AFTER THE MOVE
4085   9DCB 34 Ø6           L9DCB   PSHS  B,A                       SAVE HORIZ DIFFERENCE
4086   9DCD 8D DA                   BSR   L9DA9                     APPLY SCALE FACTOR TO VERTICAL
4087   9DCF 35 1Ø                   PULS  X                         GET HORIZ DIFFERENCE
4088   9DD1 34 Ø6                   PSHS  B,A                       SAVE VERT DIFFERENCE
4089   9DD3 8D D4                   BSR   L9DA9                     APPLY SCALE FACTOR TO HORIZONTAL
4090   9DD5 35 1Ø                   PULS  X                         GET VERT DIFFERENCE
4091   9DD7 1Ø 9E E8                LDY   ANGLE                     * GET THE DRAW ANGLE AND SCALE AND SAVE THEM ON
4092   9DDA 34 2Ø                   PSHS  Y                         * THE STACK; USE Y BECAUSE IT IS THE ONLY UNUSED REGISTER
4093   9DDC 6D E4           L9DDC   TST   ,S                        CHECK DRAW ANGLE
4094   9DDE 27 Ø8                   BEQ   L9DE8                     BRANCH IF NO ANGLE
4095   9DEØ 1E 1Ø                   EXG   X,D                       EXCH HOR AND VER DIFFERENCES
4096   9DE2 8D DF                   BSR   L9DC3                     NEGATE ACCD
4097   9DE4 6A E4                   DEC   ,S                        DECR ANGLE
4098   9DE6 2Ø F4                   BRA   L9DDC                     CHECK ANGLE AGAIN
4099   9DE8 35 2Ø           L9DE8   PULS  Y                         PULL ANGLE AND SCALE OFF THE STACK
4100   9DEA DE 8A                   LDU   ZERO                      U = Ø; DEFAULT HOR END POSITION = Ø
4101   9DEC D3 C7                   ADDD  HORDEF                    ADD DIFFERENCE TO HORIZ START
4102   9DEE 2B Ø2                   BMI   L9DF2                     HORIZ COORD = Ø IF RESULT IS NEG
4103   9DFØ 1F Ø3                   TFR   D,U                       SAVE HOR END POSITION IN U
4104   9DF2 1F 1Ø           L9DF2   TFR   X,D                       PUT VERT DIFFERENCE IN ACCD
4105   9DF4 9E 8A                   LDX   ZERO                      X = Ø; DEFAULT VER END POSITION = Ø
4106   9DF6 D3 C9                   ADDD  VERDEF                    ADD DIFFERENCE TO VER START
4107   9DF8 2B Ø2                   BMI   L9DFC                     VER COORD = Ø IF RESULT IS NEG
4108   9DFA 1F Ø1                   TFR   D,X                       SAVE VERT END POSITION IN X
4109
4110                        * MOVE THE DRAW POSITION; ENTER WITH ABSOLUTE HORIZONTAL POSITION
4111                        * IN U REGISTER AND ABSOLUTE VERTICAL POSITION IN X REGISTER.
4112   9DFC 11 83 Ø1 ØØ     L9DFC   CMPU  #256                      IS HORIZ COORD WITHIN RANGE?
4113   9EØØ 25 Ø3                   BLO   L9E05                     YES
4114   9E02 CE ØØ FF                LDU   #255                      NO - FORCE TO MAX VALUE
4115   9E05 8C ØØ CØ        L9E05   CMPX  #192                      IS VERT COORD WITHIN RANGE?
4116   9E08 25 Ø3                   BLO   L9E0D                     YES
4117   9E0A 8E ØØ BF                LDX   #191                      NO - FORCE TO MAX VALUE
4118   9E0D DC C7           L9E0D   LDD   HORDEF                    *
4119   9E0F DD BD                   STD   HORBEG                    *
4120   9E11 DC C9                   LDD   VERDEF                    * COPY THE HOR AND VER POINTERS
4121   9E13 DD BF                   STD   VERBEG                    * INTO THE DRAW LINE START POSITION
4122   9E15 9F C5                   STX   VEREND                    =
4123   9E17 DF C3                   STU   HOREND                    = SET THE DRAW LINE END POSITION
4124   9E19 ØD D5                   TST   VD5                       CHECK UPDATE FLAG
4125   9E1B 26 Ø4                   BNE   L9E21                     BRANCH IF NO UPDATE
4126   9E1D 9F C9                   STX   VERDEF                    *
4127   9E1F DF C7                   STU   HORDEF                    * UPDATE POSITION OF DRAW POINTER
4128   9E21 BD 94 2Ø        L9E21   JSR   L9420                     NORMALIZE COORDS IN HOREND, VEREND AND HORBEG,VERBEG
```

```
4129   9E24 ØD D6                TST   VD6                       GET DRAW FLAG
4130   9E26 26 Ø3                BNE   L9E2B                     BRANCH IF NO DRAW
4131   9E28 BD 94 A1             JSR   L94A1                     DRAW A LINE FROM (HORBEG,VERBEG) TO (HOREND,VEREND)
4132   9E2B ØF D5       L9E2B    CLR   VD5                       RESET UPDATE FLAG
4133   9E2D ØF D6                CLR   VD6                       RESET DRAW FLAG
4134   9E2F 7E 9C DD             JMP   L9CDD                     GO GET ANOTHER COMMAND
4135
4136                     * SET THE DRAW POSITION
4137   9E32 BD 9B 98    L9E32    JSR   L9B98                     GET A CHAR FROM COMMAND LINE
4138   9E35 34 Ø2                PSHS  A                         SAVE CHARACTER
4139 > 9E37 BD 9E 5E             JSR   L9E5E                     EVALUATE HORIZ DIFFERENCE
4140   9E3A 34 Ø6                PSHS  B,A                       SAVE IT ON STACK
4141   9E3C BD 9B 98             JSR   L9B98                     GET A CHAR FROM COMMAND LINE
4142   9E3F 81 2C                CMPA  #','                      CHECK FOR COMMA
4143   9E41 1Ø 26 FF Ø7          LBNE  L9D4C                      FC  ERROR IF NO COMMA
4144 > 9E45 BD 9E 5B             JSR   L9E5B                     EVALUATE VERT DIFFERENCE
4145   9E48 1F Ø1                TFR   D,X                       SAVE VERT DIFFERENCE IN X
4146   9E4A 35 4Ø                PULS  U                         GET HORIZ DIFFERENCE IN U
4147   9E4C 35 Ø2                PULS  A                         GET FIRST COMMAND CHARACTER
4148   9E4E 81 2B                CMPA  #'+'                      *IF FIRST COMMAND CHAR WAS EITHER + OR -, TREAT
4149   9E5Ø 27 Ø4                BEQ   L9E56                     *THE VALUES IN U & X AS DIFFERENCES AND MOVE
4150   9E52 81 2D                CMPA  #'-'                      *POINTER, OTHERWISE TREAT U & X AS AN ABSOLUTE
4151   9E54 26 A6                BNE   L9DFC                     *POSITION AND MOVE THE CURRENT POSITION THERE.
4152   9E56 1F 3Ø       L9E56    TFR   U,D                       PUT HORIZ DIFFERENCE IN ACCD
4153   9E58 7E 9D CB             JMP   L9DCB                     GO MOVE THE DRAW POSITION
4154
4155   9E5B BD 9B 98    L9E5B    JSR   L9B98                     GET A CHAR FROM COMMAND LINE
4156   9E5E 81 2B       L9E5E    CMPA  #'+'                      *CHECK FOR A LEADING PLUS SIGN (RELATIVE MOTION)
4157   9E6Ø 27 Ø7                BEQ   L9E69                     *AND BRANCH IF RELATIVE
4158   9E62 81 2D                CMPA  #'-'                      =CHECK FOR A LEADING MINUS SIGN (RELATIVE MOTION)
4159   9E64 27 Ø4                BEQ   L9E6A                     =AND BRANCH IF RELATIVE
4160   9E66 BD 9B E2             JSR   L9BE2                     MOVE COMMAND STRING BACK ONE IF NOT RELATIVE MOTION
4161   9E69 4F          L9E69    CLRA                            ACCA = Ø IS  + ; ACCA <> Ø IS  -'
4162   9E6A 34 Ø2       L9E6A    PSHS  A                         SAVE ADD/SUB FLAG
4163   9E6C BD 9C CB             JSR   L9CCB                     EVALUATE DECIMAL NUMBER IN COMMAND STRING - RETURN VALUE IN ACCB
4164                     *
4165   9E6F 35 Ø2                PULS  A                         GET ADD/SUB FLAG
4166   9E71 4D                   TSTA                            CHECK IT, Ø:ADD, <> Ø:SUB
4167   9E72 27 Ø4                BEQ   L9E78                     RETURN IF ADD
4168   9E74 4F                   CLRA                            *
4169   9E75 5Ø                   NEGB                            *
4170   9E76 82 ØØ                SBCA  #$ØØ                      *NEGATE ACCB INTO A TWO BYTE SIGNED VALUE IN ACCD
4171   9E78 39          L9E78    RTS
4172                     *
4173                     * TABLE OF SINES AND COSINES FOR CIRCLE
4174                     *
4175   9E79 ØØ ØØ ØØ Ø1 L9E79    FDB   $ØØØØ,$ØØØ1               SUBARC Ø
4176   9E7D FE C5 19 19 L9E7D    FDB   $FEC5,$1919               SUBARC 1
4177   9E81 FB 16 31 F2 L9E81    FDB   $FB16,$31F2               SUBARC 2
4178   9E85 F4 FB 4A 51 L9E85    FDB   $F4FB,$4A51               SUBARC 3
4179   9E89 EC 84 61 F9 L9E89    FDB   $EC84,$61F9               SUBARC 4
4180   9E8D E1 C7 78 AE L9E8D    FDB   $E1C7,$78AE               SUBARC 5
4181   9E91 D4 DC 8E 3B L9E91    FDB   $D4DC,$8E3B               SUBARC 6
4182   9E95 C5 E5 A2 69 L9E95    FDB   $C5E5,$A269               SUBARC 7
4183   9E99 B5 Ø6 B5 Ø6 L9E99    FDB   $B5Ø6,$B5Ø6               SUBARC 8
4184                     *
4185                     * CIRCLE
4186                     * THE CIRCLE IS ACTUALLY DRAWN AS A 64 SIDED
4187                     * POLYGON.  IT IS COMPOSED OF 64  LINE  COMMANDS
4188   9E9D 81 4Ø       CIRCLE   CMPA  #'@'                      CHECK FOR @ SIGN
4189   9E9F 26 Ø2                BNE   L9EA3                     SKIP IF NOT
4190   9EA1 9D 9F                JSR   GETNCH                    GET ANOTHER CHARACTER FROM BASIC
4191   9EA3 BD 95 22    L9EA3    JSR   L9522                     GET MAX HOR & VER COORD VALUES AND PUT THEM IN VD3 AND VD5
4192   9EA6 BD 93 B2             JSR   L93B2                     GET HOR & VER CENTER COORDS AND PUT IN HORBEG,VERBEG
4193   9EA9 BD 93 1D             JSR   L931D                     NORMALIZE START COORDS FOR PROPER PMODE
4194   9EAC AE C4                LDX   ,U                        GET HOR COORD
4195   9EAE 9F CB                STX   VCB                       SAVE IT
4196   9EBØ AE 42                LDX   $Ø2,U                     GET VERT COORD
4197   9EB2 9F CD                STX   VCD                       SAVE IT
4198   9EB4 BD B2 6D             JSR   SYNCOMMA                  SYNTAX CHECK FOR COMMA
4199   9EB7 BD B7 3D             JSR   LB73D                     EVALUATE EXPRESSION RETURN VALUE IN X
4200   9EBA CE ØØ CF             LDU   #VCF                      POINT U TO TEMP DATA STORAGE
4201   9EBD AF C4                STX   ,U                        SAVE RADIUS
4202   9EBF BD 93 2Ø             JSR   L932Ø                     NORMALIZE RADIUS
4203   9EC2 86 Ø1                LDA   #$Ø1                      SET TO PSET
4204   9EC4 97 C2                STA   SETFLG                    SAVE PSET/PRESET FLAG
4205   9EC6 BD 95 81             JSR   L9581                     GO EVALUATE COLOR EXPRESSION AND SAVE IN WCOLOR
4206   9EC9 8E Ø1 ØØ             LDX   #$1ØØ                     HEIGHT/WIDTH RATIO DEFAULT VALUE
4207   9ECC 9D A5                JSR   GETCCH                    GET AN INPUT CHARACTER FROM BASIC
4208   9ECE 27 ØF                BEQ   L9EDF                     BRANCH IF NONE
4209   9ED0 BD B2 6D             JSR   SYNCOMMA                  SYNTAX CHECK FOR COMMA
4210   9ED3 BD B1 41             JSR   LB141                     EVALUATE EXPRESSION
4211   9ED6 96 4F                LDA   FPØEXP                    *GET FPAØ EXPONENT, ADD 8 TO IT AND RESAVE IT - THIS
4212   9ED8 8B Ø8                ADDA  #$Ø8                      *WILL EFFECTIVELY MULTIPLY FPAØ BY 256.
4213   9EDA 97 4F                STA   FPØEXP                    *
4214   9EDC BD B7 4Ø             JSR   LB74Ø                     EVALUATE EXPRESSION, RETURN VALUE IN X
4215   9EDF 96 B6       L9EDF    LDA   PMODE                     GET CURRENT PMODE VALUE
4216   9EE1 85 Ø2                BITA  #$Ø2                      TEST FOR PMODE Ø,1,4
4217   9EE3 27 Ø4                BEQ   L9EE9                     BRANCH IF SO
4218   9EE5 1F 1Ø                TFR   X,D                       * MULT X BY 2 -FOR PMODES 2,3 THE HOR PIXELS ARE 2X AS LONG AS
4219   9EE7 3Ø 8B                LEAX  D,X                       * PMODES Ø,1,4; MULT HW RATIO BY 2 TO COMPENSATE
4220   9EE9 9F D1       L9EE9    STX   VD1                       SAVE HW RATIO
4221   9EEB C6 Ø1                LDB   #$Ø1                      *
4222   9EED D7 C2                STB   SETFLG                    *SET PSET/PRESET FLAG TO PSET
4223   9EEF D7 D8                STB   VD8                       FIRST TIME FLAG - SET TO Ø AFTER ARC DRAWN
4224   9EF1 BD 9F E2             JSR   L9FE2                     EVALUATE CIRCLE START POINT (OCTANT, SUBARC)
```

```
4225   9EF4 34 Ø6                    PSHS  B,A                        SAVE START POINT
4226   9EF6 BD 9F E2                 JSR   L9FE2                      EVALUATE CIRCLE END POINT (OCTANT, SUBARC)
4227   9EF9 DD D9                    STD   VD9                        SAVE END POINT
4228   9EFB 35 Ø6                    PULS  A,B                        GET START POINT
4229   9EFD 34 Ø6             L9EFD  PSHS  B,A                        STORE CURRENT CIRCLE POSITION
4230   9EFF 9E C3                    LDX   HOREND                     * MOVE HOR, VER COORDS FROM HOREND,VEREND TO
4231   9FØ1 9F BD                    STX   HORBEG                     * HORBEG, VERBEG R MOVE OLD END COORDINATES
4232   9FØ3 9E C5                    LDX   VEREND                     * NEW START COORDINATES
4233   9FØ5 9F BF                    STX   VERBEG                     *
4234   9FØ7 CE 9E 7B                 LDU   #L9E79+2                   POINT TO TABLE OF SINES & COSINES
4235   9FØA 84 Ø1                    ANDA  #$Ø1                       =GET OCTANT NUMBER
4236   9FØC 27 Ø3                    BEQ   L9F11                      =BRANCH IF EVEN
4237   9FØE 5Ø                       NEGB                             *
4238   9FØF CB Ø8                    ADDB  #$Ø8                       *CONVERT Ø-7 TO 8-1 FOR ODD OCTANT NUMBERS
4239   9F11 58                L9F11  ASLB                             =
4240   9F12 58                       ASLB                             =FOUR BYTES/TABLE ENTRY
4241   9F13 33 C5                    LEAU  B,U                        POINT U TO CORRECT TABLE ENTRY
4242   9F15 34 4Ø                    PSHS  U                          SAVE SIN/COS TABLE ENTRY
4243   9F17 BD 9F A7                 JSR   L9FA7                      CALCULATE HORIZ OFFSET
4244   9F1A 35 4Ø                    PULS  U                          GET SIN/COS TABLE PTR
4245   9F1C 33 5E                    LEAU  $-Ø2,U                     MOVE TO COSINE (VERT)
4246   9F1E 34 1Ø                    PSHS  X                          SAVE HORIZ OFFSET
4247   9F2Ø BD 9F A7                 JSR   L9FA7                      CALCULATE VERT OFFSET
4248   9F23 35 2Ø                    PULS  Y                          PUT HORIZ OFFSET IN Y
4249   9F25 A6 E4                    LDA   ,S                         *
4250   9F27 84 Ø3                    ANDA  #$Ø3                       *
4251   9F29 27 Ø6                    BEQ   L9F31                      *BRANCH IF OCTANT Ø,3,4,7
4252   9F2B 81 Ø3                    CMPA  #$Ø3                       *
4253   9F2D 27 Ø2                    BEQ   L9F31                      *BRANCH IF OCTANT Ø,3,4,7
4254   9F2F 1E 12                    EXG   X,Y                        SWAP HOR AND VERT OFFSETS
4255   9F31 9F C3             L9F31  STX   HOREND                     SAVE HORIZ OFFSET
4256                         * THE HW RATIO WILL ONLY MODIFY THE VERT COORD
4257   9F33 1F 21                    TFR   Y,X                        LOAD X WITH THE CALCULATED VERT OFFSET
4258   9F35 DC D1                    LDD   VD1                        GET HW RATIO
4259 > 9F37 BD 9F B5                 JSR   L9FB5                      MULT VERT OFFSET BY HW RATIO
4260   9F3A 1F 2Ø                    TFR   Y,D                        TRANSFER THE PRODUCT TO ACCD
4261   9F3C 4D                       TSTA                             CHECK OVERFLOW FLAG AND GET MSB RESULT
4262   9F3D 1Ø 26 15 Ø9              LBNE  LB44A                       FC  ERROR IF RESULT > 255
4263   9F41 D7 C5                    STB   VEREND                     SAVE DELTA VER MBS
4264   9F43 1F 3Ø                    TFR   U,D                        LSB RESULT TO ACCA
4265   9F45 97 C6                    STA   VEREND+1                   SAVE DELTA VER LSB
4266   9F47 A6 E4                    LDA   ,S                         *
4267   9F49 81 Ø2                    CMPA  #$Ø2                       * BRANCH IF OCTANT = Ø,1,6,7 (SUBARC HOR END
4268   9F4B 25 ØE                    BLO   L9F5B                      * POINT >= HOR CENTER)
4269   9F4D 81 Ø6                    CMPA  #$Ø6                       = BRANCH IF OCTANT = Ø,1,6,7 (SUSARC HOR END
4270   9F4F 24 ØA                    BCC   L9F5B                      = POINT >= HOR CENTER)
4271   9F51 DC CB                    LDD   VCB                        GET HOR COORD OF CENTER
4272   9F53 93 C3                    SUBD  HOREND                     SUBTRACT HORIZONTAL DIFFERENCE
4273   9F55 24 11                    BCC   L9F68                      BRANCH IF NO UNDERFLOW
4274   9F57 4F                       CLRA                             *
4275   9F58 5F                       CLRB                             * IF NEW HOR < Ø, FORCE IT TO BE Ø
4276   9F59 2Ø ØD                    BRA   L9F68                      SAVE NEW COORD
4277   9F5B DC CB             L9F5B  LDD   VCB                        GET HOR COORD OF CENTER
4278   9F5D D3 C3                    ADDD  HOREND                     ADD HORIZONTAL DIFFERENCE
4279   9F5F 25 Ø5                    BLO   L9F66                      BRANCH IF OVERFLOW
4280   9F61 1Ø 93 D3                 CMPD  VD3                        COMPARE TO MAX HOR COORD
4281   9F64 25 Ø2                    BLO   L9F68                      BRANCH IF < MAX HOR
4282   9F66 DC D3             L9F66  LDD   VD3                        GET MAX HOR COORD
4283   9F68 DD C3             L9F68  STD   HOREND                     SAVE NEW HORIZ SU8ARC END COORD
4284   9F6A A6 E4                    LDA   ,S                         *
4285   9F6C 81 Ø4                    CMPA  #$Ø4                       * BRANCH IF OCTANT = Ø,1,2,3 (SUBARC VERT END
4286   9F6E 25 ØA                    BLO   L9F7A                      * POINT >= VERT CENTER)
4287   9F7Ø DC CD                    LDD   VCD                        GET VER COORD OF CENTER
4288   9F72 93 C5                    SUBD  VEREND                     SUBTRACT VERTICAL DIFFERENCE
4289   9F74 24 11                    BCC   L9F87                      BRANCH IF NO UNDERFIOW
4290   9F76 4F                       CLRA                             *
4291   9F77 5F                       CLRB                             *IF NEW VERT < Ø, FORCE IT TO BE Ø
4292   9F78 2Ø ØD                    BRA   L9F87                      SAVE NEW COORD
4293   9F7A DC CD             L9F7A  LDD   VCD                        GET VER COORD OF CENTER
4294   9F7C D3 C5                    ADDD  VEREND                     ADD VERTICAL DIFFERENCE
4295   9F7E 25 Ø5                    BLO   L9F85                      BRANCH IF OVERFLOW
4296   9F8Ø 1Ø 93 D5                 CMPD  VD5                        COMPARE TO MAX VERT COORD
4297   9F83 25 Ø2                    BLO   L9F87                      BRANCH IF < MAX VER
4298   9F85 DC D5             L9F85  LDD   VD5                        GET MAX VERT COORD
4299   9F87 DD C5             L9F87  STD   VEREND                     SAVE NEW VERT SUSARC END COORD
4300   9F89 ØD D8                    TST   VD8                        CHECK FIRST TIME FLAG
4301   9F8B 26 Ø2                    BNE   L9F8F                      *DO NOT DRAW A LINE FIRST TIME THRU -
4302                         *                                        *BECAUSE THE FIRST TIME YOU WOULD DRAW A LINE
4303                         *                                        *FROM THE CENTER TO THE FIRST POINT ON THE CIRCLE
4304   9F8D 8D 5Ø                    BSR   L9FDF                      DRAW A LINE
4305   9F8F 35 Ø6             L9F8F  PULS  A,B                        GET END COORDS
4306   9F91 Ø4 D8                    LSR   VD8                        SHIFT FIRST TIME FLAG
4307   9F93 25 Ø5                    BLO   L9F9A                      DO NOT CHECK FOR END POINT AFTER DRAWING FIRST ARC
4308   9F95 1Ø 93 D9                 CMPD  VD9                        COMPARE CURRENT POSITION TO END POINT
4309   9F98 27 ØC                    BEQ   L9FA6                      CIRCLE DRAWING FINISHED
4310                         * INCREMENT SUBARC CTR, IF > 7 THEN INCR OCTANT CTR
4311   9F9A 5C                L9F9A  INCB                             INCR SUBARC CTR
4312   9F9B C1 Ø8                    CMPB  #$Ø8                       > 7?
4313   9F9D 26 Ø4                    BNE   L9FA3                      NO
4314   9F9F 4C                       INCA                             INCR OCTANT CTR
4315   9FAØ 5F                       CLRB                             RESET SUBARC CTR
4316   9FA1 84 Ø7                    ANDA  #$Ø7                       *KEEP IN RANGE OF Ø-7; ONCE ACCA = B, THIS WILL MAKE ACCA = Ø,
4317                         *                                        *SO THE END POINT WILL BE (Ø,Ø) AND THE CIRCLE ROUTINE WILL END.
4318   9FA3 7E 9E FD          L9FA3  JMP   L9EFD                      KEEP DRAWING CIRCLE
4319   9FA6 39                L9FA6  RTS                              EXIT CIRCLE ROUTINE
4320                         * MULTIPLY RADIUS BY SIN/COS VALUE AND RETURN OFFSET IN X
```

```
4321   9FA7 9E CF          L9FA7   LDX   VCF                       GET RADIUS
4322   9FA9 EC C4                  LDD   ,U                        GET SIN/COS TABLE MODIFIER
4323   9FAB 27 07                  BEQ   L9FB4                     BRANCH IF = 0 - OFFSET = RADIUS
4324   9FAD 83 00 01               SUBD  #1                        SUBTR 1
4325   9FB0 8D 03                  BSR   L9FB5                     MULT RADIUS BY SIN/COS
4326   9FB2 1F 21                  TFR   Y,X                       RETURN RESULT IN X REG
4327   9FB4 39             L9FB4   RTS
4328                       * MULTIPLY (UNSIGNED) TWO 16 BIT NUMBERS TOGETHER -
4329                       * ENTER WITH ONE NUMBER IN ACCD, THE OTHER IN X
4330                       * REG. THE 4 BYTE PRODUCT WILL BE STORED IN 4,S-7,S
4331                       * (Y, U REG ON THE STACK). I.E. (AA AB) X (XH XL) =
4332                       * 256*AA*XH+16*(AA*XL+AB*XH)+AB*XL. THE 2 BYTE
4333                       * MULTIPLIER AND MULTIPLICAND ARE TREATED AS A 1
4334                       * BYTE INTEGER PART (MSB) WITH A 1 BYTE FRACTIONAL PART (LSB)
4335   9FB5 34 76          L9FB5   PSHS  U,Y,X,B,A                 SAVE REGISTERS AND RESERVE STORAGE SPACE ON THE STACK
4336   9FB7 6F 64                  CLR   $04,S                     RESET OVERFLOW FLAG
4337   9FB9 A6 63                  LDA   $03,S                     =
4338   9FBB 3D                     MUL                             =
4339   9FBC ED 66                  STD   $06,S                     = CALCULATE ACCB*XL, STORE RESULT IN 6,S
4340   9FBE EC 61                  LDD   $01,S                     *
4341   9FC0 3D                     MUL                             * CALCULATE ACCB*XH
4342   9FC1 EB 66                  ADDB  $06,S                     =
4343   9FC3 89 00                  ADCA  #$00                      =
4344   9FC5 ED 65                  STD   $05,S                     = ADD THE CARRY FROM THE 1ST MUL TO THE RESULT OF THE 2ND MUL
4345   9FC7 E6 E4                  LDB   ,S                        *
4346   9FC9 A6 63                  LDA   $03,S                     *
4347   9FCB 3D                     MUL                             * CALCULATE ACCA*XL
4348   9FCC E3 65                  ADDD  $05,S                     =
4349   9FCE ED 65                  STD   $05,S                     = ADD RESULT TO TOTAL OF 2 PREVIOUS MULTS
4350   9FD0 24 02                  BCC   L9FD4                     BRANCH IF NO OVERFLOW
4351   9FD2 6C 64                  INC   $04,S                     SET OVERFLOW FLAG (ACCD > $FFFF)
4352   9FD4 A6 E4          L9FD4   LDA   ,S                        *
4353   9FD6 E6 62                  LDB   $02,S                     *
4354   9FD8 3D                     MUL                             * CALCULATE ACCA*XH
4355   9FD9 E3 64                  ADDD  $04,S                     =
4356   9FDB ED 64                  STD   $04,S                     = ADD TO PREVIOUS RESULT
4357   9FDD 35 F6                  PULS  A,B,X,Y,U,PC              RETURN RESULT IN U,Y
4358   9FDF 7E 94 A1       L9FDF   JMP   L94A1                     GO DRAW A LINE FROM (HORBEG,VERBEG) TO (HOREND,VEREND)
4359
4360                       * CALCULATE START OR END POINT WHICH IS A NUMBER FROM
4361                       * 0 TO 63 SAVED AS AN OCTANT NUMBER (0-7) AND A SUBARC NUMBER (0-7)
4362   9FE2 5F             L9FE2   CLRB                            DEFAULT VALUE OF ZERO
4363   9FE3 9D A5                  JSR   GETCCH                    GET CURRENT INPUT CHAR
4364   9FE5 27 11                  BEQ   L9FF8                     BRANCH IF NONE
4365   9FE7 BD B2 6D               JSR   SYNCOMMA                  SYNTAX CHECK FOR COMMA
4366   9FEA BD B1 41               JSR   LB141                     EVALUATE NUMERIC EXPRESSION
4367   9FED 96 4F                  LDA   FP0EXP                    GET EXPONENT OF FPA0
4368   9FEF 8B 06                  ADDA  #$06                      ADD 6 TO EXPONENT - MULTIPLY EXPONENT BY 64
4369   9FF1 97 4F                  STA   FP0EXP                    RESAVE EXPONENT
4370   9FF3 BD B7 0E               JSR   LB70E                     CONVERT FPA0 TO INTEGER IN ACCB
4371   9FF6 C4 3F                  ANDB  #$3F                      MAX VALUE OF 63
4372   9FF8 1F 98          L9FF8   TFR   B,A                       SAVE VALUE IN ACCA ALSO
4373   9FFA C4 07                  ANDB  #$07                      NOW ACCB CONTAINS SUBARC NUMBER
4374   9FFC 44                     LSRA                            *
4375   9FFD 44                     LSRA                            *
4376   9FFE 44                     LSRA                            * DIVIDE ACCA BY EIGHT - OCTANT NUMBER
4377   9FFF 39                     RTS
```

ALLCOL	00B5	GRPRAM	00BC	L81EB	81EB	L8316	8316	L857D	857D
ANGLE	00E8	HEXDOL	8BDD	L81F0	81F0	L831A	831A	L8581	8581
ARYDIS	0008	HORBEG	00BD	L81F2	81F2	L834D	834D	L858A	858A
ARYEND	001F	HORBYT	00B9	L81F4	81F4	L835E	835E	L858C	858C
ARYTAB	001D	HORDEF	00C7	L81F6	81F6	L8367	8367	L8592	8592
ATN	83B0	HOREND	00C3	L81F8	81F8	L836C	836C	L859D	859D
BAKCOL	00B3	INSTR	877E	L81FA	81FA	L837E	837E	L85AB	85AB
BASIC	A000	INT	BCEE	L81FC	81FC	L83A3	83A3	L85AF	85AF
BAWMST	A0E8	IRQVEC	010C	L81FE	81FE	L83A6	83A6	L85B3	85B3
BEGGRP	00BA	L8002	8002	L8200	8200	L83AB	83AB	L85B4	85B4
BINVAL	002B	L8031	8031	L8202	8202	L83B8	83B8	L85B6	85B6
BLKLEN	007D	L80D0	80D0	L8204	8204	L83C5	83C5	L85C2	85C2
BLKTYP	007C	L80DD	80DD	L8206	8206	L83D7	83D7	L85C3	85C3
CASBUF	01DA	L80DE	80DE	L8208	8208	L83DC	83DC	L85C7	85C7
CBUFAD	007E	L80DF	80DF	L820A	820A	L83DF	83DF	L85D1	85D1
CFNBUF	01D1	L80E1	80E1	L820C	820C	L83E0	83E0	L85D5	85D5
CHARAC	0001	L80E3	80E3	L820E	820E	L83E1	83E1	L85DE	85DE
CHARAD	00A6	L80E4	80E4	L8210	8210	L83E6	83E6	L85F3	85F3
CHGFLG	00DB	L80E6	80E6	L8212	8212	L83EB	83EB	L85F5	85F5
CINBFL	0070	L80E8	80E8	L8214	8214	L83F0	83F0	L860F	860F
CINCTR	0079	L80FF	80FF	L8216	8216	L83F5	83F5	L8613	8613
CINPTR	007A	L8100	8100	L8218	8218	L83FA	83FA	L861E	861E
CIRCLE	9E9D	L8101	8101	L821A	821A	L83FF	83FF	L8625	8625
COLOR	9542	L8112	8112	L821C	821C	L8404	8404	L8626	8626
COMVEC	0120	L8113	8113	L821E	821E	L8409	8409	L862E	862E
COS	8378	L811C	811C	L8221	8221	L840E	840E	L8630	8630
CSSVAL	00C1	L811D	811D	L8224	8224	L8413	8413	L8634	8634
CURLIN	0068	L8139	8139	L8227	8227	L8418	8418	L8646	8646
DEF	8871	L813C	813C	L822A	822A	L841D	841D	L864A	864A
DEL	8970	L8148	8148	L822D	822D	L841E	841E	L8650	8650
DEVNUM	006F	L8154	8154	L8230	8230	L8423	8423	L8659	8659
DEVPOS	006C	L8165	8165	L8233	8233	L8428	8428	L865C	865C
DLBAUD	00E6	L8168	8168	L8236	8236	L842D	842D	L8665	8665
DLOAD	8C18	L8170	8170	L823A	823A	L8432	8432	L866B	866B
DOSBAS	C000	L817D	817D	L8240	8240	L8437	8437	L8679	8679
DOTVAL	00E5	L8183	8183	L8245	8245	L843C	843C	L867C	867C
DRAW	9CB6	L8186	8186	L824A	824A	L8441	8441	L8685	8685
EDIT	8533	L818A	818A	L8250	8250	L8489	8489	L8687	8687
ENDFLG	0000	L818E	818E	L8257	8257	L8491	8491	L8694	8694
ENDGRP	00B7	L8193	8193	L8259	8259	L84AC	84AC	L86A6	86A6
ESC	001B	L8196	8196	L825B	825B	L84C4	84C4	L86D6	86D6
EVALEXPB	B70B	L8199	8199	L825D	825D	L84C9	84C9	L86EB	86EB
EXBAS	8000	L819D	819D	L825F	825F	L84CA	84CA	L86FD	86FD
EXECJP	009D	L81A1	81A1	L8261	8261	L84CF	84CF	L870E	870E
EXP	84F2	L81A5	81A5	L8263	8263	L84D4	84D4	L8724	8724
EXPJMP	011D	L81AB	81AB	L8265	8265	L84D9	84D9	L8727	8727
FILSTA	0078	L81B1	81B1	L8267	8267	L84DE	84DE	L872E	872E
FIX	8524	L81B7	81B7	L8269	8269	L84E3	84E3	L873F	873F
FORCOL	00B2	L81BC	81BC	L826B	826B	L84E8	84E8	L8746	8746
FP0EXP	004F	L81C2	81C2	L826D	826D	L84ED	84ED	L8748	8748
FP0SGN	0054	L81C7	81C7	L826F	826F	L8501	8501	L8768	8768
FP1SGN	0061	L81CA	81CA	L8271	8271	L8504	8504	L876B	876B
FPA0	0050	L81CD	81CD	L8285	8285	L8529	8529	L8776	8776
FPSBYT	0063	L81D1	81D1	L82BB	82BB	L852C	852C	L877B	877B
FRETOP	0021	L81D6	81D6	L82CF	82CF	L8538	8538	L879C	879C
GET	9755	L81DB	81DB	L82D8	82D8	L854D	854D	L87BE	87BE
GETCCH	00A5	L81DF	81DF	L82F1	82F1	L855C	855C	L87CD	87CD
GETNCH	009F	L81E4	81E4	L8310	8310	L855D	855D	L87D6	87D6
GIVABF	B4F4	L81E9	81E9	L8311	8311	L8570	8570	L87D8	87D8

L87D9	87D9	L8A9B	8A9B	L8D1D	8D1D	L8FC4	8FC4	L9249	9249
L87DF	87DF	L8AAC	8AAC	L8D26	8D26	L8FC6	8FC6	L924D	924D
L87EB	87EB	L8AB2	8AB2	L8D48	8D48	L8FD8	8FD8	L9256	9256
L88ØØ	88ØØ	L8AB9	8AB9	L8D58	8D58	L8FE3	8FE3	L9262	9262
L88ØA	88ØA	L8ACØ	8ACØ	L8D6Ø	8D6Ø	L8FE5	8FE5	L9263	9263
L88ØC	88ØC	L8AC7	8AC7	L8D62	8D62	L8FF2	8FF2	L926A	926A
L88ØE	88ØE	L8AD3	8AD3	L8D6A	8D6A	L8FFA	8FFA	L927A	927A
L881F	881F	L8ADD	8ADD	L8D6B	8D6B	L9Ø11	9Ø11	L927B	927B
L882E	882E	L8AE1	8AE1	L8D72	8D72	L9Ø15	9Ø15	L9281	9281
L8834	8834	L8AE5	8AE5	L8D7C	8D7C	L9Ø23	9Ø23	L928E	928E
L8845	8845	L8AE9	8AE9	L8D89	8D89	L9Ø2C	9Ø2C	L928F	928F
L8862	8862	L8AEB	8AEB	L8D8B	8D8B	L9Ø5Ø	9Ø5Ø	L9298	9298
L8866	8866	L8AED	8AED	L8DA7	8DA7	L9Ø54	9Ø54	L929C	929C
L886E	886E	L8AEF	8AEF	L8DB8	8DB8	L9Ø6Ø	9Ø6Ø	L929E	929E
L88A1	88A1	L8B13	8B13	L8DBC	8DBC	L9Ø65	9Ø65	L92AØ	92AØ
L88B1	88B1	L8B17	8B17	L8DC5	8DC5	L9Ø7C	9Ø7C	L92A2	92A2
L88B4	88B4	L8B1B	8B1B	L8DC9	8DC9	L9Ø96	9Ø96	L92A4	92A4
L88D9	88D9	L8B24	8B24	L8DD4	8DD4	L9Ø9E	9Ø9E	L92A6	92A6
L88EF	88EF	L8B41	8B41	L8DDE	8DDE	L9ØA9	9ØA9	L92C2	92C2
L89ØB	89ØB	L8B55	8B55	L8DE6	8DE6	L9ØAA	9ØAA	L92DD	92DD
L89ØD	89ØD	L8B67	8B67	L8DF6	8DF6	L9ØB2	9ØB2	L92E5	92E5
L89ØF	89ØF	L8B71	8B71	L8DF7	8DF7	L9ØB3	9ØB3	L92E9	92E9
L891C	891C	L8B7B	8B7B	L8DF9	8DF9	L9ØB8	9ØB8	L92ED	92ED
L8927	8927	L8B7F	8B7F	L8DFD	8DFD	L9ØBD	9ØBD	L92F3	92F3
L892C	892C	L8B8A	8B8A	L8E Ø4	8E Ø4	L9ØBF	9ØBF	L92F4	92F4
L8943	8943	L8B8C	8B8C	L8E ØC	8E ØC	L9ØCB	9ØCB	L92FC	92FC
L8944	8944	L8BAE	8BAE	L8E1D	8E1D	L9ØE2	9ØE2	L93Ø3	93Ø3
L8952	8952	L8BBE	8BBE	L8E25	8E25	L9ØEA	9ØEA	L93Ø9	93Ø9
L8955	8955	L8BC9	8BC9	L8E37	8E37	L9ØEE	9ØEE	L9317	9317
L896Ø	896Ø	L8BD9	8BD9	L8E3B	8E3B	L9ØFF	9ØFF	L931A	931A
L898C	898C	L8BE5	8BE5	L8E5F	8E5F	L91Ø8	91Ø8	L931D	931D
L899Ø	899Ø	L8BEA	8BEA	L8E69	8E69	L91ØD	91ØD	L932Ø	932Ø
L8992	8992	L8BFF	8BFF	L8E71	8E71	L9116	9116	L932C	932C
L8993	8993	L8C Ø7	8C Ø7	L8E82	8E82	L911B	911B	L9338	9338
L899F	899F	L8CØB	8CØB	L8E88	8E88	L9129	9129	L9349	9349
L89AE	89AE	L8C1B	8C1B	L8E95	8E95	L913Ø	913Ø	L9351	9351
L89B4	89B4	L8C25	8C25	L8EA8	8EA8	L913C	913C	L9356	9356
L89B8	89B8	L8C42	8C42	L8EAE	8EAE	L9141	9141	L935B	935B
L89BF	89BF	L8C44	8C44	L8EB4	8EB4	L915A	915A	L9366	9366
L89CØ	89CØ	L8C5F	8C5F	L8EB7	8EB7	L9167	9167	L9377	9377
L89D2	89D2	L8C62	8C62	L8EB9	8EB9	L916F	916F	L938F	938F
L89E1	89E1	L8C85	8C85	L8EBB	8EBB	L9177	9177	L939E	939E
L89FC	89FC	L8C96	8C96	L8ED2	8ED2	L9185	9185	L93B2	93B2
L8AØ2	8AØ2	L8C9B	8C9B	L8ED8	8ED8	L919E	919E	L93B8	93B8
L8AØ4	8AØ4	L8CB1	8CB1	L8EDD	8EDD	L91AØ	91AØ	L93CE	93CE
L8A2Ø	8A2Ø	L8CBF	8CBF	L8EE2	8EE2	L91B6	91B6	L93E8	93E8
L8A2D	8A2D	L8CC1	8CC1	L8EEF	8EEF	L91BA	91BA	L93E9	93E9
L8A3A	8A3A	L8CC5	8CC5	L8EFB	8EFB	L91C4	91C4	L942Ø	942Ø
L8A3D	8A3D	L8CC6	8CC6	L8F1A	8F1A	L91CC	91CC	L9429	9429
L8A67	8A67	L8CCD	8CCD	L8F2Ø	8F2Ø	L91CD	91CD	L943Ø	943Ø
L8A68	8A68	L8CDØ	8CDØ	L8F24	8F24	L91DØ	91DØ	L9432	9432
L8A71	8A71	L8CDD	8CDD	L8F26	8F26	L91E4	91E4	L9434	9434
L8A77	8A77	L8CE2	8CE2	L8F41	8F41	L91E9	91E9	L9443	9443
L8A83	8A83	L8CE4	8CE4	L8F4F	8F4F	L91F1	91F1	L9444	9444
L8A86	8A86	L8DØ1	8DØ1	L8F5A	8F5A	L92ØØ	92ØØ	L9451	9451
L8A9Ø	8A9Ø	L8DØ2	8DØ2	L8F74	8F74	L92Ø2	92Ø2	L945E	945E
L8A91	8A91	L8D12	8D12	L8F8F	8F8F	L9211	9211	L946B	946B
L8A95	8A95	L8D14	8D14	L8F96	8F96	L9213	9213	L946C	946C
L8A99	8A99	L8D1B	8D1B	L8FB3	8FB3	L9235	9235	L946E	946E

L947B	947B	L97Ø8	97Ø8	L9934	9934	L9B97	9B97	L9DFC	9DFC
L948A	948A	L97ØA	97ØA	L993B	993B	L9B98	9B98	L9EØ5	9EØ5
L948C	948C	L97ØC	97ØC	L9954	9954	L9B9A	9B9A	L9EØD	9EØD
L948E	948E	L97ØE	97ØE	L9958	9958	L9BAC	9BAC	L9E21	9E21
L949Ø	949Ø	L971Ø	971Ø	L9969	9969	L9BBE	9BBE	L9E2B	9E2B
L9492	9492	L9714	9714	L996C	996C	L9BC8	9BC8	L9E32	9E32
L9494	9494	L971D	971D	L996E	996E	L9BE2	9BE2	L9E56	9E56
L949D	949D	L9736	9736	L997Ø	997Ø	L9BEB	9BEB	L9E5B	9E5B
L94A1	94A1	L973F	973F	L9983	9983	L9BEE	9BEE	L9E5E	9E5E
L94B2	94B2	L9751	9751	L998C	998C	L9BF1	9BF1	L9E69	9E69
L94C1	94C1	L9752	9752	L999E	999E	L9BF2	9BF2	L9E6A	9E6A
L94CC	94CC	L975A	975A	L99A1	99A1	L9BF7	9BF7	L9E78	9E78
L94DD	94DD	L9765	9765	L99AC	99AC	L9BFC	9BFC	L9E79	9E79
L94E2	94E2	L979A	979A	L99BA	99BA	L9CØ1	9CØ1	L9E7D	9E7D
L95Ø2	95Ø2	L979F	979F	L99C6	99C6	L9CØA	9CØA	L9E81	9E81
L95Ø6	95Ø6	L97AE	97AE	L99CB	99CB	L9C1B	9C1B	L9E85	9E85
L95ØD	95ØD	L97B7	97B7	L99DF	99DF	L9C27	9C27	L9E89	9E89
L9514	9514	L97CØ	97CØ	L99E8	99E8	L9C3E	9C3E	L9E8D	9E8D
L951B	951B	L97CA	97CA	L99EE	99EE	L9C5A	9C5A	L9E91	9E91
L9522	9522	L97D3	97D3	L99F2	99F2	L9C5B	9C5B	L9E95	9E95
L9536	9536	L97DE	97DE	L9AØ9	9AØ9	L9C62	9C62	L9E99	9E99
L953B	953B	L97EB	97EB	L9AØB	9AØB	L9C7A	9C7A	L9EA3	9EA3
L9542	9542	L98Ø8	98Ø8	L9A12	9A12	L9C92	9C92	L9EDF	9EDF
L9552	9552	L98ØC	98ØC	L9A32	9A32	L9CC6	9CC6	L9EE9	9EE9
L9559	9559	L9816	9816	L9A37	9A37	L9CCB	9CCB	L9EFD	9EFD
L955A	955A	L9822	9822	L9A39	9A39	L9CD1	9CD1	L9F11	9F11
L955D	955D	L9823	9823	L9A43	9A43	L9CD3	9CD3	L9F31	9F31
L956C	956C	L982E	982E	L9A5C	9A5C	L9CDD	9CDD	L9F5B	9F5B
L9576	9576	L9831	9831	L9A6D	9A6D	L9CF4	9CF4	L9F66	9F66
L9578	9578	L9839	9839	L9A8B	9A8B	L9CFC	9CFC	L9F68	9F68
L9579	9579	L983D	983D	L9A9A	9A9A	L9D21	9D21	L9F7A	9F7A
L957B	957B	L983E	983E	L9A9F	9A9F	L9D4C	9D4C	L9F85	9F85
L9581	9581	L9842	9842	L9AAD	9AAD	L9D4F	9D4F	L9F87	9F87
L9598	9598	L9843	9843	L9AAF	9AAF	L9D57	9D57	L9F8F	9F8F
L959A	959A	L9847	9847	L9AB2	9AB2	L9D59	9D59	L9F9A	9F9A
L95A2	95A2	L9848	9848	L9ACØ	9ACØ	L9D61	9D61	L9FA3	9FA3
L95AA	95AA	L984C	984C	L9AC3	9AC3	L9D69	9D69	L9FA6	9FA6
L95AC	95AC	L984D	984D	L9ACD	9ACD	L9D6C	9D6C	L9FA7	9FA7
L95CF	95CF	L9851	9851	L9ADØ	9ADØ	L9D6D	9D6D	L9FB4	9FB4
L95F7	95F7	L9852	9852	L9AE7	9AE7	L9D72	9D72	L9FB5	9FB5
L95FB	95FB	L985D	985D	L9AEB	9AEB	L9D7B	9D7B	L9FD4	9FD4
L96ØØ	96ØØ	L9864	9864	L9AF2	9AF2	L9D82	9D82	L9FDF	9FDF
L96Ø7	96Ø7	L986E	986E	L9AFA	9AFA	L9D83	9D83	L9FE2	9FE2
L96Ø9	96Ø9	L9873	9873	L9AFF	9AFF	L9D87	9D87	L9FF8	9FF8
L96ØF	96ØF	L9877	9877	L9B15	9B15	L9D8C	9D8C	LAØE2	AØE2
L9616	9616	L9884	9884	L9B18	9B18	L9D8F	9D8F	LA171	A171
L962E	962E	L9890	9890	L9B1F	9B1F	L9D92	9D92	LA176	A176
L9650	9650	L9894	9894	L9B22	9B22	L9D98	9D98	LA282	A282
L966C	966C	L989B	989B	L9B2B	9B2B	L9DA9	9DA9	LA35F	A35F
L966D	966D	L98A1	98A1	L9B49	9B49	L9DB6	9DB6	LA3ED	A3ED
L967F	967F	L98A7	98A7	L9B57	9B57	L9DBD	9DBD	LA4Ø6	A4Ø6
L9687	9687	L98B1	98B1	L9B5A	9B5A	L9DC3	9DC3	LA429	A429
L96BD	96BD	L98CC	98CC	L9B5F	9B5F	L9DC4	9DC4	LA42D	A42D
L96CB	96CB	L98D7	98D7	L9B64	9B64	L9DC7	9DC7	LA444	A444
L96D4	96D4	L98E8	98E8	L9B72	9B72	L9DC8	9DC8	LA44C	A44C
L96DB	96DB	L98EB	98EB	L9B8Ø	9B8Ø	L9DCB	9DCB	LA491	A491
L96E6	96E6	L98F2	98F2	L9B83	9B83	L9DDC	9DDC	LA498	A498
L96EC	96EC	L990A	990A	L9B88	9B88	L9DE8	9DE8	LA5Ø5	A5Ø5
L97Ø6	97Ø6	L9931	9931	L9B8C	9B8C	L9DF2	9DF2	LA578	A578

LA59A	A59A	LB4F3	B4F3	PAINT	98EC	V4D	004D
LA5A5	A5A5	LB50F	B50F	PCLEAR	968B	VARPTR	86BE
LA5AE	A5AE	LB518	B518	PCLS	9532	VALTMP	0006
LA5C7	A5C7	LB51A	B51A	PCOPY	9723	VARDES	003B
LA5E4	A5E4	LB56D	B56D	PIA0	FF00	VARPTR	0039
LA616	A616	LB643	B643	PIA1	FF20	VARTAB	001B
LA619	A619	LB654	B654	PLAY	9A22	VCB	00CB
LA635	A635	LB657	B657	PLYTMR	00E3	VCD	00CD
LA644	A644	LB659	B659	PMOD	9621	VCF	00CF
LA648	A648	LB69B	B69B	PMODE	00B6	VD1	00D1
LA65F	A65F	LB6A4	B6A4	POS	86AC	VD3	00D3
LA7D8	A7D8	LB6AD	B6AD	PPOINT	9339	VD4	00D4
LA7E9	A7E9	LB70E	B70E	PRESET	9365	VD5	00D5
LA7F4	A7F4	LB734	B734	PSET	9361	VD6	00D6
LA974	A974	LB738	B738	PUT	9758	VD7	00D7
LA976	A976	LB73D	B73D	RELFLG	000A	VD8	00D8
LA9A2	A9A2	LB740	B740	RENUM	8A09	VD9	00D9
LA9BB	A9BB	LB7C2	B7C2	RESET	FFFE	VDA	00DA
LAC1E	AC1E	LB958	B958	RESSGN	0062	VERBEG	00BF
LAC33	AC33	LB95C	B95C	RSTVEC	0072	VERDEF	00C9
LAC46	AC46	LB99F	B99F	RVEC15	018B	VEREND	00C5
LAC60	AC60	LB9AC	B9AC	RVEC17	0191	VOLHI	00DF
LAC73	AC73	LB9AF	B9AF	RVEC18	0194	VOLLOW	00E0
LAC7C	AC7C	LB9B4	B9B4	RVEC19	0197	WCOLOR	00B4
LACA8	ACA8	LB9B9	B9B9	RVEC20	019A	XBWMST	80C0
LACEF	ACEF	LB9C2	B9C2	RVEC22	01A0	XIRQSV	894C
LACF1	ACF1	LBA1C	BA1C	RVEC23	01A3	XVEC15	8846
LAD01	AD01	LBA3A	BA3A	RVEC3	0167	XVEC17	88F0
LAD19	AD19	LBA92	BA92	RVEC4	016A	XVEC18	829C
LAD21	AD21	LBACA	BACA	RVEC8	0176	XVEC19	87E5
LAD26	AD26	LBB48	BB48	RVEC9	0179	XVEC20	82B9
LAD33	AD33	LBB5C	BB5C	SAM	FFC0	XVEC23	8304
LAD9E	AD9E	LBB6A	BB6A	SCALE	00E9	XVEC3	8273
LADC6	ADC6	LBB82	BB82	SCREEN	9670	XVEC4	8CF1
LADD4	ADD4	LBB8F	BB8F	SETFLG	00C2	XVEC8	8286
LADEB	ADEB	LBC14	BC14	SQR	8480	XVEC9	8E90
LAE15	AE15	LBC2F	BC2F	STRINOUT	B99C	ZERO	008A
LAED2	AED2	LBC35	BC35	SYNCOMMA	B26D		
LAEE0	AEE0	LBC4C	BC4C	TAN	8381		
LAF67	AF67	LBC5F	BC5F	TEMPO	00E2		
LAFA4	AFA4	LBC6D	BC6D	TEMPTR	000F		
LB035	B035	LBCA0	BCA0	TIMER	8968		
LB141	B141	LBCC8	BCC8	TIMOUT	00E7		
LB143	B143	LBD99	BD99	TIMVAL	0112		
LB146	B146	LBDC5	BDC5	TINPTR	002F		
LB156	B156	LBDCC	BDCC	TMPSTK	00DC		
LB158	B158	LBDD9	BDD9	TRCFLG	00AF		
LB244	B244	LBEE9	BEE9	TROFF	86A8		
LB262	B262	LBEF0	BEF0	TRON	86A7		
LB267	B267	LBEFF	BEFF	TXTTAB	0019		
LB26A	B26A	LBF78	BF78	USR0	013E		
LB26F	B26F	LBFA6	BFA6	USRADR	00B0		
LB277	B277	LC002	C002	V40	0040		
LB284	B284	LINBUF	02DC	V41	0041		
LB2CE	B2CE	LINE	93BB	V43	0043		
LB357	B357	LOG	8446	V45	0045		
LB35C	B35C	MEMSIZ	0027	V47	0047		
LB3A2	B3A2	NOTELN	00E1	V4A	004A		
LB44A	B44A	OCTAVE	00DE	V4B	004B		

EXPLANATION OF TERMS:

CALPOS - Refer to chapter 3 page 6 for detailed explanation.

NORMALIZING - Refer to chapter 3 page 6 for detailed explanation.

PIXEL - Refer to chapter 3 page 6 for detailed explanation.

SPECIAL NOTE: Some of the following routines require that certain registers and/or variables be set up with certain values before calling them. If an error is generated while in one of these routines, the normal error message will be generated and the routine will return control to BASIC. In order to prevent this from happening, the error must be intercepted by using the ram hook for the error processing routine (RVEC17).

MODIFIED REGISTERS	ADDRESS	DESCRIPTION
A,B,X	8524	FIX NUMBER IN FPAØ - Converts the number in FPAØ to an integer value and forces it to be positive.
A,B,X	881F	EVALUATE &H - Get the value after the &H from the program line and convert it to a numerical value.
A,U	928F	GET CALPOS ROUTINE ADDRESS - Get the address of the routine which will convert the horizontal and vertical coordinates into an absolute screen address and pixel mask depending upon the current PMODE. Return the address of the routine in the U register.
A,U	9298	CALPOS FOR CURRENT PMODE - This routine jumps to the correct calpos routine depending upon the current PMODE.
A,X,U	92A6	CALPOS 2 COLOR MODE - Calculates toe absolute screen address and pixel mask for the 2 color hires mode. Enter with X,Y coordinates in HORBEG and VERBEG and exit with address in the X register and the pixel mask in ACCA.
A,X,U	92C2	CALPOS 4 COLOR MODE - Calculates toe absolute screen address and pixel mask for the 4 color hires mode. Enter with X,Y coordinates in HORBEG and VERBEG and exit with address in the X register and the pixel mask in ACCA.
B,X	92E9	ADJUST SCREEN POINTER DOWN A ROW - Move the X

		register down one graphic row. The number of bytes per horizontal graphic row must be in HORBYT.
A,X	92ED	MOVE A PIXEL TO THE RIGHT (2 COLOR) - Adjust the X register and ACCA one pixel position to the right in the 2 color mode. Enter with the absolute screen address in the X register and the pixel mask in ACCA.
A,X	92F4	MOVE A PIXEL TO THE RIGHT (4 COLOR) - Adjust the X register and ACCA one pixel position to the right in the 4 color mode. Enter with the absolute screen address in the X register and the pixel mask in ACCA.
A,B,U	931D	NORMALIZE COORDINATES - Adjust the horizontal and vertical coordinates for the current PMODE. Enter with X,Y coordinates in HORBEG and VERBEG, the normalized coordinates will be returned in the same.
A,B	9377	TURN ON A PIXEL - Turn on the pixel which is being pointed to by the X register (absolute screen address) and ACCA (pixel mask) to the color in ALLCOL. Set CHGFLG <> Ø if pixel color was unchanged by the action.
ALL	94Ø8	DRAW A BOX - Encloses a diagonal line with a box (box function of LINE). Enter with the start and end coordinates of the original line in HORBEG, VERBEG, HOREND and VEREND.
ALL	9434	FILL BOX - Draw a series of horizontal lines from BERBEG to VEREND
ALL	9444	DRAW A HORIZONTAL LINE - Draw a horizontal line from HOREND to HORBEG at the vertical coordinate VERBEG with the color in ALLCOL.
ALL	946E	DRAW A VERTICAL LINE - Draw a vertical line from VEREND to VERBEG at the horizontal coordinate HORBEG with the color in ALLCOL.
B,U	9494	POINT TO PIXEL MOVE ROUTINE - Point the U register to the routine which will move the current pixel to the right one position for the current PMODE.
ALL	94A1	DRAW A LINE - Draw a line from (HORBEG,VERBEG) to (HOREND, VEREND).
X	95Ø6	INCREMENT HORIZONTAL POSITION - Gets the current horizontal coordinate (HORBEG) and moves it one to the right.

X	950D	INCREMENT VERTICAL POSITION - Gets the current vertical coordinate (VERBEG) and moves it one down.
X	9515	DECREMENT HORIZONTAL POSITION - Gets the current horizontal coordinate (HORBEG) and moves it one to the left.
X	951B	DECREMENT VERTICAL POSITION - Gets the current vertical coordinate (VERBEG) and moves it one up.
A,B,X,U	9522	GET MAXIMUM COORDINATES - Get the maximum values of the horizontal and vertical coordinates for the current PMODE. Return HOR in VD3 and VER in VD5.
A,B,X	9536	CLEAR GRAPHIC SCREEN - Clear the current graphics screen to the color in ACCB. If ACCB = Ø then clear to the current background color.
A,B	9710	CALCULATE ABS(VEREND-VERBEG) - Calculate the absolute value of the distance between VEREND and VERBEG. Carry flag will indicate which was the larger coordinate.
A,B	971D	CALCULATE ABS(HOREND-HORBEG) - Calculate the absolute value of the distance between HOREND and HORBEG. Carry flag will indicate which was the larger coordinate.
U,Y	9FB5	16 BIT MULTIPLY - Multiply (unsigned) two 16 bit numbers together. Enter with one number in ACCD and the other in the X register. The four byte product will be returned in the Y and U registers.

START	END	DESCRIPTION
8ØØØ	8ØØ1	EXTENDED BASIC ROM IDENTIFIER
8ØDE	8ØE7	COMMAND INTERPRETATION TABLE ROM IMAGE
8ØE8	813B	COPYRIGHT MESSAGES
8183	81E9	PRIMARY RESERVED WORD TABLE
81FØ	821D	PRIMARY RESERVED WORD DISPATCH TABLE
821E	8256	SECONDARY RESERVED WORD TABLE
8257	8272	SECONDARY RESERVED WORD DISPATCH TABLE
83AB	83AF	FLOATING POINT VALUE FOR PI/2
83EØ	841C	TAYLOR SERIES COEFFICIENTS FOR ARCTANGENT
841D	8431	TAYLOR SERIES COEFFICIENTS FOR NATURAL LOG(X)
8432	8436	FLOATING POINT VALUE FOR .5*SQR(2)
8437	843B	FLOATING POINT VALUE FOR THE SQUARE ROOT OF 2
843C	844Ø	FLOATING POINT VALUE FOR -.5
8441	8445	FLOATING POINT VALUE FOR THE NATURAL LOG OF 2
84C4	84C8	FLOATING POINT VALUE FOR CORRECTION FACTOR OF EXPONENTIAL FUNCTION
84C9	84F1	TAYLOR SERIES FOR E^X
89ØB	89ØE	ERROR MESSAGES
8BD9	8BDC	UL' (UNKNOWN LINE NUMBER) MESSAGE
929C	92A5	JUMP TABLE FOR CALPOS ROUTINES
92DD	92E4	2 COLOR MODE PIXEL MASKS
92E5	92E8	4 COLOR MODE PIXEL MASKS
948A	9493	JUMP TABLE OF ADDRESSES WHICH WILL MOVE POINTERS ONE PIXEL TO THE RIGHT
97Ø6	97ØF	TABLE OF HOW MANY BYTES PER HORIZONTAL GRAPHIC

		ROW AND HOW MUCH RAM IS USED FOR ONE HIRES SCREEN
9839	9851	LOOKUP TABLE FOR PSET, PRESET, AND, OR, AND NOT MODIFIERS FOR THE PUT COMMAND
9C5B	9C61	NUMERICAL NOTE VALUES FOR LETTER NOTES
9C62	9C79	TIMING DELAYS FOR OCTAVE 1
9C7A	9C91	TIMING DELAYS FOR OCTAVE 2
9C92	9CB5	TIMING DELAYS FOR OCTAVES 3, 4 AND 5
9E79	9E9C	TABLE OF SINES AND COSINES FOR CIRCLE

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).

BASIC/EXTENDED ERROR JUMPS

NAME	NBR	LABEL	ADDR		DESCRIPTION
NF	ØØ	LB1Ø8	B1Ø8		NEXT WITHOUT FOR
SN	Ø1	LB277	B277		SYNTAX ERROR
RG	Ø2	LAECF	AECF		RETURN WITHOUT GOSUB
OD	Ø3	LBØC3	BØC3	COND	OUT OF DATA
FC	Ø4	LB44A	B44A		ILLEGAL FUNCTION CALL
OV	Ø5	LBA92	BA92		OVERFLOW
OM	Ø6	LAC44	AC44		OUT OF MEMORY
UL	Ø7	LAED2	AED2		UNDEFINED LINE NUMBER
BS	Ø8	LB447	B447		BAD SUBSCRIPT
DD	Ø9	LB43B	B43B	COND	REDIMENSIONED ARRAY
/Ø	1Ø	LBCØ6	BCØ6		DIVISION BY ZERO
ID	11	INPUT	AFF5	COND	ILLEGAL DIRECT STATEMENT
TM	12	LB151	B151		TYPE MISMATCH
OS	13	LB585	B585	COND	OUT OF STRING SPACE
LS	14	LB625	B625		STRING TOO LONG
ST	15	LB553	B553		STRING FORMULA TOO COMPLEX
CN	16	LAE32	AE32	COND	CAN'T CONTINUE
FD	17	LAFD6	AFD6	COND	BAD FILE DATA
AO	18	LA61C	A61C		FILE ALREADY OPEN
DN	19	LA61F	A61F		DEVICE NUMBER ERROR
IO	2Ø	LA619	A619		INPUT/OUTPUT ERROR
FM	21	LA616	A616		BAD FILE MODE
NO	22	LA3FB	A3FB		FILE NOT OPEN
IE	23	LBØ3F	BØ3F	COND	INPUT PAST END OF FILE
DS	24	LAC94	AC94	COND	DIRECT STATEMENT IN FILE
UF	25	L88BF	88BF	COND	UNDEFINED FUNCTION CALL
NE	26	L8CDD	8CDD		FILE NOT FOUND

The addresses given for the entry points are valid for COLOR BASIC Versions 1.Ø, 1.1, 1.2 and Extended BASIC Versions 1.Ø, 1.1, and 2.Ø. If the address is followed by a COND, the corresponding entry point is not unconditional, meaning that ACCB will be loaded with the error but some sort of test will be imposed before program control will be passed to the error handler. As required, these conditional errors may be generated by loading a value equal to 2*(error number) into ACCB and then JMPing to $AC46.

The differences between Extended Basic 1.Ø and 1.1 are not earth-shaking. The primary difference involves the bug in the PCLEAR command which caused BASIC programs to generate a syntax or illegal function call error at certain times when a PCLEAR command was executed in a BASIC program. This error was caused by the fact that the BASIC input pointer was not adjusted when the program was moved as a result of a PCLEAR command. Accordingly, when control was returned to the BASIC program after a PCLEAR command the BASIC input pointer would invariably end up pointing to the middle of the program which would cause the error.

DIFFERENCES BETWEEN EXTENDED BASIC 1.Ø AND 1.1

ADDRESS

80D0-80DD Patch to move the BASIC input pointer during PCLEAR (see Figure G1). The originial code in the 1.Ø version was designed to allow the SAM chip to be programmed for 64K RAMs on power up but the code was never accessed by any routines in any of the 3 Basic ROMs.

```
                    * THIS CODE IS NOT USED BY ANY OF THE BASICS
     80D0  B6   FF22        LDA   PIA1+2    READ PIA PORT B
     80D3  85   02          BITA  #2        CHECK MEM SIZE JUMPER
     80D5  26   03          BNE   L80DA     BRANCH IF HIGH
     80D7  B7   FFDD        STA   SAM+29    SET SAM CNTL REG MEM SIZE TO 64K
     80DA  6E   84    L80DA JMP   ,X        JUMP TO ADDRESS IN X REG
     80DC  00   00          FCB   $00,$00   DEAD SPACE
```

Figure G1 - Version 1.Ø Code

80FF Change version number 1.Ø/1.1

8112 Change Copyright Year (units digit)

8C1B-8C22 Patch to fix the DLOAD bug (see Figure G2). The code in 1.Ø version did not allow for the fact that the current BASIC input character was not in ACCA following the CLOSEing of cassette files (JSR LA429).

```
     8C1B  6F   E2          CLR   ,-S       SAVE DEFAULT TOKEN (NON DLOADM) ON STACK
     8C1D  81   4D          CMPA  #'M'      IS IT DLOADM?
     8C1F  26   04          BNE   L8C25     NO
     8C21  A7   E4          STA   ,S        SAVE THE M ON THE STACK
```

Figure G2 - Version 1.Ø code

8C51 Change instruction op code from BNE(1.Ø) to BEQ(1.1). This change was necessitated by the previous fix to DLOAD.

9179 Change op code address field from #12(1.Ø) to #1Ø(1.1). This change and the one following fix a minor bug in

the ASCII to floating point conversion in PRINT USING.

917D Change op code address field from #'9'+3(1.Ø) to #'9'+1 (1.1)

962C-962D Change instruction from LDA #6(1.Ø) to LDA GRPRAM(1.1). This change allows the start of the first graphic page to be determined by the value in the direct page variable GRPRAM (start of graphic RAM) rather than the absolue value of 6 which is valid for an Extended Basic ONLY system.

96A3-96B3 Patch the PCLEAR command to fix the PCLEAR bug (see Figure G3). This patch merely rearranges existing code to allow for the call (JSR L8ØDØ) to the routine which will adjust the BASIC input pointer.

```
96A3  1Ø25 1DA3       LBLO  LB44A     IF TRYING TO CLEAR LESS THAN END
                                      OF CURRENT PAGE = 'ILLEGAL FUNCTION CALL'
96A7  93   19         SUBD  TXTTAB    SUBTRACT START OF RAM
96A9  D3   1B         ADDD  VARTAB    ADD END OF BASIC PROGRAM
96AB  1F   Ø1         TFR   D,X       X=TOP OF PCLEARED SPACE+SIZE OF BASIC
                                      PROGRAM
96AD  C3   ØØC8       ADDD  #2ØØ      ADD 2ØØ - LEAVE SOME ROOM FOR STACK
96BØ  93   21         SUBD  FRETOP    SUBTRACT OUT START OF CLEARED SPACE
96B2  24   B9         BCC   L966D     NO ROOM LEFT - 'ILLEGAL FUNCTION CALL'
```

Figure G3 - Version 1.Ø code

DISPLAY CHARACTER SET

HEX VALUE		CHARACTER	HEX VALUE		CHARACTER	HEX VALUE		CHARACTER
Non-Inverted	Inverted		Non-Inverted	Inverted		Non-Inverted	Inverted	
ØØ	4Ø	@	18	58	X	3Ø	4Ø	Ø
Ø1	41	A	19	59	Y	31	41	1
Ø2	42	B	1A	5A	Z	32	42	2
Ø3	43	C	1B	5B	[	33	43	3
Ø4	44	D	1C	5C	\	34	44	4
Ø5	45	E	1D	5D	]	35	45	5
Ø6	46	F	1E	5E	↑	36	46	6
Ø7	47	G	1F	5F	←	37	47	7
Ø8	48	H	2Ø	6Ø		38	48	8
Ø9	49	I	21	61	!	39	49	9
ØA	4A	J	22	62	"	3A	4A	:
ØB	4B	K	23	63	#	3B	4B	;
ØC	4C	L	24	64	$	3C	4C	<
ØD	4D	M	25	65	%	3D	4D	=
ØE	4E	N	26	66	&	3E	4E	>
ØF	4F	O	27	67	'	3F	4F	?
1Ø	5Ø	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	/			