EXTENDED BASIC UNRAVELLED II

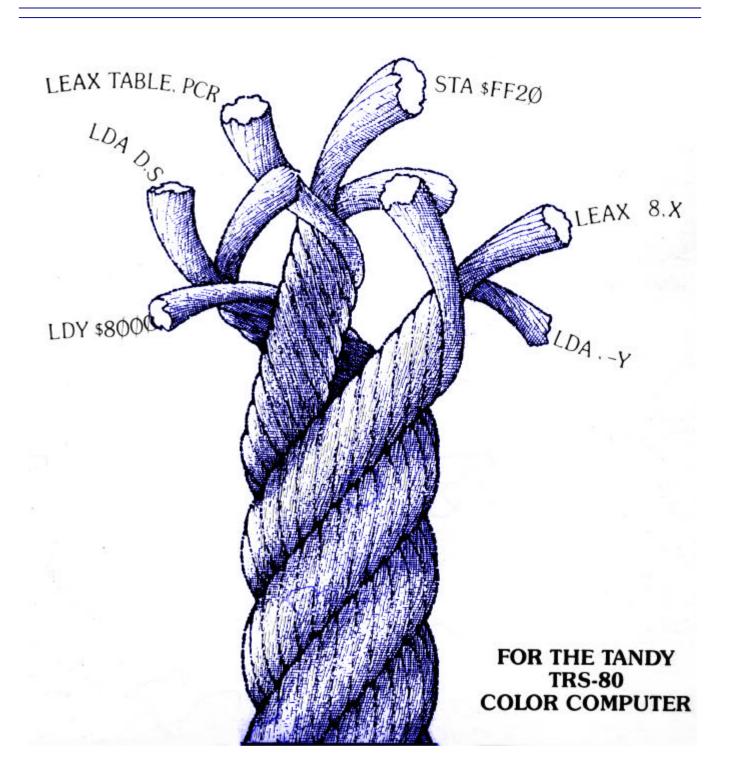


TABLE OF CONTENTS

1	FORE	NORD	1
2	INTRO	DDUCTION	3
3	HOW -	TO USE THIS BOOK	4
4		RIPTION OF EXTENDED BASIC HICS ROUTINES	5 7
		APPENDICES	
	Α	MEMORY MAP	
	В	DISASSEMBLY OF EXTENDED BASIC	
	С	EXTENDED BASIC SYMBOL TABLE	
	D	EXTENDED BASIC ROUTINES AND ENTRY POINTS	
	Е	EXTENDED BASIC S DATA/ASCII TABLES	
	F	EXTENDED BASIC ERROR ENTRY POINTS	
	G	EXTENDED BASIC 1.0 DIFFERENCES	
	н	ASCII CHADT	

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 0.
- 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 $1^{\rm st}$ 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 6809 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 6809 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 \$20.00 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 6809 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 1.0) 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 1.0 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 \emptyset 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 \emptyset ,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 #Ø 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 #Ø 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 \emptyset -255 and a vertical coordinate from \emptyset -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.

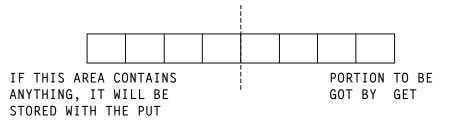
REVISED:12/26/99 WALTER K ZYDHEK

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.

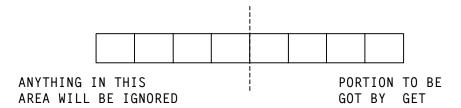
If the G option is not used, BASIC figures out which byte the GET starts in and stores the entire byte into the array (even if the start point is in the middle of the byte) which means that information that was not actually within the limits of the GET will be stored at the new location when PUT is used.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



PORTION THAT GETS STORED INTO THE ARRAY AND WILL BE PUT AT THE NEW LOCATION

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 10 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.



PORTION THAT GETS STORED INTO THE ARRAY AND WILL BE PUT AT THE NEW LOCATION

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 1.0 (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 1.0 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 0).

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 \emptyset TO 3 which was parsed from the BASIC program line gets converted to a binary \emptyset 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.

PPOTNT

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.

10 A=10:B=13 20 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 20 is equivalent to the following:

20 DRAW "BM10.10: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					
	CØØØ	ROMPAK	EQU	\$CØØØ	
0002					
0003	ØØØ8	BS	EQU	8	BACKSPACE
0004	ØØØD	CR	EQU	\$D	ENTER KEY
0005	ØØ1B	ESC	EQU	\$1B	ESCAPE CODE
0006	ØØØA	LF	EQU	\$A	LINE FEED
0007	ØØØC	FORMF	EQU	\$C	FORM FEED
0008	0020	SPACE	EQU	\$20	SPACE (BLANK)
	DULU	31 ACL	LQU	4 <i>L</i> 0	STAGE (BEARK)
0009					
0010	ØØ3A	STKBUF	EQU	58	STACK BUFFER ROOM
0011	Ø45E	DEBDEL	EQU	\$45E	DEBOUNCE DELAY
0012	ØØFA	LBUFMX	EQU	250	MAX NUMBER OF CHARS IN A BASIC LINE
0013	ØØFA	MAXLIN	EQU	\$FA	MAXIMUM MS BYTE OF LINE NUMBER
	DDIA	HAALIN	LQU	ΨIA	TAXTION IS SITE OF LIKE NOIDER
0014					
ØØ15	2600	DOSBUF	EQU	\$2600	RAM LOAD LOCATION FOR THE DOS COMMAND
0016	ØØ2Ø	DIRLEN	EQU	32	NUMBER OF BYTES IN DIRECTORY ENTRY
0017	0100	SECLEN	EQU	256	LENGTH OF SECTOR IN BYTES
ØØ18	0012	SECMAX	EQU	18	MAXIMUM NUMBER OF SECTORS PER TRACK
ØØ19	1200	TRKLEN	EQU	SECMAX*SECLEN	LENGTH OF TRACK IN BYTES
0020	0023	TRKMAX	EQU	35	MAX NUMBER OF TRACKS
0021	ØØ4A	FATLEN	EQU	6+(TRKMAX-1)*2	FILE ALLOCATION TABLE LENGTH
0022	0044	GRANMX	EQU	(TRKMAX-1)*2	MAXIMUM NUMBER OF GRANULES
ØØ23	Ø119	FCBLEN	EQU	SECLEN+25	FILE CONTROL BLOCK LENGTH
ØØ24	0010	INPFIL	EQU	\$10	INPUT FILE TYPE
	0020	OUTFIL			OUTPUT FILE TYPE
0025			EQU	\$20	
ØØ26	0040	RANFIL	EQU	\$40	RANDOM/DIRECT FILE TYPE
0027					
0028		* PSEUDO F	SEUD0	OPS	
ØØ29	0021	SKP1	EQU	\$21	OP CODE OF BRN SKIP ONE BYTE
0030	ØØ8C	SKP2	EQU	\$8C	OP CODE OF CMPX # - SKIP TWO BYTES
ØØ31	ØØ86	SKP1LD	EQU	\$86	OP CODE OF LDA # - SKIP THE NEXT BYTE
ØØ32		*			AND LOAD THE VALUE OF THAT BYTE INTO ACCA THIS
ØØ33		*			IS USUALLY USED TO LOAD ACCA WITH A NON ZERO VALUE
ØØ34					
ØØ35		* SIIDED EY	TENDE	D BASIC EQUATES	
	aa10			•	MAYIMUM NUMBER OF ROUG IN HI DEC RRINT MORE
ØØ36	ØØ18	ROWMAX	EQU	24	MAXIMUM NUMBER OF ROWS IN HI-RES PRINT MODE
ØØ37	ØØØØ	RAMLINK	EQU	Ø	DUMMY RAM LINK VECTOR
0038	2000	HRESSCRN	EQU	\$2000	ADDRESS OF THE HI-RES SCREEN IN THE CPU'S MEMORY SPACE
0038					
ØØ38 ØØ39	CØØØ	HRESBUFF	EQU	\$CØØØ	ADDRESS OF THE GET/PUT BUFFERS IN THE CPU'S MEMORY SPACE
ØØ38 ØØ39 ØØ4Ø	CØØØ DFFF	HRESBUFF TMPSTACK	EQU EQU	\$CØØØ \$DFFF	ADDRESS OF THE GET/PUT BUFFERS IN THE CPU'S MEMORY SPACE ADDRESS OF THE HI-RES GRAPHICS STACK IN THE CPU'S MEMORY SPACE
0038 0039 0040 0041	CØØØ DFFF ØØ62	HRESBUFF TMPSTACK EBHITOK	EQU EQU EQU	\$CØØØ \$DFFF \$62	ADDRESS OF THE GET/PUT BUFFERS IN THE CPU'S MEMORY SPACE ADDRESS OF THE HI-RES GRAPHICS STACK IN THE CPU'S MEMORY SPACE FIRST ENHANCED BASIC TOKEN NUMBER
0038 0039 0040 0041 0042	CØØØ DFFF ØØ62 ØØ29	HRESBUFF TMPSTACK EBHITOK EBHISTOK	EQU EQU EQU EQU	\$CØØØ \$DFFF \$62 \$29	ADDRESS OF THE GET/PUT BUFFERS IN THE CPU'S MEMORY SPACE ADDRESS OF THE HI-RES GRAPHICS STACK IN THE CPU'S MEMORY SPACE FIRST ENHANCED BASIC TOKEN NUMBER FIRST ENHANCED BASIC FUNCTION TOKEN NUMBER BUG - SHOULD BE \$28
0038 0039 0040 0041	CØØØ DFFF ØØ62	HRESBUFF TMPSTACK EBHITOK	EQU EQU EQU	\$CØØØ \$DFFF \$62	ADDRESS OF THE GET/PUT BUFFERS IN THE CPU'S MEMORY SPACE ADDRESS OF THE HI-RES GRAPHICS STACK IN THE CPU'S MEMORY SPACE FIRST ENHANCED BASIC TOKEN NUMBER
0038 0039 0040 0041 0042	CØØØ DFFF ØØ62 ØØ29	HRESBUFF TMPSTACK EBHITOK EBHISTOK	EQU EQU EQU EQU	\$CØØØ \$DFFF \$62 \$29	ADDRESS OF THE GET/PUT BUFFERS IN THE CPU'S MEMORY SPACE ADDRESS OF THE HI-RES GRAPHICS STACK IN THE CPU'S MEMORY SPACE FIRST ENHANCED BASIC TOKEN NUMBER FIRST ENHANCED BASIC FUNCTION TOKEN NUMBER BUG - SHOULD BE \$28
0038 0039 0040 0041 0042 0043 0044	CØØØ DFFF ØØ62 ØØ29	HRESBUFF TMPSTACK EBHITOK EBHISTOK CURCHAR	EQU EQU EQU EQU	\$CØØØ \$DFFF \$62 \$29 SPACE	ADDRESS OF THE GET/PUT BUFFERS IN THE CPU'S MEMORY SPACE ADDRESS OF THE HI-RES GRAPHICS STACK IN THE CPU'S MEMORY SPACE FIRST ENHANCED BASIC TOKEN NUMBER FIRST ENHANCED BASIC FUNCTION TOKEN NUMBER BUG - SHOULD BE \$28 HI-RES CURSOR CHARACTER
0038 0039 0040 0041 0042 0043 0044	CØØØ DFFF ØØ62 ØØ29 ØØ2Ø	HRESBUFF TMPSTACK EBHITOK EBHISTOK CURCHAR	EQU EQU EQU EQU EQU	\$CØØØ \$DFFF \$62 \$29 SPACE UT BUFFER HEADER	ADDRESS OF THE GET/PUT BUFFERS IN THE CPU'S MEMORY SPACE ADDRESS OF THE HI-RES GRAPHICS STACK IN THE CPU'S MEMORY SPACE FIRST ENHANCED BASIC TOKEN NUMBER FIRST ENHANCED BASIC FUNCTION TOKEN NUMBER BUG - SHOULD BE \$28 HI-RES CURSOR CHARACTER EQUATES
0038 0039 0040 0041 0042 0043 0044 0045	CØØØ DFFF ØØ62 ØØ29 ØØ2Ø	HRESBUFF TMPSTACK EBHITOK EBHISTOK CURCHAR * HBUFF HO HB.ADDR	EQU EQU EQU EQU EQU GET/HP	\$CØØØ \$DFFF \$62 \$29 SPACE UT BUFFER HEADER Ø	ADDRESS OF THE GET/PUT BUFFERS IN THE CPU'S MEMORY SPACE ADDRESS OF THE HI-RES GRAPHICS STACK IN THE CPU'S MEMORY SPACE FIRST ENHANCED BASIC TOKEN NUMBER FIRST ENHANCED BASIC FUNCTION TOKEN NUMBER BUG - SHOULD BE \$28 HI-RES CURSOR CHARACTER EQUATES ADDRESS OF THE NEXT BUFFER - 2 BYTES
0038 0039 0040 0041 0042 0043 0044 0045 0046 0047	CØØØ DFFF ØØ62 ØØ29 ØØ2Ø ØØØØ	HRESBUFF TMPSTACK EBHITOK EBHISTOK CURCHAR * HBUFF HO HB.ADDR HB.NUM	EQU EQU EQU EQU EQU GET/HP EQU EQU	\$CØØØ \$DFFF \$62 \$29 SPACE UT BUFFER HEADER Ø 2	ADDRESS OF THE GET/PUT BUFFERS IN THE CPU'S MEMORY SPACE ADDRESS OF THE HI-RES GRAPHICS STACK IN THE CPU'S MEMORY SPACE FIRST ENHANCED BASIC TOKEN NUMBER FIRST ENHANCED BASIC FUNCTION TOKEN NUMBER BUG - SHOULD BE \$28 HI-RES CURSOR CHARACTER EQUATES ADDRESS OF THE NEXT BUFFER - 2 BYTES NUMBER OF THIS BUFFER - 1 BYTES
0038 0039 0040 0041 0042 0043 0044 0045 0046 0047 0048	C000 DFFF 0062 0029 0020 0000 0002	HRESBUFF TMPSTACK EBHITOK EBHISTOK CURCHAR * HBUFF HO HB.ADDR HB.NUM HB.SIZE	EQU EQU EQU EQU EQU EQU EQU EQU	\$CØØØ \$DFFF \$62 \$29 SPACE UT BUFFER HEADER Ø 2	ADDRESS OF THE GET/PUT BUFFERS IN THE CPU'S MEMORY SPACE ADDRESS OF THE HI-RES GRAPHICS STACK IN THE CPU'S MEMORY SPACE FIRST ENHANCED BASIC TOKEN NUMBER FIRST ENHANCED BASIC FUNCTION TOKEN NUMBER BUG - SHOULD BE \$28 HI-RES CURSOR CHARACTER EQUATES ADDRESS OF THE NEXT BUFFER - 2 BYTES NUMBER OF THIS BUFFER - 1 BYTES NUMBER OF BYTES IN THE BUFFER - 2 BYTES
0038 0039 0040 0041 0042 0043 0044 0045 0046 0047	CØØØ DFFF ØØ62 ØØ29 ØØ2Ø ØØØØ	HRESBUFF TMPSTACK EBHITOK EBHISTOK CURCHAR * HBUFF HO HB.ADDR HB.NUM	EQU EQU EQU EQU EQU GET/HP EQU EQU	\$CØØØ \$DFFF \$62 \$29 SPACE UT BUFFER HEADER Ø 2	ADDRESS OF THE GET/PUT BUFFERS IN THE CPU'S MEMORY SPACE ADDRESS OF THE HI-RES GRAPHICS STACK IN THE CPU'S MEMORY SPACE FIRST ENHANCED BASIC TOKEN NUMBER FIRST ENHANCED BASIC FUNCTION TOKEN NUMBER BUG - SHOULD BE \$28 HI-RES CURSOR CHARACTER EQUATES ADDRESS OF THE NEXT BUFFER - 2 BYTES NUMBER OF THIS BUFFER - 1 BYTES
0038 0039 0040 0041 0042 0043 0044 0045 0046 0047 0048	C000 DFFF 0062 0029 0020 0000 0002	HRESBUFF TMPSTACK EBHITOK EBHISTOK CURCHAR * HBUFF HO HB.ADDR HB.NUM HB.SIZE	EQU EQU EQU EQU EQU EQU EQU EQU	\$CØØØ \$DFFF \$62 \$29 SPACE UT BUFFER HEADER Ø 2	ADDRESS OF THE GET/PUT BUFFERS IN THE CPU'S MEMORY SPACE ADDRESS OF THE HI-RES GRAPHICS STACK IN THE CPU'S MEMORY SPACE FIRST ENHANCED BASIC TOKEN NUMBER FIRST ENHANCED BASIC FUNCTION TOKEN NUMBER BUG - SHOULD BE \$28 HI-RES CURSOR CHARACTER EQUATES ADDRESS OF THE NEXT BUFFER - 2 BYTES NUMBER OF THIS BUFFER - 1 BYTES NUMBER OF BYTES IN THE BUFFER - 2 BYTES
0038 0039 0040 0041 0042 0043 0044 0045 0046 0047 0048 0049 0050	C000 DFFF 0062 0029 0020 0000 0002	HRESBUFF TMPSTACK EBHITOK EBHISTOK CURCHAR * HBUFF HO HB.ADDR HB.NUM HB.SIZE HB.LEN	EQU EQU EQU EQU GET/HP EQU EQU EQU EQU	\$CØØØ \$DFFF \$62 \$29 SPACE UT BUFFER HEADER Ø 2 3	ADDRESS OF THE GET/PUT BUFFERS IN THE CPU'S MEMORY SPACE ADDRESS OF THE HI-RES GRAPHICS STACK IN THE CPU'S MEMORY SPACE FIRST ENHANCED BASIC TOKEN NUMBER FIRST ENHANCED BASIC FUNCTION TOKEN NUMBER BUG - SHOULD BE \$28 HI-RES CURSOR CHARACTER EQUATES ADDRESS OF THE NEXT BUFFER - 2 BYTES NUMBER OF THIS BUFFER - 1 BYTES NUMBER OF BYTES IN THE BUFFER - 2 BYTES
0038 0039 0040 0041 0042 0043 0044 0045 0046 0047 0048 0049 0049 0050	C000 DFFF 0062 0029 0020 0000 0002	HRESBUFF TMPSTACK EBHITOK EBHISTOK CURCHAR * HBUFF HO HB.ADDR HB.NUM HB.SIZE HB.LEN * VIDEO RE	EQU EQU EQU EQU GET/HP EQU EQU EQU EQU	\$CØØØ \$DFFF \$62 \$29 SPACE UT BUFFER HEADER Ø 2 3 5	ADDRESS OF THE GET/PUT BUFFERS IN THE CPU'S MEMORY SPACE ADDRESS OF THE HI-RES GRAPHICS STACK IN THE CPU'S MEMORY SPACE FIRST ENHANCED BASIC TOKEN NUMBER FIRST ENHANCED BASIC FUNCTION TOKEN NUMBER BUG - SHOULD BE \$28 HI-RES CURSOR CHARACTER EQUATES ADDRESS OF THE NEXT BUFFER - 2 BYTES NUMBER OF THIS BUFFER - 1 BYTES NUMBER OF BYTES IN THE BUFFER - 2 BYTES
0038 0039 0040 0041 0042 0043 0044 0045 0046 0047 0048 0049 0050 0051	C000 DFFF 0062 0029 0020 0000 0002 0003 0005	HRESBUFF TMPSTACK EBHITOK EBHISTOK CURCHAR * HBUFF HO HB.ADDR HB.NUM HB.SIZE HB.LEN * VIDEO RI * INITØ BI	EQU	\$CØØØ \$DFFF \$62 \$29 SPACE UT BUFFER HEADER Ø 2 3 5 R EQUATES ATES	ADDRESS OF THE GET/PUT BUFFERS IN THE CPU'S MEMORY SPACE ADDRESS OF THE HI-RES GRAPHICS STACK IN THE CPU'S MEMORY SPACE FIRST ENHANCED BASIC TOKEN NUMBER FIRST ENHANCED BASIC FUNCTION TOKEN NUMBER BUG - SHOULD BE \$28 HI-RES CURSOR CHARACTER EQUATES ADDRESS OF THE NEXT BUFFER - 2 BYTES NUMBER OF THIS BUFFER - 1 BYTES NUMBER OF BYTES IN THE BUFFER - 2 BYTES NUMBER OF BYTES IN THIS HEADER
0038 0039 0040 0041 0042 0043 0044 0045 0046 0047 0048 0049 0050 0051 0052	C000 DFFF 0062 0029 0020 0000 0002 0003 0005	HRESBUFF TMPSTACK EBHITOK EBHISTOK CURCHAR * HBUFF HO HB.ADDR HB.NUM HB.SIZE HB.LEN * VIDEO RE * INITØ BE COCO	EQU	\$CØØØ \$DFFF \$62 \$29 SPACE UT BUFFER HEADER Ø 2 3 5 R EQUATES ATES \$8Ø	ADDRESS OF THE GET/PUT BUFFERS IN THE CPU'S MEMORY SPACE ADDRESS OF THE HI-RES GRAPHICS STACK IN THE CPU'S MEMORY SPACE FIRST ENHANCED BASIC TOKEN NUMBER FIRST ENHANCED BASIC FUNCTION TOKEN NUMBER BUG - SHOULD BE \$28 HI-RES CURSOR CHARACTER EQUATES ADDRESS OF THE NEXT BUFFER - 2 BYTES NUMBER OF THIS BUFFER - 1 BYTES NUMBER OF BYTES IN THE BUFFER - 2 BYTES NUMBER OF BYTES IN THIS HEADER 1 = Color Computer compatible
0038 0039 0040 0041 0042 0043 0044 0045 0046 0047 0048 0049 0050 0051 0052 0053 0054	C000 DFFF 0062 0029 0020 0000 0002 0003 0005	HRESBUFF TMPSTACK EBHITOK EBHISTOK CURCHAR * HBUFF HC HB.ADDR HB.NUM HB.SIZE HB.LEN * VIDEO RE * INITØ BE COCO MMUEN	EQU EQU EQU EQU GET/HP EQU EQU EQU EQU EQU EQU EQU EQU EQU EQU	\$CØØØ \$DFFF \$62 \$29 SPACE UT BUFFER HEADER Ø 2 3 5 R EQUATES ATES \$8Ø \$4Ø	ADDRESS OF THE GET/PUT BUFFERS IN THE CPU'S MEMORY SPACE ADDRESS OF THE HI-RES GRAPHICS STACK IN THE CPU'S MEMORY SPACE FIRST ENHANCED BASIC TOKEN NUMBER FIRST ENHANCED BASIC FUNCTION TOKEN NUMBER BUG - SHOULD BE \$28 HI-RES CURSOR CHARACTER EQUATES ADDRESS OF THE NEXT BUFFER - 2 BYTES NUMBER OF THIS BUFFER - 1 BYTES NUMBER OF BYTES IN THE BUFFER - 2 BYTES NUMBER OF BYTES IN THIS HEADER 1 = Color Computer compatible 1 = MMU enabled
0038 0039 0040 0041 0042 0043 0044 0045 0046 0047 0048 0049 0050 0051 0052	C000 DFFF 0062 0029 0020 0000 0002 0003 0005	HRESBUFF TMPSTACK EBHITOK EBHISTOK CURCHAR * HBUFF HO HB.ADDR HB.NUM HB.SIZE HB.LEN * VIDEO RE * INITØ BE COCO	EQU	\$CØØØ \$DFFF \$62 \$29 SPACE UT BUFFER HEADER Ø 2 3 5 R EQUATES ATES \$8Ø	ADDRESS OF THE GET/PUT BUFFERS IN THE CPU'S MEMORY SPACE ADDRESS OF THE HI-RES GRAPHICS STACK IN THE CPU'S MEMORY SPACE FIRST ENHANCED BASIC TOKEN NUMBER FIRST ENHANCED BASIC FUNCTION TOKEN NUMBER BUG - SHOULD BE \$28 HI-RES CURSOR CHARACTER EQUATES ADDRESS OF THE NEXT BUFFER - 2 BYTES NUMBER OF THIS BUFFER - 1 BYTES NUMBER OF BYTES IN THE BUFFER - 2 BYTES NUMBER OF BYTES IN THIS HEADER 1 = Color Computer compatible
0038 0039 0040 0041 0042 0043 0044 0045 0046 0047 0048 0049 0050 0051 0052 0053 0054	C000 DFFF 0062 0029 0020 0000 0002 0003 0005	HRESBUFF TMPSTACK EBHITOK EBHISTOK CURCHAR * HBUFF HC HB.ADDR HB.NUM HB.SIZE HB.LEN * VIDEO RE * INITØ BE COCO MMUEN	EQU EQU EQU EQU GET/HP EQU EQU EQU EQU EQU EQU EQU EQU EQU EQU	\$CØØØ \$DFFF \$62 \$29 SPACE UT BUFFER HEADER Ø 2 3 5 R EQUATES ATES \$8Ø \$4Ø	ADDRESS OF THE GET/PUT BUFFERS IN THE CPU'S MEMORY SPACE ADDRESS OF THE HI-RES GRAPHICS STACK IN THE CPU'S MEMORY SPACE FIRST ENHANCED BASIC TOKEN NUMBER FIRST ENHANCED BASIC FUNCTION TOKEN NUMBER BUG - SHOULD BE \$28 HI-RES CURSOR CHARACTER EQUATES ADDRESS OF THE NEXT BUFFER - 2 BYTES NUMBER OF THIS BUFFER - 1 BYTES NUMBER OF BYTES IN THE BUFFER - 2 BYTES NUMBER OF BYTES IN THIS HEADER 1 = Color Computer compatible 1 = MMU enabled
0038 0039 0040 0041 0042 0043 0044 0045 0046 0047 0048 0049 0050 0051 0052 0053 0054 0055	C000 DFFF 0062 0029 0020 0002 0003 0005 0080 0040 0020 0010	HRESBUFF TMPSTACK EBHITOK EBHISTOK CURCHAR * HBUFF HO HB.ADDR HB.NUM HB.SIZE HB.LEN * VIDEO RE * INITØ BE COCO MMUEN IEN FEN	EQU	\$CØØØ \$DFFF \$62 \$29 \$PACE UT BUFFER HEADER Ø 2 3 5 R EQUATES ATES \$8Ø \$4Ø \$2Ø \$1Ø	ADDRESS OF THE GET/PUT BUFFERS IN THE CPU'S MEMORY SPACE ADDRESS OF THE HI-RES GRAPHICS STACK IN THE CPU'S MEMORY SPACE FIRST ENHANCED BASIC TOKEN NUMBER FIRST ENHANCED BASIC FUNCTION TOKEN NUMBER BUG - SHOULD BE \$28 HI-RES CURSOR CHARACTER EQUATES ADDRESS OF THE NEXT BUFFER - 2 BYTES NUMBER OF THIS BUFFER - 1 BYTES NUMBER OF BYTES IN THE BUFFER - 2 BYTES NUMBER OF BYTES IN THIS HEADER 1 = Color Computer compatible 1 = MMU enabled 1 = GIME chip IRQ output enabled 1 = GIME chip FIRQ output enabled
0038 0039 0040 0041 0042 0043 0044 0045 0046 0047 0048 0049 0050 0051 0052 0053 0054 0055 0056 0057	C000 DFFF 0062 0029 0020 0000 0002 0003 0005	HRESBUFF TMPSTACK EBHITOK EBHISTOK CURCHAR * HBUFF HO HB.ADDR HB.NUM HB.SIZE HB.LEN * VIDEO RI * INITØ BI COCO MMUEN IEN FEN MC3	EQU	\$CØØØ \$DFFF \$62 \$29 SPACE UT BUFFER HEADER Ø 2 3 5 R EQUATES ATES \$8Ø \$4Ø \$1Ø 8	ADDRESS OF THE GET/PUT BUFFERS IN THE CPU'S MEMORY SPACE ADDRESS OF THE HI-RES GRAPHICS STACK IN THE CPU'S MEMORY SPACE FIRST ENHANCED BASIC TOKEN NUMBER FIRST ENHANCED BASIC FUNCTION TOKEN NUMBER BUG - SHOULD BE \$28 HI-RES CURSOR CHARACTER EQUATES ADDRESS OF THE NEXT BUFFER - 2 BYTES NUMBER OF THIS BUFFER - 1 BYTES NUMBER OF BYTES IN THE BUFFER - 2 BYTES NUMBER OF BYTES IN THIS HEADER 1 = Color Computer compatible 1 = MMU enabled 1 = GIME chip IRQ output enabled 1 = GIME chip FIRQ output enabled 1 = RAM at XFEXX is constant
0038 0039 0040 0041 0042 0043 0044 0045 0046 0047 0048 0049 0050 0051 0052 0053 0054 0055 0057 0058	C000 DFFF 0062 0029 0020 0000 0002 0003 0005	HRESBUFF TMPSTACK EBHITOK EBHISTOK CURCHAR * HBUFF HO HB.ADDR HB.NUM HB.SIZE HB.LEN * VIDEO RE * INITØ BE COCO MMUEN IEN FEN MC3 MC2	EQU EQU EQU EQU EQU EQU EQU EQU EQU EQU	\$CØØØ \$DFFF \$62 \$29 SPACE UT BUFFER HEADER Ø 2 3 5 R EQUATES ATES \$8Ø \$4Ø \$2Ø \$1Ø 8	ADDRESS OF THE GET/PUT BUFFERS IN THE CPU'S MEMORY SPACE ADDRESS OF THE HI-RES GRAPHICS STACK IN THE CPU'S MEMORY SPACE FIRST ENHANCED BASIC TOKEN NUMBER FIRST ENHANCED BASIC FUNCTION TOKEN NUMBER BUG - SHOULD BE \$28 HI-RES CURSOR CHARACTER EQUATES ADDRESS OF THE NEXT BUFFER - 2 BYTES NUMBER OF THIS BUFFER - 1 BYTES NUMBER OF BYTES IN THE BUFFER - 2 BYTES NUMBER OF BYTES IN THIS HEADER 1 = Color Computer compatible 1 = MMU enabled 1 = GIME chip IRQ output enabled 1 = GIME chip FIRQ output enabled 1 = RAM at XFEXX is constant 1 = standard SCS
0038 0039 0040 0041 0042 00443 00445 00446 0045 0046 0047 0050 0051 0052 0053 0054 0055 0056 0057 0058 0059	C000 DFFF 0062 0029 0020 0000 0002 0003 0005	HRESBUFF TMPSTACK EBHITOK EBHISTOK CURCHAR * HBUFF HO HB.ADDR HB.NUM HB.SIZE HB.LEN * VIDEO RI * INITØ BI COCO MMUEN IEN FEN MC3 MC2 MC1	EQU EQU EQU EQU EQU EQU EQU EQU EQU EQU	\$CØØØ \$DFFF \$62 \$29 SPACE UT BUFFER HEADER Ø 2 3 5 R EQUATES ATES \$8Ø \$4Ø \$2Ø \$1Ø 8 4	ADDRESS OF THE GET/PUT BUFFERS IN THE CPU'S MEMORY SPACE ADDRESS OF THE HI-RES GRAPHICS STACK IN THE CPU'S MEMORY SPACE FIRST ENHANCED BASIC TOKEN NUMBER FIRST ENHANCED BASIC FUNCTION TOKEN NUMBER BUG - SHOULD BE \$28 HI-RES CURSOR CHARACTER EQUATES ADDRESS OF THE NEXT BUFFER - 2 BYTES NUMBER OF THIS BUFFER - 1 BYTES NUMBER OF BYTES IN THE BUFFER - 2 BYTES NUMBER OF BYTES IN THIS HEADER 1 = Color Computer compatible 1 = MMU enabled 1 = GIME chip FIRQ output enabled 1 = GIME chip FIRQ output enabled 1 = RAM at XFEXX is constant 1 = standard SCS ROM map control
0038 0039 0040 0041 0042 0043 0044 0045 0046 0047 0048 0049 0050 0051 0052 0053 0054 0055 0057 0058	C000 DFFF 0062 0029 0020 0000 0002 0003 0005	HRESBUFF TMPSTACK EBHITOK EBHISTOK CURCHAR * HBUFF HO HB.ADDR HB.NUM HB.SIZE HB.LEN * VIDEO RE * INITØ BE COCO MMUEN IEN FEN MC3 MC2	EQU EQU EQU EQU EQU EQU EQU EQU EQU EQU	\$CØØØ \$DFFF \$62 \$29 SPACE UT BUFFER HEADER Ø 2 3 5 R EQUATES ATES \$8Ø \$4Ø \$2Ø \$1Ø 8	ADDRESS OF THE GET/PUT BUFFERS IN THE CPU'S MEMORY SPACE ADDRESS OF THE HI-RES GRAPHICS STACK IN THE CPU'S MEMORY SPACE FIRST ENHANCED BASIC TOKEN NUMBER FIRST ENHANCED BASIC FUNCTION TOKEN NUMBER BUG - SHOULD BE \$28 HI-RES CURSOR CHARACTER EQUATES ADDRESS OF THE NEXT BUFFER - 2 BYTES NUMBER OF THIS BUFFER - 1 BYTES NUMBER OF BYTES IN THE BUFFER - 2 BYTES NUMBER OF BYTES IN THIS HEADER 1 = Color Computer compatible 1 = MMU enabled 1 = GIME chip IRQ output enabled 1 = GIME chip FIRQ output enabled 1 = RAM at XFEXX is constant 1 = standard SCS
0038 0039 0040 0041 0042 00443 00445 00446 0045 0046 0047 0050 0051 0052 0053 0054 0055 0056 0057 0058 0059	C000 DFFF 0062 0029 0020 0000 0002 0003 0005	HRESBUFF TMPSTACK EBHITOK EBHISTOK CURCHAR * HBUFF HO HB.ADDR HB.NUM HB.SIZE HB.LEN * VIDEO RI * INITØ BI COCO MMUEN IEN FEN MC3 MC2 MC1	EQU EQU EQU EQU EQU EQU EQU EQU EQU EQU	\$CØØØ \$DFFF \$62 \$29 SPACE UT BUFFER HEADER Ø 2 3 5 R EQUATES ATES \$8Ø \$4Ø \$2Ø \$1Ø 8 4	ADDRESS OF THE GET/PUT BUFFERS IN THE CPU'S MEMORY SPACE ADDRESS OF THE HI-RES GRAPHICS STACK IN THE CPU'S MEMORY SPACE FIRST ENHANCED BASIC TOKEN NUMBER FIRST ENHANCED BASIC FUNCTION TOKEN NUMBER BUG - SHOULD BE \$28 HI-RES CURSOR CHARACTER EQUATES ADDRESS OF THE NEXT BUFFER - 2 BYTES NUMBER OF THIS BUFFER - 1 BYTES NUMBER OF BYTES IN THE BUFFER - 2 BYTES NUMBER OF BYTES IN THIS HEADER 1 = Color Computer compatible 1 = MMU enabled 1 = GIME chip FIRQ output enabled 1 = GIME chip FIRQ output enabled 1 = RAM at XFEXX is constant 1 = standard SCS ROM map control
0038 0039 0040 0041 0042 00443 0044 0045 0046 0047 0048 0049 0050 0051 0052 0053 0054 0055 0056 0057 0058 0059 0060	C000 DFFF 0062 0029 0020 0000 0002 0003 0005	HRESBUFF TMPSTACK EBHITOK EBHISTOK CURCHAR * HBUFF HC HB.ADDR HB.NUM HB.SIZE HB.LEN * VIDEO RE * INITØ BE COCO MMUEN IEN FEN MC3 MC2 MC1 MCØ	EQU EQU EQU EQU EQU EQU EQU EQU EQU EQU	\$CØØØ \$DFFF \$62 \$29 SPACE UT BUFFER HEADER Ø 2 3 5 R EQUATES ATES \$8Ø \$4Ø \$2Ø \$1Ø 8 4	ADDRESS OF THE GET/PUT BUFFERS IN THE CPU'S MEMORY SPACE ADDRESS OF THE HI-RES GRAPHICS STACK IN THE CPU'S MEMORY SPACE FIRST ENHANCED BASIC TOKEN NUMBER FIRST ENHANCED BASIC FUNCTION TOKEN NUMBER BUG - SHOULD BE \$28 HI-RES CURSOR CHARACTER EQUATES ADDRESS OF THE NEXT BUFFER - 2 BYTES NUMBER OF THIS BUFFER - 1 BYTES NUMBER OF BYTES IN THE BUFFER - 2 BYTES NUMBER OF BYTES IN THIS HEADER 1 = Color Computer compatible 1 = MMU enabled 1 = GIME chip FIRQ output enabled 1 = GIME chip FIRQ output enabled 1 = RAM at XFEXX is constant 1 = standard SCS ROM map control
0038 0039 0040 0041 0042 0043 0044 0045 0047 0048 0049 0050 0051 0052 0053 0054 0055 0056 0057 0059 0060 0061 0062	C000 DFFF 0062 0029 0020 0002 0003 0005 0080 0040 0020 0010 0008 0004 0008 0004 0002 0001	HRESBUFF TMPSTACK EBHITOK EBHISTOK CURCHAR * HBUFF HO HB.ADDR HB.NUM HB.SIZE HB.LEN * VIDEO RE * INITØ BE COCO MMUEN IEN FEN MC3 MC2 MC1 MCØ * INTERRUE	EQU	\$CØØØ \$DFFF \$62 \$29 \$PACE UT BUFFER HEADER Ø 2 3 5 R EQUATES ATES \$8Ø \$4Ø \$1Ø 8 4 2 1 UEST ENABLED	ADDRESS OF THE GET/PUT BUFFERS IN THE CPU'S MEMORY SPACE ADDRESS OF THE HI-RES GRAPHICS STACK IN THE CPU'S MEMORY SPACE FIRST ENHANCED BASIC TOKEN NUMBER FIRST ENHANCED BASIC FUNCTION TOKEN NUMBER BUG - SHOULD BE \$28 HI-RES CURSOR CHARACTER EQUATES ADDRESS OF THE NEXT BUFFER - 2 BYTES NUMBER OF THIS BUFFER - 1 BYTES NUMBER OF BYTES IN THE BUFFER - 2 BYTES NUMBER OF BYTES IN THIS HEADER 1 = Color Computer compatible 1 = MMU enabled 1 = GIME chip IRQ output enabled 1 = GIME chip FIRQ output enabled 1 = RAM at XFEXX is constant 1 = standard SCS ROM map control ROM map control
0038 0039 0040 0041 0042 0043 0044 0045 0046 0047 0048 0049 0051 0052 0053 0055 0055 0056 0057 0058 0059 0060 0061 0062 0063	C000 DFFF 0062 0029 0020 0000 0002 0003 0005 0080 0040 0010 0008 0010 0008 0004 0002 0001	HRESBUFF TMPSTACK EBHITOK EBHISTOK CURCHAR * HBUFF HO HB.ADDR HB.NUM HB.SIZE HB.LEN * VIDEO RI * INITØ BI COCO MMUEN IEN FEN MC3 MC2 MC1 MCØ * INTERRUITMR	EQU	\$CØØØ \$DFFF \$62 \$29 \$PACE UT BUFFER HEADER Ø 2 3 5 R EQUATES ATES \$8Ø \$40 \$10 8 4 2 1 UEST ENABLED \$2Ø	ADDRESS OF THE GET/PUT BUFFERS IN THE CPU'S MEMORY SPACE ADDRESS OF THE HI-RES GRAPHICS STACK IN THE CPU'S MEMORY SPACE FIRST ENHANCED BASIC TOKEN NUMBER FIRST ENHANCED BASIC FUNCTION TOKEN NUMBER BUG - SHOULD BE \$28 HI-RES CURSOR CHARACTER EQUATES ADDRESS OF THE NEXT BUFFER - 2 BYTES NUMBER OF THIS BUFFER - 1 BYTES NUMBER OF BYTES IN THE BUFFER - 2 BYTES NUMBER OF BYTES IN THIS HEADER 1 = Color Computer compatible 1 = MMU enabled 1 = GIME chip FIRQ output enabled 1 = GIME chip FIRQ output enabled 1 = RAM at XFEXX is constant 1 = standard SCS ROM map control ROM map control
0038 0039 0040 0041 0042 0043 0044 0045 0046 0047 0050 0051 0052 0053 0055 0055 0055 0057 0058 0059 0060 0061 0062 0063 0064	C000 DFFF 0062 0029 0020 0000 0002 0003 0005 0080 0040 0040 0010 0008 0004 0002 0001	HRESBUFF TMPSTACK EBHITOK EBHISTOK CURCHAR * HBUFF HO HB.ADDR HB.NUM HB.SIZE HB.LEN * VIDEO RE * INITØ BE COCO MMUEN IEN FEN MC3 MC2 MC1 MCØ * INTERRUE TMR HBORD	EQU	\$CØØØ \$DFFF \$62 \$29 \$PACE UT BUFFER HEADER Ø 2 3 5 R EQUATES ATES \$8Ø \$4Ø \$2Ø \$1Ø 8 4 2 1 UEST ENABLED \$2Ø \$1Ø	ADDRESS OF THE GET/PUT BUFFERS IN THE CPU'S MEMORY SPACE ADDRESS OF THE HI-RES GRAPHICS STACK IN THE CPU'S MEMORY SPACE FIRST ENHANCED BASIC TOKEN NUMBER FIRST ENHANCED BASIC FUNCTION TOKEN NUMBER BUG - SHOULD BE \$28 HI-RES CURSOR CHARACTER EQUATES ADDRESS OF THE NEXT BUFFER - 2 BYTES NUMBER OF THIS BUFFER - 1 BYTES NUMBER OF BYTES IN THE BUFFER - 2 BYTES NUMBER OF BYTES IN THIS HEADER 1 = Color Computer compatible 1 = MMU enabled 1 = GIME chip IRQ output enabled 1 = GIME chip FIRQ output enabled 1 = RAM at XFEXX is constant 1 = standard SCS ROM map control ROM map control TIMER HORIZONTAL BORDER
0038 0039 0040 0041 0042 00443 00446 0045 0046 0047 0048 0049 0051 0052 0053 0055 0055 0057 0058 0059 0060 0061 0062 0064 0065	C000 DFFF 0062 0029 0020 0000 0002 0003 0005 0040 0040 0010 0008 0004 0002 0001	HRESBUFF TMPSTACK EBHITOK EBHISTOK CURCHAR * HBUFF HO HB.ADDR HB.NUM HB.SIZE HB.LEN * VIDEO RI * INITØ BI COCO MMUEN IEN FEN MC3 MC2 MC1 MCØ * INTERRUF TMR HBORD VBORD	EQU	\$CØØØ \$DFFF \$62 \$29 SPACE UT BUFFER HEADER Ø 2 3 5 R EQUATES ATES \$8Ø \$40 \$2Ø \$1Ø 8 4 2 1	ADDRESS OF THE GET/PUT BUFFERS IN THE CPU'S MEMORY SPACE ADDRESS OF THE HI-RES GRAPHICS STACK IN THE CPU'S MEMORY SPACE FIRST ENHANCED BASIC TOKEN NUMBER FIRST ENHANCED BASIC FUNCTION TOKEN NUMBER BUG - SHOULD BE \$28 HI-RES CURSOR CHARACTER EQUATES ADDRESS OF THE NEXT BUFFER - 2 BYTES NUMBER OF THIS BUFFER - 1 BYTES NUMBER OF BYTES IN THE BUFFER - 2 BYTES NUMBER OF BYTES IN THIS HEADER 1 = Color Computer compatible 1 = MMU enabled 1 = GIME chip FIRQ output enabled 1 = GIME chip FIRQ output enabled 1 = RAM at XFEXX is constant 1 = standard SCS ROM map control ROM map control TIMER HORIZONTAL BORDER VERTICAL BORDER
0038 0039 0040 0041 0042 0043 0044 0045 0046 0047 0050 0051 0052 0053 0055 0055 0055 0057 0058 0059 0060 0061 0062 0063 0064	C000 DFFF 0062 0029 0020 0000 0002 0003 0005 0080 0040 0040 0010 0008 0004 0002 0001	HRESBUFF TMPSTACK EBHITOK EBHISTOK CURCHAR * HBUFF HO HB.ADDR HB.NUM HB.SIZE HB.LEN * VIDEO RE * INITØ BE COCO MMUEN IEN FEN MC3 MC2 MC1 MCØ * INTERRUE TMR HBORD	EQU	\$CØØØ \$DFFF \$62 \$29 \$PACE UT BUFFER HEADER Ø 2 3 5 R EQUATES ATES \$8Ø \$4Ø \$2Ø \$1Ø 8 4 2 1 UEST ENABLED \$2Ø \$1Ø	ADDRESS OF THE GET/PUT BUFFERS IN THE CPU'S MEMORY SPACE ADDRESS OF THE HI-RES GRAPHICS STACK IN THE CPU'S MEMORY SPACE FIRST ENHANCED BASIC TOKEN NUMBER FIRST ENHANCED BASIC FUNCTION TOKEN NUMBER BUG - SHOULD BE \$28 HI-RES CURSOR CHARACTER EQUATES ADDRESS OF THE NEXT BUFFER - 2 BYTES NUMBER OF THIS BUFFER - 1 BYTES NUMBER OF BYTES IN THE BUFFER - 2 BYTES NUMBER OF BYTES IN THIS HEADER 1 = Color Computer compatible 1 = MMU enabled 1 = GIME chip IRQ output enabled 1 = GIME chip FIRQ output enabled 1 = RAM at XFEXX is constant 1 = standard SCS ROM map control ROM map control TIMER HORIZONTAL BORDER
0038 0039 0040 0041 0042 00443 00446 0045 0046 0047 0048 0049 0051 0052 0053 0055 0055 0057 0058 0059 0060 0061 0062 0064 0065	C000 DFFF 0062 0029 0020 0000 0002 0003 0005 0040 0040 0010 0008 0004 0002 0001	HRESBUFF TMPSTACK EBHITOK EBHISTOK CURCHAR * HBUFF HO HB.ADDR HB.NUM HB.SIZE HB.LEN * VIDEO RI * INITØ BI COCO MMUEN IEN FEN MC3 MC2 MC1 MCØ * INTERRUF TMR HBORD VBORD	EQU	\$CØØØ \$DFFF \$62 \$29 SPACE UT BUFFER HEADER Ø 2 3 5 R EQUATES ATES \$8Ø \$40 \$2Ø \$1Ø 8 4 2 1	ADDRESS OF THE GET/PUT BUFFERS IN THE CPU'S MEMORY SPACE ADDRESS OF THE HI-RES GRAPHICS STACK IN THE CPU'S MEMORY SPACE FIRST ENHANCED BASIC TOKEN NUMBER FIRST ENHANCED BASIC FUNCTION TOKEN NUMBER BUG - SHOULD BE \$28 HI-RES CURSOR CHARACTER EQUATES ADDRESS OF THE NEXT BUFFER - 2 BYTES NUMBER OF THIS BUFFER - 1 BYTES NUMBER OF BYTES IN THE BUFFER - 2 BYTES NUMBER OF BYTES IN THIS HEADER 1 = Color Computer compatible 1 = MMU enabled 1 = GIME chip FIRQ output enabled 1 = GIME chip FIRQ output enabled 1 = RAM at XFEXX is constant 1 = standard SCS ROM map control ROM map control TIMER HORIZONTAL BORDER VERTICAL BORDER
0038 0039 0040 0041 0042 0043 0044 0045 0047 0047 0048 0049 0050 0051 0053 0055 0055 0056 0057	0000 0000	HRESBUFF TMPSTACK EBHITOK EBHISTOK CURCHAR * HBUFF HC HB.ADDR HB.NUM HB.SIZE HB.LEN * VIDEO RE * INITØ BE COCO MMUEN IEN FEN MC3 MC2 MC1 MCØ * INTERRUE TMR HBORD VBORD EI2 EI1	EQU	\$CØØØ \$DFFF \$62 \$29 SPACE UT BUFFER HEADER Ø 2 3 5 R EQUATES ATES \$8Ø \$4Ø \$2Ø \$1Ø 8 4 2 1	ADDRESS OF THE GET/PUT BUFFERS IN THE CPU'S MEMORY SPACE ADDRESS OF THE HI-RES GRAPHICS STACK IN THE CPU'S MEMORY SPACE FIRST ENHANCED BASIC TOKEN NUMBER FIRST ENHANCED BASIC FUNCTION TOKEN NUMBER BUG - SHOULD BE \$28 HI-RES CURSOR CHARACTER EQUATES ADDRESS OF THE NEXT BUFFER - 2 BYTES NUMBER OF THIS BUFFER - 1 BYTES NUMBER OF BYTES IN THE BUFFER - 2 BYTES NUMBER OF BYTES IN THIS HEADER 1 = Color Computer compatible 1 = MMU enabled 1 = GIME chip IRQ output enabled 1 = GIME chip FIRQ output enabled 1 = RAM at XFEXX is constant 1 = standard SCS ROM map control ROM map control TIMER HORIZONTAL BORDER VERTICAL BORDER SERIAL DATA KEYBOARD
0038 0039 0040 0041 0042 0043 0044 0045 0047 0048 0050 0051 0053 0054 0055 0056 0057 0068 0061 0062 0063 0066 0067 0068	C000 DFFF 0062 0029 0020 0000 0002 0003 0005 0080 0040 0010 0008 0004 0002 0001	HRESBUFF TMPSTACK EBHITOK EBHISTOK CURCHAR * HBUFF HC HB.ADDR HB.NUM HB.SIZE HB.LEN * VIDEO RE * INITØ BE COCO MMUEN IEN FEN MC3 MC2 MC1 MCØ * INTERRUE TMR HBORD VBORD EI2	EQU	\$CØØØ \$DFFF \$62 \$29 SPACE UT BUFFER HEADER Ø 2 3 5 R EQUATES ATES \$8Ø \$4Ø \$2Ø \$1Ø 8 4 2 1	ADDRESS OF THE GET/PUT BUFFERS IN THE CPU'S MEMORY SPACE ADDRESS OF THE HI-RES GRAPHICS STACK IN THE CPU'S MEMORY SPACE FIRST ENHANCED BASIC TOKEN NUMBER FIRST ENHANCED BASIC FUNCTION TOKEN NUMBER BUG - SHOULD BE \$28 HI-RES CURSOR CHARACTER EQUATES ADDRESS OF THE NEXT BUFFER - 2 BYTES NUMBER OF THIS BUFFER - 1 BYTES NUMBER OF BYTES IN THE BUFFER - 2 BYTES NUMBER OF BYTES IN THIS HEADER 1 = Color Computer compatible 1 = MMU enabled 1 = GIME chip FIRQ output enabled 1 = GIME chip FIRQ output enabled 1 = RAM at XFEXX is constant 1 = standard SCS ROM map control ROM map control TIMER HORIZONTAL BORDER VERTICAL BORDER SERIAL DATA
0038 0039 0040 0041 0042 0043 0044 0045 0045 0051 0052 0053 0055 0055 0057 0058 0059 0061 0066 0066 0066 0066 0068 0069	0000 0000	HRESBUFF TMPSTACK EBHITOK EBHISTOK CURCHAR * HBUFF HO HB.ADDR HB.NUM HB.SIZE HB.LEN * VIDEO RI * INITØ BI COCO MMUEN IEN MC3 MC2 MC1 MCØ * INTERRUI TMR HBORD VBORD EI12 EI11 EIØ	EQU	\$CØØØ \$DFFF \$62 \$29 \$PACE UT BUFFER HEADER Ø 2 3 5 R EQUATES ATES \$8Ø \$40 \$10 8 4 2 1 UEST ENABLED \$2Ø \$10 8 4 2 1	ADDRESS OF THE GET/PUT BUFFERS IN THE CPU'S MEMORY SPACE ADDRESS OF THE HI-RES GRAPHICS STACK IN THE CPU'S MEMORY SPACE FIRST ENHANCED BASIC TOKEN NUMBER FIRST ENHANCED BASIC FUNCTION TOKEN NUMBER BUG - SHOULD BE \$28 HI-RES CURSOR CHARACTER EQUATES ADDRESS OF THE NEXT BUFFER - 2 BYTES NUMBER OF THIS BUFFER - 1 BYTES NUMBER OF BYTES IN THE BUFFER - 2 BYTES NUMBER OF BYTES IN THIS HEADER 1 = Color Computer compatible 1 = MMU enabled 1 = GIME chip IRQ output enabled 1 = GIME chip FIRQ output enabled 1 = RAM at XFEXX is constant 1 = standard SCS ROM map control ROM map control TIMER HORIZONTAL BORDER VERTICAL BORDER SERIAL DATA KEYBOARD
0038 0039 0040 0041 0042 0043 0044 0045 0045 0047 0051 0052 0053 0055 0055 0055 0055 0055 0061 0062 0063 0064 0066 0067 0068 0069 0070	C000 DFFF 0062 0029 0020 0000 0002 0003 0005 0080 0040 0010 0008 0004 0002 0001 0008 0008 0004 0008 0008 0004 0008 0008	HRESBUFF TMPSTACK EBHITOK EBHISTOK CURCHAR * HBUFF HO HB.ADDR HB.NUM HB.SIZE HB.LEN * VIDEO RE * INITØ BE COCO MMUEN IEN FEN MC3 MC2 MC1 MCØ * INTERRUE TMR HBORD VBORD EI2 EI1 EIØ * EXPANDEI	EQU	\$CØØØ \$DFFF \$62 \$29 \$PACE UT BUFFER HEADER Ø 2 3 5 R EQUATES ATES \$8Ø \$4Ø \$2Ø \$1Ø 8 4 2 1 UEST ENABLED \$2Ø \$1Ø 8 4 2 1 RY DEFINITIONS	ADDRESS OF THE GET/PUT BUFFERS IN THE CPU'S MEMORY SPACE ADDRESS OF THE HI-RES GRAPHICS STACK IN THE CPU'S MEMORY SPACE FIRST ENHANCED BASIC TOKEN NUMBER FIRST ENHANCED BASIC FUNCTION TOKEN NUMBER BUG - SHOULD BE \$28 HI-RES CURSOR CHARACTER EQUATES ADDRESS OF THE NEXT BUFFER - 2 BYTES NUMBER OF THIS BUFFER - 1 BYTES NUMBER OF BYTES IN THE BUFFER - 2 BYTES NUMBER OF BYTES IN THIS HEADER 1 = Color Computer compatible 1 = MMU enabled 1 = GIME chip FIRQ output enabled 1 = GIME chip FIRQ output enabled 1 = RAM at XFEXX is constant 1 = standard SCS ROM map control ROM map control TIMER HORIZONTAL BORDER VERTICAL BORDER SERIAL DATA KEYBOARD CARTRIDGE
0038 0039 0040 0041 0042 00443 00446 00447 00448 00447 00551 00551 00553 00554 00557 00558 00557 0058 00661 00662 00663 00666 00668 00668 00668 00670 00670 00671	C000 DFFF 0062 0029 0020 0003 0005 0080 0040 0010 0008 0004 0002 0010 0008 0010 0008 0004 0002 0010	HRESBUFF TMPSTACK EBHITOK EBHISTOK CURCHAR * HBUFF HO HB.ADDR HB.NUM HB.SIZE HB.LEN * VIDEO RI * INITØ BI COCO MMUEN IEN FEN MC3 MC2 MC1 MCØ * INTERRUI TMR HBORD VBORD EI2 EI1 EIØ * EXPANDEI BLOCK 6.0	EQU	\$CØØØ \$DFFF \$62 \$29 \$PACE UT BUFFER HEADER Ø 2 3 5 R EQUATES ATES \$8Ø \$40 \$2Ø \$10 8 4 2 1 UEST ENABLED \$2Ø \$10 8 4 2 1 RY DEFINITIONS \$3Ø	ADDRESS OF THE GET/PUT BUFFERS IN THE CPU'S MEMORY SPACE ADDRESS OF THE HI-RES GRAPHICS STACK IN THE CPU'S MEMORY SPACE FIRST ENHANCED BASIC TOKEN NUMBER FIRST ENHANCED BASIC FUNCTION TOKEN NUMBER BUG - SHOULD BE \$28 HI-RES CURSOR CHARACTER EQUATES ADDRESS OF THE NEXT BUFFER - 2 BYTES NUMBER OF THIS BUFFER - 1 BYTES NUMBER OF BYTES IN THE BUFFER - 2 BYTES NUMBER OF BYTES IN THIS HEADER 1 = Color Computer compatible 1 = MMU enabled 1 = GIME chip IRQ output enabled 1 = GIME chip FIRQ output enabled 1 = GIME chip FIRQ output enabled 1 = RAM at XFEXX is constant 1 = standard SCS ROM map control ROM map control ROM map control TIMER HORIZONTAL BORDER VERTICAL BORDER VERTICAL BORDER SERIAL DATA KEYBOARD CARTRIDGE
0038 0039 0040 0041 00443 0044 00445 00446 00447 00448 00450 00551 00552 00553 00554 00558 00558 00558 00661 00662 00664 00665 00668 00669 0067 0067 0067 0067 0067 0067 006	C000 DFFF 0062 0029 0020 0000 0002 0003 0005 0080 0040 0010 0008 0004 0002 0001 0008 0004 0008 0004 0008 0004 0008	HRESBUFF TMPSTACK EBHITOK EBHISTOK CURCHAR * HBUFF HO HB.ADDR HB.NUM HB.SIZE HB.LEN * VIDEO RE * INITØ BE COCO MMUEN IEN FEN MC3 MC2 MC1 MCØ * INTERRUE TMR HBORD VBORD EI2 EI1 EIØ * EXPANDEI	EQU	\$CØØØ \$DFFF \$62 \$29 \$PACE UT BUFFER HEADER Ø 2 3 5 R EQUATES ATES \$8Ø \$4Ø \$2Ø \$1Ø 8 4 2 1 UEST ENABLED \$2Ø \$1Ø 8 4 2 1 RY DEFINITIONS	ADDRESS OF THE GET/PUT BUFFERS IN THE CPU'S MEMORY SPACE ADDRESS OF THE HI-RES GRAPHICS STACK IN THE CPU'S MEMORY SPACE FIRST ENHANCED BASIC TOKEN NUMBER FIRST ENHANCED BASIC FUNCTION TOKEN NUMBER BUG - SHOULD BE \$28 HI-RES CURSOR CHARACTER EQUATES ADDRESS OF THE NEXT BUFFER - 2 BYTES NUMBER OF THIS BUFFER - 1 BYTES NUMBER OF BYTES IN THE BUFFER - 2 BYTES NUMBER OF BYTES IN THIS HEADER 1 = Color Computer compatible 1 = MMU enabled 1 = GIME chip FIRQ output enabled 1 = GIME chip FIRQ output enabled 1 = RAM at XFEXX is constant 1 = standard SCS ROM map control ROM map control TIMER HORIZONTAL BORDER VERTICAL BORDER SERIAL DATA KEYBOARD CARTRIDGE
0038 0039 0040 0041 0042 00443 00446 00447 00448 00447 00551 00551 00553 00554 00557 00558 00557 0058 00661 00662 00663 00666 00668 00668 00668 00670 00670 00671	C000 DFFF 0062 0029 0020 0003 0005 0080 0040 0010 0008 0004 0002 0010 0008 0010 0008 0004 0002 0010	HRESBUFF TMPSTACK EBHITOK EBHISTOK CURCHAR * HBUFF HC HB.ADDR HB.NUM HB.SIZE HB.LEN * VIDEO RE * INITØ BE COCO MMUEN IEN FEN MC3 MC2 MC1 MCØ * INTERRUE TMR HBORD VBORD EI2 EI1 EIØ * EXPANDEE BLOCK 6.0 BLOCK 6.1	EQU	\$CØØØ \$DFFF \$62 \$29 \$PACE UT BUFFER HEADER Ø 2 3 5 R EQUATES ATES \$8Ø \$40 \$2Ø \$10 8 4 2 1 UEST ENABLED \$2Ø \$10 8 4 2 1 RY DEFINITIONS \$3Ø	ADDRESS OF THE GET/PUT BUFFERS IN THE CPU'S MEMORY SPACE ADDRESS OF THE HI-RES GRAPHICS STACK IN THE CPU'S MEMORY SPACE FIRST ENHANCED BASIC TOKEN NUMBER FIRST ENHANCED BASIC FUNCTION TOKEN NUMBER BUG - SHOULD BE \$28 HI-RES CURSOR CHARACTER EQUATES ADDRESS OF THE NEXT BUFFER - 2 BYTES NUMBER OF THIS BUFFER - 1 BYTES NUMBER OF BYTES IN THE BUFFER - 2 BYTES NUMBER OF BYTES IN THIS HEADER 1 = Color Computer compatible 1 = MMU enabled 1 = GIME chip IRQ output enabled 1 = GIME chip FIRQ output enabled 1 = GIME chip FIRQ output enabled 1 = RAM at XFEXX is constant 1 = standard SCS ROM map control ROM map control ROM map control TIMER HORIZONTAL BORDER VERTICAL BORDER VERTICAL BORDER SERIAL DATA KEYBOARD CARTRIDGE
0038 0039 0040 0041 00443 0044 00445 00446 00447 00448 00450 00551 00552 00553 00554 00558 00558 00558 00661 00662 00664 00665 00668 00669 0067 0067 0067 0067 0067 0067 006	C000 DFFF 0062 0029 0020 0000 0002 0003 0005 0080 0040 0010 0008 0004 0002 0001 0008 0004 0008 0004 0008 0004 0008	HRESBUFF TMPSTACK EBHITOK EBHISTOK CURCHAR * HBUFF HO HB.ADDR HB.NUM HB.SIZE HB.LEN * VIDEO RI * INITØ BI COCO MMUEN IEN FEN MC3 MC2 MC1 MCØ * INTERRUI TMR HBORD VBORD EI2 EI1 EIØ * EXPANDEI BLOCK 6.0	EQU	\$CØØØ \$DFFF \$62 \$29 \$PACE UT BUFFER HEADER Ø 2 3 5 R EQUATES ATES \$8Ø \$4Ø \$2Ø \$1Ø 8 4 2 1 UEST ENABLED \$2Ø \$1Ø 8 4 2 1 RY DEFINITIONS \$3Ø \$31	ADDRESS OF THE GET/PUT BUFFERS IN THE CPU'S MEMORY SPACE ADDRESS OF THE HI-RES GRAPHICS STACK IN THE CPU'S MEMORY SPACE FIRST ENHANCED BASIC TOKEN NUMBER FIRST ENHANCED BASIC FUNCTION TOKEN NUMBER BUG - SHOULD BE \$28 HI-RES CURSOR CHARACTER EQUATES ADDRESS OF THE NEXT BUFFER - 2 BYTES NUMBER OF THIS BUFFER - 1 BYTES NUMBER OF BYTES IN THE BUFFER - 2 BYTES NUMBER OF BYTES IN THIS HEADER 1 = Color Computer compatible 1 = MMU enabled 1 = GIME chip IRQ output enabled 1 = GIME chip FIRQ output enabled 1 = RAM at XFEXX is constant 1 = standard SCS ROM map control ROM map control TIMER HORIZONTAL BORDER VERTICAL BORDER VERTICAL BORDER SERIAL DATA KEYBOARD CARTRIDGE BLOCKS \$3Ø-\$33 ARE THE HI-RES GRAPHICS SCREEN HI-RES GRAPHICS SCREEN

```
ØØ75
           ØØ34
                     BLOCK 6.4 EQU
                                                       GET/PUT BUFFER
                                      $34
ØØ76
           ØØ35
                     BLOCK 6.5 EQU
                                      $35
                                                       STACK AREA FOR HI-RES GRAPHICS COMMAND
0077
           0036
                     BLOCK 6.6 EQU
                                                       CHARACTER POINTERS
                                      $36
ØØ78
           ØØ37
                     BLOCK 6.7 EQU
                                      $37
                                                       UNUSED BY BASIC
0079
øø8ø
                     * BLOCKS $48-$4F ARE USED FOR THE BASIC OPERATING SYSTEM
           ØØ38
                     BLOCK7.Ø EQU
0081
                                      $38
ØØ82
           ØØ39
                     BLOCK7.1
                                EQU
                                      $39
ØØ83
           ØØ3A
                     BLOCK7.2
                                EQU
                                      $3A
           ØØ3B
                     BLOCK7.3
0084
                                EOU
                                      $3B
ØØ85
           ØØ3C
                     BLOCK7.4
                                EQU
                                      $3C
0086
           ØØ3D
                     BLOCK7.5
                                EQU
                                      $3D
           ØØ3E
                     BLOCK7.6
ØØ87
                                FOU
                                      $3F
ØØ88
           ØØ3F
                     BLOCK7.7
                                EQU
                                      $3F
ØØ89
øø9ø
ØØ91
0092 0000
                                ORG
                                      α
ØØ93
           ØØØØ
                                SETDP Ø
ØØ94
0095 0000
                     ENDELG
                                RMR
                                                       STOP/END FLAG: POSITIVE=STOP, NEG=END
                                      1
0096 0001
                     CHARAC
                                RMB
                                                       TERMINATOR FLAG 1
                                      1
                     ENDCUR
                                                       TERMINATOR FLAG 2
0097 0002
                                RMB
                                      1
0098 0003
                     TMPL0C
                                RMB
                                                       SCRATCH VARIABLE
                                      1
0099 0004
                     TECTR
                                RMR
                                      1
                                                       IF COUNTER - HOW MANY IF STATEMENTS IN A LINE
0100 0005
                     DIMFLG
                                RMB
                                                       *DV* ARRAY FLAG Ø=EVALUATE, 1=DIMENSIONING
                                      1
0101 0006
                     VALTYP
                                RMB
                                      1
                                                       *DV* *PV TYPE FLAG: Ø=NUMERIC, $FF=STRING
                     GARBFL
                                                       *TV STRING SPACE HOUSEKEEPING FLAG
0102 0007
                                RMB
                                      1
                                RMR
                                                       DISABLE ARRAY SEARCH: ØØ=ALLOW SEARCH
0103 0008
                     ARYDIS
                                      1
0104 0009
                     INPFLG
                                RMB
                                      1
                                                       *TV INPUT FLAG: READ=Ø, INPUT<>Ø
                                                       *TV RELATIONAL OPERATOR FLAG
Ø1Ø5 ØØØA
                     RELFLG
                                RMR
                                      1
Ø1Ø6 ØØØB
                     TEMPPT
                                RMB
                                      2
                                                       *PV TEMPORARY STRING STACK POINTER
                                                       *PV ADDR OF LAST USED STRING STACK ADDRESS
Ø1Ø7 ØØØD
                     LASTPT
                                RMB
                                      2
                                                       TEMPORARY POINTER
Ø1Ø8 ØØØF
                     TFMPTR
                                RMR
                                      2
0109 0011
                     TMPTR1
                                RMB
                                      2
                                                       TEMPORARY DESCRIPTOR STORAGE (STACK SEARCH)
                     ** FLOATING POINT ACCUMULATOR #2 (MANTISSA ONLY)
Ø11Ø
0111 0013
                     FPA2
                                RMB
                                      4
                                                       FLOATING POINT ACCUMULATOR #2 MANTISSA
                                                       BOTTOM OF STACK AT LAST CHECK
0112 0017
                     BOTSTK
                                RMR
                                      2
0113 0019
                                                       *PV BEGINNING OF BASIC PROGRAM
                     TXTTAR
                                RMR
                                      2
                                                       *PV START OF VARIABLES
Ø114 ØØ1B
                     VARTAB
                                RMR
                                      2
Ø115 ØØ1D
                     ARYTAB
                                RMB
                                      2
                                                       *PV START OF ARRAYS
Ø116 ØØ1F
                     ARYEND
                                RMB
                                      2
                                                       *PV END OF ARRAYS (+1)
                                                       *PV START OF STRING STORAGE (TOP OF FREE RAM)
*PV START OF STRING VARIABLES
                                RMR
0117 0021
                     FRETOP
                                      2
0118 0023
                     STRTAB
                                RMB
                                      2
0119 0025
                     FRESPC
                                RMB
                                      2
                                                       UTILITY STRING POINTER
0120 0027
                     MEMSIZ
                                RMB
                                      2
                                                       *PV TOP OF STRING SPACE
                                                       SAVED LINE NUMBER DURING A "STOP"
0121 0029
                     OLDTXT
                                RMR
                                      2
Ø122 ØØ2B
                     BINVAL
                                RMB
                                      2
                                                       BINARY VALUE OF A CONVERTED LINE NUMBER
                                                       SAVED INPUT PTR DURING A "STOP"
Ø123 ØØ2D
                     OLDPTR
                                RMB
                                      2
Ø124 ØØ2F
                     TINPTR
                                      2
                                                       TEMPORARY INPUT POINTER STORAGE
                                RMR
                                                       *PV 'DATA' STATEMENT LINE NUMBER POINTER
*PV 'DATA' STATEMENT ADDRESS POINTER
0125 0031
                     DATTXT
                                RMB
                                      2
0126 0033
                     DATPTR
                                RMB
                                      2
                                                       DATA POINTER FOR 'INPUT' & 'READ'
0127 0035
                     DATTMP
                                RMB
                                      2
                                                       *TV TEMP STORAGE FOR A VARIABLE NAME
0128 0037
                     VARNAM
                                RMR
                                      2
0129 0039
                     VARPTR
                                RMB
                                      2
                                                       *TV POINTER TO A VARIABLE DESCRIPTOR
Ø13Ø ØØ3B
                     VARDES
                                      2
                                                       TEMP POINTER TO A VARIABLE DESCRIPTOR
                                RMB
                                                       POINTER TO RELATIONAL OPERATOR PROCESSING ROUTINE
Ø131 ØØ3D
                     RELPTR
                                RMB
                                      2
                                                       TEMPORARY RELATIONAL OPERATOR FLAG BYTE
Ø132 ØØ3F
                     TRELFL
                                RMR
Ø133
Ø134
                     * FLOATING POINT ACCUMULATORS #3,4 & 5 ARE MOSTLY
                     * USED AS SCRATCH PAD VARIABLES.
Ø135
                     ** FLOATING POINT ACCUMULATOR #3 :PACKED: ($40-$44)
Ø136
0137 0040
                     V4Ø
                                RMB
0138 0041
                     ۷41
                                RMB
                                      1
Ø139 ØØ42
                                RMR
                     V42
                                      1
0140 0043
                     V43
                                RMB
                                      1
0141 0044
                     V44
                                RMR
0142
                     ** FLOATING POINT ACCUMULATOR #4 : PACKED: ($45-$49)
0143 0045
                     V45
                                RMB
                                     1
0144 0046
                     ۷46
                                RMB
0145 0047
                     V47
                                RMR
                                      1
                                RMB
0146 0048
                     V48
                     ** FLOATING POINT ACCUMULATOR #5 : PACKED: ($4A $4E)
Ø147
Ø148 ØØ4A
                     V4A
                                RMB 1
```

Ø149 ØØ4					
	R	V4B	RMB	2	
0150 004				2	
					I A
Ø151				Γ ACCUMULATOR ≇	
0152 004	F	FPØEXP	RMB	1	*PV FLOATING POINT ACCUMULATOR #Ø EXPONENT
Ø153 ØØ5	Ø	FPAØ	RMB .	4	*PV FLOATING POINT ACCUMULATOR #Ø MANTISSA
0154 005	4	FPØSGN	RMB	1	*PV FLOATING POINT ACCUMULATOR #Ø SIGN
Ø155 ØØ5			RMB		POLYNOMIAL COEFFICIENT COUNTER
0156 005				5	TEMPORARY STRING DESCRIPTOR
Ø157 ØØ5	В	FPCARY	RMB	1	FLOATING POINT CARRY BYTE
Ø158		** FLOATIN	G POIN	Γ ACCUMULATOR ₹	#1
Ø159 ØØ5	С	FP1EXP	RMB	1	*PV FLOATING POINT ACCUMULATOR #1 EXPONENT
Ø16Ø ØØ5				- 4	*PV FLOATING POINT ACCUMULATOR #1 MANTISSA
0161 006	1	FP1SGN	RMB	1	*PV FLOATING POINT ACCUMULATOR #1 SIGN
Ø162					
Ø163 ØØ6	2	RESSGN	RMB	1	SIGN OF RESULT OF FLOATING POINT OPERATION
0164 006	3	FPSBYT	RMB	1	FLOATING POINT SUB BYTE (FIFTH BYTE)
Ø165 ØØ6				2	POLYNOMIAL COEFFICIENT POINTER
0166 006				2	CURRENT LINE POINTER DURING LIST
Ø167 ØØ6	8	CURLIN	RMB .	2	*PV CURRENT LINE # OF BASIC PROGRAM, \$FFFF = DIRECT
Ø168 ØØ6	Α	DEVCFW	RMB	1	*TV TAB FIELD WIDTH
Ø169 ØØ6	В	DEVLCF	RMB	1	*TV TAB ZONE
0170 006				1	*TV PRINT POSITION
0171 006				1	*TV PRINT WIDTH
Ø172 ØØ6		PRTDEV	RMB	1	*TV PRINT DEVICE: Ø=NOT CASSETTE, -1=CASSETTE
Ø173 ØØ6	F	DEVNUM	RMB	1	*PV DEVICE NUMBER: -3=DLOAD, -2=PRINTER,
Ø174		*			-1=CASSETTE, Ø=SCREEN, 1-15=DISK
0175 007	α	CINBFL	RMB	1	*PV CONSOLE IN BUFFER FLAG: ØØ=NOT EMPTY, \$FF=EMPTY
				<u>.</u> 1	
0176 007					*PV WARM START FLAG: \$55=WARM, OTHER=COLD
0177 007			RMB	2	*PV WARM START VECTOR - JUMP ADDRESS FOR WARM START
Ø178 ØØ7	4	TOPRAM	RMB	2	*PV TOP OF RAM
Ø179 ØØ7	6		RMB	2	SPARE: UNUSED VARIABLES
Ø18Ø ØØ7		FILSTA		1	*PV FILE STATUS FLAG: Ø=CLOSED, 1=INPUT, 2=OUTPUT
Ø181 ØØ7					
		CINCTR		1	*PV CONSOLE IN BUFFER CHAR COUNTER
0182 007		CINPTR		2	*PV CONSOLE IN BUFFER POINTER
Ø183 ØØ7	С	BLKTYP	RMB	1	*TV CASS BLOCK TYPE: Ø=HEADER, 1=DATA, \$FF=EOF
Ø184 ØØ7	D	BLKLEN	RMB	1	*TV CASSETTE BYTE COUNT
Ø185 ØØ7		CBUFAD		2	*TV CASSETTE LOAD BUFFER POINTER
Ø186 ØØ8				- 1	
					*TV CASSETTE CHECKSUM BYTE
Ø187 ØØ8	1	CSRERR	RMB	1	*TV ERROR FLAG/CHARACTER COUNT
Ø188 ØØ8	2	CPULWD	RMB	1	*TV PULSE WIDTH COUNT
Ø189 ØØ8	3	CPERTM	RMB	1	*TV BIT COUNTER
0190 008			RMB	1	*TV BIT PHASE FLAG
	4	(.BIPHA		-	
			DMD	1	*TV IAST SINE TABLE ENTO
0191 008	5	CLSTSN		1	*TV LAST SINE TABLE ENTRY
Ø191 ØØ8 Ø192 ØØ8	5 6	CLSTSN GRBLOK	RMB	1	*TV GRAPHIC BLOCK VALUE FOR SET, RESET AND POINT
0191 008	5 6 7	CLSTSN GRBLOK IKEYIM	RMB		
Ø191 ØØ8 Ø192 ØØ8	5 6 7	CLSTSN GRBLOK IKEYIM	RMB RMB	1	*TV GRAPHIC BLOCK VALUE FOR SET, RESET AND POINT
0191 008 0192 008 0193 008 0194 008	5 6 7 8	CLSTSN GRBLOK IKEYIM CURPOS	RMB RMB RMB	1 1 2	*TV GRAPHIC BLOCK VALUE FOR SET, RESET AND POINT *TV INKEY\$ RAM IMAGE *PV CURSOR LOCATION
0191 008 0192 008 0193 008 0194 008 0195 008	5 6 7 8 A	CLSTSN GRBLOK IKEYIM CURPOS ZERO	RMB RMB RMB RMB	1 1 2 2	*TV GRAPHIC BLOCK VALUE FOR SET, RESET AND POINT *TV INKEY\$ RAM IMAGE *PV CURSOR LOCATION *PV DUMMY - THESE TWO BYTES ARE ALWAYS ZERO
0191 008 0192 008 0193 008 0194 008 0195 008 0196 008	5 6 7 8 A C	CLSTSN GRBLOK IKEYIM CURPOS ZERO SNDTON	RMB RMB RMB RMB RMB	1 1 2 2 1	*TV GRAPHIC BLOCK VALUE FOR SET, RESET AND POINT *TV INKEY\$ RAM IMAGE *PV CURSOR LOCATION *PV DUMMY - THESE TWO BYTES ARE ALWAYS ZERO *TV TONE VALUE FOR SOUND COMMAND
0191 008 0192 008 0193 008 0194 008 0195 008 0196 008 0197 008	5 6 7 8 A C	CLSTSN GRBLOK IKEYIM CURPOS ZERO SNDTON	RMB RMB RMB RMB	1 1 2 2 1	*TV GRAPHIC BLOCK VALUE FOR SET, RESET AND POINT *TV INKEY\$ RAM IMAGE *PV CURSOR LOCATION *PV DUMMY - THESE TWO BYTES ARE ALWAYS ZERO
0191 008 0192 008 0193 008 0194 008 0195 008 0196 008 0197 008 0198	5 6 7 8 A C	CLSTSN GRBLOK IKEYIM CURPOS ZERO SNDTON SNDDUR	RMB RMB RMB RMB RMB	1 1 2 2 2 1	*TV GRAPHIC BLOCK VALUE FOR SET, RESET AND POINT *TV INKEY\$ RAM IMAGE *PV CURSOR LOCATION *PV DUMMY - THESE TWO BYTES ARE ALWAYS ZERO *TV TONE VALUE FOR SOUND COMMAND *TV DURATION VALUE FOR SOUND COMMAND
0191 008 0192 008 0193 008 0194 008 0195 008 0196 008 0197 008	5 6 7 8 A C	CLSTSN GRBLOK IKEYIM CURPOS ZERO SNDTON SNDDUR	RMB RMB RMB RMB RMB	1 1 2 2 1	*TV GRAPHIC BLOCK VALUE FOR SET, RESET AND POINT *TV INKEY\$ RAM IMAGE *PV CURSOR LOCATION *PV DUMMY - THESE TWO BYTES ARE ALWAYS ZERO *TV TONE VALUE FOR SOUND COMMAND *TV DURATION VALUE FOR SOUND COMMAND
0191 008 0192 008 0193 008 0194 008 0195 008 0196 008 0197 008 0198	5 6 7 8 8 A C D	CLSTSN GRBLOK IKEYIM CURPOS ZERO SNDTON SNDDUR	RMB RMB RMB RMB RMB	1 1 2 2 2 1	*TV GRAPHIC BLOCK VALUE FOR SET, RESET AND POINT *TV INKEY\$ RAM IMAGE *PV CURSOR LOCATION *PV DUMMY - THESE TWO BYTES ARE ALWAYS ZERO *TV TONE VALUE FOR SOUND COMMAND *TV DURATION VALUE FOR SOUND COMMAND
0191 008 0192 008 0193 008 0194 008 0195 008 0196 008 0197 008 0199 0200	5 6 7 8 8 A C D	CLSTSN GRBLOK IKEYIM CURPOS ZERO SNDTON SNDDUR ** THESE B	RMB RMB RMB RMB RMB	1 1 2 2 2 1	*TV GRAPHIC BLOCK VALUE FOR SET, RESET AND POINT *TV INKEY\$ RAM IMAGE *PV CURSOR LOCATION *PV DUMMY - THESE TWO BYTES ARE ALWAYS ZERO *TV TONE VALUE FOR SOUND COMMAND *TV DURATION VALUE FOR SOUND COMMAND FROM ROM INIT DESCRIPTION
0191 008 0192 008 0193 008 0194 008 0195 008 0196 008 0197 008 0199 0200 0201	5 6 7 8 8 A C D	CLSTSN GRBLOK IKEYIM CURPOS ZERO SNDTON SNDDUR ** THESE B ***	RMB RMB RMB RMB RMB RMB	1 1 2 2 2 1 2 RE MOVED DOWN	*TV GRAPHIC BLOCK VALUE FOR SET, RESET AND POINT *TV INKEY\$ RAM IMAGE *PV CURSOR LOCATION *PV DUMMY - THESE TWO BYTES ARE ALWAYS ZERO *TV TONE VALUE FOR SOUND COMMAND *TV DURATION VALUE FOR SOUND COMMAND FROM ROM INIT DESCRIPTION VALUE
0191 008 0192 008 0193 008 0194 008 0195 008 0196 008 0197 008 0198 0199 0200 0201 0202 008	5 6 7 8 A C D	CLSTSN GRBLOK IKEYIM CURPOS ZERO SNDTON SNDDUR ** THESE B *** * CMPMID	RMB RMB RMB RMB RMB RMB	1 1 2 2 1 1 2 RE MOVED DOWN	*TV GRAPHIC BLOCK VALUE FOR SET, RESET AND POINT *TV INKEY\$ RAM IMAGE *PV CURSOR LOCATION *PV DUMMY - THESE TWO BYTES ARE ALWAYS ZERO *TV TONE VALUE FOR SOUND COMMAND *TV DURATION VALUE FOR SOUND COMMAND FROM ROM INIT DESCRIPTION VALUE 18 *PV 1200/2400 HERTZ PARTITION
0191 008 0192 008 0193 008 0194 008 0195 008 0197 008 0199 0200 0201 0202 008 0203 009	5 6 7 8 8 C D	CLSTSN GRBLOK IKEYIM CKEYIM SNDTON SNDTON SNDDUR ** THESE B *** ** CMPMID CMPØ	RMB RMB RMB RMB RMB RMB YTES AI	1 1 2 2 1 2 RE MOVED DOWN	*TV GRAPHIC BLOCK VALUE FOR SET, RESET AND POINT *TV INKEY\$ RAM IMAGE *PV CURSOR LOCATION *PV DUMMY - THESE TWO BYTES ARE ALWAYS ZERO *TV TONE VALUE FOR SOUND COMMAND *TV DURATION VALUE FOR SOUND COMMAND FROM ROM INIT DESCRIPTION VALUE 18 *PV 1200/2400 HERTZ PARTITION 24 *PV UPPER LIMIT OF 1200 HERTZ PERIOD
0191 008 0192 008 0193 008 0194 008 0195 008 0196 008 0197 008 0199 0200 0201 0202 008 0203 009 0204 009	5 6 7 8 A C D	CLSTSN GRBLOK IKEYIM CURPOS ZERO SNOTON SNDDUR ** THESE B *** CMPMID CMPØ CMP1	RMB RMB RMB RMB RMB RMB RMB RMB	1 1 2 2 1 2 2 RE MOVED DOWN	*TV GRAPHIC BLOCK VALUE FOR SET, RESET AND POINT *TV INKEY\$ RAM IMAGE *PV CURSOR LOCATION *PV DUMMY - THESE TWO BYTES ARE ALWAYS ZERO *TV TONE VALUE FOR SOUND COMMAND *TV DURATION VALUE FOR SOUND COMMAND FROM ROM INIT DESCRIPTION VALUE 18 *PV 1200/2400 HERTZ PARTITION 24 *PV UPPER LIMIT OF 1200 HERTZ PERIOD 10 *PV UPPER LIMIT OF 2400 HERTZ PERIOD
0191 008 0192 008 0193 008 0194 008 0195 008 0197 008 0199 0200 0201 0202 008 0203 009	5 6 7 8 A C D	CLSTSN GRBLOK IKEYIM CKEYIM SNDTON SNDTON SNDDUR ** THESE B *** ** CMPMID CMPØ	RMB RMB RMB RMB RMB RMB RMB RMB	1 1 2 2 1 2 RE MOVED DOWN	*TV GRAPHIC BLOCK VALUE FOR SET, RESET AND POINT *TV INKEY\$ RAM IMAGE *PV CURSOR LOCATION *PV DUMMY - THESE TWO BYTES ARE ALWAYS ZERO *TV TONE VALUE FOR SOUND COMMAND *TV DURATION VALUE FOR SOUND COMMAND FROM ROM INIT DESCRIPTION VALUE 18 *PV 1200/2400 HERTZ PARTITION 24 *PV UPPER LIMIT OF 1200 HERTZ PERIOD
0191 008 0192 008 0193 008 0194 008 0195 008 0196 008 0197 008 0199 0200 0201 0202 008 0203 009 0204 009	5 6 7 8 A C D D F Ø 1 2	CLSTSN GRBLOK IKEYIM CURPOS ZERO SNOTON SNDDUR ** THESE B *** CMPMID CMPØ CMP1	RMB RMB RMB RMB RMB RMB RMB TTES AI	1 1 2 2 1 2 2 RE MOVED DOWN	*TV GRAPHIC BLOCK VALUE FOR SET, RESET AND POINT *TV INKEY\$ RAM IMAGE *PV CURSOR LOCATION *PV DUMMY - THESE TWO BYTES ARE ALWAYS ZERO *TV TONE VALUE FOR SOUND COMMAND *TV DURATION VALUE FOR SOUND COMMAND FROM ROM INIT DESCRIPTION VALUE 18 *PV 1200/2400 HERTZ PARTITION 24 *PV UPPER LIMIT OF 1200 HERTZ PERIOD 10 *PV UPPER LIMIT OF 2400 HERTZ PERIOD 128 *PV NUMBER OF \$55'S TO CASSETTE LEADER
0191 008 0192 008 0193 008 0194 008 0195 008 0197 008 0199 0200 0201 0202 008 0203 009 0204 009 0206 009	5 6 7 8 A C D F Ø 1 1 2 4	CLSTSN GRBLOK IKEYIM CURPOS ZERO SNOTON SNDDUR ** THESE B *** CMPMID CMPØ CMPI SYNCLN BLKCNT	RMB RMB RMB RMB RMB RMB RMB YTES AI	1 1 2 2 1 2 RE MOVED DOWN 1 1 1 1	*TV GRAPHIC BLOCK VALUE FOR SET, RESET AND POINT *TV INKEY\$ RAM IMAGE *PV CURSOR LOCATION *PV DUMMY - THESE TWO BYTES ARE ALWAYS ZERO *TV TONE VALUE FOR SOUND COMMAND *TV DURATION VALUE FOR SOUND COMMAND FROM ROM INIT DESCRIPTION VALUE 18 *PV 1200/2400 HERTZ PARTITION 24 *PV UPPER LIMIT OF 1200 HERTZ PERIOD 10 *PV UPPER LIMIT OF 2400 HERTZ PERIOD 128 *PV NUMBER OF \$55'S TO CASSETTE LEADER 11 *PV CURSOR BLINK DELAY
0191 008 0192 008 0193 008 0194 008 0195 008 0197 008 0199 0200 0201 0202 008 0203 009 0204 009 0205 009 0207 009	5 6 7 8 A C D F Ø 1 1 2 4 5	CLSTSN GRBLOK IKEYIM CURPOS ZERO SNDTON SNDDUR ** THESE B' *** ** CMPMID CMPØ CMPI SYNCLN BLKCNT LPTBTD	RMB RMB RMB RMB RMB RMB YTES AI	1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	*TV GRAPHIC BLOCK VALUE FOR SET, RESET AND POINT *TV INKEY\$ RAM IMAGE *PV CURSOR LOCATION *PV DUMMY - THESE TWO BYTES ARE ALWAYS ZERO *TV TONE VALUE FOR SOUND COMMAND *TV DURATION VALUE FOR SOUND COMMAND FROM ROM INIT DESCRIPTION VALUE 18 *PV 1200/2400 HERTZ PARTITION 24 *PV UPPER LIMIT OF 1200 HERTZ PERIOD 10 *PV UPPER LIMIT OF 2400 HERTZ PERIOD 128 *PV NUMBER OF \$55'S TO CASSETTE LEADER 11 *PV CURSOR BLINK DELAY 88 *PV BAUD RATE CONSTANT (600)
0191 008 0192 008 0193 008 0194 008 0195 008 0196 008 0197 008 0199 0200 0201 0202 008 0203 009 0204 009 0205 009 0207 009 0208 009	5 6 7 8 A C D F Ø 1 2 4 5 7	CLSTSN GRBLOK IKEYIM CURPOS ZERO SNDTON SNDDUR ** THESE B' *** * CMPMID CMPMI CMPMI SYNCLN BLKCNT LPTBTD LPTLND	RMB RMB RMB RMB RMB RMB YTES AI RMB RMB RMB RMB RMB RMB RMB RMB RMB	1 1 2 2 2 2 1 1 1 2 2 2 1 1 2 2 2 2 2 2	*TV GRAPHIC BLOCK VALUE FOR SET, RESET AND POINT *TV INKEY\$ RAM IMAGE *PV CURSOR LOCATION *PV DUMMY - THESE TWO BYTES ARE ALWAYS ZERO *TV TONE VALUE FOR SOUND COMMAND *TV DURATION VALUE FOR SOUND COMMAND FROM ROM INIT DESCRIPTION VALUE 18 *PV 1200/2400 HERTZ PARTITION 24 *PV UPPER LIMIT OF 1200 HERTZ PERIOD 10 *PV UPPER LIMIT OF 2400 HERTZ PERIOD 128 *PV NUMBER OF \$55'S TO CASSETTE LEADER 11 *PV CURSOR BLINK DELAY 88 *PV BAUD RATE CONSTANT (600) 1 *PV PRINTER CARRIAGE RETURN DELAY
0191 008 0192 008 0193 008 0194 008 0195 008 0197 008 0198 0199 0200 0201 0202 008 0203 009 0204 009 0205 009 0208 009 0208 009 0209 009	5 6 7 8 8 C D F Ø 1 2 4 5 7 9	CLSTSN GRBLOK IKEYIM CURYOS ZERO SNOTON SNDDUR ** THESE B ** * CMPMID CMPØ CMPI SYNCLN BLKCNT LPTBTD LPTLND LPTCFW	RMB RMB RMB RMB RMB RMB YTES AI RMB RMB RMB RMB RMB RMB RMB RMB RMB RMB	1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	*TV GRAPHIC BLOCK VALUE FOR SET, RESET AND POINT *TV INKEY\$ RAM IMAGE *PV CURSOR LOCATION *PV DUMMY - THESE TWO BYTES ARE ALWAYS ZERO *TV TONE VALUE FOR SOUND COMMAND *TV DURATION VALUE FOR SOUND COMMAND FROM ROM INIT DESCRIPTION VALUE 18 *PV 1200/2400 HERTZ PARTITION 24 *PV UPPER LIMIT OF 1200 HERTZ PERIOD 10 *PV UPPER LIMIT OF 2400 HERTZ PERIOD 128 *PV NUMBER OF \$55'S TO CASSETTE LEADER 11 *PV CURSOR BLINK DELAY 88 *PV BAUD RATE CONSTANT (600) 1 *PV PRINTER CARRIAGE RETURN DELAY 16 *PV TAB FIELD WIDTH
0191 008 0192 008 0193 008 0194 008 0195 008 0196 008 0197 008 0199 0200 0201 0202 008 0203 009 0204 009 0205 009 0207 009 0208 009	5 6 7 8 8 C D F Ø 1 2 4 5 7 9	CLSTSN GRBLOK IKEYIM CURPOS ZERO SNDTON SNDDUR ** THESE B' *** * CMPMID CMPMI CMPMI SYNCLN BLKCNT LPTBTD LPTLND	RMB RMB RMB RMB RMB RMB YTES AI RMB RMB RMB RMB RMB RMB RMB RMB RMB RMB	1 1 2 2 2 2 1 1 1 2 2 2 1 1 2 2 2 2 2 2	*TV GRAPHIC BLOCK VALUE FOR SET, RESET AND POINT *TV INKEY\$ RAM IMAGE *PV CURSOR LOCATION *PV DUMMY - THESE TWO BYTES ARE ALWAYS ZERO *TV TONE VALUE FOR SOUND COMMAND *TV DURATION VALUE FOR SOUND COMMAND FROM ROM INIT DESCRIPTION VALUE 18 *PV 1200/2400 HERTZ PARTITION 24 *PV UPPER LIMIT OF 1200 HERTZ PERIOD 10 *PV UPPER LIMIT OF 2400 HERTZ PERIOD 128 *PV NUMBER OF \$55'S TO CASSETTE LEADER 11 *PV CURSOR BLINK DELAY 88 *PV BAUD RATE CONSTANT (600) 1 *PV PRINTER CARRIAGE RETURN DELAY
0191 008 0192 008 0193 008 0194 008 0196 008 0197 008 0198 0199 0200 0201 0202 008 0203 009 0204 009 0205 009 0206 009 0207 009 0208 009 0209 009 0210 009	5 6 7 8 A C C D F Ø 1 2 4 5 7 9 A	CLSTSN GRBLOK IKEYIM CKEYIM CKEYIM SIKEYIM SIKAM SIKEYIM SIKAM SIKEYIM SIKAM	RMB RMB RMB RMB RMB RMB YTES AI RMB RMB RMB RMB RMB RMB RMB RMB RMB RMB	1 1 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1	*TV GRAPHIC BLOCK VALUE FOR SET, RESET AND POINT *TV INKEY\$ RAM IMAGE *PV CURSOR LOCATION *PV DUMMY - THESE TWO BYTES ARE ALWAYS ZERO *TV TONE VALUE FOR SOUND COMMAND *TV DURATION VALUE FOR SOUND COMMAND FROM ROM INIT DESCRIPTION VALUE 18 *PV 1200/2400 HERTZ PARTITION 24 *PV UPPER LIMIT OF 1200 HERTZ PERIOD 10 *PV UPPER LIMIT OF 2400 HERTZ PERIOD 128 *PV NUMBER OF \$55'S TO CASSETTE LEADER 11 *PV CURSOR BLINK DELAY 88 *PV BAUD RATE CONSTANT (600) 1 *PV PRINTER CARRIAGE RETURN DELAY 16 *PV TAB FIELD WIDTH 112 *PV LAST TAB ZONE
0191 008 0192 008 0193 008 0194 008 0195 008 0196 008 0197 008 0199 0200 0201 0202 008 0203 009 0204 009 0205 009 0206 009 0207 009 0208 009 0210 009 0211 009	5 6 7 8 A C C D F Ø 1 2 4 5 7 9 A B	CLSTSN GRBLOK IKEYIM CURPOS ZERO SNOTON SNDDUR ** THESE B *** CMPMID CMPØ CMP1 SYNCLN BLKCNT LPTBTD LPTLDD LPTCFW LPTLCF LPTWID	RMB RMB RMB RMB RMB RMB YTES AI RMB RMB RMB RMB RMB RMB RMB RMB RMB RMB	1 1 2 2 2 2 1 1 1 2 2 2 1 1 2 2 2 1 1 1 2 2 2 2 1	*TV GRAPHIC BLOCK VALUE FOR SET, RESET AND POINT *TV INKEY\$ RAM IMAGE *PV CURSOR LOCATION *PV DUMMY - THESE TWO BYTES ARE ALWAYS ZERO *TV TONE VALUE FOR SOUND COMMAND *TV DURATION VALUE FOR SOUND COMMAND *TV DURATION VALUE FOR SOUND COMMAND *TO DESCRIPTION VALUE 18 *PV 1200/2400 HERTZ PARTITION 24 *PV UPPER LIMIT OF 1200 HERTZ PERIOD 10 *PV UPPER LIMIT OF 2400 HERTZ PERIOD 128 *PV NUMBER OF \$55'S TO CASSETTE LEADER 11 *PV CURSOR BLINK DELAY 88 *PV BAUD RATE CONSTANT (600) 1 *PV PRINTER CARRIAGE RETURN DELAY 16 *PV TAB FIELD WIDTH 112 *PV LAST TAB ZONE 132 *PV PRINTER WIDTH
0191 008 0192 008 0193 008 0194 008 0195 008 0196 008 0197 008 0199 0200 0201 0202 008 0202 008 0204 009 0205 009 0206 009 0207 009 0208 009 0209 009 0211 009 0211 009	5 6 7 8 A C D F Ø 1 2 4 5 7 9 A B C	CLSTSN GRBLOK IKEYIM CURPOS ZERO SNOTON SNDDUR ** THESE B *** CMPMID CMPØ CMP1 SYNCLN BLKCNT LPTBTD LPTLND LPTLND LPTCFW LPTCFW LPTUD LPTLOS	RMB RMB RMB RMB RMB RMB YTES AI RMB RMB RMB RMB RMB RMB RMB RMB RMB RMB	1	*TV GRAPHIC BLOCK VALUE FOR SET, RESET AND POINT *TV INKEY\$ RAM IMAGE *PV CURSOR LOCATION *PV DUMMY - THESE TWO BYTES ARE ALWAYS ZERO *TV TONE VALUE FOR SOUND COMMAND *TV DURATION VALUE FOR SOUND COMMAND *TV DURATION VALUE FOR SOUND COMMAND *TO DESCRIPTION VALUE 18 *PV 1200/2400 HERTZ PARTITION 24 *PV UPPER LIMIT OF 1200 HERTZ PERIOD 10 *PV UPPER LIMIT OF 2400 HERTZ PERIOD 128 *PV NUMBER OF \$55'S TO CASSETTE LEADER 11 *PV CURSOR BLINK DELAY 88 *PV BAUD RATE CONSTANT (600) 1 *PV PRINTER CARRIAGE RETURN DELAY 16 *PV TAB FIELD WIDTH 112 *PV LAST TAB ZONE 132 *PV PRINTER WIDTH 0 *PV LINE PRINTER POSITION
0191 008 0192 008 0193 008 0194 008 0195 008 0196 008 0197 008 0199 0201 0202 008 0203 009 0204 009 0205 009 0206 009 0207 009 0208 009 0209 009 0211 009 0211 009	5 6 7 8 A C D F Ø 1 2 4 5 7 9 A B C	CLSTSN GRBLOK IKEYIM CURPOS ZERO SNOTON SNDDUR ** THESE B *** CMPMID CMPØ CMP1 SYNCLN BLKCNT LPTBTD LPTLDD LPTCFW LPTLCF LPTWID	RMB RMB RMB RMB RMB RMB YTES AI RMB RMB RMB RMB RMB RMB RMB RMB RMB RMB	1 1 2 2 2 2 1 1 1 2 2 2 1 1 2 2 2 1 1 1 2 2 2 2 1	*TV GRAPHIC BLOCK VALUE FOR SET, RESET AND POINT *TV INKEY\$ RAM IMAGE *PV CURSOR LOCATION *PV DUMMY - THESE TWO BYTES ARE ALWAYS ZERO *TV TONE VALUE FOR SOUND COMMAND *TV DURATION VALUE FOR SOUND COMMAND *TV DURATION VALUE FOR SOUND COMMAND *TO DESCRIPTION VALUE 18 *PV 1200/2400 HERTZ PARTITION 24 *PV UPPER LIMIT OF 1200 HERTZ PERIOD 10 *PV UPPER LIMIT OF 2400 HERTZ PERIOD 128 *PV NUMBER OF \$55'S TO CASSETTE LEADER 11 *PV CURSOR BLINK DELAY 88 *PV BAUD RATE CONSTANT (600) 1 *PV PRINTER CARRIAGE RETURN DELAY 16 *PV TAB FIELD WIDTH 112 *PV LAST TAB ZONE 132 *PV PRINTER WIDTH
0191 008 0192 008 0193 008 0194 008 0195 008 0197 008 0198 0199 0200 0201 0202 008 0203 009 0204 009 0205 009 0206 009 0208 009 0208 009 0209 009 0210 009 0211 009 0211 009 0213 009	5 6 7 8 8 C D F Ø 1 2 4 5 7 9 A B C D	CLSTSN GRBLOK IKEYIM CURPPOS ZERO SNDTON SNDDUR ** THESE B *** ** CMPMID CMPØ CMPI SYLCIN BLKCNT LPTBTD LPTLND LPTLND LPTLCF LPTWID LPTLCF LPTWID LPTPOS EXECJP	RMB RMB RMB RMB RMB RMB YTES AI RMB RMB RMB RMB RMB RMB RMB RMB RMB RMB	1 1 2 2 2 1 1 1 2 2 2 1 1 1 1 1 1 1 1 1	*TV GRAPHIC BLOCK VALUE FOR SET, RESET AND POINT *TV INKEY\$ RAM IMAGE *PV CURSOR LOCATION *PV DUMMY - THESE TWO BYTES ARE ALWAYS ZERO *TV TONE VALUE FOR SOUND COMMAND *TV DURATION VALUE FOR SOUND COMMAND **PV UPPER LIMIT OF 1200 HERTZ PERIOD 10
0191 008 0192 008 0193 008 0194 008 0195 008 0196 008 0197 008 0199 0201 0202 008 0203 009 0204 009 0205 009 0206 009 0207 009 0208 009 0209 009 0211 009 0211 009	5 6 7 8 8 C D F Ø 1 2 4 5 7 9 A B C D	CLSTSN GRBLOK IKEYIM CURPPOS ZERO SNDTON SNDDUR ** THESE B *** ** CMPMID CMPØ CMPI SYLCIN BLKCNT LPTBTD LPTLND LPTLND LPTLCF LPTWID LPTLCF LPTWID LPTPOS EXECJP	RMB RMB RMB RMB RMB RMB YTES AI RMB RMB RMB RMB RMB RMB RMB RMB RMB RMB	1 1 2 2 2 1 1 1 2 2 2 1 1 1 1 1 1 1 1 1	*TV GRAPHIC BLOCK VALUE FOR SET, RESET AND POINT *TV INKEY\$ RAM IMAGE *PV CURSOR LOCATION *PV DUMMY - THESE TWO BYTES ARE ALWAYS ZERO *TV TONE VALUE FOR SOUND COMMAND *TV DURATION VALUE FOR SOUND COMMAND *TV DURATION VALUE FOR SOUND COMMAND *TO DESCRIPTION VALUE 18 *PV 1200/2400 HERTZ PARTITION 24 *PV UPPER LIMIT OF 1200 HERTZ PERIOD 10 *PV UPPER LIMIT OF 2400 HERTZ PERIOD 128 *PV NUMBER OF \$55'S TO CASSETTE LEADER 11 *PV CURSOR BLINK DELAY 88 *PV BAUD RATE CONSTANT (600) 1 *PV PRINTER CARRIAGE RETURN DELAY 16 *PV TAB FIELD WIDTH 112 *PV LAST TAB ZONE 132 *PV PRINTER WIDTH 0 *PV LINE PRINTER POSITION
0191 008 0192 008 0193 008 0194 008 0195 008 0197 008 0198 0199 0200 0201 0202 008 0203 009 0204 009 0205 009 0206 009 0208 009 0208 009 0209 009 0210 009 0211 009 0211 009 0213 009	5 6 7 8 8 A C C D F Ø 1 2 4 5 7 9 A B C D	CLSTSN GRBLOK IKEYIM CURPOS ZERO SNOTON SNDDUR ** THESE B ** ** CMPMID CMPØ CMPI SYNCLN BLKCNT LPTBTD LPTLOF LPTLOF LPTLOF LPTLOF LPTWID LPT	RMB	1 1 2 2 2 1 2 RE MOVED DOWN 1 1 1 2 2 1 1 1 1 2 2 2 2 2 2 2 1 1 1 1 2 2 PICKS UP THE NI	*TV GRAPHIC BLOCK VALUE FOR SET, RESET AND POINT *TV INKEY\$ RAM IMAGE *PV CURSOR LOCATION *PV DUMMY - THESE TWO BYTES ARE ALWAYS ZERO *TV TONE VALUE FOR SOUND COMMAND *TV DURATION VALUE FOR SOUND COMMAND **PV UPPER LIMIT OF 1200 HERTZ PERIOD 10
0191 008 0192 008 0193 008 0194 008 0196 008 0197 008 0197 008 0198 0199 0200 0201 0202 008 0203 009 0204 009 0205 009 0206 009 0207 009 0208 009 0210 009 0211 009 0211 009 0212 009 0213 009 0214 0215 0216	5 6 7 8 8 A C C D F Ø 1 2 4 5 7 7 9 A B C D	CLSTSN GRBLOK IKEYIM CKEYIM CKEYIM SIKEYIM SIKEYIM CKEYIM SIKEYIM SIKAM SIKEYIM SIKAM SIKEYIM SIKEYIM SIKAM SIKAM SIKAM SIKAM SIKAM SIKAM SIKAM SIKAM SIKAM SI	RMB RMB RMB RMB RMB RMB YTES AI RMB RMB RMB RMB RMB RMB RMB RMB RMB RM	1 1 2 2 2 1 2 2 RE MOVED DOWN 1 1 1 2 1 1 1 2 2 1 1 1 1 2 PICKS UP THE NI DRESS OF THE NI	*TV GRAPHIC BLOCK VALUE FOR SET, RESET AND POINT *TV INKEY\$ RAM IMAGE *PV CURSOR LOCATION *PV DUMMY - THESE TWO BYTES ARE ALWAYS ZERO *TV TONE VALUE FOR SOUND COMMAND *TV DURATION VALUE FOR SOUND COMMAND *TV DURATION VALUE FOR SOUND COMMAND *TROM ROM INIT DESCRIPTION VALUE 18 *PV 1200/2400 HERTZ PARTITION 24 *PV UPPER LIMIT OF 1200 HERTZ PERIOD 10 *PV UPPER LIMIT OF 2400 HERTZ PERIOD 128 *PV NUMBER OF \$55'S TO CASSETTE LEADER 11 *PV CURSOR BLINK DELAY 88 *PV BAUD RATE CONSTANT (600) 1 *PV PRINTER CARRIAGE RETURN DELAY 16 *PV TAB FIELD WIDTH 112 *PV LAST TAB ZONE 132 *PV PRINTER WIDTH 0 *PV LINE PRINTER POSITION LB4AA *PV JUMP ADDRESS FOR EXEC COMMAND EXT INPUT CHARACTER FROM EXT BASIC BYTE TO BE
0191 008 0192 008 0193 008 0194 008 0195 008 0196 008 0197 008 0198 0199 0200 0201 0201 0202 008 0204 009 0205 009 0206 009 0207 009 0208 009 0210 009 0211 009 0212 009 0213 009 0214 0215 0216 0217	5 6 7 8 8 A C C D F Ø 1 2 4 5 7 7 9 A B C D	CLSTSN GRBLOK IKEYIM CKEYIM CKEYIM SIKEYIM SIKEYIM CKEYIM SIKEYIM SIKAM SIKEYIM SIKAM SIKEYIM SIKEYIM SIKAM SIKAM SIKAM SIKAM SIKAM SIKAM SIKAM SIKAM SIKAM SI	RMB RMB RMB RMB RMB RMB YTES AI RMB RMB RMB RMB RMB RMB RMB RMB RMB RM	1 1 2 2 2 1 2 RE MOVED DOWN 1 1 1 2 2 1 1 1 1 2 2 2 2 2 2 2 1 1 1 1 2 2 PICKS UP THE NI	*TV GRAPHIC BLOCK VALUE FOR SET, RESET AND POINT *TV INKEY\$ RAM IMAGE *PV CURSOR LOCATION *PV DUMMY - THESE TWO BYTES ARE ALWAYS ZERO *TV TONE VALUE FOR SOUND COMMAND *TV DURATION VALUE FOR SOUND COMMAND *TV DURATION VALUE FOR SOUND COMMAND *TROM ROM INIT DESCRIPTION VALUE 18 *PV 1200/2400 HERTZ PARTITION 24 *PV UPPER LIMIT OF 1200 HERTZ PERIOD 10 *PV UPPER LIMIT OF 2400 HERTZ PERIOD 128 *PV NUMBER OF \$55'S TO CASSETTE LEADER 11 *PV CURSOR BLINK DELAY 88 *PV BAUD RATE CONSTANT (600) 1 *PV PRINTER CARRIAGE RETURN DELAY 16 *PV TAB FIELD WIDTH 112 *PV LAST TAB ZONE 132 *PV PRINTER WIDTH 0 *PV LINE PRINTER POSITION LB4AA *PV JUMP ADDRESS FOR EXEC COMMAND EXT INPUT CHARACTER FROM EXT BASIC BYTE TO BE
0191 008 0192 008 0193 008 0194 008 0195 008 0197 008 0199 0200 0201 0202 008 0203 009 0204 009 0205 009 0206 009 0207 009 0208 009 0209 009 0211 009 0212 009 0213 009 0214 0215 0216 0217 0218	5 6 7 8 A C D F Ø 1 2 4 5 7 9 A B C D	CLSTSN GRBLOK IKEYIM CURPOS ZERO SNOTON SNDDUR ** THESE B *** CMPMID CMPØ CMP1 SYNCLN BLKCNT LPTBTD LPTLND LPTLVD LPTCFW LPTCFW LPTCFFW LPTLCF LPTWID LPTCFW ** THIS RO! ** BASIC. ** INTERPR	RMB RMB RMB RMB RMB RMB RMB YTES AI RMB RMB RMB RMB RMB RMB RMB RMB RMB RM	1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	*TV GRAPHIC BLOCK VALUE FOR SET, RESET AND POINT *TV INKEY\$ RAM IMAGE *PV CURSOR LOCATION *PV DUMMY - THESE TWO BYTES ARE ALWAYS ZERO *TV TONE VALUE FOR SOUND COMMAND *TV DURATION VALUE FOR SOUND COMMAND *TV DURATION VALUE FOR SOUND COMMAND *ROM ROM INIT DESCRIPTION VALUE 18 *PV 1200/2400 HERTZ PARTITION 24 *PV UPPER LIMIT OF 1200 HERTZ PERIOD 10 *PV UPPER LIMIT OF 2400 HERTZ PERIOD 128 *PV NUMBER OF \$55'S TO CASSETTE LEADER 11 *PV CURSOR BLINK DELAY 88 *PV BAUD RATE CONSTANT (600) 1 *PV PRINTER CARRIAGE RETURN DELAY 16 *PV TAB FIELD WIDTH 112 *PV LAST TAB ZONE 132 *PV PRINTER WIDTH 0 *PV LINE PRINTER POSITION LB4AA *PV JUMP ADDRESS FOR EXEC COMMAND EXT INPUT CHARACTER FROM EXT BASIC BYTE TO BE ARAD.
Ø191 Ø08 Ø192 Ø08 Ø193 Ø08 Ø194 Ø08 Ø195 Ø08 Ø197 Ø08 Ø198 Ø199 Ø2Ø0 Ø2Ø1 Ø2Ø2 Ø08 Ø2Ø3 Ø09 Ø2Ø4 Ø09 Ø2Ø5 Ø09 Ø2Ø7 Ø09 Ø2Ø9 Ø09 Ø210 Ø09 Ø211 Ø09 Ø213 Ø09 Ø214 Ø215 Ø216 Ø217 Ø217 Ø218 Ø219 Ø09	5 6 6 7 8 A C C D F Ø C A 7	CLSTSN GRBLOK IKEYIM CURPOS ZERO SNOTON SNODUR ** THESE B *** *CMPMID CMPØ CMP1 SYNCLN BLKCNT LPTBTD LPTLND LPTLND LPTCFW LPTCFW LPTCFW LPTCFW LPTCFW ** THIS RO! ** BASIC. ** INTERPR GETNCH	RMB RMB RMB RMB RMB RMB RMB YTES AI RMB RMB RMB RMB RMB RMB RMB RMB RMB RM	1 1 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2	*TV GRAPHIC BLOCK VALUE FOR SET, RESET AND POINT *TV INKEY\$ RAM IMAGE *PV CURSOR LOCATION *PV DUMMY - THESE TWO BYTES ARE ALWAYS ZERO *TV TONE VALUE FOR SOUND COMMAND *TV DURATION VALUE FOR SOUND COMMAND *TV DURATION VALUE FOR SOUND COMMAND *ROM ROM INIT DESCRIPTION VALUE 18 *PV 1200/2400 HERTZ PARTITION 24 *PV UPPER LIMIT OF 1200 HERTZ PERIOD 10 *PV UPPER LIMIT OF 2400 HERTZ PERIOD 128 *PV NUMBER OF \$55'S TO CASSETTE LEADER 11 *PV CURSOR BLINK DELAY 88 *PV BAUD RATE CONSTANT (600) 1 *PV PRINTER CARRIAGE RETURN DELAY 16 *PV TAB FIELD WIDTH 112 *PV LAST TAB ZONE 132 *PV PRINTER WIDTH Ø *PV LINE PRINTER POSITION LB4AA *PV JUMP ADDRESS FOR EXEC COMMAND EXT INPUT CHARACTER FROM EXT BASIC BYTE TO BE ARAD. *PV INCREMENT LS BYTE OF INPUT POINTER
0191 008 0192 008 0193 008 0194 008 0195 008 0197 008 0199 0200 0201 0202 008 0203 009 0204 009 0205 009 0206 009 0207 009 0208 009 0209 009 0211 009 0212 009 0213 009 0214 0215 0216 0217 0218	5 6 6 7 8 A C C D F Ø C A 7	CLSTSN GRBLOK IKEYIM CURPOS ZERO SNOTON SNODUR ** THESE B *** *CMPMID CMPØ CMP1 SYNCLN BLKCNT LPTBTD LPTLND LPTLND LPTCFW LPTCFW LPTCFW LPTCFW LPTCFW ** THIS RO! ** BASIC. ** INTERPR GETNCH	RMB RMB RMB RMB RMB RMB RMB YTES AI RMB RMB RMB RMB RMB RMB RMB RMB RMB RM	1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	*TV GRAPHIC BLOCK VALUE FOR SET, RESET AND POINT *TV INKEY\$ RAM IMAGE *PV CURSOR LOCATION *PV DUMMY - THESE TWO BYTES ARE ALWAYS ZERO *TV TONE VALUE FOR SOUND COMMAND *TV DURATION VALUE FOR SOUND COMMAND *TV DURATION VALUE FOR SOUND COMMAND *ROM ROM INIT DESCRIPTION VALUE 18 *PV 1200/2400 HERTZ PARTITION 24 *PV UPPER LIMIT OF 1200 HERTZ PERIOD 10 *PV UPPER LIMIT OF 2400 HERTZ PERIOD 128 *PV NUMBER OF \$55'S TO CASSETTE LEADER 11 *PV CURSOR BLINK DELAY 88 *PV BAUD RATE CONSTANT (600) 1 *PV PRINTER CARRIAGE RETURN DELAY 16 *PV TAB FIELD WIDTH 112 *PV LAST TAB ZONE 132 *PV PRINTER WIDTH 0 *PV LINE PRINTER POSITION LB4AA *PV JUMP ADDRESS FOR EXEC COMMAND EXT INPUT CHARACTER FROM EXT BASIC BYTE TO BE ARAD.
Ø191 Ø08 Ø192 Ø08 Ø193 Ø08 Ø194 Ø08 Ø195 Ø08 Ø197 Ø08 Ø198 Ø199 Ø2Ø0 Ø2Ø1 Ø2Ø2 Ø08 Ø2Ø3 Ø09 Ø2Ø4 Ø09 Ø2Ø5 Ø09 Ø2Ø7 Ø09 Ø2Ø9 Ø09 Ø210 Ø09 Ø211 Ø09 Ø213 Ø09 Ø214 Ø215 Ø216 Ø217 Ø217 Ø218 Ø219 Ø09	5 6 6 7 8 8 A C C D F Ø 1 1 2 2 4 4 5 5 7 9 A B C C D D	CLSTSN GRBLOK IKEYIM CURPPOS ZERO SNDTON SNDDUR ** THESE B ** ** CMPMID CMPØ CMPI SYNCLN BLKCNT LPTBTD LPTLND LPTLND LPTLCF LPTWID LPTLOF LPTWID LPTPOS EXECJP ** THIS RO ** BASIC. ** INTERPR GETNCH	RMB RMB RMB RMB RMB RMB RMB YTES AI RMB RMB RMB RMB RMB RMB RMB RMB RMB RM	1 1 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2	*TV GRAPHIC BLOCK VALUE FOR SET, RESET AND POINT *TV INKEY\$ RAM IMAGE *PV CURSOR LOCATION *PV DUMMY - THESE TWO BYTES ARE ALWAYS ZERO *TV TONE VALUE FOR SOUND COMMAND *TV DURATION VALUE FOR SOUND COMMAND *TV DURATION VALUE FOR SOUND COMMAND *ROM ROM INIT DESCRIPTION VALUE 18 *PV 1200/2400 HERTZ PARTITION 24 *PV UPPER LIMIT OF 1200 HERTZ PERIOD 10 *PV UPPER LIMIT OF 2400 HERTZ PERIOD 128 *PV NUMBER OF \$55'S TO CASSETTE LEADER 11 *PV CURSOR BLINK DELAY 88 *PV BAUD RATE CONSTANT (600) 1 *PV PRINTER CARRIAGE RETURN DELAY 16 *PV TAB FIELD WIDTH 112 *PV LAST TAB ZONE 132 *PV PRINTER WIDTH Ø *PV LINE PRINTER POSITION LB4AA *PV JUMP ADDRESS FOR EXEC COMMAND EXT INPUT CHARACTER FROM EXT BASIC BYTE TO BE ARAD. *PV INCREMENT LS BYTE OF INPUT POINTER
0191 008 0192 008 0193 008 0194 008 0195 008 0197 008 0198 0199 0200 0201 0202 008 0203 009 0204 009 0205 009 0206 009 0207 009 0208 009 0210 009 0211 009 0212 009 0213 009 0214 0215 0216 0217 0218 0219 009 0220 00A	5 6 6 7 8 8 A C C D F Ø 1 1 2 2 4 4 5 5 7 9 9 A B C C D D F Ø 2 3 Ø C A 6	CLSTSN GRBLOK IKEYIM CKEYIM CKEYIM CKEYIM CKEYIM SIKEYIM SIKAY SIKEYIM SIKAY SIKEYIM SIKEYIM SIKEYIM SIKEYIM SIKAY SIKAY SIKEYIM SIKAY SIKEYIM	RMB	1 1 2 2 2 1 1 2 2 2 1 1 2 2 2 1 1 1 2 2 2 1 1 1 1 1 2 2 2 1 1 1 1 1 2 2 2 2 1 1 1 1 1 1 2 2 2 2 1 1 1 1 1 1 2 2 2 2 1 1 1 1 1 1 2 2 2 2 1 1 1 1 1 1 2 2 2 2 1 1 1 1 1 1 1 2 2 2 2 2 1	*TV GRAPHIC BLOCK VALUE FOR SET, RESET AND POINT *TV INKEY\$ RAM IMAGE *PV CURSOR LOCATION *PV DUMMY - THESE TWO BYTES ARE ALWAYS ZERO *TV TONE VALUE FOR SOUND COMMAND *TV DURATION VALUE FOR SOUND COMMAND *TO DURATION VALUE FOR SOUND COMMAND **PV UPPER LIMIT OF 1200 HERTZ PERIOD 10 *PV UPPER LIMIT OF 2400 HERTZ PERIOD 128 *PV NUMBER OF \$55'S TO CASSETTE LEADER 11 *PV CURSOR BLINK DELAY 88 *PV BAUD RATE CONSTANT (600) 1 *PV PRINTER CARRIAGE RETURN DELAY 16 *PV TAB FIELD WIDTH 112 *PV LAST TAB ZONE 132 *PV PRINTER WIDTH 0 *PV LINE PRINTER POSITION LB4AA *PV JUMP ADDRESS FOR EXEC COMMAND EXT INPUT CHARACTER FROM EXT BASIC BYTE TO BE ARAD. *PV INCREMENT LS BYTE OF INPUT POINTER *PV BRANCH IF NOT ZERO (NO CARRY)

	ØØA6	CHARAD		2	*PV THESE 2 BYTES CONTAIN ADDRESS OF THE CURRENT
Ø224		*			CHARACTER WHICH THE BASIC INTERPRETER IS
Ø225	~~ 75	*	71.45		PROCESSING
	ØØA8 7E AA 1A		JMP	BROMHK	JUMP BACK INTO THE BASIC RUM
Ø227	aavb	VAD	DMD	1	= LOW ORDER FOUR BYTES OF THE PRODUCT
	ØØAB		RMB	1	
	ØØAC ØØAD		RMB	1 1	= OF A FLOATING POINT MULTIPLICATION - THESE BYTES ARE USE AS DANDOM DATA
			RMB		= THESE BYTES ARE USE AS RANDOM DATA
Ø231	ØØAE	VAE	RMB	1	= BY THE RND STATEMENT
Ø233		* EXTENDED	DVCIC	VADTABLES	
	ØØAF		RMB	1	*PV TRACE FLAG Ø=OFF ELSE=ON
	ØØBØ		RMB	2	*PV ADDRESS OF THE START OF USR VECTORS
	ØØB2		RMB	1	*PV FOREGROUND COLOR
	ØØB3		RMB	1	*PV BACKGROUND COLOR
	ØØB4		RMB	1	*TV WORKING COLOR BEING USED BY EX BASIC
	ØØB5		RMB	1	*TV ALL PIXELS IN THIS BYTE SET TO COLOR OF VB3
	ØØB6		RMB	1	*PV PMODE'S MODE ARGUMENT
Ø241			RMB	2	*PV END OF CURRENT GRAPHIC PAGE
	ØØB9		RMB	1	*PV NUMBER OF BYTES/HORIZONTAL GRAPHIC LINE
	ØØBA		RMB	2	*PV START OF CURRENT GRAPHIC PAGE
	ØØBC		RMB	1	*PV START OF GRAPHIC RAM (MS BYTE)
Ø245	ØØBD	HORBEG	RMB	2	*DV* *PV HORIZ COORD - START POINT
Ø246	ØØBF	VERBEG	RMB	2	*DV* *PV VERT COORD - START POINT
Ø247	ØØC1	CSSYAL	RMB	1	*PV SCREEN'S COLOR SET ARGUMENT
Ø248	ØØC2	SETFLG	RMB	1	*PV PRESET/PSET FLAG: Ø=PRESET, 1=PSET
Ø249	ØØC3	HOREND	RMB	2	*DV* *PV HORIZ COORD - ENDING POINT
Ø25Ø	ØØC5	VEREND	RMB	2	*DV* *PV VERT COORD - ENDING POINT
Ø251	ØØC7	HORDEF	RMB	2	*PV HORIZ COORD - DEFAULT COORD
	ØØC9	VERDEF	RMB	2	*PV VERT COORD - DEFAULT COORD
Ø253					
Ø254				SCRATCH PAD VAF	RIABLES
	ØØCB		RMB	2	
Ø256			RMB	2	
	ØØCF		RMB	2	
Ø258			RMB	2	
	ØØD3		RMB	1	
	ØØD4 ØØD5		RMB RMB	1 1	
	ØØD6		RMB	1	
	ØØD7		RMB	1	
	ØØD8		RMB	1	
	ØØD9		RMB	1	
Ø266			RMB	1	
Ø267					
Ø268	ØØDB	CHGFLG	RMB	1	*TV FLAG TO INDICATE IF GRAPHIC DATA HAS BEEN CHANGED
Ø269	ØØDC	TMPSTK	RMB	2	*TV STACK POINTER STORAGE DURING PAINT
Ø27Ø	ØØDE	OCTAVE	RMB	1	*PV OCTAVE VALUE (PLAY)
Ø271	ØØDF	VOLHI	RMB	1	*DV* *PV VOLUME HIGH VALUE (PLAY)
	ØØEØ		RMB	1	*DV* *PV VOLUME LOW VALUE (PLAY)
Ø273		NOTELN	RMB	1	*PV NOTE LENGTH (PLAY)
	ØØE2			1	*PV TEMPO VALUE (PLAY)
	ØØE3		RMB	2	*TV TIMER FOR THE PLAY COMMAND
	ØØE5			1	*TV DOTTED NOTE TIMER SCALE FACTOR
	ØØE6		EQU	*	SUPER EXTENDED BASIC HI-RES MODE
	ØØE6		RMB	1	*DV* *PV DLOAD BAUD RATE CONSTANT \$BØ=300, \$2C=1200
	00E7		EQU	*	SUPER EXTENDED BASIC HI-RES TEXT MODE
	ØØE7		RMB	1	*DV* *PV DLOAD TIMEOUT CONSTANT
	ØØE8		RMB	1	*DV* *PV ANGLE VALUE (DRAW)
	ØØE9	SCALE	RMB	1	*DV* *PV SCALE VALUE (DRAW)
Ø283 Ø284		* DSKCON VA	ARTARI	FS	
	ØØEA		RMB	.ES 1	*PV DSKCON OPERATION CODE Ø-3
	ØØEB			1	*PV DSKCON OPERATION CODE 0-3 *PV DSKCON DRIVE NUMBER Ø 3
	ØØEC		RMB	1	*PV DSKCON TRACK NUMBER Ø 34
	ØØED			1	*PV DSKCON TRACK NUMBER 0 34 *PV DSKCON SECTOR NUMBER 1-18
	ØØEE		RMB	2	*PV DSKCON DATA POINTER
	ØØFØ		RMB	1	*PV DSKCON STATUS BYTE
Ø291		**···			
	ØØF1	FCBTMP	RMB	2	TEMPORARY FCB POINTER
Ø293					
	ØØF3		RMB	13	SPARE: UNUSED VARIABLES
Ø295					
Ø296					

Ø297	*		BASIC EXBASI(DOSBASIC
0298	CHONEC BAD	2	AVVVV AVVVV ADDOD CLITO VECTOR
0299 0100	SW3VEC RMB	3	\$XXXX \$XXXX \$3B3B SWI3 VECTOR
0300 0103	SW2VEC RMB SWIVEC RMB	3	\$XXXX \$XXXX \$3B3B SWI2 VECTOR
0301 0106 0302 0109	SWIVEC RMB NMIVEC RMB	3 3	\$XXXX \$XXXX \$XXXX SWI VECTOR \$XXXX \$XXXX \$D7AE NMI VECTOR
0302 0109 0303 010C	IRQVEC RMB	3	\$A9B3 \$894C \$D7BC IRQ VECTOR
0304 010F	FRQVEC RMB	3	\$AØF6 \$AØF6 \$AØF6 FIRQ VECTOR
Ø3Ø5	TRQVEC KIID	3	TABLO TABLO LING VECTOR
Ø3Ø6 Ø112	TIMVAL		
Ø3Ø7 Ø112	USRJMP RMB	3	JUMP ADDRESS FOR BASIC'S USR FUNCTION
0308	* RMB	2	TIMER VALUE FOR EXBAS
Ø3Ø9	* RMB	1	UNUSED BY EXBAS OR DISK BASIC
0310 0115	RVSEED RMB	1	* FLOATING POINT RANDOM NUMBER SEED EXPONENT
Ø311 Ø116	RMB	4	* MANTISSA: INITIALLY SET TO \$804FC75259
Ø312 Ø11A	CASFLG RMB	1	UPPER CASE/LOWER CASE FLAG: \$FF=UPPER, Ø=LOWER
Ø313 Ø11B	DEBVAL RMB	2	KEYBOARD DEBOUNCE DELAY (SET TO \$45E)
Ø314 Ø11D	EXPJMP RMB	3	JUMP ADDRESS FOR EXPONENTIATION
0315	**		INITIALLY SET TO ERROR FOR BASIC, \$8489 FOR EX BASIC
Ø316	*** COMM	AND INTERDRETATION	ON VECTOR TARLE
Ø317 Ø318	^^^ CUMM	AND INTERPRETATION	JN VECTOR TABLE
Ø319	** EUIID CETC UE	10 BYTE TABLES:	
Ø32Ø	TOOK SETS OF	IN DITE TABLES.	
Ø321			
Ø322	** THE LAST USE	ED TABLE MUST BE	FOLLOWED BY A ZERO BYTE
Ø323			ND 8,9) POINT TO THE JUMP TABLE FOR
0324			R TABLES, THESE VECTORS POINT TO A
0325	* ROUTINE WHICH	H WILL VECTOR YOU	TO THE CORRECT JUMP TABLE.
0326			FIED THIS SCHEME SO THAT THE USER
Ø327			NY ADDITIONAL TABLES WILL HAVE TO BE
0328	* ACCESSED FROM	1 A NEW COMMAND H	ANDLER.
0329			
0330		DESCRIPTION	NACE HODGE
Ø331	* Ø	NUMBER OF RESER	
Ø332	* 1,2 * 3,4	LOOKUP TABLE OF	
Ø333 Ø334	^ 3,4 *		COMMANDS (FIRST TABLE) ISION COMMAND HANDLERS (ALL BUT FIRST TABLE)
Ø335	* 5	NUMBER OF SECON	
Ø336	* 6,7		SECONDARY FUNCTIONS (FIRST TABLE)
Ø337	*		ISION SECONDARY COMMAND HANDLERS (ALL BUT
Ø338	*	FIRST TABLE)	
Ø339	* 8,9	JUMP TABLE FOR	SECONDARY FUNCTIONS
0340	* 10	Ø BYTE - END OF	TABLE FLAG (LAST TABLE ONLY)
Ø341			
0342 0120	COMVEC RMB	10	BASIC'S TABLE
Ø343 Ø12A	RMB	10	EX BASIC'S TABLE
0344 0134	RMB	10	DISC BASIC'S TABLE (UNUSED BY EX BASIC)
Ø345	++++ UCD FUNCTI	ON VECTOR ADDRES	SES (EX BASIC ONLY)
Ø346 Ø347 Ø13E	RMB		USR Ø VECTOR
Ø348 Ø14Ø	RMB	2	USR 1
Ø349 Ø142	RMB	2	USR 2
Ø35Ø Ø144	RMB	2	USR 3
Ø351 Ø146	RMB	2	USR 4
Ø352 Ø148	RMB	2	USR 5
Ø353 Ø14A	RMB	2	USR 6
Ø354 Ø14C	RMB	2	USR 7
Ø355 Ø14E	RMB	2	USR 8
Ø356 Ø15Ø	RMB	2	USR 9
Ø357	*** THE *DOVE (OR DATE HOS ASSS	VECTOR TABLE IC MOVER TO
Ø358			VECTOR TABLE IS MOVED TO E 20 BYTES FROM \$13E-\$151
Ø359 Ø36Ø	*** ARE REDEFIN		IF TR DITES LVOLI 4TOF-4TOT
Ø361	AND NEDELL	ILD NO TULLUMO:	
Ø362	* RMB	10	USER (SPARE) COMMAND INTERPRETATION TABLE SPACE
Ø363	* FCB	Ø	END OF COMM INTERP TABLE FLAG
Ø364	* RMB	9	UNUSED BY DISK BASIC
Ø365			
Ø366		AND INTERPRETATION	
Ø367	*	BYTE	BASIC EX BASIDISK BASIC
Ø368	*	0	53 BASIC TABLE
Ø369 Ø37Ø	*	1,2 3,4	\$AA66 \$AB67
พง/พ		J, T	₹₽₽0 7

Ø371	*	5	20				
0372	*	6,7	\$AB1A				
Ø373	*	8,9	\$AA29				
Ø374		-,-					
Ø375	*	Ø		25		FY RAS	IC TABLE
	*			\$8183		LA DAS	TO TABLE
Ø376		1,2			+0505	/+0=01	0.43
Ø377	*	3,4			\$CE2E	(\$CFØA	2.1)
Ø378	*	5		14			
Ø379	*	6,7		\$821E			
Ø38Ø	*	8,9		\$8168	\$CE56	(\$CF32	2.1)
Ø381		3,3		10100	+0200	(+ 0 . 0 _	
	*	a		10 (20	2 1)	DICK D	ACIC TABLE
0382		Ø		19 (20	2.1)	DI2K R	ASIC TABLE
Ø383	*	1,2		\$C17F			
Ø384	*	3,4		\$C2CØ			
Ø385	*	5		6			
Ø386	*	6,7		\$C2Ø1			
Ø387	*	8,9		\$C236			
		0,3		¥6230			
Ø388							
Ø389							
Ø39Ø Ø152	KEYBUF	RMB 8	KEYBOA	RD MEMO	RY BUFF	FER	
Ø391 Ø15A	POTVAL	RMB 1	LEFT V	ERTICAL	. JOYSTI	CK DATA	1
Ø392 Ø15B		RMB 1	LEFT H	ORTZONT	AL JOYS	STICK DA	ΔΤΔ
Ø393 Ø15C		RMB 1				TICK DAT	
Ø394 Ø15D		RMB 1	KIGHI	HOKIZUN	IIAL JUI	/STICK [אוא
0395							
Ø396		RAM VECTORS – INITIA					
Ø397	* 25 SETS C	OF 3 BYTE INSTRUCTIO	NS WHICH	ARE CA	LLED BY	COLOR	BASIC
Ø398	* EXTENDED	AND DISK BASIC. THE	IR PURPO	SE IS T	O ALLOW	/ ENHANC	CEMENTS (SUCH
Ø399		SIC AND DOS BASIC) A					
0400		FFECTIVELY ALLOWI					rue
0401		IN EARLIER ROMS. TH					
0402	* AND THE A	ADDRESS TO GET TO TH	E NEW CO	DE IS I	N BYTES	5 1 & 2	OF THE
Ø4Ø3	* RAM VECTO	DR. BYTE Ø WILL CONT	AIN A \$7	E WHICH	IS THE	FIRST	BYTE OF
0404	* THE JMP T	INSTRUCTION.					
0405		Γ ADDRESS IN THIS TA	RIF TS T	HE VUUD	FSS TN	BASTC W	IHICH
0406		RAM VECTOR, THE SE					
		PUTS IN THE RAM VEC	TOR (TE	ANY) AN	D THE T	LIBN AL	
0407	" EX DASIC	TOTO IN THE NAME VEC	1010 (11	,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	D 1111E 1	III KD AL	JUKESS
0407 0408		ALUE WHICH DISK BASI				IIIKD AL	DUKE33
						IIIKU AL	JUKESS
Ø4Ø8 Ø4Ø9						IIIKD AL	JUNE 33
0408 0409 0410	* IS THE VA		C PUTS T	HERE (I	F ANY)		JUNESS
0408 0409 0410 0411	* IS THE VA	ALUE WHICH DISK BASI	C PUTS T		1.Ø	1.1	
0408 0409 0410 0411 0412 015E	* IS THE VA * RVECØ	ALUE WHICH DISK BASI RMB 3	2.Ø \$A5F6	HERE (I	1.Ø \$C426	1.1 \$C44B	OPEN COMMAND
0408 0409 0410 0411 0412 015E 0413 0161	* IS THE VA * RVECØ RVEC1	ALUE WHICH DISK BASI RMB 3 RMB 3	2.Ø \$A5F6 \$A5B9	HERE (I	1.Ø \$C426 \$C838	1.1 \$C44B \$C888	OPEN COMMAND DEVICE NUMBER VALIDITY CHECK
0408 0409 0410 0411 0412 015E	* IS THE VA * RVECØ RVEC1	ALUE WHICH DISK BASI RMB 3	2.Ø \$A5F6	HERE (I	1.Ø \$C426	1.1 \$C44B	OPEN COMMAND
0408 0409 0410 0411 0412 015E 0413 0161	* IS THE VA * RVECØ RVEC1 RVEC2	ALUE WHICH DISK BASI RMB 3 RMB 3	2.Ø \$A5F6 \$A5B9	HERE (I	1.Ø \$C426 \$C838	1.1 \$C44B \$C888	OPEN COMMAND DEVICE NUMBER VALIDITY CHECK
0408 0409 0410 0411 0412 015E 0413 0161 0414 0164 0415 0167	* IS THE VA * RVECØ RVEC1 RVEC2 RVEC3	ALUE WHICH DISK BASI RMB 3 RMB 3 RMB 3 RMB 3 RMB 3	2.Ø \$A5F6 \$A5B9 \$A35F \$A282	HERE (I 2.1 \$8273	1.Ø \$C426 \$C838 \$C843 \$CB4A	1.1 \$C44B \$C888 \$C893 \$CC1C	OPEN COMMAND DEVICE NUMBER VALIDITY CHECK SET PRINT PARAMETERS CONSOLE OUT
0408 0409 0410 0411 0412 015E 0413 0161 0414 0164 0415 0167 0416 016A	* IS THE VA * RVECØ RVEC1 RVEC2 RVEC3 RVEC4	ALUE WHICH DISK BASI RMB 3 RMB 3 RMB 3 RMB 3 RMB 3 RMB 3	2.Ø \$A5F6 \$A5B9 \$A35F \$A282 \$A176	HERE (I	1.Ø \$C426 \$C838 \$C843 \$CB4A \$C58F	1.1 \$C44B \$C888 \$C893 \$CC1C \$C5BC	OPEN COMMAND DEVICE NUMBER VALIDITY CHECK SET PRINT PARAMETERS CONSOLE OUT CONSOLE IN
0408 0409 0410 0411 0412 015E 0413 0161 0414 0164 0415 0167 0416 016A 0417 016D	* IS THE VA * RVECØ RVEC1 RVEC2 RVEC3 RVEC4 RVEC5	RMB 3	2.Ø \$A5F6 \$A5B9 \$A35F \$A282 \$A176 \$A3ED	HERE (I 2.1 \$8273	1.Ø \$C426 \$C838 \$C843 \$CB4A \$C58F \$C818	1.1 \$C44B \$C888 \$C893 \$CC1C \$C5BC \$C58C	OPEN COMMAND DEVICE NUMBER VALIDITY CHECK SET PRINT PARAMETERS CONSOLE OUT CONSOLE IN INPUT DEVICE NUMBER CHECK
0408 0409 0410 0411 0412 015E 0413 0161 0414 0164 0415 0167 0416 016A 0417 016D 0418 0170	* IS THE VA * RVECØ RVEC1 RVEC2 RVEC3 RVEC4 RVEC5 RVEC6	RMB 3	2.Ø \$A5F6 \$A5B9 \$A35F \$A282 \$A176 \$A3ED \$A4Ø6	HERE (I 2.1 \$8273	1.Ø \$C426 \$C838 \$C843 \$CB4A \$C58F \$C818 \$C81B	1.1 \$C44B \$C888 \$C893 \$CC1C \$C5BC \$C848 \$C84B	OPEN COMMAND DEVICE NUMBER VALIDITY CHECK SET PRINT PARAMETERS CONSOLE OUT CONSOLE IN INPUT DEVICE NUMBER CHECK PRINT DEVICE NUMBER CHECK
0408 0409 0410 0411 0412 015E 0413 0161 0414 0164 0415 0167 0416 016A 0417 016D 0418 0170 0419 0173	* IS THE VA * RVECØ RVEC1 RVEC2 RVEC3 RVEC4 RVEC5 RVEC6 RVEC6	RMB 3	2.Ø \$A5F6 \$A5B9 \$A35F \$A282 \$A176 \$A3ED \$A406 \$A426	#ERE (I 2.1 \$8273 \$8CF1	1.Ø \$C426 \$C838 \$C843 \$CB4A \$C58F \$C818 \$C81B \$CA3B	1.1 \$C44B \$C888 \$C893 \$CC1C \$C5BC \$C848 \$C84B \$CAE9	OPEN COMMAND DEVICE NUMBER VALIDITY CHECK SET PRINT PARAMETERS CONSOLE OUT CONSOLE IN INPUT DEVICE NUMBER CHECK PRINT DEVICE NUMBER CHECK CLOSE ALL FILES
0408 0409 0410 0411 0412 015E 0413 0161 0414 0164 0415 0167 0416 016A 0417 016D 0418 0170	* IS THE VA * RVECØ RVEC1 RVEC2 RVEC3 RVEC4 RVEC5 RVEC6 RVEC6	RMB 3	2.Ø \$A5F6 \$A5B9 \$A35F \$A282 \$A176 \$A3ED \$A4Ø6	HERE (I 2.1 \$8273	1.Ø \$C426 \$C838 \$C843 \$CB4A \$C58F \$C818 \$C81B	1.1 \$C44B \$C888 \$C893 \$CC1C \$C5BC \$C848 \$C84B	OPEN COMMAND DEVICE NUMBER VALIDITY CHECK SET PRINT PARAMETERS CONSOLE OUT CONSOLE IN INPUT DEVICE NUMBER CHECK PRINT DEVICE NUMBER CHECK
0408 0409 0410 0411 0412 015E 0413 0161 0414 0164 0415 0167 0416 016A 0417 016D 0418 0170 0419 0173	* IS THE VA * RVECØ RVEC1 RVEC3 RVEC4 RVEC5 RVEC6 RVEC6 RVEC7 RVEC8	RMB 3	2.Ø \$A5F6 \$A5B9 \$A35F \$A282 \$A176 \$A3ED \$A406 \$A426	#ERE (I 2.1 \$8273 \$8CF1	1.Ø \$C426 \$C838 \$C843 \$CB4A \$C58F \$C818 \$C81B \$CA3B	1.1 \$C44B \$C888 \$C893 \$CC1C \$C5BC \$C848 \$C84B \$CAE9	OPEN COMMAND DEVICE NUMBER VALIDITY CHECK SET PRINT PARAMETERS CONSOLE OUT CONSOLE IN INPUT DEVICE NUMBER CHECK PRINT DEVICE NUMBER CHECK CLOSE ALL FILES
0408 0409 0410 0411 0412 015E 0413 0161 0414 0164 0415 0167 0416 016A 0417 016D 0418 0170 0419 0173 0420 0176 0421 0179	* IS THE VA * RVECØ RVEC1 RVEC3 RVEC4 RVEC5 RVEC6 RVEC6 RVEC6 RVEC7 RVEC8 RVEC9	RMB 3	2.Ø \$A5F6 \$A5B9 \$A35F \$A282 \$A176 \$A3ED \$A4Ø6 \$A426 \$A42D \$B918	#8273 \$8273 \$80F1	1.0 \$C426 \$C838 \$C843 \$C58F \$C58F \$C818 \$C818 \$CA3B \$CA4B \$8E90	1.1 \$C44B \$C888 \$C893 \$CC1C \$C5BC \$C84B \$C84B \$CAE9 \$CAF9	OPEN COMMAND DEVICE NUMBER VALIDITY CHECK SET PRINT PARAMETERS CONSOLE OUT CONSOLE IN INPUT DEVICE NUMBER CHECK PRINT DEVICE NUMBER CHECK CLOSE ALL FILES CLOSE ONE FILE PRINT
0408 0409 0410 0411 0412 015E 0413 0161 0414 0164 0415 0167 0416 016A 0417 016D 0418 0170 0419 0173 0420 0176 0421 0179	* IS THE VA * RVECØ RVEC1 RVEC3 RVEC4 RVEC5 RVEC6 RVEC6 RVEC7 RVEC8 RVEC9 RVEC1Ø	RMB 3	2.0 \$A5F6 \$A5B9 \$A35F \$A282 \$A176 \$A3ED \$A406 \$A42D \$B918 \$B061	#8273 \$8273 \$80F1	1.Ø \$C426 \$C838 \$C843 \$C844 \$C58F \$C818 \$C818 \$C818 \$C828 \$C848 \$C828 \$C628 \$C	1.1 \$C44B \$C888 \$C893 \$CC1C \$C5BC \$C84B \$CA4B \$CAE9 \$CAE9 \$CAE9	OPEN COMMAND DEVICE NUMBER VALIDITY CHECK SET PRINT PARAMETERS CONSOLE OUT CONSOLE IN INPUT DEVICE NUMBER CHECK PRINT DEVICE NUMBER CHECK CLOSE ALL FILES CLOSE ONE FILE PRINT
0408 0409 0410 0411 0412 015E 0413 0161 0414 0164 0415 0167 0416 016A 0417 016D 0418 0170 0419 0173 0420 0176 0421 0179 0423 017F	* IS THE VA * RVECØ RVEC1 RVEC2 RVEC3 RVEC4 RVEC5 RVEC6 RVEC7 RVEC8 RVEC9 RVEC1Ø RVEC11	RMB 3	2.0 \$A5F6 \$A5B9 \$A35F \$A282 \$A176 \$A3ED \$A4406 \$A426 \$A426 \$B918 \$B061 \$A549	#8273 \$8273 \$80F1	1.0 \$C426 \$C838 \$C843 \$C844 \$C58F \$C818 \$CA3B \$CA3B \$CA3B \$CA3B \$CC58 \$CC58 \$CC58 \$CC58	1.1 \$C44B \$C888 \$C893 \$CC1C \$C5BC \$C84B \$CAE9 \$CAE9 \$CAE9 \$CAE9 \$CB9 \$CB9	OPEN COMMAND DEVICE NUMBER VALIDITY CHECK SET PRINT PARAMETERS CONSOLE OUT CONSOLE IN INPUT DEVICE NUMBER CHECK PRINT DEVICE NUMBER CHECK CLOSE ALL FILES CLOSE ONE FILE PRINT INPUT BREAK CHECK
0408 0409 0410 0411 0412 015E 0413 0161 0414 0164 0415 0167 0416 016A 0417 016D 0418 0170 0419 0173 0420 0176 0421 0179 0422 017C 0423 017F 0424 0182	* IS THE VA * RVECØ RVEC1 RVEC2 RVEC3 RVEC5 RVEC6 RVEC7 RVEC6 RVEC7 RVEC8 RVEC9 RVEC10 RVEC11 RVEC12	RMB 3	2.0 \$A5F6 \$A5B9 \$A35F \$A282 \$A176 \$A3ED \$A406 \$A426 \$B918 \$B061 \$A549 \$A390	#8273 \$8273 \$80F1	1.0 \$C426 \$C838 \$C848 \$C58F \$C818 \$CA3B \$CA4B \$CC58 \$CC58 \$CC58 \$CC58 \$CC58 \$CC58 \$CC58 \$CC687	1.1 \$C44B \$C888 \$C893 \$CC1C \$C5BC \$C84B \$CAE9 \$CAE9 \$CAE9 \$CAE9 \$CD35 \$CBA9 \$CCE4	OPEN COMMAND DEVICE NUMBER VALIDITY CHECK SET PRINT PARAMETERS CONSOLE OUT CONSOLE IN INPUT DEVICE NUMBER CHECK PRINT DEVICE NUMBER CHECK CLOSE ALL FILES CLOSE ONE FILE PRINT INPUT BREAK CHECK INPUTTING A BASIC LINE
0408 0409 0410 0411 0412 015E 0413 0161 0414 0164 0415 0167 0416 016A 0417 016D 0418 0170 0419 0173 0420 0176 0421 0179 0422 017C 0423 017F 0424 0182 0425 0185	* IS THE VA * RVECØ RVEC1 RVEC2 RVEC3 RVEC5 RVEC6 RVEC7 RVEC6 RVEC7 RVEC8 RVEC9 RVEC10 RVEC11 RVEC12 RVEC13	RMB 3	2.Ø \$A5F6 \$A5B9 \$A35F \$A282 \$A176 \$A4406 \$A420 \$B918 \$B918 \$B918 \$B649 \$A39Ø \$A4BF	#8273 \$8273 \$80F1	1.0 \$C426 \$C838 \$C843 \$C844 \$C818 \$C818 \$CA4B \$CA4B \$C658 \$C658 \$C658 \$C657 \$C636	1.1 \$C44B \$C888 \$C893 \$CC1C \$C5BC \$C84B \$C84B \$CAE9 \$CAF9 \$E9Ø \$CD5 \$C848 \$CAE9	OPEN COMMAND DEVICE NUMBER VALIDITY CHECK SET PRINT PARAMETERS CONSOLE OUT CONSOLE IN INPUT DEVICE NUMBER CHECK PRINT DEVICE NUMBER CHECK CLOSE ALL FILES CLOSE ONE FILE PRINT INPUT BREAK CHECK INPUTTING A BASIC LINE TERMINATING BASIC LINE INPUT
0408 0409 0411 0411 0412 015E 0413 0161 0414 0164 0415 0167 0416 016A 0417 016D 0418 0170 0419 0173 0420 0176 0421 0179 0422 017C 0423 017F 0423 017F 0425 0185 0426 0188	* IS THE VA * RVECØ RVEC1 RVEC2 RVEC3 RVEC4 RVEC5 RVEC6 RVEC7 RVEC8 RVEC1 RVEC10 RVEC11 RVEC12 RVEC13 RVEC14	RMB 3	2.0 \$A5F6 \$A5B9 \$A35F \$A282 \$A176 \$A406 \$A42D \$B918 \$B061 \$A399 \$A4BF \$A5CE	\$8273 \$8CF1 \$8286 \$8E9Ø	1.0	1.1 \$C44B \$C888 \$C893 \$CC1C \$C5BC \$C5BC \$C84B \$CAE9 \$CAE9 \$CAE9 \$CAE9 \$CD35 \$CC8A9 \$C6E4 \$CAE4	OPEN COMMAND DEVICE NUMBER VALIDITY CHECK SET PRINT PARAMETERS CONSOLE OUT CONSOLE IN INPUT DEVICE NUMBER CHECK PRINT DEVICE NUMBER CHECK CLOSE ALL FILES CLOSE ONE FILE PRINT INPUT BREAK CHECK INPUTTING A BASIC LINE TERMINATING BASIC LINE INPUT EOF COMMAND
0408 0409 0410 0411 0412 015E 0413 0161 0414 0164 0415 0167 0416 016A 0417 016D 0418 0170 0419 0173 0420 0176 0421 0179 0422 017C 0423 017F 0424 0182 0425 0185	* IS THE VA * RVECØ RVEC1 RVEC2 RVEC3 RVEC4 RVEC5 RVEC6 RVEC7 RVEC8 RVEC1 RVEC10 RVEC11 RVEC12 RVEC13 RVEC14	RMB 3	2.0 \$A5F6 \$A5B9 \$A35F \$A282 \$A176 \$A406 \$A42D \$B918 \$B061 \$A399 \$A4BF \$A5CE	#8273 \$8273 \$80F1	1.0	1.1 \$C44B \$C888 \$C893 \$CC1C \$C5BC \$C84B \$C84B \$CAE9 \$CAF9 \$E9Ø \$CD5 \$C848 \$CAE9	OPEN COMMAND DEVICE NUMBER VALIDITY CHECK SET PRINT PARAMETERS CONSOLE OUT CONSOLE IN INPUT DEVICE NUMBER CHECK PRINT DEVICE NUMBER CHECK CLOSE ALL FILES CLOSE ONE FILE PRINT INPUT BREAK CHECK INPUTTING A BASIC LINE TERMINATING BASIC LINE INPUT EOF COMMAND
0408 0409 0411 0411 0412 015E 0413 0161 0414 0164 0415 0167 0416 016A 0417 016D 0418 0170 0419 0173 0420 0176 0421 0179 0422 017C 0423 017F 0423 017F 0425 0185 0426 0188	* IS THE VA * RVECØ RVEC1 RVEC2 RVEC3 RVEC4 RVEC5 RVEC6 RVEC7 RVEC8 RVEC9 RVEC1Ø RVEC11 RVEC11 RVEC12 RVEC14 RVEC14	RMB 3	2.0 \$A5F6 \$A5B9 \$A35F \$A282 \$A176 \$A406 \$A42D \$B918 \$B061 \$A399 \$A4BF \$A5CE	\$8273 \$8CF1 \$8286 \$8E9Ø	1.0 \$C426 \$C838 \$C843 \$C844 \$C58F \$C818 \$C818 \$CA3B \$CA4B \$8E90 \$CC5B \$C659 \$C665 \$CA60 \$CDF6	1.1 \$C44B \$C888 \$C893 \$CC1C \$C5BC \$C5BC \$C84B \$CAE9 \$CAE9 \$CAE9 \$CAE9 \$CD35 \$CC8A9 \$C6E4 \$CAE4	OPEN COMMAND DEVICE NUMBER VALIDITY CHECK SET PRINT PARAMETERS CONSOLE OUT CONSOLE IN INPUT DEVICE NUMBER CHECK PRINT DEVICE NUMBER CHECK CLOSE ALL FILES CLOSE ONE FILE PRINT INPUT BREAK CHECK INPUTTING A BASIC LINE TERMINATING BASIC LINE INPUT EOF COMMAND EVALUATE AN EXPRESSION
0408 0409 0410 0411 0412 015E 0413 0161 0414 0164 0415 0167 0416 016A 0417 016D 0418 0170 0419 0173 0420 0176 0421 0179 0422 017C 0423 017F 0424 0185 0185 0426 0188 0427 018B 0428 0488	* IS THE VA * RVECØ RVEC1 RVEC2 RVEC3 RVEC4 RVEC5 RVEC6 RVEC7 RVEC8 RVEC9 RVEC1Ø RVEC11 RVEC11 RVEC12 RVEC13 RVEC14 RVEC15 RVEC16	RMB 3	2.Ø \$A5F6 \$A5B9 \$A35F \$A282 \$A176 \$A3ED \$A4Ø6 \$A420 \$B918 \$B961 \$A549 \$A390 \$A48F \$A5CE \$B223 \$AC46	\$8273 \$8273 \$8CF1 \$8286 \$8E90	1.0 \$C426 \$C838 \$C843 \$C844 \$C58F \$C818 \$CA3B \$CA4B \$8E90 \$CC5B \$C656 \$C666 \$C667	1.1 \$C44B \$C888 \$C893 \$CC1C \$C5BC \$C84B \$CAE9 \$CAF9 \$CD35 \$CBA9 \$CG4 \$C6E4 \$C9ØC \$C9ØC \$C6E4	OPEN COMMAND DEVICE NUMBER VALIDITY CHECK SET PRINT PARAMETERS CONSOLE OUT CONSOLE IN INPUT DEVICE NUMBER CHECK PRINT DEVICE NUMBER CHECK CLOSE ALL FILES CLOSE ONE FILE PRINT INPUT BREAK CHECK INPUTTING A BASIC LINE TERMINATING BASIC LINE INPUT EOF COMMAND EVALUATE AN EXPRESSION RESERVED FOR ON ERROR GOTO COMMAND
0408 0409 0410 0411 0412 015E 0413 0161 0414 0164 0415 0167 0416 016A 0417 016D 0418 0170 0419 0173 0420 0176 0421 0179 0422 017C 0423 017F 0424 0185 0425 0185 0426 0188 0427 0188 0429 0191	* IS THE VA * RVECØ RVEC1 RVEC2 RVEC3 RVEC4 RVEC5 RVEC6 RVEC7 RVEC8 RVEC10 RVEC11 RVEC11 RVEC12 RVEC13 RVEC14 RVEC15 RVEC16 RVEC16 RVEC16	RMB 3	2.0 \$A5F6 \$A5F6 \$A5B9 \$A35F \$A282 \$A176 \$A3ED \$A406 \$A426 \$A426 \$A426 \$A426 \$A426 \$A549 \$A549 \$A549 \$A549 \$A549 \$A549 \$A549 \$A549	\$8273 \$8273 \$8CF1 \$8286 \$8E90 \$88846 \$8886	1.0 \$C426 \$C838 \$C843 \$C844 \$C58F \$C818 \$CA38 \$C658 \$C687 \$C687 \$C667 \$C667 \$C667 \$C240	1.1 \$C44B \$C888 \$C893 \$CC1C \$C5BC \$C84B \$CAE9 \$CAE9 \$CAE9 \$CD35 \$C8A9 \$C6E4 \$CAE4 \$C9E2 \$C6E4	OPEN COMMAND DEVICE NUMBER VALIDITY CHECK SET PRINT PARAMETERS CONSOLE OUT CONSOLE IN INPUT DEVICE NUMBER CHECK PRINT DEVICE NUMBER CHECK CLOSE ALL FILES CLOSE ONE FILE PRINT INPUT BREAK CHECK INPUTTING A BASIC LINE TERMINATING BASIC LINE INPUT EOF COMMAND EVALUATE AN EXPRESSION RESERVED FOR ON ERROR GOTO COMMAND ERROR DRIVER
0408 0409 0410 0411 0412 015E 0413 0161 0414 0164 0415 0167 0416 016A 0417 016D 0418 0170 0419 0173 0420 0176 0421 0179 0422 0170 0423 017F 0424 0185 0426 0185 0426 0188 0427 0188 0429 0191 0430 0194	* IS THE VA * RVECØ RVEC1 RVEC2 RVEC3 RVEC5 RVEC6 RVEC7 RVEC8 RVEC1Ø RVEC10 RVEC11 RVEC12 RVEC11 RVEC12 RVEC14 RVEC15 RVEC16 RVEC17 RVEC18	RMB 3	2.0 \$A5F6 \$A5F6 \$A5B9 \$A35F \$A282 \$A176 \$A3ED \$A4426 \$A4426 \$A4426 \$A4426 \$A4426 \$A4426 \$A4426 \$A4426 \$A426 \$A426 \$A5E0	\$8273 \$8273 \$8CF1 \$8286 \$8E90 \$88846 \$88870 \$829C	1.0 \$C426 \$C838 \$C843 \$C844 \$C58F \$C818 \$CA3B \$CA4B \$8E90 \$CC5B \$C656 \$C666 \$C667	1.1 \$C44B \$C888 \$C893 \$CC1C \$C5BC \$C84B \$CAE9 \$CAE9 \$CAE9 \$CD35 \$C8A9 \$C6E4 \$CAE4 \$C9E2 \$C6E4	OPEN COMMAND DEVICE NUMBER VALIDITY CHECK SET PRINT PARAMETERS CONSOLE OUT CONSOLE IN INPUT DEVICE NUMBER CHECK PRINT DEVICE NUMBER CHECK CLOSE ALL FILES CLOSE ONE FILE PRINT INPUT BREAK CHECK INPUTTING A BASIC LINE TERMINATING BASIC LINE INPUT EOF COMMAND EVALUATE AN EXPRESSION RESERVED FOR ON ERROR GOTO COMMAND ERROR DRIVER RUN
0408 0409 0410 0411 0412 0413 0414 0414 0415 0416 0416 0416 0417 0416 0417 0419 0419 0419 0419 0419 0419 0419 0419 0417 0420 0417 0421 0417 0422 0421 0421 0421 0421 0422 0421 0422 0425 0425 0426 0428 0428 0428 0429 0429 0430 0430 0430 0440	* IS THE VA * RVECØ RVEC1 RVEC2 RVEC3 RVEC5 RVEC6 RVEC7 RVEC8 RVEC10 RVEC11 RVEC12 RVEC12 RVEC13 RVEC14 RVEC15 RVEC15 RVEC16 RVEC17 RVEC18 RVEC18 RVEC19	RMB 3 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8	2.Ø \$A5F6 \$A5B9 \$A35F \$A282 \$A176 \$A486 \$A420 \$B918 \$B918 \$A549 \$A39Ø \$A4BF \$A5CE \$B223 \$AC49 \$AC49 \$AC49 \$BD22	\$8273 \$8273 \$8CF1 \$8286 \$8E9Ø \$88846 \$88FØ \$829C \$87EF	1.0 \$C426 \$C838 \$C843 \$C844 \$C58F \$C818 \$CA38 \$C658 \$C687 \$C687 \$C667 \$C667 \$C667 \$C240	1.1 \$C44B \$C888 \$C893 \$CC1C \$C5BC \$C84B \$CAE9 \$CAE9 \$CAE9 \$CAE4 \$CQE6 \$CCE0 \$CE02 \$CE02 \$CE02 \$CE02 \$CAE4	OPEN COMMAND DEVICE NUMBER VALIDITY CHECK SET PRINT PARAMETERS CONSOLE OUT CONSOLE IN INPUT DEVICE NUMBER CHECK PRINT DEVICE NUMBER CHECK CLOSE ALL FILES CLOSE ONE FILE PRINT INPUT BREAK CHECK INPUTTING A BASIC LINE TERMINATING BASIC LINE INPUT EOF COMMAND EVALUATE AN EXPRESSION RESERVED FOR ON ERROR GOTO COMMAND ERROR DRIVER RUN ASCII TO FLOATING POINT CONVERSION
0408 0409 0410 0411 0412 015E 0413 0161 0414 0164 0415 0167 0416 016A 0417 016D 0418 0170 0419 0173 0420 0176 0421 0179 0422 0170 0423 017F 0424 0185 0426 0185 0426 0188 0427 0188 0429 0191 0430 0194	* IS THE VA * RVECØ RVEC1 RVEC2 RVEC3 RVEC5 RVEC6 RVEC7 RVEC8 RVEC10 RVEC11 RVEC12 RVEC12 RVEC13 RVEC14 RVEC15 RVEC15 RVEC16 RVEC17 RVEC18 RVEC18 RVEC19	RMB 3	2.Ø \$A5F6 \$A5B9 \$A35F \$A282 \$A176 \$A486 \$A420 \$B918 \$B918 \$A549 \$A39Ø \$A4BF \$A5CE \$B223 \$AC49 \$AC49 \$AC49 \$BD22	\$8273 \$8273 \$8CF1 \$8286 \$8E90 \$88846 \$88870 \$829C	1.0 \$C426 \$C838 \$C843 \$C844 \$C58F \$C818 \$CA38 \$C658 \$C687 \$C687 \$C667 \$C667 \$C667 \$C240	1.1 \$C44B \$C888 \$C893 \$CC1C \$C5BC \$C84B \$CAE9 \$CAE9 \$CAE9 \$CD35 \$C8A9 \$C6E4 \$CAE4 \$C9E2 \$C6E4	OPEN COMMAND DEVICE NUMBER VALIDITY CHECK SET PRINT PARAMETERS CONSOLE OUT CONSOLE IN INPUT DEVICE NUMBER CHECK PRINT DEVICE NUMBER CHECK CLOSE ALL FILES CLOSE ONE FILE PRINT INPUT BREAK CHECK INPUTTING A BASIC LINE TERMINATING BASIC LINE INPUT EOF COMMAND EVALUATE AN EXPRESSION RESERVED FOR ON ERROR GOTO COMMAND ERROR DRIVER RUN
0408 0409 0410 0411 0412 0413 0414 0414 0415 0416 0416 0416 0417 0416 0417 0419 0419 0419 0419 0419 0419 0419 0419 0417 0420 0417 0421 0417 0422 0421 0421 0421 0421 0422 0421 0422 0425 0425 0426 0428 0428 0428 0429 0429 0430 0430 0430 0440	* IS THE VA * RVECØ RVEC1 RVEC2 RVEC3 RVEC4 RVEC5 RVEC6 RVEC7 RVEC8 RVEC1Ø RVEC11 RVEC12 RVEC11 RVEC12 RVEC13 RVEC14 RVEC15 RVEC16 RVEC17 RVEC18 RVEC19 RVEC19 RVEC19 RVEC19	RMB 3 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8	2.Ø \$A5F6 \$A5B9 \$A35F \$A282 \$A176 \$A486 \$A420 \$B918 \$B918 \$A549 \$A39Ø \$A4BF \$A5CE \$B223 \$AC49 \$AC49 \$AC49 \$BD22	\$8273 \$8273 \$8CF1 \$8286 \$8E9Ø \$88846 \$88FØ \$829C \$87EF	1.0 \$C426 \$C838 \$C843 \$C844 \$C58F \$C818 \$CA38 \$C658 \$C687 \$C687 \$C667 \$C667 \$C667 \$C240	1.1 \$C44B \$C888 \$C893 \$CC1C \$C5BC \$C84B \$CAE9 \$CAE9 \$CAE9 \$CAE4 \$CQE6 \$CCE0 \$CE02 \$CE02 \$CE02 \$CE02 \$CAE4	OPEN COMMAND DEVICE NUMBER VALIDITY CHECK SET PRINT PARAMETERS CONSOLE OUT CONSOLE IN INPUT DEVICE NUMBER CHECK PRINT DEVICE NUMBER CHECK CLOSE ALL FILES CLOSE ONE FILE PRINT INPUT BREAK CHECK INPUTTING A BASIC LINE TERMINATING BASIC LINE INPUT EOF COMMAND EVALUATE AN EXPRESSION RESERVED FOR ON ERROR GOTO COMMAND ERROR DRIVER RUN ASCII TO FLOATING POINT CONVERSION
0408 0409 0410 0411 0412 015E 0413 0161 0414 0164 0415 0167 0416 016A 0417 016D 0418 0170 0419 0173 0420 0176 0421 0179 0422 017C 0423 017F 0424 0182 0425 0185 0426 0188 0427 0188 0427 0188 0429 0191 0430 0194 0431 0197 0432 019A	* IS THE VA * RVECØ RVEC1 RVEC2 RVEC3 RVEC4 RVEC5 RVEC6 RVEC7 RVEC8 RVEC1Ø RVEC1Ø RVEC11 RVEC12 RVEC12 RVEC13 RVEC14 RVEC15 RVEC16 RVEC17 RVEC16 RVEC17 RVEC18 RVEC19 RVEC19 RVEC20	RMB 3 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8	2.0 \$A5F6 \$A5B9 \$A35F \$A282 \$A176 \$A3406 \$A420 \$B918 \$B061 \$A549 \$A349 \$A348 \$A5CE \$B223 \$AC46 \$AC49 \$AC49 \$AC49 \$AC7 \$BD22 \$AD9E	\$8273 \$8273 \$8CF1 \$8286 \$8E9Ø \$88846 \$88FØ \$829C \$87EF	1.0 \$C426 \$C838 \$C843 \$C844 \$C58F \$C818 \$CA38 \$C658 \$C687 \$C687 \$C667 \$C667 \$C667 \$C240	1.1 \$C44B \$C888 \$C893 \$CC1C \$C5BC \$C84B \$CAE9 \$CAE9 \$CAE9 \$CAE4 \$CQE6 \$CCE0 \$CE02 \$CE02 \$CE02 \$CE02 \$CAE4	OPEN COMMAND DEVICE NUMBER VALIDITY CHECK SET PRINT PARAMETERS CONSOLE OUT CONSOLE IN INPUT DEVICE NUMBER CHECK PRINT DEVICE NUMBER CHECK CLOSE ALL FILES CLOSE ONE FILE PRINT INPUT BREAK CHECK INPUTTING A BASIC LINE TERMINATING BASIC LINE INPUT EOF COMMAND EVALUATE AN EXPRESSION RESERVED FOR ON ERROR GOTO COMMAND ERROR DRIVER RUN ASCII TO FLOATING POINT CONVERSION BASIC'S COMMAND INTERPRETATION LOOP
0408 0409 0410 0411 0412 015E 0413 0161 0414 0164 0415 0167 0416 0417 0160 0418 0170 0419 0173 0420 0176 0421 0179 0422 0170 0423 0176 0421 0179 0422 0170 0423 0176 0421 0188 0427 0188 0424 0188 0427 0188 0428 0191 0430 0191 0430 0194 0431 0197 0432 0190 0433 0190 0434 0190 0433 0190 0434 0190 0433 0190 0434 0190	* IS THE VA * RVECØ RVEC1 RVEC2 RVEC3 RVEC4 RVEC5 RVEC6 RVEC7 RVEC8 RVEC1Ø RVEC1Ø RVEC11 RVEC12 RVEC12 RVEC13 RVEC14 RVEC15 RVEC16 RVEC17 RVEC16 RVEC17 RVEC18 RVEC19 RVEC19 RVEC20	RMB 3 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8	2.0 \$A5F6 \$A5B9 \$A35F \$A282 \$A176 \$A3ED \$A426 \$A42D \$B918 \$B061 \$A549 \$A5CE \$B223 \$AC46 \$AC49 \$AE75 \$BD22 \$AD9E \$A8C4 \$A910	\$8273 \$8273 \$8CF1 \$8286 \$8E9Ø \$88846 \$88FØ \$829C \$87EF	1.0 \$C426 \$C838 \$C843 \$C844 \$C58F \$C818 \$CA38 \$C658 \$C687 \$C687 \$C667 \$C667 \$C667 \$C240	1.1 \$C44B \$C888 \$C893 \$CC1C \$C5BC \$C84B \$CAE9 \$CAE9 \$CAE9 \$CAE4 \$CQE6 \$CCE0 \$CE02 \$CE02 \$CE02 \$CE02 \$CAE4	OPEN COMMAND DEVICE NUMBER VALIDITY CHECK SET PRINT PARAMETERS CONSOLE OUT CONSOLE IN INPUT DEVICE NUMBER CHECK PRINT DEVICE NUMBER CHECK CLOSE ALL FILES CLOSE ONE FILE PRINT INPUT BREAK CHECK INPUTTING A BASIC LINE TERMINATING BASIC LINE INPUT EOF COMMAND EVALUATE AN EXPRESSION RESERVED FOR ON ERROR GOTO COMMAND ERROR DRIVER RUN ASCII TO FLOATING POINT CONVERSION BASIC'S COMMAND INTERPRETATION LOOP RESET/SET/POINT COMMANDS CLS
0408 0409 0410 0411 0412 015E 0413 0161 0414 0164 0415 0167 0416 016A 0417 016D 0418 0170 0419 0173 0420 0176 0421 0179 0422 0170 0423 0176 0421 0179 0422 0170 0428 0176 0421 0188 0427 0188 0426 0188 0427 0188 0427 0188 0428 0429 0191 0430 0194 0431 0194 0431 0194 0432 0194 0433 0194 0434 0190 0434 0190 0434 0190 0434 0190 0434 0190 0434 0190 0434 0190 0434 0190 0434 0190 0434 0435	* IS THE VA * RVECØ RVEC1 RVEC2 RVEC3 RVEC4 RVEC5 RVEC6 RVEC7 RVEC8 RVEC1Ø RVEC11 RVEC11 RVEC112 RVEC113 RVEC14 RVEC15 RVEC16 RVEC16 RVEC16 RVEC17 RVEC18 RVEC17 RVEC18 RVEC18 RVEC19 RVEC20 RVEC20 RVEC22 *	RMB 3 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8	2.Ø \$A5F6 \$A5B9 \$A35F \$A282 \$A176 \$A3ED \$A4Ø6 \$A420 \$B918 \$BØ61 \$A549 \$A549 \$A5CE \$B223 \$AC46 \$AC49 \$AE75 \$BD22 \$AD82 \$A	\$8273 \$8273 \$8CF1 \$8286 \$8E9Ø \$88846 \$88FØ \$829C \$87EF	1.0 \$C426 \$C838 \$C843 \$C844 \$C58F \$C818 \$CA38 \$C658 \$C687 \$C687 \$C667 \$C667 \$C667 \$C240	1.1 \$C44B \$C888 \$C893 \$CC1C \$C5BC \$C84B \$CAE9 \$CAE9 \$CAE9 \$CAE4 \$CQE6 \$CCE0 \$CE02 \$CE02 \$CE02 \$CE02 \$CAE4	OPEN COMMAND DEVICE NUMBER VALIDITY CHECK SET PRINT PARAMETERS CONSOLE OUT CONSOLE IN INPUT DEVICE NUMBER CHECK PRINT DEVICE NUMBER CHECK CLOSE ALL FILES CLOSE ONE FILE PRINT INPUT BREAK CHECK INPUTTING A BASIC LINE TERMINATING BASIC LINE INPUT EOF COMMAND EVALUATE AN EXPRESSION RESERVED FOR ON ERROR GOTO COMMAND ERROR DRIVER RUN ASCII TO FLOATING POINT CONVERSION BASIC'S COMMAND INTERPRETATION LOOP RESET/SET/POINT COMMANDS CLS EXBAS' SECONDARY TOKEN HANDLER
0408 0409 0410 0411 0412 015E 0413 0161 0414 0164 0415 0167 0416 016A 0417 016D 0418 0170 0419 0173 0420 0176 0421 0179 0422 0170 0423 017F 0424 0185 0425 0188 0427 0188 0427 0188 0428 0191 0430 0191 0431 0194 0431 0197 0432 0190 0433 0190 0434 0190 0435 0436	* IS THE VA * RVECØ RVEC1 RVEC2 RVEC3 RVEC4 RVEC5 RVEC6 RVEC7 RVEC8 RVEC10 RVEC11 RVEC12 RVEC11 RVEC12 RVEC14 RVEC15 RVEC16 RVEC15 RVEC16 RVEC17 RVEC18 RVEC19 RVEC21 RVEC20 RVEC21 RVEC21 RVEC21 RVEC22 *	RMB 3 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8	2.0 \$A5F6 \$A5B9 \$A35F \$A282 \$A176 \$A3ED \$A406 \$A426 \$A426 \$A426 \$A426 \$A5E \$A5E \$A5E \$A5E \$A5E \$A5E \$A5E \$A5E	\$8273 \$8273 \$8CF1 \$8286 \$8E9Ø \$88846 \$88FØ \$829C \$87EF	1.0 \$C426 \$C838 \$C843 \$C844 \$C58F \$C818 \$CA38 \$C687 \$C687 \$C687 \$C6990	1.1 \$C44B \$C888 \$C893 \$CC1C \$C5BC \$C84B \$CAE9 \$CAE9 \$CAE9 \$CD35 \$C8A9 \$C6E4 \$C9EC \$C6E4 \$C9EC \$C6E4 \$C265 \$CASBØ	OPEN COMMAND DEVICE NUMBER VALIDITY CHECK SET PRINT PARAMETERS CONSOLE OUT CONSOLE IN INPUT DEVICE NUMBER CHECK PRINT DEVICE NUMBER CHECK CLOSE ALL FILES CLOSE ONE FILE PRINT INPUT BREAK CHECK INPUTTING A BASIC LINE TERMINATING BASIC LINE INPUT EOF COMMAND EVALUATE AN EXPRESSION RESERVED FOR ON ERROR GOTO COMMAND ERROR DRIVER RUN ASCII TO FLOATING POINT CONVERSION BASIC'S COMMAND INTERPRETATION LOOP RESET/SET/POINT COMMANDS CLS EXBAS' SECONDARY TOKEN HANDLER EXBAS' RENUM TOKEN CHECK
0408 0409 0410 0411 0412 0413 0161 0414 0415 0167 0416 0417 0416 0417 0419 0419 0419 0419 0419 0419 0417 0420 0176 0421 0176 0421 0177 0422 0177 0422 0177 0423 0177 0424 0185 0426 0188 0427 0188 0427 0188 0428 0429 0191 0430 0194 0431 0197 0432 0194 0433 0190 0433 0190 0434 01436 0436 0437	* IS THE VA * RVECØ RVEC1 RVEC2 RVEC3 RVEC4 RVEC5 RVEC6 RVEC7 RVEC8 RVEC10 RVEC11 RVEC12 RVEC11 RVEC12 RVEC13 RVEC14 RVEC15 RVEC14 RVEC15 RVEC16 RVEC17 RVEC18 RVEC19 RVEC20 RVEC20 RVEC21 RVEC22 * *	RMB 3 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8	2.Ø \$A5F6 \$A5B9 \$A35F \$A282 \$A176 \$A406 \$A420 \$B918 \$A5649 \$A39Ø \$A4BF \$A5CE \$B223 \$AC46 \$AC49 \$AE75 \$BD22 \$AB64 \$A649 \$AE75 \$AB64 \$AE75 \$	\$8273 \$8273 \$8CF1 \$8286 \$8E9Ø \$88846 \$88FØ \$829C \$87EF \$82B9	1.0 \$C426 \$C838 \$C843 \$C844 \$C58F \$C818 \$CA38 \$C687 \$C687 \$C687 \$C6990	1.1 \$C44B \$C888 \$C893 \$CC1C \$C5BC \$C84B \$CAE9 \$CAE9 \$CAE9 \$CAE4 \$CQE6 \$CCE0 \$CE02 \$CE02 \$CE02 \$CE02 \$CAE4	OPEN COMMAND DEVICE NUMBER VALIDITY CHECK SET PRINT PARAMETERS CONSOLE OUT CONSOLE IN INPUT DEVICE NUMBER CHECK PRINT DEVICE NUMBER CHECK CLOSE ALL FILES CLOSE ONE FILE PRINT INPUT BREAK CHECK INPUTTING A BASIC LINE TERMINATING BASIC LINE INPUT EOF COMMAND EVALUATE AN EXPRESSION RESERVED FOR ON ERROR GOTO COMMAND ERROR DRIVER RUN ASCII TO FLOATING POINT CONVERSION BASIC'S COMMAND INTERPRETATION LOOP RESET/SET/POINT COMMANDS CLS EXBAS' SECONDARY TOKEN HANDLER EXBAS' RENUM TOKEN CHECK EXBAS' GET/PUT
0408 0409 0410 0411 0412 015E 0413 0161 0414 0164 0415 0167 0416 016A 0417 016D 0418 0170 0419 0173 0420 0176 0421 0179 0422 017C 0423 017F 0424 0182 0425 0185 0426 0188 0427 0188 0427 0188 0428 0194 0431 0197 0432 019A 0433 019D 0434 01A0 0435 0437 0438 01A3	* IS THE VA * RVECØ RVEC1 RVEC2 RVEC3 RVEC4 RVEC5 RVEC6 RVEC7 RVEC8 RVEC1Ø RVEC11 RVEC12 RVEC11 RVEC12 RVEC13 RVEC14 RVEC15 RVEC16 RVEC16 RVEC17 RVEC18 RVEC19 RVEC20 RVEC20 * * * RVEC23	RMB 3 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8	2.0	\$8273 \$8273 \$8CF1 \$8286 \$8E9Ø \$88846 \$88FØ \$829C \$87EF \$82B9	1.0 \$C426 \$C838 \$C843 \$C844 \$C58F \$C818 \$CA38 \$C687 \$C687 \$C687 \$C6990	1.1 \$C44B \$C888 \$C893 \$CC1C \$C5BC \$C84B \$CAE9 \$CAE9 \$CAE9 \$CD35 \$C8A9 \$C6E4 \$C9EC \$C6E4 \$C9EC \$C6E4 \$C265 \$CASBØ	OPEN COMMAND DEVICE NUMBER VALIDITY CHECK SET PRINT PARAMETERS CONSOLE OUT CONSOLE IN INPUT DEVICE NUMBER CHECK PRINT DEVICE NUMBER CHECK CLOSE ALL FILES CLOSE ONE FILE PRINT INPUT BREAK CHECK INPUTTING A BASIC LINE TERMINATING BASIC LINE INPUT EOF COMMAND EVALUATE AN EXPRESSION RESERVED FOR ON ERROR GOTO COMMAND ERROR DRIVER RUN ASCII TO FLOATING POINT CONVERSION BASIC'S COMMAND INTERPRETATION LOOP RESET/SET/POINT COMMANDS CLS EXBAS' SECONDARY TOKEN HANDLER EXBAS' RENUM TOKEN CHECK EXBAS' GET/PUT CRUNCH BASIC LINE
0408 0409 0410 0411 0412 0413 0161 0414 0415 0167 0416 0417 0416 0417 0419 0419 0419 0419 0419 0419 0417 0420 0176 0421 0176 0421 0177 0422 0177 0422 0177 0423 0177 0424 0185 0426 0188 0427 0188 0427 0188 0428 0429 0191 0430 0194 0431 0197 0432 0194 0433 0190 0433 0190 0434 01436 0436 0437	* IS THE VA * RVECØ RVEC1 RVEC2 RVEC3 RVEC4 RVEC5 RVEC6 RVEC7 RVEC8 RVEC1Ø RVEC11 RVEC12 RVEC11 RVEC12 RVEC13 RVEC14 RVEC15 RVEC16 RVEC16 RVEC17 RVEC18 RVEC19 RVEC20 RVEC20 * * * RVEC23	RMB 3 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8	2.Ø \$A5F6 \$A5B9 \$A35F \$A282 \$A176 \$A406 \$A420 \$B918 \$A5649 \$A39Ø \$A4BF \$A5CE \$B223 \$AC46 \$AC49 \$AE75 \$BD22 \$AB64 \$A649 \$AE75 \$AB64 \$AE75 \$	\$8273 \$8273 \$8CF1 \$8286 \$8E9Ø \$88846 \$88FØ \$829C \$87EF \$82B9	1.0 \$C426 \$C838 \$C843 \$C844 \$C58F \$C818 \$CA38 \$C687 \$C687 \$C687 \$C6990	1.1 \$C44B \$C888 \$C893 \$CC1C \$C5BC \$C84B \$CAE9 \$CAE9 \$CAE9 \$CD35 \$C8A9 \$C6E4 \$C9EC \$C6E4 \$C9EC \$C6E4 \$C265 \$CASBØ	OPEN COMMAND DEVICE NUMBER VALIDITY CHECK SET PRINT PARAMETERS CONSOLE OUT CONSOLE IN INPUT DEVICE NUMBER CHECK PRINT DEVICE NUMBER CHECK CLOSE ALL FILES CLOSE ONE FILE PRINT INPUT BREAK CHECK INPUTTING A BASIC LINE TERMINATING BASIC LINE INPUT EOF COMMAND EVALUATE AN EXPRESSION RESERVED FOR ON ERROR GOTO COMMAND ERROR DRIVER RUN ASCII TO FLOATING POINT CONVERSION BASIC'S COMMAND INTERPRETATION LOOP RESET/SET/POINT COMMANDS CLS EXBAS' SECONDARY TOKEN HANDLER EXBAS' RENUM TOKEN CHECK EXBAS' GET/PUT
0408 0409 0410 0411 0412 015E 0413 0161 0414 0164 0415 0167 0416 016A 0417 016D 0418 0170 0419 0173 0420 0176 0421 0179 0422 017C 0423 017F 0424 0182 0425 0185 0426 0188 0427 0188 0427 0188 0428 0194 0431 0197 0432 019A 0433 019D 0434 01A0 0435 0437 0438 01A3	* IS THE VA * RVECØ RVEC1 RVEC2 RVEC3 RVEC4 RVEC5 RVEC6 RVEC7 RVEC8 RVEC1Ø RVEC11 RVEC12 RVEC11 RVEC12 RVEC13 RVEC14 RVEC15 RVEC16 RVEC16 RVEC17 RVEC18 RVEC19 RVEC20 RVEC20 * * * RVEC23	RMB 3 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8	2.0	\$8273 \$8273 \$8CF1 \$8286 \$8E9Ø \$88846 \$88FØ \$829C \$87EF \$82B9	1.0 \$C426 \$C838 \$C843 \$C844 \$C58F \$C818 \$CA38 \$C687 \$C687 \$C687 \$C6990	1.1 \$C44B \$C888 \$C893 \$CC1C \$C5BC \$C84B \$CAE9 \$CAE9 \$CAE9 \$CD35 \$C8A9 \$C6E4 \$C9EC \$C6E4 \$C9EC \$C6E4 \$C265 \$CASBØ	OPEN COMMAND DEVICE NUMBER VALIDITY CHECK SET PRINT PARAMETERS CONSOLE OUT CONSOLE IN INPUT DEVICE NUMBER CHECK PRINT DEVICE NUMBER CHECK CLOSE ALL FILES CLOSE ONE FILE PRINT INPUT BREAK CHECK INPUTTING A BASIC LINE TERMINATING BASIC LINE INPUT EOF COMMAND EVALUATE AN EXPRESSION RESERVED FOR ON ERROR GOTO COMMAND ERROR DRIVER RUN ASCII TO FLOATING POINT CONVERSION BASIC'S COMMAND INTERPRETATION LOOP RESET/SET/POINT COMMANDS CLS EXBAS' SECONDARY TOKEN HANDLER EXBAS' RENUM TOKEN CHECK EXBAS' GET/PUT CRUNCH BASIC LINE
0408 0409 0410 0411 0412 015E 0413 0161 0414 0164 0415 0167 0416 0417 0160 0418 0170 0419 0173 0420 0176 0421 0179 0422 0170 0423 0176 0421 0179 0424 0182 0425 0188 0427 0188 0426 0188 0427 0188 0428 0191 0430 0191 0430 0194 0431 0197 0432 0190 0433 0190 0434 0435 0436 0437 0438 0439 0440	* IS THE VA * RVECØ RVEC1 RVEC2 RVEC3 RVEC4 RVEC5 RVEC6 RVEC7 RVEC8 RVEC1Ø RVEC11 RVEC112 RVEC112 RVEC14 RVEC15 RVEC14 RVEC15 RVEC16 RVEC17 RVEC18 RVEC21 RVEC20 RVEC20 * * * RVEC23 RVEC24	RMB 3 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8	2.Ø \$A5F6 \$A5B9 \$A35F \$A282 \$A176 \$A3ED \$A426 \$B918 \$B961 \$A549 \$A549 \$A549 \$A5CE \$B223 \$AC46 \$AC49 \$AC49 \$AC49 \$AC49 \$AC49 \$AC49 \$AC49 \$AC49 \$AC49 \$AC40 \$A	\$8273 \$8273 \$8CF1 \$8286 \$8286 \$8290 \$8846 \$8290 \$87EF \$82B9	1.0	1.1 \$C44B \$C888 \$C893 \$CC1C \$C5BC \$C5BC \$C45B \$C84B \$CAE9 \$C035 \$C62B \$C62B \$C664 \$C45B \$C664 \$C265 \$CA3E	OPEN COMMAND DEVICE NUMBER VALIDITY CHECK SET PRINT PARAMETERS CONSOLE OUT CONSOLE IN INPUT DEVICE NUMBER CHECK PRINT DEVICE NUMBER CHECK CLOSE ALL FILES CLOSE ONE FILE PRINT INPUT BREAK CHECK INPUTTING A BASIC LINE TERMINATING BASIC LINE INPUT EOF COMMAND EVALUATE AN EXPRESSION RESERVED FOR ON ERROR GOTO COMMAND ERROR DRIVER RUN ASCII TO FLOATING POINT CONVERSION BASIC'S COMMAND INTERPRETATION LOOP RESET/SET/POINT COMMANDS CLS EXBAS' SECONDARY TOKEN HANDLER EXBAS' RENUM TOKEN CHECK EXBAS' GET/PUT CRUNCH BASIC LINE
0408 0409 0410 0411 0412 015E 0413 0161 0414 0164 0415 0167 0416 016A 0417 016D 0418 0170 0419 0173 0420 0176 0421 0179 0422 0170 0423 0176 0421 0179 0422 0170 0423 0176 0421 0188 0427 0188 0426 0188 0427 0188 0427 0188 0428 0429 0191 0430 0194 0431 0194 0431 0194 0433 0194 0433 0190 0434 0435 0436 0437 0438 0440 0440 0441 0440 0441 0441	* IS THE VA * RVECØ RVEC1 RVEC2 RVEC3 RVEC4 RVEC5 RVEC6 RVEC7 RVEC8 RVEC9 RVEC1Ø RVEC11 RVEC11 RVEC12 RVEC13 RVEC14 RVEC15 RVEC16 RVEC17 RVEC18 RVEC21 RVEC20 RVEC20 RVEC20 RVEC20 RVEC21 RVEC22 * * * * RVEC24 STRSTK	RMB 3 8*5	2.Ø \$A5F6 \$A5B9 \$A35F \$A282 \$A176 \$A3ED \$A4Ø6 \$A420 \$B918 \$BØ61 \$A549 \$A549 \$A5CE \$B223 \$AC46 \$AC49 \$AE75 \$BD22 \$ABC4 \$ABF4 \$A	\$8273 \$8273 \$8CF1 \$8286 \$8E90 \$8846 \$8E90 \$87EF \$82B9	1.0	1.1 \$C44B \$C888 \$C893 \$CC1C \$C5BC \$C84B \$CA49 \$CA59 \$CD35 \$C8A9 \$C624 \$CA64 \$C44 \$C465 \$C45 \$C45 \$C45 \$C45 \$C45 \$C45 \$C45 \$C4	OPEN COMMAND DEVICE NUMBER VALIDITY CHECK SET PRINT PARAMETERS CONSOLE OUT CONSOLE IN INPUT DEVICE NUMBER CHECK PRINT DEVICE NUMBER CHECK CLOSE ALL FILES CLOSE ONE FILE PRINT INPUT BREAK CHECK INPUTTING A BASIC LINE TERMINATING BASIC LINE INPUT EOF COMMAND EVALUATE AN EXPRESSION RESERVED FOR ON ERROR GOTO COMMAND ERROR DRIVER RUN ASCII TO FLOATING POINT CONVERSION BASIC'S COMMAND INTERPRETATION LOOP RESET/SET/POINT COMMANDS CLS EXBAS' SECONDARY TOKEN HANDLER EXBAS' RENUM TOKEN CHECK EXBAS' GET/PUT CRUNCH BASIC LINE
0408 0409 0411 0412 0413 0414 0415 0416 0416 0416 0417 0416 0417 0416 0417 0419 0420 0419 0421 0421 0421 0422 0425 0425 0426 0428 0427 0488 0427 0488 0429 0491 0430 0491 0431 0491 0433 0490 0433 0434 0435 0436 0437 0438 0440 0441 0441 0441 0441 0441 0441 0441 0441 0441 0441 0441	* IS THE VA * RVECØ RVEC1 RVEC2 RVEC3 RVEC6 RVEC7 RVEC8 RVEC10 RVEC11 RVEC12 RVEC11 RVEC12 RVEC13 RVEC14 RVEC15 RVEC16 RVEC17 RVEC16 RVEC17 RVEC18 RVEC17 RVEC20 RVEC20 RVEC21 RVEC22 * * * RVEC23 RVEC24 STRSTK CFNBUF	RMB 3	2.Ø \$A5F6 \$A5B9 \$A35F \$A282 \$A176 \$A406 \$A420 \$B918 \$B918 \$A549 \$A390 \$A4BF \$A5CE \$B223 \$AC49 \$A	### ### ### ### ### ### ### ### ### ##	1.Ø \$C426 \$C838 \$C843 \$C844 \$C818 \$C818 \$CA4B \$CA58 \$CA6Ø \$CC58 \$CA6Ø \$CC67 \$C29A \$C29A	1.1 \$C44B \$C888 \$C893 \$CC1C \$C5BC \$C84B \$CAE9 \$CAE9 \$CAE9 \$CAE4 \$CAE4 \$CAE4 \$CAE4 \$CED2 \$CED2 \$CED2 \$CED2 \$CEBØ	OPEN COMMAND DEVICE NUMBER VALIDITY CHECK SET PRINT PARAMETERS CONSOLE OUT CONSOLE IN INPUT DEVICE NUMBER CHECK PRINT DEVICE NUMBER CHECK CLOSE ALL FILES CLOSE ONE FILE PRINT INPUT BREAK CHECK INPUTTING A BASIC LINE TERMINATING BASIC LINE INPUT EOF COMMAND EVALUATE AN EXPRESSION RESERVED FOR ON ERROR GOTO COMMAND ERROR DRIVER RUN ASCII TO FLOATING POINT CONVERSION BASIC'S COMMAND INTERPRETATION LOOP RESET/SET/POINT COMMANDS CLS EXBAS' SECONDARY TOKEN HANDLER EXBAS' RENUM TOKEN CHECK EXBAS' GET/PUT CRUNCH BASIC LINE
0408 0409 0411 0411 0412 015E 0413 0161 0414 0164 0415 0167 0416 016A 0417 016D 0418 0170 0419 0173 0420 0176 0421 0179 0422 017C 0423 017F 0424 0182 0425 0185 0426 0188 0427 0188 0427 0188 0427 0198 0428 0194 0431 0197 0432 019A 0433 019D 0434 01A0 0435 0436 0437 0438 01A3 0439 01A6 0441 01A9 0441 01A9 0441 01A9	* IS THE VA * RVECØ RVEC1 RVEC2 RVEC3 RVEC6 RVEC5 RVEC6 RVEC7 RVEC8 RVEC10 RVEC11 RVEC12 RVEC11 RVEC12 RVEC13 RVEC14 RVEC15 RVEC14 RVEC15 RVEC16 RVEC17 RVEC18 RVEC19 RVEC20 RVEC20 RVEC20 RVEC21 RVEC22 * * * RVEC23 RVEC24 STRSTK CFNBUF CASBUF	RMB 3	2.Ø \$A5F6 \$A5B9 \$A35F \$A282 \$A176 \$A406 \$A420 \$B918 \$B961 \$A549 \$A	\$8273 \$8273 \$8CF1 \$8286 \$8E9Ø \$88846 \$88FØ \$829C \$87EF \$82B9	1.Ø \$C426 \$C838 \$C843 \$C844 \$C858 \$C818 \$CA4B \$CA58 \$CA58 \$CA58 \$CA59 \$C	1.1 \$C44B \$C888 \$C893 \$CC1C \$C5BC \$C84B \$CAE9 \$CAE9 \$CAE9 \$C6E4 \$C6E4 \$C6E4 \$C2E5 \$CA3E \$C8BØ \$C2B2	OPEN COMMAND DEVICE NUMBER VALIDITY CHECK SET PRINT PARAMETERS CONSOLE OUT CONSOLE IN INPUT DEVICE NUMBER CHECK PRINT DEVICE NUMBER CHECK CLOSE ALL FILES CLOSE ONE FILE PRINT INPUT BREAK CHECK INPUTTING A BASIC LINE TERMINATING BASIC LINE INPUT EOF COMMAND EVALUATE AN EXPRESSION RESERVED FOR ON ERROR GOTO COMMAND ERROR DRIVER RUN ASCII TO FLOATING POINT CONVERSION BASIC'S COMMAND INTERPRETATION LOOP RESET/SET/POINT COMMANDS CLS EXBAS' SECONDARY TOKEN HANDLER EXBAS' RENUM TOKEN CHECK EXBAS' GET/PUT CRUNCH BASIC LINE
0408 0409 0411 0412 0413 0414 0415 0416 0416 0416 0417 0416 0417 0416 0417 0419 0420 0419 0421 0421 0421 0422 0425 0425 0426 0428 0427 0488 0427 0488 0429 0491 0430 0491 0431 0491 0433 0490 0433 0434 0435 0436 0437 0438 0440 0441 0441 0441 0441 0441 0441 0441 0441 0441 0441 0441	* IS THE VA * RVECØ RVEC1 RVEC2 RVEC3 RVEC6 RVEC5 RVEC6 RVEC7 RVEC8 RVEC10 RVEC11 RVEC12 RVEC11 RVEC12 RVEC13 RVEC14 RVEC15 RVEC14 RVEC15 RVEC16 RVEC17 RVEC18 RVEC19 RVEC20 RVEC20 RVEC20 RVEC21 RVEC22 * * * RVEC23 RVEC24 STRSTK CFNBUF CASBUF	RMB 3	2.Ø \$A5F6 \$A5B9 \$A35F \$A282 \$A176 \$A406 \$A420 \$B918 \$B961 \$A549 \$A	\$8273 \$8273 \$8CF1 \$8286 \$8E9Ø \$88846 \$88FØ \$829C \$87EF \$82B9	1.Ø \$C426 \$C838 \$C843 \$C844 \$C818 \$C818 \$CA4B \$CA58 \$CA6Ø \$CC58 \$CA6Ø \$CC67 \$C29A \$C29A	1.1 \$C44B \$C888 \$C893 \$CC1C \$C5BC \$C84B \$CAE9 \$CAE9 \$CAE9 \$C6E4 \$C6E4 \$C6E4 \$C2E5 \$CA3E \$C8BØ \$C2B2	OPEN COMMAND DEVICE NUMBER VALIDITY CHECK SET PRINT PARAMETERS CONSOLE OUT CONSOLE IN INPUT DEVICE NUMBER CHECK PRINT DEVICE NUMBER CHECK CLOSE ALL FILES CLOSE ONE FILE PRINT INPUT BREAK CHECK INPUTTING A BASIC LINE TERMINATING BASIC LINE INPUT EOF COMMAND EVALUATE AN EXPRESSION RESERVED FOR ON ERROR GOTO COMMAND ERROR DRIVER RUN ASCII TO FLOATING POINT CONVERSION BASIC'S COMMAND INTERPRETATION LOOP RESET/SET/POINT COMMANDS CLS EXBAS' SECONDARY TOKEN HANDLER EXBAS' RENUM TOKEN CHECK EXBAS' GET/PUT CRUNCH BASIC LINE

Ø445	Ø2DC	LINBUF	RMB	LBUFMX+1	BASIC LINE INPUT BUFFER
Ø446		STRBUF	RMB	41	STRING BUFFER
	พรษา	31 KD01	KIID	41	SIKING DUTTER
Ø447					
Ø448	0400	VIDRAM	RMB	200	VIDEO DISPLAY AREA
Ø449					
Ø45Ø		*START OF	ADDIT	TONAL RAM VARTARI	LE STORAGE (DISK BASIC ONLY)
Ø451	αεαα	DBUFØ	RMB	SECLEN	I/O BUFFER #Ø
Ø452		DBUF1	RMB	SECLEN	I/O BUFFER #1
Ø453	0800	FATBLØ	RMB	FATLEN	FILE ALLOCATION TABLE - DRIVE Ø
Ø454	Ø84A	FATBL1	RMB	FATLEN	FILE ALLOCATION TABLE - DRIVE 1
Ø455	Ø894	FATBL2	RMB	FATLEN	FILE ALLOCATION TABLE - DRIVE 2
Ø456		FATBL3	RMB	FATLEN	FILE ALLOCATION TABLE - DRIVE 3
Ø457		FCBV1	RMB	16*2	FILE BUFFER VECTORS (15 USER, 1 SYSTEM)
Ø458		RNBFAD	RMB	2	START OF FREE RANDOM FILE BUFFER AREA
Ø459	Ø94A	FCBADR	RMB	2	START OF FILE CONTROL BLOCKS
Ø46Ø	Ø94C	DNAMBF	RMB	8	DISK FILE NAME BUFFER
Ø461		DEXTBF	RMB	3	DISK FILE EXTENSION NAME BUFFER
Ø462		DFLTYP	RMB	1	*DV* DISK FILE TYPE: Ø=BASIC, 1=DATA, 2=MACHINE
	W331	*	KIID	1	
Ø463				_	LANGUAGE, 3=TEXT EDITOR SOURCE FILE
Ø464		DASCFL	RMB	1	*DV* ASCII FLAG: Ø=CRUNCHED OR BINARY, \$FF=ASCII
Ø465	Ø959	DRUNFL	RMB	1	RUN FLAG: (IF BIT 1=1 THEN RUN, IF BIT Ø=1, THEN CLOSE
Ø466		*			ALL FILES BEFORE RUNNING)
Ø467	Ø95A	DEFDRV	RMB	1	DEFAULT DRIVE NUMBER
Ø468		FCBACT	RMB	1	NUMBER OF FCBS ACTIVE
Ø469		DRESFL		1	RESET FLAG: ◇Ø WILL CAUSE A 'NEW' & SHUT DOWN ALL FCBS
0470		DLOADFL	RMB	1	LOAD FLAG: CAUSE A 'NEW' FOLLOWING A LOAD ERROR
Ø471	Ø95E	DMRGFL	RMB	1	MERGE FLAG: Ø=NØ MERGE, \$FF=MERGE
Ø472	Ø95F	DUSRVC	RMB	20	DISK BASIC USR COMMAND VECTORS
Ø473		*** DISK F	ILE W	ORK AREA FOR DIRE	ECTORY SEARCH
Ø474		* EXISTI			
	a0.72				CECTOD NUMBER
Ø475		V973	RMB	1	SECTOR NUMBER
Ø476		V974	RMB	2	RAM DIRECTORY IMAGE ADDRESS
Ø477	Ø976	V976	RMB	1	FIRST GRANULE NUMBER
Ø478		* UNUSED	FILE		
Ø479	Ø977	V977	RMB	1	SECTOR NUMBER
Ø48Ø		V978	RMB	2	RAM DIRECTORY IMAGE ADDRESS
	<i>0370</i>	V 370	KIID	<u></u>	KAN DIRECTORT THAGE ADDRESS
Ø481					
Ø482	И97A	WFATVL	RMB	2	WRITE FAT VALUE: NUMBER OF FREE GRANULES WHICH MUST BE TAKEN
	23771		KIID	-	
Ø483	237.1.		KITD	_	FROM THE FAT TO TRIGGER A WRITE FAT TO DISK SEQUENCE
Ø483 Ø484		DFFLEN	RMB	2	
Ø484	Ø97C	DFFLEN	RMB	2	FROM THE FAT TO TRIGGER A WRITE FAT TO DISK SEQUENCE DIRECT ACCESS FILE RECORD LENGTH
Ø484 Ø485	Ø97C Ø97E	DFFLEN DRØTRK	RMB RMB	2 4	FROM THE FAT TO TRIGGER A WRITE FAT TO DISK SEQUENCE DIRECT ACCESS FILE RECORD LENGTH CURRENT TRACK NUMBER, DRIVES Ø,1,2,3
Ø484 Ø485 Ø486	Ø97C Ø97E Ø982	DFFLEN DRØTRK NMIFLG	RMB RMB RMB	2 4 1	FROM THE FAT TO TRIGGER A WRITE FAT TO DISK SEQUENCE DIRECT ACCESS FILE RECORD LENGTH CURRENT TRACK NUMBER, DRIVES Ø,1,2,3 NMI FLAG: Ø=DON'T VECTOR <>Ø=YECTOR OUT
Ø484 Ø485 Ø486 Ø487	Ø97C Ø97E Ø982	DFFLEN DRØTRK NMIFLG DNMIVC	RMB RMB	2 4	FROM THE FAT TO TRIGGER A WRITE FAT TO DISK SEQUENCE DIRECT ACCESS FILE RECORD LENGTH CURRENT TRACK NUMBER, DRIVES Ø,1,2,3 NMI FLAG: Ø=DON'T VECTOR <>Ø=YECTOR OUT NMI VECTOR: WHERE TO JUMP FOLLOWING AN NMI
Ø484 Ø485 Ø486 Ø487 Ø488	Ø97C Ø97E Ø982 Ø983	DFFLEN DRØTRK NMIFLG DNMIVC *	RMB RMB RMB RMB	2 4 1 2	FROM THE FAT TO TRIGGER A WRITE FAT TO DISK SEQUENCE DIRECT ACCESS FILE RECORD LENGTH CURRENT TRACK NUMBER, DRIVES Ø,1,2,3 NMI FLAG: Ø=DON'T VECTOR <>Ø=YECTOR OUT NMI VECTOR: WHERE TO JUMP FOLLOWING AN NMI INTERRUPT IF THE NMI FLAG IS SET
Ø484 Ø485 Ø486 Ø487	Ø97C Ø97E Ø982 Ø983	DFFLEN DRØTRK NMIFLG DNMIVC	RMB RMB RMB	2 4 1	FROM THE FAT TO TRIGGER A WRITE FAT TO DISK SEQUENCE DIRECT ACCESS FILE RECORD LENGTH CURRENT TRACK NUMBER, DRIVES Ø,1,2,3 NMI FLAG: Ø=DON'T VECTOR <>Ø=YECTOR OUT NMI VECTOR: WHERE TO JUMP FOLLOWING AN NMI
Ø484 Ø485 Ø486 Ø487 Ø488	097C 097E 0982 0983	DFFLEN DRØTRK NMIFLG DNMIVC *	RMB RMB RMB RMB	2 4 1 2	FROM THE FAT TO TRIGGER A WRITE FAT TO DISK SEQUENCE DIRECT ACCESS FILE RECORD LENGTH CURRENT TRACK NUMBER, DRIVES Ø,1,2,3 NMI FLAG: Ø=DON'T VECTOR <>Ø=YECTOR OUT NMI VECTOR: WHERE TO JUMP FOLLOWING AN NMI INTERRUPT IF THE NMI FLAG IS SET
Ø484 Ø485 Ø486 Ø487 Ø488 Ø489	097C 097E 0982 0983 0985	DFFLEN DRØTRK NMIFLG DNMIVC * RDYTMR DRGRAM	RMB RMB RMB RMB	2 4 1 2 1 1	FROM THE FAT TO TRIGGER A WRITE FAT TO DISK SEQUENCE DIRECT ACCESS FILE RECORD LENGTH CURRENT TRACK NUMBER, DRIVES Ø,1,2,3 NMI FLAG: Ø=DON'T VECTOR <>Ø=YECTOR OUT NMI VECTOR: WHERE TO JUMP FOLLOWING AN NMI INTERRUPT IF THE NMI FLAG IS SET MOTOR TURN OFF TIMER RAM IMAGE OF DSKREG (\$FF4Ø)
0484 0485 0486 0487 0488 0489 0490 0491	097C 097E 0982 0983 0985 0986 0987	DFFLEN DRØTRK NMIFLG DNMIVC * RDYTMR DRGRAM DVERFL	RMB RMB RMB RMB	2 4 1 2 1 1 1	FROM THE FAT TO TRIGGER A WRITE FAT TO DISK SEQUENCE DIRECT ACCESS FILE RECORD LENGTH CURRENT TRACK NUMBER, DRIVES Ø,1,2,3 NMI FLAG: Ø=DON'T VECTOR <>>Ø=YECTOR OUT NMI VECTOR: WHERE TO JUMP FOLLOWING AN NMI INTERRUPT IF THE NMI FLAG IS SET MOTOR TURN OFF TIMER RAM IMAGE OF DSKREG (\$FF4Ø) VERIFY FLAG: Ø=OFF, \$FF=ON
0484 0485 0486 0487 0488 0489 0490 0491 0492	097C 097E 0982 0983 0985 0986 0987	DFFLEN DRØTRK NMIFLG DNMIVC * RDYTMR DRGRAM DVERFL ATTCTR	RMB RMB RMB RMB	2 4 1 2 1 1	FROM THE FAT TO TRIGGER A WRITE FAT TO DISK SEQUENCE DIRECT ACCESS FILE RECORD LENGTH CURRENT TRACK NUMBER, DRIVES Ø,1,2,3 NMI FLAG: Ø=DON'T VECTOR <> Ø=VECTOR OUT NMI VECTOR: WHERE TO JUMP FOLLOWING AN NMI INTERRUPT IF THE NMI FLAG IS SET MOTOR TURN OFF TIMER RAM IMAGE OF DSKREG (\$FF4Ø) VERIFY FLAG: Ø=OFF, \$FF=ON READ/WRITE ATTEMPT COUNTER: NUMBER OF TIMES THE
0484 0485 0486 0487 0488 0489 0490 0491 0492 0493	097C 097E 0982 0983 0985 0986 0987	DFFLEN DRØTRK NMIFLG DNMIVC * RDYTMR DRGRAM DVERFL ATTCTR *	RMB RMB RMB RMB	2 4 1 2 1 1 1	FROM THE FAT TO TRIGGER A WRITE FAT TO DISK SEQUENCE DIRECT ACCESS FILE RECORD LENGTH CURRENT TRACK NUMBER, DRIVES Ø,1,2,3 NMI FLAG: Ø=DON'T VECTOR <>Ø=YECTOR OUT NMI VECTOR: WHERE TO JUMP FOLLOWING AN NMI INTERRUPT IF THE NMI FLAG IS SET MOTOR TURN OFF TIMER RAM IMAGE OF DSKREG (\$FF4Ø) VERIFY FLAG: Ø=OFF, \$FF=ON READ/WRITE ATTEMPT COUNTER: NUMBER OF TIMES THE DISK WILL ATTEMPT TO RETRIEVE OR WRITE DATA
0484 0485 0486 0487 0488 0489 0490 0491 0492 0493	097C 097E 0982 0983 0985 0986 0987	DFFLEN DRØTRK NMIFLG DNMIVC * RDYTMR DRGRAM DVERFL ATTCTR	RMB RMB RMB RMB	2 4 1 2 1 1 1	FROM THE FAT TO TRIGGER A WRITE FAT TO DISK SEQUENCE DIRECT ACCESS FILE RECORD LENGTH CURRENT TRACK NUMBER, DRIVES Ø,1,2,3 NMI FLAG: Ø=DON'T VECTOR <> Ø=VECTOR OUT NMI VECTOR: WHERE TO JUMP FOLLOWING AN NMI INTERRUPT IF THE NMI FLAG IS SET MOTOR TURN OFF TIMER RAM IMAGE OF DSKREG (\$FF4Ø) VERIFY FLAG: Ø=OFF, \$FF=ON READ/WRITE ATTEMPT COUNTER: NUMBER OF TIMES THE
0484 0485 0486 0487 0488 0489 0490 0491 0492 0493 0494	097C 097E 0982 0983 0985 0986 0987 0988	DFFLEN DRØTRK NMIFLG DNMIVC * RDYTMR DRGRAM DVERFL ATTCTR * *	RMB RMB RMB RMB RMB RMB RMB	2 4 1 2 1 1 1 1	FROM THE FAT TO TRIGGER A WRITE FAT TO DISK SEQUENCE DIRECT ACCESS FILE RECORD LENGTH CURRENT TRACK NUMBER, DRIVES Ø,1,2,3 NMI FLAG: Ø=DON'T VECTOR <>Ø=YECTOR OUT NMI VECTOR: WHERE TO JUMP FOLLOWING AN NMI INTERRUPT IF THE NMI FLAG IS SET MOTOR TURN OFF TIMER RAM IMAGE OF DSKREG (\$FF4Ø) VERIFY FLAG: Ø=OFF, \$FF=ON READ/WRITE ATTEMPT COUNTER: NUMBER OF TIMES THE DISK WILL ATTEMPT TO RETRIEVE OR WRITE DATA BEFORE IT GIVES UP AND ISSUES AN ERROR.
0484 0485 0486 0487 0488 0499 0491 0492 0493 0494 0495 0496	097C 097E 0982 0983 0985 0986 0987 0988	DFFLEN DRØTRK NMIFLG DNMIVC * RDYTMR DRGRAM DVERFL ATTCTR *	RMB RMB RMB RMB	2 4 1 2 1 1 1	FROM THE FAT TO TRIGGER A WRITE FAT TO DISK SEQUENCE DIRECT ACCESS FILE RECORD LENGTH CURRENT TRACK NUMBER, DRIVES Ø,1,2,3 NMI FLAG: Ø=DON'T VECTOR <>Ø=YECTOR OUT NMI VECTOR: WHERE TO JUMP FOLLOWING AN NMI INTERRUPT IF THE NMI FLAG IS SET MOTOR TURN OFF TIMER RAM IMAGE OF DSKREG (\$FF4Ø) VERIFY FLAG: Ø=OFF, \$FF=ON READ/WRITE ATTEMPT COUNTER: NUMBER OF TIMES THE DISK WILL ATTEMPT TO RETRIEVE OR WRITE DATA
0484 0485 0486 0487 0488 0489 0490 0491 0492 0493 0494	097C 097E 0982 0983 0985 0986 0987 0988	DFFLEN DRØTRK NMIFLG DNMIVC * RDYTMR DRGRAM DVERFL ATTCTR * *	RMB RMB RMB RMB RMB RMB RMB	2 4 1 2 1 1 1 1	FROM THE FAT TO TRIGGER A WRITE FAT TO DISK SEQUENCE DIRECT ACCESS FILE RECORD LENGTH CURRENT TRACK NUMBER, DRIVES Ø,1,2,3 NMI FLAG: Ø=DON'T VECTOR <>Ø=YECTOR OUT NMI VECTOR: WHERE TO JUMP FOLLOWING AN NMI INTERRUPT IF THE NMI FLAG IS SET MOTOR TURN OFF TIMER RAM IMAGE OF DSKREG (\$FF4Ø) VERIFY FLAG: Ø=OFF, \$FF=ON READ/WRITE ATTEMPT COUNTER: NUMBER OF TIMES THE DISK WILL ATTEMPT TO RETRIEVE OR WRITE DATA BEFORE IT GIVES UP AND ISSUES AN ERROR.
0484 0485 0486 0487 0488 0499 0491 0492 0493 0494 0495 0496	097C 097E 0982 0983 0985 0986 0987 0988	DFFLEN DRØTRK NMIFLG DNMIVC * RDYTMR DRGRAM DVERFL ATTCTR * * DFLBUF	RMB RMB RMB RMB RMB RMB RMB	2 4 1 2 1 1 1 1	FROM THE FAT TO TRIGGER A WRITE FAT TO DISK SEQUENCE DIRECT ACCESS FILE RECORD LENGTH CURRENT TRACK NUMBER, DRIVES Ø,1,2,3 NMI FLAG: Ø=DON'T VECTOR <>Ø=YECTOR OUT NMI VECTOR: WHERE TO JUMP FOLLOWING AN NMI INTERRUPT IF THE NMI FLAG IS SET MOTOR TURN OFF TIMER RAM IMAGE OF DSKREG (\$FF4Ø) VERIFY FLAG: Ø=OFF, \$FF=ON READ/WRITE ATTEMPT COUNTER: NUMBER OF TIMES THE DISK WILL ATTEMPT TO RETRIEVE OR WRITE DATA BEFORE IT GIVES UP AND ISSUES AN ERROR.
0484 0485 0486 0487 0488 0499 0491 0492 0493 0494 0495 0497 0498	097C 097E 0982 0983 0985 0986 0987 0988	DFFLEN DRØTRK NMIFLG DNMIVC * RDYTMR DRGRAM DVERFL ATTCTR * * DFLBUF	RMB RMB RMB RMB RMB RMB RMB	2 4 1 2 1 1 1 1 1 1	FROM THE FAT TO TRIGGER A WRITE FAT TO DISK SEQUENCE DIRECT ACCESS FILE RECORD LENGTH CURRENT TRACK NUMBER, DRIVES Ø,1,2,3 NMI FLAG: Ø=DON'T VECTOR <>Ø=YECTOR OUT NMI VECTOR: WHERE TO JUMP FOLLOWING AN NMI INTERRUPT IF THE NMI FLAG IS SET MOTOR TURN OFF TIMER RAM IMAGE OF DSKREG (\$FF4Ø) VERIFY FLAG: Ø=OFF, \$FF=ON READ/WRITE ATTEMPT COUNTER: NUMBER OF TIMES THE DISK WILL ATTEMPT TO RETRIEVE OR WRITE DATA BEFORE IT GIVES UP AND ISSUES AN ERROR.
0484 0485 0486 0487 0488 0489 0491 0492 0493 0494 0495 0497 0498	097C 097E 0982 0983 0985 0986 0987 0988	DFFLEN DRØTRK NMIFLG DNMIVC * RDYTMR DRGRAM DVERFL ATTCTR * * DFLBUF *RANDOM FI	RMB RMB RMB RMB RMB RMB RMB RMB	2 4 1 2 1 1 1 1 SECLEN SERVED AREA	FROM THE FAT TO TRIGGER A WRITE FAT TO DISK SEQUENCE DIRECT ACCESS FILE RECORD LENGTH CURRENT TRACK NUMBER, DRIVES Ø,1,2,3 NMI FLAG: Ø=DON'T VECTOR <> Ø=YECTOR OUT NMI VECTOR: WHERE TO JUMP FOLLOWING AN NMI INTERRUPT IF THE NMI FLAG IS SET MOTOR TURN OFF TIMER RAM IMAGE OF DSKREG (\$FF4Ø) VERIFY FLAG: Ø=OFF, \$FF=ON READ/WRITE ATTEMPT COUNTER: NUMBER OF TIMES THE DISK WILL ATTEMPT TO RETRIEVE OR WRITE DATA BEFORE IT GIVES UP AND ISSUES AN ERROR. INITIALIZED TO SECLEN BY DISKBAS
0484 0485 0486 0487 0488 0489 0491 0492 0493 0494 0495 0496 0497 0498 0499	097C 097E 0982 0983 0985 0986 0987 0988	DFFLEN DRØTRK NMIFLG DNMIVC * RDYTMR DRGRAM DVERFL ATTCTR * * DFLBUF *RANDOM FI	RMB RMB RMB RMB RMB RMB RMB RMB	2 4 1 2 1 1 1 1 1 1	FROM THE FAT TO TRIGGER A WRITE FAT TO DISK SEQUENCE DIRECT ACCESS FILE RECORD LENGTH CURRENT TRACK NUMBER, DRIVES Ø,1,2,3 NMI FLAG: Ø=DON'T VECTOR <> Ø=YECTOR OUT NMI VECTOR: WHERE TO JUMP FOLLOWING AN NMI INTERRUPT IF THE NMI FLAG IS SET MOTOR TURN OFF TIMER RAM IMAGE OF DSKREG (\$FF4Ø) VERIFY FLAG: Ø=OFF, \$FF=ON READ/WRITE ATTEMPT COUNTER: NUMBER OF TIMES THE DISK WILL ATTEMPT TO RETRIEVE OR WRITE DATA BEFORE IT GIVES UP AND ISSUES AN ERROR. INITIALIZED TO SECLEN BY DISKBAS
0484 0485 0486 0487 0488 0489 0491 0492 0493 0494 0495 0496 0497 0499 0500 0501	097C 097E 0982 0983 0985 0986 0987 0988	DFFLEN DRØTRK NMIFLG DMNIVC * RDYTMR DRGRAM DVERFL ATTCTR * * DFLBUF *RANDOM FI *FILE CONT	RMB RMB RMB RMB RMB RMB RMB	2 4 1 2 1 1 1 1 SECLEN SERVED AREA LOCKS AND BUFFERS	FROM THE FAT TO TRIGGER A WRITE FAT TO DISK SEQUENCE DIRECT ACCESS FILE RECORD LENGTH CURRENT TRACK NUMBER, DRIVES Ø,1,2,3 NMI FLAG: Ø=DON'T VECTOR <> Ø=YECTOR OUT NMI VECTOR: WHERE TO JUMP FOLLOWING AN NMI INTERRUPT IF THE NMI FLAG IS SET MOTOR TURN OFF TIMER RAM IMAGE OF DSKREG (\$FF4Ø) VERIFY FLAG: Ø=OFF, \$FF=ON READ/WRITE ATTEMPT COUNTER: NUMBER OF TIMES THE DISK WILL ATTEMPT TO RETRIEVE OR WRITE DATA BEFORE IT GIVES UP AND ISSUES AN ERROR. INITIALIZED TO SECLEN BY DISKBAS
0484 0485 0486 0487 0489 0491 0492 0493 0494 0495 0496 0497 0498 0499 0501	097C 097E 0982 0983 0985 0986 0987 0988	DFFLEN DRØTRK NMIFLG DMNIVC * RDYTMR DRGRAM DVERFL ATTCTR * * DFLBUF *RANDOM FI *FILE CONT	RMB RMB RMB RMB RMB RMB RMB	2 4 1 2 1 1 1 1 SECLEN SERVED AREA	FROM THE FAT TO TRIGGER A WRITE FAT TO DISK SEQUENCE DIRECT ACCESS FILE RECORD LENGTH CURRENT TRACK NUMBER, DRIVES Ø,1,2,3 NMI FLAG: Ø=DON'T VECTOR <> Ø=YECTOR OUT NMI VECTOR: WHERE TO JUMP FOLLOWING AN NMI INTERRUPT IF THE NMI FLAG IS SET MOTOR TURN OFF TIMER RAM IMAGE OF DSKREG (\$FF4Ø) VERIFY FLAG: Ø=OFF, \$FF=ON READ/WRITE ATTEMPT COUNTER: NUMBER OF TIMES THE DISK WILL ATTEMPT TO RETRIEVE OR WRITE DATA BEFORE IT GIVES UP AND ISSUES AN ERROR. INITIALIZED TO SECLEN BY DISKBAS
0484 0485 0486 0487 0488 0489 0491 0492 0493 0494 0495 0496 0497 0499 0500 0501	097C 097E 0982 0983 0985 0986 0987 0988	DFFLEN DRØTRK NMIFLG DMNIVC * RDYTMR DRGRAM DVERFL ATTCTR * * DFLBUF *RANDOM FI *FILE CONT	RMB RMB RMB RMB RMB RMB RMB	2 4 1 2 1 1 1 1 SECLEN SERVED AREA LOCKS AND BUFFERS	FROM THE FAT TO TRIGGER A WRITE FAT TO DISK SEQUENCE DIRECT ACCESS FILE RECORD LENGTH CURRENT TRACK NUMBER, DRIVES Ø,1,2,3 NMI FLAG: Ø=DON'T VECTOR <> Ø=YECTOR OUT NMI VECTOR: WHERE TO JUMP FOLLOWING AN NMI INTERRUPT IF THE NMI FLAG IS SET MOTOR TURN OFF TIMER RAM IMAGE OF DSKREG (\$FF4Ø) VERIFY FLAG: Ø=OFF, \$FF=ON READ/WRITE ATTEMPT COUNTER: NUMBER OF TIMES THE DISK WILL ATTEMPT TO RETRIEVE OR WRITE DATA BEFORE IT GIVES UP AND ISSUES AN ERROR. INITIALIZED TO SECLEN BY DISKBAS
0484 0485 0486 0487 0489 0491 0492 0493 0494 0495 0496 0497 0498 0499 0501	097C 097E 0982 0983 0985 0986 0987 0988	DFFLEN DRØTRK NMIFLG DMNIVC * RDYTMR DRGRAM DVERFL ATTCTR * * DFLBUF *RANDOM FI *FILE CONT	RMB RMB RMB RMB RMB RMB RMB	2 4 1 2 1 1 1 1 SECLEN SERVED AREA LOCKS AND BUFFERS	FROM THE FAT TO TRIGGER A WRITE FAT TO DISK SEQUENCE DIRECT ACCESS FILE RECORD LENGTH CURRENT TRACK NUMBER, DRIVES Ø,1,2,3 NMI FLAG: Ø=DON'T VECTOR <> Ø=YECTOR OUT NMI VECTOR: WHERE TO JUMP FOLLOWING AN NMI INTERRUPT IF THE NMI FLAG IS SET MOTOR TURN OFF TIMER RAM IMAGE OF DSKREG (\$FF4Ø) VERIFY FLAG: Ø=OFF, \$FF=ON READ/WRITE ATTEMPT COUNTER: NUMBER OF TIMES THE DISK WILL ATTEMPT TO RETRIEVE OR WRITE DATA BEFORE IT GIVES UP AND ISSUES AN ERROR. INITIALIZED TO SECLEN BY DISKBAS
0484 0485 0486 0487 0489 0491 0492 0493 0495 0496 0497 0498 0499 0501 0502	097C 097E 0982 0983 0985 0986 0987 0988	DFFLEN DRØTRK NMIFLG DWIFLG * RDYTMR DRGRAM DVERFL ATTCTR * * DFLBUF *RANDOM FI *FILE CONT	RMB RMB RMB RMB RMB RMB RMB	2 4 1 2 1 1 1 1 SECLEN SERVED AREA LOCKS AND BUFFERS	FROM THE FAT TO TRIGGER A WRITE FAT TO DISK SEQUENCE DIRECT ACCESS FILE RECORD LENGTH CURRENT TRACK NUMBER, DRIVES Ø,1,2,3 NMI FLAG: Ø=DON'T VECTOR <> Ø=YECTOR OUT NMI VECTOR: WHERE TO JUMP FOLLOWING AN NMI INTERRUPT IF THE NMI FLAG IS SET MOTOR TURN OFF TIMER RAM IMAGE OF DSKREG (\$FF4Ø) VERIFY FLAG: Ø=OFF, \$FF=ON READ/WRITE ATTEMPT COUNTER: NUMBER OF TIMES THE DISK WILL ATTEMPT TO RETRIEVE OR WRITE DATA BEFORE IT GIVES UP AND ISSUES AN ERROR. INITIALIZED TO SECLEN BY DISKBAS
0484 0485 0486 0487 0488 0499 0491 0492 0493 0494 0495 0497 0498 0499 0500 0501 0503 0504 0505	097C 097E 0982 0983 0985 0986 0987 0988	DFFLEN DRØTRK NMIFLG DNMIVC * RDYTMR DRGRAM DVERFL ATTCTR * * DFLBUF *RANDOM FI *FILE CONT *GRAPHIC F	RMB RMB RMB RMB RMB RMB RMB	2 4 1 2 1 1 1 1 SECLEN SERVED AREA LOCKS AND BUFFERS	FROM THE FAT TO TRIGGER A WRITE FAT TO DISK SEQUENCE DIRECT ACCESS FILE RECORD LENGTH CURRENT TRACK NUMBER, DRIVES Ø,1,2,3 NMI FLAG: Ø=DON'T VECTOR <> Ø=YECTOR OUT NMI VECTOR: WHERE TO JUMP FOLLOWING AN NMI INTERRUPT IF THE NMI FLAG IS SET MOTOR TURN OFF TIMER RAM IMAGE OF DSKREG (\$FF4Ø) VERIFY FLAG: Ø=OFF, \$FF=ON READ/WRITE ATTEMPT COUNTER: NUMBER OF TIMES THE DISK WILL ATTEMPT TO RETRIEVE OR WRITE DATA BEFORE IT GIVES UP AND ISSUES AN ERROR. INITIALIZED TO SECLEN BY DISKBAS
0484 0485 0486 0487 0488 0499 0491 0492 0493 0494 0495 0497 0498 0499 0500 0501 0502 0506 0506	097C 097E 0982 0983 0985 0986 0987 0988	DFFLEN DRØTRK NMIFLG DWIFLG * RDYTMR DRGRAM DVERFL ATTCTR * * DFLBUF *RANDOM FI *FILE CONT	RMB RMB RMB RMB RMB RMB RMB	2 4 1 2 1 1 1 1 SECLEN SERVED AREA LOCKS AND BUFFERS	FROM THE FAT TO TRIGGER A WRITE FAT TO DISK SEQUENCE DIRECT ACCESS FILE RECORD LENGTH CURRENT TRACK NUMBER, DRIVES Ø,1,2,3 NMI FLAG: Ø=DON'T VECTOR <> Ø=YECTOR OUT NMI VECTOR: WHERE TO JUMP FOLLOWING AN NMI INTERRUPT IF THE NMI FLAG IS SET MOTOR TURN OFF TIMER RAM IMAGE OF DSKREG (\$FF4Ø) VERIFY FLAG: Ø=OFF, \$FF=ON READ/WRITE ATTEMPT COUNTER: NUMBER OF TIMES THE DISK WILL ATTEMPT TO RETRIEVE OR WRITE DATA BEFORE IT GIVES UP AND ISSUES AN ERROR. INITIALIZED TO SECLEN BY DISKBAS
0484 0485 0486 0487 0488 0499 0491 0492 0493 0494 0495 0496 0497 0498 0499 0500 0501 0502 0503 0505 0506	097C 097E 0982 0983 0985 0986 0987 0988	DFFLEN DRØTRK NMIFLG DNMIVC * RDYTMR DRGRAM DVERFL ATTCTR * * * DFLBUF *RANDOM FI *FILE CONT *GRAPHIC F *BASIC PRC *VARIABLE	RMB RMB RMB RMB RMB RMB RMB RMB RMB RMB	2 4 1 2 1 1 1 1 SECLEN SERVED AREA LOCKS AND BUFFERS ESERVED AREA	FROM THE FAT TO TRIGGER A WRITE FAT TO DISK SEQUENCE DIRECT ACCESS FILE RECORD LENGTH CURRENT TRACK NUMBER, DRIVES Ø,1,2,3 NMI FLAG: Ø=DON'T VECTOR <> Ø=YECTOR OUT NMI VECTOR: WHERE TO JUMP FOLLOWING AN NMI INTERRUPT IF THE NMI FLAG IS SET MOTOR TURN OFF TIMER RAM IMAGE OF DSKREG (\$FF4Ø) VERIFY FLAG: Ø=OFF, \$FF=ON READ/WRITE ATTEMPT COUNTER: NUMBER OF TIMES THE DISK WILL ATTEMPT TO RETRIEVE OR WRITE DATA BEFORE IT GIVES UP AND ISSUES AN ERROR. INITIALIZED TO SECLEN BY DISKBAS
0484 0485 0486 0487 0488 0499 0491 0492 0493 0494 0495 0496 0497 0498 0496 0501 0502 0503 0504 0506 0507 0508	097C 097E 0982 0983 0985 0986 0987 0988	DFFLEN DRØTRK NMIFLG DNMIVC * RDYTMR DRGRAM DVERFL ATTCTR * * DFLBUF *RANDOM FI *FILE CONT *GRAPHIC F	RMB RMB RMB RMB RMB RMB RMB RMB RMB RMB	2 4 1 2 1 1 1 1 SECLEN SERVED AREA LOCKS AND BUFFERS ESERVED AREA	FROM THE FAT TO TRIGGER A WRITE FAT TO DISK SEQUENCE DIRECT ACCESS FILE RECORD LENGTH CURRENT TRACK NUMBER, DRIVES Ø,1,2,3 NMI FLAG: Ø=DON'T VECTOR <> Ø=YECTOR OUT NMI VECTOR: WHERE TO JUMP FOLLOWING AN NMI INTERRUPT IF THE NMI FLAG IS SET MOTOR TURN OFF TIMER RAM IMAGE OF DSKREG (\$FF4Ø) VERIFY FLAG: Ø=OFF, \$FF=ON READ/WRITE ATTEMPT COUNTER: NUMBER OF TIMES THE DISK WILL ATTEMPT TO RETRIEVE OR WRITE DATA BEFORE IT GIVES UP AND ISSUES AN ERROR. INITIALIZED TO SECLEN BY DISKBAS
0484 0485 0486 0487 0488 0499 0491 0492 0493 0496 0497 0498 0499 0501 0502 0503 0504 0505 0506 0506 0506 0508	097C 097E 0982 0983 0985 0986 0987 0988	DFFLEN DRØTRK NMIFLG DNMIVC * RDYTMR DRGRAM DVERFL ATTCTR * * * DFLBUF *RANDOM FI *FILE CONT *GRAPHIC F *BASIC PRC *VARIABLE	RMB RMB RMB RMB RMB RMB RMB RMB RMB RMB	2 4 1 2 1 1 1 1 SECLEN SERVED AREA LOCKS AND BUFFERS ESERVED AREA	FROM THE FAT TO TRIGGER A WRITE FAT TO DISK SEQUENCE DIRECT ACCESS FILE RECORD LENGTH CURRENT TRACK NUMBER, DRIVES Ø,1,2,3 NMI FLAG: Ø=DON'T VECTOR <> Ø=YECTOR OUT NMI VECTOR: WHERE TO JUMP FOLLOWING AN NMI INTERRUPT IF THE NMI FLAG IS SET MOTOR TURN OFF TIMER RAM IMAGE OF DSKREG (\$FF4Ø) VERIFY FLAG: Ø=OFF, \$FF=ON READ/WRITE ATTEMPT COUNTER: NUMBER OF TIMES THE DISK WILL ATTEMPT TO RETRIEVE OR WRITE DATA BEFORE IT GIVES UP AND ISSUES AN ERROR. INITIALIZED TO SECLEN BY DISKBAS
0484 0485 0486 0487 0488 0499 0491 0492 0493 0494 0495 0496 0497 0498 0496 0501 0502 0503 0504 0506 0507 0508	097C 097E 0982 0983 0985 0986 0987 0988	DFFLEN DRØTRK NMIFLG DNMIVC * RDYTMR DRGRAM DVERFL ATTCTR * * * DFLBUF *RANDOM FI *FILE CONT *GRAPHIC F *BASIC PRC *VARIABLE	RMB RMB RMB RMB RMB RMB RMB RMB RMB RMB	2 4 1 2 1 1 1 1 SECLEN SERVED AREA LOCKS AND BUFFERS ESERVED AREA	FROM THE FAT TO TRIGGER A WRITE FAT TO DISK SEQUENCE DIRECT ACCESS FILE RECORD LENGTH CURRENT TRACK NUMBER, DRIVES Ø,1,2,3 NMI FLAG: Ø=DON'T VECTOR <> Ø=YECTOR OUT NMI VECTOR: WHERE TO JUMP FOLLOWING AN NMI INTERRUPT IF THE NMI FLAG IS SET MOTOR TURN OFF TIMER RAM IMAGE OF DSKREG (\$FF4Ø) VERIFY FLAG: Ø=OFF, \$FF=ON READ/WRITE ATTEMPT COUNTER: NUMBER OF TIMES THE DISK WILL ATTEMPT TO RETRIEVE OR WRITE DATA BEFORE IT GIVES UP AND ISSUES AN ERROR. INITIALIZED TO SECLEN BY DISKBAS
0484 0485 0486 0487 0488 0499 0491 0492 0493 0494 0496 0497 0498 0500 0501 0502 0503 0504 0505 0506 0507 0508 0508	097C 097E 0982 0983 0985 0986 0987 0988	DFFLEN DRØTRK NMIFLG DNMIVC * RDYTMR DRGRAM DVERFL ATTCTR * * * DFLBUF *RANDOM FI *FILE CONT *GRAPHIC F *BASIC PRC *VARIABLE	RMB RMB RMB RMB RMB RMB RMB RMB RMB CLE RE FROL B PAGE R OGRAM	2 4 1 2 1 1 1 1 SECLEN SERVED AREA LOCKS AND BUFFERS ESERVED AREA	FROM THE FAT TO TRIGGER A WRITE FAT TO DISK SEQUENCE DIRECT ACCESS FILE RECORD LENGTH CURRENT TRACK NUMBER, DRIVES Ø,1,2,3 NMI FLAG: Ø=DON'T VECTOR <> Ø=YECTOR OUT NMI VECTOR: WHERE TO JUMP FOLLOWING AN NMI INTERRUPT IF THE NMI FLAG IS SET MOTOR TURN OFF TIMER RAM IMAGE OF DSKREG (\$FF4Ø) VERIFY FLAG: Ø=OFF, \$FF=ON READ/WRITE ATTEMPT COUNTER: NUMBER OF TIMES THE DISK WILL ATTEMPT TO RETRIEVE OR WRITE DATA BEFORE IT GIVES UP AND ISSUES AN ERROR. INITIALIZED TO SECLEN BY DISKBAS
0484 0485 0486 0487 0488 0499 0491 0492 0493 0494 0496 0497 0498 0499 0501 0501 0503 0504 0505 0506 0507 0508 0509 0509 0501 0509	097C 097E 0982 0983 0985 0986 0987 0988	DFFLEN DRØTRK NMIFLG DNMIFLG DNMIFLG TOWNIFL * RDYTMR DRGRAM DVERFL ATTCTR * * DFLBUF *RANDOM FI *FILE CONT *GRAPHIC F *BASIC PRC *VARIABLE *ARRAY STC	RMB RMB RMB RMB RMB RMB RMB RMB RMB CLE RE FROL B PAGE R OGRAM	2 4 1 2 1 1 1 1 SECLEN SERVED AREA LOCKS AND BUFFERS ESERVED AREA	FROM THE FAT TO TRIGGER A WRITE FAT TO DISK SEQUENCE DIRECT ACCESS FILE RECORD LENGTH CURRENT TRACK NUMBER, DRIVES Ø,1,2,3 NMI FLAG: Ø=DON'T VECTOR <> Ø=YECTOR OUT NMI VECTOR: WHERE TO JUMP FOLLOWING AN NMI INTERRUPT IF THE NMI FLAG IS SET MOTOR TURN OFF TIMER RAM IMAGE OF DSKREG (\$FF4Ø) VERIFY FLAG: Ø=OFF, \$FF=ON READ/WRITE ATTEMPT COUNTER: NUMBER OF TIMES THE DISK WILL ATTEMPT TO RETRIEVE OR WRITE DATA BEFORE IT GIVES UP AND ISSUES AN ERROR. INITIALIZED TO SECLEN BY DISKBAS
0484 0485 0486 0487 0488 0499 0491 0492 0493 0494 0495 0497 0498 0499 0501 0502 0503 0504 0505 0506 0507 0508 0501 0501 0501	097C 097E 0982 0983 0985 0986 0987 0988	DFFLEN DRØTRK NMIFLG DNMIFLG DNMIFLG TOWNIFL * RDYTMR DRGRAM DVERFL ATTCTR * * DFLBUF *RANDOM FI *FILE CONT *GRAPHIC F *BASIC PRC *VARIABLE *ARRAY STC	RMB RMB RMB RMB RMB RMB RMB RMB RMB CLE RE FROL B PAGE R OGRAM	2 4 1 2 1 1 1 1 SECLEN SERVED AREA LOCKS AND BUFFERS ESERVED AREA	FROM THE FAT TO TRIGGER A WRITE FAT TO DISK SEQUENCE DIRECT ACCESS FILE RECORD LENGTH CURRENT TRACK NUMBER, DRIVES Ø,1,2,3 NMI FLAG: Ø=DON'T VECTOR <> Ø=YECTOR OUT NMI VECTOR: WHERE TO JUMP FOLLOWING AN NMI INTERRUPT IF THE NMI FLAG IS SET MOTOR TURN OFF TIMER RAM IMAGE OF DSKREG (\$FF4Ø) VERIFY FLAG: Ø=OFF, \$FF=ON READ/WRITE ATTEMPT COUNTER: NUMBER OF TIMES THE DISK WILL ATTEMPT TO RETRIEVE OR WRITE DATA BEFORE IT GIVES UP AND ISSUES AN ERROR. INITIALIZED TO SECLEN BY DISKBAS
0484 0485 0486 0487 0489 0499 0491 0492 0493 0494 0496 0497 0498 0496 0501 0502 0503 0504 0506 0507 0508 0509 0510 0512 0512	097C 097E 0982 0983 0985 0986 0987 0988	DFFLEN DRØTRK NMIFLG DNMIVC * RDYTMR DRGRAM DVERFL ATTCTR * * * * * * * * * * * * * * * * * *	RMB RMB RMB RMB RMB RMB RMB RMB RMB	2 4 1 2 1 1 1 1 SECLEN SERVED AREA LOCKS AND BUFFERS ESERVED AREA	FROM THE FAT TO TRIGGER A WRITE FAT TO DISK SEQUENCE DIRECT ACCESS FILE RECORD LENGTH CURRENT TRACK NUMBER, DRIVES Ø,1,2,3 NMI FLAG: Ø=DON'T VECTOR <> Ø=YECTOR OUT NMI VECTOR: WHERE TO JUMP FOLLOWING AN NMI INTERRUPT IF THE NMI FLAG IS SET MOTOR TURN OFF TIMER RAM IMAGE OF DSKREG (\$FF4Ø) VERIFY FLAG: Ø=OFF, \$FF=ON READ/WRITE ATTEMPT COUNTER: NUMBER OF TIMES THE DISK WILL ATTEMPT TO RETRIEVE OR WRITE DATA BEFORE IT GIVES UP AND ISSUES AN ERROR. INITIALIZED TO SECLEN BY DISKBAS
0484 0485 0486 0487 0489 0491 0492 0493 0494 0495 0496 0497 0498 0496 0501 0502 0503 0504 0506 0507 0508 0507 0508 0501 0511 0511	097C 097E 0982 0983 0985 0986 0987 0988	DFFLEN DRØTRK NMIFLG DNMIFLG DNMIFLG TOWNIFL * RDYTMR DRGRAM DVERFL ATTCTR * * DFLBUF *RANDOM FI *FILE CONT *GRAPHIC F *BASIC PRC *VARIABLE *ARRAY STC	RMB RMB RMB RMB RMB RMB RMB RMB RMB	2 4 1 2 1 1 1 1 SECLEN SERVED AREA LOCKS AND BUFFERS ESERVED AREA	FROM THE FAT TO TRIGGER A WRITE FAT TO DISK SEQUENCE DIRECT ACCESS FILE RECORD LENGTH CURRENT TRACK NUMBER, DRIVES Ø,1,2,3 NMI FLAG: Ø=DON'T VECTOR <> Ø=YECTOR OUT NMI VECTOR: WHERE TO JUMP FOLLOWING AN NMI INTERRUPT IF THE NMI FLAG IS SET MOTOR TURN OFF TIMER RAM IMAGE OF DSKREG (\$FF4Ø) VERIFY FLAG: Ø=OFF, \$FF=ON READ/WRITE ATTEMPT COUNTER: NUMBER OF TIMES THE DISK WILL ATTEMPT TO RETRIEVE OR WRITE DATA BEFORE IT GIVES UP AND ISSUES AN ERROR. INITIALIZED TO SECLEN BY DISKBAS
0484 0485 0486 0487 0489 0499 0491 0492 0493 0494 0496 0497 0498 0496 0501 0502 0503 0504 0506 0507 0508 0509 0510 0512 0512	097C 097E 0982 0983 0985 0986 0987 0988	DFFLEN DRØTRK NMIFLG DNMIVC * RDYTMR DRGRAM DVERFL ATTCTR * * * * * * * * * * * * * * * * * *	RMB RMB RMB RMB RMB RMB RMB RMB RMB	2 4 1 2 1 1 1 1 SECLEN SERVED AREA LOCKS AND BUFFERS ESERVED AREA	FROM THE FAT TO TRIGGER A WRITE FAT TO DISK SEQUENCE DIRECT ACCESS FILE RECORD LENGTH CURRENT TRACK NUMBER, DRIVES Ø,1,2,3 NMI FLAG: Ø=DON'T VECTOR <> Ø=YECTOR OUT NMI VECTOR: WHERE TO JUMP FOLLOWING AN NMI INTERRUPT IF THE NMI FLAG IS SET MOTOR TURN OFF TIMER RAM IMAGE OF DSKREG (\$FF4Ø) VERIFY FLAG: Ø=OFF, \$FF=ON READ/WRITE ATTEMPT COUNTER: NUMBER OF TIMES THE DISK WILL ATTEMPT TO RETRIEVE OR WRITE DATA BEFORE IT GIVES UP AND ISSUES AN ERROR. INITIALIZED TO SECLEN BY DISKBAS
0484 0485 0486 0487 0489 0491 0492 0493 0494 0495 0496 0497 0498 0496 0501 0502 0503 0504 0506 0507 0508 0507 0508 0501 0511 0511	097C 097E 0982 0983 0985 0986 0987 0988	DFFLEN DRØTRK NMIFLG DNMIVC * RDYTMR DRGRAM DVERFL ATTCTR * * * * * * * * * * * * * * * * * *	RMB RMB RMB RMB RMB RMB RMB RMB RMB RMB	2 4 1 2 1 1 1 1 SECLEN SERVED AREA LOCKS AND BUFFERS ESERVED AREA	FROM THE FAT TO TRIGGER A WRITE FAT TO DISK SEQUENCE DIRECT ACCESS FILE RECORD LENGTH CURRENT TRACK NUMBER, DRIVES Ø,1,2,3 NMI FLAG: Ø=DON'T VECTOR <> Ø=YECTOR OUT NMI VECTOR: WHERE TO JUMP FOLLOWING AN NMI INTERRUPT IF THE NMI FLAG IS SET MOTOR TURN OFF TIMER RAM IMAGE OF DSKREG (\$FF4Ø) VERIFY FLAG: Ø=OFF, \$FF=ON READ/WRITE ATTEMPT COUNTER: NUMBER OF TIMES THE DISK WILL ATTEMPT TO RETRIEVE OR WRITE DATA BEFORE IT GIVES UP AND ISSUES AN ERROR. INITIALIZED TO SECLEN BY DISKBAS
0484 0485 0486 0487 0488 0499 0491 0492 0493 0496 0497 0498 0496 0501 0502 0503 0504 0505 0506 0506 0507 0508 0509 0510 0511 0513 0514	097C 097E 0982 0983 0985 0986 0987 0988	DFFLEN DRØTRK NMIFLG DNMIFLG ** RDYTMR DRGRAM DVERFL ATTCTR * * DFLBUF **FILE CONT **GRAPHIC F **BASIC PRC *VARIABLE *ARRAY STC * * FREE MEM ** *STACK	RMB RMB RMB RMB RMB RMB RMB RMB RMB RMB	2 4 1 2 1 1 1 1 SECLEN SERVED AREA LOCKS AND BUFFERS ESERVED AREA	FROM THE FAT TO TRIGGER A WRITE FAT TO DISK SEQUENCE DIRECT ACCESS FILE RECORD LENGTH CURRENT TRACK NUMBER, DRIVES Ø,1,2,3 NMI FLAG: Ø=DON'T VECTOR <> Ø=YECTOR OUT NMI VECTOR: WHERE TO JUMP FOLLOWING AN NMI INTERRUPT IF THE NMI FLAG IS SET MOTOR TURN OFF TIMER RAM IMAGE OF DSKREG (\$FF4Ø) VERIFY FLAG: Ø=OFF, \$FF=ON READ/WRITE ATTEMPT COUNTER: NUMBER OF TIMES THE DISK WILL ATTEMPT TO RETRIEVE OR WRITE DATA BEFORE IT GIVES UP AND ISSUES AN ERROR. INITIALIZED TO SECLEN BY DISKBAS
0484 0485 0486 0487 0488 0499 0491 0492 0493 0494 0496 0497 0498 0501 0502 0503 0504 0505 0506 0507 0508 0509 0510 0511 0512 0513 0515	097C 097E 0982 0983 0985 0986 0987 0988	DFFLEN DRØTRK NMIFLG DNMIVC * RDYTMR DRGRAM DVERFL ATTCTR * * * DFLBUF *RANDOM FI *FILE CONT *GRAPHIC F *VARIABLE *ARRAY STC * * FREE MEM *STACK *STRING SF	RMB RMB RMB RMB RMB RMB RMB RMB RMB RMB	2 4 1 2 1 1 1 1 SECLEN SERVED AREA LOCKS AND BUFFERS ESERVED AREA	FROM THE FAT TO TRIGGER A WRITE FAT TO DISK SEQUENCE DIRECT ACCESS FILE RECORD LENGTH CURRENT TRACK NUMBER, DRIVES Ø,1,2,3 NMI FLAG: Ø=DON'T VECTOR <> Ø=YECTOR OUT NMI VECTOR: WHERE TO JUMP FOLLOWING AN NMI INTERRUPT IF THE NMI FLAG IS SET MOTOR TURN OFF TIMER RAM IMAGE OF DSKREG (\$FF4Ø) VERIFY FLAG: Ø=OFF, \$FF=ON READ/WRITE ATTEMPT COUNTER: NUMBER OF TIMES THE DISK WILL ATTEMPT TO RETRIEVE OR WRITE DATA BEFORE IT GIVES UP AND ISSUES AN ERROR. INITIALIZED TO SECLEN BY DISKBAS

```
Ø519
Ø52Ø
                     *END OF RAM
Ø521
0522 8000
                     ORG
                                $8000
Ø523
0524 8000
                                RMB
                                     $2000
                                                      EXTENDED BASIC ROM
Ø525 AØØØ
                                RMB
                                     $2000
                                                      COLOR BASIC ROM
Ø526 CØØØ
                     ROMPAK
                                EQU
Ø527 CØØØ
                     DOSBAS
                                RMB
                                     $2000
                                                      DISK BASIC ROM/ENHANCED BASIC INIT CODE
Ø528 EØØØ
                                RMB
                                     $1FØØ
                                                      ENHANCED BASIC
Ø529
Ø53Ø
                     * START OF ADDITIONAL VARIABLES USED BY SUPER EXTENDED BASIC
Ø531 FEØØ
                     H.CRSLOC
                               RMB
                                                      CURRENT LOCATION OF CURSOR
                                     2
                                                      X POSITION OF CURSOR
Ø532 FEØ2
                     H.CURSX
                                RMB
                                     1
Ø533 FEØ3
                     H.CURSY
                                RMR
                                     1
                                                      Y POSITION OF CURSOR
                     H.COLUMN
                                RMB
                                                      COLUMNS ON HI-RES ALPHA SCREEN
Ø534 FEØ4
                                     1
                     H.ROW
                                                      ROWS ON HI-RES ALPHA SCREEN
Ø535 FFØ5
                                RMR
                                     1
Ø536 FEØ6
                     H.DISPEN
                                RMB
                                     2
                                                      END OF HI-RES DISPLAY SCREEN
Ø537 FEØ8
                     H.CRSATT
                                RMB
                                      1
                                                      CURRENT CURSOR'S ATTRIBUTES
Ø538 FEØ9
                                RMB
                                                      UNUSED
                                     1
                     H.FCOLOR
                                                      FOREGROUND COLOR
                                RMR
0539 FF0A
                                     1
Ø54Ø FEØB
                     H.BCOLOR
                                RMB
                                     1
                                                      BACKGROUND COLOR
Ø541 FEØC
                     H.ONBRK
                                                      ON BRK GOTO LINE NUMBER
                                RMB
                                      2
Ø542 FEØE
                     H.ONERR
                                RMB
                                      2
                                                      ON ERR GOTO LINE NUMBER
0543 FF10
                     H.FRROR
                                RMB
                                     1
                                                      ERROR NUMBER ENCOUNTERED OR $FF (NO ERROR)
Ø544 FE11
                     H.ONERRS
                                RMB
                                      2
                                                      ON ERR SOURCE LINE NUMBER
Ø545 FE13
                     H.ERLINE
                                RMB
                                      2
                                                      LINE NUMBER WHERE ERROR OCCURRED
                     H.ONBRKS
                                RMB
                                                      ON BRK SOURCE LINE NUMBER
Ø546 FE15
                                      2
                     H.ERRBRK
                                RMR
                                                      STILL UNKNOWN, HAS TO DO WITH ERR, BRK
Ø547 FE17
                                     1
Ø548 FE18
                     H.PCOUNT
                                RMB
                                      1
                                                      PRINT COUNT, CHARACTERS TO BE HPRINTED
                                                      PRINT BUFFER, HPRINT CHARS. STORED HERE
Ø549 FE19
                     H.PBUF
                                RMR
                                     8Ø
Ø55Ø FE69
                                RMB
                                     132
                                                      UNUSED
                                                      INTERRUPT VALID FLAG. Ø=NOT VALID, $55=VALID
Ø551 FEED
                     INT.FLAG
                                RMB
                                     1
                     * TABLE OF JUMP VECTORS TO INTERRUPT SERVICING ROUTINES
0552
Ø553 FEEE
                     INT.JUMP
                     INT.SWI3
Ø554 FEEE
                                RMB
                                      3
Ø555 FEF1
                     INT.SWI2
                                RMB
                                     3
                                     3
Ø556 FEF4
                     INT.FIRQ
                                RMR
                     INT.IRO
0557 FFF7
                                RMR
                                     3
Ø558 FEFA
                     INT.SWI
                                RMR
                                     3
Ø559 FEFD
                     INT.NMI
                                RMB
                                      3
Ø56Ø
0561
                     * I/O AREA
Ø562
Ø563 FFØØ
                     PIAØ
                                EQU *
                                                      PERIPHERAL INTERFACE ADAPTER ONE
Ø564
Ø565 FFØØ
                                KEYBOARD ROW 1 AND RIGHT JOYSTICK SWITCH 1
                     BITØ
Ø566
                     BIT1
                                KEYBOARD ROW 2 AND LEFT JOYSTICK SWITCH 1
                                KEYBOARD ROW 3 AND RIGHT JOYSTICK SWITCH 2
Ø567
                     BIT2
                                KEYBOARD ROW 4 AND LEFT JOYSTICK SWITCH 2
Ø568
                     BIT3
                                KEYBOARD ROW 5
Ø569
                     BIT4
Ø57Ø
                     BIT5
                                KEYBOARD ROW 6
Ø571
                     BIT6
                                KEYBOARD ROW 7
                                JOTSTICK COMPARISON IINPUT
0572
                     BIT7
Ø573
Ø574 FFØ1
                     BITØ
                                CONTROL OF HSYNC (63.5ps)
                                                              \emptyset = IRQ* TO CPU DISABLED
Ø575
                                INTERRUPT
                                                              1 = IRQ* TO CPU ENABLED
                                CONTROL OF INTERRUPT
                     BIT1
                                                              \emptyset = FLAG SET ON FALLING EDGE OF HS
Ø576
Ø577
                                POLARITY
                                                              1 = FLAG SET ON RISING EDGE OF HS
Ø578
                     BIT2
                                NORMALLY 1
                                                              Ø = CHANGES FFØØ TO DATA DIRECTION
Ø579
                     BIT3
                                SFI 1
                                                              LSB OF TWO ANALOG MUX SELECT LINES
Ø58Ø
                     BIT4
                                ALWAYS 1
                                ALWAYS 1
Ø581
                     BIT5
Ø582
                     BIT6
                                NOT USED
                                HORIZONTAL SYNC INTERRUPT FLAG
Ø583
                     BIT7
Ø584
Ø585 FFØ2
                                KEYBOARD COLUMN 1
                     BITØ
                                KEYBOARD COLUMN 2
Ø586
                     BIT1
                                KEYBOARD COLUMN 3
Ø587
                     BIT2
Ø588
                     BIT3
                                KEYBOARD COLUMN 4
                                KEYBOARD COLUMN 5
Ø589
                     BIT4
                                KEYBOARD COLUMN 6
Ø59Ø
                     BIT5
                                KEYBOARD COLUMN 7 / RAM SIZE OUTPUT
Ø591
                     BIT6
Ø592
                     BIT7
                                KEYBOARD COLUMN 8
```

Ø593						
Ø593	FFØ3	BITØ	CONTROL OF VSVNC (16	667ms)	Ø = IRQ* TO CPU DISA	RI FD
Ø595	1195	DITE	INTERRUPT	00711137	1 = IRQ* TO CPU ENAB	
Ø596		BIT1	CONTROL OF INTERRUPT		Ø = FLAG SET ON FALL	
Ø597			POLARITY		1 = FLAG SET ON RISI	
Ø598		BIT2	NORMALLY 1		Ø = CHANGES FFØ2 TO	DATA DIRECTION
Ø599		BIT3	SEL 2		MSB OF TWO ANALOG MU	X SELECT LINES
Ø6ØØ		BIT4	ALWAYS 1			
0601		BIT5	ALWAYS 1			
0602		BIT6	NOT USED			
0603		BIT7	FIELD SYNC INTERRUPT	FLAG		
0604						
0605	FFØ4		RMB 28	PIAØ	IMAGES	
Ø6Ø6	FF2Ø	DA				
Ø6Ø7	FF2Ø	PIA1	EQU *	PERIP	HERAL INTERFACE ADAPTE	R TWO
Ø6Ø8						
Ø6Ø9	FF2Ø	BITØ	CASSETTE DATA INPUT			
Ø61Ø		BIT1	RS-232C DATA OUTPUT			
Ø611		BIT2	6 BIT D/A LSB			
Ø612		BIT3	6 BIT D/A			
Ø613		BIT4	6 BIT D/A			
Ø614		BIT5	6 BIT D/A 6 BIT D/A			
Ø615 Ø616		BIT6	6 BIT D/A MSB			
Ø617		BIT7	O BIT D/A MSB			
Ø618	EE21	BITØ	CONTROL OF CD		Ø = FIRO* TO CPU DIS	ARLED
Ø619	1121	שווט	(RS-232C STATUS)		1 = FIRO* TO CPU ENA	
Ø62Ø		BIT1	CONTROL OF INTERRUPT		Ø = FLAG SET ON FALL	
Ø621		DITT	POLARITY		1 = FLAG SET ON RISI	
Ø622		BIT2	NORMALLY 1		Ø = CHANGES FF2Ø TO	
Ø623		BIT3	CASSETTE MOTOR CONTRO	L	$\emptyset = OFF$ $1 = ON$	DATA DIRECTION
Ø624		BIT4	ALWAYS 1	_		
Ø625		BIT5	ALWAYS 1			
Ø626		BIT6	NOT USED			
Ø627		BIT7	CD INTERRUPT FLAG			
Ø628						
Ø629	FF22	BITØ	RS-232C DATA INPUT			
Ø63Ø		BIT1	SINGLE BIT SOUND OUTP	UT		
Ø631		BIT2	RAM SIZE INPUT			
Ø632		BIT3	RGB MONITOR SENSING I	NPUT	CSS	
Ø633		BIT4	VDG CONTROL OUTPUT		GMØ & UPPER/LOWER CA	SE*
Ø634		BIT5	VDG CONTROL OUTPUT		GM1 & INVERT	
Ø635		BIT6	VDG CONTROL OUTPUT		GM2	
Ø636		BIT7	VDG CONTROL OUTPUT		A*/G	
Ø637						
Ø638	FF23	BITØ	CONTROL OF CARTRIDGE		Ø = FIRQ* TO CPU DIS	
Ø639			INTERRUPT		1 = FIRQ* TO CPU ENA	
Ø64Ø		BIT1	CONTROL OF INTERRUPT		Ø = FLAG SET ON FALL	
Ø641		DITO	POLARITY		1 = FLAG SET ON RISI	
Ø642 Ø643		BIT2	NORMALLY 1		Ø = CHANGES FF22 TO	DATA DIRECTION
Ø644		BIT3	SOUND ENABLE			
Ø645		BIT4 BIT5	ALWAYS 1 ALWAYS 1			
Ø646		BIT6	NOT USED			
Ø647		BIT7	CARTRIDGE INTERRUPT F	LAG		
Ø648		DIII	CARTRIBUE INTERROTT I	LAG		
	FF24		RMB 28	PTA1	IMAGES	
	FF4Ø	PIA2	5 20		11111420	
	FF4Ø	DSKREG	RMB 1	DISK	CONTROL REGISTER	
Ø652						
Ø653	FF4Ø	BITØ	DRIVE SELECT Ø			
Ø654		BIT1	DRIVE SELECT 1			
Ø655		BIT2	DRIVE SELECT 2			
Ø656		BIT3	DRIVE MOTOR ENABLE		$\emptyset = MOTORS OFF$	1 = MOTORS ON
Ø657		BIT4	WRITE PRECOMPENSATION		$\emptyset = NO PRECOMP$	1 = PRECOMP
Ø658		BIT5	DENSITY FLAG		$\emptyset = SINGLE$	1 = DOUBLE
Ø659		BIT6	DRIVE SELECT 3			
Ø66Ø		BIT7	HALT FLAG		$\emptyset = DISABLED$	1 = ENABLED
Ø661						
	FF41		RMB 7	DSKRE	G IMAGES	
Ø663						
Ø664			DISK CONTROLLER INTERNA			
	FF48	FDCREG	RMB 1	STATU	S/COMMAND REGISTER	
Ø666						

Ø667 Ø668 Ø669 Ø670 Ø671 Ø672 Ø673 Ø674 Ø675 Ø676		COMMANDS	TYPE I I I I I I I II III III III III	COMMAND RESTORE SEEK STEP STEP IN STEP OUT READ SECTOR WRITE SECTOR READ ADDRESS READ TRACK WRITE TRACK FORCE INTERRUPT	CODE \$Ø3 \$17 \$23 \$43 \$53 \$8Ø \$AØ \$CØ \$E4 \$F4	
Ø679 Ø68Ø Ø681 Ø682 Ø683 Ø684 Ø685 Ø686		STATUS	BIT SØ S1 S2 S3 S4 S5 S6	TYPE I BUSY INDEX TRACK Ø CRC ERROR SEEK ERROR HEAD LOADED WRITE PROTECT NOT READY	READ ADDRESS/SECTOR/TRACK BUSY DRQ LOST DATA CRC ERROR (EXCEPT TRACK) RNF (EXCEPT TRACK) RECORD TYPE (SECTOR ONLY)	WRITE SECTOR/TRACK BUSY DRQ LOST DATA CRC ERROR (EXCEPT TRACK) RNF (EXCEPT TRACK) WRITE FAULT WRITE PROTECT NOT READY
Ø691 Ø692 Ø693	FF49 FF4A FF4B FF4C		RMB RMB RMB RMB	1 1 1 4	TRACK REGISTER SECTOR REGISTER DATA REGISTER FDCREG IMAGES	
Ø696 Ø697	FF61 FF62	* RS-232	RMB RMB RMB RMB	16 1 1 1 5	UNUSED SPACE X COORDINATE FOR X-PAD Y COORDINATE FOR X-PAD STATUS REGISTER FOR X-PAD UNUSED	
0701 0702 0703 0704	FF68 FF69 FF6A FF6B FF6C	K3-232	RMB RMB RMB RMB RMB	1 1 1 1 1 4	READ/WRITE DATA REGISTER STATUS REGISTER COMMAND REGISTER CONTROL REGISTER	
Ø7Ø6 Ø7Ø7	FF7Ø FF7D FF7E		RMB RMB RMB RMB	13 1 1 1	SOUND/SPEECH CARTRIDGE RESE SOUND/SPEECH CARTRIDGE REAL MULTI-PAK PROGRAMMING REGIS	D/WRITE
0711 0712 0713	FF8Ø	+ VIDEO (RMB	64 REGISTERS	RESERVED FOR FUTURE EXPANS	CON
Ø714 Ø715	FF9Ø	INITØ	RMB	1	INITIALIZATION REGISTER Ø	
0716 0717 0718 0719 0720 0721 0722 0723	FF9Ø	BITØ BIT1 BIT2 BIT3 BIT4 BIT5 BIT6 BIT7	MCØ MC1 MC2 MC3 FEN IEN M/P COCO		ROM MAP CONTROL (SEE ROM MAP CONTROL (SEE STANDARD SCS 1 = DRAM AT ØXFEXX I 1 = CHIP FIRQ OUTPUT 1 = CHIP IRQ OUTPUT 1 = MMU ENABLED 1 = COCO 1 & 2 COMPA	E TABLE BELOW) IS CONSTANT T ENABLED ENABLED
Ø724 Ø725 Ø726 Ø727 Ø728 Ø729			MC1 Ø 1	MCØ × Ø 1	ROM MAPPING 16K INTERNAL, 16K EXTERNAL 32K INTERNAL 32L EXTERNAL (EXCEPT FOR VE	ECTORS)
	FF91	INIT1	RMB	1	INITIALIZATION REGISTER 1	
	FF91	BITØ BIT1 BIT2 BIT3 BIT4 BIT5	TR TINS		MMU TASK REGISTER SE TIMER INPUT SELECT:	
Ø738 Ø739 Ø74Ø		BIT6 BIT7				·

ORIGIN:SPECTRAL ASSOC REVISED:12/26/1999 WALTER K ZYDHEK

Ø741									
Ø742	FF92	IROENR	RMB	1	IRO IN	ITERRUPT	ENABLE	REGIST	ER
Ø743		•			•				
Ø744	FF92	BITØ	EIØ			CARTRI	DGE IRQ	ENABLE	:D
Ø745		BIT1	EI1			KEYB0A	RD IRQ	ENABLED)
Ø746		BIT2	EI2			SERIAL	DATA I	RQ ENAB	LED
Ø747		BIT3	VBORD					-	ENABLED
Ø748		BIT4	HBORD						Q ENABLED
Ø749		BIT5	TMR			INTERR	UPT FRO	M TIMER	ENABLED
Ø75Ø		BIT6							
Ø751		BIT7							
Ø752 Ø753	FF03	FIRQENR	RMB	1	FIDO I	NTERRUP	T FNARI	F DEGIS	TFD
Ø753	1133	TINGLIN	KIID	-	I INQ I	MILKKOI	I LIVADE	L KLUIS) I LIX
Ø755	FF93	BITØ	EIØ			CARTRI	DGE FIR	O ENABL	.ED
Ø756		BIT1	EI1				RD FIRQ		
Ø757		BIT2	EI2				DATA F		
Ø758		BIT3	VBORD			VERTIC	AL BORD	ER FIRQ	ENABLED
Ø759		BIT4	HBORD						RQ ENABLED
Ø76Ø		BIT5	TMR			INTERR	UPT FRO	M TIMER	RENABLED
Ø761		BIT6							
Ø762		BIT7							
Ø763 Ø764	EEQ/	V.TIMER	RMB	2	TIMED	REGISTE	D		
Ø765		V. IIIIEK	RMB	2		ED FOR		FYPANST	· O N
Ø766	1150		KIID	_	KLJLKV	LD TOR	TOTORE	LXI XIIOI	.011
Ø767	FF98	VIDEOREG	RMB	1	VIDEO	MODE RE	GISTER		
Ø768			-						
Ø769	FF98	BITØ	LPRØ			LINES	PER ROW	(SEE T	ABLE BELOW)
Ø77Ø		BIT1	LPR1			LINES	PER ROW	(SEE T	ABLE BELOW)
0771		BIT2	LPR2						ABLE BELOW)
Ø772		BIT3	H5Ø				Hz VER		
Ø773		BIT4	MOCH						OMPOSITE)
Ø774		BIT5	BPI			1 = BO	RST PHA	SE INVE	RIED
Ø775 Ø776		BIT6 BIT7	ВР			α – ΔΙ	PHA, 1	_ DIT D	OLANE
Ø777		DI17	БГ			v – AL	FIIA, I	- DII F	LANL
Ø778		LPR2	LPR1	LPRØ	LINES	PER CHA	RACTER	ROW	
Ø779		Ø	Ø	Ø	1			ICS MOD	ES)
Ø78Ø		Ø	Ø	1	2		(COCO	1 & 2 0	NLY)
Ø781		Ø	1	Ø	3		(COCO	1 & 2 0	NLY)
Ø782		Ø	1	1	8				
Ø783		1	Ø	Ø	9	WED)			
Ø784 Ø785		1 1	Ø 1	1 Ø	(RESER	(VED)			
Ø786		1	1	1	(RESER	(VFD)			
Ø787		-	-	-	(11202)	,			
Ø788	FF99	VIDEOREG	RMB	1	VIDEO	MODE RE	GISTER		
Ø789									
Ø79Ø	FF99	BITØ	CRESØ				RESOLUT		
Ø791		BIT1	CRES1				RESOLUT		•••
Ø792		BIT2	HRESØ				NTAL RE		
Ø793 Ø794		BIT3 BIT4	HRES1 HRES2				NTAL RE		
Ø795		BIT5	LPFØ						TABLE BELOW)
Ø796		BIT6	LPF1						TABLE BELOW)
Ø797		BIT7							
Ø798									
Ø799			LPF1	LPFØ	LINES	PER FIE	LD		
0800			Ø	Ø	192				
0801			Ø	1	200				
Ø8Ø2			1 1	Ø	RESERV	LD			
Ø8Ø3 Ø8Ø4			1	1	225				
Ø8Ø5		* VIDEO RE	: TII 102	TON					
Ø8Ø6		.IDEO KL		$: BP = \emptyset, COCO =$	= Ø				
Ø8Ø7				MODE		HRES1	HRESØ	CRES1	CRESØ
0808				32 CHARACTER	Ø		Ø		1
Ø8Ø9				40 CHARACTER	Ø		1		1
Ø81Ø				80 CHARACTER	1		1		1
Ø811			GRAPH	ICS: $BP = 1$, COC				00563	00500
Ø812 Ø813				PIXELSxCOLORS 640x4	HRES2	HRES1 1		CRES1 Ø	
Ø814				64Øx2	1	Ø	1	Ø	1 Ø
~311				- : w/\=	-	~	-	~	-

Ø815	512x4	1	1	Ø	Ø	1			
Ø816	512x2	1	Ø	Ø	Ø	Ø			
Ø817	32Øx16	1	1	1	1	Ø			
Ø818	32Øx4	1	Ø	1	Ø	1			
Ø819	256x16	1	1	Ø	1	Ø			
0820	256x4	1	Ø	Ø	Ø	1			
0821	256x2	Ø	1	Ø	Ø	Ø			
Ø822	160×16	1	Ø	1	1	Ø			
Ø823 Ø824	* COCO MODE SELECTION								
Ø825	" COCO MODE SELECTION	DISPL	AY MODE		REG	FF22			
Ø826		V2	V1	٧Ø	7	6	5	4	3
Ø827	ALPHA	Ø	Ø	Ø	Ø	X	X	Ø	CSS
Ø828	ALPHA INVERTED	Ø	Ø	Ø	Ø	Х	х	Ø	CSS
Ø829	SEMIGRAPHICS 4	Ø	Ø	Ø	Ø	Х	Х	Ø	Х
Ø83Ø	64x64 COLOR GRAPHICS	Ø	Ø	1	1	Ø	Ø	Ø	CSS
Ø831	128x64 GRAPHICS	Ø	Ø	1	1	Ø	Ø	1	CSS
Ø832	128x64 COLOR GRAPHICS	Ø	1	Ø	1	Ø	1	Ø	CSS
Ø833	128x96 GRAPHICS	Ø	1	1	1	Ø	1	1	CSS
Ø834	128x96 COLOR GRAPHICS	1	Ø	Ø	1	1	Ø	Ø	CSS
Ø835	128x96 GRAPHICS 128x96 COLOR GRAPHICS	1 1	Ø	1 Ø	1	1	Ø	1 Ø	CSS
Ø836 Ø837	256x192 GRAPHICS	1	1 1	Ø	1 1	1 1	1 1	ข 1	CSS CSS
Ø838	230X132 dixi ii103	1	1	υ	1	1	1	1	633
Ø839	* ALPHANUMERIC MODES								
Ø84Ø	TEXT SCREEN MEMORY								
Ø841	EVEN BYTE								
Ø842	BITØ	CHARA	CTER BIT	Ø					
Ø843	BIT1		CTER BIT						
Ø844	BIT2		CTER BIT						
Ø845	BIT3		CTER BIT						
Ø846	BIT4		CTER BIT						
Ø847 Ø848	BIT5 BIT6		CTER BIT CTER BIT						
Ø849	BIT7	СПАКА	CIEK DII	O					
Ø85Ø	DITT								
Ø851	ODD BYTE								
Ø852	BITØ	BGNDØ	BACKGR	OUND	COLOR B	IT (PAL	ETTE /	ADDR)	
Ø853	BIT1	BGND1	BACKGR	OUND	COLOR B	IT (PAL	ETTE /	ADDR)	
Ø854	BIT2	BGND2	BACKGR	OUND	COLOR B	IT (PAL	ETTE /	ADDR)	
Ø855	BIT3	FGBDØ			COLOR B				
Ø856	BIT4	FGND1			COLOR B				
Ø857	BIT5	FGND2			COLOR B			AUUR)	
Ø858 Ø859	BIT6 BIT7	UNDLN BLINK			ARE UND BLINK A			\TE	
Ø86Ø	* ATTRIBUTES NOT AVA					11 1/2 3	, LC. 10,	\	
Ø861	* GRAPHICS MODES	LLNDLL	MIILIN OO		-				
Ø862	16 COLOR MODES: (CRES	S1=1, C	RESØ = 0	Ø)					
Ø863	BYTE FROM DRAM								
Ø864	BITØ		SECOND P						
Ø865	BIT1		SECOND F						
Ø866	BIT2		SECOND F						
Ø867	BIT3		SECOND F						
Ø868 Ø869	BIT4 BIT5		FIRST PI FIRST PI						
Ø87Ø	BIT6		FIRST PI						
Ø871	BIT7		FIRST PI						
Ø872	4 COLOR MODES: (CREST	-							
Ø873	BYTE FROM DRAM	,							
Ø874	BITØ	PAØ,	FOURTH F	IXEL					
Ø875	BIT1		FOURTH F						
Ø876	BIT2		THIRD PI						
Ø877	BIT3		THIRD PI						
Ø878	BIT4		SECOND F						
Ø879	BIT5	PAI,	SECOND F	'IXEL					
Ø88Ø ø991	BIT6		FIRST PI						
Ø881 Ø882	BIT7 2 COLOR MODES: (CRESI		FIRST PI FSØ = Ø						
Ø883	BYTE FROM DRAM	- ~, UN		,					
Ø884	BITØ	PAØ.	EIGHTH F	IXEL					
Ø885	BIT1		SEVENTH		_				
Ø886	BIT2	PAØ,	SIXTH PI	XEL					
Ø887	BIT3		FIFTH PI						
Ø888	BIT4	PAØ,	FORTH PI	XEL					

Ø889 Ø89Ø Ø891		* DALETTE	ADDDES	BIT5 BIT6 BIT7	PAØ, S	HIRD PI ECOND F IRST PI	PIXEL	
Ø892		* PALETTE	ADDRES		DAG	DAG	D 4 1	DAG
Ø893				ADDRESS	PA3	PA2	PA1	PAØ
Ø894				FFBØ	Ø	Ø	Ø	0
Ø895				FFB1	Ø	Ø	Ø	1
Ø896				FFB2	Ø	Ø	1	Ø
Ø897				FFB3	Ø	Ø	1	1
Ø898				FFB4	Ø	1	Ø	Ø
Ø899				FFB5	Ø	1	Ø	1
0900				FFB6	Ø	1	1	Ø
0901				FFB7	Ø	1	1	1
Ø9Ø2				FFB8	1	Ø	Ø	0
Ø9Ø3				FFB9	1	Ø	Ø	1
Ø9Ø4 Ø9Ø5				FFBA FFBB	1 1	Ø Ø	1	Ø 1
Ø9Ø6				FFBC	1	и 1	1 Ø	1 Ø
Ø9Ø7				FFBD	1	1	Ø	1
Ø9Ø8				FFBE	1	1	1	Ø
Ø9Ø9				FFBF	1	1	1	1
Ø91Ø				1101	-	-	-	1
Ø911	FF9A	V.BORDER	RMB	1	BORDER	REGIST	ΓFR	
Ø912			2	-	50115211			
Ø913	FF9A	BITØ	BLUØ			BLUE L	_SB	
Ø914		BIT1	GRNØ			GREEN		
Ø915		BIT2	REDØ			RED LS	SB	
Ø916		BIT3	BLU1			BLUE N	1SB	
Ø917		BIT4	GRN1			GREEN	MSB	
Ø918		BIT5	RED1			RED MS	SB	
Ø919		BIT6						
Ø92Ø		BIT7						
Ø921								
Ø922			RMB	1	RESERV			
Ø923	FF9C	V.SCROLL	RMB	1	VERTIC	AL SCRO	OLL REGI	ISTER
Ø924								
Ø925	FF90	BITØ	VSCØ					
Ø926		BIT1	VSC1					
Ø927 Ø928		BIT2	VSC2 VSC3					
Ø929		BIT3 BIT4	V363					
Ø93Ø		BIT5						
Ø931		BIT6						
Ø932		BIT7						
Ø933			MODE.	THE VSC'S MUST	BE INIT	IALIZED	TO \$ØF	
Ø934			,					
Ø935	FF9D	V.OFSET1	RMB	1	VERTIC	AL OFFS	SET 1 RE	EGISTER
Ø936								
Ø937	FF9D	BITØ	Y11					
Ø938		BIT1	Y12					
Ø939		BIT2	Y13					
Ø94Ø		BIT3	Y14					
0941		BIT4	Y15					
Ø942		BIT5	Y16					
Ø943		BIT6	Y17					
Ø944		BIT7	Y18					
Ø945	FFOF	V 0505T0	DMD	1	VEDITO		SET & DE	-010750
	FF9E	V.OFSETØ	RMB	1	VERIIC	AL UFFS	SET Ø RE	GISTER
Ø947	FEOF	DITα	٧ɔ					
Ø948	FFYE	BITØ	Y3					
Ø949 Ø95Ø		BIT1 BIT2	Y4 Y5					
й95й Ø951		BIT3	15 Y6					
Ø951		BIT4	Y7					
Ø953		BIT5	Y8					
Ø954		BIT6	Y9					
Ø955		BIT7	Y1Ø					
Ø956				Y9-Y15 ARE NOT	EFFECTI	VE. AND	ARE CO	NTROLLED BY
Ø957								BE 1, ALL OTHERS Ø
Ø958		5210	`		, .			, == :=
	FF9F	H.OFSETØ	RMB	1	HORIZO	NTAL OF	FFSET Ø	REGISTER
Ø96Ø								
Ø961	FF9F	BITØ	ΧØ				FSET AD	
Ø962		BIT1	X1		HORIZO	NIAL OF	FFSET AD	JURESS

Ø963 Ø964 Ø965 Ø966 Ø967 Ø968 Ø970 Ø971 Ø972 Ø973 Ø974		HRES BIT SOMEWHAT 'WINDOW' BITS. I	TS AND LARG (THE IN CHA	A HORIZONTAL SCR CRES BITS SELEC ER THAN THE DISP DISPLAYED SCREE RACTER MODE, THE (OR 64, IF DOUBL	HORIZO HORIZO HORIZO HORIZO HORIZO EEN WID TED. TH LAYED S N) BY M	IS WILL CREEN. EANS OF WIDTH	FFSET AL FFSET AL FFSET AL FFSET AL IRTUAL L 28 BYTI ALLOW THE US THE HO IS 128	DDRESS DDRESS DDRESS DDRESS ENABLE ES REGAL A 'VIR' SER CAN DRIZONTA	TUAL'S MOVE T AL OFFS	CREEN HIS ET
Ø976	FFAØ	MMUREG	RMB	16	MEMORY	MANAGE	MENT U	NIT REG	ISTERS	(6 BITS)
Ø977 Ø978		* RFLATION	ISHTP	BETWEEN DATA IN	TASK RE	GISTER	AND GFI	NFRATFD	ADDRES	S
Ø979				BIT	D5	D4	D3	D2	D1	DØ
Ø98Ø Ø981				CORRESPONDING MEMORY ADDRESS	A18	A17	A16	A15	A14	A13
Ø982										
Ø983 Ø984		* DATA FRO		MMU IS THEN USE	D AS TH	IE UPPER	R 6 ADDI	RESS LII	NES (A1	3-A18)
Ø985		TOK TIETK	ACT AC	ADDRESS RANGE	TR	A15	A14	A13	MMU L	OCATION
Ø986				X0000 - X1FFF	Ø	Ø	Ø	Ø	FFAØ	
Ø987 Ø988				X2000 - X3FFF X4000 - X5FFF	Ø Ø	Ø Ø	Ø 1	1 Ø	FFA1 FFA2	
Ø989				X6000 - X7FFF	Ø	Ø	1	1	FFA3	
Ø99Ø				X8000 - X9FFF	Ø	1	Ø	Ø	FFA4	
Ø991				XAØØØ - XBFFF	Ø	1	Ø	1	FFA5	
Ø992				XCØØØ - XDFFF	Ø	1	1	Ø	FFA6	
Ø993 Ø994				XEØØØ - XFFFF	Ø	1	1	1	FFA7	
Ø995				XØØØØ - X1FFF	1	Ø	Ø	Ø	FFA8	
Ø996				X2000 - X3FFF	1	Ø	Ø	1	FFA9	
Ø997				X4000 - X5FFF	1	Ø	1	Ø	FFAA	
Ø998				X6000 - X7FFF	1	Ø	1	1	FFAB	
0999				X8000 - X9FFF	1	1	Ø	Ø	FFAC	
1000				XAØØØ - XBFFF	1 1	1 1	Ø 1	1 Ø	FFAD FFAE	
1001 1002				XCØØØ - XDFFF XEØØØ - XFFFF	1	1	1	ข 1	FFAF	
1003				XLDDD XIIII	-	-	-	-		
	FFBØ	PALETREG	RMB	16	COLOR	PALETTE	REGIS	TERS (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	ΙØ	Р3	P2	P1	PØ
1009										
1010				ATIBILITY, THE F					N INITI	ALIZATION
1011 1012		(KGB VAL		FOR PAL VERSION, GREEN	\$12	IABLE	FUR CUI	MPUSITE		
1013				YELLOW	\$36					
1014				BLUE	\$Ø9					
1015			FFB3		\$24					
1016 1017				BUFF CYAN	\$3F \$1Ø					
1017				MAGENTA	\$10 \$2D					
1019				ORANGE	\$26					
1020			FFB8	BLACK	\$ØØ					
1021				GREEN	\$12					
1022				BLACK	\$ØØ					
1023 1024				BUFF BLACK	\$3F \$ØØ					
1024				GREEN	\$12					
1026				BLACK	\$00					
1027			FFBF	ORANGE	\$26					
1028	ΓΓCα	CAMBEC	FOU	4	CAM 00	NITRO: T		nc		
1029 1030	FFCØ	SAMREG	EQU	*	SAM CO	NTROL F	KEG151E	κ5		
	FFCØ	VØCLR	RMB	1	CLEAR	COCO GF	RAPHICS	MODE V	Ø	
	FFC1	VØSET	RMB	1		CO GRAF			-	
1033	FFC2	V1CLR	RMB	1	CLEAR	COCO GF	RAPHICS	MODE V	1	
	FFC3	V1SET	RMB	1		CO GRAF			•	
	FFC4	V2CLR	RMB	1				MODE V	2	
סכמד	FFC5	V2SET	RMB	1	SEI UU	CO GRAF	III CO MI	ODE NZ		

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		*	INTER	RUPT 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	

EXTENDED BASIC UNRAVELLED II APPENDIX B ORIGIN:SPECTRAL ASSOC DISASSEMBLY OF EXTENDED BASIC 1.1 REVISED:12/26/1999 WALTER K ZYDHEK

0001 0002			
0002	8000	ORG \$800	
	8000 45 58	EXBAS FCC 'EX'	
0003		* HOVE EVERNOED DAG	ICIC COMMAND INTERDRETATION TARLE FROM DOM TO DAM
0004 0005	8002 8E 80 DE	L8002 LDX #L801	IC'S COMMAND INTERPRETATION TABLE FROM ROM TO RAM E ROM ADDRESS
0006	8005 CE 01 2A		EC+1Ø RAM ADDRESS
0007	8008 C6 0A	LDB #10	10 BYTES TO MOVE
0008	800A BD A5 9A	JSR LA59	
0009	800D 8E B2 77	LDX #LB2	
0010	8010 AF 43	STX \$03,	
0011	8Ø12 AF 48	STX \$08,	
0012	8Ø14 8E 89 4C	LDX #XIR	
0013	8Ø17 BF Ø1 ØD	STX IRQV	
0014	8Ø1A 9E 8A	LDX ZERO	GET X=Ø
0015	801C BF 01 12	STX TIMV	L INITIALIZE TIMER = Ø
0016	8Ø1F BD 82 9C	JSR XVEC	8 INITIALIZE A BUNCH OF VARIABLES
0017	8022 CC 2C 05	LDD #\$2C	5 *INITIALIZE DLOAD TO 1200 BAUD AND
0018	8025 DD E6	STD DLBA	D *TIMEOUT CONSTANT TO 5
0019	8027 8E 01 3E	LDX #USR	
0020	802A 9F B0	STX USRAI	
0021			R CALLS TO FC ERROR
0022	802C CE B4 4A	LDU #LB4	
0023	8Ø2F C6 ØA	LDB #10	10 USR CALLS IN EX BASIC
0024	8Ø31 EF 81	L8031 STU ,X++	STORE FC ERROR AT USR ADDRESSES
0025	8Ø33 5A	DECB	FINISHED ALL 10?
0026	8Ø34 26 FB	BNE L803:	NO NO
0027		+ MODIEW THE 2	OVC FOR THE NEW PONTINES CONTAINED IN EXT CASE
ØØ28	0026 06 75		OKS FOR THE NEW ROUTINES CONTAINED IN EXT BASIC
ØØ29	8036 86 7E	LDA #\$7E STA RVEC	OP CODE OF JMP
ØØ3Ø ØØ31	8038 B7 01 9A		
ØØ31 ØØ32	803B 8E 82 B9 803E BF 01 9B	LDX #XVE	
0032 0033			
	8041 B7 01 8B 8044 8E 88 46	STA RVEC: LDX #XVE	
ØØ34 ØØ35	8044 8E 88 46 8047 BF 01 8C	STX RVEC	
0035 0036	8047 BF 01 8C	STA RVEC	
ØØ37	804D 8E 87 E5	LDX #XVE	
ØØ38	8050 BF 01 98	STX RVEC	
ØØ39	8Ø53 B7 Ø1 79	STA RVEC	
0040	8056 8E 8E 90	LDX #XVE	
0041	8Ø59 BF Ø1 7A	STX RVEC	
0042	805C B7 01 91	STA RVEC	
0043	8Ø5F 8E 88 FØ	LDX #XVE	
0044	8Ø62 BF Ø1 92	STX RVEC	
ØØ45	8Ø65 B7 Ø1 6A	STA RVEC	
0046	8Ø68 8E 8C F1	LDX #XVE	
0047	806B BF 01 6B	STX RVEC	
0048	8Ø6E B7 Ø1 67	STA RVEC:	
0049	8071 8E 82 73	LDX #XVE	3 =
0050	8074 BF 01 68	STX RVEC:	+1 = CONSOLE OUT
0051	8077 B7 Ø1 76	STA RVEC	
0052	807A 8E 82 86	LDX #XVE	*
0053	807D BF 01 77	STX RVEC	+1 * CLOSE A FILE
0054	8080 B7 01 A3	STA RVEC	3 =
0055	8Ø83 8E 83 Ø4	LDX #XVE	23 =
0056	8086 BF 01 A4	STX RVEC	3+1 = CRUNCH A BASIC LINE
0057	8Ø89 B7 Ø1 94	STA RVEC	
0058	8Ø8C 8E 82 9C	LDX #XVE	8+1 * RUN
ØØ58 ØØ59	808F BF 01 95	LDX #XVE STX RVEC	
ØØ58 ØØ59 ØØ6Ø	808F BF 01 95 8092 B7 01 1D	LDX #XVE STX RVEC STA EXPJI	P STORE OP CODE OF JMP
ØØ58 ØØ59 ØØ6Ø ØØ61	808F BF 01 95 8092 B7 01 1D 8095 8E 84 89	LDX #XVE STX RVEC STA EXPJI LDX #L841	P STORE OP CODE OF JMP 19 GET EXPONENTIATION ADDRESS
0058 0059 0060 0061 0062	808F BF 01 95 8092 B7 01 1D 8095 8E 84 89 8098 BF 01 1E	LDX #XVE STX RVEC STA EXPJI LDX #L84! STX EXPJI	P STORE OP CODE OF JMP 19 GET EXPONENTIATION ADDRESS IP+1 SAVE IT
0058 0059 0060 0061 0062 0063	808F BF 01 95 8092 B7 01 1D 8095 8E 84 89 8098 BF 01 1E 809B BD 96 E6	LDX #XVEI STX RVEC: STA EXPJI LDX #L841 STX EXPJI JSR L96E	IP STORE OP CODE OF JMP 9 GET EXPONENTIATION ADDRESS IP+1 SAVE IT 60 INITIALIZE EXBAS GRAPHICS VARIABLES
0058 0059 0060 0061 0062 0063 0064	808F BF 01 95 8092 B7 01 1D 8095 8E 84 89 8098 BF 01 1E 809B BD 96 E6 809E B6 FF 03	LDX #XVEI STX RVEC. STA EXPJI LDX #L84I STX EXPJI JSR L96EI LDA P1A6	IP STORE OP CODE OF JMP 19 GET EXPONENTIATION ADDRESS 1P+1 SAVE IT 1 GO INITIALIZE EXBAS GRAPHICS VARIABLES 23 * ENABLE PIAØ TO
0058 0059 0060 0061 0062 0063 0064 0065	808F BF 01 95 8092 B7 01 1D 8095 8E 84 89 8098 BF 01 1E 8098 BD 96 E6 809E B6 FF 03 8041 8A 01	LDX #XVEI STX RVEC. STA EXPJI LDX #L84! STX EXPJI JSR L966! LDA P1A6!	IP STORE OP CODE OF JMP 19 GET EXPONENTIATION ADDRESS 1P+1 SAVE IT 1 GO INITIALIZE EXBAS GRAPHICS VARIABLES 13 * ENABLE PIAØ TO 1 * PASS 6ØHZ
0058 0059 0060 0061 0062 0063 0064 0065	808F BF 01 95 8092 B7 01 1D 8095 8E 84 89 8098 BF 01 1E 8098 BD 96 E6 809E B6 FF 03 8041 8A 01	LDX #XVEI STX RVEC. STA EXPJI LDX #L84! STX EXPJI JSR L966! LDA P1A6!	IP STORE OP CODE OF JMP 9 GET EXPONENTIATION ADDRESS IP+1 SAVE IT 1 GO INITIALIZE EXBAS GRAPHICS VARIABLES 3 * ENABLE PIAG TO * PASS 6ØHZ 3 * INTERRUPT TO MPU
0058 0059 0060 0061 0062 0063 0064 0065 0066	808F BF 01 95 8092 B7 01 1D 8095 8E 84 89 8098 BF 01 1E 8098 BD 96 E6 809E B6 FF 03 8041 8A 01	LDX #XVE STX RVEC STA EXPJI LDX #L84 STX EXPJI JSR L96E LDA PIAØ- ORA #S0I STA PIAØ- LDX #DK	P STORE OP CODE OF JMP GET EXPONENTIATION ADDRESS P+1 SAVE IT GO INITIALIZE EXBAS GRAPHICS VARIABLES 3 * ENABLE PIAØ TO * PASS 6ØHZ 3 * INTERRUPT TO MPU FIRST TWO BYTES OF DISK ROM
0058 0059 0060 0061 0062 0063 0064 0065 0066 0067	808F BF 01 95 8092 B7 01 1D 8095 8E 84 89 8098 BF 01 1E 8098 BD 96 E6 809E B6 FF 03 8041 8A 01	LDX #XVE STX RVEC STA EXPJI LDX #L84 STX EXPJI JSR L96E LDA PIAØ- ORA #S0I STA PIAØ- LDX #DK	P STORE OP CODE OF JMP GET EXPONENTIATION ADDRESS P+1 SAVE IT GO INITIALIZE EXBAS GRAPHICS VARIABLES 3 * ENABLE PIAØ TO * PASS 6ØHZ 3 * INTERRUPT TO MPU FIRST TWO BYTES OF DISK ROM
0058 0059 0060 0061 0062 0063 0064 0065 0066 0067 0068	808F BF 01 95 8092 87 01 11 8095 BE 84 89 8098 BF 01 1E 8098 BO 96 E6 809E B6 FF 03 80A1 BA 01 80A3 B7 FF 03 80A6 BE C0 00 80AC 10 27 3F 5	LDX #XVE STX RVEC STA EXPJI LDX #L84 STX EXPJI JSR L96E LDA PIAØ- ORA #S0I STA PIAØ- LDX #DK	P STORE OP CODE OF JMP GET EXPONENTIATION ADDRESS P+1 SAVE IT GO INITIALIZE EXBAS GRAPHICS VARIABLES 3 * ENABLE PIAØ TO * PASS 6ØHZ 3 * INTERRUPT TO MPU FIRST TWO BYTES OF DISK ROM
0058 0059 0060 0061 0062 0063 0064 0065 0066 0067 0068 0069	808F BF 01 95 8092 B7 01 15 8095 BE 84 89 8098 BF 01 1E 809B BD 96 E6 809E B6 FF 03 80A1 BA 01 80A3 B7 FF 03 80A6 BE 44 4B 80A9 BC C0 00 80AC 10 27 3F 5	LDX #XVE STX RVEC. STA EXPJI LDX #L84 STX EXPJI JSR L96E LDA PIAØ- ORA #\$0I STA PIAØ- LDX #'DK CMPX DOSB, LBEQ DOSB, ANDCC #SAF	IP STORE OP CODE OF JMP 9 GET EXPONENTIATION ADDRESS P+1 SAVE IT GO INITIALIZE EXBAS GRAPHICS VARIABLES 3 * ENABLE PIAO TO * PASS 60HZ 3 * INTERRUPT TO MPU FIRST TWO BYTES OF DISK ROM COMPARE TO DISK ROM ADDRESS S+2 BRANCH IF DISK BASIC EXISTS ENABLE INTERRUPTS
0058 0059 0060 0061 0062 0063 0064 0065 0066 0067 0068 0069 0070	808F BF 01 95 8092 B7 01 10 8095 BE 84 89 8098 BF 01 1E 8098 BD 96 65 809E B6 FF 03 80A1 BA 01 80A3 BF FF 03 80A6 BE 44 4B 80A9 BC C0 00 80AC 10 27 3F 5 80B0 1C AF	LDX #XVEI STX RVEC. STA EXPJI LDX #L84! STX EXPJI JSR L966! LDA P1A6- ORA #\$01 STA P1A6- LDX #10K CMPX DOSB, LBEQ DOSB, ANDCC #\$AF LDX #L86!	STORE OP CODE OF JMP
0058 0059 0060 0061 0062 0063 0064 0065 0066 0067 0068 0069 0071 0072	808F BF 01 95 8092 B7 01 1 8095 BE 84 89 8098 BF 01 1E 8098 BD 96 E6 809E B6 FF 03 80A1 BA 01 80A3 B7 FF 03 80A6 BE 44 48 80A9 BC C0 00 80AC 10 27 3F 5 80B0 1C AF 80B2 BE 80 E7 80B5 BD B9 9C	LDX #XVE STX RVE STA EXPJI LDX #L84 STX EXPJI JSR L96E LDA P1A6 ORA #\$01 STA P1A6 LDX #10K CMPX DOSB LBEQ DOSB ANDCC #\$AF LDX #L86 JSR STRII	IP STORE OP CODE OF JMP 9 GET EXPONENTIATION ADDRESS IP+1 SAVE IT GO INITIALIZE EXBAS GRAPHICS VARIABLES 3 * ENABLE PIAO TO * PASS 60HZ 3 * INTERRUPT TO MPU FIRST TWO BYTES OF DISK ROM COMPARE TO DISK ROM ADDRESS IS+2 BRANCH IF DISK BASIC EXISTS ENABLE INTERRUPTS 6+1 POINT OS IGN ON MESSAGE OUT DISPLAY IT
0058 0059 0060 0061 0062 0063 0064 0065 0066 0067 0068 0069 0070	808F BF 01 95 8092 B7 01 10 8095 BE 84 89 8098 BF 01 1E 8098 BD 96 65 809E B6 FF 03 80A1 BA 01 80A3 BF FF 03 80A6 BE 44 4B 80A9 BC C0 00 80AC 10 27 3F 5 80B0 1C AF	LDX #XVEL STX RVEC. STA EXPJI LDX #L84 STX EXPJI JSR L96E LDA PIAØ- ORA #\$01 STA PIAØ- LDX #DK CMPX DOSB, LBEQ DOSB, ANDCC #\$AF LDX #L86 JSR STRII LDX #KBMI	IP STORE OP CODE OF JMP GET EXPONENTIATION ADDRESS IP+1 SAVE IT GO INITIALIZE EXBAS GRAPHICS VARIABLES 3 * ENABLE PIAO TO * PASS 60HZ 3 * INTERRUPT TO MPU FIRST TWO BYTES OF DISK ROM COMPARE TO DISK ROM ADDRESS SS+2 BRANCH IF DISK BASIC EXISTS ENABLE INTERRUPTS 160+1 POINT TO SIGN ON MESSAGE DISPLAY IT ST GET EXBAS WARM START (RESET) VECTOR
0058 0059 0060 0061 0062 0063 0064 0065 0066 0067 0068 0069 0070 0071 0072	808F BF 01 95 8092 B7 01 15 8095 BE 84 89 8098 BF 01 1E 8098 BD 96 65 8091 B6 FF 03 80A1 BA 01 80A3 BF FF 03 80A6 BF 44 4B 80A9 BC C0 00 80AC 10 27 3F 5 80B0 1C AF 80B2 BC BO BO FF 80B5 BD B9 9C 80B8 BE 80 C0	LDX #XVEI STX RVEC. STA EXPJI LDX #L84 STX EXPJI JSR L96E LDA P1A6- ORA #\$61 STA P1A6- LDX #OK CMPX DOSB. LBEQ DOSB. ANDCC #\$AF LDX #186 JSR STKII LDX #R86 STX RSTVI	P STORE OP CODE OF JMP GET EXPONENTIATION ADDRESS PP+1 SAVE IT GO INITIALIZE EXBAS GRAPHICS VARIABLES ** ENABLE PIAØ TO ** PASS 6ØHZ ** INTERRUPT TO MPU FIRST TWO BYTES OF DISK ROM COMPARE TO DISK ROM ADDRESS S+2 BRANCH IF DISK BASIC EXISTS ENABLE INTERRUPTS 66+1 POINT TO SIGN ON MESSAGE OUT DISPLAY IT GET EXBAS WARM START (RESET) VECTOR SAVE IT CC SAVE IT
0058 0059 0060 0061 0062 0063 0064 0065 0066 0067 0068 0070 0071 0071 0073	808F BF 01 95 8092 B7 01 10 8095 BE 84 89 8098 BF 01 1E 8099 BD 96 E6 809E B6 FF 03 80A1 BA 01 80A3 B7 FF 03 80A6 BE 44 4B 80A9 BC C0 00 80AC 10 27 F5 80B0 10 A7 80B2 BE 80 E7 80B8 BB 99 C0 80B8 BE 80 C0	LDX #XVEL STX RVEC. STA EXPJI LDX #L84 STX EXPJI JSR L96E LDA PIAØ- ORA #\$0I STA PIAØ- LDX #OK CMPX DOSB, ANDCC #SAF LDX #L80 JSR STRII LDX #L80 STX RSTVI JMP LAØE. * EXBAS WARM START	P STORE OP CODE OF JMP GET EXPONENTIATION ADDRESS PP+1 SAVE IT GO INITIALIZE EXBAS GRAPHICS VARIABLES ** ENABLE PIAØ TO ** PASS 6ØHZ ** INTERRUPT TO MPU FIRST TWO BYTES OF DISK ROM COMPARE TO DISK ROM ADDRESS S+2 BRANCH IF DISK BASIC EXISTS ENABLE INTERRUPTS 66+1 POINT TO SIGN ON MESSAGE OUT DISPLAY IT GET EXBAS WARM START (RESET) VECTOR SAVE IT CC SAVE IT
0058 0059 0060 0061 0062 0063 0064 0066 0067 0068 0069 0070 0071 0072 0073 0074 0075	808F BF 01 95 8092 B7 01 10 8095 BE 84 89 8098 BF 01 1E 8099 BD 96 E6 809E B6 FF 03 80A1 BA 01 80A3 B7 FF 03 80A6 BE 44 4B 80A9 BC C0 00 80AC 10 27 F5 80B0 10 A7 80B2 BE 80 E7 80B8 BB 99 C0 80B8 BE 80 C0	LDX #XVEL STX RVEC. STA EXPJI LDX #L84 STX EXPJI JSR L96E LDA PIAØ- ORA #\$0I STA PIAØ- LDX #OK CMPX DOSB, ANDCC #SAF LDX #L80 JSR STRII LDX #L80 STX RSTVI JMP LAØE. * EXBAS WARM START	P STORE OP CODE OF JMP GET EXPONENTIATION ADDRESS PP+1 SAVE IT GO INITIALIZE EXBAS GRAPHICS VARIABLES ** ENABLE PIAØ TO ** PASS 6ØHZ ** INTERRUPT TO MPU FIRST TWO BYTES OF DISK ROM COMPARE TO DISK ROM ADDRESS S+2 BRANCH IF DISK BASIC EXISTS ENABLE INTERRUPTS 66+1 POINT TO SIGN ON MESSAGE OUT DISPLAY IT GET EXBAS WARM START (RESET) VECTOR SAVE IT CC SAVE IT
0058 0059 0060 0061 0062 0063 0064 0065 0066 0067 0068 0067 0071 0072 0073 0074 0075 0077	808F BF 01 95 8092 B7 01 16 8095 BE 84 89 8098 BF 01 1E 8098 BD 96 65 8092 B6 FF 03 80A1 BA 01 80A3 B7 FF 03 80A1 BA 01 80A6 BE 44 4B 80A9 BC C0 00 80AC 10 27 3F 5 80B0 1C AF 80B2 BE 80 C7 80BB BB B9 9C 80BB BB B9 9C 80BB 9F 72 80BB 7F A8 E2	LDX #XVE STX REC. STA REC. STA REC. STA REC. LDX #L84 STX EXPJI JSR L96E LDA PIAØ- ORA #\$0I STA PIAØ- LDX #'DK CMPX DOSB. LBEQ DOSB. ANDCC #\$AF LDX #L80 JSR STRII LDX #XBWI STX RSTV! JMP LAØE: * EXBAS WARM START XBWMST NOP CLR PLYTI	STORE OP CODE OF JMP
0058 0059 0060 0061 0062 0063 0064 0065 0066 0067 0068 0070 0071 0072 0073 0074 0075 0077	808F BF 01 95 8092 B7 01 1E 8095 BE 84 89 8098 BF 01 1E 8098 BD 96 E6 8092 B6 FF 03 80A1 BA 01 80A3 B7 FF 03 80A1 BA 01 80A6 BC C0 00 80A6 BC C4 4B 80A9 BC C0 00 80B0 1C AF 80B2 BE 80 C7 80B8 BC 97 80B8 BC 97 80B8 BC 97 80B8 BC 80 80B8 97 80B8 P7 80B8 BC 80 80B8 97 80B8 BC 80 80B8 P7 80B8 BC 80 80B8 BF E8	LDX #XVE STX RVEC. STA EXPJI LDX #L84 STX EXPJI JSR L966: LDA PIAØ- ORA #\$01 STA PIAØ- LDX #CMPX DOSB, ANDCC #\$AF LDX #L86 STA LBEQ DOSB, ANDCC #\$AF LDX #L86 STA LBQ LDX #L86 STA LDX #L86 LBC LDX #L86 STA LDX #L86 LDX #	STORE OP CODE OF JMP
0058 0059 0060 0061 0062 0063 0064 0066 0066 0067 0070 0071 0072 0073 0074 0075 0077 0078 0077 0078	808F BF 01 95 8092 B7 01 10 8095 BE 84 89 8098 BF 01 1E 8099 BD 96 E7 8092 BC FF 03 80A1 BA 01 80A3 BF FF 03 80A6 BE 44 4B 80A9 BC C0 00 80BA 10 27 AF 80B2 BE 80 E7 80B8 BE 80 C0 80BB 9F 72 80BB 9F 72 80BB 7F 72 80BB 7F 72 80BB 7E A0 80BB 12 80C0 12 80C1 0F E3 80C3 0F E4 80C5 B6 FF 03	LDX #XVEC STX RVEC STX RVEC STX RVEC STX EXPJI LDX #L844 STX EXPJI JSR L966: LDA P1A6- LDA #501 STA P1A6- LDX #105 CMPX DOSB, ANDCC #5AF LDX #L861 JSR STRII LDX #L861 JSR STRII LDX #1861 STX RSTVI JMP LAGE: * EXBAS WARM STATX XBWMST NOP CLR PLYTI LDA P1A6-	STORE OP CODE OF JMP
0058 0059 0060 0061 0062 0063 0064 0066 0067 0068 0070 0071 0072 0073 0074 0075 0076 0076 0077 0078	808F 8F 01 95 8092 87 01 18 8095 8E 84 89 8098 8F 01 16 8098 8D 96 16 8098 8D 96 6 804 8A 97 8041 8A 91 8048 8C C0 00 8048 8C C7 00 8080 1C AF 8082 8E 80 C0 8088 8E 80 C0 8088 9F 72 8080 12 8080 12 8080 12 8080 9F 22 8080 12 8080 07 E A0 8080 10 E E 8080 10 E 8080 07 E A0 8080 A0 8	LDX #XVE STX REPOIL LDX #L84 STX EXPOIL LDX #L84 STX EXPOIL JSR L96E LDA PIAG- ORA #501 STA PIAG- LDX #DX CMPX DOSB. ANDCC #SAF LDX #S0K STX STRII LDX #XBWI STX STRII LDX #XBWI STX RSTVI JMP LAGE: * EXBAS WARM START XBWMST NOP CLR PLYTI CLR PLYTI LDA PIAG- ORA #501	STORE OP CODE OF JMP
0058 0059 0060 0061 0062 0063 0064 0066 0067 0068 0070 0071 0072 0073 0074 0075 0076 0077 0077 0078 0079 0079 0079	808F BF 01 95 8092 B7 01 1E 8095 BE 84 89 8098 BF 01 1E 8098 BD 96 E6 8092 B6 FF 03 80A1 8A 01 80A3 B7 FF 03 80A1 BA 01 80A6 BC C0 00 80B0 1C AF 80B2 BE 80 E7 80B8 BE 80 C0 80BB 9F 72 80BD 7E A0 E2 80C0 12 80C0 12 80C0 BF E3 80C3 BF E4 80C3 BF E4 80C3 BF E4 80C3 BF F 03 80C8 BA 01	LDX #XVE STX RECL STA EXPJI LDX #L84 STX EXPJI JSR L966 LDA PIA6- ORA #\$01 STA PIA6- LDX #CMPX DOSB, ANDCC #\$AF LDX #L88 STX EXPJI LDX #CMPX DOSB, ANDCC #SAF LDX #L80 STX RSTVI JMP LA0E: * EXBAS WARM START XBWMST NOP CLR PLYTI LDA PIA6- ORA #\$01 STA PIA6-	STORE OP CODE OF JMP
0058 0059 0060 0061 0062 0063 0064 0066 0067 0068 0071 0072 0073 0074 0075 0077 0078 0077 0078 0077 0078 0077 0078	808F 8F 01 95 8092 87 01 18 8095 8E 84 89 8098 8F 01 16 8098 8D 96 16 8098 8D 96 6 804 8A 97 8041 8A 91 8048 8C C0 00 8048 8C C7 00 8080 1C AF 8082 8E 80 C0 8088 8E 80 C0 8088 9F 72 8080 12 8080 12 8080 12 8080 9F 22 8080 12 8080 07 E A0 8080 10 E E 8080 10 E 8080 07 E A0 8080 A0 8	LDX #XVEC STX RVEC STX RVEC STX REPJI LDX #L844 STX EXPJI JSR L966: LDA P1A6- LDA #S01 STA P1A6- LDX #D0SB, ANDCC #SAF LDX #LB0 JSR STRII LDX #LB0 JSR STRII LDX #LB0 STX RSTV JMP LA0E: * EXBAS WARM START XBWMST NOP CLR PLYTI LDA P1A6- ORA #S01 STA P1A6- STA P1A6- JMP BAMM:	STORE OP CODE OF JMP
0058 0059 0060 0061 0062 0063 0064 0066 0067 0069 0070 0071 0072 0073 0074 0075 0076 0077 0078 0078 0078 0078	808F BF 01 95 8092 B7 01 1E 8095 BE 84 89 8098 BF 01 1E 8098 BD 96 E6 8092 B6 FF 03 80A1 8A 01 80A3 B7 FF 03 80A1 BA 01 80A6 BC C0 00 80B0 1C AF 80B2 BE 80 E7 80B8 BE 80 C0 80BB 9F 72 80BD 7E A0 E2 80C0 12 80C0 12 80C0 BF E3 80C3 BF E4 80C3 BF E4 80C3 BF E4 80C3 BF F 03 80C8 BA 01	LDX #XVE STX REFOIL STX REFOIL LDX #L84 STX EXPJI JSR L96E LDA PIAG- ORA #501 STA PIAG- LDX #DXB LDX #DXB LDX #DXB LDX #SST LDX #SST LDX #SST LDX #SST LDX #SST STRII LDX #XBWI STX RSTVI JMP LAGE: CLR PLYTI CLR PLYTI CLR PLYTI LDA PIAG- ORA #501 STA PIAG- JMP BAWM:	STORE OP CODE OF JMP
0058 0059 0060 0061 0062 0063 0066 0066 0067 0068 0070 0071 0072 0073 0074 0077 0077 0077 0077 0078 0079 0079 0080 0081 0082 0083 0083	808F BF 01 95 8092 B7 01 1E 8098 BF 01 1E 8098 BF 01 1E 8098 BF 02 66 8092 B6 FF 03 80A1 8A 01 80A3 BF FF 03 80A6 BE 44 4B 80A9 BC C0 00 80B0 1C AF 80B2 BE 80 C7 80B8 BC 80 C0 80BB 9F 72 80BB 7E A0 E2 80C0 12 80C0 12 80C0 BF E3 80C1 BF E3 80C3 BF E4 80C5 B6 FF 03 80CA B7 FF 03	LDX #XVE STX RECC. STA RECC. LDX #L84 STX EXPJI JSR L966: LDA PIAG- ORA #501 LDX #1DK CMPX DOSB. ANDCC #SAF LDX #L80 JSR STRII LDX #XBWI STX RSTV! JMP LAGE: * EXBAS WARM START XBWMST NOP CLR PLYTI LDA PIAG- ORA #501 STA PIAG- JMP BAWM: * THIS CODE IS A PAG-	STORE OP CODE OF JMP GET EXPONENTIATION ADDRESS PP+1 SAVE IT GO INITIALIZE EXBAS GRAPHICS VARIABLES ** ENABLE PIAØ TO ** PASS 6ØHZ ** INTERRUPT TO MPU FIRST TWO BYTES OF DISK ROM COMPARE TO DISK ROM ADDRESS SS-2 BRANCH IF DISK BASIC EXISTS ENABLE INTERRUPTS FOR ADDRESS ENABLE INTERRUPTS GOT SET WARM START (RESET) VECTOR SAVE IT SET WARM START FLAG, ENTER BASIC ENTRY POINT WARM START ENABLE BR-1 BR-1 BR-1 BR-1 BR-1 BR-1 BR-1 BR-
0058 0059 0060 0061 0063 0064 0066 0067 0068 0070 0071 0072 0073 0074 0075 0076 0077 0078 0077 0078 0077 0078 0077 0078 0078 0088 0088 0088 0088	808F BF 01 95 8092 B7 01 10 8095 BE 84 89 8098 BF 01 1E 8099 BD 96 6E 8092 BG FF 03 80A1 BA 01 80A3 BF FF 03 80A6 BE 44 4B 80A9 BC C0 00 80BC 10 27 3F 5 80B0 1C AF 80B2 BE 80 E7 80B8 BB 80 C0 80BB 9F 72 80BB 9F 72 80BB 7E A0 E2 80C0 12 80C0 12 80C0 12 80C1 BF E3 80C3 BF E4 80C3 BF E7 80CA BF FF 03	LDX #XVEC STX RVEC STX RVEC STX RVEC STX EXPJI LDX #L844 STX EXPJI JSR L966: LDA P1A6- CMPX DOSB, ANDCC #SAF LDX #LB6I JSR STRII LDX #LB8I JSR STRII LDX #LB8I STX RSTVI JMP LA0E: * EXBAS WARM START XBWMST NOP CLR PLYTI CLR PLYT CLR PLYTI CLR PLYTI CLR PLYT	STORE OP CODE OF JMP
0058 0059 0060 0061 0062 0063 0064 0066 0067 0069 0070 0071 0072 0073 0074 0075 0076 0077 0078 0078 0078 0078 0078 0078	808F 8F 01 95 8092 87 01 1E 8098 8F 01 1E 8098 8F 01 1E 8098 8D 96 66 8096 86 FF 03 80A1 8A 01 80A3 8F FF 03 80A6 8E 44 4B 80A9 8C C0 00 80B0 1C AF 80B2 8E 00 7 3F 5 80B0 1C AF 80B2 8E 00 C0 80B8 8E 80 C0 80B8 9F 72 80B0 12 80C0 12 80C0 12 80C0 12 80C0 12 80C0 14 80C3 0F E4 80C3 0F E4 80C3 0F E4 80C5 86 FF 03 80CC 8A 01 80CA 8F FF 03 80CC 8A 01 80CA 8F FF 03 80CD 7E A0 E8	LDX #XVE STX REXPJI LDX #L84 STX EXPJI JSR L96E LDA PIAØ- ORA #80I STA PIAØ- LDX #DXB LDX #DXB LDX #DXB LDX #SBI LDX #ABØI JSR STRII LDX #RBWI STX RSTRII LDX #COMP LD	STORE OP CODE OF JMP
0058 0059 0060 0061 0062 0063 0064 0066 0067 0068 0069 0071 0072 0073 0074 0075 0077 0077 0078 0077 0078 0079 0081 0081 0081 0081 0081 0081 0083 0084 0083	808F BF 01 95 8092 B7 01 1E 8095 BE 84 89 8098 BF 01 1E 8098 BD 96 66 8092 BC FF 03 80A1 BA 01 80A3 BF FF 03 80A6 BE 44 4B 80A9 BC C0 00 80B0 1C AF 80B2 BE 80 C7 80BB D B9 67 80BB D B9 72 80BB PF 72 80BB D B9 72 80BB PF 72 80BB BF 80 C0 80BB PF 72 80BB BF 72 80BB BF 83 80BB BF 74 80BB BF 75 80BB BF 76 80BB BF 77 80BB BF 83 80BB BF 78 80BB BF 78 80BB BF 83 80BB BF 84 80BB BF 85 80BB 85 80BB 85 80BB 85 80B 85	LDX #XVEC STX RVEC STX RVEC STX RVEC STX EXPJI LDX #184 STX EXPJI JSR L966: LDA P1A6- CRA #\$01 STA P1A6- LDX #10K CMPX DOSB. ANDCC #5AF LDX #186 JSR STRII LDX #186 JSR STRII LDX #20K STX RSTV: JMP LAGE: * EXBAS WARM STAT XBWMST NOP CLR PLYTI CLR PLYTI CLR PLYTI LDA P1A6- ORA #501 STA P1A6- JMP BAWM: * * THIS CODE IS A P/ L80DØ LDA CURL INCA BEQ L80DI	STORE OP CODE OF JMP GET EXPONENTIATION ADDRESS PP-1 SAVE IT GO INITIALIZE EXBAS GRAPHICS VARIABLES ** ENABLE PIAØ TO ** PASS 6ØHZ ** INTERRUPT TO MPU FIRST TWO BYTES OF DISK ROM COMPARE TO DISK ROM ADDRESS SS-2 BRANCH IF DISK BASIC EXISTS ENABLE INTERRUPTS FOR THE POINT TO SIGN ON MESSAGE OUT DISPLAY IT GET EXBAS WARM START (RESET) VECTOR SAVE IT SET WARM START FLAG, ENTER BASIC ENTRY POINT WARM START ENABLE RETIER BIR = CLEAR PLAY TIMER ** ENABLE PIAØ TO ** PASS 6ØHZ ** INTERRUPT TO MPU JUMP TO BASIC S WARM START TCH TO FIX THE PCLEAR BUG N GET THE CURRENT LINE NUMBER TEST FOR DIRECT MODE RETURN IF DIRECT MODE
0058 0059 0060 0061 0063 0064 0066 0067 0068 0070 0071 0072 0073 0074 0077 0078 0077 0078 0077 0078 0077 0078 0077 0078 0080 0081 0082 0083 0084	808F 8F 01 95 8092 87 01 18 8095 8E 84 89 8098 8F 01 1E 8098 8D 96 E6 809E 86 FF 03 80A1 8A 01 80A3 8F FF 03 80A6 8E 44 48 80A9 8C C0 00 80B 1C AF 80B2 8E 80 C0 80B8 9F 72 80BB 9F 72 80BC 12 80C0 12 80C0 12 80C0 12 80C0 8F E3 80C3 8F E4 80C0 8F E3 80C3 8F F7 03 80C3 8A 01 80CA 8F FF 03 80CA 87 FF 03	LDX #XVEC STX RVEC STX RVEC STA EXPJI LDX #L844 STX EXPJI JSR L966: LDA P1A6- ORA #\$01 STA P1A6- LDX #104 CMPX DOSB, ANDCC #5AF LDX #L861 JSR STRII LDX #L881 JSR STRII LDX #L881 STX RSTV JMP LA0E: * EXBAS WARM START XBWMST NOP CLR PLYTIC CLR	PP STORE OP CODE OF JMP GET EXPONENTIATION ADDRESS PP+1 SAVE IT GO INITIALIZE EXBAS GRAPHICS VARIABLES ** EMABLE PIAM TO ** PASS 60H Z ** INTERRUPT TO MPU FIRST TWO BYTES OF DISK ROM COMPARE TO DISK ROM ADDRESS S+2 BRANCH IF DISK BASIC EXISTS EMBLE INTERRUPTS FOR THE POINT TO SIGN ON MESSAGE FOR THE STATE OF THE BASIC ENTRY POINT WARM START ENABLE RE R
0058 0059 0060 0061 0062 0063 0064 0066 0067 0069 0070 0071 0072 0073 0074 0075 0076 0077 0088 0088 0088 0088 0088 0088	808F 8F 01 95 8092 87 01 1E 8098 8F 01 1E 8098 8F 01 1E 8098 8D 96 6F 76 03 80A1 8A 01 80A3 8F 7F 03 80A6 8E 44 4B 80A9 8C C0 00 80AC 10 27 3F 5 80B0 1C AF 80B2 8E 80 C0 80B8 8E 80 C0 80B8 9F 72 80B0 7E A0 E2 80C0 12 80C0 12 80C0 12 80C0 12 80C0 12 80C0 8F F 03 80C1 0F E3 80C3 0F E4 80C5 86 FF 03 80C0 7E A0 E8 80D0 96 68 80D0 4C 80D3 27 08 80D0 17 20 800D 17 20	LDX #XVE STX REFORM STX REFORM LDX #L84 STX EXPJI JSR L96E LDA P1A6 ORA #\$01 STA P1A6 LDX #DX LDX #DX LDX #DX LDX #DX LDX #S01 LDX #S01 LDX #RS01 STA P1A6 LDX #RS01 STA P1A6 LDX #RS01 STA STRII LDX #RS01 STX RSTRII LDX #RS01 STX RS01 STX RS	STORE OP CODE OF JMP GET EXPONENTIATION ADDRESS PP+1 SAVE IT GO INITIALIZE EXBAS GRAPHICS VARIABLES ** ENABLE PIAØ TO ** PASS 6ØHZ ** INTERRUPT TO MPU FIRST TWO BYTES OF DISK ROM COMPARE TO DISK ROM ADDRESS SS+2 BRANCH IF DISK BASIC EXISTS ENABLE INTERRUPTS 66+1 POINT TO SIGN ON MESSAGE DISPLAY IT GET EXBAS WARM START (RESET) VECTOR SAVE IT SET WARM START FLAG, ENTER BASIC ENTRY POINT WARM START ENABLE ### CLEAR PLAY TIMER ** ENABLE PIAØ TO ** PASS 6ØHZ ** INTERRUPT TO MPU JUMP TO BASIC S WARM START TCH TO FIX THE PCLEAR BUG N GET THE CURRENT LINE NUMBER TEST FOR DIRECT MODE SAVE OFFSET IN ACCD BUSTRACT OUT START OF BASIC
0058 0059 0060 0061 0063 0064 0066 0067 0068 0069 0070 0071 0072 0073 0074 0075 0077 0078 0077 0078 0079 0081 0081 0081 0082 0083 0084 0086 0086 0087	808F BF 01 95 8092 B7 01 1E 8095 BE 84 89 8098 BF 01 1E 8098 BD 96 66 8092 BG FF 03 80A1 8A 01 80A3 BF FF 03 80A6 BE 44 4B 80A9 BC C0 00 80B0 1C AF 80B2 BE 80 FF 80B8 BE 80 C0 80BB 9F 72 80BD 7E A0 E2 80C0 12 80C0 12 80C0 12 80C0 BF E4 80C3 BF E4 80C5 BF F 03 80C3 BF E4 80C3 BF E7 80C3 BF E4 80C3 BF E7 80C3	LDX #XVE STX REFORM STX REFORM LDX #L84 STX EXPJI JSR L96E LDA P1A6 ORA #\$01 STA P1A6 LDX #DX LDX #DX LDX #DX LDX #DX LDX #S01 LDX #S01 LDX #RS01 STA P1A6 LDX #RS01 STA P1A6 LDX #RS01 STA STRII LDX #RS01 STX RSTRII LDX #RS01 STX RS01 STX RS	STORE OP CODE OF JMP
0058 0059 0060 0061 0063 0064 0066 0067 0068 0071 0072 0073 0074 0075 0076 0077 0078 0076 0077 0078 0076 0077 0078 0082 0082 0082 0082 0082 0088 0088	808F 8F 01 95 8092 87 01 18 8095 8E 84 89 8098 8F 01 1E 8098 8D 96 E6 8092 86 FF 03 80A1 8A 01 80A3 8F FF 03 80A6 8E 44 4B 80A9 8C C0 00 80B 9E CC 00 80B 9E CC 00 80B 9E CC 00 80B 9F 72 80BC 0F E3 80C 0F E4 80C 0F E4 80C 0F E3 80C 0F E4	LDX #XVE STX REFORM STX REFORM LDX #L84 STX EXPJI JSR L96E LDA P1A6 ORA #\$01 STA P1A6 LDX #DX LDX #DX LDX #DX LDX #DX LDX #S01 LDX #S01 LDX #RS01 STA P1A6 LDX #RS01 STA P1A6 LDX #RS01 STA STRII LDX #RS01 STX RSTRII LDX #RS01 STX RS01 STX RS	STORE OP CODE OF JMP
0058 0059 0060 0061 0062 0063 0064 0066 0067 0069 0070 0071 0072 0073 0074 0075 0076 0077 0078 0078 0078 0082 0083 0084 0085 0088 0088 0088 0088 0088 0088	808F 8F 01 95 8092 87 01 18 8095 8E 84 89 8098 8F 01 1E 8098 8D 96 E6 8092 86 FF 03 80A1 8A 01 80A3 8F FF 03 80A6 8E 44 4B 80A9 8C C0 00 80B 9E CC 00 80B 9E CC 00 80B 9F 72 80B0 1C AF 80B2 8E 80 C0 80B8 9F 72 80B0 7E A0 80B0 8F FF 03 80C0 7E A0 E8	LDX #XVEC STX RVEC STX RVEC STX RVEC STX RVEC STX EXPJI LDX #184 STX EXPJI JSR L966: LDA P1A6: LDA #104 CMPX DOSB. ANDCC #5AF LDX #186 JSR STRII LDX #186 JSR STRII LDX #186 JSR STRII LDX #26 CMPX STRII LDX #26 ST	STORE OP CODE OF JMP
0058 0059 0060 0061 0063 0064 0066 0067 0077 0072 0073 0077 0078 0077 0078 0077 0078 0079 0081 0081 0082 0083 0084 0084 0088 0088 0088 0088 0088	808 P	LDX #X94E STX RVEC STA EXPJI LDX #L844 STX EXPJI LDX #L946 LDA P1A6- LDA #501 STA P1A6- LDX #104 ORA #\$01 STA P1A6- LDX #104 LDX	STORE OP CODE OF JMP GET EXPONENTIATION ADDRESS SAVE IT GO INITIALIZE EXBAS GRAPHICS VARIABLES ** ENABLE PIAØ TO ** PASS 6ØMZ ** INTERRUPT TO MPU FIRST TWO BYTES OF DISK ROM COMPARE TO DISK ROM ADDRESS SS+2 BRANCH IF DISK BASIC EXISTS ENABLE INTERRUPTS FOULT BYTES OF DISK ROM ADDRESS STATE EXBAS WARM START (RESET) VECTOR CC SAVE IT SET WARM START FLAG, ENTER BASIC ENTRY POINT WARM START ENABLE BERTY POINT WARM START ENABLE BERTY SET WARM START FLAG, ENTER BASIC STATE ENABLE PIAØ TO ** PASS 6ØHZ ** INTERRUPT ** ENABLE PIAØ TO ** PASS 6ØHZ ** INTERRUPT TO MPU JUMP TO BASIC S WARM START TCH TO FIX THE PCLEAR BUG N GET THE CURRENT LINE NUMBER TEST FOR DIRECT MODE SAVE OFFSET IN ACCO SUBTRACT OUT START OF BASIC ADD THE CURRENT BASIC INPUT POINTER SAVE NEW BASIC INPUT POINTER
0058 0059 0060 0061 0062 0063 0064 0066 0067 0069 0070 0071 0072 0073 0074 0075 0076 0077 0078 0078 0078 0082 0083 0084 0085 0088 0088 0088 0088 0088 0088	808F 8F 01 95 8092 87 01 10 8095 8E 84 89 8098 8F 01 16 8098 8D 96 16 8098 8D 96 16 8040 8E 44 4B 8040 8C C0 00 8040 1C AF 86 8080 8E 80 C0 8088 9F 72 8080 0F EA 86 8000 1C AF 83 8000 0F 84 8000 0F 85 8000 0F	LDX #XVE STX REXCI STA EXPJI LDX #L84I STX EXPJI JSR L96EI LDA P1A6- ORA #\$01 STA P1A6- LDX #DSB. LBEQ DOSB. ANDCC #5AF LDX #L86I JSR STRII LDX #L86I JSR STRII LDX #RSWI STA P1A6- LDX #RSWI STA P1A6- LDX #RSWI STX RSTVI JWP LA0E- * EXBAS WARM START XBWMST NOP CLR PLYTI LDA P1A6- ORA #\$01 STA P1A6- JMP BAWM: * THIS CODE IS A P1A6- JMP L80D0 LDA CURL INCA BEQ L80D1 FFR Y,D SUBD TXTT, ADDD CHAR. STD CHAR. STD CHAR. L80DD RTS	STORE OP CODE OF JMP GET EXPONENTIATION ADDRESS PP-1 SAVE IT GO INITIALIZE EXBAS GRAPHICS VARIABLES ** EMABLE PIAD TO ** PASS 60HZ ** INTERRUPT TO MPU FIRST TWO BYTES OF DISK ROM COMPARE TO DISK ROM ADDRESS S+2 BRANCH IF DISK BASIC EXISTS ENABLE INTERRUPTS 6+1 POINT TO SIGN ON MESSAGE DISPLAY IT GET EXBAS WARM START (RESET) VECTOR 15T GET EXBAS WARM START (RESET) VECTOR 15C SAVE IT SET WARM START FLAG, ENTER BASIC ENTRY POINT WARM START ENABLE 16R = CLEAR PLAY TIMER 3 * ENABLE PIAD TO ** PASS 60HZ ** INTERRUPT TO MPU JUMP TO BASIC S WARM START TCH TO FIX THE PCLEAR BUG N GET THE CURRENT LINE NUMBER TEST FOR DIRECT MODE SAVE OFFSET IN ACCD SUBTRACT OUT START OF BASIC DD ADD THE CURRENT BASIC INPUT POINTER 25 EXBAS COMMANDS

EXTENDED BASIC UNRAVELLED II APPENDIX B ORIGIN:SPECTRAL ASSOC DISASSEMBLY OF EXTENDED BASIC 1.1 REVISED:12/26/1999 WALTER K ZYDHEK

0097				
	8ØE1 81 3C	L8ØE1 FDB	L813C	EXBAS RESERVED WORD HANDLER
0098	8ØE3 ØE	L8ØE1 FDB L8ØE3 FCB L8ØE4 FDB	14	14 EXBAS SECONDARY COMMANDS
0099 0100	8ØE4 82 1E 8ØE6 81 68	L8ØE6 FDB		EXBAS SECONDARY RESERVED WORD TABLE EXBAS SECONDARY RESERVED WORD HANDLER
0101				
0102	8ØE8 45 58 54 45 4E 44		'EXTENDED COLOR BASIC 1.1'	
	8ØEE 45 44 2Ø 43 4F 4C 8ØF4 4F 52 2Ø 42 41 53			
	80FA 49 43 20 31 2E 31			
	8100 ØD	L8100 FCB	CR	
	8101 43 4F 50 59 52 49		'COPYRIGHT (C) 1982'	
	8107 47 48 54 20 28 43			
Ø1Ø9 Ø11Ø	810D 29 20 31 39 38 32 8113 20 42 59 20 54 41		' BY TANDY'	
Ø111	8119 4E 44 59		DI TANDI	
Ø112	811C ØD	L811C FCB		
			'UNDER LICENSE FROM MICROSOFT'	
	8123 4C 49 43 45 4E 53 8129 45 2Ø 46 52 4F 4D			
	812F 2Ø 4D 49 43 52 4F			
Ø117				
	8139 ØD ØD ØØ	L8139 FCB	CR,CR,Ø	
Ø119 Ø12Ø		* EADVE CUMMVI	ND INTERPRETATION LOOP	
Ø120	813C 81 CB	L813C CMPA	#\$CB	\$CB IS LARGEST EX BASIC COMMAND TOKEN
Ø122	813E 22 Ø8	BHI	L8148	BRANCH IF > LARGEST TOKEN
Ø123	8140 8E 81 FØ	LDX	#L81FØ	POINT X TO EXBAS DISPATCH TABLE FOR COMMANDS
0124	8143 8Ø B5	SUBA	#\$B5	\$B5 IS SMALLEST EXBAS TOKEN
Ø125	8148 81 FF	I 8148 CMPA	#\$FF	CHECK FOR SECONDARY TOKEN
Ø127	814A 27 Ø8	BEQ	L8154	BRANCH IF IT IS SECONDARY
Ø128	814C 81 CD	CMPA	#\$CD	LARGEST EXBAS TOKEN
Ø129	814E 23 15	BLS	L8165	SYNTAX ERROR FOR USING & FN
0131 0130	0100 DE 9F 01 3/ 8154 9D 9F	JMP 18154 .1SP	GFTNCH	GET AN INPILE CHARACTER FROM RASIC
Ø132	8156 81 90	CMPA	#\$90	*CB IS LARGEST EX BASIC COMMAND TOKEN BRANCH IF > LARGEST TOKEN POINT X TO EXBAS DISPATCH TABLE FOR COMMANDS \$85 IS SMALLEST EXBAS TOKEN INTERRET BASIC TOKEN HANDLER CHECK FOR SECONDARY TOKEN BRANCH IF IT IS SECONDARY LARGEST EXBAS TOKEN SYNTAX ERROR FOR USING & FN GO TO DISK BASIC RESERVED WORD HANDLER GET AN IMPUT CHARACTER FROM BASIC TOKEN FOR MID\$ BRANCH IF MID\$ TOKEN FOR TIMER BRANCH IF TIMER HOOK INTO RAM SYNTAX ERROR
Ø133	8158 10 27 05 7A	LBEQ	L86D6	BRANCH IF MID\$
Ø134	815C 81 9F	CMPA	#\$9F	TOKEN FOR TIMER
Ø135 Ø136	815E 10 2/ 0/ FE 8162 RD 01 A0	.1SP	L8960 RVFC22	BRANCH IF TIMER
Ø137	8165 7E B2 77	L8165 JMP	LB277	SYNTAX ERROR
Ø138				
Ø139	8168 C1 42	* EXBAS SECUNI	JARY CUMMAND HANDLER #2*33	*RA+33 IS LARGEST EXRAS SECONDARY COMMAND
Ø141	8168 C1 42 816A 23 Ø4	BLS	L817Ø	BRANCH IF LEGITIMATE EXBAS SECONDARY TOKEN
	816C 6E 9F Ø1 3C	JMP	[COMVEC+28]	GO TO DISK BASIC SECONDARY COMMAND HANDLER
Ø143	8170 CØ 28	L817Ø SUBB	#2*20	SUBTRACT OUT 20 BASIC SECONDARY COMMANDS
Ø144 Ø145	8174 22 Ø7	CMPB RHT	#2^8 I 817D	RRANCH IF > HFX\$
Ø146	8176 34 Ø4	PSHS	В	SAVE TOKEN OFFSET
0147	8178 BD B2 62	JSR	LB262	EVALUATE EXPRESSION IN PARENTHESES
Ø148 Ø149	8170 C0 28 8172 C1 10 8174 22 07 8176 34 04 8178 BD B2 62 817B 35 04 817D 8E 82 57 8180 7 E B2 CE	PULS	B	GET TOKEN OFFSET BACK
Ø149 Ø15Ø	818Ø 7E B2 CE	JMP	LB2CE	*80+33 IS LARGEST EXBAS SECONDARY COMMAND BRANCH IF LEGITIMATE EXBAS SECONDARY TOKEN GO TO DISK BASIC SECONDARY COMMAND HANDLER SUBTRACT OUT 20 BASIC SECONDARY COMMANDS HEX\$ TOKEN BRANCH IF > HEX\$ SAVE TOKEN OFFSET EVALUATE EXPRESSION IN PARENTHESES GET TOKEN OFFSET BACK EXBAS SECONDARY COMMAND JUMP TABLE JUMP TO SECONDARY FUNCTION HANDLER
Ø151				
Ø152			RD TABLE FOR EXTENDED BASIC	
		* RESERVED WOI	ND TABLE TOK EXTENDED DASIC	TOVEN #
Ø153		* KEZEKAED MOI		TOKEN # B5
Ø153 Ø154	8183 44 45 CC	* KESERVED WOI * L8183 FCC	'DE'.\$8Ø+'L'	TOKEN # B5 B6
Ø153 Ø154	8183 44 45 CC	* L8183 FCC L8186 FCC	'DE',\$80+'L' 'EDI',\$80+'T'	B5 B6 B7
Ø153 Ø154 Ø155 Ø156 Ø157	8183 44 45 CC 8186 45 44 49 D4 818A 54 52 4F CE 818E 54 52 4F 46 C6	* L8183 FCC L8186 FCC L818A FCC L818E FCC	'DE',\$80+'L' 'EDI',\$80+'T'	B5 B6 B7 B8
Ø153 Ø154 Ø155 Ø156 Ø157	8183 44 45 CC 8186 45 44 49 D4 818A 54 52 4F CE 818E 54 52 4F 46 C6	* L8183 FCC L8186 FCC L818A FCC L818E FCC	'DE',\$80+'L' 'EDI',\$80+'T'	B5 B6 B7
Ø153 Ø154 Ø155 Ø156 Ø157	8183 44 45 CC 8186 45 44 49 D4 818A 54 52 4F CE 818E 54 52 4F 46 C6	* L8183 FCC L8186 FCC L818A FCC L818E FCC	'DE',\$80+'L' 'EDI',\$80+'T' 'TRO',\$80+'N' 'TROF',\$80+'F' 'DE',\$80+'F' 'LE',\$80+'T' 'LIN',\$80+'E'	B5 B6 B7 B8 B9
0153 0154 0155 0156 0157 0158 0159 0160 0161	8183 44 45 CC 8186 45 44 49 D4 818A 54 52 4F CE 818E 54 52 4F 46 C6 8193 44 45 C6 8196 4C 45 D4 8199 4C 49 4E C5 819D 50 43 4C D3	* L8183 FCC L8186 FCC L818A FCC L8193 FCC L8196 FCC L8199 FCC L8190 FCC L8190 FCC	'DE',\$80+'L' 'EDI',\$80+'T' 'TRO',\$80+'N' 'TROF',\$80+'F' 'DE',\$80+'F' 'LE',\$80+'T' 'LIN',\$80+'E' 'PCL',\$80+'S'	B5 B6 B7 B8 B9 BA BB BC
0153 0154 0155 0156 0157 0158 0159 0160 0161 0162	8183 44 45 CC 8186 45 44 49 D4 818A 54 52 4F CE 818E 54 52 4F 46 C6 8193 44 45 C6 8196 4C 45 D4 8199 4C 49 4E C5 819D 50 43 4C D3 81A1 50 53 45 D4	* RESERVED WOIL * L8183 FCC L8186 FCC L818A FCC L8193 FCC L8196 FCC L8199 FCC L8190 FCC L8191 FCC L8191 FCC	'DE',\$80+'L' 'EDI',\$80+'T' 'TRO',\$80+'N' 'TROF',\$80+'F' 'DE',\$80+'F' 'LE',\$80+'T' 'LIN',\$80+'E' 'PCL',\$80+'S'	B5 B6 B7 B8 B9 BA BB BC
0153 0154 0155 0156 0157 0158 0159 0160 0161	8183 44 45 CC 8186 45 44 49 D4 818A 54 52 4F CE 818E 54 52 4F 46 C6 8193 44 45 C6 8196 4C 45 D4 8199 4C 49 4E C5 819D 50 43 4C D3	* RESERVED WOIL * L8183 FCC L8186 FCC L8188 FCC L8193 FCC L8199 FCC L8190 FCC L8141 FCC L8145 FCC	'DE', \$80+'L' 'EDI', \$80+'T' 'TRO', \$80+'N' 'TROF', \$80+'F' 'DE', \$80+'F' 'LE', \$80+'T' 'LIN', \$80+'E' 'PCL', \$80+'T' 'PRE'SE', \$80+'T' 'PRESE', \$80+'T'	B5 B6 B7 B8 B9 BA BB BC
0153 0154 0155 0156 0157 0158 0159 0160 0161 0162 0163 0164 0165	8183 44 45 CC 8186 45 44 49 D4 818A 54 52 4F CE 818E 54 52 4F 46 C6 8193 44 45 C6 8199 4C 45 D4 8199 4C 49 4E C5 819D 50 43 4C D3 81A1 50 53 45 D4 81A5 50 52 45 53 45 D4 81A8 53 43 52 45 45 C4 81BB 53 43 52 45 45 C4	** L8188 FCC L818A FCC L818A FCC L819A FCC L8193 FCC L8199 FCC L819D FCC L814A FCC L814A FCC L814B FCC L814B FCC L814B FCC L814B FCC L818B FCC	'DE',\$80+'L' 'EDI',\$80+'T' 'TRO',\$80+'N' 'TROF',\$80+'F' 'DE',\$80+'F' 'LE',\$80+'E' 'PCL',\$80+'E' 'PSE',\$80+'T' 'PRESE',\$80+'T' 'SCREE',\$80+'T' 'PCLEA',\$80+'N'	B5 B6 B7 B8 B9 BA BB BC BD BE BF
0153 0154 0155 0156 0157 0158 0159 0160 0161 0162 0163 0164 0165 0166	8183 44 45 CC 8186 45 44 49 D4 818A 54 52 4F CE 818E 54 52 4F 46 C6 8193 44 45 C6 8199 4C 45 D4 8199 4C 49 4E C5 819D 50 43 4C D3 81A1 50 53 45 D4 81A5 50 52 45 53 45 D4 81A8 53 43 52 45 45 CE 81B1 50 43 4C Q5 41 D2 81B7 43 4F 4C 45 D2	* RESERVED WOIN * L8183 FCC L8186 FCC L8188 FCC L8193 FCC L8199 FCC L8199 FCC L8199 FCC L814A1 FCC L81A5 FCC L81A5 FCC L81A5 FCC L81B6 FCC L81B7 FCC L81B7 FCC	'DE', \$88+'L' 'EDI', \$88+'T' 'TRO', \$80+'N' 'TROF', \$80+'F' 'DE', \$80+'F' 'LE', \$88+'T' 'LIN', \$80+'E' 'PCL', \$80+'T' 'PRESE', \$80+'T' 'SCREE', \$80+'N' 'PCLEA', \$80+'R' 'COLO', \$80+'R'	B5 B6 B7 B8 B9 BA BB BC BD BE BF CØ
0153 0154 0155 0156 0157 0158 0169 0161 0162 0163 0164 0165 0166 0167	8183 44 45 CC 8186 45 44 49 D4 818A 54 52 4F CE 818E 54 52 4F 46 C6 8193 44 45 C6 8196 4C 45 D4 8199 4C 49 4E C5 819D 50 43 4C D3 81A1 50 53 45 D4 81A5 50 52 45 53 45 D4 81AB 53 43 52 45 45 CE 81B1 50 43 4C 4F D2 81BC 43 49 52 43 4C C5	* RESERVED WOIL * L8183 FCC L8186 FCC L8188 FCC L8193 FCC L8199 FCC L8190 FCC L8191 FCC L8141 FCC L8148 FCC L8148 FCC L8148 FCC L8148 FCC L8181 FCC L8181 FCC L8185 FCC L8185 FCC	'DE',\$80+'L' 'EDI',\$80+'T' 'TRO',\$80+'N' 'TROF',\$80+'F' 'DE',\$80+'F' 'LE',\$80+'T' 'LIN',\$80+'E' 'PCL',\$80+'T' 'PRESE',\$80+'T' 'SCREE',\$80+'N' 'PCLEA',\$80+'R' 'COLO',\$80+'R' 'COLO',\$80+'R'	B5 B6 B7 B8 B9 BA BB BC BD BE BF CØ C1 C2
0153 0154 0155 0156 0157 0158 0159 0160 0161 0162 0163 0164 0165 0166 0167	8183 44 45 CC 8186 45 44 49 D4 818A 54 52 4F CE 818E 54 52 4F 46 C6 8193 44 45 C6 8199 4C 45 D4 8199 4C 49 4E C5 819D 50 43 4C D3 81A1 50 53 45 D4 81A5 50 52 45 53 45 D4 81A5 50 52 45 53 45 D4 81A5 50 43 4C 45 41 D2 81B7 43 4F 4C 4F D2 81B7 43 4F 4C 4F D2 81B6 43 49 52 43 4C C5 81C2 50 41 49 4E D4	** L8183 FCC L8186 FCC L818A FCC L8193 FCC L8193 FCC L8199 FCC L819D FCC L814A FCC L814A FCC L814B FCC L814B FCC L814B FCC L814B FCC L814B FCC L818B FCC L818B FCC L818B FCC L818C FCC L816C FCC L816C FCC	'DE',\$80+'L' 'EDI',\$80+'T' 'TRO',\$80+'N' 'TROF',\$80+'F' 'DE',\$80+'F' 'LE',\$80+'T' 'LIN',\$80+'E' 'PCL',\$80+'T' 'PRESE',\$80+'T' 'PRESE',\$80+'T' 'PRESE',\$80+'R' 'COLO',\$80+'R' 'COLO',\$80+'R' 'CIRCL',\$80+'E' 'PAIN',\$80+'T'	B5 B6 B7 B8 B9 BA BB BC BD BE C0 C1 C2 C3
0153 0154 0155 0156 0157 0158 0169 0161 0162 0163 0164 0165 0166 0167	8183 44 45 CC 8186 45 44 49 D4 818A 54 52 4F CE 818E 54 52 4F 46 C6 8193 44 45 C6 8196 4C 45 D4 8199 4C 49 4E C5 819D 50 43 4C D3 81A1 50 53 45 D4 81A5 50 52 45 53 45 D4 81AB 53 43 52 45 45 CE 81B1 50 43 4C 4F D2 81BC 43 49 52 43 4C C5	* RESERVED WOIL * L8183 FCC L8186 FCC L8188 FCC L8193 FCC L8199 FCC L8190 FCC L8191 FCC L8141 FCC L8148 FCC L8148 FCC L8148 FCC L8148 FCC L8181 FCC L8181 FCC L8185 FCC L8185 FCC	'DE',\$80+'L' 'EDI',\$80+'T' 'TRO',\$80+'N' 'TROF',\$80+'F' 'DE',\$80+'F' 'LE',\$80+'T' 'LI',\$80+'E' 'PCL',\$80+'T' 'SCREE',\$80+'T' 'SCREE',\$80+'T' 'SCREE',\$80+'R' 'CCLO',\$80+'R' 'CCLO',\$80+'E' 'PAIN',\$80+'E' 'PAIN',\$80+'T'	B5 B6 B7 B8 B9 BA BB BC BD BE BF CØ C1 C2
0153 0154 0155 0156 0157 0158 0160 0161 0162 0163 0164 0165 0166 0167 0168 0169 0170	8183 44 45 CC 8186 45 44 49 D4 818A 54 52 4F CE 818E 54 52 4F 46 C6 8193 44 45 C6 8199 4C 45 D4 8199 4C 49 4E C5 819D 50 43 4C D3 81A1 50 53 45 D4 81A5 50 52 45 53 45 D4 81A5 50 43 4C 45 41 D2 81B7 43 4F 4C 4F D4 81C7 47 45 D4 81C7 47 45 D4 81CA 50 55 D4 81CA 50 55 D4	** L8188 FCC L818A FCC L818A FCC L819A FCC L8193 FCC L8199 FCC L819D FCC L814A FCC L814A FCC L814B FCC L814B FCC L814B FCC L814B FCC L814B FCC L816C FCC	'DE',\$80+'L' 'EDI',\$80+'T' 'TRO',\$80+'N' 'TROF',\$80+'F' 'DE',\$80+'F' 'LE',\$80+'E' 'PCL',\$80+'S' 'PSE',\$80+'T' 'PRESE',\$80+'T' 'PRESE',\$80+'T' 'PCLEA',\$80+'R' 'COLO',\$80+'R' 'COLO',\$80+'R' 'CIRCL',\$80+'R' 'CIRCL',\$80+'T' 'GE',\$80+'T' 'GE',\$80+'T' 'DE',\$80+'T' 'DE',\$80+'T' 'DE',\$80+'T'	B5 B6 B7 B8 B9 BA BB BC BD BE C0 C1 C2 C3 C4 C5 C6
0153 0154 0155 0156 0157 0158 0159 0160 0161 0162 0163 0164 0165 0166 0167 0168 0171 0171	8183 44 45 CC 8186 45 44 49 D4 818A 54 52 4F CE 818E 54 52 4F 46 C6 8193 44 45 C6 8199 4C 45 D4 8199 4C 49 4E C5 819D 50 43 4C D3 81A1 50 53 45 D4 81A5 50 52 45 53 45 D4 81A5 50 43 4C 45 D2 81B1 50 43 4C 45 D2 81B1 50 43 4C 4F D2 81BC 43 49 52 43 4C C5 81C2 50 41 49 4E D4 81C7 47 45 D4 81CA 50 55 D4 81CA 50 55 D4 81CA 50 55 D4 81CD 44 52 41 D7 81D1 50 43 4F 50 D9	** L8183 FCC L8186 FCC L8188 FCC L8198 FCC L8199 FCC L8199 FCC L8199 FCC L8145 FCC L8145 FCC L8145 FCC L8145 FCC L8148 FCC L8148 FCC L8181 FCC L8161 FCC L8162 FCC L8164 FCC L8164 FCC L8165 FCC L8165 FCC L8165 FCC L8166 FCC L8167 FCC L8167 FCC L8168 FCC L8167 FCC L8168 FCC	'DE',\$80+'L' 'EDI',\$80+'T' 'TRO',\$80+'N' 'TROF',\$80+'F' 'DE',\$80+'F' 'LE',\$80+'E' 'PCL',\$80+'S' 'PSE',\$80+'T' 'PRESE',\$80+'T' 'PRESE',\$80+'T' 'PCLEA',\$80+'R' 'COLO',\$80+'R' 'COLO',\$80+'R' 'CIRCL',\$80+'R' 'CIRCL',\$80+'T' 'GE',\$80+'T' 'GE',\$80+'T' 'DE',\$80+'T' 'DE',\$80+'T' 'DE',\$80+'T'	B5 B6 B7 B8 B9 BA BB BC BD BE CØ C1 C2 C3 C4 C5 C6 C6 C7
0153 0154 0155 0156 0157 0158 0159 0160 0161 0162 0163 0164 0165 0166 0167 0168 0169 0170 0171 0172	8183 44 45 CC 8186 45 44 49 D4 818A 54 52 4F CE 818E 54 52 4F CE 818B 54 52 4F 6 C6 8193 44 45 C6 8196 43 4C D3 8191 50 43 4C D3 81A1 50 53 45 D4 81AB 53 43 52 45 45 CE 81B1 50 43 4C 45 41 D2 81B7 43 4F 4C 4F D2 81BC 43 49 52 43 4C C5 81C2 50 41 49 4E D4 81CA 50 55 D4 81CD 44 52 41 D7 81D1 50 43 47 50 D9 81D6 50 40 4F 50 D9	** ** ** ** ** ** ** ** ** **	'DE',\$80+'L' 'EDI',\$80+'T' 'TRO',\$80+'T' 'TROF',\$80+'F' 'DE',\$80+'F' 'LE',\$80+'T' 'LIN',\$80+'E' 'PCL',\$80+'T' 'PRESE',\$80+'T' 'SCREE',\$80+'T' 'SCREE',\$80+'R' 'CCLOL',\$80+'R' 'CIRCL',\$80+'R' 'CIRCL',\$80+'T' 'PAIN',\$80+'T' 'GE',\$80+'T' 'DEA',\$80+'T' 'DEA',\$80+'T' 'DRA',\$80+'T' 'DRA',\$80+'T' 'POL',\$80+'T' 'POL',\$80+'T' 'POL',\$80+'T'	B5 B6 B7 B8 B9 BA BB BC BC BC C0 C1 C2 C3 C4 C5 C6 C7 C8
0153 0154 0155 0156 0157 0158 0159 0160 0161 0162 0163 0164 0165 0166 0167 0168 0171 0171	8183 44 45 CC 8186 45 44 49 D4 818A 54 52 4F CE 818E 54 52 4F 46 C6 8193 44 45 C6 8199 4C 45 D4 8199 4C 49 4E C5 819D 50 43 4C D3 81A1 50 53 45 D4 81A5 50 52 45 53 45 D4 81A5 50 43 4C 45 D2 81B1 50 43 4C 45 D2 81B1 50 43 4C 4F D2 81BC 43 49 52 43 4C C5 81C2 50 41 49 4E D4 81C7 47 45 D4 81CA 50 55 D4 81CA 50 55 D4 81CA 50 55 D4 81CD 44 52 41 D7 81D1 50 43 4F 50 D9	** L8183 FCC L8186 FCC L8188 FCC L8198 FCC L8199 FCC L8199 FCC L8199 FCC L8145 FCC L8145 FCC L8145 FCC L8145 FCC L8148 FCC L8148 FCC L8181 FCC L8161 FCC L8162 FCC L8164 FCC L8164 FCC L8165 FCC L8165 FCC L8165 FCC L8166 FCC L8167 FCC L8167 FCC L8168 FCC L8167 FCC L8168 FCC	'DE',\$80+'L' 'EDI',\$80+'T' 'TRO',\$80+'N' 'TROF',\$80+'F' 'DE',\$80+'F' 'LE',\$80+'E' 'PCL',\$80+'S' 'PSE',\$80+'T' 'PRESE',\$80+'T' 'PRESE',\$80+'T' 'PCLEA',\$80+'R' 'COLO',\$80+'R' 'COLO',\$80+'R' 'CIRCL',\$80+'R' 'CIRCL',\$80+'T' 'GE',\$80+'T' 'GE',\$80+'T' 'DE',\$80+'T' 'DE',\$80+'T' 'DE',\$80+'T'	B5 B6 B7 B8 B9 BA BB BC BD BE CØ C1 C2 C3 C4 C5 C6 C6 C7
0153 0154 0155 0156 0157 0158 0159 0161 0162 0163 0164 0165 0166 0167 0170 0171 0172 0173 0174 0175	8183 44 45 CC 8186 54 54 49 D4 8188 54 52 4F CE 8188 54 52 4F CE 8189 44 45 C6 8193 44 45 C6 8199 4C 45 D4 8199 4C 49 4E C5 8191 50 53 45 D4 8134 50 52 45 53 45 D4 8134 50 52 45 53 45 D4 8138 53 43 52 45 45 CE 8181 50 43 4C 45 41 D2 8186 43 49 52 43 4C C5 81C2 50 41 49 4E D4 81C7 47 45 D4 81C0 44 52 41 D7 81D1 50 43 4F 50 D9 81D6 50 4D 4F 44 C5 81D8 50 4C 4I D9 81DF 44 4C 4F B2 81DB 50 4C 4I D9 81DF 44 4C 4F E4 C4	** ** ** ** ** ** ** ** ** **	'DE',\$80+'L' 'EDI',\$80+'T' 'TRO',\$80+'T' 'TROF',\$80+'F' 'DE',\$80+'F' 'LE',\$80+'F' 'LE',\$80+'T' 'LIN',\$80+'E' 'PCL',\$80+'T' 'PRESE',\$80+'T' 'PRESE',\$80+'T' 'PCLEA',\$80+'R' 'CCLCL',\$80+'R' 'CCLCL',\$80+'R' 'CGRCL',\$80+'T' 'PAIN',\$80+'T' 'GE',\$80+'T' 'DRA',\$80+'T' 'PAOP',\$80+'T' 'PLON',\$80+'T' 'PLON',\$80+'T' 'PLON',\$80+'T' 'PLON',\$80+'T' 'PLON',\$80+'T' 'PLON',\$80+'T' 'PLON',\$80+'T'	B5 B6 B7 B8 B9 BA BB BC BC BC C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB
0153 0154 0155 0156 0157 0158 0159 0160 0161 0162 0163 0164 0165 0166 0167 0170 0171 0173 0174 0175 0177	8183 44 45 CC 8186 45 44 49 D4 818A 54 52 4F CE 818E 54 52 4F C6 8193 44 45 C6 8199 4C 45 D4 8199 4C 49 4E C5 819D 50 43 4C D3 81A1 50 53 45 D4 81A5 50 52 45 53 45 D4 81A5 50 52 45 53 45 D4 81A5 50 43 45 C45 41 D2 81B7 43 4F 4C 4F D2 81BC 43 49 52 43 4C C5 81C2 50 41 49 4E D4 81CA 747 45 D4 81CA 50 55 D4 81CA 50 55 D4 81CA 50 45 D4 81CA	** L8183 FCC L8186 FCC L818A FCC L818A FCC L8193 FCC L8193 FCC L8199 FCC L819D FCC L814A FCC L814B FCC L814B FCC L814B FCC L816C FCC	'DE',\$80+'L' 'EDI',\$80+'T' 'TRO',\$80+'N' 'TROF',\$80+'F' 'DE',\$80+'F' 'LE',\$80+'E' 'PL',\$80+'E' 'PCL',\$80+'S' 'PSE',\$80+'T' 'PRESE',\$80+'T' 'SCREE',\$80+'T' 'PCLEA',\$80+'R' 'COLO',\$80+'R' 'COLO',\$80+'R' 'CTRCL',\$80+'E' 'PAIN',\$80+'T' 'GE',\$80+'T' 'DE',\$80+'T' 'ROM',\$80+'T'	B5 B6 B7 B8 B9 BA BB BC BD BE CØ C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC
0153 0154 0155 0156 0157 0158 0159 0161 0162 0163 0164 0165 0166 0167 0170 0171 0172 0173 0174 0175	8183 44 45 CC 8186 54 54 49 D4 8188 54 52 4F CE 8188 54 52 4F CE 8189 44 45 C6 8193 44 45 C6 8199 4C 45 D4 8199 4C 49 4E C5 8191 50 53 45 D4 8134 50 52 45 53 45 D4 8134 50 52 45 53 45 D4 8138 53 43 52 45 45 CE 8181 50 43 4C 45 41 D2 8186 43 49 52 43 4C C5 81C2 50 41 49 4E D4 81C7 47 45 D4 81C0 44 52 41 D7 81D1 50 43 4F 50 D9 81D6 50 4D 4F 44 C5 81D8 50 4C 4I D9 81DF 44 4C 4F B2 81DB 50 4C 4I D9 81DF 44 4C 4F E4 C4	** ** ** ** ** ** ** ** ** **	'DE',\$80+'L' 'EDI',\$80+'T' 'TRO',\$80+'T' 'TROF',\$80+'F' 'DE',\$80+'F' 'LE',\$80+'F' 'LE',\$80+'T' 'LIN',\$80+'E' 'PCL',\$80+'T' 'PRESE',\$80+'T' 'PRESE',\$80+'T' 'PCLEA',\$80+'R' 'CCLCL',\$80+'R' 'CCLCL',\$80+'R' 'CGRCL',\$80+'T' 'PAIN',\$80+'T' 'GE',\$80+'T' 'DRA',\$80+'T' 'PAOP',\$80+'T' 'PLON',\$80+'T' 'PLON',\$80+'T' 'PLON',\$80+'T' 'PLON',\$80+'T' 'PLON',\$80+'T' 'PLON',\$80+'T' 'PLON',\$80+'T'	B5 B6 B7 B8 B9 BA BB BC BC BC C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB
0153 0154 0155 0156 0157 0158 0159 0160 0161 0162 0163 0164 0165 0166 0167 0170 0171 0173 0174 0177 0177 0178 0177 0178	8183 44 45 CC 8186 45 44 49 D4 818A 54 52 4F CE 818E 54 52 4F C6 8193 44 45 C6 8199 4C 45 D4 8199 4C 49 4E C5 819D 50 43 4C D3 81A1 50 53 45 D4 81A5 50 52 45 53 45 D4 81A5 50 52 45 53 45 D4 81A5 50 43 45 C45 41 D2 81B7 43 4F 4C 4F D2 81BC 43 49 52 43 4C C5 81C2 50 41 49 4E D4 81CA 747 45 D4 81CA 50 55 D4 81CA 50 55 D4 81CA 50 45 D4 81CA	** ** ** ** ** ** ** ** ** **	'DE',\$80+'L' 'EDI',\$80+'T' 'TRO',\$80+'N' 'TROF',\$80+'F' 'DE',\$80+'F' 'LE',\$80+'E' 'PL',\$80+'E' 'PCL',\$80+'S' 'PSE',\$80+'T' 'PRESE',\$80+'T' 'SCREE',\$80+'T' 'PCLEA',\$80+'R' 'COLO',\$80+'R' 'COLO',\$80+'R' 'CTRCL',\$80+'E' 'PAIN',\$80+'T' 'GE',\$80+'T' 'DE',\$80+'T' 'ROM',\$80+'T'	B5 B6 B7 B8 B9 BA BB BC BD BE CØ C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CC CD
0153 0154 0155 0156 0157 0158 0159 0161 0162 0163 0164 0165 0166 0167 0168 0170 0171 0172 0173 0174 0175 0176 0177 0177 0178 0179 0179 0179 0179	8183 44 45 CC 8186 45 44 49 D4 818A 54 52 4F CE 818E 54 52 4F 66 8193 44 45 C6 8199 4C 45 D4 8199 4C 49 4E C5 8190 50 43 4C D3 81A1 50 53 45 D4 81A5 50 52 45 53 45 D4 81A5 50 43 4C 45 41 D2 81B7 43 4F 4C 4F D2 81B7 43 4F 4C 4F D2 81BC 43 49 52 43 4C C5 81C2 50 41 49 4E D4 81C7 47 45 D4 81C7 47 45 D4 81C7 47 45 D4 81C7 47 45 D4 81C8 50 55 D4 81C9 40 41 49 4E D7 81D1 50 43 4F 50 D9 81D6 50 40 4F 44 C5 81D8 50 4C 41 D9 81DF 44 4C 4F 41 C4 81E4 52 45 4E 55 CD 81EB 55 53 49 4E C7	** L8183 FCC L8186 FCC L8188 FCC L8188 FCC L8193 FCC L8199 FCC L8199 FCC L8191 FCC L8145 FCC L8145 FCC L8145 FCC L8146 FCC L8147 FCC L8148 FCC L8167 FCC L8168 FCC ** DISPATCH TAI	"DE',\$80+'L' 'EDI',\$80+'T' 'TRO',\$80+'T' 'TROF',\$80+'F' 'DE',\$80+'F' 'LE',\$80+'F' 'LE',\$80+'T' 'LIN',\$80+'E' 'PCL',\$80+'T' 'PRESE',\$80+'T' 'SCREE',\$80+'T' 'SCREE',\$80+'R' 'COLO',\$80+'R' 'COLO',\$80+'R' 'COLO',\$80+'E' 'PAIN',\$80+'T' 'PCLE',\$80+'T' 'PU',\$80+'T' 'DRA',\$80+'T' 'PDA',\$80+'Y' 'PMOD',\$80+'Y' 'PMOD',\$80+'Y' 'PMOD',\$80+'Y' 'PMOD',\$80+'Y' 'BLOA',\$80+'P' 'RENU',\$80+'M' 'F',\$80+'N' 'USIN',\$80+'G'	B5 B6 B7 B8 B9 BA BB BC CØ C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD TOKEN #
0153 0154 0155 0156 0157 0158 0159 0160 0161 0162 0163 0164 0165 0166 0167 0170 0171 0173 0174 0177 0177 0178 0177 0178	8183 44 45 CC 8186 45 44 49 D4 818A 54 52 4F CE 818E 54 52 4F C6 8193 44 45 C6 8199 4C 45 D4 8199 4C 49 4E C5 819D 50 43 4C D3 81A1 50 53 45 D4 81A5 50 52 45 53 45 D4 81A5 50 52 45 53 45 D4 81A5 50 43 45 C45 41 D2 81B7 43 4F 4C 4F D2 81BC 43 49 52 43 4C C5 81C2 50 41 49 4E D4 81CA 747 45 D4 81CA 50 55 D4 81CA 50 55 D4 81CA 50 45 D4 81CA	** ** ** ** ** ** ** ** ** **	'DE',\$80+'L' 'EDI',\$80+'T' 'TRO',\$80+'N' 'TROF',\$80+'F' 'DE',\$80+'F' 'LE',\$80+'T' 'LIN',\$80+'E' 'PCL',\$80+'T' 'PRESE',\$80+'T' 'PRESE',\$80+'T' 'PRESE',\$80+'R' 'COLO',\$80+'R' 'COLO',\$80+'R' 'CIRCL',\$80+'R' 'CIRCL',\$80+'T' 'PAIN',\$80+'T' 'GE',\$80+'T' 'DE',\$80+'T' 'DE',\$80+'T' 'DLO',\$80+'E' 'PAIN',\$80+'Y' 'DLO',\$80+'E' 'PLA',\$80+'Y' 'PLO',\$80+'E' 'PLA',\$80+'N' 'SENU',\$80+'N' 'F',\$80+'N' 'USIN',\$80+'N'	B5 B6 B7 B8 B9 BA BB BC BD BE CØ C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CC CD
0153 0154 0155 0156 0157 0158 0159 0160 0161 0162 0163 0164 0165 0167 0168 0170 0171 0172 0173 0174 0175 0176 0177 0177 0178 0179 0181 0182 0181	8183 44 45 CC 8186 54 54 44 99 D4 818A 54 52 4F CE 818E 54 52 4F 46 C6 8193 44 45 C6 8196 4C 45 D4 8199 4C 49 4E C5 8181 50 43 4C D3 81A1 50 53 45 D4 81A5 50 52 45 53 45 D4 81A6 53 43 52 45 45 CE 81B1 50 43 4C 45 41 D2 81B7 43 4F 4C 4F D2 81BC 43 49 52 43 4C C5 81C2 50 41 49 4E D4 81C7 47 45 D4 81CA 50 55 D4 81CA 50 55 D4 81CA 50 55 D4 81CA 44 52 41 D7 81D1 50 43 4F 50 D9 81D6 50 4D 4F 44 C5 81D8 50 4C 41 D9 81D6 50 4D 4F 44 C5 81B8 50 4C 41 D9 81DF 44 4C 4F 41 C4 81E4 52 45 4E 55 CD 81E9 46 CE 81EB 55 53 49 4E C7	** ** ** ** ** ** ** ** ** **	"DE',\$80+'L' 'EDI',\$80+'T' 'TRO',\$80+'T' 'TROF',\$80+'F' 'DE',\$80+'F' 'LE',\$80+'T' 'LIN',\$80+'E' 'PCL',\$80+'T' 'PRESE',\$80+'T' 'SCREE',\$80+'T' 'SCREE',\$80+'T' 'SCREE',\$80+'R' 'COLO',\$80+'R' 'COLO',\$80+'R' 'COLO',\$80+'R' 'COLO',\$80+'E' 'PAIN',\$80+'T' 'PU',\$80+'T' 'PU',\$80+'T' 'DRA',\$80+'Y' 'PMOD',\$80+'Y' 'PMOD',\$80+'Y' 'PMOD',\$80+'Y' 'PMOD',\$80+'Y' 'BLOA',\$80+'D' 'RENU',\$80+'M' 'F',\$80+'N' 'USIN',\$80+'G' BLE FOR EXTENDED BASIC COMMANDS DEL EDIT TRON	B5 B6 B7 B8 B9 BA BB BC BD BE CØ C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CC CD TOKEN # DEL B5 EDIT B6 TRON B7
0153 0154 0155 0156 0157 0158 0159 0161 0162 0163 0164 0165 0166 0167 0170 0171 0172 0173 0174 0177 0177 0178 0179 0180 0181 0181 0182 0183 0184	8183 44 45 CC 8186 45 44 49 D4 8188 54 52 4F CE 8188 54 52 4F CE 8193 44 45 C6 8193 44 45 C6 8193 44 45 C6 8199 4C 49 4C C5 8190 50 43 4C D3 811A1 50 53 45 D4 81AB 53 43 52 45 45 CE 81B1 50 43 4C 4F 1D2 81B7 43 4F 4C 4F D2 81BC 43 49 52 43 4C C5 81C2 50 41 49 4E D4 81C7 47 45 D4 81CD 44 52 41 D7 81D1 50 43 4F 50 D9 81D6 50 40 4F 44 C5 81DB 50 4C 41 D9 81DF 44 4C 4F 41 C4 81E4 52 45 4E 55 CD 81E9 46 CE 81EB 55 53 49 4E C7	* RESERVED WOIN * ** L818A FCC L818A FCC L818A FCC L819A FCC L8193 FCC L8199 FCC L819D FCC L814A FCC L814A FCC L814A FCC L816F FCC	'DE',\$80+'L' 'EDI',\$80+'T' 'TRO',\$80+'N' 'TROF',\$80+'F' 'DE',\$80+'F' 'LE',\$80+'F' 'LE',\$80+'S' 'PSE',\$80+'T' 'PRESE',\$80+'T' 'PRESE',\$80+'N' 'PCLEA',\$80+'R' 'COLO',\$80+'R' 'COLO',\$80+'R' 'CIRCL',\$80+'E' 'PAIN',\$80+'T' 'GE',\$80+'T' 'PU',\$80+'T' 'DRA',\$80+'W' 'PCOP',\$80+'Y' 'DLOA',\$80+'E' 'PLA',\$80+'W' 'PCOP',\$80+'Y' 'DLOA',\$80+'W' 'PCOP',\$80+'Y' 'BLOA',\$80+'W' 'F',\$60+'W' 'SLE FOR EXTENDED BASIC COMMANDS DEL EDIT TRON TROFF	B5 B6 B7 B8 B9 B9 BA BB BC BD BE C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CC CC CD TOKEN # DEL B5 EDIT B6 TRON B7 TROFF B8
0153 0154 0155 0156 0157 0158 0159 0161 0162 0163 0164 0167 0168 0170 0171 0172 0173 0174 0177 0178 0176 0177 0178 0179 0181 0181 0181 0184 0183 0184	8183 44 45 CC 8186 45 44 49 D4 818A 54 52 4F CE 818E 54 52 4F C6 8193 44 45 C6 8193 44 45 C6 8199 4C 45 D4 8199 4C 49 4E C5 819D 50 43 4C D3 81A1 50 53 45 D4 81A5 50 52 45 53 45 D4 81A5 50 52 45 53 45 D4 81A5 50 43 45 C45 41 C2 81B7 43 4F 4C 4F D2 81B7 43 4F 4C 4F D2 81BC 43 49 52 43 4C C5 81C2 50 41 49 4E D4 81CA 50 55 D4 81CA 50 55 D4 81CA 50 45 D4 81CA 50 45 D4 81CA 50 55 D4 81CA 50 45 D4 81CA 50 55 D4 81CA 50 45 D4 81CA 50 55 D4	* RESERVED WOIN* ** L818A FCC L818A FCC L818A FCC L819A FCC L8193 FCC L8199 FCC L819D FCC L819D FCC L814A FCC L814B FCC L814B FCC L816C FCC L816D FCC L816D FCC L816D FCC L816D FCC L816B	'DE',\$80+'L' 'EDI',\$80+'T' 'TRO',\$80+'N' 'TROF',\$80+'F' 'DE',\$80+'F' 'LE',\$80+'T' 'LI',\$80+'E' 'PCL',\$80+'T' 'SCREE',\$80+'T' 'SCREE',\$80+'T' 'SCREE',\$80+'N' 'PCLEA',\$80+'R' 'CCLO',\$80+'E' 'PAIN',\$80+'E' 'PAIN',\$80+'T' 'DE',\$80+'T' 'PU',\$80+'T' 'PU',\$80+'T' 'PU',\$80+'B' 'PCOP',\$80+'Y' 'PMOD',\$80+'B' 'PLA',\$80+'B' 'PLA',\$80+'N' 'USIN',\$80+'M' 'F',\$60+'N' 'USIN',\$80+'G' SLE FOR EXTENDED BASIC COMMANDS DEL EDIT TRON TROFF DEF	B5 B6 B7 B8 B9 BA BB BC BD BE CØ C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD TOKEN # DEL B5 EDIT B6 TROOF BB B7 TROFF BB DEF B9 B8 B7 TROFF BB DEF B9
0153 0154 0155 0156 0157 0158 0159 0161 0162 0163 0164 0165 0166 0167 0170 0171 0172 0173 0174 0177 0177 0178 0179 0180 0181 0181 0182 0183 0184	8183 44 45 CC 8186 45 44 49 D4 8188 54 52 4F CE 8188 54 52 4F CE 8193 44 45 C6 8193 44 45 C6 8193 44 45 C6 8199 4C 49 4C C5 8190 50 43 4C D3 811A1 50 53 45 D4 81AB 53 43 52 45 45 CE 81B1 50 43 4C 4F 1D2 81B7 43 4F 4C 4F D2 81BC 43 49 52 43 4C C5 81C2 50 41 49 4E D4 81C7 47 45 D4 81CD 44 52 41 D7 81D1 50 43 4F 50 D9 81D6 50 40 4F 44 C5 81DB 50 4C 41 D9 81DF 44 4C 4F 41 C4 81E4 52 45 4E 55 CD 81E9 46 CE 81EB 55 53 49 4E C7	* RESERVED WOIN * ** L818A FCC L818A FCC L818A FCC L819A FCC L8193 FCC L8199 FCC L819D FCC L814A FCC L814A FCC L814A FCC L816F FCC	'DE',\$80+'L' 'EDI',\$80+'T' 'TRO',\$80+'N' 'TROF',\$80+'F' 'DE',\$80+'F' 'LE',\$80+'F' 'LE',\$80+'S' 'PSE',\$80+'T' 'PRESE',\$80+'T' 'PRESE',\$80+'N' 'PCLEA',\$80+'R' 'COLO',\$80+'R' 'COLO',\$80+'R' 'CIRCL',\$80+'E' 'PAIN',\$80+'T' 'GE',\$80+'T' 'PU',\$80+'T' 'DRA',\$80+'W' 'PCOP',\$80+'Y' 'DLOA',\$80+'E' 'PLA',\$80+'W' 'PCOP',\$80+'Y' 'DLOA',\$80+'W' 'PCOP',\$80+'Y' 'BLOA',\$80+'W' 'F',\$60+'W' 'SLE FOR EXTENDED BASIC COMMANDS DEL EDIT TRON TROFF	B5 B6 B7 B8 B9 B9 BA BB BC BD BE C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CC CD TOKEN # DEL B5 EDIT B6 TRON B7 TROFF B8
0153 0154 0155 0156 0157 0158 0159 0161 0162 0163 0164 0165 0166 0167 0170 0171 0172 0173 0174 0175 0176 0177 0178 0178 0181 0181 0182 0184 0187 0187	8183 44 45 CC 8186 45 44 49 D4 818A 54 52 4F CE 818E 54 52 4F CE 818E 54 52 4F 66 C6 8193 44 45 C6 8199 4C 45 D4 8199 4C 49 4E C5 819D 50 43 4C D3 81A1 50 53 45 D4 81A5 50 52 45 53 45 D4 81AB 53 43 52 45 45 C2 81BT 43 4F 4C 4F D2 81BC 43 49 52 43 4C C5 81C2 50 41 49 4E D4 81C7 47 45 D4 81C8 50 55 D4 81C9 40 47 47 50 D9 81D6 50 40 47 47 C5 81BB 50 4C 41 D9 81DF 44 4C 4F 41 C4 81E4 52 45 4E 55 CD 81EB 55 53 49 4E C7 81FØ 89 7Ø 81FØ 89 7Ø 81FØ 85 33 81F4 86 A7 81F6 86 A8 81F8 88 71 81FA AF 89 81FC 93 BB 81FC 93 BB	* RESERVED WOIN* ** L818A FCC L818A FCC L818A FCC L819A FCC L819A FCC L819A FCC L819B FCC L819D FCC L819D FCC L814A FCC L814A FCC L814B FCC L816C FCC L816D FCC L816D FCC L816D FCC L816B	"DE',\$80+'L' 'EDI',\$80+'T' 'TRO',\$80+'N' 'TROF',\$80+'F' 'DE',\$80+'F' 'LE',\$80+'F' 'LE',\$80+'E' 'PCL',\$80+'S' 'PSE',\$80+'T' 'SCREE',\$80+'T' 'SCREE',\$80+'T' 'SCREE',\$80+'R' 'CCLO',\$80+'E' 'PAIN',\$80+'E' 'PAIN',\$80+'E' 'PAIN',\$80+'T' 'DRA',\$80+'T' 'DRA',\$80+'T' 'DRA',\$80+'Y' 'PMOD',\$80+'Y' 'PMOD',\$80+'Y' 'PMOD',\$80+'Y' 'DLOA',\$80+'N' 'USIN',\$80+'M' 'F',\$80+'N' 'USIN',\$80+'G' BEL EDIT TRON TROFF DEF LET LINE PCLS	B5 B6 B7 B8 B9 BA BB BC BD BE CØ C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD TOKEN # DEL B5 EDIT B6 TROFF B8 DEF B9 LET BA LINE BB PCLS BC
0153 0154 0155 0156 0157 0158 0159 0161 0162 0163 0164 0165 0166 0167 0170 0171 0172 0173 0174 0175 0176 0177 0178 0178 0181 0182 0183 0184 0185 0188 0188 0188 0188 0188 0188 0188	8183 44 45 CC 8186 45 44 49 D4 818A 54 52 4F CE 818E 54 52 4F CE 818E 54 52 4F GC 8193 44 45 C6 8193 44 45 C6 8199 40 49 4E C5 8190 50 43 4C D3 81A1 50 53 45 D4 81A5 50 52 45 53 45 D4 81A6 53 43 52 45 45 CE 81B1 50 43 4C 45 41 D2 81B7 43 4F 4C 4F D2 81BC 43 49 52 43 4C C5 81C2 50 41 49 4E D4 81C7 47 45 D4 81CD 44 52 41 D7 81D1 50 43 4F 50 D9 81D6 50 4D 4F 44 C5 81D8 50 4C 41 D9 81D6 45 60 4D 4F 44 C5 81D8 50 4C 41 D9 81D7 44 4C 4F 41 C4 81D8 50 4C 41 D9 81D6 45 60 4D 4F 44 C5 81B8 55 53 49 4E C7 81F0 89 70 81F2 85 33 81F4 86 A7 81F6 86 A8 81F8 88 71 81FC 93 B8 81FC 93 C6	* RESERVED WOIN* ** L818A FCC L818A FCC L818A FCC L819A FCC L819A FCC L819A FCC L819B FCC L819B FCC L819B FCC L819B FCC L819B FCC L81A5 FCC L81A5 FCC L81A6 FCC L81A6 FCC L81A6 FCC L81A6 FCC L81B7 FCC L81B7 FCC L81B7 FCC L81B7 FCC L81B7 FCC L81C2 FCC L81C2 FCC L81C4 FCC L81C5 FCC L81C7 FCC L81C6 FCC L81C7 FCC L81C8 FCC L81C8 FCC L81C8 FCC L81C8 FCC L81C9 FCC L81DB FCC L81CB FCC L81C	"DE',\$80+'L' 'EDI',\$80+'T' 'TRO',\$80+'T' 'TROF',\$80+'F' 'DE',\$80+'F' 'LE',\$80+'T' 'LIN',\$80+'E' 'PCL',\$80+'T' 'PRESE',\$80+'T' 'SCREE',\$80+'T' 'SCREE',\$80+'R' 'COLO',\$80+'R' 'COLO',\$80+'R' 'COLO',\$80+'R' 'COLO',\$80+'T' 'PCLEA',\$80+'T' 'PCLA',\$80+'T' 'PLA',\$80+'T' 'PDA',\$80+'T' 'PDA',\$80+'T' 'DRA',\$80+'W' 'PCOP',\$80+'Y' 'PMOD',\$80+'E' 'PLA',\$80+'W' 'PCOP',\$80+'B' 'SENU',\$80+'B' 'SENU',\$80+'B' 'SENU',\$80+'B' 'RENU',\$80+'B' 'SENU',\$80+'B' 'RENU',\$80+'G' 'SELE FOR EXTENDED BASIC COMMANDS DEL EDIT TRON TROFF DEF LET LINE PCLS PSET	B5 B6 B7 B8 B9 B8 B9 BA BB BC BC BD BE CØ C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CC CD TOKEN # DEL B5 EDIT B6 TRON B7 TROFF B8 DEF B9 LET BA LINE BB PCLS BC PSET BD
0153 0154 0155 0156 0157 0158 0159 0161 0162 0163 0164 0165 0166 0167 0170 0171 0172 0173 0174 0175 0176 0177 0178 0178 0181 0181 0182 0184 0187 0187	8183 44 45 CC 8186 45 44 49 D4 818A 54 52 4F CE 818E 54 52 4F CE 818E 54 52 4F 66 C6 8193 44 45 C6 8199 4C 45 D4 8199 4C 49 4E C5 819D 50 43 4C D3 81A1 50 53 45 D4 81A5 50 52 45 53 45 D4 81AB 53 43 52 45 45 C2 81BT 43 4F 4C 4F D2 81BC 43 49 52 43 4C C5 81C2 50 41 49 4E D4 81C7 47 45 D4 81C8 50 55 D4 81C9 40 47 47 50 D9 81D6 50 40 47 47 C5 81BB 50 4C 41 D9 81DF 44 4C 4F 41 C4 81E4 52 45 4E 55 CD 81EB 55 53 49 4E C7 81FØ 89 7Ø 81FØ 89 7Ø 81FØ 85 33 81F4 86 A7 81F6 86 A8 81F8 88 71 81FA AF 89 81FC 93 BB 81FC 93 BB	* RESERVED WOIN* ** L818A FCC L818A FCC L818A FCC L819A FCC L819A FCC L819A FCC L819B FCC L819D FCC L819D FCC L814A FCC L814A FCC L814B FCC L816C FCC L816D FCC L816D FCC L816D FCC L816B	"DE',\$80+'L' 'EDI',\$80+'T' 'TRO',\$80+'N' 'TROF',\$80+'F' 'DE',\$80+'F' 'LE',\$80+'F' 'LE',\$80+'E' 'PCL',\$80+'S' 'PSE',\$80+'T' 'SCREE',\$80+'T' 'SCREE',\$80+'T' 'SCREE',\$80+'R' 'CCLO',\$80+'E' 'PAIN',\$80+'E' 'PAIN',\$80+'E' 'PAIN',\$80+'T' 'DRA',\$80+'T' 'DRA',\$80+'T' 'DRA',\$80+'Y' 'PMOD',\$80+'Y' 'PMOD',\$80+'Y' 'PMOD',\$80+'Y' 'DLOA',\$80+'N' 'USIN',\$80+'M' 'F',\$80+'N' 'USIN',\$80+'G' BEL EDIT TRON TROFF DEF LET LINE PCLS	B5 B6 B7 B8 B9 BA BB BC BD BE CØ C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD TOKEN # DEL B5 EDIT B6 TROFF B8 DEF B9 LET BA LINE BB PCLS BC

Ø194 Ø195 Ø196	82Ø8 95 46 82ØA 9E 9D	L8208 FDB L820A FDB	PCLEAR COLOR CIRCLE PAINT	PCLEAR CØ COLOR C1 CIRCLE C2 PAINT C3
	82ØE 97 55 821Ø 97 58	L820E FDB L8210 FDB		GET C4 PUT C5
	821Ø 97 58 8212 9C B6	L8212 FDB	DRAW	DRAW C6
a2a1	8214 97 23 8216 96 21	L8214 FDB L8216 FDB	PCOPY PMOD	PCOPY C7 PMODE C7
Ø2Ø2 Ø2Ø3	8218 9A 22	L8218 FDB	PLAY DLOAD	PLAY C9 DLOAD CA
0204	8218 9A 22 821A 8C 18 821C 8A Ø9	L821C FDB	RENUM	RENUM CB
0205 0206 0207 0208	,		UNCTION FOR EXTENDED BASIC PRECEEDED WITH AN \$FF BYTE	TOKEN #
	821E 41 54 CE	L821E FCC		94
0210 0211	8221 43 4F D3 1 8224 54 41 CE 1	L8221 FCC L8224 FCC	'TA',\$80+'N'	95 96
Ø212	8227 45 58 DØ	L8227 FCC	'EX',\$80+'P'	97 98
Ø214	822D 4C 4F C7	L822D FCC	'LO',\$8Ø+'G'	99
Ø215 Ø216	8230 50 4F D3 8233 53 51 D2	L823Ø FCC	'PO',\$80+'S' 'SO' \$80+'R'	9A 9B
Ø217	8236 48 45 58 A4	L8236 FCC	'HEX',\$8Ø+'\$'	9C
Ø218 Ø219	823A 56 41 52 50 54 D2 8240 49 4E 53 54 D2	L823A FCC L824Ø FCC	'VARPT',\$80+'R' 'INST'.\$80+'R'	9D 9E
0220	821E 41 54 CE 8221 43 4F D3 8224 54 41 CE 8227 45 58 DØ 822A 46 49 D8 822D 4C 4F C7 823Ø 5Ø 4F D3 8233 53 51 D2 8236 48 45 58 A4 823A 56 41 52 5Ø 54 D2 824Ø 49 4E 53 54 D2 8248 50 50 4F 9 4E D4 8248 50 50 6F 49 4E D4 825Ø 53 54 52 49 4E D4	L8245 FCC	'TIME', \$8Ø+'R'	9F
Ø221	825Ø 53 54 52 49 4E 47 1	L824A FCC L825Ø FCC	'STRING',\$80+'\$'	AØ A1
Ø223 Ø224	8256 A4	* .1IIMP TARIF I	FOR EXTENDED BASIC SECONDARY FUNC	TIONS
Ø225	,	*		TOKEN #
	0.000 0.0 7.0	L8257 FDB L8259 FDB	COS	ATN 94 COS 95
Ø228	825B 83 81 I 825D 84 F2 I	L825B FDB		TAN 96
0230	825F 85 24	L825D FDB L825F FDB		EXP 97 FIX 98
Ø231	8261 84 46 8263 86 AC	L8261 FDB L8263 FDB		LOG 99 POS 9A
Ø233	8265 84 80	L8265 FDB	SQR	SQR 9B
	8267 8B DD 1 8269 86 BE 1	L8267 FDB L8269 FDB	HEXDOL VARPTR	HEXDOL 9C VARPT 9D
Ø236	826B 87 7E 826D 89 68	L826B FDB	INSTR	INSTR 9E
Ø237 Ø238	826F 93 39 8271 87 4E		TIMER PPOINT	TIMER 9F PPOINT AØ
	02/1 0/ 4L	102/1 100	STRING	STRING A1
Ø241	,	* CONSOLE OUT	RAM HOOK DEVNUM L95AC B DEVNUM #-3 B L8285 \$02,S	
Ø242 Ø243	8273 ØD 6F 2 8275 10 27 13 33	XVEC3 TST LBEO	DEVNUM L95AC	CHECK DEVICE NUMBER BRANCH IF SCREEN
Ø244	8279 34 Ø4	PSHS	B	SAVE CHARACTER
Ø245 Ø246	827D C1 FD	CMPB	#-3	*GET DEVICE NUMBER AND *CHECK FOR DLOAD
Ø247 Ø248	827F 35 Ø4 8281 26 Ø2	PULS BNF	B 18285	GET CHARACTER BACK RETURN IF NOT DLOAD
Ø249	8283 32 62	LEAS	\$02,5	*TAKE RETURN OFF STACK & GO BACK TO ROUTINE
		* L8285 RTS		*THAT CALLED CONSOLE OUT
Ø252 Ø253	,	* CLOSE FILES	RAM HOOK - THIS CODE CORRECTS A	
Ø254	2006 06 65	* BUG IN 1.0 I	BASIC WHICH WAS FIXED BY 1.1 BASI DEVNUM	C
Ø255 Ø256	8286 96 6F 3 8288 4C	XVEC8 LDA INCA	DEVNUM	GET DEVICE NUMBER CHECK FOR CASSETTE
Ø257	8289 26 FA		L8285	RETURN IF NOT CASSETTE
Ø258 Ø259	828B 96 78 828D 81 Ø2	CMPA	FILSTA #\$02	GET FILE STATUS OPEN FOR OUTPUT?
	828F 26 F4 8291 96 79		L8285 CINCTR	RETURN IF NOT OPEN FOR OUTPUT GET CHARACTER BUFFER COUNTER
Ø262	8293 26 FØ	BNE	L8285	RETURN IF NOT EMPTY
Ø264	8295 ØF 6F 8297 32 62		DEVNUM \$02,S	SET DEVICE NUMBER TO SCREEN GET RETURN ADDRESS OFF OF STACK
Ø265 Ø266	8299 7E A4 44	JMP * RUN RAM HOOI	LA444 /	WRITE END OF FILE TAPE BLOCK
Ø267	829C CC BA 42	XVEC18 LDD	#\$BA42	MID HIGH VALUE + MID LOW VALUE
	829F DD DF 82A1 86 Ø2		VOLHI #\$Ø2	INITIALIZE PLAY VOLUME
	82A3 97 E2 82A5 97 DE		TEMPO OCTAVE	INITIALIZE TEMPO TO 2 INITIALIZE OCTAVE TO 3
0272	82A7 48	ASLA		X2
	82A8 97 E1 82AA ØF E5		NOTELN DOTVAL	INITIALIZE NOTE LENGTH TO 5 LEAR NOTE TIMER SCALE FACTOR
0275	82AC DC 8A 82AE DD E8	LDD	ZERO	ZERO ACCD INITIALIZE DRAW ANGLE AND SCALE TO 1
Ø277	82BØ C6 8Ø	LDB	#128	* INITIALIZE HORIZONTAL DEFAULT
	82B2 DD C7 82B4 C6 6Ø	STD LDB		* COORDINATE TO MID POSITION = INITIALIZE VERTICAL DEFAULT
0280	82B6 DD C9	STD		= COORDINATE TO MID POSITION
Ø281 Ø282	82B8 39	RTS * COMMAND INTI	ERPRETATION LOOP RAM HOOK	
Ø283	82B9 32 62	XVEC2Ø LEAS		PURGE RETURN ADDRESS FROM STACK ENABLE INTERRUPTS
Ø285	82BD BD AD EB	JSR	LADEB	CHECK FOR KEYBOARD BREAK
	82CØ 9E A6 82C2 9F 2F		CHARAD TINPTR	* GET CURRENT BASIC LINE * POINTER AND SAVE IT
Ø288	82C4 A6 8Ø	LDA		GET CURRENT INPUT CHARACTER AND ADVANCE POINTER

Ø289	82C6 27 Ø7		BEQ	L82CF	BRANCH IF END OF LINE
0290	82C8 81 3A		CMPA		CHECK FOR COLON
Ø291	82CA 27 25		BEQ	L82F1	CONTINUE INTERPRETING IF COLON
	82CC 7E B2 77			LB277	SNYTAX ERROR - COLON ONLY LEGAL LINE SEPARATOR
Ø293	82CF A6 81	L82CF	LDA	, X++	* GET 1ST BYTE OF ADDRESS OF NEXT
Ø294	82D1 97 ØØ			ENDFLG	* BASIC LINE AND SAVE IT
Ø295	82D3 26 Ø3	L82D8	BNE	L82D8	BRANCH IF NOT END OF PROGRAM
Ø296	82D5 7E AE 15		JMP	LAE15	RETURN TO DIRECT MODE - PRINT OK
Ø297	82D8 EC 8Ø	L82D8	LDD	, X+	GET LINE NUMBER OF NEXT LINE SAVE LINE NUMBER
Ø298	82DA DD 68		STD	CURLIN	SAVE LINE NUMBER
Ø299	82DC 9F A6		STX	CHARAD	SAVE ADDRESS NEXT BYTE TO INTERPRET
	82DE 96 AF		LDA	TRCFLG	TEST THE TRACE FLAG
0301	82EØ 27 ØF		BEQ	L82F1	BRANCH IF TRACE OFF
0302	82E2 86 5B		LDA	#\$5B	<pre><left for="" hand="" line="" marker="" number<="" pre="" tron=""></left></pre>
0303	82E4 BD A2 82		JSR	LA282	OUTPUT A CHARACTER
	82E7 96 68	L82F1	LDA	CURLIN	GET MS BYTE OF LINE NUMBER
	82E9 BD BD CC		JSR	LBDCC	CONVERT ACCD TO DECIMAL AND PRINT ON SCREEN
	82EC 86 5D		LDA	#\$5D	> RIGHT HAND MARKER FOR TRON LINE NUMBER
0307	82EE BD A2 82		JSR	LA282	OUTPUT A CHARACTER
Ø3Ø8	82F1 9D 9F	L82F1	JSR	GETNCH	GET A CHARACTER FROM BASIC
Ø3Ø9	82F3 1F A9		TFR	CC,B	SAVE STATUS IN ACCB
0310	82F5 81 98		CMPA	#\$98	CSAVE TOKEN
Ø311	82F7 27 1D		BEQ	L8316	GO DO A CSAVE
Ø312	82F5 81 98 82F7 27 1D 82F9 81 97 82FB 27 14		CMPA	#\$97 L8311 B,CC LADG6	CLOAD TOKEN
Ø313	82FB 27 14		BEQ	L8311	PROCESS CLOAD
Ø314	82FD 1F 9A		TFR	B,CC	GET STATUS REG BACK
Ø315	82FF BD AD C6		JSR	LADC6	LINK BACK TO BASIC S INTERPRETATION LOOP
Ø316	8302 20 B7		BRA	L82BB	GO TO MAIN INTERPRETATION LOOP
Ø317		* CRUNCE	I RAM I	100K	
0318	8304 AE 62	XVEC23	LDX	\$02,5	*CHECK TO SEE IF THE ROUTINE CALLING CRUNCH
Ø319	82FF 8D AD C6 8302 20 87 8304 AE 62 8306 8C AC 9D 8309 26 05 8308 8E 82 F1 830E AF 62 8310 39 8311 8D 8C 62 8314 20 A5 8316 8D 02 8318 20 A1 831A 9D 9F 831C 81 4D 831E 10 26 21 2A		CMPX	#LACYU	*IS COMING FROM THE MAIN LOOP IN BASIC
Ø32Ø	8309 Z6 Ø5		RNE	FR310	*AND BRANCH IF NOT
Ø321	830B 8F 87 F1		LUX	#L8211	=IF IT IS, DO NOT RETURN TO COLOR BASIC
Ø322	830E AF 62	10017	21X	⊅ 0∠,5	=BUT TO THE EXBAS PATCH INSTEAD
Ø323	8310 39	L8310	KIS	10000	CHECK EXPAC CLOAD HANDIES
0324	8311 BD 8C 62	L8311	JSR	L8C62	CHECK EXBAS CLOAD HANDLER
0325	8314 20 A5		RKA	L8ZBB	GO TO MAIN INTERPRETATION LOOP
0326	8316 8D Ø2	L8316	BSR	L831A	DO A CSAVE
0327	8318 20 AI		RKA	L8ZBB	GO TO MAIN INTERPRETATION LOOP
0328	831A 9D 9F	L831A * CSAVEN	JSR	GEINCH	GET A CHAR FROM BASIC
0329	8310 81 40		CMPA	#'M'	CHECK FOR CSAVEM
0330	831E 10 26 21 2A	+ 0041151	LBNE	LA44C	BRANCH IF IT S NOT CSAVEM
	0200 00 05	* CSAVE	100	CETNOU	OFT A CHAR FROM RACIO
	8322 9D 9F		J2K	GETNCH LA578 L836C CASBUF+13 L836C \$\%2.\S \$\mathrm{8}\mathrm{9}\mathrm{2}\mathrm{8}\mathrm{9}\mathrm{2}\mathrm{8}\mathrm{9}\mathrm{2}\mathrm{9}\mathrm{1}\mathrm{9}\mathrm{6}\mathrm{1}\mathrm{1}\mathrm{9}\mathrm{6}\mathrm{1}\mathrm{9}\mathrm{1}\mathrm{9}\mathrm{2}\mathrm{2}\mathrm{2}\mathrm{8}\mathrm{9}\mathrm{2}\mathrm{2}\mathrm{2}\mathrm{8}\mathrm{9}\mathrm{2}\mathrm{2}\mathrm{8}\mathrm{9}\mathrm{2}\mathrm{2}\mathrm{8}\mathrm{9}\mathrm{2}\mathrm{8}\mathrm{9}\mathrm{2}\mathrm{2}\mathrm{8}\mathrm{9}\mathrm{2}\mathrm{8}\mathrm{9}\mathrm{2}\mathrm{8}\mathrm{9}\mathrm{2}\mathrm{8}\mathrm{9}\mathrm{2}\mathrm{8}\mathrm{9}\mathrm{9}\mathrm{2}\mathrm{8}\mathrm{9}\m	GET A CHAR FROM BASIC
	8324 BD A5 78		J2K	LA5/8	GET NAME OF FILE FROM BASIC
Ø334	8327 8D 43		R2K	L838U	GO GET THE START ADDRESS PUT IT IN HEADER BUFFER GO GET END ADDRESS COMPARE TO START ADDRESS FC ERROR IF START > END GO GET XFER ADDRESS PUT IT IN HEADER BUFFER GET NEW CHARACTER RETURN IF NOT END OF LINE FILE TYPE (MACHINE LANGUAGE) X = 0000 FILE MODE AND ASCII FLAG WRITE HEADER BLOCK CLOSE CASSETTE FILES
Ø335	8329 BF Ø1 E7		217	CASBUTTIS	PUT IT IN HEADER BUFFER
Ø336	832C 8D 3E 832E AC 62		R2K	L838U	GU GET ENU AUDRESS
Ø337	832E AU 82		LDCC	\$02,S	COMPARE TO START ADDRESS
	8330 10 25 31 16		FRC2	LB44A	FC ERROR IF START > END
Ø339	8334 8D 36		R2K	CASSUS 11	GU GET AFEK ADDRESS
	8336 BF Ø1 E5		211	CETCCH CASBUT+11	PUT IT IN HEADER BUFFER
Ø341 Ø342	8339 9D A5 833B 26 D3		DNE	10210	RETURN IF NOT END OF LINE
Ø343	833D 86 Ø2		LDA	H¢ 0.2	FILE TYPE (MACHINE LANGUAGE)
Ø344	833F 9E 8A		LDX	7FD0	X = ØØØØ FILE MODE AND ASCII FLAG
Ø345	8341 BD A6 5F		.1CD	LAGE	WELLE HEVDED BLUCK
Ø346	8344 ØF 78		CLD	LA65F FILSTA	CLOSE CASSETTE FILES
Ø347	8346 ØC 7C		TNC	BLKTYP	BLOCK TYPE = 1
Ø348	8348 BD A7 D8			LA7D8	GO WRITE LEADER
	834B AE 64			\$Ø4,S	GET STARTING ADDRESS
	834D 9F 7E	L834D	STX	CRIIFAD	STORE BUFFER START ADOR
Ø351	834F 86 FF		LDA	#255	BLOCK SIZE = 2SS
Ø352	8351 97 7D			BLKLEN	STORE IN BLOCK SIZE
Ø353	8353 EC 62			\$Ø2,S	GET ENDING ADDRESS
Ø354					SUBTRACT START ADDRESS
Ø355	8357 24 Ø5			L835E	BRANCH IF MORE TO BE WRITTEN
Ø356	8359 32 66		LEAS	\$Ø6,S	CLEAN UP STACK
Ø357	835B 7E A4 91			LA491	WRITE FINAL BLOCK
Ø358	835E 10 83 00 FF	L835E		#\$00FF	AT LEAST 1 FULL BLK LEFT?
Ø359	8362 24 Ø3		BCC	L8367	YES
Ø36Ø	8364 5C		INCB		NO - PUT WHAT S LEFT IN BLKLEN
Ø361	8365 D7 7D			BLKLEN	BUFFER SIZE
Ø362	8367 BD A7 F4	L8367		LA7F4	WRITE A BLOCK
Ø363	836A 2Ø E1			L834D	GO DO SOME MORE
Ø364	836C BD B2 6D	L836C		SYNCOMMA	SYNTAX CHECK FOR COMMA
Ø365	836F BD B7 3D			LB73D	EVAL EXPR - RETURN VALUE IN X
Ø366	8372 EE E4		LDU	,S	SAVE RETURN ADDRESS IN U
Ø367	8374 AF E4			,\$	PUT THE EXPRESSION ON THE STACK
Ø368	8376 1F 35		TFR	U,PC	RETURN TO CALLING ADDRESS
Ø369					
0370		* COS		- 000/V) 10 DETERMINE	770 TDENTITY 000/V) 0711//F= '' "''
0371	0070 05 00 :-				RIG IDENTITY COS(X)=SIN((PI/2)+X)
0372	8378 8E 83 AB	COS		#L83AB	POINT X TO FP CONSTANT (P1/2)
Ø373	837B BD B9 C2			LB9C2	ADD FPAØ TO (X)
0374	837E 7E BF 78	L837E	JMP	LBF78	JUMP TO SIN ROUTINE
0375					
Ø376		* TAN		- TANKY) 16 DETERMINES	DIO IDENTITY TANAVA CANAVA (COCAVA
Ø377	0201 DD DC 05				RIG IDENTITY TAN(X)=SIN(X)/COS(X)
Ø378	8381 BD BC 2F	TAN		LBC2F	PACK FPAØ AND MOVE IT TO FPA3
Ø379	8384 ØF ØA			RELFLG	RESET QUADRANT FLAG
Ø38Ø	0206 OD E6			L837E	CALCULATE SIN OF ARGUMENT
	8386 8D F6		IDV		
Ø381	8388 8E ØØ 4A		LDX		POINT X TO FPA5
Ø382	8388 8E ØØ 4A 838B BD BC 35		JSR	LBC35	PACK FPAØ AND MOVE IT TO FPA5
Ø382 Ø383	8388 8E ØØ 4A 838B BD BC 35 838E 8E ØØ 4Ø		JSR LDX	LBC35 #V40	PACK FPAØ AND MOVE IT TO FPA5 POINT X TO FPA3
Ø382	8388 8E ØØ 4A 838B BD BC 35		JSR LDX	LBC35	PACK FPAØ AND MOVE IT TO FPA5

```
FORCE FPAG MANTISSA TO BE POSITIVE
ดวยร
         8394 ØF 54
                                                  CIR
                                                         FPASGN
                                                                                                   FORCE FFAM MANTISSA TO BE POSITIVE
GET THE QUADRANT FLAG - COS NEGATIVE IN QUADS 2,3
CALCULATE VALUE OF COS(FFAM)
CHECK EXPONENT OF FFAM
         8396 96 ØA
                                                         RELFLG
Ø386
                                                  LDA
Ø387
         8398 8D ØC
839A ØD 4F
                                                  BSR
                                                         18346
                                                         FPØEXP
Ø388
                                                  TST
                                                                                                   OV ERROR IF COS(X)=Ø
POINT X TO FPA5
Ø389
         839C 10 27 36 F2
                                                  IRFO IRA92
         83AØ 8E ØØ 4A
                                                         #V4A
0390
                                                  LDX
         83A3 7E BB 8F
                                       L83A3
                                                  JMP
                                                         LBB8F
                                                                                                   DIVIDE (X) BY FPAØ - SIN(X)/COS(X)
SAVE SIGN FLAG ON STACK
Ø391
                                                  PSHS
Ø392
         83A6 34 Ø2
                                       L83A6
         83A8 7E BF A6
                                                         LBFA6
                                                                                                   EXPAND POLYNOMIAL
Ø393
                                                  JMP
Ø394
         83AB 81 49 ØF DA A2
Ø395
                                       L83AB
                                                 FCB $81,$49,$ØF,$DA,$A2
                                                                                                   1.57079633 (PI/2)
0396
                                       * ATN
Ø397
                                      * A 12 TERM TAYLOR SERIES IS USED TO EVALUATE THE

* ARCTAN EXPRESSION. TWO DIFFERENT FORMULI ARE USED

* TO EVALUATE THE EXPRESSION DEPENDING UPON

* WHETHER OR NOT THE ARGUMENT SQUARED IS > OR < 1.0
Ø398
Ø399
0400
0401
0402
                                       * IF X**2<1 THEN ATN=X-(X**3)/3+(X**5)/5-(X**7)/7
0403
                                       * IF X**2>=1 THEN ATN=PI/2-(1/X-1/((X**3)*3)+(1/((X**5)*5)- )
Ø4Ø4
0405
                                                                                                   * GET THE SIGN OF THE MANTISSA AND
Ø4Ø6
         83BØ 96 54
                                                  LDA
                                                         FPØSGN
0407
         83B2 34 Ø2
                                                  PSHS A
                                                                                                   * SAVE IT ON THE STACK
BRANCH IF POSITIVE MANTISSA
         83B4 2A Ø2
                                                         L83B8
0408
                                                  BPL
                                                                                                   CHANGE SIGN OF FPAØ
* GET EXPONENT OF FPAØ AND
Ø4Ø9
         83B6 8D 24
                                                  BSR
                                                         L83D0
         83B8 96 4F
                                       L83B8
                                                         FPØEXP
0410
                                                  LDA
Ø411
         83BA 34 Ø2
                                                  PSHS
                                                                                                   * SAVE IT ON THE STACK
IS FPAO < 1.0?
                                                         A
#$81
0412
         83BC 81 81
                                                  CMPA
Ø413
         83BE 25 Ø5
                                                  BLO
                                                         L83C5
                                                                                                   POINT X TO FP CONSTANT 1.0
         83CØ 8E BA C5
Ø414
                                                  LDX
                                                         #LBAC5
                                                                                                   GET RECIPROCAL OF FPAØ
POINT (X) TO TAYLOR SERIES COEFFICIENTS
Ø415
         83C3 8D DE
                                                         L83A3
Ø416
         83C5 8E 83 EØ
                                       L83C5
                                                 LDX
                                                         #L83EØ
         83C8 BD BE FØ
                                                                                                   EXPAND POLYNOMIAL
                                                  JSR
                                                         LBEFØ
                                                                                                   GET EXPONENT OF ARGUMENT
Ø418
         83CB 35 Ø2
                                                  PIIIS
         83CD 81 81
                                                        #$81
                                                                                                   WAS ARGUMENT < 1.0?
Ø419
                                                  CMPA
9429
         83CF 25 Ø6
                                                  BLO
                                                         L83D7
                                                                                                   YFS
Ø421
         83D1 8E 83 AB
                                                        #L83AB
                                                                                                   POINT (X) TO FP NUMBER (PI/2)
                                                  LDX
                                                                                                   SUBTRACT FPAØ FROM (PI/2)
* GET SIGN OF INITIAL ARGUMENT MANTISSA
Ø422
         83D4 BD B9 B9
                                                  JSR
                                                         LB9B9
         83D7 35 Ø2
                                       L83D7
Ø423
                                                  PULS
                                                         Α
         83D9 4D
                                                                                                   * AND SET FLAGS ACCORDING TO IT
Ø424
                                                  TSTA
         83DA 2A Ø3
Ø425
                                                  BPL
                                                         L83DF
                                                                                                   RETURN IF ARGUMENT WAS POSITIVE
Ø426
         83DC 7E BE E9
                                                  JMP
                                                                                                   CHANGE MANTISSA SIGN OF FPAØ
                                       L83DC
0427
         83DF 39
                                       L83DF
                                                  RTS
Ø428
                                       * TCHEBYSHEV MODIFIED TAYLOR SERIES COEFFICIENTS FOR ARCTANGENT
0429
0430
         83EØ ØB
                                       L83EØ
                                                  FCB
                                                         $ØB
                                                                                                   TWELVE COEFFICIENTS
         83E1 76 B3 83 BD D3
83E6 79 1E F4 A6 F5
Ø431
                                       L83F1
                                                         $76,$B3,$83,$BD,$D3
                                                                                                   -6.84793912F-Ø4 1/23
                                                                                                   +4.85Ø94216E-Ø3 1/21
                                                         $79,$1E,$F4,$A6,$F5
Ø432
                                       L83E6
                                                  FCB
         83EB 7B 83 FC BØ 1Ø
83FØ 7C ØC 1F 67 CA
                                                         $7B,$83,$FC,$BØ,$1Ø
$7C,$ØC,$1F,$67,$CA
Ø433
                                       L83EB
                                                  FCB
                                                                                                   -0.0161117018
                                                                                                                       1/19
Ø434
                                                                                                   +0.0342096381
                                       L83F0
                                                  FCB
                                                                                                                        1/17
Ø435
         83F5 7C DE 53 CB C1
                                       L83F5
                                                         $7C,$DE,$53,$CB,$C1
                                                                                                    -0.0542791328
                                                         $7D,$14,$64,$70,$4C
$7D,$B7,$EA,$51,$7A
$7D,$63,$30,$88,$7E
Ø436
         83FA 7D 14 64 70 4C
                                       L83FA
                                                  FCB
                                                                                                   +0.0724571965
                                                                                                                        1/13
         83FF 7D B7 EA 51 7A
                                                                                                   -0.0898023954
Ø437
                                       L83FF
                                                  FCB
                                                                                                                        1/11
         84Ø4 7D 63 3Ø 88 7E
Ø438
                                       L84Ø4
                                                  FCB
                                                                                                   +0.110932413
                                                                                                                        1/9
Ø439
         84Ø9 7E 92 44 99 3A
                                       L84Ø9
                                                         $7E,$92,$44,$99,$3A
                                                                                                    -0.142839808
                                                                                                                        1/7
9449
         840F 7F 4C CC 91 C7
                                       184ØF
                                                  FCB
                                                         $7E,$4C,$CC,$91,$C7
$7F,$AA,$AA,$AA,$13
                                                                                                   +Ø 199999121
                                                                                                                        1/5
         8413 7F AA AA AA 13
Ø441
                                       L8413
                                                  FCB
                                                                                                   -0.333333316
                                                                                                                        1/3
0442
         8418 81 00 00 00 00
                                       L8418
                                                  FCB
                                                         $81,$00,$00,$00,$00
                                                                                                   +1.0000000000
                                                                                                                        1/1
Ø443
Ø444
                                       *** TCHEBYSHEV MODIFIED TAYLOR SERIES COEFFICIENTS FOR LN(X)
Ø445
Ø446
                                       L841D
                                                                                                   FOUR COEFFICIENTS
         841D Ø3
                                                                                                   Ø.434255942 (2/7)*(1/LN(2))
Ø.576584541 (2/5)*(1/LN(2))
Ø447
         841E 7F 5E 56 CB 79
                                       L841E
                                                  FCB
                                                         $7F.$5E.$56.$CB.$79
         8423 8Ø 13 9B ØB 64
Ø448
                                       L8423
                                                  FCB
                                                         $80,$13,$9B,$0B,$64
Ø449
         8428 80 76 38 93 16
                                       18428
                                                  FCB
                                                         $80 $76 $38 $93 $16
                                                                                                   0.961800759
                                                                                                                   (2/3)*(1/IN(2))
                                                                                                                   (2/1)*(1/LN(2))
Ø45Ø
         842D 82 38 AA 3B 20
                                       L842D
                                                  FCB
                                                         $82.$38.$AA.$3B.$20
                                                                                                   2.88539007
0451
Ø452
         8432 8Ø 35 Ø4 F3 34
                                       L8432
                                                        $80,$35,$04,$F3,$34
                                                                                                   1/SQR(2)
Ø453
         8437 81 35 Ø4 F3 34
                                       L8437
                                                  FCB
                                                        $81,$35,$Ø4,$F3,$34
                                                                                                   SQR(2)
Ø454
Ø455
                                                 FCB
Ø456
         843C 8Ø 8Ø ØØ ØØ ØØ
                                       L843C
                                                       $80,$80,$00,$00,$00
                                                                                                   -.5
0457
                                                 FCB $80.$31.$72.$17.$F8
         8441 8Ø 31 72 17 F8
                                       L8441
                                                                                                   LN(2)
0458
Ø459
0460
                                       * LOG - NATURAL LOGARITHM (LN)
Ø461
                                      * THE NATURAL OR NAPERIAN LOGARITHM IS CALCULATED USING
* MATHEMATICAL IDENTITIES. FPAB IS OF THE FORM FPAB=A*(2**B) (SCIENTIFIC
* NOTATION). THEREFORE, THE LOG ROUTINE DETERMINES THE VALUE OF
* LN(A*(2**B)). A SERIES OF MATHEMATICAL IDENTITIES WILL EXPAND THIS
0462
Ø463
Ø464
Ø465
Ø466
                                       * TERM: LN(A*(2**B))=(-1/2+(1/LN(2))*(LN(A*SQR(2)))+B)*LN(2). ALL OF 
* THE TERMS OF THE LATTER EXPRESSION ARE CONSTANTS EXCEPT FOR THE
Ø467
                                       * LN(A*SQR(2)) TERM WHICH IS EVALUATED USING THE TAYLOR SERIES EXPANSION
Ø468
         8446 BD BC 6D
                                                 JSR LBC6D
LBLE LB44A
                                                                                                   CHECK STATUS OF FPAØ
Ø469
                                       LOG
                                                                                                   FC ERROR IF NEGATIVE OR ZERO
POINT (X) TO FP NUMBER (1/SOR(2))
0470
         8449 10 2F 2F FD
Ø471
         844D 8F 84 32
                                                  IDX
                                                        #18432
                                                         FPØEXP
                                                                                                   *GET EXPONENT OF ARGUMENT
Ø472
         845Ø 96 4F
                                                  LDA
                                                                                                   *SUBTRACT OFF THE BIAS AND *SAVE IT ON THE STACK
0473
         8452 80 80
                                                  SUBA #$8Ø
         8454 34 Ø2
Ø474
                                                  PSHS
                                                        Α
Ø475
         8456 86 8Ø
8458 97 4F
                                                  I DA
                                                         #$8Ø
                                                                                                   =FORCE EXPONENT OF FPAØ
=TO BE ZERO
                                                         FPØEXP
Ø476
                                                  STA
         845A BD B9 C2
845D 8E 84 37
                                                                                                   ADD FPAØ TO (X)
POINT X TO SQR(2)
Ø477
                                                  JSR
                                                         LB9C2
Ø478
                                                         #L8437
                                                  LDX
                                                                                                   DIVIDE SQR(2) BY FPAØ
POINT X TO FP VALUE OF 1.00
Ø479
         8460 BD BB 8F
                                                         LBB8F
         8463 8E BA C5
                                                         #LBAC5
0480
                                                  LDX
```

EXTENDED BASIC UNRAVELLED II APPENDIX B ORIGIN:SPECTRAL ASSOC DISASSEMBLY OF EXTENDED BASIC 1.1 REVISED:12/26/1999 WALTER K ZYDHEK

Ø481	8466 BD B9 B9		JSR	LB9B9	SUBTRACT FPAØ FROM (X)
Ø482		* NOW FP	PAØ = (1-SQR(2)*X)/(1+SQR(2)*X) WHERE X	(IS ARGUMENT
Ø483	8469 8E 84 1D			#L841D	POINT X TO TABLE OF COEFFICIENTS
Ø484	846C BD BE FØ		JSR		EXPAND POLYNOMIAL
Ø485	846F 8E 84 3C			#L843C	POINT X TO FP VALUE OF (5)
Ø486 Ø487	8472 BD B9 C2 8475 35 Ø4		PULS	LB9C2	ADD FPAØ TO X GET EXPONENT OF ARGUMENT BACK (WITHOUT BIAS)
Ø488	8477 BD BD 99			LBD99	ADD ACCB TO FPAØ
Ø489	847A 8E 84 41			#L8441	POINT X TO LN(2)
Ø49Ø	847D 7E BA CA			LBACA	MULTIPLY FPAØ * LN(2)
0491					
Ø492		* SQR			
Ø493	848Ø BD BC 5F	SQR	JSR	LBC5F	MOVE FPAØ TO FPA1
Ø494	8483 8E BE CØ			#LBECØ	POINT (X) TO FP NUMBER (.5)
Ø495	8486 BD BC 14		JSR	LBC14	COPY A PACKED NUMBER FROM (X) TO FPAØ
Ø496 Ø497		* ADITUM	METIC O	PERATOR FOR EXPONENTIATION JUMPS	
Ø497				RMULA USED TO EVALUATE EXPONENTI	
Ø499				$X \perp N = E**(FPAØ*LN(FPA1)), E=$	
0500	8489 27 67	L8489	BEQ		DO A NATURAL EXPONENTIATION IF EXPONENT = Ø
0501	848B 4D		TSTA		*CHECK VALUE BEING EXPONENTIATED
0502	848C 26 Ø3		BNE	L8491	*AND BRANCH IF IT IS ⇔ Ø
0503	848E 7E BA 3A		JMP		FPAØ=Ø IF RAISING ZERO TO A POWER
Ø5Ø4	8491 8E ØØ 4A	L8491	LDX		* PACK FPAØ AND SAVE
Ø5Ø5	8494 BD BC 35		JSR	LBC35	* IT IN FPA5 (ARGUMENT S EXPONENT)
Ø5Ø6 Ø5Ø7	8497 5F 8498 96 61		CLRB	FP1SGN	ACCB=DEFAULT RESULT SIGN FLAG; Ø=POSITIVE *CHECK THE SIGN OF ARGUMENT
Ø5Ø8	849A 2A 1Ø		BPL		*BRANCH IF POSITIVE
Ø5Ø9	849C BD BC EE		JSR		CONVERT EXPONENT INTO AN INTEGER
0510	849F 8E ØØ 4A		LDX		POINT X TO FPA5 (ORIGINAL EXPONENT)
Ø511	84A2 96 61			FP1SGN	GET MANTISSA SIGN OF FPA1 (ARGUMENT)
Ø512	84A4 BD BC AØ		JSR		*COMPARE FPAØ TO (X) AND
Ø513	84A7 26 Ø3		BNE	L84AC	*BRANCH IF NOT EQUAL
Ø514 Ø515	84A9 43		COMA	CHARAC	TOGGLE FPA1 MANTISSA SIGN - FORCE POSITIVE
Ø515 Ø516	84AA D6 Ø1 84AC BD BC 4C	L84AC		LBC4C	GET LS BYTE OF INTEGER VALUE OF EXPONENT (RESULT SIGN FLAG) COPY FPA1 TO FPA0; ACCA = MANTISSA SIGN
Ø517	84AF 34 Ø4	LOTAC	PSHS		PUT RESULT SIGN FLAG ON THE STACK
	84B1 BD 84 46		JSR		GET NATURAL LOGARITHM OF FPAØ
Ø519	84B4 8E ØØ 4A		LDX		POINT (X) TO FPA5
0520	84B7 BD BA CA		JSR	LBACA	MULTIPLY FPAØ BY FPA5
Ø521	84BA 8D 36		BSR		CALCULATE E**(FPAØ)
Ø522	84BC 35 Ø2		PULS	A	* GET RESULT SIGN FLAG FROM THE STACK
Ø523 Ø524	84BE 46 84BF 10 25 3A 26		RORA LBCS	LREEO	* AND BRANCH IF NEGATIVE
Ø525	84C3 39		RTS	LDEES	CHANGE SIGN OF FPAØ MANTISSA
Ø526	0403 33		ICI S		
Ø527		* CORREC	CTION F	ACTOR FOR EXPONENTIAL FUNCTION	
	0404 04 00 44 00 00				
Ø528	84C4 81 38 AA 3B 29	L84C4	FCB	\$81,\$38,\$AA,\$3B,\$29	1.44269504 (CF)
Ø529	84C4 81 38 AA 3B 29	*			
Ø529 Ø53Ø	84C4 81 38 AA 3B 29	* * TCHEBY		<pre>\$81,\$38,\$AA,\$3B,\$29 ODIFIED TAYLOR SERIES COEFFICIEN</pre>	
Ø529 Ø53Ø Ø531		* * TCHEBY *	YSHEV M	ODIFIED TAYLOR SERIES COEFFICIEN	NTS FOR E**X
Ø529 Ø53Ø Ø531 Ø532	84C9 Ø7	* * TCHEBY * L84C9	YSHEV M	ODIFIED TAYLOR SERIES COEFFICIEN	NTS FOR E**X EIGHT COEFFICIENTS
0529 0530 0531 0532 0533	84C9 Ø7 84CA 71 34 58 3E 56	* * TCHEBY * L84C9 L84CA	YSHEV M FCB FCB	ODIFIED TAYLOR SERIES COEFFICIEN \$07 \$71,\$34,\$58,\$3E,\$56	NTS FOR E**X EIGHT COEFFICIENTS 2.14987637E-05: 1/(7!*(CF**7))
Ø529 Ø53Ø Ø531 Ø532	84C9 Ø7 84CA 71 34 58 3E 56 84CF 74 16 7E B3 1B	* * TCHEBY * L84C9	YSHEV M FCB FCB FCB	ODIFIED TAYLOR SERIES COEFFICIEN \$07 \$71,\$34,\$58,\$3E,\$56 \$74,\$16,\$7E,\$83,\$18	HTS FOR E**X EIGHT COEFFICIENTS 2.14987637E-Ø5: 1/(7!*(CF**7)) 1.4352314E-Ø4 : 1/(6!*(CF**6))
0529 0530 0531 0532 0533 0534	84C9 Ø7 84CA 71 34 58 3E 56	* * TCHEBY * L84C9 L84CA L84CF	FCB FCB FCB FCB FCB	ODIFIED TAYLOR SERIES COEFFICIEN \$07 \$71,\$34,\$58,\$3E,\$56	NTS FOR E**X EIGHT COEFFICIENTS 2.14987637E-05: 1/(7!*(CF**7))
0529 0530 0531 0532 0533 0534 0535	84C9 Ø7 84CA 71 34 58 3E 56 84CF 74 16 7E B3 1B 84D4 77 2F EE E3 85	*	FCB FCB FCB FCB FCB FCB	ODIFIED TAYLOR SERIES COEFFICIEN \$07 \$71,\$34,\$58,\$3E,\$56 \$74,\$16,\$7E,\$83,\$1B \$77,\$2F,\$EE,\$E3,\$85	NTS FOR E**X EIGHT COEFFICIENTS 2.14987637E-05: 1/(7!*(CF**7)) 1.435231E-04: 1/(6!*(CF**6)) 1.34226348E-03: 1/(5!*(CF**5))
0529 0530 0531 0532 0533 0534 0535 0536 0537 0538	84C9 Ø7 84CA 71 34 58 3E 56 84CF 74 16 7E B3 1B 84D4 77 2F EE E3 85 84D9 7A 1D 84 1C 2A 84DE 7C 63 59 58 ØA 84E3 7E 75 FD E7 C6	* * TCHEBY * L84C9 L84CA L84CF L84D4 L84D9 L84DE L84E3	FCB FCB FCB FCB FCB FCB FCB FCB	ODIFIED TAYLOR SERIES COEFFICIEN \$07 \$71,\$34,\$58,\$3E,\$56 \$74,\$16,\$7E,\$83,\$1B \$77,\$2F,\$EE,\$E3,\$85 \$7A,\$1D,\$84,\$1C,\$2A \$7C,\$63,\$59,\$58,\$ØA \$7E,\$75,\$FD,\$E7,\$C6	EIGHT COEFFICIENTS 2.14987637E-05: 1/(7!*(CF**7)) 1.4352314E-04: 1/(6!*(CF**6)) 1.34226348E-03: 1/(5!*(CF**5)) 9.61401701E-03: 1/(4!*(CF**4)) 0.0555051269: 1/(3!*(CF**3)) 0.240226385: 1/(2!*(CF**2))
0529 0530 0531 0532 0533 0534 0535 0536 0537 0538 0539	84C9 Ø7 84CA 71 34 58 3E 56 84CF 74 16 7E B3 1B 84D4 77 2F EE E3 85 84D9 7A 1D 84 1C 2A 84DE 7C 63 59 58 ØA 84E3 7E 75 FD E7 C6 84E8 8Ø 31 72 18 1Ø	* TCHEBY * L84C9 L84CA L84CF L84D4 L84D9 L84DE L84E3 L84E8	FCB FCB FCB FCB FCB FCB FCB FCB FCB	ODIFIED TAYLOR SERIES COEFFICIEN \$07 \$71,\$34,\$58,\$3E,\$56 \$74,\$16,\$7E,\$83,\$1B \$77,\$2F,\$EE,\$E3,\$85 \$7A,\$1D,\$84,\$1C,\$2A \$7C,\$63,\$59,\$58,\$0A \$7E,\$75,\$FD,\$E7,\$C6 \$80,\$31,\$72,\$18,\$10	EIGHT COEFFICIENTS 2.14987637E-05: 1/(7!*(CF**7)) 1.4352314E-04: 1/(6!*(CF**6)) 1.34226348E-03: 1/(5!*(CF**5)) 9.61401701E-03: 1/(4!*(CF**4)) 0.0555051269: 1/(3!*(CF**3)) 0.240226395: 1/(2!*(CF**2)) 0.693147186: 1/(2!*(CF**2))
0529 0530 0531 0532 0533 0534 0535 0536 0537 0538 0539 0540	84C9 Ø7 84CA 71 34 58 3E 56 84CF 74 16 7E B3 1B 84D4 77 2F EE E3 85 84D9 7A 1D 84 1C 2A 84DE 7C 63 59 58 ØA 84E3 7E 75 FD E7 C6	* * TCHEBY * L84C9 L84CA L84CF L84D4 L84D9 L84DE L84E3	FCB FCB FCB FCB FCB FCB FCB FCB FCB	ODIFIED TAYLOR SERIES COEFFICIEN \$07 \$71,\$34,\$58,\$3E,\$56 \$74,\$16,\$7E,\$83,\$1B \$77,\$2F,\$EE,\$E3,\$85 \$7A,\$1D,\$84,\$1C,\$2A \$7C,\$63,\$59,\$58,\$ØA \$7E,\$75,\$FD,\$E7,\$C6	EIGHT COEFFICIENTS 2.14987637E-05: 1/(7!*(CF**7)) 1.4352314E-04: 1/(6!*(CF**6)) 1.34226348E-03: 1/(5!*(CF**5)) 9.61401701E-03: 1/(4!*(CF**4)) 0.0555051269: 1/(3!*(CF**3)) 0.240226385: 1/(2!*(CF**2))
0529 0530 0531 0532 0533 0534 0535 0536 0537 0538 0539 0540 0541	84C9 Ø7 84CA 71 34 58 3E 56 84CF 74 16 7E B3 1B 84D4 77 2F EE E3 85 84D9 7A 1D 84 1C 2A 84DE 7C 63 59 58 ØA 84E3 7E 75 FD E7 C6 84E8 8Ø 31 72 18 1Ø	* * TCHEBY * L84C9 L84CA L84CF L84D4 L84D9 L84DE L84E3 L84E8 L84ED *	FCB FCB FCB FCB FCB FCB FCB FCB FCB	ODIFIED TAYLOR SERIES COEFFICIEN \$07 \$71,\$34,\$58,\$3E,\$56 \$74,\$16,\$7E,\$83,\$1B \$77,\$2F,\$EE,\$E3,\$85 \$7A,\$1D,\$84,\$1C,\$2A \$7C,\$63,\$59,\$58,\$0A \$7E,\$75,\$FD,\$E7,\$C6 \$80,\$31,\$72,\$18,\$10	EIGHT COEFFICIENTS 2.14987637E-05: 1/(7!*(CF**7)) 1.4352314E-04: 1/(6!*(CF**6)) 1.34226348E-03: 1/(5!*(CF**5)) 9.61401701E-03: 1/(4!*(CF**4)) 0.0555051269: 1/(3!*(CF**3)) 0.240226395: 1/(2!*(CF**2)) 0.693147186: 1/(2!*(CF**2))
0529 0530 0531 0532 0533 0534 0535 0536 0537 0538 0539 0541	84C9 Ø7 84CA 71 34 58 3E 56 84CF 74 16 7E B3 1B 84D4 77 2F EE E3 85 84D9 7A 1D 84 1C 2A 84DE 7C 63 59 58 ØA 84E3 7E 75 FD E7 C6 84E8 8Ø 31 72 18 1Ø	* * TCHEBY * L84C9 L84CA L84CF L84D4 L84D9 L84DE L84E3 L84E8 L84EB * * EXP (FCB FCB FCB FCB FCB FCB FCB FCB FCB FCB	0DIFIED TAYLOR SERIES COEFFICIEN \$07 \$71,\$34,\$58,\$3E,\$56 \$74,\$16,\$7E,\$83,\$1B \$77,\$2F,\$EE,\$E3,\$85 \$7A,\$1D,\$84,\$1C,\$2A \$7C,\$66,\$59,\$55,\$0A \$7E,\$75,\$FD,\$E7,\$C6 \$80,\$31,\$72,\$18,\$10 \$81,\$00,\$00,\$00,\$00	EIGHT COEFFICIENTS 2.14987637E-05: 1/(7!*(CF**7)) 1.4352314E-04: 1/(6!*(CF**6)) 1.34226348E-03: 1/(5!*(CF**5)) 9.61401701E-03: 1/(4!*(CF**4)) 0.0555051269: 1/(3!*(CF**3)) 0.24022635: 1/(2!*(CF**2)) 0.693147186: 1/(1!*(CF**1)) 1.
0529 0530 0531 0532 0533 0534 0535 0536 0537 0538 0539 0540 0541	84C9 Ø7 84CA 71 34 58 3E 56 84CF 74 16 7E B3 1B 84D4 77 2F EE E3 85 84D9 7A 1D 84 1C 2A 84DE 7C 63 59 58 ØA 84E3 7E 75 FD E7 C6 84E8 8Ø 31 72 18 1Ø	* TCHEBY * L84C9 L84CA L84CF L84D4 L84D9 L84D9 L84E3 L84E8 L84E0 * EXP (* THE EX	FCB FCB FCB FCB FCB FCB FCB FCB FCB FCB	ODIFIED TAYLOR SERIES COEFFICIEN \$07 \$71,\$34,\$58,\$3E,\$56 \$74,\$16,\$7E,\$83,\$1B \$77,\$2F,\$EE,\$E3,\$85 \$7A,\$1D,\$84,\$1C,\$2A \$7C,\$63,\$59,\$58,\$0A \$7E,\$75,\$FD,\$E7,\$C6 \$80,\$31,\$72,\$18,\$10	EIGHT COEFFICIENTS 2.14987637E-05: 1/(7!*(CF**7)) 1.4352314E-04: 1/(6!*(CF**6)) 1.34226314E-03: 1/(5!*(CF**5)) 9.61401701E-03: 1/(4!*(CF**4)) 0.0555051269: 1/(3!*(CF**4)) 0.240226385: 1/(2!*(CF**2)) 0.693147186: 1/(1!*(CF**1)) 1. RST MULTIPLYING THE
0529 0530 0531 0532 0533 0534 0535 0536 0537 0538 0539 0540 0541 0542 0543	84C9 Ø7 84CA 71 34 58 3E 56 84CF 74 16 7E B3 1B 84D4 77 2F EE E3 85 84D9 7A 1D 84 1C 2A 84DE 7C 63 59 58 ØA 84E3 7E 75 FD E7 C6 84E8 8Ø 31 72 18 1Ø	* * TCHEBY * L84C9 L84CA L84CF L84D4 L84D9 L84DE L84E3 L84E8 L84ED * * EXP (* ARGUME * ARGUME	FCB FCB FCB FCB FCB FCB FCB FCB FCB FCB	ODIFIED TAYLOR SERIES COEFFICIEN \$07 \$71,\$34,\$58,\$3E,\$56 \$74,\$16,\$7E,\$83,\$1B \$77,\$2F,\$EE,\$E3,\$85 \$77,\$12,\$18,\$10,\$2A \$7C,\$63,\$59,\$50,\$0A \$7E,\$75,\$FD,\$E7,\$C6 \$80,\$31,\$72,\$18,\$10 \$81,\$00,\$00,\$00,\$00 IAL FUNCTION IS EVALUATED BY FIR A CORRECTION FACTOR (CF). AFTER 127 WILL YIELD A ZERO RESULT (NC	EIGHT COEFFICIENTS 2.14987637E-05: 1/(7!*(CF**7)) 1.4352314E-04: 1/(6!*(CF**6)) 1.34226348E-03: 1/(5!*(CF**5)) 9.6140179IE-03: 1/(4!*(CF**4)) 0.0555051269: 1/(3!*(CF**3)) 0.240226385: 1/(2!*(CF**2)) 0.693147186: 1/(1!*(CF**1)) 1. RST MULTIPLYING THE THIS IS DONE, AN 0 UNDERFLOW) FOR A
0529 0530 0531 0532 0533 0534 0535 0536 0537 0538 0539 0541 0542 0543 0544 0545	84C9 Ø7 84CA 71 34 58 3E 56 84CF 74 16 7E B3 1B 84D4 77 2F EE E3 85 84D9 7A 1D 84 1C 2A 84DE 7C 63 59 58 ØA 84E3 7E 75 FD E7 C6 84E8 8Ø 31 72 18 1Ø	* TCHEBY L84C9 L84C4 L84C4 L84C9 L84D9 L84D9 L84B1 L84E8 L84E0 * EXP (* THE EX * ARGUME * NEGATI	FCB FCB FCB FCB FCB FCB FCB FCB FCB FCB	0DIFIED TAYLOR SERIES COEFFICIEN \$07 \$71,\$34,\$58,\$3E,\$56 \$74,\$16,\$7E,\$83,\$1B \$77,\$2F,\$EE,\$E3,\$85 \$7A,\$1D,\$84,\$1C,\$2A \$7C,\$63,\$59,\$58,\$0A \$7E,\$75,\$FD,\$E7,\$C6 \$80,\$31,\$72,\$18,\$10 \$81,\$00,\$00,\$00,\$00 IAL FUNCTION IS EVALUATED BY FIFA A CORRECTION FACTOR (CF). AFTER 127 WILL YIELD A ZERO RESULT (NC	EIGHT COEFFICIENTS 2.14987637F-055: 1/(7!*(CF**7)) 1.4352314F-04: 1/(6!*(CF**6)) 1.34226314F-03: 1/(5!*(CF**5)) 9.61401701F-03: 1/(4!*(CF**4)) 0.0555051269: 1/(3!*(CF**4)) 0.240226385: 1/(2!*(CF**2)) 0.693147186: 1/(1!*(CF**1)) 1. RST MULTIPLYING THE THIS IS DONE, AN 0 UNDERFLOW) FOR A OR FOR A POSITIVE
0529 0530 0531 0532 0533 0534 0535 0536 0537 0538 0540 0541 0542 0543 0544 0545 0546 0547	84C9 Ø7 84CA 71 34 58 3E 56 84CF 74 16 7E B3 1B 84D4 77 2F EE E3 85 84D9 7A 1D 84 1C 2A 84DE 7C 63 59 58 ØA 84E3 7E 75 FD E7 C6 84E8 8Ø 31 72 18 1Ø	* TCHEBY * L84C9 L84C4 L84C4 L84D4 L84D9 L84D9 L84B8 * * EXP (FCB	0DIFIED TAYLOR SERIES COEFFICIEN \$07 \$71,\$34,\$58,\$3E,\$56 \$74,\$16,\$7E,\$83,\$1B \$77,\$2F,\$EE,\$E3,\$85 \$7A,\$10,\$84,\$1C,\$2A \$7C,\$63,\$59,\$58,\$0A \$7E,\$75,\$FD,\$E7,\$C6 \$80,\$31,\$72,\$18,\$10 \$81,\$00,\$00,\$00,\$00 IAL FUNCTION IS EVALUATED BY FIFA A CORRECTION FACTOR (CF). AFTER 127 WILL YIELD A ZERO RESULT (NC UMENT OR AN 'OV' (OVERFLOW) ERRC E POLYNOMIAL COEFFICIENTS ARE MC	EIGHT COEFFICIENTS 2.14987637E-Ø5: 1/(7!*(CF**7)) 1.4352314E-Ø4: 1/(6!*(CF**6)) 1.34226348E-Ø3: 1/(5!*(CF**5)) 9.61401701E-Ø3: 1/(4!*(CF**4)) Ø.0555951269: 1/(3!*(CF**3)) Ø.240226385: 1/(2!*(CF**2)) Ø.693147186: 1/(1!*(CF**1)) 1. RST MULTIPLYING THE THIS IS DONE, AN 0 UNDERFLOW) FOR A DR FOR A POSITIVE DDIFIED TO REFLECT
0529 0530 0531 0532 0533 0534 0535 0536 0537 0538 0539 0541 0542 0543 0544 0545 0546 0547	84C9 Ø7 84CA 71 34 58 3E 56 84CF 74 16 7E B3 1B 84D4 77 2F EE E3 85 84D9 7A 1D 84 1C 2A 84DE 7C 63 59 58 ØA 84E3 7E 75 FD E7 C6 84E8 8Ø 31 72 18 1Ø	* TCHEBY * L84C9 L84C4 L84C4 L84D4 L84D9 L84D9 L84B8 * * EXP (FCB	0DIFIED TAYLOR SERIES COEFFICIEN \$07 \$71,\$34,\$58,\$3E,\$56 \$74,\$16,\$7E,\$83,\$1B \$77,\$2F,\$EE,\$E3,\$85 \$7A,\$1D,\$84,\$1C,\$2A \$7C,\$63,\$59,\$58,\$0A \$7E,\$75,\$FD,\$E7,\$C6 \$80,\$31,\$72,\$18,\$10 \$81,\$00,\$00,\$00,\$00 IAL FUNCTION IS EVALUATED BY FIFA A CORRECTION FACTOR (CF). AFTER 127 WILL YIELD A ZERO RESULT (NC	EIGHT COEFFICIENTS 2.14987637E-Ø5: 1/(7!*(CF**7)) 1.4352314E-Ø4: 1/(6!*(CF**6)) 1.34226348E-Ø3: 1/(5!*(CF**5)) 9.61401701E-Ø3: 1/(4!*(CF**4)) Ø.0555951269: 1/(3!*(CF**3)) Ø.240226385: 1/(2!*(CF**2)) Ø.693147186: 1/(1!*(CF**1)) 1. RST MULTIPLYING THE THIS IS DONE, AN 0 UNDERFLOW) FOR A DR FOR A POSITIVE DDIFIED TO REFLECT
## ## ## ## ## ## ## ## ## ## ## ## ##	84C9 Ø7 84CA 71 34 58 3E 56 84CF 74 16 7E B3 1B 84D4 77 2F EE E3 85 84D9 7A 1D 84 1C 2A 84DE 7C 63 59 58 ØA 84E3 7E 75 FD E7 C6 84E8 8Ø 31 72 18 1Ø 84ED 81 ØØ ØØ ØØ	* TCHEBY * L84C9 L84C9 L84CF L84D4 L84D9 L84D8 L84E3 L84E3 * * EXP (* * * * * * * * * * * * * * * * * * *	FCB FCB FCB FCB FCB FCB FCB FCB FCB FCB	0DIFIED TAYLOR SERIES COEFFICIEN \$07 \$71,\$34,\$58,\$3E,\$56 \$74,\$16,\$7E,\$83,\$1B \$77,\$2F,\$EE,\$E3,\$85 \$7A,\$1D,\$84,\$1C,\$2A \$7C,\$63,\$59,\$58,\$0A \$7E,\$75,\$FD,\$E7,\$C6 \$80,\$31,\$72,\$18,\$10 \$81,\$00,\$00,\$00,\$00 IAL FUNCTION IS EVALUATED BY FIFA A CORRECTION FACTOR (CF). AFTER 127 WILL YIELD A ZERO RESULT (NC UMENT OR AN 'OV' (OVERFLOM) ERRC E POLYNOMIAL COEFFICIENTS ARE MC PLICATION AT THE START OF THE EV	EIGHT COEFFICIENTS 2.14987637F-Ø5: 1/(7!*(CF**7)) 1.4352314F-Ø4: 1/(6!*(CF**6)) 1.34226314F-Ø4: 1/(6!*(CF**5)) 9.61401701F-Ø3: 1/(4!*(CF**4)) 0.0555051269: 1/(3!*(CF**4)) 0.240226385: 1/(2!*(CF**2)) 0.693147186: 1/(1!*(CF**1)) 1. RST MULTIPLYING THE THIS IS DONE, AN 0 UNDERFLOW) FOR A OR FOR A POSITIVE DOIFIED TO REFLECT //ALUATION PROCESS.
0529 0530 0531 0532 0533 0534 0535 0536 0537 0538 0539 0541 0542 0543 0544 0545 0546 0547 0548 0548	84C9 Ø7 84CA 71 34 58 3E 56 84CF 74 16 7E B3 1B 84D4 77 2F EE E3 85 84D9 7A 1D 84 1C 2A 84DE 7C 63 59 58 ØA 84E3 7E 75 FD E7 C6 84E8 8Ø 31 72 18 1Ø 84ED 81 ØØ ØØ ØØ	* TCHEBY * L84C9 L84C4 L84C4 L84D4 L84D9 L84D9 L84B8 * * EXP (FCB FCB FCB FCB FCB FCB FCB FCB FCB FCB	0DIFIED TAYLOR SERIES COEFFICIEN \$07 \$71,\$34,\$58,\$3E,\$56 \$74,\$16,\$7E,\$83,\$1B \$77,\$2F,\$EE,\$E3,\$85 \$7A,\$10,\$84,\$1C,\$2A \$7C,\$63,\$59,\$58,\$0A \$7E,\$75,\$FD,\$E7,\$C6 \$80,\$31,\$72,\$18,\$10 \$81,\$00,\$00,\$00,\$00 IAL FUNCTION IS EVALUATED BY FIFA A CORRECTION FACTOR (CF). AFTER 127 WILL YIELD A ZERO RESULT (NC UMENT OR AN 'OV' (OVERFLOW) ERRC E POLYNOMIAL COEFFICIENTS ARE MC	EIGHT COEFFICIENTS 2.14987637E-Ø5: 1/(7!*(CF**7)) 1.4352314E-Ø4: 1/(6!*(CF**6)) 1.34226348E-Ø3: 1/(5!*(CF**5)) 9.61401701E-Ø3: 1/(4!*(CF**4)) Ø.0555081269: 1/(3!*(CF**3)) Ø.240226385: 1/(2!*(CF**2)) Ø.693147186: 1/(1!*(CF**1)) 1. RST MULTIPLYING THE THIS IS DONE, AN) UNDEFFLOW) FOR A DR FOR A POSITIVE DDIFIED TO REFLECT /ALUATION PROCESS. POINT X TO THE CORRECTION FACTOR
## ## ## ## ## ## ## ## ## ## ## ## ##	84C9 Ø7 84CA 71 34 58 3E 56 84CF 74 16 7E B3 1B 84D4 77 2F EE E3 85 84D9 7A 1D 84 1C 2A 84DE 7C 63 59 58 ØA 84E3 7E 75 FD E7 C6 84E8 8Ø 31 72 18 1Ø 84ED 81 ØØ ØØ ØØ	* TCHEBY * L84C9 L84C9 L84CF L84D4 L84D9 L84D8 L84E3 L84E3 * * EXP (* * * * * * * * * * * * * * * * * * *	FCB FCB FCB FCB FCB FCB FCB FCB FCB FCB	0DIFIED TAYLOR SERIES COEFFICIEN \$07 \$71,\$34,\$58,\$3E,\$56 \$74,\$16,\$7E,\$83,\$18 \$77,\$2F,\$EE,\$E3,\$85 \$7A,\$10,\$84,\$1C,\$2A \$7C,\$63,\$59,\$50,\$80 \$7E,\$75,\$FD,\$E7,\$C6 \$88,\$31,\$72,\$18,\$10 \$81,\$00,\$00,\$00,\$00 IAL FUNCTION IS EVALUATED BY FIFA CORRECTION FACTOR (CF). AFTER TO A CORRECTION FACTOR (CF). AFTER TO A CORRECTION FACTOR (CF). AFTER EVALUATED BY FIFA CONTROL OF THE EVAL	EIGHT COEFFICIENTS 2.14987637F-Ø5: 1/(7!*(CF**7)) 1.4352314F-Ø4: 1/(6!*(CF**6)) 1.34226314F-Ø4: 1/(6!*(CF**5)) 9.61401701F-Ø3: 1/(4!*(CF**4)) 0.0555051269: 1/(3!*(CF**4)) 0.240226385: 1/(2!*(CF**2)) 0.693147186: 1/(1!*(CF**1)) 1. RST MULTIPLYING THE THIS IS DONE, AN 0 UNDERFLOW) FOR A OR FOR A POSITIVE DOIFIED TO REFLECT //ALUATION PROCESS.
0529 0530 0531 0532 0533 0534 0535 0536 0537 0538 0540 0541 0542 0543 0544 0545 0546 0547 0548 0549 0551	84C9 Ø7 84CA 71 34 58 3E 56 84CF 74 16 7E B3 1B 84D4 77 2F EE E3 85 84D9 7A 1D 84 1C 2A 84DE 7C 63 59 58 ØA 84E3 7E 75 FD E7 C6 84E8 8Ø 31 72 18 1Ø 84ED 81 ØØ ØØ ØØ 84ED 81 84 C4	* TCHEBY * L84C9 L84C9 L84CF L84D4 L84D9 L84D8 L84E3 L84E3 * * EXP (* * * * * * * * * * * * * * * * * * *	YSHEV M FCB FCB FCB FCB FCB FCB FCB FCB FCB FC	0DIFIED TAYLOR SERIES COEFFICIEN \$07 \$71,\$34,\$58,\$3E,\$56 \$74,\$16,\$7E,\$83,\$18 \$77,\$2F,\$EE,\$E3,\$85 \$7A,\$10,\$84,\$1C,\$2A \$7C,\$63,\$59,\$50,\$80 \$7E,\$75,\$FD,\$E7,\$C6 \$88,\$31,\$72,\$18,\$10 \$81,\$00,\$00,\$00,\$00 IAL FUNCTION IS EVALUATED BY FIFA CORRECTION FACTOR (CF). AFTER TO A CORRECTION FACTOR (CF). AFTER TO A CORRECTION FACTOR (CF). AFTER EVALUATED BY FIFA CONTROL OF THE EVAL	EIGHT COEFFICIENTS 2.14987637E-05: 1/(7!*(CF**7)) 1.4352314E-04: 1/(6!*(CF**6)) 1.34226348E-03: 1/(5!*(CF**5)) 9.61401701E-03: 1/(4!*(CF**4)) 0.0555051269: 1/(3!*(CF**3)) 0.240226385: 1/(2!*(CF**2)) 0.693147186: 1/(1!*(CF**1)) 1. RST MULTIPLYING THE THIS IS DONE, AN 01 UNDERFLOW) FOR A DR FOR A POSITIVE DOIFIED TO REFLECT VALUATION PROCESS. POINT X TO THE CORRECTION FACTOR MULTIPLY FPA0 BY (X)
0529 0530 0531 0532 0533 0534 0535 0536 0537 0538 0539 0541 0542 0543 0544 0545 0546 0547 0546 0547 0549 0550 0551	84C9 Ø7 84CA 71 34 58 3E 56 84CF 74 16 7E B3 1B 84D4 77 2F EE E3 85 84D9 7A 1D 84 1C 2A 84DE 7C 63 59 58 ØA 84E3 7E 75 FD E7 C6 84E8 8Ø 31 72 18 1Ø 84ED 81 ØØ ØØ ØØ ØØ	* TCHEBY * L84C9 L84C9 L84CF L84D4 L84D9 L84D8 L84E3 L84E3 * * EXP (* * * * * * * * * * * * * * * * * * *	YSHEV M FCB FCB FCB FCB FCB FCB FCB FCB FCB FC	ODIFIED TAYLOR SERIES COEFFICIEN \$07 \$71,\$34,\$58,\$3E,\$56 \$74,\$16,\$7E,\$83,\$1B \$77,\$2F,\$EE,\$E3,\$85 \$7A,\$10,\$84,\$1C,\$2A \$7C,\$63,\$59,\$58,\$0A \$7E,\$75,\$FD,\$E7,\$C6 \$80,\$31,\$72,\$18,\$10 \$81,\$00,\$00,\$00,\$00 IAL FUNCTION IS EVALUATED BY FIFA A CORRECTION FACTOR (CF). AFTER 127 WILL YIELD A ZERO RESULT (NR UMENT OR AN 'OV' (OVERFLOW) EXECTED FOR THE POLYNOMIAL COEFFICIENTS ARE MC PLICATION AT THE START OF THE EV #L84C4 LBACA LBB2F FP0EXP	EIGHT COEFFICIENTS 2.14987637F-Ø5: 1/(7!*(CF**7)) 1.4352314F-Ø4: 1/(6!*(CF**6)) 1.34226314F-Ø4: 1/(6!*(CF**5)) 9.61401701F-Ø3: 1/(4!*(CF**4)) 0.0555051269: 1/(3!*(CF**4)) 0.240226385: 1/(2!*(CF**2)) 0.693147186: 1/(1!*(CF**1)) 1. RST MULTIPLYING THE THIS IS DONE, AN 0 UNDERFLOW) FOR A OR FOR A POSITIVE DOIFIED TO REFLECT //ALUATION PROCESS. POINT X TO THE CORRECTION FACTOR MULTIPLY FPAØ BY (X) PACK FPAØ AND STORE IT IN FPA3
0529 0530 0531 0532 0533 0534 0535 0536 0537 0538 0539 0541 0542 0543 0544 0545 0546 0547 0546 0547 0552 0553 0552	84C9 Ø7 84CA 71 34 58 3E 56 84CF 74 16 7E B3 1B 84D4 77 2F EE E3 85 84D9 7A 1D 84 1C 2A 84DE 7C 63 59 58 ØA 84E3 7E 75 FD E7 C6 84E8 8Ø 31 72 18 1Ø 84ED 81 ØØ ØØ ØØ ØØ 84ED 81 ØØ ØØ ØØ ØØ	* TCHEBY * L84C9 L84C9 L84C9 L84CF L84D4 L84D9 L84E3 L84E3 * * EXP (* THE EX * ARGUME * NEGATI * ARGUME * THE CF	YSHEV M FCB FCB FCB FCB FCB FCB FCB FCB FCB FC	0DIFIED TAYLOR SERIES COEFFICIEN \$07 \$71,\$34,\$58,\$3E,\$56 \$74,\$16,\$7E,\$83,\$1B \$77,\$2F,\$EE,\$E3,\$85 \$7A,\$1D,\$84,\$1C,\$2A \$7C,\$63,\$59,\$58,\$0A \$7E,\$75,\$FD,\$E7,\$C6 \$80,\$31,\$72,\$18,\$10 \$81,\$00,\$00,\$00,\$00 IAL FUNCTION IS EVALUATED BY FIFA A CORRECTION FACTOR (CF). AFTER 127 WILL YIELD A ZERO RESULT (NC UMENT OR AN 'OV' (OVERFLOW) ERRC E POLYNOMIAL COEFFICIENTS ARE MC PLICATION AT THE START OF THE EV #L84C4 LB8CA LBC2F FP0EXP #\$88 L8504	EIGHT COEFFICIENTS 2.14987637E-95: 1/(7!*(CF**7)) 1.4352314E-04: 1/(6!*(CF**6)) 1.34226314E-03: 1/(6!*(CF**5)) 9.61401701E-03: 1/(4!*(CF**4)) 0.0555051269: 1/(3!*(CF**4)) 0.240226385: 1/(2!*(CF**2)) 0.693147186: 1/(1!*(CF**1)) 1. RST MULTIPLYING THE THIS IS DONE, AN 0 UNDERFLOW) FOR A OR FOR A POSITIVE DOIFIED TO REFLECT //ALUATION PROCESS. POINT X TO THE CORRECTION FACTOR MULTIPLY FPAØ BY (X) PACK FPAØ AND STORE IT IN FPA3 *GET EXPONENT OF FPAØ AND *COMPARE TO THE MAXIMUM VALUE BRANCH IF FPAØ < 128
0529 0530 0531 0532 0533 0534 0535 0536 0537 0540 0541 0542 0543 0544 0544 0545 0545 0551 0550 0551 0552 0553	84C9 Ø7 84CA 71 34 58 3E 56 84CF 74 16 7E B3 1B 84D4 77 2F EE E3 85 84D9 7A 1D 84 1C 2A 84DE 7C 63 59 58 ØA 84E3 7E 75 FD E7 C6 84E8 8Ø 31 72 18 1Ø 84ED 81 ØØ ØØ ØØ ØØ 84ED 81 ØØ ØØ ØØ ØØ	* TCHEBY * L84C9 L84C4 L84C9 L84C4 L84D4 L84D9 L84B8 L84E8 L84E8 L84E8 * * EXP (EX	YSHEV M FCB	0DIFIED TAYLOR SERIES COEFFICIEN \$07 \$71,\$34,\$58,\$3E,\$56 \$74,\$16,\$7E,\$83,\$1B \$77,\$2F,\$EE,\$E3,\$85 \$78,\$10,\$84,\$10,\$2A \$76,\$63,\$59,\$58,\$0A \$7E,\$75,\$FD,\$E7,\$C6 \$80,\$31,\$72,\$18,\$10 \$81,\$00,\$00,\$00,\$00 IAL FUNCTION IS EVALUATED BY FIFA A CORRECTION FACTOR (CF). AFTER 127 WILL YIELD A ZERO RESULT (NC UMENT OR AN 'OV' (OVERFLOW) FERE E POLYNOMIAL COEFFICIENTS ARE MC PLICATION AT THE START OF THE EV #L84C4 LBACA LBACA LBC2F FP0EXP #\$88 LB504 LBB5C	EIGHT COEFFICIENTS 2.14987637E-Ø5: 1/(7!*(CF**7)) 1.4352314E-Ø4: 1/(6!*(CF**6)) 1.34226348E-Ø3: 1/(5!*(CF**5)) 9.614Ø1701E-Ø3: 1/(4!*(CF**4)) Ø.0555Ø1269: 1/(3!*(CF**3)) Ø.24Ø226385: 1/(2!*(CF**2)) Ø.693147186: 1/(1!*(CF**1)) 1. RST MULTIPLYING THE THIS IS DONE, AN 0 UNDERFLOW) FOR A NP FOR A POSITIVE DOIFIED TO REFLECT VALUATION PROCESS. POINT X TO THE CORRECTION FACTOR MULTIPLY FPAØ BY (X) PPACK FPAØ AND STORE IT IN FPA3 *GET EXPONENT OF FPAØ AND *COMPARE TO THE MAXIMUM VALUE BBRANCH IF FPAØ < 128 SET FPAØ = Ø OR OV ERROR
## ## ## ## ## ## ## ## ## ## ## ## ##	84C9 07 84CA 71 34 58 3E 56 84CF 74 16 7E B3 1B 84D4 77 2F EE E3 85 84D9 7A 1D 84 1C 2A 84DE 7C 63 59 58 0A 84E3 7E 75 FD E7 C6 84E8 80 31 72 18 10 84ED 81 00 00 00 00 84ED 81 00 00 20 00 84F2 8E 84 C4 84F5 8D 8A CA 84F8 BD 8C 2F 84FB 96 4F 84FD 81 88 84FF 25 03 8501 7E 8B 5C 8504 8D 8C EE	* TCHEBY * L84C9 L84C9 L84C9 L84CF L84D4 L84D9 L84E3 L84E3 * * EXP (* THE EX * ARGUME * NEGATI * ARGUME * THE CF	YSHEV M FCB	0DIFIED TAYLOR SERIES COEFFICIEN \$07 \$71,\$34,\$58,\$3E,\$56 \$71,\$34,\$58,\$3E,\$56 \$77,\$2F,\$EE,\$E3,\$85 \$77,\$10,\$84,\$10,\$2A \$77,\$63,\$59,\$58,\$6A \$77,\$75,\$FD,\$E7,\$C6 \$80,\$31,\$72,\$18,\$10 \$81,\$00,\$00,\$00,\$00 IAL FUNCTION IS EVALUATED BY FIFA CORRECTION FACTOR (CF). AFTER ACTOR (CF). AFTER EVALUATED BY FIFA CORRECTION FACTOR (CF). AFTER EVALUATED BY FIFA CORRECTION AT THE START OF THE S	EIGHT COEFFICIENTS 2.14987637E-Ø5: 1/(7!*(CF**7)) 1.4352314E-Ø4: 1/(6!*(CF**6)) 1.34226348E-Ø3: 1/(5!*(CF**5)) 9.614017Ø1E-Ø3: 1/(4!*(CF**4)) Ø.0555951269: 1/(3!*(CF**3)) Ø.240226385: 1/(2!*(CF**2)) Ø.693147186: 1/(1!*(CF**1)) 1. RST MULTIPLYING THE THIS IS DONE, AN OUNDERFLOW) FOR A OR FOR A POSITIVE DUIFIED TO REFLECT VALUATION PROCESS. POINT X TO THE CORRECTION FACTOR MULTIPLY FPAØ BY (X) PACK FPAØ AND STORE IT IN FPA3 *GET EXPONENT OF FPAØ AND *COMPARE TO THE MAXIMUM VALUE BRANCH IF FPAØ < 128 SET FPAØ = Ø OR OV ERROR CONVERT FPAØ TO INTEGER
## ## ## ## ## ## ## ## ## ## ## ## ##	84C9 07 84CA 71 34 58 3E 56 84CF 74 16 7E B3 1B 84D4 77 2F EE E3 85 84D9 7A 1D 84 1C 2A 84DE 7C 63 59 58 0A 84E3 7E 75 FD E7 C6 84E8 80 31 72 18 10 84ED 81 00 00 00 00 00	* TCHEBY * L84C9 L84C4 L84C9 L84C4 L84D4 L84D9 L84B8 L84E8 L84E8 L84E8 * * EXP (EX	YSHEV M FCB	0DIFIED TAYLOR SERIES COEFFICIEN \$07 \$71,\$34,\$58,\$3E,\$56 \$74,\$16,\$7E,\$83,\$1B \$77,\$2F,\$EE,\$E3,\$85 \$73,\$10,\$84,\$1C,\$2A \$77,\$72,\$10,\$84,\$1C,\$2A \$77,\$75,\$F0,\$E7,\$C6 \$80,\$31,\$72,\$18,\$10 \$81,\$00,\$00,\$00,\$00 IAL FUNCTION IS EVALUATED BY FIFA A CORRECTION FACTOR (CF). AFTER 127 WILL YIELD A ZERO RESULT (NC UMENT OR AN 'OV' (OVERFLOW) ERRC E POLYNOMIAL COEFFICIENTS ARE MC PLICATION AT THE START OF THE EV #L84C4 LB8CA LBC2F FPØEXP #\$88 LB8C6 LBB5C LINT CHARAC	EIGHT COEFFICIENTS 2.14987637E-95: 1/(7!*(CF**7)) 1.4352314E-04: 1/(6!*(CF**6)) 1.34226348E-03: 1/(6!*(CF**5)) 9.61401701E-03: 1/(4!*(CF**4)) 0.0555051269: 1/(3!*(CF**3)) 0.240226385: 1/(2!*(CF**2)) 0.693147186: 1/(1!*(CF**1)) 1. RST MULTIPLYING THE THIS IS DONE, AN 0 UNDERFLOW) FOR A 0R FOR A POSITIVE DOIFIED TO REFLECT //ALUATION PROCESS. POINT X TO THE CORRECTION FACTOR MULTIPLY FA00 BY (X) PACK FPA00 AND STORE IT IN FPA3 *GET EXPONENT OF FPA0 AND *COMPARE TO THE MAXIMUM VALUE BRANCH IF FPA00 < 128 SET FPA00 = 0 OR OV ERROR CONVERT FPA00 TO INTEGER GET LS BYTE OF INTEGER
## ## ## ## ## ## ## ## ## ## ## ## ##	84C9 07 84CA 71 34 58 3E 56 84CF 74 16 7E B3 1B 84D4 77 2F EE E3 85 84D9 7A 1D 84 1C 2A 84DE 7C 63 59 58 0A 84E3 7E 75 FD E7 C6 84E8 80 31 72 18 10 84ED 81 00 00 00 00 84ED 81 00 00 20 00 84F2 8E 84 C4 84F5 8D 8A CA 84F8 BD 8C 2F 84FB 96 4F 84FD 81 88 84FF 25 03 8501 7E 8B 5C 8504 8D 8C EE	* TCHEBY * L84C9 L84C4 L84C9 L84C4 L84D4 L84D9 L84B8 L84E8 L84E8 L84E8 * * EXP (EX	YSHEV M FCB	0DIFIED TAYLOR SERIES COEFFICIEN \$07 \$71,\$34,\$58,\$3E,\$56 \$74,\$16,\$7E,\$83,\$1B \$77,\$2F,\$EE,\$E3,\$85 \$77,\$2F,\$EE,\$E3,\$85 \$77,\$19,\$84,\$1C,\$2A \$7C,\$63,\$59,\$58,\$0A \$7E,\$75,\$FD,\$E7,\$C6 \$80,\$31,\$72,\$18,\$10 \$81,\$00,\$00,\$00,\$00 IAL FUNCTION IS EVALUATED BY FIFA A CORRECTION FACTOR (CF). AFTER 127 WILL YIELD A ZERO RESULT (NC UMENT OR AN 'OV' COVERFLOW) ERR 127 WILL YIELD A ZERO RESULT (NC UMENT OR AN 'OV' COVERFLOW) ERR 128 POLYNOMIAL COEFFICIENTS ARE MC PLICATION AT THE START OF THE EV #L84C4 LBACA LBC2F FP0EXP #\$88 LB504 LBB5C INT CHARAC #\$81	EIGHT COEFFICIENTS 2.14987637E-Ø5: 1/(7!*(CF**7)) 1.4352314E-Ø4: 1/(6!*(CF**6)) 1.34226348E-Ø3: 1/(6!*(CF**5)) 9.6140179IE-Ø3: 1/(4!*(CF**4)) Ø.0555051269: 1/(3!*(CF**3)) Ø.240226385: 1/(2!*(CF**2)) Ø.693147186: 1/(1!*(CF**1)) 1. RST MULTIPLYING THE THIS IS DONE, AN 0 UNDERFLOW) FOR A NO UNDERFLOW) FOR A NO FOR A POSITIVE DOIFIED TO REFLECT VALUATION PROCESS. POINT X TO THE CORRECTION FACTOR MULTIPLY FPAØ BY (X) PACK FPAØ AND STORE IT IN FPA3 *GET EXPONENT OF FPAØ AND *COMPARE TO THE MAXIMUM VALUE BRANCH IF FPAØ < 128 SET FPAØ = Ø Ø R OV ERROR CONVERT FPAØ FO INTEGER GET LS BYTE OF INTEGER * WAS THE ARGUMENT =127, IF SO
0529 0530 0531 0532 0533 0534 0535 0536 0537 0548 0541 0542 0543 0544 0545 0546 0552 0552 0552 0556 0556 0557 0556	84C9 Ø7 84CF 74 16 7E B3 1B 84D4 77 2F EE E3 85 84D9 7A 10 84 1C 2A 84DE 7C 63 59 58 ØA 84E3 7E 75 FD E7 C6 84E8 8Ø 31 72 18 1Ø 84ED 81 ØØ ØØ ØØ ØØ 84F2 8E 84 C4 84F5 BD BA CA 84F8 BD BC 2F 84FP 81 88 84FF 25 Ø3 8507 7E 8B 5C 85Ø7 80 61	* TCHEBY * L84C9 L84C4 L84C9 L84C4 L84D4 L84D9 L84B8 L84E8 L84E8 L84E8 * * EXP (EX	YSHEV M FCB	0DIFIED TAYLOR SERIES COEFFICIEN \$07 \$71,\$34,\$58,\$3E,\$56 \$74,\$16,\$7E,\$83,\$1B \$77,\$2F,\$EE,\$E3,\$85 \$77,\$2F,\$EE,\$E3,\$85 \$77,\$19,\$84,\$1C,\$2A \$7C,\$63,\$59,\$58,\$0A \$7E,\$75,\$FD,\$E7,\$C6 \$80,\$31,\$72,\$18,\$10 \$81,\$00,\$00,\$00,\$00 IAL FUNCTION IS EVALUATED BY FIFA A CORRECTION FACTOR (CF). AFTER 127 WILL YIELD A ZERO RESULT (NC UMENT OR AN 'OV' COVERFLOW) ERR 127 WILL YIELD A ZERO RESULT (NC UMENT OR AN 'OV' COVERFLOW) ERR 128 POLYNOMIAL COEFFICIENTS ARE MC PLICATION AT THE START OF THE EV #L84C4 LBACA LBC2F FP0EXP #\$88 LB504 LBB5C INT CHARAC #\$81	EIGHT COEFFICIENTS 2.14987637E-95: 1/(7!*(CF**7)) 1.4352314E-04: 1/(6!*(CF**6)) 1.34226348E-03: 1/(6!*(CF**5)) 9.61401701E-03: 1/(4!*(CF**4)) 0.0555051269: 1/(3!*(CF**3)) 0.240226385: 1/(2!*(CF**2)) 0.693147186: 1/(1!*(CF**1)) 1. RST MULTIPLYING THE THIS IS DONE, AN 0 UNDERFLOW) FOR A 0R FOR A POSITIVE DOIFIED TO REFLECT //ALUATION PROCESS. POINT X TO THE CORRECTION FACTOR MULTIPLY FA00 BY (X) PACK FPA00 AND STORE IT IN FPA3 *GET EXPONENT OF FPA0 AND *COMPARE TO THE MAXIMUM VALUE BRANCH IF FPA00 < 128 SET FPA00 = 0 OR OV ERROR CONVERT FPA00 TO INTEGER GET LS BYTE OF INTEGER
9529 9539 9531 9532 9533 9534 9535 9536 9549 9549 9544 9545 9552 9552 9555 9556 9557 9559 9569 9569 9569	84C9 07 84C9 77 84CF 74 16 7E B3 1B 84D4 77 2F EE E3 85 84D9 7A 10 84 1C 2A 84DE 7C 63 59 58 0A 84E3 7E 75 FD E7 C6 84E8 80 31 72 18 10 84F5 8D 8A CA 84F5 8D 8A CA 84F8 8D 8C 2F 84FD 81 88 84FF 25 03 8501 7E 8B 5C 8507 96 01 8509 88 81 8500 4A	* TCHEBY * L84C9 L84C4 L84C9 L84C4 L84D4 L84D9 L84E3 L84E8 * * EXP (* THE EX * ARGUME * NEGATI * ARGUME * THE CF EXP	YSHEV M FCB	0DIFIED TAYLOR SERIES COEFFICIEN \$07 \$71,\$34,\$58,\$3E,\$56 \$74,\$16,\$7E,\$83,\$1B \$77,\$2F,\$EE,\$E3,\$85 \$77,\$2F,\$EE,\$E3,\$85 \$77,\$19,\$84,\$1C,\$2A \$7C,\$63,\$59,\$50,\$0A \$7E,\$75,\$FD,\$E7,\$C6 \$80,\$31,\$72,\$18,\$10 \$81,\$00,\$00,\$00,\$00 IAL FUNCTION IS EVALUATED BY FIFA A CORRECTION FACTOR (CF). AFTER 127 WILL YIELD A ZERO RESULT (NC UMENT OR AN '0V' (OVERFLOW) FRRE POLYNOMIAL COEFFICIENTS ARE MC PLICATION AT THE START OF THE EV #L84C4 LBACA LBC2F FP0EXP #\$88 LB504 LBB5C INT CHARAC ##\$81 LB501	EIGHT COEFFICIENTS 2.14987637E-Ø5: 1/(7!*(CF**7)) 1.4352314E-Ø4: 1/(6!*(CF**6)) 1.34226348E-Ø3: 1/(6!*(CF**5)) 9.6149179IE-Ø3: 1/(4!*(CF**4)) Ø.08555051269: 1/(3!*(CF**3)) Ø.240226385: 1/(2!*(CF**2)) Ø.693147186: 1/(1!*(CF**1)) 1. RST MULTIPLYING THE THIS IS DONE, AN OUNDERFLOW) FOR A OR FOR A POSITIVE DOIFIED TO REFLECT VALUATION PROCESS. POINT X TO THE CORRECTION FACTOR MULTIPLY FPAØ BY (X) PACK FPAØ AND STORE IT IN FPA3 **GET EXPONENT OF FPAØ AND *COMPARE TO THE MAXIMUM VALUE BRANCH IF FPAØ < 128 SET FPAØ = Ø OR OV ERROR CONVERT FPAØ TO INTEGER GET LS BYTE OF INTEGER GET LS BYTE OF INTEGER K ** MAS THE ARGUMENT =127, IF SO ** THEN OV ERROR; THIS WILL ALSO ADD THE \$8Ø BIAS ** REQUIRED WHEN THE NEW EXPONENT; BECAUSE \$81, NOT \$8Ø WAS USED ABOVE
## ## ## ## ## ## ## ## ## ## ## ## ##	84C9 07 84CA 71 34 58 3E 56 84CF 74 16 7E B3 1B 84D4 77 2F EE E3 85 84D9 7A 1D 84 1C 2A 84DE 7C 63 59 58 0A 84E3 7E 75 FD E7 C6 84E8 80 31 72 18 10 84ED 81 00 00 00 00 00 84ED 81 00 00 00 00 00 84ED 81 00 00 00 00 00 84F2 8E 84 C4 84F5 8D 8A CA 84F8 BD 8C 2F 84F8 96 4F 84F0 81 88 84FF 25 03 8501 7E 8B 5C 8504 8D 8C EE 8507 96 01 8509 8B 81 8508 27 F4 8500 4A 8500 4A	* TCHEBY * L84C9 L84C4 L84C9 L84C4 L84D4 L84D9 L84E3 L84E8 * * EXP (* THE EX * ARGUME * NEGATI * ARGUME * THE CF EXP	YSHEV M FCB	\$07 \$71,\$34,\$58,\$3E,\$56 \$77,\$71,\$34,\$58,\$3E,\$56 \$77,\$74,\$16,\$7E,\$83,\$1B \$77,\$2F,\$EE,\$E3,\$85 \$7A,\$1D,\$84,\$1C,\$2A \$7C,\$63,\$59,\$58,\$0A \$7E,\$75,\$FD,\$E7,\$C6 \$80,\$31,\$72,\$18,\$10 \$81,\$00,\$00,\$00 IAL FUNCTION IS EVALUATED BY FIFA CORRECTION FACTOR (CF). AFTER A CORRECTION FACTOR (CF). AFTER CY WILL VIELD A ZERO RESULT (NC UMENT OR AN 'OV' (OVERFLOW) ERRC E POLYNOMIAL COEFFICIENTS ARE MC PLICATION AT THE START OF THE EV #L84C4 LB8CA LBC2F FP0EXP #\$88 LB504 LBB5C INT CHARAC #\$81 LB501	EIGHT COEFFICIENTS 2.14987637E-Ø5: 1/(7!*(CF**7)) 1.4352314E-Ø4: 1/(6!*(CF**6)) 1.34226348E-Ø3: 1/(5!*(CF**5)) 9.614017Ø1E-Ø3: 1/(4!*(CF**5)) 9.614017Ø1E-Ø3: 1/(4!*(CF**4)) Ø.0555951269: 1/(3!*(CF**2)) Ø.693147186: 1/(2!*(CF**2)) Ø.693147186: 1/(1!*(CF**1)) 1. RST MULTIPLYING THE THIS IS DONE, AN OUNDERFLOW) FOR A OR FOR A POSITIVE DDIFIED TO REFLECT VALUATION PROCESS. POINT X TO THE CORRECTION FACTOR MULTIPLY FPAØ BY (X) PACK FPAØ AND STORE IT IN FPA3 *GET EXPONENT OF FPAØ AND *COMPARE TO THE MAXIMUM VALUE BRANCH IF FPAØ < 128 SET FPAØ = Ø OR OV ERROR CONVERT FPAØ TO INTEGER GET LS BYTE OF INTEGER * WAS THE ARGUMENT =127, IF SO * THEN OV ERROR; THIS WILL ALSO ADD THE \$8Ø BIAS * REQUIRED WHEN THE NEW EXPONENT IS CALCULATED BELOW DECREMENT ONE FROM THE EXPONENT, BECAUSE \$81, NOT \$8Ø WAS USED ABOVE SAVE EXPONENT OF INTEGER PORTION ON STACK
### ### ### ### ### ### ### ### ### ##	84C9 07 84CA 71 34 58 3E 56 84CF 74 16 7E B3 1B 84D4 77 2F EE E3 85 84D9 7A 1D 84 1C 2A 84DE 7C 63 59 58 0A 84E3 7E 75 FD E7 C6 84E8 80 31 72 18 10 84ED 81 00 00 00 00 00 84ED 81 00 00 00 00 00 84ED 81 00 00 00 00 00 84F8 8D BC 2F 84F8 96 4F 84F8 96 4F 84F8 96 4F 84F9 26 03 8501 7E B8 5C 8504 BD BC EE 8507 96 01 8509 88 81 8508 27 F4 8500 4A 850E 34 02 8501 86 00 40	* TCHEBY * L84C9 L84C4 L84C9 L84C4 L84D4 L84D9 L84E3 L84E8 * * EXP (* THE EX * ARGUME * NEGATI * ARGUME * THE CF EXP	YSHEV M FCB FCB FCB FCB FCB FCB FCB FCB FCB FC	\$07 \$71,\$34,\$58,\$3E,\$56 \$74,\$16,\$7E,\$83,\$1B \$77,\$2F,\$EE,\$E3,\$85 \$74,\$10,\$84,\$1C,\$2A \$77,\$72F,\$EE,\$E3,\$85 \$73,\$10,\$84,\$1C,\$2A \$76,\$75,\$F0,\$E7,\$C6 \$80,\$31,\$72,\$18,\$10 \$81,\$00,\$00,\$00,\$00 IAL FUNCTION IS EVALUATED BY FIFA A CORRECTION FACTOR (CF). AFTER 127 WILL YIELD A ZERO RESULT (NC UNENT OR AN 'OV' (OVERFLON) ERRC E POLYNOMIAL COEFFICIENTS ARE MC PLICATION AT THE START OF THE EV #L84C4 LB8CA LBC2F FPØEXP #\$88 LB850 LB850 INT CHARAC #\$81 L8501 A #V40	EIGHT COEFFICIENTS 2.14987637E-95: 1/(7!*(CF**7)) 1.4352314E-04: 1/(6!*(CF**6)) 1.34226348E-03: 1/(6!*(CF**5)) 9.61401701E-03: 1/(4!*(CF**4)) 0.0555051269: 1/(3!*(CF**3)) 0.240226385: 1/(2!*(CF**2)) 0.693147186: 1/(1!*(CF**1)) 1. RST MULTIPLYING THE THIS IS DONE, AN 0 UNDERFLOW) FOR A 0R FOR A POSITIVE DOIFIED TO REFLECT /ALUATION PROCESS. POINT X TO THE CORRECTION FACTOR MULTIPLY FA00 BY (X) PACK FPA00 AND STORE IT IN FPA3 *GET EXPONENT OF FPA0 AND *COMPARE TO THE MAXIMUM VALUE BRANCH IF FPA00 < 128 SET FPA00 = 0 OR OV ERROR CONVERT FPA00 TO INTEGER GET LS BYTE OF INTEGER ET LS BYTE OF INTEGER * WAS THE ARGUMENT =127, IF SO * THEN OV ERROR; THIS WILL ALSO ADD THE \$800 BIAS * REQUIRED WHEN THE NEW EXPONENT IS CALCULATED BELOW DECREMENT ONE FROM THE EXPONENT, BECAUSE \$81, NOT \$800 WAS USED ABOVE SAVE EXPONENT OF INTEGER PORTION ON STACK POINT (X) TO FPA3
## ## ## ## ## ## ## ## ## ## ## ## ##	84C9 07 84CA 71 34 58 3E 56 84CF 74 16 7E B3 1B 84D4 77 2F EE E3 85 84D9 7A 10 84 1C 2A 84DE 7C 63 59 58 0A 84E3 7E 75 FD E7 C6 84E8 80 31 72 18 10 84FD 81 00 00 00 00 84FD 81 00 00 00 00 84FD 81 88 84 C4 84FS 8D 8C 2F 84FD 81 88 84FF 25 03 8501 7E 8B 5C 8504 8D 8C EE 8507 96 01 8509 88 81 8508 27 F4 8500 4A 850E 34 02 8510 8E 00 40 88513 8D 8P 89	* TCHEBY * L84C9 L84C4 L84C9 L84C4 L84D4 L84D9 L84E3 L84E8 * * EXP (* THE EX * ARGUME * NEGATI * ARGUME * THE CF EXP	YSHEV M FCB FCB FCB FCB FCB FCB FCB FCB FCB FC	0DIFIED TAYLOR SERIES COEFFICIEN \$07 \$71,\$34,\$58,\$3E,\$56 \$74,\$16,\$7E,\$83,\$1B \$77,\$2F,\$EE,\$E3,\$85 \$77,\$2F,\$EE,\$E3,\$85 \$77,\$19,\$84,\$1C,\$2A \$7C,\$63,\$59,\$58,\$0A \$7E,\$75,\$FD,\$E7,\$C6 \$80,\$31,\$72,\$18,\$10 \$81,\$00,\$00,\$00,\$00 IAL FUNCTION IS EVALUATED BY FIR A CORRECTION FACTOR (CF). AFTER 127 WILL YIELD A ZERO RESULT (NC UMENT OR AN 'OV' (OVERFLOW) ERRC E POLYNOMIAL COEFFICIENTS ARE MC PULICATION AT THE START OF THE EV #L84C4 LBACA LBC2F FP0EXP #\$88 LB504 LBB5C INT CHARAC #\$81 L8501	EIGHT COEFFICIENTS 2.14987637E-Ø5: 1/(7!*(CF**7)) 1.4352314E-Ø4: 1/(6!*(CF**6)) 1.34226348E-Ø3: 1/(5!*(CF**5)) 9.6149179IE-Ø3: 1/(4!*(CF**4)) 0.0555051269: 1/(3!*(CF**3)) 0.240226385: 1/(2!*(CF**2)) 0.693147186: 1/(1!*(CF**1)) 1. RST MULTIPLYING THE THIS IS DONE, AN 0 UNDERFLOW) FOR A 0R FOR A POSITIVE DIFIED TO REFLECT VALUATION PROCESS. POINT X TO THE CORRECTION FACTOR MULTIPLY FPAØ BY (X) PACK FPAØ AND STORE IT IN FPA3 *SET FPAØ AND *COMPARE TO THE MAXIMUM VALUE BRANCH IF FPAØ < 128 SET FPAØ = Ø OR OV ERROR CONVERT FPAØ TO INTEGER GET LS BYTE OF INTEGER EY MAS THE ARGUMENT =127, IF SO * THEN OV ERROR; THIS WILL ALSO ADD THE \$8Ø BIAS * REQUIRED WHEN THE NEW EXPONENT IS CALCULATED BELOW DECREMENT ONE FROM THE EXPONENT, BECAUSE \$81, NOT \$8Ø WAS USED ABOVE SAVE EXPONENT OF INTEGER PORTION ON STACK POINT (X) TO FPAØ
## ## ## ## ## ## ## ## ## ## ## ## ##	84C9 07 84CA 71 34 58 3E 56 84CF 74 16 7E B3 1B 84D4 77 2F EE E3 85 84D9 7A 1D 84 1C 2A 84DE 7C 63 59 58 0A 84E3 7E 75 FD E7 C6 84E8 80 31 72 18 10 84ED 81 00 00 00 00 00 84ED 81 00 00 00 00 00 84ED 81 00 EE 84F2 8E 84 C4 84F5 8D 8A CA 84F8 8D 8C 2F 84F8 96 4F 84F0 81 88 84FF 25 03 8501 7E 8B 5C 8504 8D 8C EE 8507 96 01 8509 8B 81 8508 27 F4 8500 4A 850E 34 02 8513 8D 89 89 8516 8E 84 C9	* TCHEBY * L84C9 L84C4 L84C9 L84C4 L84D4 L84D9 L84E3 L84E8 * * EXP (* THE EX * ARGUME * NEGATI * ARGUME * THE CF EXP	YSHEV M FCB FCB FCB FCB FCB FCB FCB FCB FCB FC	\$07 \$71,\$34,\$58,\$3E,\$56 \$77,\$71,\$34,\$58,\$3E,\$56 \$77,\$74,\$16,\$7E,\$B3,\$1B \$77,\$2F,\$EE,\$E3,\$85 \$7A,\$1D,\$84,\$1C,\$2A \$7C,\$63,\$59,\$58,\$0A \$7E,\$75,\$FD,\$E7,\$C6 \$80,\$31,\$72,\$18,\$10 \$81,\$00,\$00,\$00 IAL FUNCTION IS EVALUATED BY FIFA A CORRECTION FACTOR (CF). AFTER A CORRECTION FACTOR (CF). AFTER COMMENT OR AN 'OV' (OVERFLOW) ERRCE POLYNOMIAL COEFFICIENTS ARE MC PLICATION AT THE START OF THE EV #L84C4 LBBCA LBC2F FP0EXP #\$88 LB504 LBB5C INT CHARAC #\$81 LB501 A #V40 LB9B9 #L84C9	EIGHT COEFFICIENTS 2.14987637E-05: 1/(7!*(CF**7)) 1.4352314E-04: 1/(6!*(CF**6)) 1.34226348E-03: 1/(5!*(CF**5)) 9.61401701E-03: 1/(4!*(CF**5)) 9.61401701E-03: 1/(4!*(CF**4)) 0.0555951269: 1/(3!*(CF**2)) 0.0455951269: 1/(3!*(CF**2)) 0.693147186: 1/(1!*(CF**2)) 0.693147186: 1/(1!*(CF**1)) 1. RST MULTIPLYING THE THIS IS DONE, AN OUNDERFLOW) FOR A OR FOR A POSITIVE DOIFIED TO REFLECT VALUATION PROCESS. POINT X TO THE CORRECTION FACTOR MULTIPLY FPAØ BY (X) PACK FPAØ AND STORE IT IN FPA3 *GET EXPONENT OF FPAØ AND *COMPARE TO THE MAXIMUM VALUE BRANCH IF FPAØ 7 128 SET FPAØ = 0 OR OV ERROR CONVERT FPAØ TO INTEGER GET LS BYTE OF INTEGER * WAS THE ARGUMENT =127, IF SO * THEN OV ERROR; THIS WILL ALSO ADD THE \$80 BIAS * REQUIRED WHEN THE NEW EXPONENT IS CALCULATED BELOW DECREMENT ONE FROM THE EXPONENT, BECAUSE \$81, NOT \$80 WAS USED ABOVE SAVE EXPONENT OF INTEGER PORTION ON STACK POINT (X) TO FPA3 SUBTRACT FPAØ FROM (X) - GET FRACTIONAL PART OF ARGUMENT POINT X TO COEFFICIENTS
## ## ## ## ## ## ## ## ## ## ## ## ##	84C9 07 84CA 71 34 58 3E 56 84CF 74 16 7E B3 1B 84D4 77 2F EE E3 85 84D9 7A 10 84 1C 2A 84DE 7C 63 59 58 0A 84E3 7E 75 FD E7 C6 84E8 80 31 72 18 10 84FD 81 00 00 00 00 84FD 81 00 00 00 00 84FD 81 88 84 C4 84FS 8D 8C 2F 84FD 81 88 84FF 25 03 8501 7E 8B 5C 8504 8D 8C EE 8507 96 01 8509 88 81 8508 27 F4 8500 4A 850E 34 02 8510 8E 00 40 88513 8D 8P 89	* TCHEBY * L84C9 L84C4 L84C9 L84C4 L84D4 L84D9 L84E3 L84E8 * * EXP (* THE EX * ARGUME * NEGATI * ARGUME * THE CF EXP	YSHEV M FCB FCB FCB FCB FCB FCB FCB FCB FCB FC	0DIFIED TAYLOR SERIES COEFFICIEN \$07 \$71,\$34,\$58,\$3E,\$56 \$74,\$16,\$7E,\$83,\$1B \$77,\$2F,\$EE,\$E3,\$85 \$73,\$10,\$84,\$1C,\$2A \$76,\$75,\$F0,\$E7,\$66 \$80,\$31,\$72,\$18,\$10 \$81,\$00,\$00,\$00,\$00 IAL FUNCTION IS EVALUATED BY FIFA A CORRECTION FACTOR (CF). AFTER 127 WILL YIELD A ZERO RESULT (NC UNENT OR AN 'OV' (OVERFLON) ERRC E POLYNOMIAL COEFFICIENTS ARE MC PLICATION AT THE START OF THE EV #L84C4 L86C4 FF0EXP #\$88 L8504 LBB5C INT CHARAC #\$81 L8501 A #V40 LB9B9 #L84C9 LBEFF	EIGHT COEFFICIENTS 2.14987637E-Ø5: 1/(7!*(CF**7)) 1.4352314E-Ø4: 1/(6!*(CF**6)) 1.34226348E-Ø3: 1/(5!*(CF**5)) 9.6149179IE-Ø3: 1/(4!*(CF**4)) 0.0555051269: 1/(3!*(CF**3)) 0.240226385: 1/(2!*(CF**2)) 0.693147186: 1/(1!*(CF**1)) 1. RST MULTIPLYING THE THIS IS DONE, AN 0 UNDERFLOW) FOR A 0R FOR A POSITIVE DIFIED TO REFLECT VALUATION PROCESS. POINT X TO THE CORRECTION FACTOR MULTIPLY FPAØ BY (X) PACK FPAØ AND STORE IT IN FPA3 *SET FPAØ AND *COMPARE TO THE MAXIMUM VALUE BRANCH IF FPAØ < 128 SET FPAØ = Ø OR OV ERROR CONVERT FPAØ TO INTEGER GET LS BYTE OF INTEGER EY MAS THE ARGUMENT =127, IF SO * THEN OV ERROR; THIS WILL ALSO ADD THE \$8Ø BIAS * REQUIRED WHEN THE NEW EXPONENT IS CALCULATED BELOW DECREMENT ONE FROM THE EXPONENT, BECAUSE \$81, NOT \$8Ø WAS USED ABOVE SAVE EXPONENT OF INTEGER PORTION ON STACK POINT (X) TO FPAØ
### ### ### ### ### ### ### ### ### ##	84C9 07 84CA 71 34 58 3E 56 84CF 74 16 7E B3 1B 84D4 77 2F EE E3 85 84D9 7A 1D 84 1C 2A 84DE 7C 63 59 58 0A 84E3 7E 75 FD E7 C6 84E8 80 31 72 18 10 84ED 81 00 00 00 00 00 84ED 81 00 00 00 00 84ED 81 00 00 00 00 84F5 BD BA CA 84F8 BD BC 2F 84F8 96 4F 84F9 86 88 88 88 88 88 88 88 88 88 88 88 88	* TCHEBY * L84C9 L84C4 L84C9 L84C4 L84D4 L84D9 L84E3 L84E8 * * EXP (* THE EX * ARGUME * NEGATI * ARGUME * THE CF EXP	YSHEV M FCB FCB FCB FCB FCB FCB FCB FCB FCB FC	0DIFIED TAYLOR SERIES COEFFICIEN \$07 \$71,\$34,\$58,\$3E,\$56 \$74,\$16,\$7E,\$83,\$1B \$77,\$2F,\$EE,\$E3,\$85 \$77,\$2F,\$EE,\$E3,\$85 \$77,\$19,\$84,\$10,\$2A \$7C,\$63,\$59,\$58,\$8A \$7E,\$75,\$FD,\$E7,\$C6 \$80,\$31,\$72,\$18,\$10 \$81,\$00,\$00,\$00,\$00 IAL FUNCTION IS EVALUATED BY FIR A CORRECTION FACTOR (CF). AFTER A CORRECTION FACTOR (CF). AFTER EPOLYNOMIAL COEFFICIENTS ARE MC PLICATION AT THE START OF THE EV #L84C4 LBACA LBC2F FPPEXP #\$88 LB504 LBB5C INT CHARAC #\$81 L8501 A #V40 LB9B9 #L84C9 LBEFF RESSGN	EIGHT COEFFICIENTS 2.14987637E-05: 1/(7!*(CF**7)) 1.4352314E-04: 1/(6!*(CF**6)) 1.34226348E-03: 1/(6!*(CF**5)) 9.61401701E-03: 1/(4!*(CF**4)) 0.0555051269: 1/(3!*(CF**3)) 0.240226385: 1/(2!*(CF**2)) 0.693147186: 1/(1!*(CF**1)) 1. RST MULTIPLYING THE THIS IS DONE, AN 0 UNDERFLOW) FOR A 0R FOR A POSITIVE DOIFIED TO REFLECT //ALUATION PROCESS. POINT X TO THE CORRECTION FACTOR MULTIPLY FPAØ BY (X) PACK FPAØ AND STORE IT IN FPA3 *GET EXPONENT OF FPAØ AND *COMPARE TO THE MAXIMUM VALUE BRANCH IF FPAØ < 128 SET FPAØ = Ø OR OV ERROR CONVERT FPAØ TO INTEGER GET LS BYTE OF INTEGER * WAS THE ARGUMENT =127, IF SO * THEN OV ERROR; THIS WILL ALSO ADD THE \$8Ø BIAS * REQUIRED WHEN THE NEW EXPONENT IS CALCULATED BELOW DECREMENT ONE FROM THE EXPONENT, BECAUSE \$81, NOT \$8Ø WAS USED ABOVE SAVE EXPONENT OF INTEGER PORTION ON STACK POINT (X) TO FPA3 SUBTRACT FPAØ FROM (X) - GET FRACTIONAL PART OF ARGUMENT POINT X TO COEFFICIENTS EVALUATE POLYNOMIAL FOR FRACTIONAL PART
### ### ### ### ### ### ### ### ### ##	84C9 07 84CA 71 34 58 3E 56 84CF 74 16 7E B3 1B 84D4 77 2F EE E3 85 84D9 7A 1D 84 1C 2A 84DE 7C 63 59 58 0A 84E3 7E 75 FD E7 C6 84E8 80 31 72 18 10 84ED 81 00 00 00 00 84F2 8E 84 C4 84F5 8D BA CA 84F8 8D BC 2F 84FD 81 88 84FF 25 03 8501 7E 8B 5C 8504 BD BC EE 8507 96 01 8509 8B 81 8508 27 F4 8500 4A 850E 34 02 8510 8E 00 40 8513 BD 89 8516 8E 84 C9 8519 BD BE FF	* TCHEBY * L84C9 L84C9 L84C9 L84C9 L84D4 L84D9 L84B3 L84E8 * * EXP (* THE EX * ARGUME * NEGATI * ARGUME * THE CF EXP L8501 L8504	YSHEV M FCB FCB FCB FCB FCB FCB FCB FCB FCB FC	\$07 \$71,\$34,\$58,\$3E,\$56 \$77,\$74,\$16,\$7E,\$B3,\$1B \$77,\$2F,\$EE,\$E3,\$85 \$77A,\$10,\$84,\$1C,\$2A \$77,\$63,\$59,\$58,\$0A \$75,\$75,\$FD,\$E7,\$C6 \$80,\$31,\$72,\$18,\$10 \$81,\$00,\$00,\$00 IAL FUNCTION IS EVALUATED BY FIFA A CORRECTION FACTOR (CF). AFTER A CORRECTION FACTOR (CF). AFTER COMMENT OR AN 'OV' (OVERFLOW) ERRCE POLYNOMIAL COEFFICIENTS ARE MC PLICATION AT THE START OF THE EV #L84C4 LBBC2F FP0EXP #\$88 L8504 LBB50 INT CHARAC #\$81 LB501 A #V40 LB9B9 #L84C9 LBFF RESSGN A	EIGHT COEFFICIENTS 2.14987637E-95: 1/(7!*(CF**7)) 1.4352314E-04: 1/(6!*(CF**6)) 1.34226348E-03: 1/(6!*(CF**5)) 9.61401701E-03: 1/(4!*(CF**4)) 0.0555051269: 1/(3!*(CF**3)) 0.240226385: 1/(2!*(CF**2)) 0.693147186: 1/(1!*(CF**1)) 1. RST MULTIPLYING THE THIS IS DONE, AN 0 UNDERFLOW) FOR A 0R FOR A POSITIVE DOIFIED TO REFLECT //ALUATION PROCESS. POINT X TO THE CORRECTION FACTOR MULTIPLY FPAØ BY (X) PACK FPAØ AND STORE IT IN FPA3 *GET EXPONENT OF FPAØ AND *COMPARE TO THE MAXIMUM VALUE BRANCH IF FPAØ < 128 SET FPAØ = Ø OR OV ERROR CONVERT FPAØ TO INTEGER GET LS BYTE OF INTEGER * WAS THE ARGUMENT =127, IF SO * THEN OV ERROR; THIS WILL ALSO ADD THE \$8Ø BIAS * REQUIRED WHEN THE NEW EXPONENT IS CALCULATED BELOW DECREMENT ONE FROM THE EXPONENT, BECAUSE \$81, NOT \$8Ø WAS USED ABOVE SAVE EXPONENT OF INTEGER PORTION ON STACK POINT (X) TO FPA3 SUBTRACT FPAØ FROM (X) - GET FRACTIONAL PART OF ARGUMENT FORCE THE MANTISSA TO BE POSITIVE GET INTEGER EXPONENT FOR MSTACK * CALCULATE EXPONENT FOR STACK * CALCULATE EXPONENT FOR WE FPAØ BY ADDING THE EXPONENTS OF THE
## ## ## ## ## ## ## ## ## ## ## ## ##	84C9 07 84CA 71 34 58 3E 56 84CF 74 16 7E B3 1B 84D4 77 2F EE E3 85 84D9 7A 1D 84 1C 2A 84DE 7C 63 59 58 0A 84E3 7E 75 FD E7 C6 84E8 80 31 72 18 10 84ED 81 00 00 00 00 84F2 8E 84 C4 84F5 8D 8A CA 84F8 8D 8C 2F 84F9 81 88 84F7 25 03 8501 7E 8B 5C 8504 8D 8C EE 8507 96 01 8509 8B 81 8508 27 F4 8500 4A 850E 34 02 8510 8E 00 40 8513 8D 89 89 8516 8E 84 C9 8519 8D 8E FF 851C 0F 62 851E 35 02 8520 8D 8B 48	* TCHEBY * L84C9 L84C4 L84C9 L84C4 L84D4 L84D9 L84E3 L84E8 * * EXP (* THE EX * ARGUME * NEGATI * ARGUME * THE CF EXP	YSHEV M FCB	\$07 \$71,\$34,\$58,\$3E,\$56 \$77,\$74,\$16,\$7E,\$B3,\$1B \$77,\$2F,\$EE,\$E3,\$85 \$77A,\$10,\$84,\$1C,\$2A \$77,\$63,\$59,\$58,\$0A \$75,\$75,\$FD,\$E7,\$C6 \$80,\$31,\$72,\$18,\$10 \$81,\$00,\$00,\$00 IAL FUNCTION IS EVALUATED BY FIFA A CORRECTION FACTOR (CF). AFTER A CORRECTION FACTOR (CF). AFTER COMMENT OR AN 'OV' (OVERFLOW) ERRCE POLYNOMIAL COEFFICIENTS ARE MC PLICATION AT THE START OF THE EV #L84C4 LBBC2F FP0EXP #\$88 L8504 LBB50 INT CHARAC #\$81 LB501 A #V40 LB9B9 #L84C9 LBFF RESSGN A	EIGHT COEFFICIENTS 2.14987637E-05: 1/(7!*(CF**7)) 1.4352314E-04: 1/(6!*(CF**6)) 1.34226348E-03: 1/(5!*(CF**5)) 9.61401701E-03: 1/(4!*(CF**5)) 9.61401701E-03: 1/(4!*(CF**4)) 0.0555951269: 1/(3!*(CF**2)) 0.240226335: 1/(2!*(CF**2)) 0.693147186: 1/(1!*(CF**1)) 1. RST MULTIPLYING THE THIS IS DONE, AN 0 UNDERFLOW) FOR A OR FOR A POSITIVE DOIFIED TO REFLECT VALUATION PROCESS. POINT X TO THE CORRECTION FACTOR MULTIPLY FPAØ BY (X) PACK FPAØ AND STORE IT IN FPA3 *GET EXPONENT OF FPAØ AND *COMPARE TO THE MAXIMUM VALUE BRANCH IF FPAØ < 128 SET FPAØ = 0 OR OV ERROR CONVERT FPAØ TO INTEGER GET LS BYTE OF INTEGER * WAS THE ARGUMENT =127, IF SO * THEN OV ERROR; THIS WILL ALSO ADD THE \$80 BIAS * REQUIRED WHEN THE NEW EXPONENT IS CALCULATED BELOW DECREMENT ONE FROM THE EXPONENT, BECAUSE \$81, NOT \$80 WAS USED ABOVE SAVE EXPONENT OF INTEGER PORTION ON STACK POINT (X) TO FPA3 SUBTRACT FPAØ FROM (X) - GET FRACTIONAL PART OF ARGUMENT POINT X TO COEFFICIENTS EVALUATE POLYNOMIAL FOR FRACTIONAL PART FORCE THE MANTISSA TO BE POSITIVE GET INTEGER EXPONENT FROM STACK
## ## ## ## ## ## ## ## ## ## ## ## ##	84C9 07 84CA 71 34 58 3E 56 84CF 74 16 7E B3 1B 84D4 77 2F EE E3 85 84D9 7A 1D 84 1C 2A 84DE 7C 63 59 58 0A 84E3 7E 75 FD E7 C6 84E8 80 31 72 18 10 84ED 81 00 00 00 00 00 84ED 80 2F 84F2 8E 84 C4 84F5 8D 8A CA 84F8 8D 8C 2F 84F8 8D 8C 2F 84F8 96 4F 84F9 25 03 8501 7E 8B 5C 8504 8D 8C EE 8507 96 01 8509 8B 81 8508 27 F4 8500 4A 8508 27 F4 8500 4A 8508 34 02 8513 8D 89 89 8516 8E 00 40 8513 8D 89 89 8516 8E 84 C9 8510 0F 62	* TCHEBY * L84C9 L84C9 L84C9 L84C9 L84D4 L84D9 L84B3 L84E8 * * EXP (* THE EX * ARGUME * NEGATI * ARGUME * THE CF EXP L8501 L8504	YSHEV M FCB FCB FCB FCB FCB FCB FCB FCB FCB FC	\$07 \$71,\$34,\$58,\$3E,\$56 \$77,\$74,\$16,\$7E,\$B3,\$1B \$77,\$2F,\$EE,\$E3,\$85 \$77A,\$10,\$84,\$1C,\$2A \$77,\$63,\$59,\$58,\$0A \$75,\$75,\$FD,\$E7,\$C6 \$80,\$31,\$72,\$18,\$10 \$81,\$00,\$00,\$00 IAL FUNCTION IS EVALUATED BY FIFA A CORRECTION FACTOR (CF). AFTER A CORRECTION FACTOR (CF). AFTER COMMENT OR AN 'OV' (OVERFLOW) ERRCE POLYNOMIAL COEFFICIENTS ARE MC PLICATION AT THE START OF THE EV #L84C4 LBBC2F FP0EXP #\$88 L8504 LBB50 INT CHARAC #\$81 LB501 A #V40 LB9B9 #L84C9 LBFF RESSGN A	EIGHT COEFFICIENTS 2.14987637E-95: 1/(7!*(CF**7)) 1.4352314E-04: 1/(6!*(CF**6)) 1.34226348E-03: 1/(6!*(CF**5)) 9.61401701E-03: 1/(4!*(CF**4)) 0.0555051269: 1/(3!*(CF**3)) 0.240226385: 1/(2!*(CF**2)) 0.693147186: 1/(1!*(CF**1)) 1. RST MULTIPLYING THE THIS IS DONE, AN 0 UNDERFLOW) FOR A 0R FOR A POSITIVE DOIFIED TO REFLECT //ALUATION PROCESS. POINT X TO THE CORRECTION FACTOR MULTIPLY FPAØ BY (X) PACK FPAØ AND STORE IT IN FPA3 *GET EXPONENT OF FPAØ AND *COMPARE TO THE MAXIMUM VALUE BRANCH IF FPAØ < 128 SET FPAØ = Ø OR OV ERROR CONVERT FPAØ TO INTEGER GET LS BYTE OF INTEGER * WAS THE ARGUMENT =127, IF SO * THEN OV ERROR; THIS WILL ALSO ADD THE \$8Ø BIAS * REQUIRED WHEN THE NEW EXPONENT IS CALCULATED BELOW DECREMENT ONE FROM THE EXPONENT, BECAUSE \$81, NOT \$8Ø WAS USED ABOVE SAVE EXPONENT OF INTEGER PORTION ON STACK POINT (X) TO FPA3 SUBTRACT FPAØ FROM (X) - GET FRACTIONAL PART OF ARGUMENT FORCE THE MANTISSA TO BE POSITIVE GET INTEGER EXPONENT FOR MSTACK * CALCULATE EXPONENT FOR STACK * CALCULATE EXPONENT FOR WE FPAØ BY ADDING THE EXPONENTS OF THE
## ## ## ## ## ## ## ## ## ## ## ## ##	84C9 07 84CA 71 34 58 3E 56 84CF 74 16 7E B3 1B 84D4 77 2F EE E3 85 84D9 7A 1D 84 1C 2A 84DE 7C 63 59 58 0A 84E3 7E 75 FD E7 C6 84E8 80 31 72 18 10 84ED 81 00 00 00 00 84F2 8E 84 C4 84F5 8D 8A CA 84F8 8D 8C 2F 84F9 81 88 84F7 25 03 8501 7E 8B 5C 8504 8D 8C EE 8507 96 01 8509 8B 81 8508 27 F4 8500 4A 850E 34 02 8510 8E 00 40 8513 8D 89 89 8516 8E 84 C9 8519 8D 8E FF 851C 0F 62 851E 35 02 8520 8D 8B 48	* TCHEBY * L84C9 L84C4 L84CF L84D4 L84D9 L84E3 L84E8 * * EXP (* THE EX* * ARGUME * NEGATI* * THE CF EXP L8501 L8504 *	YSHEV M FCB	\$07 \$71,\$34,\$58,\$3E,\$56 \$77,\$74,\$16,\$7E,\$B3,\$1B \$77,\$2F,\$EE,\$E3,\$85 \$77A,\$10,\$84,\$1C,\$2A \$77,\$63,\$59,\$58,\$0A \$75,\$75,\$FD,\$E7,\$C6 \$80,\$31,\$72,\$18,\$10 \$81,\$00,\$00,\$00 IAL FUNCTION IS EVALUATED BY FIFA A CORRECTION FACTOR (CF). AFTER A CORRECTION FACTOR (CF). AFTER COMMENT OR AN 'OV' (OVERFLOW) ERRCE POLYNOMIAL COEFFICIENTS ARE MC PLICATION AT THE START OF THE EV #L84C4 LBBC2F FP0EXP #\$88 L8504 LBB50 INT CHARAC #\$81 LB501 A #V40 LB9B9 #L84C9 LBFF RESSGN A	EIGHT COEFFICIENTS 2.14987637E-95: 1/(7!*(CF**7)) 1.4352314E-04: 1/(6!*(CF**6)) 1.34226348E-03: 1/(6!*(CF**5)) 9.61401701E-03: 1/(4!*(CF**4)) 0.0555051269: 1/(3!*(CF**3)) 0.240226385: 1/(2!*(CF**2)) 0.693147186: 1/(1!*(CF**1)) 1. RST MULTIPLYING THE THIS IS DONE, AN 0 UNDERFLOW) FOR A 0R FOR A POSITIVE DOIFIED TO REFLECT //ALUATION PROCESS. POINT X TO THE CORRECTION FACTOR MULTIPLY FPAØ BY (X) PACK FPAØ AND STORE IT IN FPA3 *GET EXPONENT OF FPAØ AND *COMPARE TO THE MAXIMUM VALUE BRANCH IF FPAØ < 128 SET FPAØ = Ø OR OV ERROR CONVERT FPAØ TO INTEGER GET LS BYTE OF INTEGER * WAS THE ARGUMENT =127, IF SO * THEN OV ERROR; THIS WILL ALSO ADD THE \$8Ø BIAS * REQUIRED WHEN THE NEW EXPONENT IS CALCULATED BELOW DECREMENT ONE FROM THE EXPONENT, BECAUSE \$81, NOT \$8Ø WAS USED ABOVE SAVE EXPONENT OF INTEGER PORTION ON STACK POINT (X) TO FPA3 SUBTRACT FPAØ FROM (X) - GET FRACTIONAL PART OF ARGUMENT FORCE THE MANTISSA TO BE POSITIVE GET INTEGER EXPONENT FOR MSTACK * CALCULATE EXPONENT FOR STACK * CALCULATE EXPONENT FOR WE FPAØ BY ADDING THE EXPONENTS OF THE
9529 95301 9531 9532 9533 9534 9535 9536 9549 9544 9544 9544 9544 9544 9544 954	84C9 07 84CA 71 34 58 3E 56 84CF 74 16 7E B3 1B 84D4 77 2F EE E3 85 84D9 7A 1D 84 1C 2A 84DE 7C 63 59 58 0A 84E3 7E 75 FD E7 C6 84E8 80 31 72 18 10 84ED 81 00 00 00 00 84ED 81 00 00 00 00 84ED 81 80 80 C2 84F5 8D 8A CA 84F8 8D 8C 2F 84F8 8D 8C 2F 84F9 81 88 84F7 25 03 8501 7E 88 5C 8504 8D 8C EE 8507 96 01 8509 8B 81 8509 8 81 8509 8 7 F4 8500 4A 8508 27 F4 8500 4A 8508 34 02 8513 8D 99 89 8516 8E 84 C9 8519 8D 8E FF 8510 07 662 8510 8D 8B 48 8523 39	* TCHEBY * L84C9 L84C4 L84C9 L84C4 L84D4 L84D9 L84E3 L84E8 * EXP (* THE EX * ARGUME * NEGATI * ARGUME * THE CF EXP L8501 L8504 * * FIX	YSHEV M FCB	\$07 \$71,\$34,\$58,\$3E,\$56 \$74,\$16,\$7E,\$83,\$1B \$77,\$2F,\$EE,\$E3,\$85 \$77,\$2F,\$EE,\$E3,\$85 \$77,\$10,\$84,\$10,\$2A \$7C,\$63,\$59,\$50,\$6A \$7E,\$75,\$FD,\$E7,\$C6 \$80,\$31,\$72,\$18,\$10 \$81,\$00,\$00,\$00,\$00 IAL FUNCTION IS EVALUATED BY FIFA CORRECTION FACTOR (CF). AFTER TAX TO THE PULL TO THE POLYNOMIAL COEFFICIENTS ARE MC WHARACC HEACA HEACA HEACA HEBBEC INT CHARACC #\$81 LB501 A #V40 LB9B9 #B44C9 LBEFF RESSECN A LBB48	EIGHT COEFFICIENTS 2.14987637E-Ø5: 1/(7!*(CF**7)) 1.4352314E-Ø4: 1/(6!*(CF**6)) 1.34226348E-Ø3: 1/(5!*(CF**5)) 9.614Ø17Ø1E-Ø3: 1/(4!*(CF**4)) 0.0555051269: 1/(3!*(CF**4)) 0.0555051269: 1/(3!*(CF**2)) 0.693147186: 1/(1!*(CF**1)) 1. RST MULTIPLYING THE THIS IS DONE, AN DUNDERFLOW) FOR A DR FOR A POSITIVE DUIFIED TO REFLECT VALUATION PROCESS. POINT X TO THE CORRECTION FACTOR MULTIPLY FPAØ BY (X) PACK FPAØ AND STORE IT IN FPA3 *GET EXPONENT OF FPAØ AND *COMPARE TO THE MAXIMUM VALUE BRANCH IF FPAØ < 128 SET FPAØ = Ø OR OV ERROR CONVERT FPAØ TO INTEGER GET LS BYTE OF INTEGER K * MAS THE ARGUMENT =127, IF SO * THEN OV ERROR; THIS WILL ALSO ADD THE \$8Ø BIAS * REQUIRED WHEN THE NEW EXPONENT IS CALCULATED BELOW DECREMENT ONE FROM THE EXPONENT, BECAUSE \$81, NOT \$8Ø WAS USED ABOVE SAVE EXPONENT OF FPAØ SUBTRACT FPAØ FROM (X) - GET FRACTIONAL PART OF ARGUMENT POINT X TO COEFFICIENTS EVALUATE POLYNOMIAL FOR FRACTIONAL PART FORCE THE MANTISSA TO BE POSITIVE GET INTEGER EXPONENT FROM STACK * CALCULATE EXPONENT OF NEW FPAØ BY ADDING THE EXPONENTS OF THE * INTEGER AND FRACTIONAL PARTS
## ## ## ## ## ## ## ## ## ## ## ## ##	84C9 07 84CA 71 34 58 3E 56 84CF 74 16 7E B3 1B 84D4 77 2F EE E3 85 84D9 7A 1D 84 1C 2A 84DE 7C 63 59 58 0A 84E3 7E 75 FD E7 C6 84E8 80 31 72 18 10 84ED 81 00 00 00 00 84F2 8E 84 C4 84F5 8D 8A CA 84F8 8D 8C 2F 84F9 81 88 84F7 25 03 8501 7E 8B 5C 8504 8D 8C EE 8507 96 01 8509 8B 81 8508 27 F4 8500 4A 850E 34 02 8510 8E 00 40 8513 8D 89 89 8516 8E 84 C9 8519 8D 8E FF 851C 0F 62 851E 35 02 8520 8D 8B 48	* TCHEBY * L84C9 L84C4 L84CF L84D4 L84D9 L84E3 L84E8 * * EXP (* THE EX* * ARGUME * NEGATI* * THE CF EXP L8501 L8504 *	YSHEV M FCB	## STAND SERIES COEFFICIENT ## STAND SERIES COEFFICIENT ## STAND SERIES SERIES COEFFICIENT ## STAND SERIES SERIES SERIES SERIES STAND SERIES SERIE	EIGHT COEFFICIENTS 2.14987637E-95: 1/(7!*(CF**7)) 1.4352314E-04: 1/(6!*(CF**6)) 1.34226348E-03: 1/(6!*(CF**5)) 9.61401701E-03: 1/(4!*(CF**4)) 0.0555051269: 1/(3!*(CF**3)) 0.240226385: 1/(2!*(CF**2)) 0.693147186: 1/(1!*(CF**1)) 1. RST MULTIPLYING THE THIS IS DONE, AN 0 UNDERFLOW) FOR A 0R FOR A POSITIVE DOIFIED TO REFLECT //ALUATION PROCESS. POINT X TO THE CORRECTION FACTOR MULTIPLY FPAØ BY (X) PACK FPAØ AND STORE IT IN FPA3 *GET EXPONENT OF FPAØ AND *COMPARE TO THE MAXIMUM VALUE BRANCH IF FPAØ < 128 SET FPAØ = Ø OR OV ERROR CONVERT FPAØ TO INTEGER GET LS BYTE OF INTEGER * WAS THE ARGUMENT =127, IF SO * THEN OV ERROR; THIS WILL ALSO ADD THE \$8Ø BIAS * REQUIRED WHEN THE NEW EXPONENT IS CALCULATED BELOW DECREMENT ONE FROM THE EXPONENT, BECAUSE \$81, NOT \$8Ø WAS USED ABOVE SAVE EXPONENT OF INTEGER PORTION ON STACK POINT (X) TO FPA3 SUBTRACT FPAØ FROM (X) - GET FRACTIONAL PART OF ARGUMENT FORCE THE MANTISSA TO BE POSITIVE GET INTEGER EXPONENT FOR MSTACK * CALCULATE EXPONENT FOR STACK * CALCULATE EXPONENT FOR WE FPAØ BY ADDING THE EXPONENTS OF THE

Ø577 Ø578	8529 7E BC EE 852C Ø3 54	L8529 L852C	JMP	INT FPØSGN	CONVERT FPAØ TO INTEGER TOGGLE SIGN OF FPAØ MANTISSA
Ø576	852E 8D F9	L032U		L8529	CONVERT FPAØ TO INTEGER
Ø58Ø	853Ø 7E BE E9			LBEE9	TOGGLE SIGN OF FPAØ
Ø581					
Ø582		* EDIT			
Ø583		EDIT		L89AE	GET LINE NUMBER FROM BASIC
Ø584 Ø585	8536 32 62 8538 86 Ø1	L8538	LEAS	\$02,S	PURGE RETURN ADDRESS OFF OF THE STACK LIST FLAG
Ø586	853A 97 D8	L0330	STA	VD8	SET FLAG TO LIST LINE
Ø587	853C BD AD Ø1		JSR	#\$Ø1 VD8 LADØ1 LAED2 LB7C2 Y,D #LINBUF+2 VD7	GO FIND THE LINE NUMBER IN PROGRAM
Ø588	853F 10 25 29 8F		LBCS	LAED2	ERROR #7 UNDEFINED LINE #'
Ø589	8543 BD B7 C2		JSR	LB7C2	GO UNCRUNCH LINE INTO BUFFER AT LINBUF+1
	8546 1F 2Ø		TFR	Y,D	PUT ABSOLUTE ADDRESS OF END OF LINE TO ACCD
	8548 83 Ø2 DE 854B D7 D7		SUBD	#LINBUF+2	SUBTRACT OUT THE START OF LINE SAVE LENGTH OF LINE
Ø592	854D DC 2B	L854D	316	RTNVAI	GET THE HEX VALUE OF LINE NUMBER
Ø594	854F BD BD CC	200.0	JSR	LBDCC	LIST THE LINE NUMBER ON THE SCREEN
Ø595	8552 BD B9 AC		JSR	LB9AC	PRINT A SPACE
Ø596	8555 8E Ø2 DD		LDX	#LINBUF+1	POINT X TO BUFFER
	8558 D6 D8		LDB	VD7 BINVAL LBDCC LB9AC #LINBUF+1 VD8 L8581	* CHECK TO SEE IF LINE IS TO BE
Ø598 Ø599	855A 26 25	LOEEC	RNF	L8581	* LISTED TO SCREEN - BRANCH IF IT IS RESET DIGIT ACCUMULATOR - DEFAULT VALUE
Ø6ØØ	855C 5F 855D BD 86 87	1855D	JSR	L8687	GET KEY STROKE
0601	8560 BD 90 AA	20005	JSR	L9ØAA	SET CARRY IF NOT NUMERIC
0602	8563 25 ØB		BLO	L857Ø	BRANCH IF NOT NUMERIC
0603	8565 80 30			#'0'	MASK OFF ASCII
0604	8567 34 Ø2 8569 86 ØA		PSHS		SAVE IT ON STACK
Ø6Ø5 Ø6Ø6	856B 3D		LDA MUL	#10	NUMBER BEING CONVERTED IS BASE 10 MULTIPLY ACCUMULATED VALUE BY BASE (10)
Ø6Ø7	856C EB EØ			,\$+	ADD DIGIT TO ACCUMULATED VALUE
Ø6Ø8				L855D	CHECK FOR ANOTHER DIGIT
0609		L857Ø			* REPEAT PARAMETER IN ACCB; IF IT
0610	8572 C9 Ø1			#\$01	*IS Ø, THEN MAKE IT 1
0611	8574 81 41 8576 26 Ø5			#'A'	ABORT?
Ø612	85/6 26 05 8578 BD B9 58			L857D	NO
Ø613 Ø614	857B 2Ø BB			LB958 L8538	PRINT CARRIAGE RETURN TO SCREEN RESTART EDIT PROCESS - CANCEL ALL CHANGES
		L857D		#'L'	LIST?
Ø616	857F 26 ØB			L858C	NO
Ø617	8581 8D 31	L8581	BSR	L85B4	LIST THE LINE
Ø618	8583 ØF D8		CLR	VD8	RESET THE LIST FLAG TO NO LIST
Ø619	8585 BD B9 58		JSR	LB958	PRINT CARRIAGE RETURN
	8588 20 C3 858A 32 62	L858A L858C	RKA	L854D \$02,S	GO INTERPRET ANOTHER EDIT COMMAND PURGE RETURN ADDRESS OFF OF THE STACK
Ø622	858C 81 ØD	L858C	CMPA	#CR	ENTER KEY?
	858E 26 ØD	20000	BNE	L859D	NO
0624	8590 RD 22		RSR	L85B4	ECHO THE LINE TO THE SCREEN
Ø625	8592 BD B9 58	L8592	JSR	LB958	PRINT CARRIAGE RETURN
Ø626	8595 8E Ø2 DD		LDX	#LINBUF+1	* RESET BASIC S INPUT POINTER
Ø627 Ø628	8598 9F A6 859A 7E AC A8		211	CHARAD LACA8	* TO THE LINE INPUT BUFFER GO PUT LINE BACK IN PROGRAM
Ø629		L859D		#'E'	EXIT?
Ø63Ø	859F 27 F1	20330		L8592	YES - SAME AS ENTER EXCEPT NO ECHO
Ø631	85A1 81 51			#'Q'	QUIT?
Ø632	85A3 26 Ø6			L85AB	NO
Ø633	85A5 BD B9 58			LB958	PRINT CARRIAGE RETURN TO SCREEN
Ø634 Ø635	85A8 7E AC 73 85AB 8D Ø2	L85AB		LAC73 L85AF	GO TO COMMAND LEVEL - MAKE NO CHANGES INTERPRET THE REMAINING COMMANDS AS SUBROUTINES
Ø636	85AD 20 AD	LOSAB		L855C	GO INTERPRET ANOTHER EDIT COMMAND
Ø637	85AF 81 20	L85AF		#SPACE	SPACE BAR?
Ø638	85B1 26 10		BNE	L85C3	NO
Ø639	85B3 8C	L85B3		SKP2	SKIP TWO BYTES
Ø64Ø		* DISPL	AY THE	NEXT ACCB BYTES OF THE LINE IN T	THE BUFFER TO THE SCREEN
Ø641	OEDA CE EO	* 10ED/	LDD	#I DIIEMY 1	SEW DALES MAN IN DIRECT
Ø642 Ø643	85B4 C6 F9 85B6 A6 84	L85B4 L85B6	LDB		250 BYTES MAX IN BUFFER GET A CHARACTER FROM BUFFER
Ø644	85B8 27 Ø8	20050		L85C2	EXIT IF IT S A Ø
Ø645	85BA BD A2 82		JSR	LA282	SEND CHAR TO CONSOLE OUT
Ø646	85BD 30 01			\$Ø1,X	MOVE POINTER UP ONE
Ø647	85BF 5A		DECB		DECREMENT CHARACTER COUNTER LOOP IF NOT DONE
Ø648 Ø649	85CØ 26 F4 85C2 39	L85C2	RTS	L85B6	LOOP IF NOT DONE
Ø65Ø	85C3 81 44	L85C3		#'D'	DELETE?
Ø651	85C5 26 48	20303		L860F	NO
Ø652	85C7 6D 84	L85C7	TST		* CHECK FOR END OF LINE
Ø653	85C9 27 F7		-	L85C2	* AND BRANCH IF SO
Ø654	85CB 8D Ø4			L85D1	REMOVE A CHARACTER
Ø655 Ø656	85CD 5A 85CE 26 F7		DECB	L85C7	DECREMENT REPEAT PARAMETER BRANCH IF NOT DONE
Ø657	85UE 26 F7 85DØ 39		RTS	10007	DIMMONI II NOI DONE
Ø658		* REMOV		CHARACTER FROM BUFFER	
Ø659	85D1 ØA D7	L85D1	DEC	VD7	DECREMENT LENGTH OF BUFFER
0660	85D3 31 1F			\$-Ø1,X	POINT Y TO ONE BEFORE CURRENT BUFFER POINTER
Ø661	85D5 31 21	L85D5		\$01,Y	INCREMENT TEMPORARY BUFFER POINTER
Ø662 Ø663	85D7 A6 21 85D9 A7 A4		LDA STA	\$01,Y	GET NEXT CHARACTER PUT IT IN CURRENT POSITION
Ø664	85DB 26 F8			, t L85D5	BRANCH IF NOT END OF LINE
Ø665	85DD 39		RTS		I NOT END OF EIGH
Ø666	85DE 81 49	L85DE	CMPA	#'I'	INSERT?
Ø667	85EØ 27 13			L85F5	YES
Ø668	85E2 81 58			#'X'	EXTEND?
Ø669 Ø67Ø	85E4 27 ØD 85E6 81 48			L85F3 #'H'	YES HACK?
Ø671	85E8 26 5C			L8646	NO
Ø672	85EA 6F 84		CLR		TURN CURRENT BUFFER POINTER INTO END OF LINE FLAG

Ø673	85EC 1F 1Ø		TFR X,D	PUT CURRENT BUFFER POINTER IN ACCD
Ø674	85EE 83 Ø2 DE		SUBD #LINBUF+2	SUBTRACT INITIAL POINTER POSITION
Ø675	85F1 D7 D7		STB VD7	SAVE NEW BUFFER LENGTH
Ø676	85F3 8D BF	L85F3	BSR L85B4	DISPLAY THE LINE ON THE SCREEN
Ø677	85F5 BD 86 87	L85F5	JSR L8687	GET A KEYSTROKE
Ø678	85F8 81 ØD		CMPA #CR	ENTER KEY?
Ø679	85FA 27 8E		BEQ L858A	YES - INTERPRET ANOTHER COMMAND - PRINT LINE
Ø68Ø	85FC 81 1B		CMPA #ESC	ESCAPE?
Ø681	85FE 27 25		BEQ L8625	YES - RETURN TO COMMAND LEVEL - DON T PRINT LINE
Ø682	8600 81 08		CMPA #BS	BACK SPACE?
Ø683	8602 26 22		BNE L8626	NO
Ø684	8604 8C 02 DD		CMPX #LINBUF+1	COMPARE POINTER TO START OF BUFFER
Ø685	8607 27 EC		BEQ L85F5	DO NOT ALLOW BS IF AT START
	8609 8D 45		BSR L865Ø	MOVE POINTER BACK ONE, BS TO SCREEN
	860B 8D C4		BSR L85D1	REMOVE ONE CHARACTER FROM BUFFER
	860D 20 E6		BRA L85F5	GET INSERT SUB COMMAND
Ø689	860F 81 43	L86ØF	CMPA #'C'	CHANGE?
	8611 26 CB		BNE L85DE	NO
	8613 6D 84	L8613		CHECK CURRENT BUFFER CHARACTER
	8615 27 ØE		BEQ L8625	BRANCH IF END OF LINE
	8617 BD 86 87		JSR L8687	GET A KEYSTROKE
	861A 25 Ø2		BLO L861E	BRANCH IF LEGITIMATE KEY
	861C 20 F5	10615	BRA L8613	TRY AGAIN IF ILLEGAL KEY
	861E A7 8Ø 862Ø 8D 37	L861E	STA ,X+	INSERT NEW CHARACTER INTO BUFFER
Ø697	8622 5A		BSR L8659 DECB	SEND NEW CHARACTER TO SCREEN DECREMENT REPEAT PARAMETER
	8623 26 EE		BNE L8613	BRANCH IF NOT DONE
0700	8625 39	L8625	RTS	DIVARCII II NOI DONE
	8626 D6 D7	L8626	LDB VD7	GET LENGTH OF LINE
	8628 C1 F9	_0020	CMPB #LBUFMX-1	COMPARE TO MAXIMUM LENGTH
	862A 26 Ø2		BNE L862E	BRANCH IF NOT AT MAXIMUM
	862C 2Ø C7		BRA L85F5	IGNORE INPUT IF LINE AT MAXIMUM LENGTH
0705	862E 34 1Ø	L862E	PSHS X	SAVE CURRENT BUFFER POINTER
	863Ø 6D 8Ø	L863Ø	TST ,X+	* SCAN THE LINE UNTIL END OF
0707	8632 26 FC		BNE L863Ø	* LINE (Ø) IS FOUND
0708	8634 E6 82	L8634	LDB ,-X	DECR TEMP LINE POINTER AND GET A CHARACTER
0709	8636 E7 Ø1		STB \$Ø1,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
	863E A7 8Ø		STA ,X+	INSERT NEW CHARACTER INTO THE LINE
	864Ø 8D 17		BSR L8659	SEND A CHARACTER TO CONSOLE OUT
	8642 ØC D7		INC VD7	ADD ONE TO BUFFER LENGTH
	8644 20 AF		BRA L85F5	GET INSERT SUB COMMAND
	8646 81 Ø8	L8646	CMPA #BS	BACKSPACE?
	8648 26 12		BNE L865C	NO
	864A 8D Ø4	L864A	BSR L865Ø	MOVE POINTER BACK 1, SEND BS TO SCREEN
	864C 5A		DECB	DECREMENT REPEAT PARAMETER
	864D 26 FB		BNE L864A	LOOP UNTIL DONE
	864F 39	10659	RTS	COMPANE DOINTED TO START OF BUFFER
		L865Ø	CMPX #LINBUF+1	COMPARE POINTER TO START OF BUFFER
Ø724 Ø725	8653 27 DØ 8655 3Ø 1F		BEQ L8625 LEAX \$-01,X	DO NOT ALLOW BS IF AT START
	8657 86 Ø8		LDA #BS	MOVE POINTER BACK ONE BACK SPACE
Ø727	8659 7E A2 82	L8659	JMP LA282	SEND TO CONSOLE OUT
	865C 81 4B	L865C	CMPA #'K'	KILL?
	865E 27 Ø5	20000	BEQ L8665	YES
0730	8660 80 53		SUBA #'S'	SEARCH?
Ø731	8662 27 Ø1		BEQ L8665	YES
	8664 39		RTS	
Ø733	8665 34 Ø2	L8665	PSHS A	SAVE KILL/SEARCH FLAG ON STACK
Ø734	8667 8D 1E		BSR L8687	* GET A KEYSTROKE (TARGET CHARACTER)
Ø735	8669 34 Ø2		PSHS A	* AND SAVE IT ON STACK
Ø736	866B A6 84	L866B	LDA ,X	= GET CURRENT BUFFER CHARACTER
Ø737	866D 27 16		BEQ L8685	= AND RETURN IF END OF LINE
Ø738	866F 6D 61		TST \$Ø1,S	CHECK KILL/SEARCH FLAG
Ø739	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 \$Ø1,X	INCREMENT BUFFER POINTER
	8677 20 03	10070	BRA L867C	CHECK NEXT INPUT CHARACTER
Ø743 Ø744		L8679	JSR L85D1 LDA .X	REMOVE ONE CHARACTER FROM BUFFER GET CURRENT INPUT CHARACTER
Ø744 Ø745	867C A6 84 867E A1 E4	L867C	,	GET CURRENT INPUT CHARACTER COMPARE TO TARGET CHARACTER
Ø745 Ø746	867E AI E4 868Ø 26 E9		CMPA ,S BNE L866B	CUMPARE TO TARGET CHARACTER BRANCH IF NO MATCH
Ø745 Ø747	8682 5A		DECB	DECREMENT REPEAT PARAMETER
Ø747	8683 26 E6		BNE L866B	BRANCH IF NOT DONE
Ø749	8685 35 AØ	L8685	PULS Y,PC	THE Y PULL WILL CLEAN UP THE STACK FOR THE 2 PSHS A
Ø75Ø		*		THE TOTAL MILE SEEM OF THE STROK FOR THE E TOHON
Ø751		* GET A	KEYSTRKE	
Ø752	8687 BD A1 71	L8687	JSR LA171	CALL CONSOLE IN : DEV NBR=SCREEN
Ø753	868A 81 7F		CMPA #\$7F	GRAPHIC CHARACTER?
Ø754	868C 24 F9		BCC L8687	YES - GET ANOTHER CHAR
Ø755	868E 81 5F		CMPA #\$5F	SHIFT UP ARROW (QUIT INSERT)
Ø756	8690 26 02		BNE L8694	NO
Ø757	8692 86 1B		LDA #ESC	REPLACE W/ESCAPE CODE
		L8694	CMPA #CR	ENTER KEY
Ø758	8694 81 ØD		BEQ L86A6	YES
Ø759	8696 27 ØE			
Ø759 Ø76Ø	8696 27 ØE 8698 81 1B		CMPA #ESC	ESCAPE?
0759 0760 0761	8696 27 ØE 8698 81 1B 869A 27 ØA		CMPA #ESC BEQ L86A6	YES
0759 0760 0761 0762	8696 27 ØE 8698 81 1B 869A 27 ØA 869C 81 Ø8		CMPA #ESC BEQ L86A6 CMPA #BS	YES BACKSPACE?
0759 0760 0761 0762 0763	8696 27 ØE 8698 81 1B 869A 27 ØA 869C 81 Ø8 869E 27 Ø6		CMPA #ESC BEQ L86A6 CMPA #BS BEQ L86A6	YES BACKSPACE? YES
0759 0760 0761 0762 0763 0764	8696 27 ØE 8698 81 1B 869A 27 ØA 869C 81 Ø8 869E 27 Ø6 86AØ 81 2Ø		CMPA #ESC BEQ L86A6 CMPA #BS BEQ L86A6 CMPA #SPACE	YES BACKSPACE? YES SPACE
0759 0760 0761 0762 0763 0764 0765	8696 27 ØE 8698 81 1B 869A 27 ØA 869C 81 Ø8 869E 27 Ø6 86AØ 81 2Ø 86A2 25 E3		CMPA #ESC BEQ L86A6 CMPA #BS BEQ L86A6 CMPA #SPACE BLO L8687	YES BACKSPACE? YES SPACE GET ANOTHER CHAR IF CONTROL CHAR
0759 0760 0761 0762 0763 0764 0765 0766	8696 27 ØE 8698 81 1B 869A 27 ØA 869C 81 Ø8 869E 27 Ø6 86AØ 81 2Ø 86A2 25 E3 86A4 1A Ø1	18646	CMPA #ESC BEQ L86A6 CMPA #BS BEQ L86A6 CMPA #SPACE BLO L8687 ORCC #\$Ø1	YES BACKSPACE? YES SPACE
0759 0760 0761 0762 0763 0764 0765	8696 27 ØE 8698 81 1B 869A 27 ØA 869C 81 Ø8 869E 27 Ø6 86AØ 81 2Ø 86A2 25 E3	L86A6	CMPA #ESC BEQ L86A6 CMPA #BS BEQ L86A6 CMPA #SPACE BLO L8687	YES BACKSPACE? YES SPACE GET ANOTHER CHAR IF CONTROL CHAR

Ø769		* TRON			
0770	86A7 86	TRON	FCB	SKP1LD	SKIP ONE BYTE AND LDA #\$4F
0771					
Ø772		* TROFF			
Ø773	86A8 4F	TROFF	CLRA		TROFF FLAG
0774	86A9 97 AF		STA	TRCFLG	TRON/TROFF FLAG:Ø=TROFF, <> Ø=TRON
Ø775	86AB 39		RTS		
Ø776					
Ø777		* POS			
Ø778	86AC 96 6F	POS	LDA	DEVNUM	GET DEVICE NUMBER
Ø779	86AE 34 Ø2		PSHS	A	SAVE IT ON STACK
Ø78Ø	86BØ BD A5 AE		JSR	LA5AE	GET DEVICE NUMBER
Ø781	86B3 BD A4 Ø6		JSR	LA406	FILE STATUS CHECK
Ø782	86B6 BD A3 5F		JSR	LA35F	SET UP TAB FIELD WIDTH
Ø783	86B9 D6 6C		LDB	DEVPOS	GET PRINT POSITION
Ø784	86BB 7E A5 E4		JMP	LA5E4	CONVERT PRINT POSITION TO FLOATING POINT
Ø785					
Ø786		* VARPTE	₹		
Ø787	86BE BD B2 6A	VARPTR		LB26A	SYNTAX CHECK FOR (
Ø788	86C1 DC 1F		LDD	ARYEND	GET ADDR OF END OF ARRAYS
Ø789	86C3 34 Ø6		PSHS		SAVE IT ON STACK
0790	86C5 BD B3 57			LB357	GET VARIABLE DESCRIPTOR
Ø791	86C8 BD B2 67			LB267	SYNTAX CHECK FOR)
Ø792	86CB 35 Ø6		PULS		GET END OF ARRAYS ADDR BACK
Ø793	86CD 1E 1Ø		EXG		SWAP END OF ARRAYS AND VARIABLE DESCRIPTOR
Ø794	86CF 9C 1F			ARYEND	COMPARE TO NEW END OF ARRAYS
Ø795	86D1 26 51			L8724	FC ERROR IF VARIABLE WAS NOT DEFINED PRIOR TO CALLING VARPTR
Ø796	86D3 7E B4 F4		JMP	GIVABF	CONVERT VARIABLE DESCRIPTOR INTO A FP NUMBER
Ø797					
Ø798				ING, POSITION, LENGTH)=REPLACEMENT	
Ø799	86D6 9D 9F	L86D6		GETNCH	GET INPUT CHAR FROM BASIC
0800	86D8 BD B2 6A			LB26A	SYNTAX CHECK FOR (
0801	86DB BD B3 57			LB357	* GET VARIABLE DESCRIPTOR ADDRESS AND
0802	86DE 34 10		PSHS		* SAVE IT ON THE STACK
0803	86EØ EC Ø2			\$Ø2,X	POINT ACCD TO START OF OLDSTRING
Ø8Ø4	86E2 1Ø 93 21			FRETOP	COMPARE TO START OF CLEARED SPACE
0805	86E5 23 Ø4			L86EB	BRANCH IF <=
0806	86E7 93 27			MEMSIZ	SUBTRACT OUT TOP OF CLEARED SPACE
Ø8Ø7	86E9 23 12			L86FD	BRANCH IF STRING IN STRING SPACE
0808	86EB E6 84	L86EB		, Х	GET LENGTH OF OLDSTRING
0809	86ED BD B5 6D			LB56D	RESERVE ACCB BYTES IN STRING SPACE
Ø81Ø	86FØ 34 1Ø		PSHS		SAVE RESERVED SPACE STRING ADDRESS ON STACK
Ø811	86F2 AE 62			\$Ø2,S	POINT X TO OLDSTRING DESCRIPTOR
Ø812	86F4 BD B6 43			LB643	MOVE OLDSTRING INTO STRING SPACE
Ø813	86F7 35 5Ø		PULS		* GET OLDSTRING DESCRIPTOR ADDRESS AND RESERVED STRING
Ø814	86F9 AF 42			\$Ø2,U	* ADDRESS AND SAVE RESERVED ADDRESS AS OLDSTRING ADDRESS
Ø815	86FB 34 4Ø		PSHS		SAVE OLDSTRING DESCRIPTOR ADDRESS
Ø816	86FD BD B7 38	L86FD		LB738	SYNTAX CHECK FOR COMMA AND EVALUATE LENGTH EXPRESSION
Ø817	8700 34 04		PSHS	В	SAVE POSITION PARAMETER ON STACK
Ø818	87Ø2 5D		TSTB		* CHECK POSITION PARAMETER AND BRANCH
Ø819	87Ø3 27 1F			L8724	* IF START OF STRING
Ø82Ø	87Ø5 C6 FF			#\$FF	DEFAULT REPLACEMENT LENGTH = \$FF
Ø821	8707 81 29			#')'	* CHECK FOR END OF MID\$ STATEMENT AND
Ø822	8709 27 03			L87ØE	* BRANCH IF AT END OF STATEMENT
Ø823	87ØB BD B7 38			LB738	SYNTAX CHECK FOR COMMA AND EVALUATE LENGTH EXPRESSION
Ø824	87ØE 34 Ø4	L87ØE	PSHS		SAVE LENGTH PARAMETER ON STACK
Ø825	871Ø BD B2 67			LB267	SYNTAX CHECK FOR)
Ø826	8713 C6 B3			#\$B3	TOKEN FOR =
Ø827	8715 BD B2 6F			LB26F	SYNTAX CHECK FOR =
Ø828	8718 8D 2E			L8748	EVALUATE REPLACEMENT STRING
Ø829	871A 1F 13			X,U	SAVE REPLACEMENT STRING ADDRESS IN U
0830	871C AE 62		LDX	\$Ø2,S	POINT X TO OLOSTRING DESCRIPTOR ADDRESS
Ø831	871E A6 84		LDA	, Х	GET LENGTH OF OLDSTRING
Ø832	8720 AØ 61			\$Ø1,S	SUBTRACT POSITION PARAMETER
Ø833	8722 24 Ø3			L8727	INSERT REPLACEMENT STRING INTO OLDSTRING
Ø834	8724 7E B4 4A 8727 4C	L8724 L8727	JMP INCA	LD44A	FC ERROR IF POSITION > LENGTH OF OLDSTRING * NOW ACCA = NUMBER OF CHARACTERS TO THE RIGHT
Ø835	8/2/ 40	L8/2/ *	INCA		
Ø836 Ø837	8728 A1 E4		CMPA	c	* (INCLUSIVE) OF THE POSITION PARAMETER COMPARE TO LENGTH PARAMETER
Ø837 Ø838	8728 AI E4 872A 24 Ø2			,5 L872E	
Ø838 Ø839	872C A7 E4		STA	.S	BRANCH IF NEW STRING WILL FIT IN OLDSTRING
Ø84Ø	872C A7 E4 872E A6 61	L872E		,5 \$01,S	IF NOT, USE AS MUCH OF LENGTH PARAMETER AS WILL FIT GET POSITION PARAMETER
Ø841	873Ø 1E 89	LU/ ZE	EXG		ACCA=LENGTH OF REPL STRING, ACCB=POSITION PARAMETER
Ø842	8732 AE Ø2			\$02,X	POINT X TO OLDSTRING ADDRESS
Ø843	8734 5A		DECB	*DL, A	* BASIC S POSITION PARAMETER STARTS AT 1; THIS ROUTINE
Ø844	5.5T 5N	*	0200		* WANTS IT TO START AT ZERO
Ø845	8735 3A		ABX		POINT X TO POSITION IN OLDSTRING WHERE THE REPLACEMENT WILL GO
Ø846	8736 4D		TSTA		* IF THE LENGTH OF THE REPLACEMENT STRING IS ZERO
Ø847	8737 27 ØD			L8746	* THEN RETURN
Ø848	8739 A1 E4		CMPA		= IF THE LENGTH OF REPLACEMENT STRING IS <= THE
Ø849	873B 23 Ø2			L873F	ADJUSTED LENGTH PARAMETER, THEN BRANCH
Ø85Ø	873D A6 E4		LDA	,S	OTHERWISE USE AS MUCH ROOM AS IS AVAILABLE
Ø851	873F 1F 89	L873F	TFR		SAVE NUMBER OF BYTES TO MOVE IN ACCB
Ø852	8741 1E 31			U,X	SWAP SOURCE AND DESTINATION POINTERS
Ø853	8743 BD A5 9A			LA59A	MOVE (B) BYTES FROM (X) TO (U)
Ø854	8746 35 96	L8746		A,B,X,PC	CLEAN UP THE STACK AND RETURN
Ø855	8748 BD B1 56	L8748		LB156	EVALUATE EXPRESSION
Ø856	874B 7E B6 54	- · · -	JMP	LB654	* TM ERROR IF NUMERIC; RETURN WITH X POINTING
Ø857		*			*TO STRING, ACCB = LENGTH
Ø858					•, ••
Ø859		* STRING	à		
Ø86Ø	874E BD B2 6A	STRING		LB26A	SYNTAX CHECK FOR (
Ø861	8751 BD B7 ØB			EVALEXPB	EVALUATE EXPRESSION; ERROR IF > 255
Ø862	8754 34 Ø4		PSHS	В	SAVE LENGTH OF STRING
Ø863	8756 BD B2 6D			SYNCOMMA	SYNTAX CHECK FOR COMMA
Ø864	8759 BD B1 56			LB156	EVALUATE EXPRESSION

Ø865	875C BD B2 67		JSR	LB267	SYNTAX CHECK FOR)
Ø866	875F 96 Ø6		LDA	VALTYP	GET VARIABLE TYPE
Ø867	8761 26 Ø5		BNE		BRANCH IF STRING
Ø868	8763 BD B7 ØE		JSR		CONVERT FPAØ INTO AN INTEGER IN ACCB
Ø869	8766 20 03			L876B	SAVE THE STRING IN STRING SPACE
Ø87Ø	8768 BD B6 A4	L8768	JSR		GET FIRST BYTE OF STRING
Ø871	876B 34 Ø4	L876B	PSHS	В	SAVE FIRST BYTE OF EXPRESSION
Ø872	876D E6 61		LDB		GET LENGTH OF STRING
Ø873	876F BD B5 ØF				RESERVE ACCB BYTES IN STRING SPACE
Ø874	8772 35 Ø6		PULS		GET LENGTH OF STRING AND CHARACTER
Ø875	8774 27 Ø5		BEQ		BRANCH IF NULL STRING
Ø876	8776 A7 8Ø	L8776	STA		SAVE A CHARACTER IN STRING SPACE
Ø877	8778 5A	20770	DECB		DECREMENT LENGTH
Ø878	8779 26 FB		BNE		BRANCH IF NOT DONE
Ø879	877B 7E B6 9B	L877B	JMP		PUT STRING DESCRIPTOR ONTO STRING STACK
Ø88Ø	0,,0 ,2 50 55	20775	0111	25035	TOT OTHER DECORET TON ONTO OTHER OTHER
Ø881		* INSTR			
Ø882	877E BD B2 6A	INSTR	JSR	LB26A	SYNTAX CHECK FOR (
Ø883	8781 BD B1 56	2110111	JSR		EVALUATE EXPRESSION
Ø884	8784 C6 Ø1		LDB		DEFAULT POSITION = 1 (SEARCH START)
Ø885	8786 34 Ø4		PSHS		SAVE START
Ø886	8788 96 Ø6				GET VARIABLE TYPE
Ø887	878A 26 1Ø				BRANCH IF STRING
Ø888	878C BD B7 ØE				CONVERT FPAØ TO INTEGER IN ACCB
Ø889	878F E7 E4		STB		SAVE START SEARCH VALUE
Ø89Ø	8791 27 91				BRANCH IF START SEARCH AT ZERO
Ø891	8793 BD B2 6D		JSR		SYNTAX CHECK FOR COMMA
Ø892	8796 BD B1 56				EVALUATE EXPRESSION - SEARCH STRING
Ø893	8799 BD B1 46		JSR	LB146	TM ERROR IF NUMERIC
Ø894	879C 9E 52	L879C	LDX		SEARCH STRING DESCRIPTOR ADDRESS
Ø895	879E 34 1Ø	20,30	PSHS		SAVE ON THE STACK
Ø896	87AØ BD B2 6D				SYNTAX CHECK FOR COMMA
	87A3 BD 87 48				EVALUATE TARGET STRING EXPRESSION
Ø898	87A6 34 14		PSHS		SAVE ADDRESS AND LENGTH ON STACK
Ø899	87A8 BD B2 67				SYNTAX CHECK FOR ')'
0900	87AB AE 63				* LOAD X WITH SEARCH STRING DESCRIPTOR ADDRESS
Ø9Ø1	87AD BD B6 59				* AND GET THE LENGTH ANDADDRESS OF SEARCH STRING
Ø9Ø2	87BØ 34 Ø4		PSHS		SAVE LENGTH ON STACK
Ø9Ø3	0/00 34 04	*	1 3113	ь	SAVE ELIMINI ON STACK
Ø9Ø4			c DOT	NT THE STACK HAS THE FOLLOWING IN	EODMATION
Ø9Ø5				SEARCH LENGTH; 1,S-TARGET LENGTH;	
Ø9Ø6				5,S-SEARCH DESCRIPTOR ADDRESS; 6,	
0907	87B2 E1 66	ADDICES			COMPARE LENGTH OF SEARCH STRING TO START
Ø9Ø8	87B4 25 23			L87D9	POSITION; RETURN Ø IF LENGTH < START
Ø9Ø9	87B6 A6 61		LDA		GET LENGTH OF TARGET STRING
Ø91Ø					
Ø911	87B8 27 1C		BEQ		BRANCH IF TARGET STRING = NULL
Ø911 Ø912	87BA E6 66 87BC 5A		LDB DECB		GET START POSITION MOVE BACK ONE
Ø913	87BD 3A	10705	ABX		POINT X TO POSITION IN SEARCH STRING WHERE SEARCHING WILL START
Ø914	87BE 31 84	L87BE	LEAY		POINT Y TO SEARCH POSITION
Ø915	87CØ EE 62				POINT U TO START OF TARGET
Ø916	87C2 E6 61		LDB		LOAD ACCE WITH LENGTH OF TARGET
Ø917	87C4 A6 E4		LDA		LOAD ACCA WITH LENGTH OF SEARCH
Ø918	87C6 AØ 66				SUBTRACT SEARCH POSITION FROM SEARCH LENGTH
Ø919	87C8 4C		INCA		ADD ONE
0920	87C9 A1 61				COMPARE TO TARGET LENGTH
Ø921	87CB 25 ØC	10700			RETURN Ø IF TARGET LENGTH > WHAT S LEFT OF SEARCH STRING
Ø922	87CD A6 8Ø	L87CD	LDA		GET A CHARACTER FROM SEARCH STRING
Ø923	87CF A1 CØ		CMPA		COMPARE IT TO TARGET STRING
Ø924	87D1 26 ØC				BRANCH IF NO MATCH
Ø925	87D3 5A		DECB		DECREMENT TARGET LENGTH
Ø926	87D4 26 F7	10706	BNE		CHECK ANOTHER CHARACTER
Ø927	87D6 E6 66	L87D6	LDB	,	GET MATCH POSITION
Ø928	87D8 21	L87D8	FCB		SKIP NEXT BYTE
Ø929	87D9 5F	L87D9	CLRB		MATCH ADDRESS = Ø
Ø93Ø Ø931	87DA 32 67			\$07,S LB4F3	CLEAN UP THE STACK
	87DC 7E B4 F3	10705			CONVERT ACCB TO FP NUMBER
	87DF 6C 66 87E1 30 21	L87DF			INCREMENT SEARCH POSITION MOVE X TO NEXT SEARCH POSITION
	87E3 20 D9				
Ø934 Ø935	טונט עט עס		DRA	LU/UL	KEEP LOOKING FOR A MATCH
Ø936		+ ACCTT	TO F1	DATING POINT CONVERSION RAM HOOK	
Ø936 Ø937		XVEC19			*
	0755 01 26	AVELLY			* DETIIDN TE NOT HEY OD OCTAL VARIARIE
Ø938 Ø939	87E5 81 26			L8845	* RETURN IF NOT HEX OR OCTAL VARIABLE
พรรร	87E7 26 5C				
(AU V A			LEAS		PURGE RETURN ADDRESS FROM STACK
Ø94Ø	87E7 26 5C 87E9 32 62	* PROCES	LEAS S A V	ARIABLE PRECEEDED BY A & (&H,&O)
0941	87E7 26 5C 87E9 32 62 87EB ØF 52	* PROCES	LEAS S A V	ARIABLE PRECEEDED BY A & (&H,&O FPAØ+2) * CLEAR BOTTOM TWO
Ø941 Ø942	87E7 26 5C 87E9 32 62 87EB ØF 52 87ED ØF 53	* PROCES	LEAS S A V CLR CLR	ARIABLE PRECEEDED BY A & (&H,&O FPAØ+2 FPAØ+3) * CLEAR BOTTOM TWO * BYTES OF FPAØ
Ø941 Ø942 Ø943	87E7 26 5C 87E9 32 62 87EB ØF 52 87ED ØF 53 87EF 8E ØØ 52	* PROCES	LEAS S A V CLR CLR LDX	ARIABLE PRECEEDED BY A & (&H,&O FPAØ+2 FPAØ+3 #FPAØ+2) * CLEAR BOTTOM TWO * BYTES OF FPAØ BYTES 2,3 OF FPAØ = (TEMPORARY ACCUMULATOR)
0941 0942 0943 0944	87E7 26 5C 87E9 32 62 87EB ØF 52 87ED ØF 53 87EF 8E ØØ 52 87F2 9D 9F	* PROCES	LEAS S A V CLR CLR LDX JSR	ARIABLE PRECEEDED BY A & (&H,&O FPAØ+2 FPAØ+3 #FPAØ+2 GETNCH) * CLEAR BOTTOM TWO * BYTES OF FPAØ BYTES 2,3 OF FPAØ = (TEMPORARY ACCUMULATOR) GET A CHARACTER FROM BASIC
0941 0942 0943 0944 0945	87E7 26 5C 87E9 32 62 87EB 0F 52 87ED 0F 53 87EF 8E 00 52 87F2 9D 9F 87F4 81 4F	* PROCES	LEAS S A V/ CLR CLR LDX JSR CMPA	RRIABLE PRECEEDED BY A & (&H,&O FPAØ+2 FPAØ+3 #FPAØ+2 GETNCH #'O') * CLEAR BOTTOM TWO * BYTES OF FPAØ BYTES 2,3 OF FPAØ = (TEMPORARY ACCUMULATOR) GET A CHARACTER FROM BASIC OCTAL VALUE?
0941 0942 0943 0944 0945 0946	87E7 26 5C 87E9 32 62 87EB 0F 52 87ED 0F 53 87EF 8E 00 52 87E9 9D 9F 87F4 81 4F 87F6 27 12	* PROCES	LEAS S A V/ CLR CLR LDX JSR CMPA BEQ	NRIABLE PRECEEDED BY A & (&H,&O FPAØ+2 FPAØ+3 #FPAØ+2 GETNCH #'O' L88ØA) * CLEAR BOTTOM TWO * ENTES OF FPAØ BYTES 2,3 OF FPAØ = (TEMPORARY ACCUMULATOR) GET A CHARACTER FROM BASIC OCTAL VALUE? YES
0941 0942 0943 0944 0945 0946 0947	87E7 26 5C 87E9 32 62 87EB 0F 52 87ED 0F 53 87EF 8E 00 52 87F2 9D 9F 87F4 81 4F 87F6 27 12 87F8 81 48	* PROCES	LEAS S A VA CLR CLR LDX JSR CMPA BEQ CMPA	ARIABLE PRECEEDED BY A & (&H,&O FPAØ+2 FPAØ+3 #FPAØ+2 GETNCH #'O' L88ØA #'H') * CLEAR BOTTOM TWO * SHYTES OF FPAØ BYTES 2,3 OF FPAØ = (TEMPORARY ACCUMULATOR) GET A CHARACTER FROM BASIC OCTAL VALUE? YES HEX VALUE?
0941 0942 0943 0944 0945 0946 0947	87E7 26 5C 87E9 32 62 87EB 0F 52 87ED 0F 53 87EF 8E 00 52 87F2 9D 9F 87F4 81 4F 87F6 27 12 87F8 81 48 87FA 27 23	* PROCES	LEAS S A V/ CLR CLR LDX JSR CMPA BEQ CMPA BEQ	ARIABLE PRECEEDED BY A & (&H,&O FPAØ+2 FPAØ+3 #FPAØ+2 GETNCH #'O' L880A #'H') * CLEAR BOTTOM TWO * BYTES OF FPAØ BYTES 2,3 OF FPAØ = (TEMPORARY ACCUMULATOR) GET A CHARACTER FROM BASIC OCTAL VALUE? YES HEX VALUE? YES
0941 0942 0943 0944 0945 0946 0947 0948 0949	87E7 26 5C 87E9 32 62 87EB 0F 52 87ED 0F 53 87EF 8E 00 52 87E7 9D 9F 87F4 81 4F 87F6 27 12 87F8 81 48 87FA 27 23 87FC 9D A5	* PROCES	LEAS S A V/ CLR CLR LDX JSR CMPA BEQ CMPA BEQ JSR	ARIABLE PRECEEDED BY A & (&H,&O FPAØ+2 FPAØ+3 #FPAØ+2 GETNCH #'O' L88ØA #'H' BERNER BE) * CLEAR BOTTOM TWO * ENTES OF FPAØ BYTES 2,3 OF FPAØ = (TEMPORARY ACCUMULATOR) GET A CHARACTER FROM BASIC OCTAL VALUE? YES HEX VALUE? YES GET CURRENT INPUT CHARACTER
0941 0942 0943 0944 0945 0946 0947 0948 0949	87E7 26 5C 87E9 32 62 87EB 0F 52 87EB 0F 53 87EF 8E 00 52 87F2 9D 9F 87F4 81 4F 87F6 27 12 87F8 81 48 87FA 27 23 87FC 9D 05 87FE 20 0C	* PROCES L87EB	LEAS S A V/ CLR CLR LDX JSR CMPA BEQ CMPA BEQ JSR BRA	ARIABLE PRECEEDED BY A & (&H,&O FPAØ+2 FPAØ+3 #FPAØ+2 GETNCH #'O' L88ØA #'H' L88IF GETCCH L88ØC) * CLEAR BOTTOM TWO * BYTES OF FPAØ BYTES 2,3 OF FPAØ = (TEMPORARY ACCUMULATOR) GET A CHARACTER FROM BASIC OCTAL VALUE? YES HEX VALUE? YES
0941 0942 0943 0944 0945 0946 0947 0948 0949 0950	87E7 26 5C 87E9 32 62 87EB 0F 52 87ED 0F 53 87EF 8E 00 52 87EP 9D 9F 87F4 81 4F 87F6 27 12 87F8 81 48 87F8 14 48 87F8 27 23 87FC 9D A5 87FC 9D A5 87FE 20 0C 8800 81 38	* PROCES	LEAS S A V/ CLR CLR LDX JSR CMPA BEQ CMPA BEQ JSR BRA CMPA	ARIABLE PRECEDED BY A & (&H,&O FPAØ+2 FPAØ+3 #FPAØ+2 GETNCH #'O' L880A #'H' L881F GETCCH L880C #'8') * CLEAR BOTTOM TWO * BYTES OF FPAØ BYTES 2,3 OF FPAØ = (TEMPORARY ACCUMULATOR) GET A CHARACTER FROM BASIC OCTAL VALUE? YES HEX VALUE? YES GET CURRENT INPUT CHARACTER DEFAULT TO OCTAL (&0) *
0941 0942 0943 0944 0945 0946 0947 0948 0949 0950 0951	87E7 26 5C 87E9 32 62 87EB 0F 52 87ED 0F 53 87EF 8E 00 52 87F2 9D 9F 87F4 81 4F 87F6 27 12 87F8 81 48 87FA 27 23 87FC 9D A5 87FE 20 0C 8800 81 38 8802 10 22 2A 71	* PROCES L87EB	LEAS S A V/CLR CLR LDX JSR CMPA BEQ CMPA BEQ JSR BRA CMPA LBHI	ARIABLE PRECEEDED BY A & (&H,&O FPAØ+2 FPAØ+3 #FPAØ+2 GETNCH #'O' L88ØA #'H' BETCCH L88ØC #'8' L8277) * CLEAR BOTTOM TWO * BYTES OF FPAØ BYTES 2,3 OF FPAØ = (TEMPORARY ACCUMULATOR) GET A CHARACTER FROM BASIC OCTAL VALUE? YES HEX VALUE? YES GET CURRENT INPUT CHARACTER DEFAULT TO OCTAL (&O) * * SYNTAX ERROR IF
0941 0942 0943 0944 0945 0946 0947 0948 0949 0950 0951 0952	87E7 26 5C 87E9 32 62 87EB 0F 52 87ED 0F 53 87EF 8E 00 52 87FE 9D 9F 87F4 81 4F 87F6 27 12 87F8 81 48 87FA 27 23 87FC 9D A5 87FE 20 0C 8800 81 38 8802 10 22 2A 71 8806 C6 03	* PROCES L87EB	LEAS S A V CLR CLR LDX JSR CMPA BEQ CMPA BEQ JSR BRA CMPA LBHI LDB	ARIABLE PRECEEDED BY A & (&H,&O FPAØ+2 FPAØ+3 #FPAØ+3 #FPAØ+2 GETNCH #'O' L88ØA #'H' L88IF GETCCH L88ØC #'8' L8277 #\$03	CLEAR BOTTOM TWO * CLEAR BOTTOM TWO * BYTES 0, FPAØ BYTES 2, 3 OF FPAØ = (TEMPORARY ACCUMULATOR) GET A CHARACTER FROM BASIC OCTAL VALUE? YES HEX VALUE? YES GET CURRENT INPUT CHARACTER DEFAULT TO OCTAL (&0) * * SYNTAX ERROR IF BASE 8 MULTIPLIER
0941 0942 0943 0944 0945 0946 0947 0948 0949 0950 0951 0952 0953	87E7 26 5C 87E9 32 62 87EB 0F 52 87ED 0F 53 87EF 8E 00 52 87F2 9D 9F 87F4 81 4F 87F6 27 12 87F8 81 48 87FA 27 23 87FC 9D A5 87FE 20 0C 8800 81 38 8802 10 22 2A 71	* PROCES L87EB	LEAS S A V/ CLR CLR LDX JSR CMPA BEQ CMPA BEQ JSR CMPA BEQ LSR CMPA LBHI LDB BSR	ARIABLE PRECEEDED BY A & (&H,&O FPAØ+2 FPAØ+3 #FPAØ+2 GETNCH #'O' L88ØA #'H' H' L881F GETCCH L880C L8277 #\$Ø3 LB277 #\$Ø3 LB234) * CLEAR BOTTOM TWO * BYTES OF FPAØ BYTES 2,3 OF FPAØ = (TEMPORARY ACCUMULATOR) GET A CHARACTER FROM BASIC OCTAL VALUE? YES HEX VALUE? YES GET CURRENT INPUT CHARACTER DEFAULT TO OCTAL (&O) * * SYNTAX ERROR IF
0941 0942 0943 0944 0945 0946 0947 0948 0949 0950 0951 0952 0953 0954	87E7 26 5C 87E9 32 62 87EB ØF 52 87ED ØF 53 87EF 8E ØØ 52 87F2 9D 9F 87F4 81 4F 87F6 27 12 87F8 81 48 87FA 27 23 87FC 9D A5 87FC 9D A5 87FC 9D A5 88ØØ 81 38 88ØØ 62 60 Ø3 88ØØ 80 2A	* PROCES L87EB	LEAS S A V/ CLR CLR LDX JSR CMPA BEQ CMPA BEQ JSR CMPA CMPA LBHI LDB BSR TE AN	ARIABLE PRECEEDED BY A & (&H,&O FPAØ+2 FPAØ+3 #FPAØ+2 GETNCH #'O' L88ØA #'H' L88IF GETCCH L88ØC #'B' S' L8277 #\$03 L8834 & VARIABLE	* CLEAR BOTTOM TWO * CLEAR BOTTOM TWO * BYTES 0, FPAØ BYTES 2, 3 OF FPAØ = (TEMPORARY ACCUMULATOR) GET A CHARACTER FROM BASIC OCTAL VALUE? YES HEX VALUE? YES GET CURRENT INPUT CHARACTER DEFAULT TO OCTAL (&O) * * SYNTAX ERROR IF BASE 8 MULTIPLIER ADD DIGIT TO TEMPORARY ACCUMULATOR
0941 0942 0943 0944 0945 0946 0947 0948 0949 0950 0951 0952 0953 0954 0955	87E7 26 5C 87E9 32 62 87EB 0F 52 87ED 0F 53 87EF 8E 00 52 87F2 9D 9F 87F4 81 4F 87F6 27 12 87F8 81 48 87FA 27 23 87FC 9D A5 87FE 20 0C 8800 81 38 8802 10 22 2A 71 8806 C6 03 8808 8D 2A	* PROCES L87EB	LEAS S A V CLR CLR LDX JSR CMPA BEQ CMPA BEQ JSR BRA CMPA LBHI LDB BSR TE AN JSR	RIABLE PRECEEDED BY A & (&H,&O FPAØ+2 FPAØ+3 #FPAØ+3 #FPAØ+2 GETNCH #'O' L88ØA #'H' L880F GETCCH L880C #'8' L8277 #\$93 L8834 & & O VARIABLE GETNCH) * CLEAR BOTTOM TWO * BYTES OF FPAØ BYTES 2,3 OF FPAØ = (TEMPORARY ACCUMULATOR) GET A CHARACTER FROM BASIC OCTAL VALUE? YES HEX VALUE? YES GET CURRENT INPUT CHARACTER DEFAULT TO OCTAL (&0) * * SYNTAX ERROR IF BASE 8 MULTIPLIER ADD DIGIT TO TEMPORARY ACCUMULATOR GET A CHARACTER FROM BASIC
0941 0942 0943 0944 0945 0946 0947 0950 0950 0951 0952 0953 0954 0956	87E7 26 5C 87E9 32 62 87EB 0F 52 87ED 0F 53 87EF 8E 00 52 87F2 9D 9F 87F4 81 4F 87F6 27 12 87F8 81 48 87FA 27 23 87FC 9D A5 87FC 20 0C 8800 81 38 8802 10 22 2A 71 8806 C6 03 8808 8D 2A	* PROCES L87EB L88ØØ * EVALUA L88ØA L88ØC	LEAS S A V CLR CLR LDX JSR CMPA BEQ CMPA BEQ JSR BRA CMPA LBHI LDB BSR TE AN JSR BLO	ARIABLE PRECEEDED BY A & (&H,&O FPAØ+2 FPAØ+3 #FPAØ+2 GETNCH #'O' L88ØA #'H' H' L881F GETCCH L880C L8'8C L8'8C L8'8C L8'8C LB'8C LB') * CLEAR BOTTOM TWO * BYTES OF FPAØ BYTES 2,3 OF FPAØ = (TEMPORARY ACCUMULATOR) GET A CHARACTER FROM BASIC OCTAL VALUE? YES HEX VALUE? YES GET CURRENT INPUT CHARACTER DEFAULT TO OCTAL (&O) * * SYNTAX ERROR IF BASE 8 MULTIPLIER ADD DIGIT TO TEMPORARY ACCUMULATOR GET A CHARACTER FROM BASIC BRANCH IF NUMERIC
0941 0942 0943 0944 0945 0947 0948 0949 0951 0952 0953 0954 0955 0956	87E7 26 5C 87E9 32 62 87EB ØF 52 87ED ØF 53 87EF 8E ØØ 52 87F2 9D 9F 87F4 81 4F 87F6 27 12 87F8 81 48 87FA 27 23 87FC 9D A5 87FC 9D A5 87FC 20 ØC 88ØØ 81 38 88ØZ 10 22 2A 71 88Ø6 C6 Ø3 88Ø8 8D 2A	* PROCES L87EB	LEAS S A V/ CLR CLR LDX JSR CMPA BEQ JSR CMPA BEQ JSR CMPA BEQ JSR CMPA LBHI LDB BSR LTE AN JSR CLR	ARIABLE PRECEEDED BY A & (&H,&O FPAØ+2 FPAØ+3 #FPAØ+2 GETNCH #'O' L880A #'H' L881F GETCCH L880C #'B' L8277 #\$03 L8277 #\$03 L8834 &O VARIABLE GETNCH L880Ø FPAØ FPAØ	* CLEAR BOTTOM TWO * CLEAR BOTTOM TWO * BYTES 0, FPAØ BYTES 2, 3 OF FPAØ = (TEMPORARY ACCUMULATOR) GET A CHARACTER FROM BASIC OCTAL VALUE? YES HEX VALUE? YES GET CURRENT INPUT CHARACTER DEFAULT TO OCTAL (&O) * * SYNTAX ERROR IF BASE 8 MULTIPLIER ADD DIGIT TO TEMPORARY ACCUMULATOR GET A CHARACTER FROM BASIC BRANCH IF NUMERIC * CLEAR 2 HIGH ORDER
8941 8942 9943 8944 8945 9946 8947 8948 8959 8959 8951 8952 8953 8954 8956 8957 8958	87E7 26 5C 87E9 32 62 87EB 0F 52 87ED 0F 53 87EF 8E 00 52 87FE 9D 9F 87F4 81 4F 87F6 27 12 87F8 81 48 87FA 27 23 87FC 9D A5 87FE 20 0C 8800 81 38 8802 10 22 2A 71 8806 C6 03 8808 8D 2A 880A 9D 9F 880C 25 F2 880C 8F 50 8810 0F 50	* PROCES L87EB L88ØØ * EVALUA L88ØA L88ØC	LEAS S A V/ CLR CLR LDX JSR CMPA BEQ JSR CMPA BEQ JSR CMPA LBHI LDB BSR LTE AN JSR CLR CLR	ARIABLE PRECEEDED BY A & (&H,&O FPAØ+2 FPAØ+3 #FPAØ+2 GETNCH #'O' L88ØA #'H' L88ØC #'8' L8277 #\$63 A & C&H L88ØC #\$73 L834 & CO VARIABLE GETNCH L88ØØ FPAØ FPAØ FPAØ FPAØ FPAØ FPAØ FPAØ FP) * CLEAR BOTTOM TWO * BYTES OF FPAØ BYTES 2,3 OF FPAØ = (TEMPORARY ACCUMULATOR) GET A CHARACTER FROM BASIC OCTAL VALUE? YES HEX VALUE? YES GET CURRENT INPUT CHARACTER DEFAULT TO OCTAL (&0) * * SYNTAX ERROR IF BASE 8 MULTIPLIER ADD DIGIT TO TEMPORARY ACCUMULATOR GET A CHARACTER FROM BASIC BRANCH IF NUMERIC * CLEAR 2 HIGH ORDER * BYTES OF FPAØ
0941 0942 0943 0944 0945 0947 0948 0949 0951 0952 0953 0954 0955 0956	87E7 26 5C 87E9 32 62 87EB ØF 52 87ED ØF 53 87EF 8E ØØ 52 87F2 9D 9F 87F4 81 4F 87F6 27 12 87F8 81 48 87FA 27 23 87FC 9D A5 87FC 9D A5 87FC 20 ØC 88ØØ 81 38 88ØZ 10 22 2A 71 88Ø6 C6 Ø3 88Ø8 8D 2A	* PROCES L87EB L88ØØ * EVALUA L88ØA L88ØC	LEAS S A V/ CLR CLR LDX JSR CMPA BEQ JSR CMPA BEQ JSR CMPA LBHI LDB BSR LTE AN JSR CLR CLR	ARIABLE PRECEEDED BY A & (&H,&O FPAØ+2 FPAØ+3 #FPAØ+2 GETNCH #'O' L880A #'H' L881F GETCCH L880C #'B' L8277 #\$03 L8277 #\$03 L8834 &O VARIABLE GETNCH L880Ø FPAØ FPAØ	* CLEAR BOTTOM TWO * CLEAR BOTTOM TWO * BYTES 0, FPAØ BYTES 2, 3 OF FPAØ = (TEMPORARY ACCUMULATOR) GET A CHARACTER FROM BASIC OCTAL VALUE? YES HEX VALUE? YES GET CURRENT INPUT CHARACTER DEFAULT TO OCTAL (&O) * * SYNTAX ERROR IF BASE 8 MULTIPLIER ADD DIGIT TO TEMPORARY ACCUMULATOR GET A CHARACTER FROM BASIC BRANCH IF NUMERIC * CLEAR 2 HIGH ORDER

0961	8814 ØF 63		FPSBYT	ZERO OUT SUB BYTE OF FPAØ
Ø962	8816 ØF 54		FPØSGN	ZERO OUT MANTISSA SIGN OF FPAØ
Ø963 Ø964	8818 C6 AØ 881A D7 4F		#\$AØ FPØEXP	* SET EXPONENT OF FPAØ *
Ø965	881C 7E BA 1C		LBA1C	GO NORMALIZE FPAØ
Ø966			N &H VARIABLE	
Ø967	881F 9D 9F	L881F JSR		GET A CHARACTER FROM BASIC
Ø968	8821 25 ØB	BLO	L882E	BRANCH IF NUMERIC
Ø969	8823 BD B3 A2	JSR BLO CMPA	LB3A2	SET CARRY IF NOT ALPHA
Ø97Ø Ø971	8826 25 E6 8828 81 47	BLU BLU	L880E	BRANCH IF NOT ALPHA OR NUMERIC CHECK FOR LETTERS A-F
Ø972	882A 24 E2	CMP/ BCC SUB/ L882E LDB BSR BRA	L88ØE	BRANCH IF >= G (ILLEGAL HEX LETTER)
Ø973	882C 8Ø Ø7	SUBA	\ #'A'-('9'+1)	SUBTRACT ASCII DIFFERENCE BETWEEN A AND 9
Ø974	882E C6 Ø4	L882E LDB	#\$04	BASE 16 DIGIT MULTIPLIER = 2**4
Ø975	883Ø 8D Ø2	BSR	L8834	ADD DIGIT TO TEMPORARY ACCUMULATOR
Ø976	8832 20 EB	BRA	L881F	KEEP EVALUATING VARIABLE
Ø977	8834 68 01	L8834 ASL	\$01,X	* MULTIPLY TEMPORARY
Ø978 Ø979	8836 69 84 8838 10 25 32 56	KUL	,X S LBA92	* ACCUMULATOR BY TWO OV' OVERFLOW ERROR
Ø98Ø	883C 5A	DECE		DECREMENT SHIFT COUNTER
Ø981	883D 26 F5	BNE	L8834 A #'0' A \$01,X	MULTIPLY TEMPORARY ACCUMULATOR AGAIN
Ø982	883F 8Ø 3Ø	SUBA	A #'Ø'	MASK OFF ASCII
Ø983	8841 AB Ø1	ADDA	A \$01,X	* ADD DIGIT TO TEMPORARY
	8843 A7 Ø1		\$Ø1,X	* ACCUMULATOR AND SAVE IT
	8845 39	L8845 RTS		
Ø986 Ø987		* EADDECCIUN	EVALUATION RAM HOOK	
Ø988	8846 35 40	XVEC15 PULS	EVALUATION RAM HOUR	PULL RETURN ADDRESS AND SAVE IN U REGISTER
Ø989	8848 ØF Ø6	CLR	VALTYP	SET VARIABLE TYPE TO NUMERIC
0990	884A 9E A6		CHARAD	CURRENT INPUT POINTER TO X
0991	884C 9D 9F	JSR	GETNCH	GET CHARACTER FROM BASIC
Ø992	884E 81 26	CMPA	4'&'	HEX AND OCTAL VARIABLES ARE PRECEEDED BY &
	8850 27 99	BEQ	A # %' L87EB A #\$CC L88B4 A #\$FF L8862 GETNCH A #\$83 L L892C CLAABA	PROCESS A & VARIABLE
	8852 81 CC 8854 27 5E	CMP#	1 #\$CC 1 8884	TOKEN FOR FN PROCESS FN CALL
Ø995	8856 81 FF	CWE	\ #\$FF	CHECK FOR SECONDARY TOKEN
Ø997	8858 26 Ø8	BNE	L8862	NOT SECONDARY
Ø998	885A 9D 9F	JSR	GETNCH	GET CHARACTER FROM BASIC
Ø999	885A 9D 9F 885C 81 83 885F 10 27 00 CA	CMPA	A #\$83	TOKEN FOR USR
1000	885C 81 83 885E 10 27 00 CA 8862 9F A6	LBEC	L892C	PROCESS USR CALL
1001 1002	8862 9F A6 8864 6E C4	L0002 31X	CHARAD	RESTORE BASIC S INPUT POINTER
1002			,U CURLIN	RETURN TO CALLING ROUTINE GET CURRENT LINE NUMBER
1004	8868 3Ø Ø1			IN DIRECT MODE?
1005	886A 26 D9		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	0071 AF OF GG AC	* DEF	FOUADADA	CET THE INDIT CHARC
1010 1011	8871 AE 9F ØØ A6 8875 8C FF 83	DEF LDX CMP) LBE(BSR	(#¢FFQ3	GET TWO INPUT CHARS TOKEN FOR USR
1011	8878 10 27 00 93	I RF() 1890F	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	888Ø BD B2 6A	BSR JSR LDB STB JSR BSR JSR LDB	LB26A	SYNTAX CHECK FOR (
1016	8883 C6 8Ø	LDB	#\$80	* GET THE FLAG TO INDICATE ARRAY VARIABLE SEARCH DISABLE
1017	8885 D7 Ø8	218	ARYDIS	* AND SAVE IT IN THE ARRAY DISABLE FLAG
1018 1019	8887 BD B3 57 888A 8D 25	JSK RCD	LB35/	GET VARIABLE DESCRIPTOR 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		V4B	GET THE ADDRESS OF THE FN NAME DESCRIPTOR
1024	8896 DC A6		CHARAD	* GET THE CURRENT INPUT POINTER ADDRESS AND
1025	8898 ED 84		,X	* SAVE IT IN FIRST 2 BYTES OF THE DESCRIPTOR
1026	889A DC 39 889C ED Ø2		VARPTR \$Ø2,X	= GET THE DESCRIPTOR ADDRESS OF THE ARGUMENT = VARIABLE AND SAVE IT IN THE DESCRIPTOR OF THE FN NAME
1028	889E 7E AE EØ	JMP		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		DO A SYNTAX CHECK FOR FN
1031	88A6 C6 8Ø	LDB		* GET THE FLAG TO INDICATE ARRAY VARIABLE SEARCH DISABLE FLAG
1032	88A8 D7 Ø8	STB		* AND SAVE IT IN ARRAY VARIABLE FLAG
1033 1034	88AA 8A 8Ø 88AC BD B3 5C	JSR	#\$80 LB35C	SET BIT 7 OF CURRENT INPUT CHARACTER TO INDICATE AN FN VARIABLE * GET THE DESCRIPTOR ADDRESS OF THIS
1035	88AF 9F 4B	STX		* VARIABLE AND SAVE IT IN V4B
1036	88B1 7E B1 43		LB143	TM ERROR IF STRING VARIABLE
1037		* EVALUATE A	N FN CALL	
1038	88B4 8D EB		L88A1	* GET THE DESCRIPTOR OF THE FN NAME
1039	88B6 34 1Ø	PSHS		* VARIABLE AND SAVE IT ON THE STACK
1040 1041	88B8 BD B2 62 88BB 8D F4		LB262 L88B1	SYNTAX CHECK FOR (& EVALUATE EXPR TM ERROR IF STRING VARIABLE
1042	88BD 35 4Ø	PULS		POINT U TO FN NAME DESCRIPTOR
1043	88BF C6 32		#2*25	UNDEFINED FUNCTION CALL ERROR
1044	88C1 AE 42	LDX	\$Ø2,U	POINT X TO ARGUMENT VARIABLE DESCRIPTOR
1045	88C3 27 A9		L886E	BRANCH TO ERROR HANDLER
1046	88C5 1Ø 9E A6	LDY		SAVE CURRENT INPUT POINTER IN Y
1047	88C8 EE C4	LDU		* POINT U TO START OF FN FORMULA AND
1048 1049	88CA DF A6 88CC A6 Ø4		CHARAD \$04,X	* SAVE IT IN INPUT POINTER = GET FP VALUE OF
1049	88CE 34 Ø2		эр4,х 5 А	= ARGUMENT VARIABLE, CURRENT INPUT
1050	88DØ EC 84	LDD		= POINTER, AND ADDRESS OF START
1052	88D2 EE Ø2	LDU	\$02,X	= OF FN FORMULA AND SAVE
1053	88D4 34 76		S U,Y,X,B,A	= THEM ON THE STACK
1054	88D6 BD BC 35		LBC35	PACK FPAØ AND SAVE IT IN (X)
1055	88D9 BD B1 41	L88D9 JSR	LB141	EVALUATE FN EXPRESSION
1056		PIII	SARXYII	RESTORE REGISTERS
1056	88DC 35 76	PULS	S A,B,X,Y,U	RESTORE REGISTERS

1058 1059 1060 1061 1062 1063 1064	88DE ED 84 88EØ EF Ø2 88E2 35 Ø2 88E4 A7 Ø4 88E6 9D A5 88E8 10 26 29 8B 88EC 10 9 F A6 88EF 39	STD STU PULS STA JSR LBNE STY L88EF RTS	,X \$02,X A \$04,X GETCCH LB277 CHARAD	* GET THE FP * VALUE OF THE ARGUMENT * VARIABLE OFF OF THE * STACK AND RE-SAVE IT GET FINAL CHARACTER OF THE FN FORMULA SYNTAX ERROR IF NOT END OF LINE RESTORE INPUT POINTER
1068 1069 1070 1071 1072 1073 1074 1075	88FØ C1 32 88F2 25 FB 88F4 BD A7 E9 88F7 BD A9 74 88FA BD AD 33 88FD ØF 6F 88FF BD B9 5C 89Ø2 BD B9 AF 89Ø5 8E 88 D9 89Ø8 7E AC 6Ø	JSR JSR JSR CLR JSR JSR LDX JMP	#2*25 L88EF LA7F9 LA974 LA033 DEVNUM L895C L89AF #L890B-25*2 LAC60	CHECK FOR EXBAS ERROR NUMBER BRANCH IF < EXBAS ERROR TURN CASSETTE MOTOR OFF DISABLE ANALOG MULTIPLEXER DO PART OF A NEW SET DEVICE NUMBER TO SCREEN MOVE CURSOR TO START OF NEXT LINE SEND A ? TO CONSOLE OUT POINT X TO EXBAS ERRORS PROCESS ERROR
1078 1079		* ADDITIONAL	ERROR MESSAGES ADDED BY EXTENDED	BASIC
1081	89ØB 55 46 89ØD 4E 45	L890B FCC L890D FCC	'UF' 'NE'	25 UNDEFINED FUNCTION (FN) CALL 26 FILE NOT FOUND
1085 1086 1087 1088	890F 9D 9F 8911 8D 09 8913 34 10 8915 8D 2D 8917 35 40 8919 AF C4 8918 39	PSHS	L891C X L8944 U	SKIP PAST SECOND BYTE OF DEF USR TOKEN GET FN NUMBER SAVE FN EXEC ADDRESS STORAGE LOC CALCULATE EXEC ADDRESS GET FN EXEC ADDRESS STORAGE LOC SAVE EXEC ADDRESS
1091 1092 1093 1094 1095 1096 1097 1098 1099	891C 5F 891D 9D 9F 891F 24 06 8921 80 30 8923 1F 89 8925 9D 9F	L891C CLRB JSR BCC SUBA TFR JSR	GETNCH L8927 #'0' A,B GETNCH USRADR	DEFAULT TO USRØ IF NO ARGUMENT GET A CHARACTER FROM BASIC BRANCH IF NOT NUMERIC MASK OFF ASCII SAVE USR NUMBER IN ACCB GET A CHARACTER FROM BASIC GET ADDRESS OF STORAGE LOCS FOR USR ADDRESS X2 - 2 BYTES/USR ADDRESS ADD OFFSET TO START ADDRESS OF STORAGE LOCS
1101		* PROCESS A U	SR CALL	0FT 0700105 100 05 FW50 4000500 500 W00 W
1103 1104 1105 1106 1107 1108 1109 1110 1111 1112 1113 1114 1115	892C 80 EE 892E AE 84 8938 34 10 8932 8D 82 62 8938 96 06 8938 97 07 893C 8D 86 57 893C 8D 86 57 893F 9E 52 8941 96 06 8943 39 8944 C6 83 8946 8D 82 6F 8949 7E 87 3D			GET STORAGE LOC OF EXEC ADDRESS FOR USR N * GET EXEC ADDRESS AND * PUSH IT ONTO STACK SYNTAX CHECK FOR (& EVALUATE EXPR POINT X TO FPAØ GET VARIABLE TYPE BRANCH IF NUMERIC, STRING IF Ø GET LENGTH & ADDRESS OF STRING VARIABLE GET POINTER TO STRING DESCRIPTOR GET VARIABLE TYPE JUMP TO USR ROUTINE (PSHS X ABOVE) TOKEN FOR = DO A SYNTAX CHECK FOR = EVALUATE EXPRESSION, RETURN VALUE IN X
1116 1117		* EXTENDED BA	SIC S IRQ ROUTINE	
1120 1121 1122 1123 1124 1125 1126	894C B6 FF Ø3 894F 28 Ø1 8951 38 8952 B6 FF Ø2 8955 BE Ø1 12 8958 39 Ø1 895A BF Ø1 12 895D 7E 9C 3E	RTI L8952 LDA L8955 LDX LEAX STX	PIA#+3 L8952 PIA#+2 TIMVAL \$01, X TIMVAL L9C3E	GET PIAØ, PORT B CONTROL REGISTER BRANCH IF 60 HZ INTERRUPT RETURN IF 63.5 MICROSECOND INTERRUPT RESET PIA INTERRUPT FLAG GET REAL TIME CLOCK INCREMENT IT SAVE IT GO CHECK SOME MORE STUFF
1129 1130	896Ø 9D 9F 8962 8D EØ 8964 BF Ø1 12 8967 39		GETNCH L8944 TIMVAL	GET A CHARACTER FROM BASIC GET NEW TIMER VALUE SET TIMER COUNTER
1135 1136 1137	8968 BE Ø1 12 896B 9F 52 896D 7E 88 ØE	JMP	TIMVAL FPAØ+2 L88ØE	GET TIMER VALUE SAVE TIMER VALUE IN BOTTOM OF FPAØ CONVERT BALANCE OF FPAØ TO POSITIVE INTEGER
1140 1141 1142 1143 1144 1145 1146 1147 1148 1149 1150 1151	8970 10 27 2A D6 8974 8D AF 67 8977 8D AD 01 897A 9F D3 897C 9D A5 897E 27 10 8988 81 AC 8982 26 38 8984 9D 9F 8986 8D 24 8988 8D 24 8988 8D 24 898A 20 04 898C 96 FF 898E 97 2B	JSR STX JSR BEQ CMPA BNE JSR BEQ BSR BEQ BSR L898C LDA	LAF67 LADØ1 VD3 GETCCH L8990 #\$AC L89BF GETNCH L89BC L89AE L8990	FC ERROR IF NO ARGUMENT CONVERT A DECIMAL BASIC NUMBER TO BINARY FIND RAM ADDRESS OF START OF A BASIC LINE SAVE RAM ADDRESS OF STARTING LINE NUMBER GET CURRENT INPUT CHARACTER BRANCH IF END OF LINE TOKEN FOR -' TERNINATE COMMAND IF LINE NUMBER NOT FOLLOWED BY - GET A CHARACTER FROM BASIC IF END OF LINE, USE DEFAULT ENDING LINE NUMBER * CONVERT ENDING LINE NUMBER TO BINARY * AND SAVE IT IN BINVAL = USE \$FFXX AS DEFAULT ENDING = LINE NUMBER - SAVE IT IN BINVAL

1153	8990 DE D3	L899Ø	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 Ø6		BEQ	L899F	BRANCH IF END OF PROGRAM
1158	8999 EC 42		LDD	\$Ø2,U	LOAD ACCD WITH THIS LINE S NUMBER
1159	899B 93 2B			BINVAL	SUBTRACT ENDING LINE NUMBER ADDRESS
1160	899D 23 F4			L8993	BRANCH IF = < ENDING LINE NUMBER
1161	899F 9E D3	L899F		VD3	GET STARTING LINE NUMBER
1162		L0331		L89B8	
	89A1 8D 15				MOVE (U) TO (X) UNTIL END OF PROGRAM
1163	89A3 BD AD 21			LAD21	RESET BASIC S INPUT POINTER AND ERASE VARIABLES
1164	89A6 9E D3		LDX		GET STARTING LINE NUMBER ADDRESS
1165	89A8 BD AC F1			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 CØ	L89B4		,U+	GET A BYTE FROM (U)
1170	89B6 A7 8Ø			,X+	MOVE THE BYTE TO (X)
1171	89B8 11 93 1B	L89B8		VARTAB	COMPARE TO END OF BASIC
		20300			
1172	89BB 26 F7			L89B4	BRANCH IF NOT AT END
1173	89BD 9F 1B			VARTAB	SAVE (X) AS NEW END OF BASIC
1174	89BF 39	L89BF	RTS		
1175					
1176		* LINE	INPUT		
1177	89CØ BD 88 66	L89CØ	JSR	L8866	BS ERROR IF IN DIRECT MODE
1178	89C3 9D 9F			GETNCH	GET A CHAR FROM BASIC
1179	89C5 81 23			#'#'	* CHECK FOR DEVICE NUMBER FLAG AND
1180					
	89C7 26 Ø9			L89D2	* BRANCH IF NOT THERE
1181	89C9 BD A5 A5			LA5A5	CHECK FOR VALID DEVICE NUMBER
1182	89CC BD A3 ED			LA3ED	CHECK FOR OPEN FILE
1183	89CF BD B2 6D			SYNCOMMA	SYNTAX CHECK FOR COMMA
1184	89D2 81 22	L89D2		#'"'	CHECK FOR PROMPT STRING
1185	89D4 26 ØB		BNE	L89E1	BRANCH IF NO PROMPT STRING
1186	89D6 BD B2 44			LB244	STRIP OFF PROMPT STRING & PUT IT ON STRING STACK
1187	89D9 C6 3B			#';'	*
1188	89DB BD B2 6F			LB26F	* DO A SYNTAX CHECK FOR;
1189	89DE BD B9 9F			LB99F	REMOVE PROMPT STRING FROM STRING STACK & SEND TO CONSOLE OUT
1190	89E1 32 7E	L89E1		\$-02,S	RESERVE TWO STORAGE SLOTS ON STACK
1191	89E3 BD BØ 35			LBØ35	INPUT A LINE FROM CURRENT INPUT DEVICE
1192	89E6 32 62			\$02,S	CLEAN UP THE STACK
1193	89E8 ØF 6F		CLR	DEVNUM	SET DEVICE NUMBER TO SCREEN
1194	89EA BD B3 57			LB357	SEARCH FOR A VARIABLE
1195	89ED 9F 3B			VARDES	SAVE POINTER TO VARIABLE DESCRIPTOR
1196	89EF BD B1 46			LB146	TM ERROR IF VARIABLE TYPE = NUMERIC
1197	89F2 8E Ø2 DC			#LINBUF	POINT X TO THE STRING BUFFER WHERE THE INPUT STRING WAS STORED
1198	89F5 4F		CLRA		TERMINATOR CHARACTER Ø (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			BINVAL	GET BINARY VALUE
1202	0 / 01 20				GET BINAKT VALUE
1203	8AØ1 39	10400	RTS		
1204	8AØ2 9E D1	L8AØ2	RTS LDX	VD1	GET CURRENT OLD NUMBER BEING RENUMBERED
1204 1205	8AØ2 9E D1 8AØ4 9F 2B	L8AØ2 L8AØ4	RTS LDX STX	VD1 BINVAL	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR
1204	8AØ2 9E D1		RTS LDX	VD1	GET CURRENT OLD NUMBER BEING RENUMBERED
1204 1205	8AØ2 9E D1 8AØ4 9F 2B		RTS LDX STX	VD1 BINVAL	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR
1204 1205 1206	8AØ2 9E D1 8AØ4 9F 2B		RTS LDX STX	VD1 BINVAL	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR
1204 1205 1206 1207	8AØ2 9E D1 8AØ4 9F 2B	L8AØ4	RTS LDX STX JMP	VD1 BINVAL	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR
1204 1205 1206 1207 1208 1209	8A02 9E D1 8A04 9F 2B 8A06 7E AD 01 8A09 BD AD 26	L8AØ4 * RENUM	RTS LDX STX JMP	VD1 BINVAL LADØ1 LAD26	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR GO FIND THE LINE NUMBER IN BASIC PROGRAM ERASE VARIABLES
1204 1205 1206 1207 1208 1209 1210	8A02 9E D1 8A04 9F 2B 8A06 7E AD 01 8A09 BD AD 26 8A0C CC 00 0A	L8AØ4 * RENUM	RTS LDX STX JMP JSR LDD	VD1 BINVAL LAD01 LAD26 #10	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR GO FIND THE LINE NUMBER IN BASIC PROGRAM ERASE VARIABLES DEFAULT LINE NUMBER INTERVAL
1204 1205 1206 1207 1208 1209 1210	8AØ2 9E D1 8AØ4 9F 2B 8AØ6 7E AD Ø1 8AØ9 BD AD 26 8AØC CC ØØ ØA 8AØF DD D5	L8AØ4 * RENUM	RTS LDX STX JMP JSR LDD STD	VD1 BINVAL LADØ1 LAD26 #10 VD5	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR GO FIND THE LINE NUMBER IN BASIC PROGRAM ERASE VARIABLES DEFAULT LINE NUMBER INTERVAL SAVE DEFAULT RENUMBER START LINE NUMBER
1204 1205 1206 1207 1208 1209 1210 1211 1212	8A02 9E D1 8A04 9F 2B 8A06 7E AD 01 8A09 BD AD 26 8A0F DD D5 8A11 DD CF	L8AØ4 * RENUM	RTS LDX STX JMP JSR LDD STD STD	VD1 BINVAL LAD01 LAD26 #10	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR GO FIND THE LINE NUMBER IN BASIC PROGRAM ERASE VARIABLES DEFAULT LINE NUMBER INTERVAL SAVE DEFAULT RENUMBER START LINE NUMBER SAVE DEFAULT INTERVAL
1204 1205 1206 1207 1208 1209 1210 1211 1212 1213	8A02 9E D1 8A04 9F 2B 8A06 7E AD 01 8A09 BD AD 26 8A0C CC 00 0A 8A0F DD D5 8A11 DD CF 8A13 5F	L8AØ4 * RENUM	RTS LDX STX JMP JSR LDD STD STD CLRB	VD1 BINVAL LAD01 LAD26 #10 VD5 VCF	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR GO FIND THE LINE NUMBER IN BASIC PROGRAM ERASE VARIABLES DEFAULT LINE NUMBER INTERVAL SAVE DEFAULT RENUMBER START LINE NUMBER SAVE DEFAULT INTERVAL NOW ACCD = Ø
1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214	8AØ2 9E D1 8AØ4 9F 2B 8AØ6 7E AD Ø1 8AØ9 BD AD 26 8AØC CC ØØ ØA 8AØF DD D5 8A11 DD CF 8A13 5F 8A14 DD D1	L8AØ4 * RENUM	RTS LDX STX JMP JSR LDD STD STD CLRB STD	VD1 BINVAL LADØ1 LAD26 #10 VD5 VCF	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR GO FIND THE LINE NUMBER IN BASIC PROGRAM ERASE VARIABLES DEFAULT LINE NUMBER INTERVAL SAVE DEFAULT RENUMBER START LINE NUMBER SAVE DEFAULT INTERVAL NOW ACCD = Ø DEFAULT LINE NUMBER OF WHERE TO START RENUMBERING
1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215	8A02 9E D1 8A04 9F 2B 8A06 7E AD 01 8A09 BD AD 26 8A0F DD D5 8A11 DD CF 8A13 5F 8A14 DD D1 8A16 9D A5	L8AØ4 * RENUM	RTS LDX STX JMP JSR LDD STD STD CLRB STD JSR	VD1 BINVAL LADØ1 LADØ6 #10 VD5 VCF VD1 GETCCH	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR GO FIND THE LINE NUMBER IN BASIC PROGRAM ERASE VARIABLES DEFAULT LINE NUMBER INTERVAL SAVE DEFAULT RENUMBER START LINE NUMBER SAVE DEFAULT INTERVAL NOW ACCD = 0 DEFAULT LINE NUMBER OF WHERE TO START RENUMBERING GET CURRENT INPUT CHARACTER
1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216	8A02 9E D1 8A04 9F 2B 8A06 7E AD 01 8A09 BD AD 26 8A0C CC 00 0A 8A0F DD D5 8A11 DD CF 8A13 5F 8A14 DD D1 8A16 9D A5 8A18 24 06	L8AØ4 * RENUM	RTS LDX STX JMP JSR LDD STD STD CLRB STD JSR BCC	VD1 BINVAL LAD01 LAD26 #10 VD5 VCF VD1 GETCCH L8A20	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR GO FIND THE LINE NUMBER IN BASIC PROGRAM ERASE VARIABLES DEFAULT LINE NUMBER INTERVAL SAVE DEFAULT INTERVAL NOW ACCD = Ø DEFAULT LINE NUMBER OF WHERE TO START RENUMBERING GET CURRENT INPUT CHARACTER BRANCH IF NOT NUMERIC
1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215	8A02 9E D1 8A04 9F 2B 8A06 7E AD 01 8A09 BD AD 26 8A0F DD D5 8A11 DD CF 8A13 5F 8A14 DD D1 8A16 9D A5	L8AØ4 * RENUM	RTS LDX STX JMP JSR LDD STD STD CLRB STD JSR BCC	VD1 BINVAL LADØ1 LADØ6 #10 VD5 VCF VD1 GETCCH	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR GO FIND THE LINE NUMBER IN BASIC PROGRAM ERASE VARIABLES DEFAULT LINE NUMBER INTERVAL SAVE DEFAULT RENUMBER START LINE NUMBER SAVE DEFAULT INTERVAL NOW ACCD = 0 DEFAULT LINE NUMBER OF WHERE TO START RENUMBERING GET CURRENT INPUT CHARACTER
1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218	8A02 9E D1 8A04 9F 2B 8A06 7F AD 01 8A09 BD AD 26 8A0F DD D5 8A11 DD CF 8A13 5F 8A14 DD D1 8A16 DD A5 8A18 24 06 8A18 8D EØ 8A18 8D EØ 8A18 9F D5	L8AØ4 * RENUM	RTS LDX STX JMP JSR LDD STD STD CLRB STD JSR BCC	VD1 BINVAL LADØ1 LAD26 #10 VD5 VCF VD1 GETCCH L8A20 L89FC	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR GO FIND THE LINE NUMBER IN BASIC PROGRAM ERASE VARIABLES DEFAULT LINE NUMBER INTERVAL SAVE DEFAULT INTERVAL NOW ACCD = Ø DEFAULT LINE NUMBER OF WHERE TO START RENUMBERING GET CURRENT INPUT CHARACTER BRANCH IF NOT NUMERIC
1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217	8AØ2 9E D1 8AØ4 9F 2B 8AØ6 7E AD Ø1 8AØ9 BD AD 26 8AØC CC ØØ ØA 8AØF DD D5 8A11 DD CF 8A13 5F 8A14 DD D1 8A16 9D A5 8A18 9D A5 8A18 8D EØ	L8AØ4 * RENUM	RTS LDX STX JMP JSR LDD STD STD CLRB STD JSR BCC BSR STX	VD1 BINVAL LADØ1 LAD26 #10 VD5 VCF VD1 GETCCH L8A20 L89FC	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR GO FIND THE LINE NUMBER IN BASIC PROGRAM ERASE VARIABLES DEFAULT LINE NUMBER INTERVAL SAVE DEFAULT RENUMBER START LINE NUMBER SAVE DEFAULT INTERVAL NOW ACCD = Ø DEFAULT LINE NUMBER OF WHERE TO START RENUMBERING GET CURRENT INPUT CHARACTER BRANCH IF NOT NUMBERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE LINE NUMBER WHERE RENUMBERING STARTS GET CURRENT INPUT CHARACTER
1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218	BAM2 9E D1 BAM9 9F 2B BAM6 7F AD Ø1 BAM9 BD AD 26 BAMC CC ØØ ØA BAMF DD D5 BA11 DD CF BA13 5F BA14 DD D1 BA16 9D A5 BA18 24 Ø6 BA1A 8D EØ BA1C 9F D5 BA1E 9D A5 BA1E 9D A5 BA1E 9D A5 BA1E 9D A5 BA1E 9D A5	L8AØ4 * RENUM	RTS LDX STX JMP JSR LDD STD STD CLRB STD JSR BCC BSR STX JSR	VD1 BINVAL LADØ1 LADØ6 #10 VD5 VCF VD1 GETCCH L8A2Ø L89FC VD5	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR GO FIND THE LINE NUMBER IN BASIC PROGRAM ERASE VARIABLES DEFAULT LINE NUMBER INTERVAL SAVE DEFAULT RENUMBER START LINE NUMBER SAVE DEFAULT INTERVAL NOW ACCD = 0 DEFAULT LINE NUMBER OF WHERE TO START RENUMBERING GET CURRENT INPUT CHARACTER BRANCH IF NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE LINE NUMBER WHERE RENUMBERING STARTS
1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218	8A02 9E D1 8A04 9F 2B 8A06 7E AD 01 8A09 BD AD 26 8A0C CC 00 0A 8A0F DD D5 8A11 DD CF 8A13 5F 8A14 DD D1 8A16 9D A5 8A18 24 06 8A1A 8D E0 8A1C 9F D5 8A1E 9D A5	L8AØ4 ★ RENUM RENUM	RTS LDX STX JMP JSR LDD STD STD CLRB STD JSR BCC BSR STX JSR BEQ	VD1 BINVAL LADØ1 LAD26 #10 VD5 VCF VD1 GETCCH L8A20 L89FC VD5 GETCCH	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR GO FIND THE LINE NUMBER IN BASIC PROGRAM ERASE VARIABLES DEFAULT LINE NUMBER INTERVAL SAVE DEFAULT RENUMBER START LINE NUMBER SAVE DEFAULT INTERVAL NOW ACCD = Ø DEFAULT LINE NUMBER OF WHERE TO START RENUMBERING GET CURRENT INPUT CHARACTER BRANCH IF NOT NUMBERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE LINE NUMBER WHERE RENUMBERING STARTS GET CURRENT INPUT CHARACTER
1204 1205 1206 1207 1208 1209 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220	BAM2 9E D1 BAM9 9F 2B BAM6 7F AD Ø1 BAM9 BD AD 26 BAMC CC ØØ ØA BAMF DD D5 BA11 DD CF BA13 5F BA14 DD D1 BA16 9D A5 BA18 24 Ø6 BA1A 8D EØ BA1C 9F D5 BA1E 9D A5 BA1E 9D A5 BA1E 9D A5 BA1E 9D A5 BA1E 9D A5	L8AØ4 ★ RENUM RENUM	JSR LDD STD STD CLRB STD JSR BCC BSR STX JSR BEQ JSR	VD1 BINVAL LADØ1 LADØ6 #10 VD5 VCF VD1 GETCCH L8A20 L89FC VD5 GETCCH L8A3D	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR GO FIND THE LINE NUMBER IN BASIC PROGRAM ERASE VARIABLES DEFAULT LINE NUMBER INTERVAL SAVE DEFAULT RENUMBER START LINE NUMBER SAVE DEFAULT INTERVAL NOW ACCD = Ø DEFAULT LINE NUMBER OF WHERE TO START RENUMBERING GET CURRENT INPUT CHARACTER BRANCH IF NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE LINE NUMBER WHERE RENUMBERING STARTS GET CURRENT INPUT CHARACTER BRANCH IF END OF LINE
1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221	8A02 9E D1 8A04 9F 2B 8A06 7E AD 01 8A09 BD AD 26 8A0C CC 00 0A 8A0F DD D5 8A11 DD CF 8A13 5F 8A14 DD D1 8A16 9D A5 8A18 24 06 8A1A 8D E0 8A1C 9F D5 8A1E 9D A5 8A2E DB BE 6D 8A22 BD BE 6D 8A25 24 06	L8AØ4 ★ RENUM RENUM	RTS LDX STX JMP JSR LDD STD CLRB STD JSR BCC BSR STX JSR BEQ JSR BEQ JSR BEQ BEQ BCC	VD1 BINVAL LADØ1 LADØ6 #1Ø VD5 VCF VD1 GETCCH LBA2Ø L89FC VD5 GETCCH L8A3D SYNCOMMA L8A2D	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR GO FIND THE LINE NUMBER IN BASIC PROGRAM ERASE VARIABLES DEFAULT LINE NUMBER INTERVAL SAVE DEFAULT RENUMBER START LINE NUMBER SAVE DEFAULT INTERVAL NOW ACCD = Ø DEFAULT LINE NUMBER OF WHERE TO START RENUMBERING GET CURRENT INPUT CHARACTER BRANCH IF NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE LINE NUMBER WHERE RENUMBERING STARTS GET CURRENT INPUT CHARACTER BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF EXT CHARACTER NOT NUMERIC
1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1223	BAM2 9E D1 BAM9 9F 2B BAM6 7F AD Ø1 BAM9 BD AD 26 BAM6 CC ØØ ØA BAMF DD D5 BA11 DD CF BA13 5F BA14 DD D1 BA16 9D A5 BA18 24 Ø6 BA1A 8D EØ BA1C 9F D5 BA1E 9D A5 BA1E 9D B2 BD B2 B	L8AØ4 ★ RENUM RENUM	RTS LDX STX JMP JSR LDD STD CLRB STD JSR BCC BSR STX JSR BEQ JSR BEQ JSR	VD1 BINVAL LADØ1 LADØ6 #10 VD5 VCF VD1 GETCCH L8A20 L89FC VD5 GETCCH L8A3D SYNCOMMA L8A2D L8A2D	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR GO FIND THE LINE NUMBER IN BASIC PROGRAM ERASE VARIABLES DEFAULT LINE NUMBER INTERVAL SAVE DEFAULT INTERVAL NOW ACCD = Ø DEFAULT LINE NUMBER OF WHERE TO START RENUMBERING GET CURRENT INPUT CHARACTER BRANCH IF NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE LINE NUMBER WHERE RENUMBERING STARTS GET CURRENT INPUT CHARACTER BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY
1204 1205 1206 1207 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1223	8A02 9E D1 8A04 9F 2B 8A06 7F AD 01 8A09 BD AD 26 8A0C CC 00 0A 8A0F DD D5 8A11 DD CF 8A13 5F 8A14 DD D1 8A16 9D A5 8A18 24 06 8A1A 8D E0 8A1C 9F D5 8A1E 9D A5 8A1E 9D A5 8A1E 9D A5 8A20 27 1B 8A20 27 1B 8A22 BD B2 6D 8A27 8D D3 8A29 9F D1	L8AØ4 ★ RENUM RENUM	RTS LDX STX JMP JSR LDD STD STD CLRB STD JSR BCC BSR STX JSR BEQ JSR BEQ JSR BEQ JSR BEQ JSR BCC BSR STX	VD1 BINVAL LADØ1 LADØ6 #10 VD5 VCF VD1 GETCCH L8A2Ø L89FC VD5 GETCCH L8A3D SYNCOMMA L8A2D L89FC VD1	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR GO FIND THE LINE NUMBER IN BASIC PROGRAM ERASE VARIABLES DEFAULT LINE NUMBER INTERVAL SAVE DEFAULT INTERVAL NOW ACCD = 0 DEFAULT LINE NUMBER OF WHERE TO START RENUMBERING GET CURRENT INPUT CHARACTER BRANCH IF NOT NUMBERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE LINE NUMBER WHERE RENUMBERING STARTS GET CURRENT INPUT CHARACTER BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER LINE
1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1223 1224	8A02 9E D1 8A04 9F 2B 8A06 7F AD 01 8A09 BD AD 26 8A0C CC 00 0A 8A0F DD D5 8A11 DD CF 8A13 5F 8A14 DD D1 8A16 9D A5 8A18 24 06 8A1A 8D E0 8A1A 8D E0 8A1C 9F D5 8A1E 9D A5 8A2E DB B2 6D 8A2Z BD B2 6D 8A2Z BD B2 6D 8A2Z 9T BB 8A2Z 9T BB 8A2Z 9F D1 8A2Z 9F D1 8A2Z 9F D1	* RENUM RENUM	RTS LDX STX JMP JSR LDD STD STD CLRB STD JSR BCC BSR STX JSR BCC BSR STX JSR	VD1 BINVAL LADØ1 LADØ6 #1Ø VD5 VCF VD1 GETCCH LBA2Ø L89FC VD5 GETCCH LBA3D SYNCOMMA L8A2D L89FC VD1 GETCCH L8A2D L89FC VD1 GETCCH UBA3D SYNCOMMA L8A2D L89FC VD1 GETCCH GETCCH GETCCH GETCCH GETCCH	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR GO FIND THE LINE NUMBER IN BASIC PROGRAM ERASE VARIABLES DEFAULT LINE NUMBER INTERVAL SAVE DEFAULT INTERVAL NOW ACCD = Ø DEFAULT LINE NUMBER OF WHERE TO START RENUMBERING GET CURRENT INPUT CHARACTER BRANCH IF NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE LINE NUMBER WHERE RENUMBERING STARTS GET CURRENT INPUT CHARACTER BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER LINE GET CURRENT INPUT CHARACTER
1204 1205 1206 1207 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226	BAM2 9E D1 BAM9 9F 2B BAM6 7F AD Ø1 BAM9 BD AD 26 BAM6 CC ØØ ØA BAM6 DD D5 BA11 DD CF BA13 5F BA14 DD D1 BA16 9D A5 BA18 24 Ø6 BA1A 8D EØ BA1C 9F D5 BA1E 9D A5 BA2Ø 27 1B BA22 BD B2 6D BA25 24 Ø6 BA27 8D D3 BA29 9F D1 BA29 9F D1 BA29 9F D1 BA29 9F D1 BA20 27 ØE	L8AØ4 ★ RENUM RENUM	RTS LDX STX JMP JSR LDD STD STD CLRB STD JSR BCC BSR STX JSR BCC BSR STX JSR BCC BSR STX JSR BCC BSR STX JSR BCC BSR STX JSR BCC BSR STX JSR BCC BSR BSR BCC BSR BCC BSR BCC BSR BSR BCC BSR BCC BSR BCC BSR BCC BSR BSR BCC BSR BSR BCC BSR BSR BSR BSR BSR BSR BSR BSR BSR BSR	VD1 BINVAL LADØ1 LADØ6 #10 VD5 VCF VD1 GETCCH L8A20 L89FC VD5 GETCCH L8A3D SYNCOMMA L8A2D L8A2D L8A9C VD1 GETCCH L8A3D	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR GO FIND THE LINE NUMBER IN BASIC PROGRAM ERASE VARIABLES DEFAULT LINE NUMBER INTERVAL SAVE DEFAULT INTERVAL NOW ACCD = Ø DEFAULT LINE NUMBER OF WHERE TO START RENUMBERING GET CURRENT INPUT CHARACTER BRANCH IF NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE LINE NUMBER WHERE RENUMBERING STARTS GET CURRENT INPUT CHARACTER BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER LINE GET CURRENT INPUT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER LINE GET CURRENT INPUT CHARACTER BRANCH IF END OF LINE
1204 1205 1206 1207 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226	8A02 9E D1 8A04 9F 2B 8A06 7F AD 01 8A09 BD AD 26 8A0C CC 00 0A 8A0F DD D5 8A11 DD CF 8A13 5F 8A14 DD D1 8A16 9D A5 8A18 24 06 8A1A 8D E0 8A1C 9F D5 8A1E 9D A5 8A1E 9D A5 8A20 27 1B 8A20 27 1B 8A22 BD B2 6D 8A25 24 06 8A27 8D D3 8A29 9F D1 8A29 9F D1 8A29 07 08 8A29 07 08 8A29 07 08	* RENUM RENUM	RTS LDX STX JMP JSR LDD STD STD JSR BCC LSR BCC BSR STX JSR BEQ JSR BCC BSR STX JSR BCC BSR STX JSR BCC BCC BCC BCC BCC BCC BCC BCC BCC BC	VD1 BINVAL LADØ1 LADØ6 #10 VD5 VCF VD1 GETCCH L8A2Ø L889FC VD5 GETCCH L8A3D SYNCOMMA L8A2D L8B9FC VD1 GETCCH L8B9FC VD1 GETCCH L8A3D SYNCOMMA	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR GO FIND THE LINE NUMBER IN BASIC PROGRAM ERASE VARIABLES DEFAULT LINE NUMBER INTERVAL SAVE DEFAULT INTERVAL NOW ACCD = 0 DEFAULT LINE NUMBER START LINE NUMBER GET CURRENT INPUT CHARACTER BRANCH IF NOT NUMBERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE LINE NUMBER WHERE RENUMBERING STARTS GET CURRENT INPUT CHARACTER BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER LINE GET CURRENT INPUT CHARACTER BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER LINE GET CURRENT INPUT CHARACTER BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA
1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1227 1228	8A02 9E D1 8A04 9F 2B 8A06 7F AD 01 8A09 BD AD 26 8A0C CC 00 0A 8A0F DD D5 8A11 DD CF 8A13 5F 8A14 DD D1 8A16 9D A5 8A18 24 06 8A1A 8D E0 8A1A 8D E0 8A1C 9F D5 8A1E 9D A5 8A22 BD B2 6D 8A27 BD B3 8A29 9F D1 8A28 9D A5 8A29 9F D1 8A29 9F D1 8A29 9F D1 8A29 9F D1 8A29 9F D1 8A29 9F D1 8A20 27 0E 8A20 27 0E 8A21 9D A5 8A22 D3 B2 6D	* RENUM RENUM	RTS LDX STX JMP JSR LDD STD STD STD JSR BSR STX JSR BEQ JSR BEQ JSR BEQ JSR BEQ JSR BEQ JSR BEQ JSR BEQ JSR BEQ JSR BEQ JSR BEQ JSR JSR JSR JSR JSR JSR JSR JSR JSR JSR	VD1 BINVAL LADØ1 LADØ6 #1Ø VD5 VCF VD1 GETCCH LBA2Ø L89FC VD5 GETCCH L8A3D SYNCOMMA L8A2D L89FC VD1 GETCCH L8A3D SYNCOMMA L8A3D SYNCOMMA L8A3D SYNCOMMA L8A3D SYNCOMMA	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR GO FIND THE LINE NUMBER IN BASIC PROGRAM ERASE VARIABLES DEFAULT LINE NUMBER INTERVAL SAVE DEFAULT INTERVAL NOW ACCD = Ø DEFAULT LINE NUMBER OF WHERE TO START RENUMBERING GET CURRENT INPUT CHARACTER BRANCH IF NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE LINE NUMBER WHERE RENUMBERING STARTS GET CURRENT INPUT CHARACTER BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER LINE GET CURRENT INPUT CHARACTER BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF END OF LINE SYNTAX CHECK TOR COMMA BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF NEXT CHARACTER NOT NUMERIC
1204 1205 1206 1207 1208 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228	8A02 9E D1 8A04 9F 2B 8A06 7F AD 01 8A09 BD AD 26 8A0C CC 00 0A 8A0F DD D5 8A11 DD CF 8A13 5F 8A14 DD D1 8A16 9D A5 8A18 24 06 8A1A 8D E0 8A1C 9F D5 8A1E 9D A5 8A1E 9D A5 8A20 27 1B 8A20 27 1B 8A22 BD B2 6D 8A25 24 06 8A27 8D D3 8A29 9F D1 8A29 9F D1 8A29 07 08 8A29 07 08 8A29 07 08	* RENUM RENUM	RTS LDX STX JMP JSR LDD STD CLRB STD JSR BEC BSR STJ JSR BEQ JSR BEQ JSR BEQ JSR BEQ JSR BEQ JSR BEQ JSR BEQ BEC BSR STJ STD BEC BSR STD BEC BSR STD BEC BSR BEC BSR BEC BSR BEC BSR BEC BSR BEC BSR BEC BSR BEC BSR BEC BSR BEC BSR BEC BSR BEC BSR BEC BSR BSR BEC BSR BSR BSR BSR BSR BSR BSR BSR BSR BSR	VD1 BINVAL LAD26 #10 VD5 VCF VD1 GETCCH L897C VD5 GETCCH L8A3D SYNCOMMA L8A2D L89FC VD1 GETCCH L8A3D SYNCOMMA L8A3D SYNCOMMA L8A3D SYNCOMMA	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR GO FIND THE LINE NUMBER IN BASIC PROGRAM ERASE VARIABLES DEFAULT LINE NUMBER INTERVAL SAVE DEFAULT INTERVAL NOW ACCD = 0 DEFAULT LINE NUMBER START LINE NUMBER GET CURRENT INPUT CHARACTER BRANCH IF NOT NUMBERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE LINE NUMBER WHERE RENUMBERING STARTS GET CURRENT INPUT CHARACTER BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER LINE GET CURRENT INPUT CHARACTER BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER LINE GET CURRENT INPUT CHARACTER BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA
1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1227 1228	8A02 9E D1 8A04 9F 2B 8A06 7F AD 01 8A09 BD AD 26 8A0C CC 00 0A 8A0F DD D5 8A11 DD CF 8A13 5F 8A14 DD D1 8A16 9D A5 8A18 24 06 8A1A 8D E0 8A1A 8D E0 8A1C 9F D5 8A1E 9D A5 8A22 BD B2 6D 8A27 BD B3 8A29 9F D1 8A28 9D A5 8A29 9F D1 8A29 9F D1 8A29 9F D1 8A29 9F D1 8A29 9F D1 8A29 9F D1 8A20 27 0E 8A20 27 0E 8A21 9D A5 8A22 D3 B2 6D	* RENUM RENUM	RTS LDX STX JMP JSR LDD STD STD STD JSR BSR STX JSR BEQ JSR BEQ JSR BEQ JSR BEQ JSR BEQ JSR BEQ JSR BEQ JSR BEQ JSR BEQ JSR BEQ JSR JSR JSR JSR JSR JSR JSR JSR JSR JSR	VD1 BINVAL LAD26 #10 VD5 VCF VD1 GETCCH L897C VD5 GETCCH L8A3D SYNCOMMA L8A2D L89FC VD1 GETCCH L8A3D SYNCOMMA L8A3D SYNCOMMA L8A3D SYNCOMMA	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR GO FIND THE LINE NUMBER IN BASIC PROGRAM ERASE VARIABLES DEFAULT LINE NUMBER INTERVAL SAVE DEFAULT INTERVAL NOW ACCD = Ø DEFAULT LINE NUMBER OF WHERE TO START RENUMBERING GET CURRENT INPUT CHARACTER BRANCH IF NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE LINE NUMBER WHERE RENUMBERING STARTS GET CURRENT INPUT CHARACTER BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER LINE GET CURRENT INPUT CHARACTER BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF END OF LINE SYNTAX CHECK TOR COMMA BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF NEXT CHARACTER NOT NUMERIC
1204 1205 1206 1207 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229	BAM2 9E D1 BAM9 9F 2B BAM6 7F AD Ø1 BAM9 BD AD 26 BAM6 CC ØØ ØA BAM6 DD D5 BA11 DD CF BA13 5F BA14 DD D1 BA16 9D A5 BA18 24 Ø6 BA16 9F D5 BA18 24 Ø6 BA16 9F D5 BA18 9D A5 BA20 27 1B BA22 BD B2 6D BA25 24 Ø6 BA27 8D D3 BA29 9F D1 BA28 9D A5 BA29 9F D1 BA28 PD A5 BA20 27 ØE BA27 BD B2 6D BA32 24 Ø6 BA32 24 Ø6 BA32 24 Ø6 BA33 24 Ø6 BA33 24 Ø6 BA34 BD C6	* RENUM RENUM	RTS LDX STX JMP JSR LDD STD CLRB STD JSR BEC BSR STX JSR BEQ JSR BEQ JSR BEQ JSR BEQ JSR BEQ STX JSR STX STX STX STX STX STX STX STX STX STX	VD1 BINVAL LADØ1 LADØ6 #10 VD5 VCF VD1 GETCCH L8A2Ø L889FC VD5 GETCCH L8A3D SYNCOMMA L8A2D L8B9FC VD1 GETCCH L8A3A L89FC VD1 SYNCOMMA	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR GO FIND THE LINE NUMBER IN BASIC PROGRAM ERASE VARIABLES DEFAULT LINE NUMBER INTERVAL SAVE DEFAULT INTERVAL NOW ACCD = 0 DEFAULT LINE NUMBER START LINE NUMBER GET CURRENT INPUT CHARACTER BRANCH IF NOT NUMBERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE LINE NUMBER WHERE RENUMBERING STARTS GET CURRENT INPUT CHARACTER BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER LINE GET CURRENT INPUT CHARACTER BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER LINE GET CURRENT INPUT CHARACTER BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF FEND OF LINE SYNTAX CHECK FOR COMMA BRANCH IF FEND OF LINE SYNTAX CHECK FOR COMMA BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW INTERVAL
1204 1205 1206 1207 1208 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229 1229 1230	8A02 9E D1 8A04 9F 2B 8A06 7F AD 01 8A09 BD AD 26 8A0C CC 00 0A 8A0F DD D5 8A11 DD CF 8A13 5F 8A14 DD D1 8A16 9D A5 8A18 24 06 8A1A 8D E0 8A1A 8D E0 8A1C 9F D5 8A1E 9D A5 8A22 BD B2 6D 8A25 24 06 8A27 8D D3 8A29 9F D1 8A28 9D A5 8A29 9F D1 8A29 9F D1	± RENUM RENUM LSA2Ø LSA2D	RTS LDX STM JSR LDD STD CCLRB STD BCC BSR BCC BSR BCC BSR JSR BCC BSR JSR BCC BSR JSR BCC BSR STJ JSR BCC BSR STJ STJ STJ STJ STJ STJ STJ STJ STJ STJ	VD1 BINVAL LADØ1 LAD26 #1Ø VD5 VCF VD1 GETCCH L8A2Ø L89FC VD5 GETCCH L8A3D SYNCOMMA L8A2CD L89FC VD1 GETCCH L8A3A L89FC VCF L8A3A	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR GO FIND THE LINE NUMBER IN BASIC PROGRAM ERASE VARIABLES DEFAULT LINE NUMBER INTERVAL SAVE DEFAULT INTERVAL NOW ACCD = Ø DEFAULT LINE NUMBER START LINE NUMBER SAVE DEFAULT INTERVAL NOW ACCD = Ø DEFAULT LINE NUMBER OF WHERE TO START RENUMBERING GET CURRENT INPUT CHARACTER BRANCH IF NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE LINE NUMBER WHERE RENUMBERING STARTS GET CURRENT INPUT CHARACTER BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER LINE GET CURRENT INPUT CHARACTER BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF END OF LINE SYNTAX CHECK THARACTER BRANCH IF NEXT CHARACTER BRANCH IF CHARACTER BRANCH IF NEXT CHARACTER BRANCH IT NUMBER IN BASIC PROGRAM TO BINARY BRANCH IT NUMBER IN BASIC PROGRAM TO BINARY BRANCH IT
1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229 1230 1231	BAM2 9E D1 BAM9 9F 2B BAM6 7F AD Ø1 BAM9 BD AD 26 BAM6 CC ØØ ØA BAM6 DD D5 BA11 DD CF BA13 5F BA14 DD D1 BA16 9D A5 BA18 24 Ø6 BA16 9D A5 BA18 24 Ø6 BA16 9D A5 BA18 9D A5 BA20 27 1B BA22 BD B2 6D BA25 24 Ø6 BA27 8D D3 BA29 9F D1 BA28 9D A5 BA29 9F D1 BA28 9D A5 BA29 9F D1 BA28 BD B2 6D BA36 BD C6 BA36 9F CF BA34 BD C6 BA36 9F CF BA38 BD A5 C7	* RENUM RENUM LBA2Ø LBA2D LBA3A	RTS LDX JMP JSR LDD STD STD STD JSR BCC JSR BCC JSR BCQ JSR BCQ JSR BCQ JSR BCQ JSR BCQ JSR STD JSR D JSR D JSR D JSR D JSR D JSR D JSR D JSR D D D D D D D D D D D D D D D D D D D	VD1 BINVAL LAD26 #10 VD5 VCF VD1 GETCCH L892C VD5 GETCCH L8A3D SYNCOMMA L8A2D L89FC VD1 GETCCH L8A3D SYNCOMMA L8A2D L89FC VD1 GETCH L8A3D SYNCOMMA L8A3D SYNCOMMA L8A3D SYNCOMMA L8A3D SYNCOMMA L8A3D SYNCOMMA	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR GO FIND THE LINE NUMBER IN BASIC PROGRAM ERASE VARIABLES DEFAULT LINE NUMBER INTERVAL SAVE DEFAULT INTERVAL NOW ACCD = Ø DEFAULT LINE NUMBER OF WHERE TO START RENUMBERING GET CURRENT INPUT CHARACTER BRANCH IF NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE LINE NUMBER WHERE RENUMBERING STARTS GET CURRENT INPUT CHARACTER BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER LINE GET CURRENT INPUT CHARACTER BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER LINE GET CURRENT INPUT CHARACTER BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER IN BASIC PROGRAM TO BINARY SAVE NEW INTERVAL FC' ERROR
1204 1205 1206 1207 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229 1230 1231 1231 1232 1233	8A02 9E D1 8A04 9F 2B 8A06 7F AD 01 8A09 BD AD 26 8A0C CC 00 0A 8A0F DD D5 8A11 DD CF 8A13 5F 8A14 DD D1 8A16 9D A5 8A18 24 06 8A1A 8D E0 8A1C 9F D5 8A1E 9D A5 8A1E 9D A5 8A20 27 1B 8A20 27 1B 8A22 BD B2 6D 8A25 24 06 8A27 8D D3 8A29 9F D1 8A29 9F D1 8A29 9F D1 8A29 BD B2 6D 8A32 8D B3 8A29 9F D1 8A29 9F D1 8A30 8D C6 8A34 8D C6 8A36 9F CF 8A38 27 49 8A3A BD A5 C7 8A30 8D C3	± RENUM RENUM LSA2Ø LSA2D	RTS LDX JMP JSR LDD STD STD STD JSR BCC STX JSR BCQ JSR BCS BSS STX JSR BCQ JSR BCG STX JSR BCG STX JSR BCG STX JSR BCG BCG JSR BCG JSR BCG BCG JSR BCG BCG BCG BCG BCG BCG BCG BCG BCG BCG	VD1 BINVAL LAD26 #10 VD5 VCF VD1 GETCCH L8A20 L89FC VD5 GETCCH L8A3D SYNCOMMA L8A2D L89FC VD1 GETCCH L8A3D SYNCOMMA L8A2D L89FC VD1 GETCCH L8A3D SYNCOMMA L8A2D L89FC VD1 GETCCH L8A3D SYNCOMMA L8A3A L89FC VCF L8A3A L89FC VCF L8A3B L85C7 L8A62	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR GO FIND THE LINE NUMBER IN BASIC PROGRAM ERASE VARIABLES DEFAULT LINE NUMBER INTERVAL SAVE DEFAULT RENUMBER START LINE NUMBER SAVE DEFAULT INTERVAL NOW ACCD = Ø DEFAULT LINE NUMBER OF WHERE TO START RENUMBERING GET CURRENT INPUT CHARACTER BRANCH IF NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE LINE NUMBER WHERE RENUMBERING STARTS GET CURRENT INPUT CHARACTER BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER LINE GET CURRENT INPUT CHARACTER BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF FEND OF LINE SYNTAX CHECK FOR COMMA BRANCH IF FEND OF LINE SYNTAX CHECK FOR COMMA BRANCH IF FEND OF LINE SYNTAX CHECK FOR COMMA BRANCH IF FEND OF LINE SYNTAX CHECK FOR COMMA BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW INTERVAL FC' ERROR CHECK FOR MORE CHARACTERS ON LINE - SYNTAX ERROR IF ANY GO GET ADDRESS OF OLD NUMBER BEING RENUMBERED
1204 1205 1206 1207 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229 1231 1231 1231 1232 1231 1231	BAM2 9E D1 BAM9 9F 2B BAM06 7E AD 01 BAM9 BD AD 26 BAM0C CC 00 0A BAM6 DD D5 BA11 DD CF BA13 5F BA14 DD D1 BA16 9D A5 BA18 24 06 BA1A 8D E0 BA1C 9F D5 BA1E 9D A5 BA1E 9D A5 BA1E 9D A5 BA2E 27 1B BA22 2B B2 6D BA26 27 1B BA27 BD D3 BA29 9F D1 BA28 9F D5 BA28 7 0E BA27 8D D3 BA28 9F C7 BA38 27 49 BA3A BD A5 BA3A BD A5 BA3B D A5	* RENUM RENUM LBA2Ø LBA2D LBA3A	RTS LDX JMP JSR LDD STD STD STD JSR BCC BSR STX JSR BEQ JSR BEQ JSR BEQ JSR BEQ JSR BEQ JSR STJ STJ STJ STJ STJ STJ STJ STD STD STD STD STD STD STD STD STD STD	VD1 BINVAL LADØ1 LADØ6 #10 VD5 VCF VD1 GETCCH L8A20 L89FC VD5 GETCCH L8A3D SYNCOMMA L8A2D L89FC VD1 GETCCH L8A3D SYNCOMMA L8A2D L89FC VD1 GETCCH L8A3D SYNCOMMA L8A3D L8COT L8A02 VD3	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR GO FIND THE LINE NUMBER IN BASIC PROGRAM ERASE VARIABLES DEFAULT LINE NUMBER INTERVAL SAVE DEFAULT INTERVAL NOM ACCD = Ø DEFAULT LINE NUMBER OF WHERE TO START RENUMBERING GET CURRENT INPUT CHARACTER BRANCH IF NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE LINE NUMBER WHERE RENUMBERING STARTS GET CURRENT INPUT CHARACTER BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER LINE GET CURRENT INPUT CHARACTER BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER LINE BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW INTERVAL FC' ERROR CHECK FOR MORE CHARACTERS ON LINE - SYNTAX ERROR IF ANY GO GET ADDRESS OF OLD NUMBER BEING RENUMBERED SAVE ADDRESS
1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229 1230 1231 1232 1233 1234	BAM2 9E D1 BAM9 9F 2B BAM06 7F AD Ø1 BAM9 BD AD 26 BAM0C CC ØØ ØA BAM0F DD D5 BA11 DD CF BA13 5F BA14 DD D1 BA16 9D A5 BA18 24 Ø6 BA16 9D A5 BA18 BD EØ BA10 BD EØ BA12 BD B2 6D BA25 24 Ø6 BA26 27 BB BA29 9F D1 BA28 9D A5 BA29 9F D1 BA28 BD B2 6D BA27 BD D3 BA29 9F D1 BA28 BD B3 BA29 9F D1 BA28 BD B3 BA29 9F D1 BA38 BD A5 BA30 27 ØE BA31 BD C3 BA33 BD C5 BA33 BD C5 BA33 BD C3 BA34 9F D3 BA34 PD D3 BA34 PD D3 BA34 PF D3 BA34 PD C3	* RENUM RENUM LBA2Ø LBA2D LBA3A	RTS LDX JMP JSR LDD STD STD STD JSR BCC JSR BCC JSR BCQ JSR BCQ JSR BCQ JSR BCQ JSR BCQ JSR BCQ JSR BCQ STD JSR BCQ STD JSR D JSR D JSR D JSR D JSR D JSR D JSR D JSR D JS D D D D D D D D D D D D D D D D D	VD1 BINVAL LAD26 #10 VD5 VCF VD1 GETCCH L89FC VD5 GETCCH L8A3D SYNCOMMA L8A2D L88FC VD1 GETCCH L8A3D SYNCOMMA L8A2D L88FC VD1 CETCCH L8A3D SYNCOMMA L8A2D L88FC VD1 L8A3D SYNCOMMA L8A3A L8A83 L8A6C L8A83 L8A6C L8A82 VD3 VD5	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR GO FIND THE LINE NUMBER IN BASIC PROGRAM ERASE VARIABLES DEFAULT LINE NUMBER INTERVAL SAVE DEFAULT RENUMBER START LINE NUMBER SAVE DEFAULT INTERVAL NOW ACCD = Ø DEFAULT LINE NUMBER OF WHERE TO START RENUMBERING GET CURRENT INPUT CHARACTER BRANCH IF NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE LINE NUMBER WHERE RENUMBERING STARTS GET CURRENT INPUT CHARACTER BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER LINE GET CURRENT INPUT CHARACTER BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER LINE GET CURRENT INPUT CHARACTER BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW INTERVAL FC' ERROR CHECK FOR MORE CHARACTERS ON LINE - SYNTAX ERROR IF ANY GO GET ADDRESS OF OLD NUMBER BEING RENUMBERED SAVE ADDRESS GET NEXT RENUMBERED LINE NUMBER TO USE
1204 1205 1206 1207 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229 1230 1231 1233 1234 1233 1234 1233 1234 1235	8A02 9E D1 8A04 9F 2B 8A06 7E AD 01 8A09 BD AD 26 8A0C CC 00 0A 8A0F DD D5 8A11 DD CF 8A13 5F 8A14 DD D1 8A16 9D AS 8A18 24 06 8A18 BE 80 8A16 9F D5 8A1E 9D AS 8A20 27 1B 8A22 BD B2 6D 8A25 24 06 8A27 8D D3 8A29 9F D1 8A28 9D A5 8A29 9F D1 8A30 BD A5 8A34 BD C6 8A34 BD C6 8A34 BD C6 8A34 BD C6 8A35 PF D3 8A36 BD A5 C7 8A30 BD C3 8A34 9F D3 8A41 9E D5	* RENUM RENUM LBA2Ø LBA2D LBA3A	RTS LDX JMP JSR LDD STD STD STD JSR BEC BSR STX JSR BEQ JSR BEQ JSR BEQ JSR BEQ JSR BEQ JSR STX LDX STX BEQ JSR BEC BSR STX STX STX STX STX STX STX STX STX STX	VD1 BINVAL LADØ1 LADØ6 #1Ø VD5 VCF VD1 GETCCH LBA2Ø L89FC VD5 GETCCH L8A3D SYNCOMMA L8A2D L89FC VD1 GETCCH L8A3D SYNCOMMA L8A2D L89FC VD1 GETCCH L8A3D SYNCOMMA L8A2D L89FC VD1 L8A3A L89FC VCF L8A83 L85FC VCF L8A83 LASC7 L8A82 VD3 VD5 L8A04	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR GO FIND THE LINE NUMBER IN BASIC PROGRAM ERASE VARIABLES DEFAULT LINE NUMBER INTERVAL SAVE DEFAULT INTERVAL NOW ACCD = Ø DEFAULT LINE NUMBER START LINE NUMBER SAVE DEFAULT INTERVAL NOW ACCD = Ø DEFAULT LINE NUMBER OF WHERE TO START RENUMBERING GET CURRENT INPUT CHARACTER BRANCH IF NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE LINE NUMBER WHERE RENUMBERING STARTS GET CURRENT INPUT CHARACTER BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER LINE GET CURRENT INPUT CHARACTER BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER LINE GET CURRENT INPUT CHARACTER BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW INTERVAL FC' ERROR CHECK FOR MORE CHARACTERS ON LINE - SYNTAX ERROR IF ANY GO GET ADDRESS OF OLD NUMBER BEING RENUMBERED SAVE ADDRESS GET NEXT RENUMBERED LINE NUMBER TO USE FIND THE LINE NUMBER IN THE BASIC PROGRAM
1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229 1230 1231 1232 1233 1234	BAM2 9E D1 BAM9 9F 2B BAM06 7F AD Ø1 BAM9 BD AD 26 BAM0C CC ØØ ØA BAM0F DD D5 BA11 DD CF BA13 5F BA14 DD D1 BA16 9D A5 BA18 24 Ø6 BA16 9D A5 BA18 BD EØ BA10 BD EØ BA12 BD B2 6D BA25 24 Ø6 BA26 27 BB BA29 9F D1 BA28 9D A5 BA29 9F D1 BA28 BD B2 6D BA27 BD D3 BA29 9F D1 BA28 BD B3 BA29 9F D1 BA28 BD B3 BA29 9F D1 BA38 BD A5 BA30 27 ØE BA31 BD C3 BA33 BD C5 BA33 BD C5 BA33 BD C3 BA34 9F D3 BA34 PD D3 BA34 PD D3 BA34 PF D3 BA34 PD C3	* RENUM RENUM LBA2Ø LBA2D LBA3A	RTS LDX JMP JSR LDD STD STD STD JSR BCC JSR BCC JSR BCQ JSR BCQ JSR BCQ JSR BCQ JSR BCQ JSR BCQ JSR BCQ STD JSR BCQ STD JSR D JSR D JSR D JSR D JSR D JSR D JSR D JSR D JS D D D D D D D D D D D D D D D D D	VD1 BINVAL LADØ1 LADØ6 #1Ø VD5 VCF VD1 GETCCH LBA2Ø L89FC VD5 GETCCH L8A3D SYNCOMMA L8A2D L89FC VD1 GETCCH L8A3D SYNCOMMA L8A2D L89FC VD1 GETCCH L8A3D SYNCOMMA L8A2D L89FC VD1 L8A3A L89FC VCF L8A83 L85FC VCF L8A83 LASC7 L8A82 VD3 VD5 L8A04	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR GO FIND THE LINE NUMBER IN BASIC PROGRAM ERASE VARIABLES DEFAULT LINE NUMBER INTERVAL SAVE DEFAULT RENUMBER START LINE NUMBER SAVE DEFAULT INTERVAL NOW ACCD = Ø DEFAULT LINE NUMBER OF WHERE TO START RENUMBERING GET CURRENT INPUT CHARACTER BRANCH IF NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE LINE NUMBER WHERE RENUMBERING STARTS GET CURRENT INPUT CHARACTER BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER LINE GET CURRENT INPUT CHARACTER BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER LINE GET CURRENT INPUT CHARACTER BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW INTERVAL FC' ERROR CHECK FOR MORE CHARACTERS ON LINE - SYNTAX ERROR IF ANY GO GET ADDRESS OF OLD NUMBER BEING RENUMBERED SAVE ADDRESS GET NEXT RENUMBERED LINE NUMBER TO USE
1204 1205 1206 1207 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229 1230 1231 1233 1234 1233 1234 1233 1234 1235	8A02 9E D1 8A04 9F 2B 8A06 7E AD 01 8A09 BD AD 26 8A0C CC 00 0A 8A0F DD D5 8A11 DD CF 8A13 5F 8A14 DD D1 8A16 9D AS 8A18 24 06 8A18 BE 80 8A16 9F D5 8A1E 9D AS 8A20 27 1B 8A22 BD B2 6D 8A25 24 06 8A27 8D D3 8A29 9F D1 8A28 9D A5 8A29 9F D1 8A30 BD A5 8A34 BD C6 8A34 BD C6 8A34 BD C6 8A34 BD C6 8A35 PF D3 8A36 BD A5 C7 8A30 BD C3 8A34 9F D3 8A41 9E D5	* RENUM RENUM LBA2Ø LBA2D LBA3A	RTS LDX JMP JSR LDD STD STD STD STD JSR BCC BSR STX JSR BEQ JSR BEQ JSR BEQ JSR BEQ JSR BEQ JSR STX BEQ JSR BEQ JSR STX STX STX STX STX STX STX STX STX STX	VD1 BINVAL LADØ1 LADØ6 #1Ø VD5 VCF VD1 GETCCH LBA2Ø L89FC VD5 GETCCH L8A3D SYNCOMMA L8A2D L89FC VD1 GETCCH L8A3D SYNCOMMA L8A2D L89FC VD1 GETCCH L8A3D SYNCOMMA L8A2D L89FC VD1 L8A3A L89FC VCF L8A83 L85FC VCF L8A83 LASC7 L8A82 VD3 VD5 L8A04	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR GO FIND THE LINE NUMBER IN BASIC PROGRAM ERASE VARIABLES DEFAULT LINE NUMBER INTERVAL SAVE DEFAULT INTERVAL NOW ACCD = Ø DEFAULT LINE NUMBER START LINE NUMBER SAVE DEFAULT INTERVAL NOW ACCD = Ø DEFAULT LINE NUMBER OF WHERE TO START RENUMBERING GET CURRENT INPUT CHARACTER BRANCH IF NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE LINE NUMBER WHERE RENUMBERING STARTS GET CURRENT INPUT CHARACTER BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER LINE GET CURRENT INPUT CHARACTER BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER LINE GET CURRENT INPUT CHARACTER BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW INTERVAL FC' ERROR CHECK FOR MORE CHARACTERS ON LINE - SYNTAX ERROR IF ANY GO GET ADDRESS OF OLD NUMBER BEING RENUMBERED SAVE ADDRESS GET NEXT RENUMBERED LINE NUMBER TO USE FIND THE LINE NUMBER IN THE BASIC PROGRAM
1204 1205 1206 1207 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229 1231 1231 1231 1231 1232 1231 1231	BAM2 9E D1 BAM9 9F 2B BAM06 7E AD 01 BAM9 BD AD 26 BAM0C CC 00 0A BAM6 DD D5 BA11 DD CF BA13 5F BA14 DD D1 BA16 9D A5 BA18 24 06 BA18 8D E0 BA12 9F D5 BA18 9D A5 BA19 DD B5 BA19 DD B5 BA19 DA5 BA19 DA5 BA19 DA5 BA10 9F D5 BA25 24 06 BA27 8D D3 BA29 9F D1 BA28 9F D1 BA28 9F D1 BA28 9F D1 BA28 BD A5 BA30 BC G5 BA31 BD C6 BA33 27 49 BA33 BD C3 BA31 BD C3 BA33 BD C3 BA31 BD C5 BA33 BD C3 BA31 BD	* RENUM RENUM LBA2Ø LBA2D LBA3A	RTS LDX JMP JSR LDD STD STD STD JSR BCC JSR BCC JSR BCQ JSR BCQ JSR BCQ JSR BCQ JSR BCQ JSR BCQ JSR BCQ STX JSR BCQ STX JSR BCQ STX JSR BCQ STX JSR DS DS DS DS DS DS DS DS DS DS DS DS DS	VD1 BINVAL LADØ1 LADØ6 #10 VD5 VCF VD1 GETCCH LBA20 LB9FC VD5 GETCCH LBA3D SYNCOMMA LBA2D LB9FC VD1 GETCCH LBA3D SYNCOMMA LBA2D LB9FC VD1 LBA3D SYNCOMMA LBA2D LB9FC VD1 LBA3D SYNCOMMA	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR GO FIND THE LINE NUMBER IN BASIC PROGRAM ERASE VARIABLES DEFAULT LINE NUMBER INTERVAL SAVE DEFAULT RENUMBER START LINE NUMBER SAVE DEFAULT INTERVAL NOW ACCD = Ø DEFAULT LINE NUMBER OF WHERE TO START RENUMBERING GET CURRENT INPUT CHARACTER BRANCH IF NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE LINE NUMBER WHERE RENUMBERING STARTS GET CURRENT INPUT CHARACTER BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER LINE GET CURRENT INPUT CHARACTER BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER LINE GET CURRENT INPUT CHARACTER BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW INTERVAL FC' ERROR CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW INTERVAL FC' ERROR CHECK FOR MORE CHARACTERS ON LINE - SYNTAX ERROR IF ANY GO GET ADDRESS OF OLD NUMBER BEING RENUMBERED SAVE ADDRESS GET NEXT RENUMBERED LINE NUMBER TO USE FIND THE LINE NUMBER IN THE BASIC PROGRAM COMPARE TO ADDRESS OF OLD LINE NUMBER FC ERROR IF NEW ADDRESS < OLD ADDRESS
1204 1205 1206 1207 1208 1219 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229 1230 1231 1232 1233 1234 1235 1236 1237 1238	8A02 9E D1 8A04 9F 2B 8A06 7F AD 01 8A09 BD AD 26 8A0C CC 00 0A 8A0F DD D5 8A13 DD CF 8A13 5F 8A14 DD D1 8A16 9D AS 8A18 24 06 8A1A 8D E0 8A12 9F D5 8A18 9D A5 8A18 9D A5 8A20 27 1B 8A20 27 1B 8A22 BD B2 6D 8A25 24 06 8A27 8D D3 8A29 9F D1 8A28 9D A5 8A27 BD B2 8A27 BD B2 8A28 9D A5 8A29 9F D1 8A38 BD A5 8A39 BD A5 8A34 BD B6 8A34 BD C6 8A34 BD C7 8A30 BD A5 C7 8A30 BD A5 C7 8A30 BD C3 8A37 9F D3 8A41 9E D5 8A43 8D BF 8A44 9C D3 8A47 25 3A 8A47 25 3A	* RENUM RENUM LBA2Ø LBA2D LBA3A	RTS LDX JMP JSR LDD STD STD STD JSR BEC BSR STX JSR BEQ JSR BEQ JSR BEQ JSR BEQ JSR BEQ JSR STX BEQ JSR BEC BSR STX BEQ BSR STX BEQ BSR STX BEC BSR STX BEC BSR STX BEC BSR BSR BEC BSR BEC BSR BEC BSR BEC BSR BEC BSR BEC BSR BEC BSR BEC BSR BEC BSR BEC BSR BEC BSR BEC BSR BEC BSR BEC BSR BEC BSR BEC BSR BEC BSR BSR BEC BSR BSR BEC BSR BSR BSR BSR BSR BSR BSR BSR BSR BSR	VD1 BINVAL LAD26 #10 VD5 VCF VD1 GETCCH LBA20 L89FC VD5 GETCCH L8A30 SYNCOMMA L8A2D L89FC VD1 GETCCH L8A30 SYNCOMMA L8A2D L89FC VD1 GETCCH L8A30 SYNCOMMA L8A2D L89FC VD1 GETCCH L8A30 SYNCOMMA L8A30 L8A67	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR GO FIND THE LINE NUMBER IN BASIC PROGRAM ERASE VARIABLES DEFAULT LINE NUMBER INTERVAL SAVE DEFAULT INTERVAL NOW ACCD = Ø DEFAULT LINE NUMBER START LINE NUMBER SAVE DEFAULT INTERVAL NOW ACCD = Ø DEFAULT LINE NUMBER OF WHERE TO START RENUMBERING GET CURRENT INPUT CHARACTER BRANCH IF NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE LINE NUMBER WHERE RENUMBERING STARTS GET CURRENT INPUT CHARACTER BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF END OF LINE GET CURRENT INPUT CHARACTER DEANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW INTERVAL FC' ERROR CHECK FOR MORE CHARACTERS ON LINE - SYNTAX ERROR IF ANY GO GET ADDRESS OF OLD NUMBER BEING RENUMBERED SAVE ADDRESS GET NEXT RENUMBERED LINE NUMBER TO USE FIND THE LINE NUMBER IN THE BASIC PROGRAM COMPARE TO ADDRESS OF OLD LINE NUMBER FC ERROR IF NEW ADDRESS < OLD ADDRESS MAKE SURE RENUMBERED LINE NUMBERS SILL BE IN RANGE
1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229 1230 1231 1232 1231 1232 1233 1234 1235 1236 1237 1238 1239 1240	BAM2 9E D1 BAM9 9F 2B BAM6 7F AD Ø1 BAM9 BD AD 26 BAM6 CC ØØ ØA BAM6 DD D5 BA11 DD CF BA13 5F BA14 DD D1 BA16 9D A5 BA18 24 Ø6 BA1A 8D EØ BA1C 9F D5 BA1E 9D A5 BA2E 27 1B BA22 BD B2 6D BA22 BD B2 6D BA26 27 1B BA27 BD B3 BA29 9F D1 BA28 9D A5 BA39 9F D1 BA28 9D A5 BA30 03 BA49 9F D1 BA28 9D A5 BA30 C7 ØE BA31 BD C6 BA33 C7 ØE BA34 BD C6 BA34 BD C6 BA34 BD C6 BA34 BD C7 BA38 BD A5 C7 BA3B BD C3 BA31 BD C3 BA44 BD BB	* RENUM RENUM LBA2Ø LBA2D LBA3A	RTS LDX JSR LDD STD STD STD JSR BCC BSR STX JSR BEQ D BEQ D BE D BE	VD1 BINVAL LAD26 #10 VD5 VCF VD1 GETCCH L8A20 L89FC VD5 GETCCH L8A3D SYNCOMMA L8A2D L89FC VD1 GETCCH L8A3D SYNCOMMA L8A2D L89FC VD1 GETCCH L8A3D SYNCOMMA L8A2D L8OFC VD1 GETCCH L8A3D SYNCOMMA L8A3D SYNCOMMA L8A3D SYNCOMMA L8A3A L8OFC VD1 L8A3B L8CC7 L8A02 VD3 VD5 L8A04 VD3 VD5 L8A04 VD3 L8A04 L8A07 L8A07 L8A07 L8A07 L8A07 L8A01	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR GO FIND THE LINE NUMBER IN BASIC PROGRAM ERASE VARIABLES DEFAULT LINE NUMBER INTERVAL SAVE DEFAULT INTERVAL NOW ACCD = Ø DEFAULT LINE NUMBER OF WHERE TO START RENUMBERING GET CURRENT INPUT CHARACTER BRANCH IF NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE LINE NUMBER WHERE RENUMBERING STARTS GET CURRENT INPUT CHARACTER BRANCH IF NOT OF LINE SYNTAX CHECK FOR COMMA BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER LINE GET CURRENT INPUT CHARACTER BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER LINE GET CURRENT INPUT CHARACTER BRANCH IF NEXT CHARACTER BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW INTERVAL FC' ERROR CHECK FOR MORE CHARACTERS ON LINE - SYNTAX ERROR IF ANY GO GET ADDRESS OF OLD NUMBER BEING RENUMBERED SAVE ADDRESS GET NEXT RENUMBERED LINE NUMBER TO USE FIND THE LINE NUMBER IN THE BASIC PROGRAM COMPARE TO ADDRESS OF OLD LINE NUMBER FC ERROR IF NEW ADDRESS < OLD ADDRESS MAKE SURE RENUMBERED LINE NUMBERS TO EXPANDED BINARY
1204 1205 1206 1207 1208 1210 1211 1212 1213 1214 1215 1216 1227 1228 1229 1230 1231 1232 1233 1234 1235 1236 1237 1238 1238 1239 1241	BAM2 9E D1 BAM9 F ZB BAM6 7F AD Ø1 BAM9 BD AD 26 BAM6 CC ØØ ØA BAM6 DD D5 BA11 DD CF BA13 5F BA14 DD D1 BA16 9D A5 BA18 B4 Ø6 BA18 B4 Ø6 BA18 B7 B5 BA18 B7 B7 B5 BA19 B7 B7 B7 BA19 B7 BA19 B7 BA19 B7 BA19 B7 BA29 B7 BA20 B7 BA30 B7 BA30 B7 BA30 B7 BA30 B7 BA31 B7 BA33 B7 BA34 B7 BA34 B7 BA35 B7 BA36 B7 BA36 B7 BA37 B7 BA38 B7 BA39 B7 BA39 B7 BA41 B7 BA45 B7 BA45 B7 BA46 B7 B	* RENUM RENUM LBA2Ø LBA2D LBA3A	RTS LDX JSR LDD STD STD STD JSR BCC JSR BCC JSR BCQ BCQ BCQ BCQ BCQ BCQ BCQ BCQ BCQ BCQ	VD1 BINVAL LAD26 #10 VD5 VCF VD1 GETCCH LBA20 LB9FC VD5 GETCCH LBA3D SYNCOMMA LBA2D LBA8FC VD1 GETCCH LBA3D SYNCOMMA LBA2D LBABAD SYNCOMMA LBA2D LBABAD SYNCOMMA LBA2D LBABAD SYNCOMMA LBA3D SYNCOMMA LBA3D SYNCOMMA LBA3D SYNCOMMA LBA3A LBABAD SYNCOMMA LBA3A LBABAD SYNCOMMA LBA3A LBABAD LBABAB LBABAB LBABAB VD3 VD5 LBABB LBABAB LBABAB LBABAB LBABAB LBABB	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR GO FIND THE LINE NUMBER IN BASIC PROGRAM ERASE VARIABLES DEFAULT LINE NUMBER INTERVAL SAVE DEFAULT INTERVAL NOW ACCD = Ø DEFAULT LINE NUMBER OF WHERE TO START RENUMBERING GET CURRENT INPUT CHARACTER BRANCH IF NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE LINE NUMBER WHERE RENUMBERING STARTS GET CURRENT INPUT CHARACTER BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER LINE GET CURRENT INPUT CHARACTER BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER LINE GET CURRENT INPUT CHARACTER BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW INTERVAL FC' ERROR CHECK FOR COMMA BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW INTERVAL FC' ERROR CHECK FOR MORE CHARACTERS ON LINE - SYNTAX ERROR IF ANY GO GET ADDRESS OF OLD NUMBER BEING RENUMBERED SAVE ADDRESS GET NEXT RENUMBERED LINE NUMBER TO USE FIND THE LINE NUMBER IN THE BASIC PROGRAM COMPARE TO ADDRESS OF OLD LINE NUMBER FC ERROR IF NEW ADDRESS > OLD ADDRESS MAKE SURE RENUMBERED LINE NUMBERS WILL BE IN RANGE CONVERT ASCII LINE NUMBERS TO EXPANDED BINARY RECALCULATE NEXT LINE RAM ADDRESSES
1204 1205 1206 1207 1208 1219 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229 1230 1231 1232 1233 1234 1235 1236 1237 1238 1239 1240 1241 1242	8A02 9E D1 8A04 9F 2B 8A06 7E AD 01 8A09 BD AD 26 8A0C CC 00 0A 8A0F DD D5 8A11 DD CF 8A13 5F 8A14 DD D1 8A18 0P AS 8A18 24 06 8A18 24 06 8A18 BD AS 8A18 9D AS 8A18 9D AS 8A20 27 1B 8A22 BD B2 6D 8A25 24 06 8A27 8D D3 8A29 9F D1 8A28 9D A5 8A29 9F D1 8A28 9D B2 6D 8A33 BD A5 8A34 8D C6 8A34 8D C7 8A30 8D C3 8A3F 9F D3 8A47 9E D5 8A48 BD A5 DD 8A44 BD AC EF 8A48 BD BA DD	* RENUM RENUM LBA2Ø LBA2D LBA3A	RTS LDX JMP JSR LDD STD STD STD JSR BEC BSR STX JSR BEQ D JSR BEQ JSR BEQ D JSR BEQ D JSR BEQ D JSR BEQ D D D D D D D D D D D D D D D D D D D	VD1 BINVAL LADØ1 LADØ6 #1Ø VD5 VCF VD1 GETCCH L8A2Ø L89FC VD5 GETCCH L8A3D SYNCOMMA L8A2D L89FC VD1 GETCCH L8A3D SYNCOMMA L8A2D L89FC VD1 GETCCH L8A3D SYNCOMMA L8A2D L89FC VD1 GETCCH L8A3D SYNCOMMA L8A2D L8AB3 L85C7 L8AB3 LASC7 L8AB3 LASC7 L8AB4 VD3 VD5 L8AB4 VD3 L8AB4 VD3 L8AB4 VD3 L8AB4 VD3 L8AB7 L8AB7 L8AB0 LACEF L8AB0 LACEF L8AB0	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR GO FIND THE LINE NUMBER IN BASIC PROGRAM ERASE VARIABLES DEFAULT LINE NUMBER INTERVAL SAVE DEFAULT INTERVAL NOW ACCD = Ø DEFAULT LINE NUMBER OF WHERE TO START RENUMBERING GET CURRENT INPUT CHARACTER BRANCH IF NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE LINE NUMBER WHERE RENUMBERING STARTS GET CURRENT INPUT CHARACTER BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA DRANCH IF END OF LINE GET CURRENT INPUT CHARACTER BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF END OF LINE GET CURRENT INPUT CHARACTER BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW INTERVAL FC' ERROR CHECK FOR MORE CHARACTERS ON LINE - SYNTAX ERROR IF ANY GO GET ADDRESS OF OLD NUMBER BEING RENUMBERED SAVE ADDRESS GET NEXT RENUMBERED LINE NUMBER TO USE FIND THE LINE NUMBER IN THE BASIC PROGRAM COMPARE TO ADDRESS OF OLD LINE NUMBER FC ERROR IF NEW ADDRESS < OLD ADDRESS MAKE SURE RENUMBERED LINE NUMBERS WILL BE IN RANGE CONVERT ASCII LINE NUMBERS TO EXPANDED BINARY RECALCULATE NEXT LINE RAM ADDRESSES GET RAM ADDRESS OF FIRST LINE TO BE RENUMBERED
1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229 1230 1231 1232 1233 1234 1235 1236 1237 1238 1239 1240 1241	BAM2 9E D1 BAM9 9F 2B BAM6 7F AD Ø1 BAM9 BD AD 26 BAM6 CC ØØ ØA BAM6 DD D5 BA11 DD CF BA13 5F BA14 DD D1 BA16 9D A5 BA18 24 Ø6 BA18 8D EØ BA10 9F D5 BA18 9D A5 BA18 9D A5 BA18 9D A5 BA28 27 1B BA22 BD B2 6D BA22 BD B2 6D BA25 24 Ø6 BA26 27 1B BA27 BD B2 6D BA28 9F D1 BA28 9F D1 BA28 9F D1 BA28 9D A5 BA38 BD B5 BA41 9E D5 BA43 BB BF BA44 9E D5 BA45 9C D3 BA47 55 3A BA49 BD 1C BA48 BD A5 DD BA48 BD A6 DF BA45 9C D3 BA47 55 3A BA49 BD 1C BA48 BD A6 DF BA45 BD B7 BA46 BD	* RENUM RENUM LBA2Ø LBA2D LBA3A	RTS LDX JSR LDD STD STD STD JSR BCC BSR STX JSR BEQ JSR BEQ JSR BCC BSR STX JSR BCC BSR STX JSR BCC BSR STX JSR BCC BSR STX JSR BCC BCC BCC BCC BCC BCC BCC BCC BCC BC	VD1 BINVAL LAD26 #10 VD5 VCF VD1 GETCCH L8A20 L89FC VD5 GETCCH L8A3D SYNCOMMA L8A2D L89FC VD1 GETCCH L8A3D SYNCOMMA L8A2D L89FC VD1 GETCCH L8A3D SYNCOMMA L8A3D L8CF L8A62 VD3 VD5 L8A67 L8A67 L8A67 L8A67 L8A67 L8A67 L8A67 L8A62 VD3	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR GO FIND THE LINE NUMBER IN BASIC PROGRAM ERASE VARIABLES DEFAULT LINE NUMBER INTERVAL SAVE DEFAULT INTERVAL NOW ACCD = Ø DEFAULT LINE NUMBER OF WHERE TO START RENUMBERING GET CURRENT INPUT CHARACTER BRANCH IF NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE LINE NUMBER WHERE RENUMBERING STARTS GET CURRENT INPUT CHARACTER BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER LINE GET CURRENT INPUT CHARACTER BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER LINE GET CURRENT INPUT CHARACTER BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW INTERVAL FC' ERROR CHECK FOR MORE CHARACTERS ON LINE - SYNTAX ERROR IF ANY GO GET ADDRESS OF OLD NUMBER BEING RENUMBERED SAVE ADDRESS GET NEXT RENUMBERED LINE NUMBER TO USE FIND THE LINE NUMBER IN THE BASIC PROGRAM COMPARE TO ADDRESS OF OLD LINE NUMBER FC ERROR IF NEW ADDRESS
1204 1205 1206 1207 1208 1210 1211 1212 1213 1214 1215 1216 1227 1228 1229 1223 1224 1225 1226 1227 1228 1230 1231 1231 1232 1233 1234 1235 1236 1237 1238 1239 1241 1242 1241	8A02 9E D1 8A04 9F 2B 8A06 7E AD 01 8A09 BD AD 26 8A06 CC 00 0A 8A06 DD D5 8A11 DD CF 8A13 5F 8A14 DD D1 8A16 9D A5 8A18 24 06 8A18 80 E0 8A10 9F D5 8A18 9D A5 8A18 9D A5 8A19 9F D5 8A20 27 1B 8A22 BD B2 6D 8A25 24 06 8A27 8D D3 8A29 9F D1 8A28 9D A5 8A29 9F D1 8A28 BD A5 8A20 27 0E 8A27 8D B2 6D 8A33 8D C6 8A36 9F CF 8A38 8C C7 8A38 8C C7 8A30 8D C3 8A37 9F D3 8A37 9F D3 8A38 9F D3 8A38 9F D3 8A39 9F D5 8A31 8D C3 8A37 9F D3 8A38 BD A5 C7 8A30 BD C3 8A37 9F D3 8A38 BD A5 C7 8A39 BD A5 C7 8A39 BD A5 C7 8A31 BD A5 C7	* RENUM RENUM LBA2Ø LBA2D LBA3A	RTS LDX JSR LDD STD STD STD JSR BCC JSR BCC JSR BCQ JSR BCQ JSR BCQ JSR BCQ JSR BCQ JSR BCQ JSR BCQ JSR BCQ JSR BCQ JSR BCQ STD JSR BCQ JSR BCQ JSR BCQ STD JSR STD STD STD STD STD STD JSR STD JSR STD STD STD STD STD STD STD STD STD STD	VD1 BINVAL LAD26 #10 VD5 VCF VD1 GETCCH L8A20 L89FC VD5 GETCCH L8A3D SYNCOMMA L8A2D L8B9FC VD1 GETCCH L8A3D SYNCOMMA L8A2D L8A5C L8A3D SYNCOMMA L8A2D L8A6C VD1 GETCCH L8A3D SYNCOMMA L8A7C L8AAB SYNCOMMA L8A3A L8A6C	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR GO FIND THE LINE NUMBER IN BASIC PROGRAM ERASE VARIABLES DEFAULT LINE NUMBER INTERVAL SAVE DEFAULT RENUMBER START LINE NUMBER SAVE DEFAULT INTERVAL NOW ACCD = Ø DEFAULT LINE NUMBER OF WHERE TO START RENUMBERING GET CURRENT INPUT CHARACTER BRANCH IF NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE LINE NUMBER WHERE RENUMBERING STARTS GET CURRENT INPUT CHARACTER BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER LINE GET CURRENT INPUT CHARACTER BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER LINE GET CURRENT INPUT CHARACTER BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW INTERVAL FC' ERROR CHECK FOR MORE CHARACTERS ON LINE - SYNTAX ERROR IF ANY GO GET ADDRESS OF OLD NUMBER BEING RENUMBERED SAVE ADDRESS GET NEXT RENUMBERED LINE NUMBER TO USE FIND THE LINE NUMBER IN THE BASIC PROGRAM COMPARE TO ADDRESS OF OLD LINE NUMBER FC ERROR IF NEW ADDRESS < OLD ADDRESS MAKE SURE RENUMBERED LINE NUMBERS WILL BE IN RANGE CONVERT ASCIL LINE NUMBERS TO EXPANDED BINARY RECALCULATE NEXT LINE RAM ADDRESSS GET RAM ADDRESS OF FIRST LINE TO BE RENUMBERED SAVE IT MAKE SURE LINE NUMBERS EXIST
1204 1205 1206 1207 1208 1219 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229 1230 1231 1232 1233 1234 1235 1236 1237 1238 1239 1240 1241 1242 1243 1244	BAM2 9E D1 BAM9 9F 2B BAM6 7F AD Ø1 BAM9 BD AD 26 BAM6 CC ØØ ØA BAM6 DD D5 BA11 DD CF BA13 5F BA14 DD D1 BA16 9D A5 BA18 24 Ø6 BA18 8D EØ BA16 9D A5 BA18 24 Ø6 BA17 BD BB BA19 BD BB BA20 27 IB BA22 BD BB BA20 27 IB BA22 BD BB BA25 24 Ø6 BA36 BD BB BA26 BD BB BA27 BD BB BA28 BD BB BA28 BD BB BA28 BD BB BA28 BD BB BA29 BF D1 BA38 BD BB BA38 BD A5 BA39 BB C6 BA34 BD C6 BA34 BD C6 BA34 BD C6 BA36 9F CF BA36 BD C7 BA37 BD BB BA41 BD BB BA43 BD BB BA43 BD BB BA44 BD BC BA48 BD BB BA48 BD BB BA49 BD IC BA48 BD BB BA49 BD IC BA48 BD BA BD BA48 BD BA BA BD BA55 BD BA	* RENUM RENUM LBA2Ø LBA2D LBA3A	RTS LDX JMP JSR LDD STD STD STD JSR BEC BSC BSR STX BEQ JSR BEC BSR STX BEQ JSR BCC BSR STX BEQ JSR STX BEQ JSR STX BEQ JSR STX STX BEQ STX STX STX STX STX STX STX STX STX STX	VD1 BINVAL LADØ1 LADØ6 #1Ø VD5 VCF VD1 GETCCH LBA2Ø L89FC VD5 GETCCH L8A3D SYNCOMMA L8A2D L89FC VD1 GETCCH L8A3D SYNCOMMA L8A2D L89FC VD1 GETCCH L8A3D SYNCOMMA L8A2D L89FC VD1 GETCCH L8A3D SYNCOMMA L8A2D L8A6AD L8A6B	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR GO FIND THE LINE NUMBER IN BASIC PROGRAM ERASE VARIABLES DEFAULT LINE NUMBER INTERVAL SAVE DEFAULT INTERVAL NOW ACCD = Ø DEFAULT LINE NUMBER START LINE NUMBER SAVE DEFAULT INTERVAL NOW ACCD = Ø DEFAULT LINE NUMBER OF WHERE TO START RENUMBERING GET CURRENT INPUT CHARACTER BRANCH IF NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE LINE NUMBER WHERE RENUMBERING STARTS GET CURRENT INPUT CHARACTER BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER LINE GET CURRENT INPUT CHARACTER BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW INTERVAL FC' ERROR CHECK FOR MORE CHARACTERS ON LINE - SYNTAX ERROR IF ANY GO GET ADDRESS OF OLD NUMBER BEING RENUMBERED SAVE ADDRESS GET NEXT RENUMBERED LINE NUMBER TO USE FIND THE LINE NUMBER IN THE BASIC PROGRAM COMPARE TO ADDRESS OF OLD LINE NUMBER FC ERROR IF NEW ADDRESS < OLD ADDRESS MAKE SURE RENUMBERED LINE NUMBERS WILL BE IN RANGE CONVERT ASCII LINE NUMBERS TO EXPANDED BINARY RECALCULATE NEXT LINE RAM ADDRESSES GET RAM ADDRESS OF FIRST LINE TO BE RENUMBERED SAVE IT MAKE SURE LINE NUMBERS EXIST INSERT NEW LINE NUMBERS XIN LINE HEADERS
1204 1205 1206 1207 1208 1210 1211 1212 1213 1214 1215 1216 1227 1228 1229 1223 1224 1225 1226 1227 1228 1230 1231 1231 1232 1233 1234 1235 1236 1237 1238 1239 1241 1242 1241	8A02 9E D1 8A04 9F 2B 8A06 7E AD 01 8A09 BD AD 26 8A06 CC 00 0A 8A06 DD D5 8A11 DD CF 8A13 5F 8A14 DD D1 8A16 9D A5 8A18 24 06 8A18 80 E0 8A10 9F D5 8A18 9D A5 8A18 9D A5 8A19 9F D5 8A20 27 1B 8A22 BD B2 6D 8A25 24 06 8A26 27 0E 8A27 8D D3 8A29 9F D1 8A28 9D A5 8A20 27 0E 8A27 8D D3 8A30 8D C3 8A37 8D C6 8A38 27 49 8A38 BD A5 8A38 9F CF 8A38 BD A5 8A39 9F CF 8A38 BD A5 8A39 9F D3 8A39 BD C3 8A37 9F D3 8A31 BD C3 8A37 9F D3 8A38 BD A5 C7 8A30 BD C3 8A37 9F D3 8A38 BD A5 C7 8A30 BD C3 8A37 9F D3 8A34 BD BF 8A45 9C D3 8A47 25 3A 8A49 8D 1C 8A48 BD BA DD 8A48 BD AC EF 8A51 8D AF 8A53 9F D3 8A48 BD AC EF 8A51 BD AF	* RENUM RENUM LBA2Ø LBA2D LBA3A	RTS LDX JMP JSR LDD STD STD STD JSR BEC BSC BSR STX BEQ JSR BEC BSR STX BEQ JSR BCC BSR STX BEQ JSR STX BEQ JSR STX BEQ JSR STX STX BEQ STX STX STX STX STX STX STX STX STX STX	VD1 BINVAL LAD26 #10 VD5 VCF VD1 GETCCH L8A20 L89FC VD5 GETCCH L8A3D SYNCOMMA L8A2D L8B9FC VD1 GETCCH L8A3D SYNCOMMA L8A2D L8A5C L8A3D SYNCOMMA L8A2D L8A6C VD1 GETCCH L8A3D SYNCOMMA L8A7C L8AAB SYNCOMMA L8A3A L8A6C	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR GO FIND THE LINE NUMBER IN BASIC PROGRAM ERASE VARIABLES DEFAULT LINE NUMBER INTERVAL SAVE DEFAULT INTERVAL NOW ACCD = Ø DEFAULT LINE NUMBER OF WHERE TO START RENUMBERING GET CURRENT INPUT CHARACTER BRANCH IF NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE LINE NUMBER WHERE RENUMBERING STARTS GET CURRENT INPUT CHARACTER BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER LINE GET CURRENT INPUT CHARACTER BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER LINE GET CURRENT INPUT CHARACTER BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW INTERVAL FC' ERROR CHECK FOR MORE CHARACTERS ON LINE - SYNTAX ERROR IF ANY GO GET ADDRESS OF OLD NUMBER BEING RENUMBERED SAVE ADDRESS GET NEXT RENUMBERED LINE NUMBER TO USE FIND THE LINE NUMBER IN THE BASIC PROGRAM COMPARE TO ADDRESS OF OLD LINE NUMBER FC ERROR IF NEW ADDRESS < OLD ADDRESS MAKE SURE RENUMBERED LINE NUMBERS WILL BE IN RANGE CONVERT ASCII LINE NUMBERS TO EXPANDED BINARY RECALCULATE NEXT LINE RAM ADDRESSES GET RAM ADDRESS OF FIRST LINE TO BE RENUMBERED SAVE IT MAKE SURE LINE NUMBERS SIN LINE HEADERS INSERT NEW LINE NUMBERS IN LINE HEADERS INSERT NEW LINE NUMBERS IN LINE HEADERS INSERT NEW LINE NUMBERS IN PROGRAM STATEMENTS
1204 1205 1206 1207 1208 1219 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229 1230 1231 1232 1233 1234 1235 1236 1237 1238 1239 1240 1241 1242 1243 1244	BAM2 9E D1 BAM9 9F 2B BAM6 7F AD Ø1 BAM9 BD AD 26 BAM6 CC ØØ ØA BAM6 DD D5 BA11 DD CF BA13 5F BA14 DD D1 BA16 9D A5 BA18 24 Ø6 BA18 8D EØ BA16 9D A5 BA18 24 Ø6 BA17 BD BB BA19 BD BB BA20 27 IB BA22 BD BB BA20 27 IB BA22 BD BB BA25 24 Ø6 BA36 BD BB BA26 BD BB BA27 BD BB BA28 BD BB BA28 BD BB BA28 BD BB BA28 BD BB BA29 BF D1 BA38 BD BB BA38 BD A5 BA39 BB C6 BA34 BD C6 BA34 BD C6 BA34 BD C6 BA36 9F CF BA36 BD C7 BA37 BD BB BA41 BD BB BA43 BD BB BA43 BD BB BA44 BD BC BA48 BD BB BA48 BD BB BA49 BD IC BA48 BD BB BA49 BD IC BA48 BD BA BD BA48 BD BA BA BD BA55 BD BA	* RENUM RENUM LBA2Ø LBA2D LBA3A	RTS LDX JSR LDD STD STD STD JSR BCC JSR BCC BSR STX JSR BEQ JSR BCC BSR STX JSR BCC BSR STX JSR BCC BSR STX JSR BCC BCC BSR STX JSR BCC BCC BCC BCC BCC BCC BCC BCC BCC BC	VD1 BINVAL LADØ1 LADØ6 #1Ø VD5 VCF VD1 GETCCH LBA2Ø L89FC VD5 GETCCH L8A3D SYNCOMMA L8A2D L89FC VD1 GETCCH L8A3D SYNCOMMA L8A2D L89FC VD1 GETCCH L8A3D SYNCOMMA L8A2D L89FC VD1 GETCCH L8A3D SYNCOMMA L8A2D L8A6AD L8A6B	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR GO FIND THE LINE NUMBER IN BASIC PROGRAM ERASE VARIABLES DEFAULT LINE NUMBER INTERVAL SAVE DEFAULT INTERVAL NOW ACCD = Ø DEFAULT LINE NUMBER START LINE NUMBER SAVE DEFAULT INTERVAL NOW ACCD = Ø DEFAULT LINE NUMBER OF WHERE TO START RENUMBERING GET CURRENT INPUT CHARACTER BRANCH IF NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE LINE NUMBER WHERE RENUMBERING STARTS GET CURRENT INPUT CHARACTER BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER LINE GET CURRENT INPUT CHARACTER BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW INTERVAL FC' ERROR CHECK FOR MORE CHARACTERS ON LINE - SYNTAX ERROR IF ANY GO GET ADDRESS OF OLD NUMBER BEING RENUMBERED SAVE ADDRESS GET NEXT RENUMBERED LINE NUMBER TO USE FIND THE LINE NUMBER IN THE BASIC PROGRAM COMPARE TO ADDRESS OF OLD LINE NUMBER FC ERROR IF NEW ADDRESS < OLD ADDRESS MAKE SURE RENUMBERED LINE NUMBERS WILL BE IN RANGE CONVERT ASCII LINE NUMBERS TO EXPANDED BINARY RECALCULATE NEXT LINE RAM ADDRESSES GET RAM ADDRESS OF FIRST LINE TO BE RENUMBERED SAVE IT MAKE SURE LINE NUMBERS EXIST INSERT NEW LINE NUMBERS XIN LINE HEADERS
1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229 1230 1231 1232 1233 1234 1235 1236 1237 1238 1239 1240 1241 1242 1243 1244 1245 1244 1245	BAM2 9E D1 BAM9 9F 2B BAM06 7F AD 01 BAM9 BD AD 26 BAM0C CC 00 0A BAM6C DD D5 BA11 DD CF BA13 5F BA14 DD D1 BA16 9D A5 BA18 24 06 BA1A 8D E0 BA18 9D A5 BA18 9D A5 BA18 9D A5 BA19 9F D1 BA22 BD B2 6D BA22 27 1B BA22 BD B2 6D BA25 24 06 BA36 9F CF BA36 9F D1 BA28 9D A5 BA37 8D D3 BA29 9F D1 BA28 9D A5 BA38 27 0E BA27 7 0E BA27 8D D3 BA38 9D A5 BA39 9F D1 BA38 BD A5 BA31 9F D3 BA34 BD C3 BA34 BD C3 BA34 BD A5 BA34 BD BF BA34 BD BF BA45 9C D3 BA41 9E D5 BA43 BD BF BA44 9B D1C BA48 BD BA DD BA48 BD AC EF BA55 BD AS BA57 9F D3 BA48 BD AC EF BA55 BD AS BA57 BD BS BA48 BD BC BA48 BD BC BA48 BD BC BA48 BD BC BA59 BD BC BA50 BD	* RENUM RENUM LBA2Ø LBA2D LBA3A	RTS LDX JMP JSR LDD STD STD JSR BCC BSR STX JSR BCC BSR STX JSR BCC BSR STX JSR BCC BSR STX JSR BCC BSR STX JSR BCC BSR STX JSR BCC BSR BSR BCC BSR BSR BCC BSR BSR BCC BSR BSR BCC BSR BSR BSR BCC BSR BSR BCC BSR BSR BCC BSR BSR BCC BSR BSR BCC BSR BSR BCC BSR BSR BCC BSR BSR BSR BCC BSR BSR BCC BSR BSR BCC BSR BSR BCC BSR BSR BCC BSR BSR BSR BSR BSR BSR BSR BSR BSR BSR	VD1 BINVAL LAD26 #10 VD5 VCF VD1 GETCCH L8A20 L89FC VD5 GETCCH L8A3D SYNCOMMA L8A2D L89FC VD1 GETCCH L8A3D SYNCOMMA L8A2A L89FC VD1 SETCCH L8A3D SYNCOMMA L8A3D SYNCOMMA L8A3D SYNCOMMA L8A3D SYNCOMMA L8A3A L8OFC VD1 L8A3B SYNCOMMA L8A3B L8A67 L8A67 L8A62 VD3 L8A67 L8A62 VD3 L8A67 L8A67 L8A68 L8A68 L8A68 L8A68	GET CURRENT OLD NUMBER BEING RENUMBERED SAVE THE LINE NUMBER BEING SEARCHED FOR GO FIND THE LINE NUMBER IN BASIC PROGRAM ERASE VARIABLES DEFAULT LINE NUMBER INTERVAL SAVE DEFAULT INTERVAL NOW ACCD = Ø DEFAULT LINE NUMBER OF WHERE TO START RENUMBERING GET CURRENT INPUT CHARACTER BRANCH IF NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE LINE NUMBER WHERE RENUMBERING STARTS GET CURRENT INPUT CHARACTER BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER LINE GET CURRENT INPUT CHARACTER BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW RENUMBER LINE GET CURRENT INPUT CHARACTER BRANCH IF NEXT CHARACTER NOT NUMERIC CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY SAVE NEW INTERVAL FC' ERROR CHECK FOR MORE CHARACTERS ON LINE - SYNTAX ERROR IF ANY GO GET ADDRESS OF OLD NUMBER BEING RENUMBERED SAVE ADDRESS GET NEXT RENUMBERED LINE NUMBER TO USE FIND THE LINE NUMBER IN THE BASIC PROGRAM COMPARE TO ADDRESS OF OLD LINE NUMBER FC ERROR IF NEW ADDRESS < OLD ADDRESS MAKE SURE RENUMBERED LINE NUMBERS WILL BE IN RANGE CONVERT ASCII LINE NUMBERS TO EXPANDED BINARY RECALCULATE NEXT LINE RAM ADDRESSES GET RAM ADDRESS OF FIRST LINE TO BE RENUMBERED SAVE IT MAKE SURE LINE NUMBERS SIN LINE HEADERS INSERT NEW LINE NUMBERS IN LINE HEADERS INSERT NEW LINE NUMBERS IN LINE HEADERS INSERT NEW LINE NUMBERS IN PROGRAM STATEMENTS

1249	8A61 BD AC EF		JSR	LACEF	RECALCULATE NEXT LINE RAM ADDRESS
1250	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 RETURN IF END OF PROGRAM
1256 1257	8A6F 8D 15 8A71 ØD D8	L8A71	BSR TST	L8A86 VD8	CHECK NEW LINE NUMBER FLAG
1258	8A73 26 Ø2	LOA/I	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
1260	8A77 AE 84	L8A77	LDX	,χ	POINT X TO THE NEXT LINE IN BASIC
1261	8A79 8D ØB		BSR	L8A86	RETURN IF END OF PROGRAM
1262	8A7B D3 CF			VCF	ADD INTERVAL TO CURRENT RENUMBERED LINE NUMBER
1263	8A7D 25 Ø4		BL0	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		LB44A	FC ERROR IF LINE NUMBER MS BYTE > \$F9
1267				O BYTES POINTED TO BY (X).	
1268				RN IF \Leftrightarrow Ø. IF = Ø (END OF	
1269				TURN IS PULLED OFF STACK AND	
1270	0.000 0.4 0.0			TO PREVIOUS SUBROUTINE CALL.	04115 4000
1271	8A86 34 Ø6	L8A86		В,А	SAVE ACCD
1272 1273	8A88 EC 84 8A8A 35 Ø6		LDD	, X	TEST THE 2 BYTES POINTED TO BY X RESTORE ACCD
1273	8A8C 26 Ø2			A,B L8A90	BRANCH IF NOT END OF PROGRAM
1274	8A8E 32 62			\$02,5	PURGE RETURN ADDRESS FROM STACK
1276	8A9Ø 39	L8A9Ø	RTS	¥02,3	TORGE RETORM ADDRESS TROTT STACK
1277	8A91 9E 19	L8A91		TXTTAB	GET START OF BASIC PROGRAM
1278	8A93 3Ø 1F			\$-Ø1,X	MOVE POINTER BACK ONE
1279	8A95 3Ø Ø1	L8A95		\$Ø1,X	MOVE POINTER UP ONE
1280	8A97 8D ED			L8A86	RETURN IF END OF PROGRAM
1281	8A99 3Ø Ø3	L8A99		\$Ø3,X	SKIP OVER NEXT LINE ADDRESS AND LINE NUMBER
1282	8A9B 3Ø Ø1	L8A9B		\$Ø1,X	MOVE POINTER TO NEXT CHARACTER
1283	8A9D A6 84		LDA	, х	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		= MOVE TO NEVI CHADACTED IF > 2
1291 1292	8AAA 26 EF 8AAC 86 Ø3	L8AAC	LDA	L8A9B #\$Ø3	=MOVE TO NEXT CHARACTER IF > 3 * SET 1ST BYTE = 3 TO INDICATE LINE
1293	8AAE A7 8Ø	LOAAC	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	LOADZ	DEC	\$02,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		\$Ø3,X	GET LS BYTE OF LINE NUMBER
1300	8ABB 6A Ø4		DEC	\$Ø4,X	DECREMENT ZERO CHECK FLAG
1301	8ABD 27 Ø1		BEQ	L8ACØ	BRANCH IF IS BYTE ⇔ Ø
1302	8ABF 5F		CLRB		CLEAR LS BYTE
1303	8ACØ ED Ø1	L8ACØ	STD	\$Ø1,X	SAVE BINARY LINE NUMBER
1304	8AC2 DD 2B		STD	BINVAL	SAVE TRIAL LINE NUMBER
1305	8AC4 BD AD Ø1		JSR	LADØ1	FIND RAM ADDRESS OF A BASIC LINE NUMBER
1306	8AC7 9E ØF	L8AC7	LDX	TEMPTR	GET BACK POINTER TO START OF PACKED LINE NUMBER
1307	8AC9 25 E1		BLO	L8AAC	BRANCH IF NO LINE NUMBER MATCH FOUND
1308	8ACB DC 47		LDD	V47	GET START ADDRESS OF LINE NUMBER
1309	8ACD 6C 8Ø		INC	, X+	* SET 1ST BYTE = 2, TO INDICATE LINE NUMBER EXISTS IF CHECKING FOR
1310 1311		^			* EXISTENCE OF LINE NUMBER, SET IT = 1 IF INSERTING LINE NUMBERS
1311	8ACF ED 84		STD	, Х	SAVE RAM ADDRESS OF CORRECT LINE NUMBER
1312	8AD1 20 C6		BRA	, A L8A99	GO GET ANOTHER CHARACTER
1313	8AD3 6F 84	L8AD3		.X	CLEAR CARRY FLAG AND 1ST BYTE
1315	8AD5 AE Ø1	20/100		\$01,X	POINT X TO RAM ADDRESS OF CORRECT LINE NUMBER
1316	8AD7 AE Ø2			\$02,X	PUT CORRECT LINE NUMBER INTO (X)
1317	8AD9 9F 47			V47	SAVE IT TEMPORARILY
1318	8ADB 2Ø EA			L8AC7	GO INSERT IT INTO BASIC LINE
1319	8ADD 9E 19	L8ADD	LDX	TXTTAB	GET BEGINNING OF BASIC PROGRAM
1320	8ADF 20 04		BRA	L8AE5	
1321	8AE1 9E A6	L8AE1		CHARAD	*GET CURRENT INPUT POINTER
1322	8AE3 30 01		LEAX	\$Ø1,X	*AND BUMP IT ONE
1323	8AE5 8D 9F	L8AE5		L8A86	RETURN IF END OF PROGRAM
1324	8AE7 30 02		LEAX	\$Ø2,X	SKIP PAST NEXT LINE ADDRESS
1325	8AE9 30 01	L8AE9		\$Ø1,X	ADVANCE POINTER BY ONE
1326	8AEB 9F A6	L8AEB		CHARAD	SAVE NEW BASIC INPUT POINTER
1327	8AED 9D 9F	L8AED		GETNCH	GET NEXT CHARACTER FROM BASIC
1328	8AEF 4D	L8AEF	TSTA		CHECK THE CHARACTER
1329	8AFØ 27 EF			L8AE1	BRANCH IF END OF LINE
1330	8AF2 2A F9			L8AED	BRANCH IF NOT A TOKEN
1331	8AF4 9E A6			CHARAD	GET CURRENT INPUT POINTER
1332	8AF6 81 FF 8AF8 27 EF			#\$FF L8AE9	IS THIS A SECONDARY TOKEN?
1333 1334					YES - IGNORE IT
1334	8AFA BD Ø1 AØ 8AFD 81 A7			RVEC22 #\$A7	HOOK INTO RAM AND CHECK FOR USER ADDED TOKENS TOKEN FOR THEN?
1335	8AFF 27 12			#\$A/ L8B13	YES
1336	8BØ1 81 84			#\$84	TOKEN FOR ELSE?
1337	8BØ3 27 ØE			L8B13	YES
1339	8BØ5 81 81			#\$81	TOKEN FOR GO?
1340	8BØ7 26 E4			L8AED	NO
1341	8BØ9 9D 9F			GETNCH	GET A CHARACTER FROM BASIC
1342	8BØB 81 A5		CMPA	#\$A5	TOKEN FOR TO?
1343	8BØD 27 Ø4			L8B13	YES
1344	8BØF 81 A6		CMPA	#\$A6	TOKEN FOR SUB?

1345	8B11 26 D8		BNE L8AE	В	NO
1346	8B13 9D 9F	L8B13	JSR GETN		GET A CHARACTER FROM BASIC
1347	8B15 25 Ø4		BLO L8B1	В	BRANCH IF NUMERIC
1348	8B17 9D A5	L8B17	JSR GETC		GET CURRENT BASIC INPUT CHARRACTER
1349	8B19 20 D4		BRA L8AE		KEEP CHECKING THE LINE
1350	8B1B 9E A6	L8B1B	LDX CHAR	AD	GET CURRENT INPUT ADDRESS
1351 1352	8B1D 34 10		PSHS X JSR LAF6	7	SAVE IT ON THE STACK
1352	8B1F BD AF 67 8B22 9E A6		LDX CHAR		CONVERT DECIMAL BASIC NUMBER TO BINARY GET CURRENT INPUT POINTER
1354	8B24 A6 82	L8B24	LDA ,-X	40	GET PREVIOUS INPUT CHARACTER
1355	8B26 BD 9Ø AA	2002.	JSR L9ØA	Δ.	CLEAR CARRY IF NUMERIC INPUT VALUE
1356	8B29 25 F9		BLO L8B2		BRANCH IF NON-NUMERIC
1357	8B2B 3Ø Ø1		LEAX \$Ø1,		MOVE POINTER UP ONE
1358	8B2D 1F 10		TFR X,D		NOW ACCD POINTS TO ONE PAST END OF LINE NUMBER
1359	8B2F EØ 61		SUBB \$01,		SUBTRACT PRE-NUMERIC POINTER LS BYTE
1360	8B31 CØ Ø5		SUBB #\$Ø5		MAKE SURE THERE ARE AT LEAST 5 CHARACTERS IN THE NUMERIC LINE
1361	0000 07 00	*	DE0 10DE	-	DDANGU IF EVACTIVE
1362 1363	8B33 27 20 8B35 25 0A		BEQ L8B5 BLO L8B4		BRANCH IF EXACTLY 5 BRANCH IF < 5
1364	8B37 33 84		LEAU ,X	1	TRANSFER X TO U
1365	8B39 5Ø		NEGB		NEGATE B
1366	8B3A 3Ø 85		LEAX B,X		MOVE X BACK B BYTES
1367	8B3C BD 89 B8		JSR L89B	8	*MOVE BYTES FROM (U) TO (X) UNTIL
1368		*			*U = END OF BASIC; (I) = NEW END OF BASIC
1369	8B3F 2Ø 14		BRA L8B5	5	
1370				OF SPACE FOR THE LINE NUMBER	
1371	8B41 9F 47	L8B41	STX V47		SAVE END OF NUMERIC VALUE
1372	8B43 9E 1B		LDX VART	AB	GET END OF BASIC PROGRAM
1373 1374	8B45 9F 43 8B47 5Ø		STX V43 NEGB		SAVE IT NEGATE B
1374	8B48 3Ø 85		LEAX B,X		ADD IT TO END OF NUMERIC POINTER
1376	8B4A 9F 41		STX V41		SAVE POINTER
1377	8B4C 9F 1B		STX VART	AB	STORE END OF BASIC PROGRAM
1378	8B4E BD AC 1E		JSR LAC1	E	ACCD = TOP OF ARRAYS - CHECK FOR ENOUGH ROOM
1379	8B51 9E 45		LDX V45		* GET AND SAVE THE
1380	8B53 9F A6		STX CHAR	AD	* NEW CURRENT INPUT POINTER
1381	8B55 35 1Ø	L8B55	PULS X		RESTORE POINTER TO START OF NUMERIC VALUE
1382	8B57 86 Ø1		LDA #\$Ø1		NEW LINE NUMBER FLAG
1383	8B59 A7 84		STA ,X		* SAVE NEW LINE FLAG
1384 1385	8B5B A7 Ø2 8B5D A7 Ø4		STA \$02, STA \$04,		*
1386	8B5F D6 2B		STA \$Ø4, LDB BINV		GET MS BYTE OF BINARY LINE NUMBER
1387	8B61 26 Ø4		BNE L8B6		BRANCH IF IT IS NOT ZERO
1388	8B63 C6 Ø1		LDB #\$Ø1		SAVE A 1 IF BYTE IS Ø; OTHERWISE, BASIC WILL
1389					THINK IT IS THE END OF A LINE
1390	8B65 6C Ø2		INC \$02,	X	IF 2,X = 2, THEN PREVIOUS BYTE WAS A ZERO
1391	8B67 E7 Ø1	L8B67	STB \$01,	X	SAVE MS BYTE OF BINARY LINE NUMBER
1392	8B69 D6 2C		LDB BINV		GET IS BYTE OF BINARY LINE NUMBER
1393	8B6B 26 Ø4		BNE L8B7		BRANCH IF NOT A ZERO BYTE
1394	8B6D C6 Ø1		LDB #\$Ø1		SAVE A 1 IF BYTE IS A Ø
1395	8B6F 6C Ø4	10071	INC \$04,		IF 4,X = 2, THEN PREVIOUS BYTE WAS A Ø
1396 1397	8B71 E7 Ø3 8B73 9D A5	L8B71	STB \$Ø3, JSR GETC		SAVE LS BYTE OF BINARY LINE NUMBER GET CURRENT INPUT CHARACTER
1398	8B75 81 2C		CMPA #','		IS IT A COMMA?
1399	8B77 27 9A		BEQ L8B1		YES - PROCESS ANOTHER NUMERIC VALUE
1400	8B79 2Ø 9C		BRA L8B1		NO - GO GET AND PROCESS AN INPUT CHARACTER
1401	8B7B 9E 19	L8B7B	LDX TXTT	AB	POINT X TO START OF BASIC PROGRAM
1402	8B7D 3Ø 1F		LEAX \$-01		MOVE POINTER BACK ONE
1403	8B7F 3Ø Ø1	L8B7F	LEAX \$Ø1,		MOVE POINTER UP ONE
1404	8B81 EC Ø2		LDD \$02,		GET ADDRESS OF NEXT LINE
1405	8B83 DD 68		STD CURL		SAVE IT IN CURLIN RETURN IF END OF PROGRAM
1406 1407	8B85 BD 8A 86 8B88 3Ø Ø3		JSR L8A8 LEAX \$03,		SKIP OVER ADDRESS OF NEXT LINE AND 1ST BYTE OF LINE NUMBER
1407	8B8A 30 01	L8B8A	LEAX \$03,		MOVE POINTER UP ONE
1409	8B8C A6 84	L8B8C	LDA ,X		GET CURRENT CHARACTER
1410	8B8E 27 EF		BEQ L8B7	F	BRANCH IF END OF LINE
	8B9Ø 4A		DECA		INPUT CHARACTER = 1? - VALID LINE NUMBER
	8B91 27 1B		BEQ L8BA		YES
	8B93 8Ø Ø2 8B95 26 F3		SUBA #\$Ø2		INPUT CHARACTER 3? - UL LINE NUMBER NO
	8B95 26 F3 8B97 34 1Ø		BNE L8B8 PSHS X	4	NU SAVE CURRENT POSITION OF INPUT POINTER
	8B99 8E 8B D8		LDX #L8B	D9-1	POINT X TO UL MESSAGE
	8B9C BD B9 9C		JSR STRI		PRINT STRING TO THE SCREEN
	8B9F AE E4		LDX ,S		GET INPUT POINTER
1419	8BA1 EC Ø1		LDD \$01,		GET THE UNDEFINED LINE NUMBER
	8BA3 BD BD CC		JSR LBDC		CONVERT NUMBER IN ACCD TO DECIMAL AND DISPLAY IT
	8BA6 BD BD C5		JSR LBDC		PRINT IN XXXX XXXX = CURRENT LINE NUMBER
	8BA9 BD B9 58		JSR LB95	В	SEND A CR TO CONSOLE OUT
	8BAC 35 10	LODAT	PULS X		GET INPUT POINTER BACK
	8BAE 34 10 8BB0 EC 01	L8BAE	PSHS X LDD \$01,	Y	SAVE CURRENT POSITION OF INPUT POINTER LOAD ACCD WITH BINARY VALUE OF LINE NUMBER
	8BB2 DD 52		STD FPA0		SAVE IN BOTTOM 2 BYTES OF FPAØ
	8BB4 BD 88 ØE		JSR L880		ADJUST REST OF FPAØ AS AN INTEGER
	8BB7 BD BD D9		JSR LBDD		CONVERT FPAØ TO ASCII, STORE IN LINE NUMBER
	8BBA 35 40		PULS U		LOAD U WITH PREVIOUS ADDRESS OF INPUT POINTER
	8BBC C6 Ø5		LDB #\$Ø5		EACH EXPANDED LINE NUMBER USES 5 BYTES
		L8BBE		X	MOVE POINTER FORWARD ONE
	8BCØ A6 84		LDA ,X	2	GET AN ASCII BYTE
	8BC2 27 Ø5		BEQ L8BC	9	BRANCH IF END OF NUMBER
	8BC4 5A 8BC5 A7 CØ		DECB		DECREMENT BYTE COUNTER
	8BC7 2Ø F5		STA ,U+ BRA L8BB	F	STORE ASCII NUMBER IN BASIC LINE CHECK FOR ANOTHER DIGIT
	8BC9 3Ø C4	L8BC9	LEAX ,U	=	TRANSFER NEW LINE POINTER TO (X)
	8BCB 5D		TSTB		DOES THE NEW LINE NUMBER REQUIRE 5 BYTES?
1439	8BCC 27 BE		BEQ L8B8	C	YES - GO GET ANOTHER INPUT CHARACTER
1440	8BCE 31 C4		LEAY ,U		SAVE NEW LINE POINTER IN Y

1441	8BDØ			0.0		LEAU		POINT U TO END OF 5 BYTE PACKED LINE NUMBER BLOCK
1442 1443	8BD2 8BD5			88		LEAX	L89B8 ,Y	MOVE BYTES FROM (U) TO (X) UNTIL END OF PROGRAM LOAD (X) WITH NEW LINE POINTER
1444	8BD7						Ĺ8B8C	GO GET ANOTHER INPUT CHARACTER
1445 1446	8BD9	55 4	4C	20	L8BD9	FCC	'UL '	UNKNOWN LINE NUMBER MESSAGE
1447	8BDC	ØØ				FCB	Ø	
1448 1449					* CONVER	T AN I	INTEGER INTO AN ASCII STRING AND	PRINT IT ON THE SCREEN
1450	8BDD				HEXDOL			CONVERT FPAØ INTO A POSITIVE 2 BYTE INTEGER
1451 1452	8BEØ 8BE3			D9		LDX	#STRBUF+2 #\$Ø4	POINT TO TEMPORARY BUFFER CONVERT 4 NIBBLES
1453	8BE5	34 (L8BE5	PSHS		SAVE NIBBLE COUNTER
1454 1455	8BE7 8BE8		яΔ			CLRB LDA	#\$Ø4	CLEAR CARRY FLAG 4 SHIFTS
1456	8BEA	Ø8 !	53		L8BEA	ASL	FPAØ+3	* SHIFT BOTTOM TWO BYTES OF
1457 1458	8BEC 8BEE		52			ROL ROLB	FPAØ+2	* FPAØ LEFT ONE BIT (X2) IF OVERFLOW, ACCB ⇔ Ø
1459	8BEF					DECA		* DECREMENT SHIFT COUNTER AND
1460 1461	8BFØ 8BF2		F8			BNE TSTB	L8BEA	* BRANCH IF NOT DONE CHECK FOR OVERFLOW
1462	8BF3		ØΑ				L8BFF	BRANCH IF OVERFLOW
1463	8BF5		E4			LDA	,\$	* GET NIBBLE COUNTER,
1464 1465	8BF7 8BF8		Ø5			DECA BEQ	L8BFF	* DECREMENT IT AND * BRANCH IF DONE
1466	8BFA	80 (Ø3	D9		CMPX	#STRBUF+2	= DO NOT DO A CONVERSION UNTIL A NON-ZERO
1467 1468	8BFD 8BFF				L8BFF	BEQ ADDB	L8CØB #'Ø'	= BYTE IS FOUND - LEADING ZERO SUPPRESSION ADD IN ASCII ZERO
1469	8CØ1	C1 3	39			CMPB	#'9'	COMPARE TO ASCII 9
147Ø 1471	8CØ3 8CØ5						L8CØ7 #'A'-('9'+1)	BRANCH IF < 9 ADD ASCII OFFSET IF HEX LETTER
1472	8CØ7				L8CØ7	STB	, χ+	STORE HEX VALUE AND ADVANCE POINTER
1473 1474	8CØ9 8CØB				L8CØB	CLR PULS		CLEAR NEXT BYTE - END OF STRING FLAG * GET NIBBLE COUNTER,
1475	8CØD					DECB	10055	* DECREMENT IT AND
1476 1477	8CØE 8C1Ø						L8BE5 \$02,S	* BRANCH IF NOT DONE PURGE RETURN ADDRESS OFF OF STACK
1478	8C12					LDX	#STRBUF+1	RESET POINTER
1479 1480	8C15	/E I	85	18		JMP	LB518	SAVE STRING ON STRING STACK
1481	0010			00	* DLOAD	100	14400	01005 51150
1482 1483	8C18 8C1B			29	DLOAD L8C1B		LA429 GETCCH	CLOSE FILES GET THE CURRENT INPUT CHARACTER
1484 1485	8C1D 8C1F					SUBA PSHS		CHECK FOR DLOADM
	8C21						L8C25	SAVE DLOADM (=∅), OLOAD (<>∅) FLAG BRANCH IF OLOAD
1487 1488	8C23 8C25			70	L8C25		GETNCH LA578	GET AN INPUT CHAR FROM BASIC GET THE NAME OF FILE FROM BASIC
1489	8C28			70	L0023		GETCCH	GET CURRENT INPUT CHAR FROM BASIC
149Ø 1491	8C2A 8C2C			6 D			L8C44 SYNCOMMA	BRANCH IF END OF LINE SYNTAX CHECK FOR COMMA
1492	8C2F			OD			#','	CHECK FOR TWO CONSECUTIVE COMMAS
1493 1494	8C31	27	11		*	BEQ	L8C44	*BRANCH IF,, - IF THIS CASE IS SELECTED *THE BAUD DELAY MUST HAVE BEEN PREVIOUSLY STORED IN DIBAUD
1495	8C33	BD I	В7	ØB		JSR	EVALEXPB	EVAL EXPR, RETURN VALUE IN ACCB
1496 1497	8C36 8C38		ВØ			LDA TSTB	#44*4	DELAY VALUE FOR 300 BAUD WAS ARGUMENT = 0?
1498	8C39		07				L8C42	YES - 300 BAUD
1499 1500	8C3B 8C3D		2C			LDA DECB	#44	DELAY VALUE FOR 1200 BAUD CHECK FOR ARGUMENT OF 1
1501			26	28 Ø8			LB44A	FC ERROR IF NOT ZERO OR ONE OR COMMA
1502 1503	8C42 8C44			DØ	L8C42 L8C44		DLBAUD L8CDØ	SAVE DELAY VALUE TRANSMIT FILE NAME AND READ IN FILE STATUS
1504	8C47			55	20044	PSHS		SAVE ACCA
15Ø5 15Ø6	8C49 8C4B					LDA STA	#-3 DEVNUM	DLOAD DEVICE NUMBER TO -3 SET DEVICE NUMBER TO DLOAD
1507	8C4D	35 (02			PULS	A	RESTORE ACCA
1508 1509	8C4F 8C51					TST BEO	,S+ L8C85	DLOAD OR DLOADM? DLOADM
1510		•	-		,			
1511 1512	8C53	BD A	A5	C7	* READ I		LOAD FILE LASC7	CHECK FOR END OF LINE - SYNTAX ERROR IF NOT
1513	8C56	5D				TSTB		CHECK ASCII FLAG
1514 1515	8C57 8C59			19			L8C5F LAD19	FM ERROR IF NOT ASCII GO DO A NEW
1516	8C5C						LAC7C	*JUMP BACK TO BASIC S MAIN INPUT LOOP;
1517 1518	8C5F	7E /	A6	16	* L8C5F	JMP	LA616	*DLOAD FILES MUST BE ASCII FILES BAD FILE MODE ERROR
1519							PROCESSOR	
1520 1521	8062	9D 9	9 F		L8C62		GETNCH	GET A CHAR FROM BASIC
1522				10 25		CMPA		CHECK FOR CLOADM
	8C6A	ØF :	78	18 2E			LA498 FILSTA	GO DO A CLOAD CLOSE FILES
1525				79			GETNCH LA578	GET A CHAR FROM BASIC
1526 1527							LA648	STRIP A FILENAME OFF OF THE BASIC LINE SEARCH FOR FILE
1528							CASBUF+10 LA505	CHECK FILE MODE
1529 1530	8C7B	FE (01			LDU	CASBUF+8	BRANCH TO CLOADM IF NOT BLOCK LOAD SAVE FILE TYPE AND ASCII FLAG IN U
1531 1532	8C7E 8C8Ø			35			DEVNUM LA635	SET DEVICE NUMBER TO -1 (CASSETTE) GO READ IN A DATA BLOCK
1533	8083			55		TFR		PUT FILE TYPE & ASCII FLAG BACK IN ACCD
1534 1535					* STRIP	Α LΩΔΓ	O OFFSET FROM THE BASIC LINE, TH	EN LOAD IN BLOCKS OF
1536							M,DLOADM) WHICH ARE PRECEEDED BY	

1537				MBLE CONTAINS A BLOCK LENGTH	
1538				OF NON-CONTIGUOUS BLOCKS MAY	
1539 1540	8C85 83 Ø2 ØØ		IILL IERM. SUBD #\$2:	INATE THE LOADING PROCESS AND	* CHECK FILE STATUS;
1541	8C88 26 D5		BNE L8C		* FM ERROR IF MODE ⇒ 2 OR TYPE ⇒ Ø
1542	8C8A 9E 8A		LDX ZER		ZERO THE X REG - DEFAULT OFFSET
1543	8C8C 9D A5		JSR GET		GET CURRENT INPUT CHARACTER
1544	8C8E 27 Ø6		BEQ L8C		BRANCH IF END OF LINE
1545	8C9Ø BD B2 6D		JSR SYN		SYNTAX CHECK FOR COMMA
1546 1547	8C93 BD B7 3D 8C96 9F D3		JSR LB7: STX VD3		EVAL INTEGER EXPR - RETURN VALUE IN X SAVE OFFSET
1548	8C98 BD A5 C7		JSR LA5		SYNTAX ERROR IF MORE CHARS ON LINE
1549	8C9B 8D 29	L8C9B	BSR L8C		GO GET EOF FLAG FROM CONSOLE IN
1550	8C9D 34 Ø2		PSHS A		SAVE IT ON THE STACK
1551	8C9F 8D 1E		BSR L8C		* READ IN BLOCK LENGTH FROM CONSOLE IN
1552 1553	8CA1 1F Ø2 8CA3 8D 1A		TFR D,Y BSR L8C		* AND SAVE IT IN Y GET LOAD ADDRESS FROM CONSOLE IN
1554	8CA5 D3 D3		ADDD VD3		ADD OFFSET TO LOAD ADDRESS
1555	8CA7 DD 9D		STD EXE		SAVE IN EXEC ADDRESS
1556	8CA9 1F Ø1		TFR D,X		SAVE LOAD ADDRESS IN X
1557	8CAB A6 EØ		LDA ,S+		GET EOF FLAG FROM STACK
1558	8CAD 10 26 17 7C		LBNE LA4		CLOSE FILES IF POSTAMBLE BLOCK
1559 1560	8CB1 8D 13 8CB3 A7 84		BSR L8C STA ,X		GET A CHARACTER FROM CONSOLE IN SAVE IT IN RAM
1561	8CB5 A1 8Ø		CMPA ,X+		COMPARE SAVED BYTE TO ACTUAL BYTE
1562	8CB7 26 14		BNE L8C		'IO ERROR IF NOT = (SAVED IN ROM OR BAD RAM)
1563	8CB9 31 3F		LEAY \$-Ø		DECREMENT BYTE COUNT
1564	8CBB 26 F4		BNE L8C	B1	READ MORE CHARACTERS
1565	8CBD 20 DC	* CET 2 C	RKA FRC	AR DETIION THEM IN VCCD	LOOK FOR ANOTHER BLOCK OF DATA
1566 1567	8CBF 8D ØØ	18CRF	RSR IRC	9B S - RETURN THEM IN ACCD C1	GET A CHARACTER IN ACCB
1568	8CC1 8D Ø3		BSR L8C		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 LA1		GET A CHARACTER FROM CONSOLE IN
1573 1574	8CC9 ØD 7Ø 8CCB 27 F8		TST CIN BEQ L8C		IS FILE EMPTY? RETURN IF NOT EMPTY
1575	8CCD 7E A6 19	18000	JMP LA6		IO ERROR IF EMPTY
1576	8CDØ 8D 42		BSR L8D		TRANSMIT FILE NAME, RETURN FILE STATUS
1577	8CD2 34 Ø6		PSHS B,A		SAVE FILE STATUS ON STACK
1578	8CD4 4C		INCA		CHECK FILE TYPE
1579	8CD5 27 Ø6		BEQ L8C		NE ERROR IF FILE NOT FOUND
158Ø 1581	8CD7 DE 8A		LDU ZER		ZERO U REG :FIRST BLOCK NUMBER
1582	8CD9 8D Ø9 8CDB 35 86		BSR L8C PULS A,B		READ IN 128 CHARACTERS GET FILE STATUS BACK AND RETURN
1583	8CDD C6 34	L8CDD	LDB #2*	26	NE ERROR
1584	8CDF 7E AC 46		JMP LAC	46	GO TO ERROR SERVICING ROUTINE
1585		* REFILL	CONSOLE 1	IN CHARACTER BUFFER FROM DLOA	D
1586	8CE2 DE 7E	L8CE2	LDU CBU	FAD	GET BLOCK NUMBER
1586 1587	8CE4 3Ø 41	L8CE2 L8CE4	LDU CBU LEAX \$Ø1	FAD ,U	GET BLOCK NUMBER * INCREMENT BLOCK NUMBER
1586 1587 1588	8CE4 3Ø 41 8CE6 9F 7E	L8CE2 L8CE4	LDU CBU LEAX \$Ø1 STX CBU	FAD ,U FAD	GET BLOCK NUMBER * INCREMENT BLOCK NUMBER * AND SAVE IT
1586 1587 1588 1589	8CE4 3Ø 41 8CE6 9F 7E 8CE8 8E Ø1 DA	L8CE2 L8CE4	LDU CBU LEAX \$01 STX CBU LDX #CA	FAD ,U FAD SBUF	GET BLOCK NUMBER * INCREMENT BLOCK NUMBER * AND SAVE IT USE CASBUF AS DLOAD INPUT BUFFER
1586 1587 1588	8CE4 3Ø 41 8CE6 9F 7E	L8CE2 L8CE4	LDU CBU LEAX \$Ø1 STX CBU	FAD ,U FAD SBUF 7C	GET BLOCK NUMBER * INCREMENT BLOCK NUMBER * AND SAVE IT
1586 1587 1588 1589 1590 1591 1592	8CE4 3Ø 41 8CE6 9F 7E 8CE8 8E Ø1 DA 8CEB BD 8D 7C	L8CE2 L8CE4	LDU CBU LEAX \$Ø1 STX CBU LDX #CA JSR L8D JMP LA6	FAD ,U FAD SBUF 7C 44	GET BLOCK NUMBER * INCREMENT BLOCK NUMBER * AND SAVE IT USE CASBUF AS DLOAD INPUT BUFFER READ 128 CHARACTERS (ONE BLOCK) INTO BUFFER
1586 1587 1588 1589 1590 1591 1592 1593	8CE4 3Ø 41 8CE6 9F 7E 8CE8 8E Ø1 DA 8CEB BD 8D 7C 8CEE 7E A6 44	L8CE2 L8CE4 * CONSOLE	LDU CBU LEAX \$Ø1 STX CBU LDX #CA: JSR L8D JMP LA6	FAD , U FAD SBUF 7C 44	GET BLOCK NUMBER * INCREMENT BLOCK NUMBER * AND SAVE IT USE CASBUF AS DLOAD INPUT BUFFER READ 128 CHARACTERS (ONE BLOCK) INTO BUFFER RESET CONSOLE IN BUFFER
1586 1587 1588 1589 1590 1591 1592 1593 1594	8CE4 3Ø 41 8CE6 9F 7E 8CE8 8E Ø1 DA 8CEB BD 8D 7C 8CEE 7E A6 44	L8CE2 L8CE4 * CONSOLE XVEC4	LDU CBU LEAX \$01 STX CBU LDX #CA JSR L8D JMP LA6 E IN RAM H LDA DEVI	FAD FAD SBUF 7C 444 HOOK NUM	GET BLOCK NUMBER * INCREMENT BLOCK NUMBER * AND SAVE IT USE CASBUF AS DLOAD INPUT BUFFER READ 128 CHARACTERS (ONE BLOCK) INTO BUFFER RESET CONSOLE IN BUFFER GET DEVICE NUMBER
1586 1587 1588 1589 1590 1591 1592 1593 1594 1595	8CE4 3Ø 41 8CE6 9F 7E 8CE8 8E Ø1 DA 8CEB BD 8D 7C 8CEE 7E A6 44 8CF1 96 6F 8CF3 81 FD	L8CE2 L8CE4 * CONSOLE XVEC4	LDU CBU LEAX \$01 STX CBU LDX #CA JSR L8D JMP LA6 E IN RAM F LDA DEVI CMPA #-3	FAD , U FAD SBUF 7C 44 HOOK NUM	GET BLOCK NUMBER * INCREMENT BLOCK NUMBER * AND SAVE IT USE CASBUF AS DLOAD INPUT BUFFER READ 128 CHARACTERS (ONE BLOCK) INTO BUFFER RESET CONSOLE IN BUFFER GET DEVICE NUMBER DLOAD DEVICE NUMBER
1586 1587 1588 1589 1590 1591 1592 1593 1594 1595 1596	8CE4 3Ø 41 8CE6 9F 7E 8CE8 8E Ø1 DA 8CEB BD 8D 7C 8CEE 7E A6 44 8CF1 96 6F 8CF3 81 FD 8CF5 26 ØA	L8CE2 L8CE4 * CONSOLE XVEC4	LDU CBU LEAX \$Ø1 STX CBU LDX #CA JSR L8D JMP LA6 E IN RAM H LDA DEVI CMPA #-3 BNE L8D	FAD , U FAD SBUF 7C 44 HOOK NUM	GET BLOCK NUMBER * INCREMENT BLOCK NUMBER * AND SAVE IT USE CASBUF AS DLOAD INPUT BUFFER READ 128 CHARACTERS (ONE BLOCK) INTO BUFFER RESET CONSOLE IN BUFFER GET DEVICE NUMBER DLOAD DEVICE NUMBER BRANCH IF NOT OLOAD
1586 1587 1588 1589 1590 1591 1592 1593 1594 1595	8CE4 3Ø 41 8CE6 9F 7E 8CE8 8E Ø1 DA 8CEB BD 8D 7C 8CEE 7E A6 44 8CF1 96 6F 8CF3 81 FD	L8CE2 L8CE4 * CONSOLE XVEC4	LDU CBU LEAX \$01 STX CBU LDX #CA JSR L8D JMP LA6 E IN RAM F LDA DEVI CMPA #-3	FAD ,U FAD SBUF 7C 444 HOOK NUM	GET BLOCK NUMBER * INCREMENT BLOCK NUMBER * AND SAVE IT USE CASBUF AS DLOAD INPUT BUFFER READ 128 CHARACTERS (ONE BLOCK) INTO BUFFER RESET CONSOLE IN BUFFER GET DEVICE NUMBER DLOAD DEVICE NUMBER
1586 1587 1588 1589 1590 1591 1592 1593 1594 1595 1596 1597	8CE4 3Ø 41 8CE6 9F 7E 8CE8 8E Ø1 DA 8CEB BD 8D 7C 8CEE 7E A6 44 8CF1 96 6F 8CF3 81 FD 8CF5 26 ØA 8CF7 32 62	L8CE2 L8CE4 * CONSOLE XVEC4	LDU CBU LEAX \$Ø1 STX CBU LDX #CA JSR L8D JMP LA6 SIN RAM H LDA DEV CMPA #-3 BNE L8D LEAS \$Ø2 CLR CIN TST CIN	FAD , U FAD SBUF 7C 44 HOOK NUM Ø1 , S BBL CTR	GET BLOCK NUMBER * INCREMENT BLOCK NUMBER * AND SAVE IT USE CASBUF AS DLOAD INPUT BUFFER READ 128 CHARACTERS (ONE BLOCK) INTO BUFFER RESET CONSOLE IN BUFFER GET DEVICE NUMBER BLOAD DEVICE NUMBER BRANCH IF NOT OLOAD PURGE 1ST RETURN ADDR FROM STACK RESET EMPTY/FULL FLAG ANY CHARACTERS LEFT IN BUFFER?
1586 1587 1588 1589 1590 1591 1592 1593 1594 1595 1596 1597 1598 1599 1600	8CE4 3Ø 41 8CE6 9F 7E 8CE8 8E Ø1 DA 8CEB BD 8D 7C 8CEE 7E A6 44 8CF1 96 6F 8CF3 81 FD 8CF5 26 ØA 8CF7 32 62 8CF9 ØF 7Ø 8CFB ØD 79 8CFD 26 Ø3	L8CE2 L8CE4 * CONSOLE XVEC4	LDU CBU LEAX \$01 STX CBU LDX #CA JSR L8D JMP LA6 : IN RAM H LDA DEV CMPA #-3 BNE L8D LEAS \$02 CCLR CIN TST CIN BNE L8D	FAD ,U FAD SBUF 7C 444 HOOK NUM Ø1 ,S BFL CTR	GET BLOCK NUMBER * INCREMENT BLOCK NUMBER * AND SAVE IT USE CASBUF AS DLOAD INPUT BUFFER READ 128 CHARACTERS (ONE BLOCK) INTO BUFFER RESET CONSOLE IN BUFFER GET DEVICE NUMBER DLOAD DEVICE NUMBER BRANCH IF NOT OLOAD PURGE 1ST RETURN ADDR FROM STACK RESET EMPTY/FULL FLAG ANY CHARACTERS LEFT IN BUFFER? YES, GO GET ONE
1586 1587 1588 1589 1590 1591 1592 1593 1594 1595 1596 1597 1598 1599 1600 1601	8CE4 3Ø 41 8CE6 9F 7E 8CE8 8E Ø1 DA 8CEB BD 8D 7C 8CEE 7E A6 44 8CF1 96 6F 8CF3 81 FD 8CF5 26 ØA 8CF7 32 62 8CF9 ØF 7Ø 8CFB ØD 79 8CFB ØD 79 8CFB ØD 79	LBCE2 LBCE4 * CONSOLE XVEC4	LDU CBU LEAX \$01 LEAX \$01 STX CBU LDX #CA JSR L8D JMP LA6- : IN RAM H LDA DEV! CMPA #-3 BNE L8D LEAS \$02 CLR CIN TST CIN BNE L8D COM CIN	FAD ,U FAD SBUF 7C 444 HOOK NUM Ø1 ,S BFL CTR	GET BLOCK NUMBER * INCREMENT BLOCK NUMBER * AND SAVE IT USE CASBUF AS DLOAD INPUT BUFFER READ 128 CHARACTERS (ONE BLOCK) INTO BUFFER RESET CONSOLE IN BUFFER GET DEVICE NUMBER BLOAD DEVICE NUMBER BRANCH IF NOT OLOAD PURGE 1ST RETURN ADDR FROM STACK RESET EMPTY/FULL FLAG ANY CHARACTERS LEFT IN BUFFER?
1586 1587 1588 1599 1591 1592 1593 1594 1595 1596 1597 1598 1599 1600 1601	8CE4 3Ø 41 8CE6 9F 7E 8CE8 8E Ø1 DA 8CEB BD 8D 7C 8CEE 7E A6 44 8CF1 96 6F 8CF3 81 FD 8CF5 26 ØA 8CF7 32 62 8CF9 ØF 7Ø 8CFB ØD 79 8CFD 26 Ø3 8CFD 26 Ø3 8CFD 37 7Ø 8DØ1 39	L8CE2 L8CE4 * CONSOLE XVEC4 L8DØ1	LDU CBU CBU LEAX \$01 LEAX \$01 LEAX \$01 LEAX \$02 LEAX \$03	FAD SBUF 7C 444 HOOK WUM Ø1 ,S BFL CTR Ø2 BFL	GET BLOCK NUMBER * INCREMENT BLOCK NUMBER * AND SAVE IT USE CASBUF AS DLOAD INPUT BUFFER READ 128 CHARACTERS (ONE BLOCK) INTO BUFFER RESET CONSOLE IN BUFFER GET DEVICE NUMBER BLOAD DEVICE NUMBER BRANCH IF NOT OLOAD PURGE 1ST RETURN ADDR FROM STACK RESET EMPTY/FULL FLAG ANY CHARACTERS LEFT IN BUFFER? YES, GO GET ONE SET EMPTY/FULL FLAG TO EOF
1586 1587 1588 1589 1590 1591 1592 1593 1594 1595 1596 1597 1598 1599 1600 1601 1602	8CE4 3Ø 41 8CE6 9F 7E 8CE8 8E Ø1 DA 8CEB BD 8D 7C 8CEE 7E A6 44 8CF1 96 6F 8CF3 81 FD 8CF5 26 ØA 8CF7 32 62 8CF9 ØF 7Ø 8CFB ØD 79 8CFD 26 Ø3 8CFF Ø3 7Ø 8DØ1 39 8DØ2 34 74	LBCE2 LBCE4 * CONSOLE XVEC4 LBD01 LBD02	LDU CBU CBU LEAX \$01 STX CBU LDX #CA JSR L8D JMP LA6. IN RAM H LDA DEVICEMPA #-3 BBE L8D LEAS \$02 CLR CIN TST CIN. BNE L8D COM CIN RTS PSHS U, Y	FAD ,U FAD SBUF 7C 444 HOOK NUM Ø1 ,S BFL CTR 02 BFL	GET BLOCK NUMBER * INCREMENT BLOCK NUMBER * AND SAVE IT USE CASBUF AS DLOAD INPUT BUFFER READ 128 CHARACTERS (ONE BLOCK) INTO BUFFER RESET CONSOLE IN BUFFER GET DEVICE NUMBER DLOAD DEVICE NUMBER BRANCH IF NOT OLOAD PURGE 1ST RETURN ADDR FROM STACK RESET EMPTY/FULL FLAG ANY CHARACTERS LEFT IN BUFFER? YES, GO GET ONE SET EMPTY/FULL FLAG TO EOF SAVE REGISTERS
1586 1587 1588 1599 1591 1592 1593 1594 1595 1596 1597 1598 1599 1600 1601	8CE4 3Ø 41 8CE6 9F 7E 8CE8 8E Ø1 DA 8CEB BD 8D 7C 8CEE 7E A6 44 8CF1 96 6F 8CF3 81 FD 8CF5 26 ØA 8CF7 32 62 8CF9 ØF 7Ø 8CFB ØD 79 8CFD 26 Ø3 8CFD 26 Ø3 8CFD 37 7Ø 8DØ1 39	L8CE2 L8CE4 * CONSOLE XVEC4 L8DØ1 L8DØ2	LDU CBU LEAX \$01 STX CBU LDX #CA JSR L8D JMP LAG- : IN RAM H LDA DEV CMPA #-3 BME L8D LEAS \$02 CLR CIN TST CIN BME L8D COM CIN RTS PSHS U, Y LDX CIN	FAD ,U FAD SBUF 7C 444 HOOK NUM Ø1 ,S BFL CTR Ø2 BFL ,X,B PPTR	GET BLOCK NUMBER * INCREMENT BLOCK NUMBER * AND SAVE IT USE CASBUF AS DLOAD INPUT BUFFER READ 128 CHARACTERS (ONE BLOCK) INTO BUFFER RESET CONSOLE IN BUFFER GET DEVICE NUMBER BLOAD DEVICE NUMBER BRANCH IF NOT OLOAD PURGE 1ST RETURN ADDR FROM STACK RESET EMPTY/FULL FLAG ANY CHARACTERS LEFT IN BUFFER? YES, GO GET ONE SET EMPTY/FULL FLAG TO EOF
1586 1587 1588 1589 1590 1591 1592 1593 1594 1595 1596 1597 1598 1599 1600 1601 1602 1603	8CE4 3Ø 41 8CE6 9F 7E 8CE8 8E Ø1 DA 8CEB BD 8D 7C 8CEE 7E A6 44 8CF1 96 6F 8CF3 81 FD 8CF5 26 ØA 8CF7 32 62 8CF9 ØF 7Ø 8CFB ØD 79 8CFD 26 Ø3 8CFF Ø3 7Ø 8DØ1 39 8DØ2 34 74	L8CE2 L8CE4 * CONSOLE XVEC4 L8DØ1 L8DØ2	LDU GBU LEAX \$01 STX CBU LDX #CA LBD LDX #CA LBD JMP LA6 : IN RAM H LDA DEV' CMPA #-3 BNE LBD LEAS \$02 CLR GIN TST CIN BNE LBD COM GIN RTS PSHS U, Y LDA , X+ PSHS A	FAD ,U FAD SBUF 7C 444 HOOK NUM Ø1 ,S BFL CTR Ø2 BFL ,X,B PTR	GET BLOCK NUMBER * INCREMENT BLOCK NUMBER * AND SAVE IT USE CASBUF AS DLOAD INPUT BUFFER READ 128 CHARACTERS (ONE BLOCK) INTO BUFFER GET DEVICE NUMBER DLOAD DEVICE NUMBER BRANCH IF NOT OLOAD PURGE 1ST RETURN ADDR FROM STACK RESET EMPTY/FULL FLAG ANY CHARACTERS LEFT IN BUFFER? YES, GO GET ONE SET EMPTY/FULL FLAG TO EOF SAVE REGISTERS GET CONSOLE IN CHARACTER BUFFER
1586 1587 1588 1589 1590 1591 1592 1593 1595 1596 1597 1598 1599 1600 1601 1603 1604 1605 1606	8CE4 3Ø 41 8CE6 9F 7E 8CE8 8E Ø1 DA 8CEB BD 8D 7C 8CEE 7E A6 44 8CF1 96 6F 8CF3 81 FD 8CF5 26 ØA 8CF7 32 62 8CF9 ØF 7Ø 8CFB ØD 79 8CFD 26 Ø3 8CFF Ø3 7Ø 8DØ1 39 8DØ2 34 74 8DØ4 9E 7A 8DØ6 A6 8Ø 8DØ8 34 Ø2 8DØA 9F 7A	* CONSOLE XVEC4	LDU CBU LEAX \$01 STX CBU LDX #CA JSR L8D JJMP LA6. SIN RAM H LDA DEV CMPA #-3 BNE L8D LEAS \$02 CLR CIN TST CIN BNE L8D COM CIN RTS COM CIN RTS LY LDA CIN LDA CH LY LDA CH LY	FAD ,U FAD SBUF 7C 444 HOOK NUM 81 ,S BFL CTR 82 BFL pX,B PTR	GET BLOCK NUMBER * INCREMENT BLOCK NUMBER * AND SAVE IT USE CASBUF AS DLOAD INPUT BUFFER READ 128 CHARACTERS (ONE BLOCK) INTO BUFFER GET DEVICE NUMBER DLOAD DEVICE NUMBER BRANCH IF NOT OLOAD PURGE 1ST RETURN ADDR FROM STACK RESET EMPTY/FULL FLAG ANY CHARACTERS LEFT IN BUFFER? YES, GO GET ONE SET EMPTY/FULL FLAG TO EOF SAVE REGISTERS GET CONSOLE IN CHARACTER BUFFER GET A CHARACTER SAVE IT ON THE STACK SAVE NEW CHARACTER BUFFER
1586 1587 1588 1599 1591 1592 1593 1594 1595 1596 1597 1598 1600 1601 1602 1603 1604 1605 1606 1606 1607	8CE4 3Ø 41 8CE6 9F 7E 8CE8 8E Ø1 DA 8CEB BD 8D 7C 8CEE 7E A6 44 8CF1 96 6F 8CF3 81 FD 8CF5 26 ØA 8CF7 32 62 8CF9 ØF 7Ø 8CFB ØD 79 8CFD 26 Ø3 8CFF Ø3 7Ø 8DØ1 39 8DØ2 34 74 8DØ4 9E 7A 8DØ6 A6 8Ø 8DØ8 34 Ø2 8DØ8 9F 7A 8DØ6 0A 79	LBCE2 LBCE4 * CONSOLE XVEC4 LBDØ1 LBDØ2	LDU CBU LEAX \$01 STX CBU LDX #CA JSR L8D JMP LA6. : IN RAM H LDA DEVI CMPA #-3 BNE L8D BNE L8D LEAS \$02 CLR CIN TST CIN BNE L8D COM CIN RTS PSHS U,Y LDX CIN LDA X+ PSHS A ST DEC CIN	FAD ,U FAD SBUF 7C 444 HOOK NUM Ø1 ,S BFL CTR Ø2 BFL X,B PTR PTR	GET BLOCK NUMBER * INCREMENT BLOCK NUMBER * AND SAVE IT USE CASBUF AS DLOAD INPUT BUFFER READ 128 CHARACTERS (ONE BLOCK) INTO BUFFER GET DEVICE NUMBER BLOAD DEVICE NUMBER BRANCH IF NOT OLOAD PURGE 1ST RETURN ADDR FROM STACK RESET EMPTY/FULL FLAG ANY CHARACTERS LEFT IN BUFFER? YES, GO GET ONE SET EMPTY/FULL FLAG TO EOF SAVE REGISTERS GET CONSOLE IN CHARACTER BUFFER GET A CHARACTER SAVE IT ON THE STACK SAVE NEW CHARACTER BUFFER DECREMENT CHARACTER BUFFER DECREMENT CHARACTER BUFFER
1586 1587 1588 1589 1591 1591 1592 1593 1594 1595 1596 1699 1601 1601 1603 1604 1606 1607 1607	8CE4 3Ø 41 8CE6 9F 7E 8CE8 8E Ø1 DA 8CEB BD 8D 7C 8CEE 7E A6 44 8CF1 96 6F 8CF3 81 FD 8CF5 26 ØA 8CF7 32 62 8CF9 ØF 7Ø 8CFB ØD 79 8CFD 26 Ø3 8CFF Ø3 7Ø 8DØ1 39 8DØ2 34 74 8DØ4 9E 7A 8DØ6 A6 8Ø 8DØ8 34 Ø2 8DØ8 34 Ø2 8DØ8 9F 7A 8DØC ØA 79 8DØC ØA 79 8DØC ØA 79 8DØC ØA 79 8DØC 26 Ø2	L8CE2 L8CE4 * CONSOLE XVEC4 L8D01 L8D02	LDU GBU LEAX \$01 STX CBU LDX #CA STX LBU LDX #CA LBD JMP LA6 : IN RAM LDA DEV' CMPA #-3 BNE LBD LEAS \$02 CLR CIN TST CIN BNE LBD LCOM CIN RTS PSHS U, Y LDX CIN LDA , X+ PSHS CIN DBC CIN BNE LBD LCA (CIN DBC CIN BNE LBD	FAD ,U FAD SBUF 7C 444 HOOK NUM Ø1 ,S BFL CTR 02 BFL ,X,B PTR	GET BLOCK NUMBER * INCREMENT BLOCK NUMBER * AND SAVE IT USE CASBUF AS DLOAD INPUT BUFFER READ 128 CHARACTERS (ONE BLOCK) INTO BUFFER GET DEVICE NUMBER DLOAD DEVICE NUMBER DLOAD DEVICE NUMBER BRANCH IF NOT OLOAD PURGE 1ST RETURN ADDR FROM STACK RESET EMPTY/FULL FLAG ANY CHARACTERS LEFT IN BUFFER? YES, GO GET ONE SET EMPTY/FULL FLAG TO EOF SAVE REGISTERS GET CONSOLE IN CHARACTER BUFFER GET A CHARACTER SAVE IT ON THE STACK SAVE NEW CHARACTER BUFFER DECREMENT CHARACTER GUPTER DECREMENT CHARACTER COUNTER RETURN IF BUFFER NOT EMPTY
1586 1587 1598 1599 1591 1592 1593 1594 1595 1596 1597 1598 1599 1600 1601 1602 1603 1604 1605 1606 1607 1608	8CE4 3Ø 41 8CE6 9F 7E 8CE8 8E Ø1 DA 8CEB BD 8D 7C 8CEE 7E A6 44 8CF1 96 6F 8CF3 81 FD 8CF5 26 ØA 8CF7 32 62 8CF9 ØF 7Ø 8CFB ØD 79 8CFD 26 Ø3 8CFF Ø3 7Ø 8DØ1 39 8DØ2 34 74 8DØ4 9E 7A 8DØ6 A6 8Ø 8DØ8 34 02 8DØA 9F 7A 8DØC ØA 79 8DØC ØA 79 8DØE 26 Ø2 8DØ 8D Ø	* CONSOLE XVEC4	LDU CBU LEAX \$01 STX CBU LDX #CA JSR L8D JJMP LA6. SIN RAM H LDA DEV' CMPA #-3 BNE L8D LEAS \$02 CLR CIN TST CIN BNE L8D COM CIN RTS PSHS LY LDA X+ PSHS LY LDA X+ STX CIN DEC CIN BNE L8D BNE L8D LBD LBD LBD LBD LBD LBD LBD LBD LBD LB	FAD , U FAD SBUF 7C 444 HOOK NUM Ø1 ,S BFL CTR Ø2 BFL PTR PTR PTR CTR	GET BLOCK NUMBER * INCREMENT BLOCK NUMBER * AND SAVE IT USE CASBUF AS DLOAD INPUT BUFFER READ 128 CHARACTERS (ONE BLOCK) INTO BUFFER GET DEVICE NUMBER DLOAD DEVICE NUMBER BRANCH IF NOT OLOAD PURGE 1ST RETURN ADDR FROM STACK RESET EMPTY/FULL FLAG ANY CHARACTERS LEFT IN BUFFER? YES, GO GET ONE SET EMPTY/FULL FLAG TO EOF SAVE REGISTERS GET CONSOLE IN CHARACTER BUFFER GET A CHARACTER SAVE IT ON THE STACK SAVE NEW CHARACTER BUFFER DECREMENT CHARACTER COUNTER RETURN IF BUFFER NOT EMPTY GO REFILL THE CHARACTER BUFFER
1586 1587 1588 1599 1591 1592 1593 1594 1595 1596 1600 1601 1602 1603 1604 1605 1606 1607 1608 1609 1611	8CE4 3Ø 41 8CE6 9F 7E 8CE8 8E Ø1 DA 8CEB BD 8D 7C 8CEE 7E A6 44 8CF1 96 6F 8CF3 81 FD 8CF5 26 ØA 8CF7 32 62 8CF9 ØF 7Ø 8CFB ØD 79 8CFD 26 Ø3 8CFF Ø3 7Ø 8DØ1 39 8DØ2 34 74 8DØ4 9E 7A 8DØ6 A6 8Ø 8DØ8 34 02 8DØA 9F 7A 8DØC ØA 79 8DØC ØA 79 8DØE 26 Ø2 8DØ 8D Ø	L8CE2 L8CE4 * CONSOLE XVEC4 L8DØ1 L8DØ2	LDU CBU LEAX \$01 STX CBU LDX #CA LDX #CA LBD LDX #CA LBD	FAD ,U FAD SBUF 7C 444 HOOK WUM Ø1 ,S BFL CTR Ø2 BFL ,X,B PTR PTR CTR 12 E2 ,X,Y,U,PC	GET BLOCK NUMBER * INCREMENT BLOCK NUMBER * AND SAVE IT USE CASBUF AS DLOAD INPUT BUFFER READ 128 CHARACTERS (ONE BLOCK) INTO BUFFER GET DEVICE NUMBER BLOAD DEVICE NUMBER BRANCH IF NOT OLOAD PURGE 1ST RETURN ADDR FROM STACK RESET EMPTY/FULL FLAG ANY CHARACTERS LEFT IN BUFFER? YES, GO GET ONE SET EMPTY/FULL FLAG TO EOF SAVE REGISTERS GET CONSOLE IN CHARACTER BUFFER GET A CHARACTER SAVE IT ON THE STACK SAVE NEW CHARACTER BUFFER DECREMENT CHARACTER BUFFER DECREMENT CHARACTER BUFFER RETURN IF BUFFER NOT EMPTY GO REFILL THE CHARACTER BUFFER RESTORE REGISTERS AND RETURN
1586 1587 1598 1599 1591 1592 1593 1595 1596 1597 1598 1599 1600 1601 1602 1603 1604 1605 1606 1607 1608	8CE4 3Ø 41 8CE6 9F 7E 8CE8 8E Ø1 DA 8CEB BD 8D 7C 8CEE 7E A6 44 8CF1 96 6F 8CF3 81 FD 8CF5 26 ØA 8CF7 32 62 8CF9 ØF 7Ø 8CFB ØD 79 8CFD 26 Ø3 8CFF Ø3 7Ø 8DØ1 39 8DØ2 34 74 8DØ4 9E 7A 8DØ6 A6 8Ø 8DØ8 34 02 8DØA 9F 7A 8DØC ØA 79 8DØC ØA 79 8DØE 26 Ø2 8DØ 8D Ø	* CONSOLE XVEC4 * L8D01 L8D02 L8D12 * TRANSMI	LDU CBU LEAX \$01 STX CBU LDX #CA LDX #CA LBD LDX #CA LBD	FAD , U FAD SBUF 7C 444 HOOK NUM Ø1 ,S BFL CTR Ø2 BFL PTR PTR PTR CTR	GET BLOCK NUMBER * INCREMENT BLOCK NUMBER * AND SAVE IT USE CASBUF AS DLOAD INPUT BUFFER READ 128 CHARACTERS (ONE BLOCK) INTO BUFFER GET DEVICE NUMBER BLOAD DEVICE NUMBER BRANCH IF NOT OLOAD PURGE 1ST RETURN ADDR FROM STACK RESET EMPTY/FULL FLAG ANY CHARACTERS LEFT IN BUFFER? YES, GO GET ONE SET EMPTY/FULL FLAG TO EOF SAVE REGISTERS GET CONSOLE IN CHARACTER BUFFER GET A CHARACTER SAVE IT ON THE STACK SAVE NEW CHARACTER BUFFER DECREMENT CHARACTER BUFFER DECREMENT CHARACTER BUFFER RETURN IF BUFFER NOT EMPTY GO REFILL THE CHARACTER BUFFER RESTORE REGISTERS AND RETURN
1586 1587 1588 1599 1591 1592 1593 1594 1595 1596 1600 1601 1602 1603 1604 1605 1606 1607 1608 1609 1611 1612	8CE4 3Ø 41 8CE6 9F 7E 8CE8 8E Ø1 DA 8CEB BD 8D 7C 8CEE 7E A6 44 8CF1 96 6F 8CF3 81 FD 8CF5 26 ØA 8CF7 32 62 8CF9 ØF 7Ø 8CFB ØD 79 8CFD 26 Ø3 8CFF Ø3 7Ø 8DØ1 39 8DØ2 34 74 8DØ4 9E 7A 8DØ4 9E 7A 8DØ6 A6 8Ø 8DØ8 34 Ø2 8DØ8 9F 7A 8DØ6 26 Ø2 8DØ8 9F 7A 8DØC ØA 79 8DØC 26 Ø2 8DØ8 9F 7A 8DØC 35 F6	L8D12 L8D12 * TRANSMILBULA	LDU CBU LEAX \$01 STX CBU LDX #CA LDX #CA LBD LDX #CA LBD LDX #CA LBD LBD LEAS \$02 CRR CIN TST CIN LBD LEAS \$02 CRR CIN TST CIN LBD	FAD ,U FAD SBUF 7C 444 HOOK NUM Ø1 ,S BFL CTR Ø2 BFL ,X,B PTR PTR CTR 12 E2 ,X,Y,U,PC MME - READ FILE STATUS FROM S	GET BLOCK NUMBER * INCREMENT BLOCK NUMBER * AND SAVE IT USE CASBUF AS DLOAD INPUT BUFFER READ 128 CHARACTERS (ONE BLOCK) INTO BUFFER GET DEVICE NUMBER BRANCH IF NOT OLOAD PURGE 1ST RETURN ADDR FROM STACK RESET EMPTY/FULL FLAG ANY CHARACTERS LEFT IN BUFFER? YES, GO GET ONE SET EMPTY/FULL FLAG TO EOF SAVE REGISTERS GET CONSOLE IN CHARACTER BUFFER GET A CHARACTER SAVE IT ON THE STACK SAVE NEW CHARACTER BUFFER DECREMENT CHARACTER BUFFER RETURN IF BUFFER NOT EMPTY GO REFILL THE CHARACTER BUFFER RESTOR REGISTERS AND RETURN ENDER RESET RESET ATTEMPT COUNTER RESET RESET ATTEMPT COUNTER RESET REGISTERS AND RETURN ENDER RESET ATTEMPT COUNTER SAVE SPACE ON STACK FOR TEMP VARIABLES
1586 1587 1588 1589 1591 1592 1593 1594 1595 1596 1597 1598 1599 1600 1601 1602 1603 1604 1606 1607 1606 1607 1610 1610 1611 1612 1613 1614	8CE4 3Ø 41 8CE6 9F 7E 8CE8 8E Ø1 DA 8CEB BD 8D 7C 8CEE 7E A6 44 8CF1 96 6F 8CF3 81 FD 8CF5 26 ØA 8CF7 32 62 8CF9 ØF 7Ø 8CFB ØD 79 8CFB ØD 79 8CFB ØD 79 8CFD 26 Ø3 8CFF Ø3 7Ø 8DØ1 39 8DØ2 34 74 8DØ4 9E 7A 8DØ6 A6 8Ø 8DØ8 34 Ø2 8DØA 9F 7A 8DØC ØA 79 8DØB 26 Ø2 8D1Ø 8D DØ 8D12 35 F6 8D14 4F 8D15 34 16 8D17 31 E4	* CONSOLE XVEC4 * CONSOLE XVEC4 L8D01 L8D02 * TRANSMI L8D14	LDU GBU LEAX \$01 STX CBU LDX #CA LBD JSR LBD JMP LA6 : IN RAM H LDA DEV: CMPA #-3 BNE LBD LEAS \$02 CUR GIN LEAS \$03 CUR GIN L	FAD ,U FAD SBUF 7C 444 HOOK NUM Ø1 ,S SBFL CCTR 02 BFL ,X,B PTR PTR CTR 12 E2 ,X,Y,U,PC MME - READ FILE STATUS FROM S ,A	GET BLOCK NUMBER * INCREMENT BLOCK NUMBER * AND SAVE IT USE CASBUF AS DLOAD INPUT BUFFER READ 128 CHARACTERS (ONE BLOCK) INTO BUFFER GET DEVICE NUMBER DLOAD DEVICE NUMBER BRANCH IF NOT OLOAD PURGE 1ST RETURN ADDR FROM STACK RESET EMPTY/FULL FLAG ANY CHARACTERS LEFT IN BUFFER? YES, GO GET ONE SET EMPTY/FULL FLAG TO EOF SAVE REGISTERS GET CONSOLE IN CHARACTER BUFFER GET A CHARACTER BUFFER DECREMENT CHARACTER BUFFER DECREMENT CHARACTER COUNTER RETURN IF BUFFER NOT EMPTY GO REFILL THE CHARACTER BUFFER RESTORE REGISTERS AND RETURN ENDER
1586 1587 1588 1599 1591 1592 1593 1594 1595 1596 1597 1598 1600 1601 1602 1603 1604 1605 1606 1606 1611 1612 1613 1614 1614 1615	8CE4 3Ø 41 8CE6 9F 7E 8CE8 8E Ø1 DA 8CEB BD 8D 7C 8CEE 7E A6 44 8CF1 96 6F 8CF3 81 FD 8CF5 26 ØA 8CF7 32 62 8CF9 ØF 7Ø 8CFB ØD 79 8CFD 26 Ø3 8CFF Ø3 7Ø 8DØ1 39 8DØ2 34 74 8DØ6 A6 8Ø 8DØ8 34 02 8DØA 9F 7A 8DØC ØA 79 8DØE 26 Ø2 8DØA 9F 7A 8DØC ØA 79 8DØE 26 Ø2 8DØA 9F 7A 8DØC ØA 79 8DØE 26 Ø2 8DØA 9F 7A 8DØC ØA 79 8DØE 26 Ø2 8DØA 9F 7A 8DØC ØA 79 8DØE 26 Ø2 8DØA 9F 7A 8DØC ØA 79 8DØE 26 Ø2 8DØA 9F 7A 8DØC ØA 79 8DØE 36 Ø2 8DØA 9F 7A 8DØC ØA 79 8DØE 36 Ø2 8DØA 9F 7A 8DØC ØA 79 8DØE 36 Ø2 8DØA 9F 7A 8DØC ØA 79 8DØE 36 Ø2 8DØA 9F 7A 8DØC ØA 79 8DØE 36 Ø2 8DØA 9F 7A 8DØC ØA 79 8DØE 36 Ø2 8DØA 9F 7A 8DØC ØA 79 8DØE 36 Ø2 8DØA 9F 7A 8DØC ØA 79 8DØE 36 Ø2 8DØA 9F 7A 8DØC ØA 79 8DØE 36 Ø2 8DØA 9F 7A 8DØC ØA 79 8DØE 36 Ø2 8DØA 9F 7A 8DØC ØA 79 8DØE 36 Ø2 8DØA 9F 7A 8DØC ØA 79 8DØE 36 Ø2 8DØA 9F 7A 8DØC ØA 79 8DØE 36 Ø2 8DØA 9F 7A 8DØC ØA 79 8DØE 36 Ø2 8DØA 9F 7A 8DØC ØA 79 8DØE 36 Ø2 8DØA 9F 7A 8DØC ØA 79 8DØE 36 Ø2 8DØA 9F 7A 8DØC ØA 79 8DØE 36 Ø2 8DØA 9F 7A 8DØC ØA 79 8DØE 36 Ø2 8DØA 9F 7A 8DØE 36 Ø2 8DØE	L8D12 * TRANSMIL8D14	LDU CBU LEAX \$01 STX CBU LDX #CA JSR L8D JMP LA6. : IN RAM H LDA DEV CMPA #-3 BNE L8D BNE L8D LEAS \$02 CLR CIN TST CIN BNE L8D COM CIN RTS PSHS U,Y LDA ,X PSHS A STX CIN DEC CIN BNE L8D CRA RS PSHS A STX CIN DEC CIN BNE L8D CCM CCN RS PSHS A STX CIN LDA ,X PSHS A STX CIN LDA ,X PSHS A STX CIN LCA CCIN BNE L8D BNE L8	FAD ,U FAD SBUF 7C 44 HOOK NUM ### ### ### ### ### ### ### ### ###	GET BLOCK NUMBER * INCREMENT BLOCK NUMBER * AND SAVE IT USE CASBUF AS DLOAD INPUT BUFFER READ 128 CHARACTERS (ONE BLOCK) INTO BUFFER RESET CONSOLE IN BUFFER GET DEVICE NUMBER BRANCH IF NOT OLOAD PURGE 1ST RETURN ADDR FROM STACK RESET EMPTY/FULL FLAG ANY CHARACTERS LEFT IN BUFFER? YES, GO GET ONE SET EMPTY/FULL FLAG TO EOF SAVE REGISTERS GET CONSOLE IN CHARACTER BUFFER GET A CHARACTER SLEFT SAVE IT ON THE STACK SAVE NEW CHARACTER BUFFER DECREMENT CHARACTER BUFFER DECREMENT CHARACTER BUFFER DECREMENT CHARACTER SUFFER RETURN IF BUFFER NOT EMPTY GO REFILL THE CHARACTER BUFFER RESTORE REGISTERS AND RETURN ENDER RESET ATTEMPT COUNTER SAVE SPACE ON STACK FOR TEMP VARIABLES STACK TO Y (TFR S, Y) - SAVE VARIABLE POINTER
1586 1587 1588 1589 1591 1592 1593 1594 1595 1596 1597 1598 1599 1600 1601 1605 1606 1607 1608 1609 1611 1612 1614 1615 1616	8CE4 3Ø 41 8CE6 9F 7E 8CE8 8E Ø1 DA 8CEB BD 8D 7C 8CEE 7E A6 44 8CF1 96 6F 8CF3 81 FD 8CF5 26 ØA 8CF7 32 62 8CF9 ØF 7Ø 8CFB ØD 79 8CFD 26 Ø3 8CFF Ø3 7Ø 8DØ1 39 8DØ2 34 74 8DØ4 9E 7A 8DØ6 A6 8Ø 8DØ8 34 Ø2 8DØA 9F 7A 8DØC A6 A7 8DØC 26 Ø2 8DØC 26 Ø2 8DØC 26 Ø2 8DØC 27 8D	L8D12 * TRANSMILBUILBUILBUILBUILBUILBUILBUILBUILBUILBU	LDU CBU LEAX \$01 STX CBU LDX #CA LDX #CA LBDX #CA LBDX HSA LBD JMP LA6. : IN RAM H LDA DEV! CMPA #-3 BNE LBD LEAS \$02 CUR CIN LDA X+ PSHS A STX CIN LDA X- L	FAD ,U FAD SBUF 7C 444 HOOK NUM Ø1 ,S BFL CTR Ø2 BFL ,X,B PTR PTR PTR CTR 12 E2 ,X,Y,U,PC MME - READ FILE STATUS FROM S ,A	GET BLOCK NUMBER * INCREMENT BLOCK NUMBER * AND SAVE IT USE CASBUF AS DLOAD INPUT BUFFER READ 128 CHARACTERS (ONE BLOCK) INTO BUFFER GET DEVICE NUMBER BANCH IF NOT OLOAD PURGE 1ST RETURN ADDR FROM STACK RESET EMPTY/FULL FLAG ANY CHARACTERS LEFT IN BUFFER? YES, GO GET ONE SET EMPTY/FULL FLAG TO EOF SAVE REGISTERS GET CONSOLE IN CHARACTER BUFFER GET A CHARACTER SAVE IT ON THE STACK SAVE NEW CHARACTER SUFFER EVENTY OF THE STACK SAVE NEW CHARACTER BUFFER RETURN IF BUFFER NOT EMPTY GO REFILL THE CHARACTER BUFFER RESTORE REGISTERS AND RETURN ENDER RESET ATTEMPT COUNTER SAVE SAVE SAVE ON STACK FOR TEMP VARIABLES STACK TO Y (TFR S,Y) - SAVE VARIABLE POINTER INCREMENT ATTEMPT COUNTER
1586 1587 1588 1589 1591 1592 1593 1594 1595 1596 1597 1598 1599 1600 1601 1602 1603 1604 1606 1607 1610 1610 1611 1612 1613 1615 1616 1615 1616 1617 1616	8CE4 3Ø 41 8CE6 9F 7E 8CE8 8E Ø1 DA 8CEB BD 8D 7C 8CEE 7E A6 44 8CF1 96 6F 8CF3 81 FD 8CF5 26 ØA 8CF7 32 62 8CF9 ØF 7Ø 8CFB ØD 79 8CFD 26 Ø3 8CFF 03 7Ø 8DØ1 39 8DØ2 34 74 8DØ4 9E 7A 8DØ6 A6 8Ø 8DØ8 34 Ø2 8DØA 9F 7A 8DØC ØA 79 8DØE 6 Ø2 8D1Ø 8D DØ 8D12 35 F6 8D14 4F 8D15 34 16 8D17 31 E4 8D19 2Ø Ø2 8D18 8D 2B	* CONSOLE XVEC4 * CONSOLE XVEC4 L8D01 L8D02 * TRANSMI L8D14 L8D1B L8D1D	LDU CBU LEAX \$01 STX CBU LDX #CA JUSP LA6 : IN RAM ! LDA DEV: CMPA #-3 BNE L8D LEAS \$02 CUR CIN BNE L8D LEAS \$02 CUR CIN BNE L8D LEAS \$02 CUR CIN BNE L8D LDA ,X+ PSHS CIN BNE L8D LDA ,X+ PSHS A,B T FILE N BNE L8D LDA ,X+ PSHS A,B T FILE X BRA L8D BSR L8C BRA L8D BSR L8D LBAY ,S BRA L8D BSR L8D LBAY ,S BRA L8D BSR L8D LBAY ,S BRA L8D BSR L8D LBA #58.	FAD ,U FAD SBUF 7C 444 HOOK NUM Ø1 ,S SBFL CCTR 02 BFL ,X,B PTR PTR CTR 112 E2 ,X,Y,U,PC ,X,Y,U,PC ,X,Y,U,PC ,X,Y,U,PC ,X,A A ID	GET BLOCK NUMBER * INCREMENT BLOCK NUMBER * AND SAVE IT USE CASBUF AS DLOAD INPUT BUFFER READ 128 CHARACTERS (ONE BLOCK) INTO BUFFER GET DEVICE NUMBER DLOAD DEVICE NUMBER BRANCH IF NOT OLOAD PURGE 1ST RETURN ADDR FROM STACK RESET EMPTY/FULL FLAG ANY CHARACTERS LEFT IN BUFFER? YES, GO GET ONE SET EMPTY/FULL FLAG TO EOF SAVE REGISTERS GET CONSOLE IN CHARACTER BUFFER GET A CHARACTER SAVE IT ON THE STACK SAVE NEW CHARACTER BUFFER DECREMENT CHARACTER BUFFER DECREMENT CHARACTER BUFFER DECREMENT CHARACTER SUFFER RESTOR REGISTERS AND RETURN ENDER RESET ATTEMPT COUNTER RESTOR TEMPTY COUNTER SAVE SPACE ON STACK FOR TEMP VARIABLES STACK TO Y (TFR S, Y) - SAVE VARIABLE POINTER * GET FILE REQUEST CONTROL CODE
1586 1587 1588 1589 1591 1592 1593 1594 1595 1596 1600 1601 1602 1603 1604 1605 1606 1607 1608 1609 1611 1612 1613 1614 1615 1616 1617 1616	8CE4 3Ø 41 8CE6 9F 7E 8CE8 8E Ø1 DA 8CEB BD 8D 7C 8CEE 7E A6 44 8CF1 96 6F 8CF3 81 FD 8CF5 26 ØA 8CF7 32 62 8CF9 ØF 7Ø 8CFB ØD 79 8CFD 26 Ø3 8CFF Ø3 7Ø 8DØ1 39 8DØ2 34 74 8DØ4 9E 7A 8DØ6 A6 8Ø 8DØ8 34 Ø2 8DØA 9F 7A 8DØC A6 A7 8DØC 26 Ø2 8DØC 26 Ø2 8DØC 26 Ø2 8DØC 27 8D	* CONSOLE XVEC4 * CONSOLE XVEC4 L8D01 L8D02 * TRANSMI L8D14 L8D1B L8D1D	LDU CBU LEAX \$01 STX CBU LDX #CA JUSP LA6 SIN RAM H LDA DEV: CMPA #-3 BNE L8D LEAS \$02 CUR CIN BNE L8D LCAS \$02 CUR CIN BNE L8D LDA AFR STX CIN BNE L8D LDA AFR STX CIN BNE L8D LDA AFR STX CIN BNE L8D LCAS \$02 CRR CIN BNE L8D LCAS \$02 CRR CIN BNE L8D LCAS \$02 CRR CIN BNE L8D LCAS \$02 COMBA	FAD ,U FAD SBUF 7C 444 HOOK NUM Ø1 ,S BFL CTR CZ EZ SK,Y,U,PC MME - READ FILE STATUS FROM S ,A 10 48 A 58	GET BLOCK NUMBER * INCREMENT BLOCK NUMBER * AND SAVE IT USE CASBUF AS DLOAD INPUT BUFFER READ 128 CHARACTERS (ONE BLOCK) INTO BUFFER GET DEVICE NUMBER BANCH IF NOT OLOAD PURGE 1ST RETURN ADDR FROM STACK RESET EMPTY/FULL FLAG ANY CHARACTERS LEFT IN BUFFER? YES, GO GET ONE SET EMPTY/FULL FLAG TO EOF SAVE REGISTERS GET CONSOLE IN CHARACTER BUFFER GET A CHARACTER SAVE IT ON THE STACK SAVE NEW CHARACTER BUFFER DECREMENT CHARACTER BUFFER RETURN IF BUFFER NOT EMPTY GO REFILL THE CHARACTER BUFFER RESTORE REGISTERS AND RETURN ENDER RESET ATTEMPT COUNTER RESET ATTEMPT COUNTER RESET ATTEMPT COUNTER RESET ATTEMPT COUNTER SAVE SPACE ON STACK FOR TEMP VARIABLES STACK TO Y (TFR S,Y) - SAVE VARIABLE POINTER INCREMENT ATTEMPT COUNTER
1586 1587 1588 1589 1591 1592 1593 1594 1595 1596 1600 1601 1602 1603 1604 1605 1606 1607 1608 1609 1611 1612 1613 1614 1615 1616 1617 1616	8CE4 3Ø 41 8CE6 9F 7E 8CE8 8E Ø1 DA 8CEB BD 8D 7C 8CEE 7E A6 44 8CF1 96 6F 8CF3 81 FD 8CF5 26 ØA 8CF7 32 62 8CF9 ØF 7Ø 8CFB ØD 79 8CFD 26 Ø3 8CFF Ø3 7Ø 8DØ1 39 8DØ2 34 74 8DØ4 9E 7A 8DØ6 A6 8Ø 8DØ8 34 Ø2 8DØA 9F 7A 8DØ6 ØA 79 8DØE 26 Ø2 8DIØ 8D DØ 8D1Ø 8D DØ 8D1Ø 35 F6 8D14 4F 8DØC ØA 79 8DØE 26 Ø2 8D1Ø 8D DØ 8D12 35 F6 8D14 4F 8D15 34 16 8D17 31 E4 8D19 2Ø Ø2 8D1B 8D 2B 8D1D 86 8A 8D1F 8D 37 8D1E 26 F8	* CONSOLE XVEC4 * CONSOLE XVEC4 L8D01 L8D02 * TRANSMI L8D14 L8D1B L8D1D	LDU CBU LEAX \$01 STX CBU LDX #CA JUSP LA6 SIN RAM H LDA DEV: CMPA #-3 BNE L8D LEAS \$02 CUR CIN BNE L8D LCAS \$02 CUR CIN BNE L8D LDA AFR STX CIN BNE L8D LDA AFR STX CIN BNE L8D LDA AFR STX CIN BNE L8D LCAS \$02 CRR CIN BNE L8D LCAS \$02 CRR CIN BNE L8D LCAS \$02 CRR CIN BNE L8D LCAS \$02 COMBA	FAD ,U FAD SBUF 7C 444 HOOK NUM Ø1 ,S BFL CTR CZ EZ SK,Y,U,PC MME - READ FILE STATUS FROM S ,A 10 48 A 58	GET BLOCK NUMBER * INCREMENT BLOCK NUMBER * AND SAVE IT USE CASBUF AS DLOAD INPUT BUFFER READ 128 CHARACTERS (ONE BLOCK) INTO BUFFER RESET CONSOLE IN BUFFER GET DEVICE NUMBER BRANCH IF NOT OLOAD PURGE 1ST RETURN ADDR FROM STACK RESET EMPTY/FULL FLAG ANY CHARACTERS LEFT IN BUFFER? YES, GO GET ONE SET EMPTY/FULL FLAG TO EOF SAVE REGISTERS GET CONSOLE IN CHARACTER BUFFER GET A CHARACTER SAVE IT ON THE STACK SAVE NEW CHARACTER BUFFER DECREMENT CHARACTER BUFFER DECREMENT CHARACTER BUFFER RETURN IF BUFFER NOT EMPTY GO REFILL THE CHARACTER BUFFER RESET ATTEMPT COUNTER RESET ATTEMPT COUNTER SAVE SPACE ON STACK FOR TEMP VARIABLES STACK TO Y (TFR S,Y) - SAVE VARIABLE POINTER INCREMENT ATTEMPT COUNTER * GET FILE REQUEST CONTROL CODE * AND TRANSMIT IT
1586 1587 1588 1599 1591 1592 1593 1594 1595 1596 1600 1601 1602 1603 1604 1605 1606 1607 1608 1609 1611 1612 1613 1614 1615 1616 1617 1618	8CE4 3Ø 41 8CE6 9F 7E 8CE8 8E Ø1 DA 8CEB BD 8D 7C 8CEE 7E A6 44 8CF1 96 6F 8CF3 81 FD 8CF5 26 ØA 8CF7 32 62 8CF9 ØF 7Ø 8CFB ØD 79 8CFD 26 Ø3 8CFF Ø3 7Ø 8DØ1 39 8DØ2 34 74 8DØ4 9E 7A 8DØ6 A6 8Ø 8DØ8 34 02 8DØ8 34 02 8DØ8 34 02 8DØ8 35 F6 8D14 4F 8D15 35 F6 8D14 4F 8D15 34 16 8D17 31 E4 8D19 2Ø 02 8D18 8D 2B 8D10 86 8A 8D17 8D 37 8D21 26 F8 8D11 8D 37 8D21 26 F8 8D11 8D 37 8D21 26 F8 8D13 8D 37 8D21 26 F8 8D14 8D 37 8D21 26 F8 8D15 8D 37 8D21 26 F8 8D16 8D 37 8D21 26 F8 8D23 8E Ø1 D2 8D26 A6 8Ø	LBD12 * TRANSMI LBD14 LBD1B LBD1D	LDU CBU LEAX \$01 STX CBU LDX #CA JSR L8D JMP LA6. : IN RAM LDA DEVI CMPA #-3 BNE L8D BNE L8D LEAS \$02 CLR CIN TST CIN BNE L8D COM CIN RTS PSHS U,Y LDX CIN LDA X+ PSHS A STX CIN DEC CIN BNE L8D	FAD ,U FAD SBUF 7C 444 HOOK WUM Ø1 ,S BFL CTR Ø2 BFL CTR Ø2 FL CTR 12 E2 ,X,Y,U,PC MME - READ FILE STATUS FROM S ,A ID 48 A A B B B BBUF+1	GET BLOCK NUMBER * INCREMENT BLOCK NUMBER * AND SAVE IT USE CASBUF AS DLOAD INPUT BUFFER READ 128 CHARACTERS (ONE BLOCK) INTO BUFFER RESET CONSOLE IN BUFFER GET DEVICE NUMBER BLOAD DEVICE NUMBER BRANCH IF NOT OLOAD PURGE 1ST RETURN ADDR FROM STACK RESET EMPTY/FULL FLAG ANY CHARACTERS LEFT IN BUFFER? YES, GO GET ONE SET EMPTY/FULL FLAG TO EOF SAVE REGISTERS GET CONSOLE IN CHARACTER BUFFER GET A CHARACTER SAVE IT ON THE STACK SAVE NEW CHARACTER BUFFER DECREMENT CHARACTER BUFFER DECREMENT CHARACTER BUFFER DECREMENT CHARACTER BUFFER DECREMENT CHARACTER BUFFER BUFFER NOT EMPTY GO REFILL THE CHARACTER BUFFER RESTORE REGISTERS AND RETURN ENDER RESET ATTEMPT COUNTER SAVE SPACE ON STACK FOR TEMP VARIABLES STACK TO Y (TFF S, Y) - SAVE VARIABLE POINTER INCREMENT ATTEMPT COUNTER * GET FILE REQUEST CONTROL CODE * AND TRANSMIT IT BRANCH IF NO ECHO OR ERROR POINT TO CASS FILE NAME BUFFER GET CHARACTER FROM NAME BUFFER GET CHARACTER FROM NAME BUFFER
1586 1587 1588 1589 1591 1592 1593 1594 1595 1596 1697 1600 1601 1602 1603 1604 1604 1605 1606 1607 1608 1611 1612 1613 1614 1615 1616 1617 1618 1617 1618 1619 1620 1621 1622 1623	8CE4 3Ø 41 8CE6 9F 7E 8CE8 8E Ø1 DA 8CEB BD 8D 7C 8CEE 7E A6 44 8CF1 96 6F 8CF3 81 FD 8CF5 26 ØA 8CF7 32 62 8CF9 ØF 7Ø 8CFB D0 79 8CFD 26 Ø3 8CFF Ø3 7Ø 8DØ1 39 8CFF Ø3 7Ø 8DØ1 39 8DØ2 34 74 8DØ4 9E 7A 8DØ6 A6 8Ø 8DØ8 34 Ø2 8DØA 9F 7A 8DØ6 06 Ø2 8D1Ø 8D DØ 8D12 35 F6 8D14 4F 8D15 34 16 8D17 31 E4 8D19 2Ø 02 8D18 8D DØ 8D12 35 F6 8D14 4F 8D15 34 16 8D17 31 E4 8D19 2Ø 02 8D18 8D DØ 8D12 35 F6 8D14 4F 8D15 34 16 8D17 31 E4 8D19 2Ø 02 8D18 8D DØ 8D12 35 F6	L8D12 * TRANSMI L8D14 L8D18 L8D14 L8D18 L8D19 L8D18 L8D10	LDU CBU LEAX \$01 STX CBU LDX #CA LDX #CA LBD #CA LBD H LDA DEV CMPA #-3 BNE LBD LEAS \$02 CUR CIN BNE LBD LDA X+ PSHS A LBD LDA X+ PSHS A LBD BCC CIN BNE LBD BCC CIN BNE LBD BCC CIR LBA	FAD ,U FAD SBUF 7C 444 HOOK NUM Ø1 ,S BFL CTR Ø2 BFL ,X,B PTR PTR CTR 12 E2 E2 E2 E2 E3 A,Y,U,PC MME - READ FILE STATUS FROM S ,A 1D 48 A BB BBUF+1 894	GET BLOCK NUMBER * INCREMENT BLOCK NUMBER * AND SAVE IT USE CASBUF AS DLOAD INPUT BUFFER READ 128 CHARACTERS (ONE BLOCK) INTO BUFFER RESET CONSOLE IN BUFFER GET DEVICE NUMBER BRANCH IF NOT OLOAD PURGE 1ST RETURN ADDR FROM STACK RESET EMPTY/FULL FLAG ANY CHARACTERS LEFT IN BUFFER? YES, GO GET ONE SET EMPTY/FULL FLAG TO EOF SAVE REGISTERS GET CONSOLE IN CHARACTER BUFFER GET A CHARACTER SAVE IT ON THE STACK SAVE NEW CHARACTER BUFFER DECREMENT CHARACTER BUFFER RESTORE REGISTERS AND RETURN ENDER RESTOR REGISTERS AND RETURN ENDER INCREMENT ATTEMPT COUNTER * GET FILE REQUEST CONTROL CODE * AND TRANSMIT IT BRANCH IF NO ECHO OR ERROR POINT TO CASS FILE NAME BUFFER GET CHARACTER FROM NAME BUFFER OUTPUT IT TO RS 232 PORT
1586 1587 1598 1599 1591 1592 1593 1594 1595 1596 1597 1598 1599 1600 1601 1602 1603 1604 1606 1607 1610 1611 1612 1613 1614 1615 1616 1617 1618 1619 1621 1622 1622 1622	8CE4 3Ø 41 8CE6 9F 7E 8CE8 8E Ø1 DA 8CEB BD 8D 7C 8CEE 7E A6 44 8CF1 96 6F 8CF3 81 FD 8CF5 26 ØA 8CF7 32 62 8CF9 ØF 7Ø 8CFB ØD 79 8D002 24 74 8D04 9E 7A 8D06 A6 8Ø 8D08 34 Ø2 8D08 9F 7A 8D0C ØA 79 8D0E 26 Ø2 8D10 8D 0Ø 8D12 35 F6 8D14 4F 8D15 34 16 8D17 31 E4 8D19 2Ø 02 8D18 8D 2B 8D18 8D 2B 8D1B 8D 2B 8D1B 8D 2B 8D1B 8D 2B 8D1B 8D 37 8D21 26 F8 8D23 8E Ø1 D2 8D26 A6 8Ø 8D28 8D 8E Ø4 8D28 8D 8E Ø4 8D28 8D 8E Ø4	L8CE2 L8CE4 * CONSOLE XVEC4 L8D01 L8D02 * TRANSMI L8D14 L8D1B L8D1D	LDU GBU LEAX \$01 STX GBU LDX #CA BD JSR L8D JMP LA6 : IN RAM ! LDA DEV: CMPA #-3 BBNE L8D LEAS \$02 CUR GIN BBNE L8D LEAS \$02 CUR GIN BBNE L8D LDA ,X+ PSHS A,B T FILE N BBNE L8D BSR L8C BBNE L8D BSR L8C BBNE L8D LDA ,X+ BSTX GIN BBNE L8D BSR L8C BBNE L8D LDA #58 BSR L8D BSR L8D LDA #58 BSR L8D LDA #58 BSR L8D LDA #58 BSR L8D LDA #58 LBD	FAD , U FAD SBUF 7C 444 HOOK NUM Ø1 ,S SBFL CCTR 02 BFL ,X,B PTR PTR CTR 112 E2 ,X,Y,U,PC ,ME - READ FILE STATUS FROM S ,A 1D 48 A A 58 NBUF+1 Ø4 NBUF+9	GET BLOCK NUMBER * INCREMENT BLOCK NUMBER * AND SAVE IT USE CASBUF AS DLOAD INPUT BUFFER READ 128 CHARACTERS (ONE BLOCK) INTO BUFFER RESET CONSOLE IN BUFFER GET DEVICE NUMBER DLOAD DEVICE NUMBER BRANCH IF NOT OLOAD PURGE 1ST RETURN ADDR FROM STACK RESET EMPTY/FULL FLAG ANY CHARACTERS LEFT IN BUFFER? YES, GO GET ONE SET EMPTY/FULL FLAG TO EOF SAVE REGISTERS GET CONSOLE IN CHARACTER BUFFER GET A CHARACTER SAVE IT ON THE STACK SAVE NEW CHARACTER BUFFER DECREMENT CHARACTER BUFFER DECREMENT CHARACTER BUFFER RESTORE REGISTERS AND RETURN ENDER RESET ATTEMPT COUNTER RESTOR REGISTERS AND RETURN ENDER INCREMENT ATTEMPT COUNTER * GET FILE REQUEST CONTROL CODE * AND TRANSMIT IT BRANCH IF NO ECHO OR ERROR POINT TO CASS FILE NAME BUFFER GET CHARACTER FROM NAME BUFFER GUTPUT IT TO RS 232 PORT COMPARE TO END OF BUFFER
1586 1587 1588 1599 1591 1592 1593 1594 1595 1596 1600 1601 1602 1603 1604 1605 1606 1607 1608 1609 1611 1612 1613 1614 1615 1616 1617 1618 1619 1611 1612 1613 1614 1615 1616 1617 1618 1619 1619 1620 1621 1622 1623 1624 1625 1626 1627 1628 1629 1629 1629 1629 1629 1629 1629 1629	8CE4 3Ø 41 8CE6 9F 7E 8CE8 8E Ø1 DA 8CEB BD 8D 7C 8CEE 7E A6 44 8CF1 96 6F 8CF3 81 FD 8CF5 26 ØA 8CF7 32 62 8CF9 ØF 7Ø 8CFB ØD 79 8CFD 26 Ø3 8CFF Ø3 7Ø 8DØ1 39 8DØ2 34 74 8DØ4 9E 7A 8DØ6 A6 8Ø 8DØ8 34 Ø2 8DØ8 34 Ø2 8DØ8 34 Ø2 8DØ8 35 F6 8D14 4F 8D15 34 16 8D17 31 E4 8D19 2Ø Ø2 8D18 8D 2B 8D12 85 F6 8D14 4F 8D15 34 16 8D17 31 E4 8D19 2Ø Ø2 8D18 8D 2B 8D10 86 8A 8D17 8D 37 8D21 26 F8 8D18 8D 37 8D21 26 F8 8D23 8E Ø1 D2 8D26 A6 8Ø 8D28 BC Ø1 DA 8D2E 26 F6	LBD12 * TRANSMI LBD14 LBD18 LBD19	LDU CBU LEAX \$01 STX CBU LDX #CA JSR L8D JMP LA6. : IN RAM LDA DEVI CMPA #-3 BNE L8D BNE L8D LEAS \$02 CLR CIN TST CIN BNE L8D LCAS \$02 CLR CIN TST CIN BNE L8D LDA X+ PSHS A STX CIN DEC CIN BNE L8D LDA #\$4. BNE L8D LDA #\$8. BNA L8D LDA #\$8. BNA L8D LDA #\$8. BNA L8D LDA #\$7. BNA LBA LCAY LCRA LCRA LCRA LCRA LCRA LCRA LCRA LCRA	FAD ,U FAD SBUF 7C 444 1000K WIM ### ### ### ### ### ### ### ### ###	GET BLOCK NUMBER * INCREMENT BLOCK NUMBER * AND SAVE IT USE CASBUF AS DLOAD INPUT BUFFER READ 128 CHARACTERS (ONE BLOCK) INTO BUFFER RESET CONSOLE IN BUFFER GET DEVICE NUMBER BRANCH IF NOT OLOAD PURGE 1ST RETURN ADDR FROM STACK RESET EMPTY/FULL FLAG ANY CHARACTERS LEFT IN BUFFER? YES, GO GET ONE SET EMPTY/FULL FLAG TO EOF SAVE REGISTERS GET CONSOLE IN CHARACTER BUFFER GET A CHARACTER SAVE IT ON THE STACK SAVE NEW CHARACTER BUFFER DECREMENT CHARACTER BUFFER DECREMENT CHARACTER BUFFER RETURN IF BUFFER NOT EMPTY GO REFILL THE CHARACTER BUFFER RESET ATTEMPT COUNTER RESET ATTEMPT COUNTER SAVE SPACE ON STACK FOR TEMP VARIABLES STACK TO Y (TFR S, Y) - SAVE VARIABLE POINTER * GET FILE REQUEST CONTROL CODE * AND TRANSMIT IT BRANCH IF NO ECHO OR ERROR POINT TO CASS FILE NAME BUFFER OUTPUT IT TO RS 232 PORT COMPARE TO END OF BUFFER OUTPUT IT TO RS 232 PORT COMPARE TO END OF BUFFER OUTPUT IT TO RS 232 PORT COMPARE TO END OF BUFFER OUTPUT ILD ONE
1586 1587 1588 1599 1591 1592 1593 1594 1595 1596 1600 1601 1602 1603 1604 1605 1606 1607 1608 1609 1611 1612 1613 1614 1615 1616 1617 1618 1619 1611 1612 1613 1614 1615 1616 1617 1618 1619 1619 1620 1621 1622 1623 1624 1625 1626 1627 1628 1629 1629 1629 1629 1629 1629 1629 1629	8CE4 3Ø 41 8CE6 9F 7E 8CE8 8E Ø1 DA 8CEB BD 8D 7C 8CEE 7E A6 44 8CF1 96 6F 8CF3 81 FD 8CF5 26 ØA 8CF7 32 62 8CF9 ØF 7Ø 8CFB ØD 79 8D002 24 74 8D04 9E 7A 8D06 A6 8Ø 8D08 34 Ø2 8D08 9F 7A 8D0C ØA 79 8D0E 26 Ø2 8D10 8D 0Ø 8D12 35 F6 8D14 4F 8D15 34 16 8D17 31 E4 8D19 2Ø 02 8D18 8D 2B 8D18 8D 2B 8D1B 8D 2B 8D1B 8D 2B 8D1B 8D 2B 8D1B 8D 37 8D21 26 F8 8D23 8E Ø1 D2 8D26 A6 8Ø 8D28 8D 8E Ø4 8D28 8D 8E Ø4 8D28 8D 8E Ø4	L8CE2 L8CE4 * CONSOLE XVEC4 L8D01 L8D02 * TRANSMI L8D14 L8D1B L8D1D L8D26	LDU GBU LEAX \$01 STX GBU LDX #CA BD JSR L8D JMP LA6 : IN RAM ! LDA DEV: CMPA #-3 BBNE L8D LEAS \$02 CUR GIN BBNE L8D LEAS \$02 CUR GIN BBNE L8D LDA ,X+ PSHS A,B T FILE N BBNE L8D BSR L8C BBNE L8D BSR L8C BBNE L8D LDA ,X+ BSTX GIN BBNE L8D BSR L8C BBNE L8D LDA #58 BSR L8D BSR L8D LDA #58 BSR L8D LDA #58 BSR L8D LDA #58 BSR L8D LDA #58 LBD	FAD ,U FAD SBUF 7C 444 HOOK NUM Ø1 ,S BFL CTR Ø2 BFL ,X,B PTR PTR CTR 12 E2 E2 E2 E2 E3 A4 10 48 A BB BB BBUF+1 Ø4 BBUF+1 Ø4 BBUF+9 266	GET BLOCK NUMBER * INCREMENT BLOCK NUMBER * AND SAVE IT USE CASBUF AS DLOAD INPUT BUFFER READ 128 CHARACTERS (ONE BLOCK) INTO BUFFER RESET CONSOLE IN BUFFER GET DEVICE NUMBER DLOAD DEVICE NUMBER BRANCH IF NOT OLOAD PURGE 1ST RETURN ADDR FROM STACK RESET EMPTY/FULL FLAG ANY CHARACTERS LEFT IN BUFFER? YES, GO GET ONE SET EMPTY/FULL FLAG TO EOF SAVE REGISTERS GET CONSOLE IN CHARACTER BUFFER GET A CHARACTER SAVE IT ON THE STACK SAVE NEW CHARACTER BUFFER DECREMENT CHARACTER BUFFER DECREMENT CHARACTER BUFFER RESTORE REGISTERS AND RETURN ENDER RESET ATTEMPT COUNTER RESTOR REGISTERS AND RETURN ENDER INCREMENT ATTEMPT COUNTER * GET FILE REQUEST CONTROL CODE * AND TRANSMIT IT BRANCH IF NO ECHO OR ERROR POINT TO CASS FILE NAME BUFFER GET CHARACTER FROM NAME BUFFER GUTPUT IT TO RS 232 PORT COMPARE TO END OF BUFFER
1586 1587 1588 1589 1591 1592 1593 1594 1595 1596 1600 1601 1602 1603 1604 1605 1606 1607 1608 1609 1611 1612 1613 1614 1615 1616 1617 1618 1619 1620 1621 1622 1623 1624 1625 1626 1627 1628	8CE4 30 41 8CE6 9F 7E 8CE8 8E 01 DA 8CEB BD 8D 7C 8CEE 7E A6 44 8CF1 96 6F 8CF3 81 FD 8CF5 26 0A 8CF7 32 62 8CF9 0F 70 8CFB 0D 79 8D02 34 74 8D06 A6 80 8D08 34 02 8D08 34 02 8D08 34 02 8D08 34 02 8D08 35 F6 8D14 4F 8D15 34 16 8D17 31 E4 8D19 20 02 8D10 8D 00 8D12 35 F6 8D14 4F 8D15 34 16 8D17 31 E4 8D19 20 02 8D18 8D 00 8D18 8D 2B 8D1B 8D 8B 8D2 8B 8D2 8B 01 D2 8D26 66 80 8D28 8D 8B 04 8D28 8C 01 DA 8D2E 8D 30	L8CE2 L8CE4 * CONSOLE XVEC4 L8DØ1 L8DØ2 * TRANSMI L8D14 L8D1B L8D1D	LDU GBU LEAX \$01 STX CBU LDX #CA LBDX #CA LBDX #CA LBDX #CA LBDA DEVI CMPA #-3 BNE LBD LEAS \$02 CUR CIN TST CIN BNE LBD LEAS \$02 CUR CIN TST CIN BNE LBD LEAS A STST CIN BNE LBD BNE L	FAD ,U FAD SBUF 7C 444 HOOK WUM ### ### ### ### ### ### ### ### ###	GET BLOCK NUMBER * INCREMENT BLOCK NUMBER * AND SAVE IT USE CASBUF AS DLOAD INPUT BUFFER READ 128 CHARACTERS (ONE BLOCK) INTO BUFFER RESET CONSOLE IN BUFFER GET DEVICE NUMBER DLOAD DEVICE NUMBER BRANCH IF NOT OLOAD PURGE 1ST RETURN ADDR FROM STACK RESET EMPTY/FULL FLAG ANY CHARACTERS LEFT IN BUFFER? YES, GO GET ONE SET EMPTY/FULL FLAG TO EOF SAVE REGISTERS GET CONSOLE IN CHARACTER BUFFER GET A CHARACTER SAVE IT ON THE STACK SAVE NEW CHARACTER BUFFER DECREMENT CHARACTER BUFFER RESTORE REGISTERS AND RETURN ENDER INCREMENT ATTEMPT COUNTER * GET FILE REQUEST CONTROL CODE * AND TRANSMIT IT BRANCH IF NO ECHO OR ERROR POINT TO CASS FILE NAME BUFFER GUT CHARACTER FROM NAME BUFFER OUTPUT IT TO RS 232 PORT COMPARE TO END OF BUFFER LOUP UNTIL DONE OUTPUT CHECK BYTE AND LOOK FOR ACKNOWLEDGE
1586 1587 1588 1589 1591 1592 1593 1594 1595 1596 1697 1600 1601 1602 1603 1604 1606 1607 1608 1609 1610 1611 1612 1613 1614 1615 1616 1617 1618 1617 1618 1621 1622 1623 1624 1623 1624 1626 1627 1628 1629	8CE4 3Ø 41 8CE6 9F 7E 8CE8 8E Ø1 DA 8CEB BD 8D 7C 8CEE 7E A6 44 8CF1 96 6F 8CF3 81 FD 8CF5 26 ØA 8CF7 32 62 8CF9 ØF 7Ø 8CFB ØD 79 8CFB ØD 79 8CFB ØD 79 8CFB ØD 79 80FB ØD 79 80B Ø	L8CE2 L8CE4 * CONSOLE XVEC4 L8D01 L8D02 * TRANSMI L8D14 L8D1B L8D1D	LDU GBU LEAX \$01 STX CBU LDX #CA STX CBU LDX #CA LBDX #CA LBDY LBDA LBDA LBDA LEAS \$02 CURR CIN LBDA LEAS \$02 CURR CIN LBDA LEAS \$02 CURR CIN LBDA LBDA LBDA LBDA LBDA LBDA LBDA LBDA	FAD ,U FAD SBUF 7C 444 HOOOK NUM Ø1 ,S BFL CTR Ø2 BFL ,X,B PTR PTR CTR 12 E2 E2 K,Y,Y,U,PC AME - READ FILE STATUS FROM S ,A 1D 48 A 68 1B NBUF+1 Ø4 NBUF+9 26 62 1B 72 21B	GET BLOCK NUMBER * INCREMENT BLOCK NUMBER * AND SAVE IT USE CASBUF AS DLOAD INPUT BUFFER READ 128 CHARACTERS (ONE BLOCK) INTO BUFFER RESET CONSOLE IN BUFFER GET DEVICE NUMBER DLOAD DEVICE NUMBER BRANCH IF NOT OLOAD PURGE 1ST RETURN ADDR FROM STACK RESET EMPTY/FULL FLAG ANY CHARACTERS LEFT IN BUFFER? YES, GO GET ONE SET EMPTY/FULL FLAG TO EOF SAVE REGISTERS GET CONSOLE IN CHARACTER BUFFER GET A CHARACTER SAVE IT ON THE STACK SAVE NEW CHARACTER BUFFER RECEMENT CHARACTER BUFFER RESTOR REGISTERS AND RETURN ENDER RESTOR REGISTERS AND RETURN ENDER RESET ATTEMPT COUNTER SAVE SPACE ON STACK FOR TEMP VARIABLES STACK TO Y (TFR S,Y) - SAVE VARIABLE POINTER INCREMENT ATTEMPT COUNTER * GET FILE REQUEST CONTROL CODE * AND TRANSMIT IT BRANCH IF NO ECHO OR ERROR POINT TO CASS FILE NAME BUFFER GET CHARACTER FROM NAME BUFFER GET CHARACTER FROM NAME BUFFER GOUTPUT IT TO RS 232 PORT COMPARE TO END OF BUFFER LOOP UNTIL DONE OUTPUT CHECK BYTE AND LOOK FOR ACKNOWLEDGE TRANSMIT NAME AGAIN IF NO ACKNOWLEDGE
1586 1587 1588 1589 1599 1591 1592 1593 1594 1595 1596 1600 1601 1602 1603 1604 1605 1606 1611 1612 1613 1614 1616 1617 1618 1619 1620 1621 1622 1623 1624 1625 1624 1625 1626 1627 1628 1629 1629 1629 1620 1621 1622 1623 1624 1625 1624 1625 1626 1627 1628 1629 1629 1629 1629 1629 1629 1629 1629	8CE4 3Ø 41 8CE6 9F 7E 8CE8 8E Ø1 DA 8CEB BD 8D 7C 8CEE 7E A6 44 8CF1 96 6F 8CF3 81 FD 8CF5 26 ØA 8CF7 32 62 8CF9 ØF 7Ø 8CFB 0D 79 8CFD 26 Ø3 8CFF 03 7Ø 80Ø1 39 8CFD 03 44 74 8D04 9E 7A 8D06 A6 8Ø 8D08 34 02 8D08 34 02 8D08 34 02 8D10 8D 00 8D12 35 F6 8D14 4F 8D15 34 16 8D17 31 E4 8D19 2Ø 02 8D18 8D 00 8D12 35 F6 8D14 4F 8D15 34 16 8D17 31 E4 8D19 2Ø 02 8D18 8D 08 8D18 8D 08 8D18 8D 8B 8D2 8B 8D2 8B 8D3 8D 8D3 8B 8D	L8D12 * TRANSMIL8D14 L8D1B L8D1B L8D1D	LDU CBU LEAX \$01 STX CBU LDX #CA JSR L8D JMP LA6. : IN RAM H LDA DEVI CMPA #-3 BNE L8D LEAS \$02 CLR CIN TST CIN BNE L8D LEAS \$02 CLR CIN TST CIN BNE L8D LEAS \$02 CLR CIN TST CIN BNE L8D COM CIN RTS PSHS U, Y LDA ,X FSHS A STX CIN DEC CIN BNE L8D BNE L8D BNE L8D BNE L8D BNE L8D BNE L8D LEAY ,S BNE L8D LDA #\$8. BSR L8D LLDA #\$8. BSR L8D LLDA #CFF LDA ,X FSF BNE L8D	FAD ,U FAD SBUF 76 74 44 1000K NUM Ø1 ,S BFL CTR Ø2 BFL CTR Ø2 FFL CTR 02 FFL A DEPTR PTR PTR PTR CTR 12 E2 ,X,Y,U,PC MME - READ FILE STATUS FROM S ,A 1D 48 A 58 1B NBUF+1 89 44 NBUF+9 26 62 1B 72 1B 72 1B	GET BLOCK NUMBER * INCREMENT BLOCK NUMBER * AND SAVE IT USE CASBUF AS DLOAD INPUT BUFFER READ 128 CHARACTERS (ONE BLOCK) INTO BUFFER RESET CONSOLE IN BUFFER GET DEVICE NUMBER BRANCH IF NOT OLOAD PURGE 1ST RETURN ADDR FROM STACK RESET EMPTY/FULL FLAG ANY CHARACTERS LEFT IN BUFFER? YES, GO GET ONE SET EMPTY/FULL FLAG TO EOF SAVE REGISTERS GET CONSOLE IN CHARACTER BUFFER GET A CHARACTER SUFFER DECREMENT CHARACTER BUFFER SAVE NEW CHARACTER BUFFER DECREMENT ATTEMPT COUNTER SAVE SPACE ON STACK FOR TEMP VARIABLES STACK TO Y (TFR S,Y) - SAVE VARIABLE POINTER INCREMENT ATTEMPT COUNTER * GET FILE ROQUEST CONTROL CODE * AND TRANSMIT IT BRANCH IF NO ECHO OR ERROR POINT TO CASS FILE NAME BUFFER GET CHARACTER FROM NAME BUFFER GUT CHARACTER FROM SAVE FILE TYPE \$FF = NOT FOUND BRANCH IF ERROR SAVE FILE TYPE \$FF = NOT FOUND BRANCH IF ERROR
1586 1587 1588 1589 1591 1592 1593 1594 1595 1596 1697 1600 1601 1602 1603 1604 1606 1607 1608 1609 1610 1611 1612 1613 1614 1615 1616 1617 1618 1617 1618 1621 1622 1623 1624 1623 1624 1626 1627 1628 1629	8CE4 3Ø 41 8CE6 9F 7E 8CE8 8E Ø1 DA 8CEB BD 8D 7C 8CEE 7E A6 44 8CF1 96 6F 8CF3 81 FD 8CF5 26 ØA 8CF7 32 62 8CF9 ØF 7Ø 8CFB 0D 79 8CFD 26 Ø3 8CFF 03 7Ø 80Ø1 39 8CFD 03 44 74 8D04 9E 7A 8D06 A6 8Ø 8D08 34 02 8D08 34 02 8D08 34 02 8D10 8D 00 8D12 35 F6 8D14 4F 8D15 34 16 8D17 31 E4 8D19 2Ø 02 8D18 8D 00 8D12 35 F6 8D14 4F 8D15 34 16 8D17 31 E4 8D19 2Ø 02 8D18 8D 08 8D18 8D 08 8D18 8D 8B 8D2 8B 8D2 8B 8D3 8D 8D3 8B 8D	L8D12 * TRANSMI L8D14 L8D1B L8D1D L8D26	LDU GBU LEAX \$01 STX CBU LDX #CA STX CBU LDX #CA LBDX #CA LBDY LBDA LBDA LBDA LEAS \$02 CURR CIN LBDA LEAS \$02 CURR CIN LBDA LEAS \$02 CURR CIN LBDA LBDA LBDA LBDA LBDA LBDA LBDA LBDA	FAD ,U FAD SBUF 7C 444 HOOK WUM ### ### ### ### ### ### ### ### ###	GET BLOCK NUMBER * INCREMENT BLOCK NUMBER * AND SAVE IT USE CASBUF AS DLOAD INPUT BUFFER READ 128 CHARACTERS (ONE BLOCK) INTO BUFFER RESET CONSOLE IN BUFFER GET DEVICE NUMBER DLOAD DEVICE NUMBER BRANCH IF NOT OLOAD PURGE 1ST RETURN ADDR FROM STACK RESET EMPTY/FULL FLAG ANY CHARACTERS LEFT IN BUFFER? YES, GO GET ONE SET EMPTY/FULL FLAG TO EOF SAVE REGISTERS GET CONSOLE IN CHARACTER BUFFER GET A CHARACTER SAVE IT ON THE STACK SAVE NEW CHARACTER BUFFER RECEMENT CHARACTER BUFFER RESTOR REGISTERS AND RETURN ENDER RESTOR REGISTERS AND RETURN ENDER RESET ATTEMPT COUNTER SAVE SPACE ON STACK FOR TEMP VARIABLES STACK TO Y (TFR S,Y) - SAVE VARIABLE POINTER INCREMENT ATTEMPT COUNTER * GET FILE REQUEST CONTROL CODE * AND TRANSMIT IT BRANCH IF NO ECHO OR ERROR POINT TO CASS FILE NAME BUFFER GET CHARACTER FROM NAME BUFFER GET CHARACTER FROM NAME BUFFER GOUTPUT IT TO RS 232 PORT COMPARE TO END OF BUFFER LOOP UNTIL DONE OUTPUT CHECK BYTE AND LOOK FOR ACKNOWLEDGE TRANSMIT NAME AGAIN IF NO ACKNOWLEDGE

1633	8D3E A7 23			\$Ø3,Y	SAVE ASCII FLAG
1634 1635	8D4Ø 8D 29 8D42 26 D7			L8D6B L8D1B	READ CHECK BYTE FROM SENDER BRANCH IF NO CHECKBYTE MATCH
1636	8D44 32 62			\$Ø2,S	PURGE ATTEMPT COUNTER & CHECK BYTE FROM STACK
1637	8D46 35 86	+ THORE		A,B,PC	RETURN FILE STATUS IN ACCD
1638 1639	8D48 6C A4	L8D48	INC	TTEMPT COUNTER - AFTER 5 TRIES, G ,Y	INCREMENT ATTEMPT COUNTER
1640	8D4A A6 A4		LDA	,Υ	GET ATTEMPT COUNTER
1641 1642	8D4C 81 Ø5 8D4E 25 1A			#\$Ø5 L8D6A	IS THIS THE FIFTH TRY?
1643	8D5Ø 86 BC			#\$BC	YES ; TIME TO QUIT-GET ABORT CODE
1644	8D52 BD 8E ØC		JSR	L8EØC	OUTPUT ABORT CODE OVER THE RS 232 PORT
1645 1646	8D55 7E A6 19	* ECHU (- OUTPUT A CHARACTER, READ A CHAR	IO ERROR
1647				TO THE OUTPUT CHARACTER. Z=Ø IF N	
1648	8D58 34 Ø2	L8D58	PSHS		SAVE COMPARE CHARACTER ON STACK
1649 1650	8D5A 8D 5C 8D5C 26 Ø2			L8DB8 L8D60	SEND A CHARACTER OUT BRANCH IF READ ERROR
1651	8D5E A1 E4		CMPA		COMPARE RECEIVED CHARACTER TO TRANSMITTED CHARACTER
1652	8D6Ø 35 82	L8D6Ø	PULS	A,PC	RESTORE COMPARE CHARACTER AND RETURN
1653 1654				R CHECKBYTE AND READ ACKNOWLEGE (FLAG SET IF NO ERROR AND ACKNOWL	
1655	8D62 A6 21	L8D62		\$01,Y	GET XOR CHECKBYTE
1656	8D64 8D 52		BSR	L8DB8	OUTPUT XOR CHECKBYTE AND READ ONE BYTE
1657	8D66 26 Ø2			L8D6A	BRANCH IF READ ERROR
1658 1659	8D68 81 C8 8D6A 39	L8D6A	RTS	#\$C8	COMPARE INPUT BYTE TO ACKNOWLEDGE CODE
1660				ECKBYTE THEN LOAD ACCUMULATED XOR	R CHECKBYTE.
1661 1662	oned on ar	* SET ZE L8D6B		AG IF ACCUMULATED CHECK BYTE = Ø L8D72	INPUT A CHARACTER FROM RS 232
1663	8D6B 8D Ø5 8D6D 26 FB	LODUB		L8D6A	BRANCH IF TIMEOUT
1664	8D6F A6 21			\$Ø1,Y	GET CHECK BYTE
1665	8D71 39	10070	RTS	LODDC	TARRET A CHARACTER FROM DC 000
1666 1667	8D72 8D 48 8D74 34 Ø3	L8D72		L8DBC A,CC	INPUT A CHARACTER FROM RS 232 SAVE CHARACTER AND ZERO FLAG ON STACK
1668	8D76 A8 21			\$Ø1,Y	* EXCLUSIVE OR INPUT
1669	8D78 A7 21			\$Ø1,Y	* CHARACTER WITH CHECK BYTE
167Ø 1671	8D7A 35 83	* REULIES		CC,A,PC LOCK FROM RS 232 INPUT -	RESTORE CHARACTER AND ZERO FLAG
1672				CEIVED DATA INTO THE BUFFER POINT	TED TO BY X
1673		* U REGE	ISTER	CONTAINS THE BLOCK NUMBER; RETUR	RN Z=1 IF NO
1674 1675	8D7C 4F	* ERRORS	CLRA	RACTER COUNT IN ACCA; ACCA = Ø IF	· FILE EMPTY RESET ATTEMPT COUNTER
1676	8D7D 34 76	LODYC		U,Y,X,B,A	SAVE SPACE FOR STACK BUFFER
1677	8D7F 68 67		ASL	\$07,S	* 6,7 S (U REG) CONTAIN THE 14 BIT BLOCK NUMBER -
1678 1679	8D81 69 66			\$06,S	* PUT THE BOTTOM 7 BITS IN 7,S AND THE
1680	8D83 64 67 8D85 31 E4		LEAY	\$07,S .S	* TOP SEVEN BITS IN 6,S STACK POINTER TO Y (TFR S,Y)
1681	8D87 20 02		BRA	L8D8B	
1682	8D89 8D BD	L8D89		L8D48	INCREMENT ATTEMPT COUNTER
1683 1684	8D8B 86 97 8D8D 8D C9	L8D8B	LDA BSR	#\$97 L8D58	* TRANSMIT A BLOCK REQUEST CODE, ECHO * CHECK AND RESET CHECK BYTE
1685	8D8F 26 F8			L8D89	BRANCH IF NO MATCH OR ERROR
1686	8D91 A6 26			\$Ø6,Y	* SEND OUT HIGH ORDER SEVEN BITS
1687 1688	8D93 8D 6F 8D95 A6 27			L8EØ4 \$Ø7,Y	* OF BLOCK NUMBER = SEND OUT LOW ORDER SEVEN BITS
1689	8D97 8D 6B			L8EØ4	= OF BLOCK NUMBER
1690	8D99 8D C7			L8D62	TRANSMIT CHECK BYTE AND GET ACKNOWLEDGE
1691 1692	8D9B 26 EC 8D9D 8D D3			L8D89 L8D72	BRANCH IF ERROR OR NO ACKNOWLEDGE READ CHARACTER COUNT
1693	8D9F 26 E8			L8D89	BRANCH IF READ ERROR
1694	8DA1 A7 24		STA	\$Ø4,Y	SAVE CHARACTER COUNT IN STACK VARIABLES
1695 1696	8DA3 AE 22	* RFAD I		\$02,Y LOCK OF 128 CHARACTERS - THE HOST	GET VARIABLES POINTER FROM STACK BUFFER
1697				REGARDLESS OF HOW MANY ARE VALID.	
1698	8DA5 C6 8Ø	10047		#128	128 CHARACTERS/BUFFER
1699 1700	8DA7 8D C9 8DA9 26 DE	L8DA7		L8D72 L8D89	READ A CHARACTER RESTART PROCESS IF READ ERROR
1701	8DAB A7 8Ø		STA	, χ+	SAVE THE CHARACTER IN BUFFER
1702	8DAD 5A		DECB		DECREMENT CHARACTER COUNTER
17Ø3 17Ø4	8DAE 26 F7 8DBØ 8D B9			L8DA7 L8D6B	BRANCH IF NOT DONE INPUT XOR CHECKBYTE
1705	8DB2 26 D5			L8D89	RESTART PROCESS IF READ ERROR OR BAD CHECKBYTE
1706	8DB4 32 64			\$Ø4,S	PURGE ATTEMPT COUNTER, CHECK BYTE AND LOAD ADDRESS FROM STACK
1707 1708	8DB6 35 96 8DB8 6F 21	L8DB8		A,B,X,PC \$Ø1,Y	RETURN CHARACTER COUNT IN ACCA CLEAR CHECK BYTE
1709	8DBA 8D 5Ø	LODDO		L8EØC	OUTPUT A CHARACTER OVER RS 232 PORT
1710				ACTER FROM THE RS 232 INPUT PORT.	
1711 1712				ACTER IN ACCA. EXIT WITH Z=0 ERROR, Z = 1 FOR VALID BYTE INPU	IT
1713	8DBC 4F	L8DBC	CLRA	,	CLEAR ATTEMPT COUNTER
1714	8DBD 34 15			X,B,CC	SAVE REGISTERS AND INTERRUPT STATUS
1715 1716	8DBF 1A 5Ø 8DC1 96 E7			#\$50 TIMOUT	DISABLE INTERRUPTS GET TIMEOUT VARIABLE DELAY
1716	8DC3 9E 8A			ZERO	X=0: TIMEOUT CONSTANT DELAY
1718	8DC5 8D 1F	L8DC5	BSR	L8DE6	GO GET RS 232 STATUS
1719	8DC7 24 FC	10000		L8DC5	LOOP IF SPACING
1720 1721	8DC9 8D 1B 8DCB 25 FC	L8DC9		L8DE6 L8DC9	GET RS 232 STATUS LOOP IF MARKING
1722	8DCD 8D 2A		BSR	L8DF9	DELAY 1/2 BIT TIME
	8DCF C6 Ø1			#\$Ø1	* GET BIT SHIFT COUNTER AND BIT * MASK AND SAVE IT ON STACK
1724 1725	8DD1 34 Ø4 8DD3 4F		PSHS CLRA		* MASK AND SAVE IT ON STACK RESET DATA BYTE
1726	8DD4 8D 21	L8DD4	BSR	L8DF7	GO DELAY ONE BIT TIME
1727 1728	8DD6 F6 FF 22 8DD9 56		LDB RORB	PIA1+2	* RS 232 INPUT TO * CARRY FLAG
1/20	5005 50		NUKB		VANICE I LAW

1729	8DDA 24 Ø2		BCC	L8DDE	BRANCH IF RS 232 INPUT = Ø (SPACING)
1730	8DDC AA E4		ORA	,\$	IF MARKING, OR A 1 BIT INTO DATA BYTE
1731	8DDE 68 E4	L8DDE		,S	SHIFT BIT COUNTER ONE BIT TO LEFT
1732	8DEØ 24 F2			L8DD4	CARRY WILL BE SET AFTER 8 SHIFTS
1733	8DE2 32 61				PULL BIT COUNTER OFF THE STACK
1734	8DE4 35 95		PULS	CC,B,X,PC	RESTORE INTERRUPT STATUS & RETURN
1735		+ DUT DO		CTATUS INTO THE CARRY FLAG AND OH	FOU FOR TIMEOUT
1736	ODE6 E6 EE 22			STATUS INTO THE CARRY FLAG AND CH	
1737 1738	8DE6 F6 FF 22 8DE9 56	L8DE6	RORB	PIA1+2	* RS 232 INPUT TO * CARRY FLAG
1739	8DEA 30 01			\$Ø1,X	= INCREMENT CONSTANT TIMEOUT
1740	8DEC 26 Ø8				= DELAY, RETURN IF ⇔ Ø
1741	8DEE 4A		DECA		* DECREMENT VARIABLE TIMEOUT
1742	8DEF 26 Ø5			L8DF6	* DELAY: RETURN IF ⇔ Ø
1743		* DLOAD		IMED OUT HERE	
1744	8DF1 32 62			\$Ø2,S	PURGE RETURN ADDRESS OFF STACK
1745 1746	8DF3 35 15 8DF5 4C			CC,B,X	CLEAN UP STACK/RESTORE INTERRUPTS SET ACCA = 1; ZERO FLAG = Ø
1747	8DF6 39	L8DF6	INCA RTS		SET ACCA - 1; ZERO FLAG - W
1748	0510 33			COUNT DOWN DLBAUD	
1749	8DF7 8D ØØ	L8DF7		L8DF9	CALL DELAY ROUTINE
1750	8DF9 34 Ø2	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	8EØØ 26 FB			L8DFD	NOT DONE
1755 1756	8EØ2 35 82	*	PULS	A,PC	RESTORE ACCA AND RETURN
1757	8EØ4 34 Ø2	L8EØ4	PSHS	A	SAVE CHARACTER ON STACK
1758	8EØ6 A8 21			\$Ø1,Y	* EOR CHARACTER WITH 1,Y AND
1759	8EØ8 A7 21			\$Ø1,Y	* SAVE RESULT IN 1,Y
1760	8EØA 35 Ø2		PULS	A	GET CHARACTER BACK
1761					
1762	0500 24 07			N ACCA OUT OVER RS232 OUTPUT	CAVE ACCD AND INTERDURT CTATUS
1763 1764	8EØC 34 Ø7 8EØE 1A 5Ø	L8EØC		B,A,CC #\$50	SAVE ACCD AND INTERRUPT STATUS DISABLE INTERRUPTS
1764	8E1Ø 8D E5			L8DF7	DELAY AWHILE
1766	8E12 8D E3			L8DF7	DELAY SOME MORE
1767	8E14 7F FF 2Ø			PIA1	SET R5232 OUTPUT TO SPACING
	8E17 8D DE			L8DF7	DELAY SOME MORE - START BIT
1769	8E19 C6 Ø1		LDB	#\$Ø1	BIT CTR - SEND 8 BITS
1770	8E1B 34 Ø4		PSHS		SAVE BIT CTR ON STACK
1771	8E1D A6 62	L8E1D		\$02,S	GET OUTPUT BYTE
1772	8E1F A4 E4		ANDA		AND IT W/THE BIT CTR
1773 1774	8E21 27 Ø2 8E23 86 Ø2			L8E25 #\$Ø2	THIS BIT IN OUTPUT BYTE = Ø OUTPUT BIT = 1; SET R5232 TO MARKING
1775	8E25 B7 FF 2Ø	L8E25	STA	PIA1	BET R5232 TO VALUE IN ACCA
1776	8E28 8D CD	LOLLS	BSR	L8DF7	DELAY FOR AWHILE
1777	8E2A 68 E4		ASL	,S	SHIFT BIT CTR
1778	8E2C 24 EF			L8E1D	WHEN CARRY SET, 8 BITS DONE
1779	8E2E 86 Ø2		LDA	#\$02	WHEN DONE, SET R5232 TO MARKING
1780	8E3Ø B7 FF 2Ø		STA	PIA1	SET R5232 OUTPUT
1781	8E33 32 61			\$01,S	PULL BIT CTR OFF THE STACK
1782	8E35 35 87		PULS	CC,A,B,PC	RESTORE ACCD, INTERRUPTS & RETURN
1783 1784		* DDOCES	C EVC	LAMATION POINT	
	8E37 86 Ø1		LDA		* SET SPACES
1786	8E39 97 D9	LOLO	STA		* COUNTER = 1
1787		* PROCES		ING 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 ⇔Ø
1790	8E3F 9D A5			GETCCH	GET CURRENT INPUT CHARACTER
1791	8E41 10 27 00 93			L8ED8	EXIT PRINT USING IF END OF LINE
1792 1793	8E45 D7 D3 8E47 BD B1 56		STB JSR	VD3 LB156	SAVE REMAINDER FORMAT STRING LENGTH EVALUATE EXPRESSION
1793	8E4A BD B1 46			LB146	TM ERROR IF NUMERIC VARIABLE
1795	8E4D 9E 52			FPAØ+2	* GET ITEM - LIST DESCRIPTOR ADDRESS
1796	8E4F 9F 4D		STX		* AND SAVE IT IN V4D
1797	8E51 D6 D9			VD9	GET SPACES COUNTER
1798	8E53 BD B6 AD			LB6AD	PUT ACCB BYTES INTO STRING SPACE & PUT DESCRIPTOR ON STRING STACK
1799	8E56 BD B9 9F	+ 0.0 50		LB99F	PRINT THE FORMATTED STRING TO CONSOLE OUT
1800	SEEO OE ES	* PAD F0		STRING WITH SPACES IF ITEM - LIST	
18Ø1 18Ø2	8E59 9E 52 8E5B D6 D9		LDX	FPAØ+2	POINT X TO FORMATTED STRING DESCRIPTOR ADDRESS GET SPACES COUNTER
1803	8E5D EØ 84		SUBB		SUBTRACT LENGTH OF FORMATTED STRING
1804	8E5F 5A	L8E5F	DECB	• · · ·	DECREMENT DIFFERENCE
1805	8E6Ø 1Ø 2B Ø1 4F			L8FB3	GO INTERPRET ANOTHER ITEM - LIST
1806	8E64 BD B9 AC			LB9AC	PAD FORMAT STRING WITH A SPACE
1807	8E67 2Ø F6			L8E5F	KEEP PADDING
1808	0500 87 80			N - PROCESS A %SPACES% COMMAND	+ CAVE THE CURRENT FORMAT CIRING
1809 1810	8E69 D7 D3 8E6B 9F ØF	L8E69	STB	VD3 TEMPTR	* SAVE THE CURRENT FORMAT STRING * COUNTER AND POINTER
	8E6D 86 Ø2			#\$Ø2	* COUNTER AND POINTER INITIAL SPACES COUNTER = 2
	8E6F 97 D9			VD9	SAVE IN SPACES COUNTER - 2
1813	8E71 A6 84	L8E71		,х	GET A CHARACTER FROM FORMAT STRING
1814	8E73 81 25		CMPA		COMPARE TO TERMINATOR CHARACTER
	8E75 27 C4			L8E3B	BRANCH IF END OF SPACES COMMAND
1816	8E77 81 20			#' '	BLANK
1817	8E79 26 Ø7			L8E82	BRANCH IF ILLEGAL CHARACTER
1818 1819	8E7B ØC D9 8E7D 3Ø Ø1		INC	VD9 \$Ø1,X	ADD ONE TO SPACES COUNTER MOVE FORMAT POINTER UP ONE
1819	8E7F 5A		DECB	₩D1,A	DECREMENT LENGTH COUNTER
1821	8E8Ø 26 EF			L8E71	BRANCH IF NOT END OF FORMAT STRING
1822	8E82 9E ØF	L8E82	LDX	TEMPTR	* RESTORE CURRENT FORMAT STRING COUNTER
1823	8E84 D6 D3		LDB		* AND POINTER TO POSITION BEFORE SPACES COMMAND
1824	8E86 86 25		LDA	#'%'	SEND A % TO CONSOLE OUT AS A DEBUGGING AID

1825 1826	8E88 BD 8F D8	* ERROR PROCESSOR - ILLEGAL CHARACTER OR BA	SEND A +' TO CONSOLE OUT IF VDA ⇔ Ø
1827 1828 1829	8E8B BD A2 82 8E8E 20 29	JSR LA282 BRA L8EB9	SEND CHARACTER TO CONSOLE OUT GET NEXT CHARACTER IN FORMAT STRING
1830 1831 1832 1833	8E9Ø 81 CD 8E92 27 Ø1 8E94 39	* PRINT RAM HOOK XVEC9 CMPA #\$CD BEQ L8E95 RTS	USING TOKEN BRANCH IF PRINT USING
1834 1835		* PRINT USING	
1836 1837		* VDA IS USED AS A STATUS BYTE: BIT 6 = CON * BIT 5=LEADING ASTERISK FORCE; BIT 4 = FLO	
1838 1839	8E95 32 62	* BIT 3 = PRE SIGN FORCE; BIT 2 = POST SIGN L8E95 LEAS \$02,S	I FORCE; BIT Ø = EXPONENTIAL FORCE PURGE RETURN ADDRESS OFF THE STACK
184Ø	8E97 BD B1 58	JSR LB158	EVALUATE FORMAT STRING TM ERROR IF VARIABLE TYPE = NUMERIC
1841	8E9A BD B1 46	JSR LB146	
1842	8E9D C6 3B	LDB #';'	CHECK FOR ITEM LIST SEPARATOR
1843	8E9F BD B2 6F	JSR LB26F	SYNTAX CHECK FOR ; * GET FORMAT STRING DESCRIPTOR ADDRESS
1844	8EA2 9E 52	LDX FPAØ+2	
1845	8EA4 9F D5	STX VD5	* AND SAVE IT IN VD5
1846	8EA6 20 06	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
185Ø	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 FC ERROR IF FORMAT STRING = NULL
1853	8EB4 7E B4 4A	L8EB4 JMP LB44A	
1854	8EB7 AE Ø2	L8EB7 LDX \$02,X	POINT X TO START OF FORMAT STRING
1855 1856	8EB9 ØF DA	* INTERPRET THE FORMAT STRING 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 10 27 FF 72	LBEO 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 POINT X TO FORMAT STRING DESCRIPTOR
1872	8EDD 9E D5	L8EDD LDX VD5	
1873	8EDF 7E B6 59	JMP LB659	RETURN ADDRESS AND LENGTH OF FORMAT STRING - EXIT PRINT USING CHECK FOR + (PRE-SIGN FORCE)
1874	8EE2 81 2B	L8EE2 CMPA #'+'	
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 20 CC	BRA L8EBB	INTERPRET THE REST OF THE FORMAT STRING
188Ø	8EEF 81 2E	L8EEF CMPA #'.'	DECIMAL POINT?
1881	8EF1 27 4E	BEQ L8F41	YES
1882	8EF3 81 25	CMPA #'%'	PERCENT SIGN?
1883	8EF5 10 27 FF 70	LBEQ L8E69	YES
1884	8EF9 A1 84	CMPA ,X	COMPARE THE PRESENT FORMAT STRING INPUT
1885	8EFB 26 8B	*	CHARACTER TO THE NEXT ONE IN THE STRING
1886		L8EFB BNE L8E88	NO MATCH - ILLEGAL CHARACTER
1887 1888	8EFD 81 24	* TWO CONSECUTIVE EQUAL CHARACTERS IN FORMA	
1889	8EFF 27 19	BEQ L8F1A	YES - MAKE THE DOLLAR SIGN FLOAT
1890	8FØ1 81 2A	CMPA #'*'	ASTERISK?
1891	8FØ3 26 F6	BNE L8EFB	NO – ILLEGAL CHARACTER
1892	8FØ5 96 DA	LDA VDA	* GRAB THE STATUS BYTE AND BET BIT 5 * TO INDICATE THAT THE OUTPUT WILL
1893	8FØ7 8A 2Ø	ORA #\$20	
1894	8FØ9 97 DA	STA VDA	* BE LEFT PADDED WITH ASTERISKS
1895	8FØB C1 Ø2	CMPB #2	= CHECK TO SEE IF THE \$\$ ARE THE LAST TWO
1896	8FØD 25 11	BLO L8F2Ø	= CHARACTERS IN THE FORMAT STRING AND BRANCH IF SO
1897	8FØF A6 Ø1	LDA \$01,X	GET THE NEXT CHARACTER AFTER ** CHECK FOR **\$
1898	8F11 81 24	CMPA #'\$'	
1899	8F13 26 ØB	BNE L8F2Ø	CHECK FOR MORE CHARACTERS DECREMENT STRING LENGTH COUNTER
1900	8F15 5A	DECB	
1901	8F16 3Ø Ø1	LEAX \$01,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 #\$10	* BIT 4 TO INDICATE A
1906	8F1E 97 DA	STA VDA	* FLOATING DOLLAR SIGN
1907	8F2Ø 3Ø Ø1	L8F2Ø LEAX \$01,X	MOVE FORMAT STRING POINTER UP ONE
1908 1909	8F22 ØC D9	INC VD9 * PROCESS CHARACTERS TO THE LEFT OF THE DEC	ADD ONE TO LEFT DIGIT (FLOATING \$ OR ASTERISK PAD)
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 4Ø		ORA	#\$40	* AND SET BIT 6 WHICH IS THE
	8F3D 97 DA			VDA	* COMMA SEPARATOR FLAG
	8F3F 2Ø E5			L8F26	PROCESS MORE CHARACTERS TO LEFT OF DECIMAL POINT
1925 1926	0541 46 04			IMAL POINT IF NO DIGITS TO LEFT (GET NEXT FORMAT CHARACTER
1926	8F41 A6 84 8F43 81 23	L8F41	CMPA		IS IT A NUMBER SIGN?
	8F45 10 26 FF 3F			L8E88	NO
	8F49 86 Ø1		LDA		* SET THE RIGHT DIGIT COUNTER TO 1 -
	8F4B 97 D8		STA		* ALLOW ONE SPOT FOR DECIMAL POINT
	8F4D 3Ø Ø1			\$Ø1,X	MOVE FORMAT POINTER UP ONE
1932		* PROCES	SS DIG	ITS TO RIGHT OF DECIMAL POINT	
	8F4F ØC D8	L8F4F	INC	VD8	ADD ONE TO RIGHT DIGIT COUNTER
	8F51 5A		DECB		DECREMENT FORMAT LENGTH COUNTER
	8F52 27 2Ø			L8F74	BRANCH IF END OF FORMAT STRING
	8F54 A6 8Ø		LDA		GET A CHARACTER FROM FORMAT STRING
	8F56 81 23 8F58 27 F5			#'#' L8F4F	IS IT NUMBER SIGN? YES - KEEP CHECKING
1939	0F30 27 F3	* CHECK		XPONENTIAL FORCE	1E3 - KEEF CHECKING
	8F5A 81 5E	L8F5A			CHECK FOR UP ARROW
	8F5C 26 16	201 071		L8F74	NO UP ARROW
	8F5E A1 84			, х	IS THE NEXT CHARACTER AN UP ARROW?
	8F6Ø 26 12			Ĺ8F74	NO
1944	8F62 A1 Ø1			\$Ø1,X	AND THE NEXT CHARACTER?
1945	8F64 26 ØE		BNE	L8F74	NO NO
	8F66 A1 Ø2			\$Ø2,X	HOW ABOUT THE 4TH CHARACTER?
1947	8F68 26 ØA			L8F74	NO, ALSO
	8F6A C1 Ø4			#4	* CHECK TO SEE IF THE 4 UP ARROWS ARE IN THE
	8F6C 25 Ø6 8F6E CØ Ø4			L8F74 #4	* FORMAT STRING AND BRANCH IF NOT = MOVE POINTER UP 4 AND SUBTRACT
1950	8F7Ø 3Ø Ø4			#4 \$Ø4,X	= MOVE POINTER OF 4 AND SUBTRACT = FOUR FROM LENGTH
	8F72 ØC DA		INC		INCREMENT STATUS BYTE - EXPONENTIAL FORM
1953					
1954		* CHECK	FOR A	PRE OR POST - SIGN FORCE AT END	OF FORMAT STRING
	8F74 3Ø 1F	L8F74			MOVE POINTER BACK ONE
1956	8F76 ØC D9		INC	VD9	ADD ONE TO LEFT DIGIT COUNTER FOR PRE-SIGN FORCE
1957	8F78 96 DA		LDA	VDA	* PRE-SIGN
	8F7A 85 Ø8			#\$Ø8	* FORCE AND
	8F7C 26 18			L8F96	* BRANCH IF SET
	8F7E ØA D9			VD9	DECREMENT LEFT DIGIT NO PRE-SIGN FORCE
	8F8Ø 5D		TSTB		* CHECK LENGTH COUNTER AND BRANCH * IF END OF FORMAT STRING
	8F81 27 13 8F83 A6 84		LDA	L8F96	GET NEXT FORMAT STRING CHARACTER
	8F85 8Ø 2D		SIIRA	,^ #'-'	CHECK FOR MINUS SIGN
	8F87 27 Ø6			L8F8F	BRANCH IF MINUS SIGN
	8F89 81 FE			#('+')-('-')	CHECK FOR PLUS SIGN
	8F8B 26 Ø9			L8F96	BRANCH IF NO PLUS SIGN
	8F8D 86 Ø8			#\$Ø8	GET THE PRE-SIGN FORCE FLAG
1969	8F8F 8A Ø4	L8F8F	ORA	#\$Ø4	OR IN POST-SIGN FORCE FLAG
	8F91 9A DA		ORA		OR IN THE STATUS BYTE
	8F93 97 DA		STA	VDA	SAVE THE STATUS BYTE
	8F95 5A		DECB		DECREMENT FORMAT STRING LENGTH
1973 1974		* EVALUA	ATE NII	MERIC ITEM-LIST	
1974	8F96 9D A5	L8F96			GET CURRENT CHARACTER
1976	8F98 1Ø 27 FF 3C	LO1 30		L8ED8	BRANCH IF END OF LINE
1977	8F9C D7 D3			VD3	SAVE FORMAT STRING LENGTH WHEN FORMAT EVALUATION ENDED
1978	8F9E BD B1 41			LB141	EVALUATE EXPRESSION
1979	8FA1 96 D9		LDA	VD9	GET THE LEFT DIGIT COUNTER
1980	8FA3 9B D8		ADDA		ADD IT TO THE RIGHT DIGIT COUNTER
1981	8FA5 81 11		CMPA		*
1982	8FA/ 10 22 24 9F			LB44A	* FC ERROR IF MORE THAN 16 DIGITS AND DECIMAL POINT
	8FAB BD 8F E5			L8FE5	CONVERT ITEM-LIST TO FORMATTED ASCII STRING
1984 1985	8FAE 3Ø 1F 8FBØ BD B9 9C			\$-01,X STRINOUT	MOVE BUFFER POINTER BACK ONE DISPLAY THE FORMATTED STRING TO CONSOLE OUT
1985	8FB3 ØF D7	L8FB3	CLR		RESET NEXT PRINT ITEM FLAG
1987	8FB5 9D A5	201 00		GETCCH	GET CURRENT INPUT CHARACTER
1988	8FB7 27 ØD			L8FC6	BRANCH IF END OF LINE
1989	8FB9 97 D7		STA	VD7	SAVE CURRENT CHARACTER (◇Ø) IN NEXT PRINT ITEM FLAG
	8FBB 81 3B			#';'	* CHECK FOR ; - ITEM-LIST SEPARATOR AND
1991	8FBD 27 Ø5			L8FC4	* BRANCH IF SEMICOLON
	8FBF BD B2 6D			SYNCOMMA	SYNTAX CHECK FOR COMMA
1993 1994	8FC2 20 02 8FC4 9D 9F	L8FC4		L8FC6 GETNCH	PROCESS NEXT PRINT ITEM GET NEXT INPUT CHARACTER
	8FC4 9D 9F 8FC6 9E D5	L8FC4 L8FC6	LDX		GET FORMAT STRING DESCRIPTOR ADDRESS
1995	8FC8 E6 84	201.00	LDX		GET LENGTH OF FORMAT STRING
1997	8FCA DØ D3		SUBB		SUBTRACT AMOUNT OF FORMAT STRING LEFT AFTER LAST PRINT ITEM
	8FCC AE Ø2			\$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		=GET AMOUNT OF UNUSED FORMAT STRING
2001	8FD1 10 26 FE E4			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		+ 00107		TO CONCOLE OUT TO THE CTATUS SYS	TF ~ A
2005 2006	8FD8 34 Ø2	* PRINI L8FD8	A + PSHS	TO CONSOLE OUT IF THE STATUS BYT	IE ◇ Ø RESTORE ACCA AND RETURN
2006 2007	8FDA 86 2B	LOFUÖ		#'+'	GET ASCII PLUS SIGN
2007	8FDC ØD DA		TST		* CHECK THE STATUS BYTE AND
2009	8FDE 27 Ø3			L8FE3	* RETURN IF = Ø
2010	8FEØ BD A2 82			LA282	SEND A CHARACTER TO CONSOLE OUT
2011	8FE3 35 82	L8FE3		A,PC	RETURN ACCA AND RETURN
2012					
2013					
				M-LIST TO DECIMAL ASCII STRING	
2014	8FE5 CE Ø3 DB	* CONVER	LDU	#STRBUF+4	POINT U TO STRING BUFFER
2014 2015	8FE8 C6 20		LDU LDB	#STRBUF+4 #SPACE	BLANK
2014			LDU	#STRBUF+4 #SPACE	

2017					
	8FEC 85 Ø8		BITA	#\$Ø8	* CHECK FOR A PRE-SIGN FORCE
2018	8FEE 27 Ø2			L8FF2	* BRANCH IF NO PRE-SIGN FORCE
2019	8FFØ C6 2B		LDB	#'+'	PLUS SIGN
2020	8FF2 ØD 54	L8FF2		FPØSGN	CHECK THE SIGN OF FPAØ
2021	8FF4 2A Ø4			L8FFA	BRANCH IF POSITIVE
2022	8FF6 ØF 54			FPØSGN	FORCE FPAØ SIGN TO BE POSITIVE
2023	8FF8 C6 2D		LDB		MINUS SIGN
2024	8FFA E7 CØ	L8FFA	STB	,U+	SAVE THE SIGN IN BUFFER * PUT A ZERO INTO THE BUFFER
2025 2026	8FFC C6 3Ø		LDB STB		* PUI A ZERU INIU IHE BUFFER
2020	8FFE E7 CØ 9000 84 01		ANDA		= CHECK THE EXPONENTIAL FORCE FLAG IN
2027	9002 10 26 01 07			L910D	= THE STATUS BYTE - BRANCH IF ACTIVE
2029	9006 8E BD C0			#LBDCØ	POINT X TO FLOATING POINT 1E + Ø9
2030	9009 BD BC A0			LBCAØ	COMPARE FPAØ TO (X)
2031	900C 2B 15			L9023	BRANCH IF FPAØ < 1E+Ø9
2032	900E BD BD D9			LBDD9	CONVERT FP NUMBER TO ASCII STRING
2033	9011 A6 80	L9Ø11	LDA	,X+	* ADVANCE POINTER TO END OF
2034	9013 26 FC			Ĺ9Ø11	* ASCII STRING (ZERO BYTE)
2035	9Ø15 A6 82	L9Ø15		, - X	= MOVE THE
2036	9017 A7 01		STA	\$Ø1,X	= ENTIRE STRING
2037	9019 8C 03 DA		CMPX	#STRBUF+3	= UP ONE
2038	901C 26 F7		BNE	L9Ø15	= BYTE
2039	901E 86 25		LDA	#'%'	* INSERT A % SIGN AT START OF
2040	9020 A7 84		STA	, Х	* STRING - OVERFLOW ERROR
2041	9022 39		RTS		
2042					
2043	9023 96 4F	L9Ø23	LDA	FPØEXP	= GET EXPONENT OF FPAØ
2044	9025 97 47			V47	= AND SAVE IT IN V47
2045 2046	9027 27 03 9029 BD 91 CD			L902C L91CD	BRANCH IF FPAØ = Ø
	9029 BD 91 CD	*	JSR	L91CD	CONVERT FPAØ TO NUMBER WITH 9 SIGNIFICANT
2047 2048	9Ø2C 96 47	L9Ø2C	LDA	V47	PLACES TO LEFT OF DECIMAL POINT GET BASE 10 EXPONENT OFFSET
2046	902E 10 2B 00 81	LJUZU		L9ØB3	BRANCH IF FPAØ < 100,000,000
2049	9032 40		NEGA	2,000	* CALCULATE THE NUMBER OF LEADING ZEROES TO INSERT -
2050	9033 9B D9		ADDA	VD9	* SUBTRACT BASE 10 EXPONENT OFFSET AND 9 (FPA0 HAS
2051	9035 80 09			#\$09	* 9 PLACES TO LEFT OF EXPONENT) FROM LEFT DIGIT COUNTER
2053	9037 BD 90 EA			L9ØEA	PUT ACCA ZEROES IN STRING BUFFER
2054	903A BD 92 63			L9263	INITIALIZE DECIMAL POINT AND COMMA COUNTERS
2055	903D BD 92 02			L9202	CONVERT FPAØ 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			L9249	CHECK FOR DECIMAL POINT
2060	904A 96 D8		LDA		GET THE RIGHT DIGIT COUNTER
2061	904C 26 02			L9050	BRANCH IF RIGHT DIGIT COUNTER ⇔ Ø
2062 2063	904E 33 5F	*	LEAU	\$-Ø1,U	* MOVE BUFFER POINTER BACK ONE - DELETE * DECIMAL POINT IF NO RIGHT DIGITS SPECIFIED
2064	9050 4A	L9Ø5Ø	DECA		SUBTRACT ONE (DECIMAL POINT)
2065	9051 BD 90 EA	23202		L9ØEA	PUT ACCA ZEROES INTO BUFFER (TRAILING ZEROES)
2066	9054 BD 91 85	L9Ø54		L9185	INSERT ASTERISK PADDING, FLOATING \$, AND POST-SIGN
2067	9Ø57 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	9Ø5C 27 Ø2			L9Ø6Ø	YES
2071	905E E7 C0			,U+ 	STORE THE POST-SIGN
2072	9060 6F C4	L9060	CLR	, U	CLEAR THE LAST CHARACTER IN THE BUFFER
2073 2074		* DEMOVE	ANV	EXTRA BLANKS OR ASTERISKS FROM TH	JE
2075				ER TO THE LEFT OF THE DECIMAL POI	
				#STRBUF+3	POINT X TO THE START OF THE BUFFER
20/6	9062 8F 03 DA				
2076 2077	9062 8E 03 DA 9065 30 01				
2076 2077 2078	9062 8E 03 DA 9065 30 01 9067 9F 0F		LEAX	TEMPTR	MOVE BUFFER POINTER UP ONE SAVE BUFFER POINTER IN TEMPTR
2077	9065 30 01		LEAX STX		MOVE BUFFER POINTER UP ONE
2077 2078	9065 30 01 9067 9F 0F		LEAX STX LDA	TEMPTR	MOVE BUFFER POINTER UP ONE SAVE BUFFER POINTER IN TEMPTR
2077 2078 2079 2080 2081	9065 30 01 9067 9F 0F 9069 96 3A		LEAX STX LDA	TEMPTR VARPTR+1 TEMPTR+1	MOVE BUFFER POINTER UP ONE SAVE BUFFER POINTER IN TEMPTR * GET ADDRESS OF DECIMAL POINT IN BUFFER, SUBTRACT * CURRENT POSITION AND SUBTRACT LEFT DIGIT COUNTER - * THE RESULT WILL BE ZERO WHEN TEMPTR+1 IS POINTING
2077 2078 2079 2080 2081 2082	9065 30 01 9067 9F 0F 9069 96 3A 906B 90 10 906D 90 D9		LEAX STX LDA SUBA SUBA	TEMPTR VARPTR+1 TEMPTR+1 VD9	MOVE BUFFER POINTER UP ONE SAVE BUFFER POINTER IN TEMPTR * GET ADDRESS OF DECIMAL POINT IN BUFFER, SUBTRACT * CURRENT POSITION AND SUBTRACT LEFT DIGIT COUNTER - * THE RESULT WILL BE ZERO WHEN TEMPTR+1 IS POINTING * TO THE FIRST DIGIT OF THE FORMAT STRING
2077 2078 2079 2080 2081 2082 2083	9065 30 01 9067 9F 0F 9069 96 3A 906B 90 10 906D 90 D9		LEAX STX LDA SUBA SUBA BEQ	TEMPTR VARPTR+1 TEMPTR+1 VD9 L9ØA9	MOVE BUFFER POINTER UP ONE SAVE BUFFER POINTER IN TEMPTR * GET ADDRESS OF DECIMAL POINT IN BUFFER, SUBTRACT * CURRENT POSITION AND SUBTRACT LEFT DIGIT COUNTER - * THE RESULT WILL BE ZERO WHEN TEMPTR+1 IS POINTING * TO THE FIRST DIGIT OF THE FORMAT STRING RETURN IF NO DIGITS TO LEFT OF THE DECIMAL POINT
2077 2078 2079 2080 2081 2082 2083 2084	9065 30 01 9067 9F 0F 9069 96 3A 906B 90 10 906D 90 D9 906F 27 38 9071 A6 84		LEAX STX LDA SUBA SUBA BEQ LDA	TEMPTR VARPTR+1 TEMPTR+1 VD9 L90A9	MOVE BUFFER POINTER UP ONE SAVE BUFFER POINTER IN TEMPTR * GET ADDRESS OF DECIMAL POINT IN BUFFER, SUBTRACT * CURRENT POSITION AND SUBTRACT LEFT DIGIT COUNTER - * THE RESULT WILL BE ZERO WHEN TEMPTR+1 IS POINTING * TO THE FIRST DIGIT OF THE FORMAT STRING RETURN IF NO DIGITS TO LEFT OF THE DECIMAL POINT GET THE CURRENT BUFFER CHARACTER
2077 2078 2079 2080 2081 2082 2083 2084 2085	9055 30 01 9067 9F 0F 9069 96 3A 906B 90 10 906B 90 D9 906F 27 38 9071 A6 84 9073 81 20		LEAX STX LDA SUBA SUBA BEQ LDA CMPA	TEMPTR VARPTR+1 TEMPTR+1 VD9 L9ØA9 ,X #SPACE	MOVE BUFFER POINTER UP ONE SAVE BUFFER POINTER IN TEMPTR * GET ADDRESS OF DECIMAL POINT IN BUFFER, SUBTRACT * CURRENT POSITION AND SUBTRACT LEFT DIGIT COUNTER - * THE RESULT WILL BE ZERO WHEN TEMPTR+1 IS POINTING * TO THE FIRST DIGIT OF THE FORMAT STRING RETURN IF NO DIGITS TO LEFT OF THE DECIMAL POINT GET THE CURRENT BUFFER CHARACTER SPACE?
2077 2078 2079 2080 2081 2082 2083 2084 2085 2086	9065 30 01 9067 9F 0F 9069 96 3A 906B 90 10 906D 90 D9 906F 27 38 9071 A6 84 9073 81 20 9075 27 EE		LEAX STX LDA SUBA SUBA BEQ LDA CMPA BEQ	TEMPTR VARPTR+1 TEMPTR+1 VD9 L90A9 ,X #SPACE L9065	MOVE BUFFER POINTER UP ONE SAVE BUFFER POINTER IN TEMPTR * GET ADDRESS OF DECIMAL POINT IN BUFFER, SUBTRACT * CURRENT POSITION AND SUBTRACT LEFT DIGIT COUNTER - * THE RESULT WILL BE ZERO WHEN TEMPTR+1 IS POINTING * TO THE FIRST DIGIT OF THE FORMAT STRING RETURN IF NO DIGITS TO LEFT OF THE DECIMAL POINT GET THE CURRENT BUFFER CHARACTER SPACE? YES - ADVANCE POINTER
2077 2078 2079 2080 2081 2082 2083 2084 2085 2086 2087	9065 30 01 9067 9F 0F 9069 96 3A 906B 90 10 906D 90 D9 906F 27 38 9071 A6 84 9073 81 20 9075 27 EE 9077 81 2A		LEAX STX LDA SUBA SUBA BEQ LDA CMPA BEQ CMPA	TEMPTR VARPTR+1 TEMPTR+1 VD9 L90A9 ,X #SPACE L9065 #'*	MOVE BUFFER POINTER UP ONE SAVE BUFFER POINTER IN TEMPTR * GET ADDRESS OF DECIMAL POINT IN BUFFER, SUBTRACT * CURRENT POSITION AND SUBTRACT LEFT DIGIT COUNTER - * THE RESULT WILL BE ZERO WHEN TEMPTR+1 IS POINTING * TO THE FIRST DIGIT OF THE FORMAT STRING RETURN IF NO DIGITS TO LEFT OF THE DECIMAL POINT GET THE CURRENT BUFFER CHARACTER SPACE? YES - ADVANCE POINTER ASTERISK?
2077 2078 2079 2080 2081 2082 2083 2084 2085 2086 2087 2088	9065 30 01 9067 9F 0F 9069 96 3A 906B 90 10 906D 90 D9 906F 27 38 9071 A6 84 9073 81 20 9075 27 EE 9077 81 2A 9079 27 EA		STX LDA SUBA SUBA BEQ LDA CMPA BEQ CMPA BEQ	TEMPTR VARPTR+1 TEMPTR+1 VD9 L90A9 ,X #SPACE L9065	MOVE BUFFER POINTER UP ONE SAVE BUFFER POINTER IN TEMPTR * GET ADDRESS OF DECIMAL POINT IN BUFFER, SUBTRACT * CURRENT POSITION AND SUBTRACT LEFT DIGIT COUNTER - * THE RESULT WILL BE ZERO WHEN TEMPTR+1 IS POINTING * TO THE FIRST DIGIT OF THE FORMAT STRING RETURN IF NO DIGITS TO LEFT OF THE DECIMAL POINT GET THE CURRENT BUFFER CHARACTER SPACE? YES - ADVANCE POINTER ASTERISK? YES - ADVANCE POINTER
2077 2078 2079 2080 2081 2082 2083 2084 2085 2086 2087 2088 2089	9065 30 01 9067 9F 0F 9069 96 3A 9068 90 10 906D 90 D9 906F 27 38 9071 A6 84 9073 81 20 9075 27 EE 9077 81 2A 9078 4F	L9065	LEAX STX LDA SUBA SUBA BEQ LDA CMPA BEQ CMPA BEQ CLRA	TEMPTR VARPTR+1 TEMPTR+1 VD9 L90A9 ,X #SPACE L9065 #'*'	MOVE BUFFER POINTER UP ONE SAVE BUFFER POINTER IN TEMPTR * GET ADDRESS OF DECIMAL POINT IN BUFFER, SUBTRACT * CURRENT POSITION AND SUBTRACT LEFT DIGIT COUNTER - * THE RESULT WILL BE ZERO WHEN TEMPTR+1 IS POINTING * TO THE FIRST DIGIT OF THE FORMAT STRING RETURN IF NO DIGITS TO LEFT OF THE DECIMAL POINT GET THE CURRENT BUFFER CHARACTER SPACE? YES - ADVANCE POINTER ASTERISK? YES - ADVANCE POINTER A ZERO ON THE STACK IS END OF DATA POINTER
2077 2078 2079 2080 2081 2082 2083 2084 2085 2086 2086 2087 2088 2089	9065 30 01 9067 9F 0F 9069 96 3A 906B 90 10 906D 90 D9 906F 27 38 9071 A6 84 9073 81 20 9075 27 EE 9077 81 2A 9079 27 EA 9079 4F 9070 34 02		LEAX STX LDA SUBA SUBA BEQ LDA CMPA BEQ CMPA BEQ CLRA PSHS	TEMPTR VARPTR+1 TEMPTR+1 VD9 L90A9 ,X #SPACE L9065 #'*'	MOVE BUFFER POINTER UP ONE SAVE BUFFER POINTER IN TEMPTR * GET ADDRESS OF DECIMAL POINT IN BUFFER, SUBTRACT * CURRENT POSITION AND SUBTRACT LEFT DIGIT COUNTER - * THE RESULT WILL BE ZERO WHEN TEMPTR+1 IS POINTING * TO THE FIRST DIGIT OF THE FORMAT STRING RETURN IF NO DIGITS TO LEFT OF THE DECIMAL POINT GET THE CURRENT BUFFER CHARACTER SPACE? YES - ADVANCE POINTER ASTERISK? YES - ADVANCE POINTER A ZERO ON THE STACK IS END OF DATA POINTER PUSH A CHARACTER ONTO THE STACK
2077 2078 2079 2080 2081 2082 2083 2084 2085 2086 2087 2088 2089 2090 2091	9065 30 01 9067 9F 0F 9069 96 3A 906B 90 10 906D 90 D9 906F 27 38 9071 A6 84 9073 81 20 9075 27 EE 9077 81 2A 9079 27 EA 9078 4F 907C 34 02 907C 34 02	L9065	LEAX STX LDA SUBA SUBA BEQ LDA CMPA BEQ CMPA BEQ CLRA PSHS LDA	TEMPTR VARPTR+1 VARPTR+1 VD9 L90A9 ,X #SPACE L9065 #'*' L9065 A ,X+	MOVE BUFFER POINTER UP ONE SAVE BUFFER POINTER IN TEMPTR * GET ADDRESS OF DECIMAL POINT IN BUFFER, SUBTRACT * CURRENT POSITION AND SUBTRACT LEFT DIGIT COUNTER - * THE RESULT WILL BE ZERO WHEN TEMPTR+1 IS POINTING * TO THE FIRST DIGIT OF THE FORMAT STRING RETURN IF NO DIGITS TO LEFT OF THE DECIMAL POINT GET THE CURRENT BUFFER CHARACTER SPACE? YES - ADVANCE POINTER ASTERISK? YES - ADVANCE POINTER A ZERO ON THE STACK IS END OF DATA POINTER PUSH A CHARACTER ONTO THE STACK GET NEXT CHARACTER FROM BUFFER
2077 2078 2079 2080 2081 2082 2083 2084 2085 2086 2087 2088 2089 2090 2091	9065 30 01 9067 9F 0F 9069 96 3A 906B 90 10 906D 90 D9 906F 27 38 9071 A6 84 9073 81 20 9075 27 EE 9077 81 2A 9079 27 EA 9079 4F 9070 34 02	L9065	LEAX STX LDA SUBA BEQ LDA CMPA BEQ CMPA BEQ CLRA PSHS LDA CMPA	TEMPTR VARPTR+1 VARPTR+1 VD9 L90A9 ,X #SPACE L9065 #'*' L9065 A ,X+	MOVE BUFFER POINTER UP ONE SAVE BUFFER POINTER IN TEMPTR * GET ADDRESS OF DECIMAL POINT IN BUFFER, SUBTRACT * CURRENT POSITION AND SUBTRACT LEFT DIGIT COUNTER - * THE RESULT WILL BE ZERO WHEN TEMPTR+1 IS POINTING * TO THE FIRST DIGIT OF THE FORMAT STRING RETURN IF NO DIGITS TO LEFT OF THE DECIMAL POINT GET THE CURRENT BUFFER CHARACTER SPACE? YES - ADVANCE POINTER ASTERISK? YES - ADVANCE POINTER A ZERO ON THE STACK IS END OF DATA POINTER PUSH A CHARACTER ONTO THE STACK
2077 2078 2079 2080 2081 2082 2083 2084 2085 2086 2087 2088 2089 2090 2090 2090 2092	9065 30 01 9067 9F 0F 9069 96 3A 9068 90 10 906D 90 D9 906F 27 38 9071 A6 84 9073 81 20 9075 27 EE 9077 81 2A 9079 27 EA 9078 4F 907C 34 02 907B 4F 907C 34 02 9080 81 2D	L9065	LEAX STX LDA SUBA BEQ CMPA BEQ CMPA PSHS LDA CMPA BEQ CMPA BEQ CMPA	TEMPTR VARPTR+1 TEMPTR+1 VD9 L9ØA9 ,X #SPACE L9Ø65 #'*' L9Ø65 A ,X+ #'-' L9Ø7C #'+'	MOVE BUFFER POINTER UP ONE SAVE BUFFER POINTER IN TEMPTR * GET ADDRESS OF DECIMAL POINT IN BUFFER, SUBTRACT * CURRENT POSITION AND SUBTRACT LEFT DIGIT COUNTER - * THE RESULT WILL BE ZERO WHEN TEMPTR+1 IS POINTING * TO THE FIRST DIGIT OF THE FORMAT STRING RETURN IF NO DIGITS TO LEFT OF THE DECIMAL POINT GET THE CURRENT BUFFER CHARACTER SPACE? YES - ADVANCE POINTER A ZERO ON THE STACK IS END OF DATA POINTER PUSH A CHARACTER ONTO THE STACK GET NEXT CHARACTER FROM BUFFER MINUS SIGN?
2077 2078 2079 2080 2081 2082 2083 2084 2085 2086 2087 2088 2089 2090 2091 2092 2093 2093	9065 30 01 9067 9F 0F 9069 96 3A 9068 90 10 906D 90 D9 906F 27 38 9071 A6 84 9073 81 20 9075 27 EE 9077 81 2A 9079 27 EA 9079 4F 9070 34 02 9078 4F 9070 34 02 9080 81 2D 9080 81 2D 9082 27 F8 9084 81 2B	L9065	LEAX STX LDA SUBA BEQ CMPA BEQ CMPA CMPA CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ	TEMPTR VARPTR+1 TEMPTR+1 VD9 L90A9 ,X #SPACE L9065 #'*' L9065 A ,X,+ #'-' L907C #'+'	MOVE BUFFER POINTER UP ONE SAVE BUFFER POINTER IN TEMPTR * GET ADDRESS OF DECIMAL POINT IN BUFFER, SUBTRACT * CURRENT POSITION AND SUBTRACT LEFT DIGIT COUNTER - * THE RESULT WILL BE ZERO WHEN TEMPTR+1 IS POINTING * TO THE FIRST DIGIT OF THE FORMAT STRING RETURN IF NO DIGITS TO LEFT OF THE DECIMAL POINT GET THE CURRENT BUFFER CHARACTER SPACE? YES - ADVANCE POINTER A ZERO ON THE STACK IS END OF DATA POINTER PUSH A CHARACTER ONTO THE STACK GET NEXT CHARACTER FROM BUFFER MINUS SIGN? YES PLUS SIGN? YES
2077 2078 2079 2080 2081 2082 2083 2084 2085 2086 2087 2090 2090 2091 2092 2093 2094 2095	9055 30 01 9067 9F 0F 9069 96 3A 9068 90 10 9060 90 D9 906F 27 38 9071 A6 84 9073 81 20 9075 27 EE 9077 81 2A 9079 27 EA 9078 4F 9077 A6 80 9080 81 2D 9082 27 F8 9084 81 2B 9086 27 F8 9088 81 24	L9065	LEAX STX LDA SUBA BEQ LDA CMPA BEQ CLRA PSHS LDA CMPA BEQ CMPA BEQ CMPA	TEMPTR VARPTR+1 TEMPTR+1 VD9 L90A9 ,X #SPACE L9065 #'*' L9065 A ,X+ #'-' L907C #'+'	MOVE BUFFER POINTER UP ONE SAVE BUFFER POINTER IN TEMPTR * GET ADDRESS OF DECIMAL POINT IN BUFFER, SUBTRACT * CURRENT POSITION AND SUBTRACT LEFT DIGIT COUNTER - * THE RESULT WILL BE ZERO WHEN TEMPTR+1 IS POINTING * TO THE FIRST DIGIT OF THE FORMAT STRING RETURN IF NO DIGITS TO LEFT OF THE DECIMAL POINT GET THE CURRENT BUFFER CHARACTER SPACE? YES - ADVANCE POINTER ASTERISK? YES - ADVANCE POINTER A ZERO ON THE STACK IS END OF DATA POINTER PUSH A CHARACTER ONTO THE STACK GET NEXT CHARACTER FROM BUFFER MINUS SIGN? YES PLUS SIGN? YES DOLLAR SIGN?
2077 2078 2079 2080 2081 2082 2083 2084 2085 2086 2087 2090 2091 2092 2093 2094 2095 2096 2097	9065 30 01 9067 9F 0F 9069 96 3A 906B 90 10 906F 27 38 9071 A6 84 9073 81 20 9075 27 EE 9077 81 2A 9079 27 EA 9078 4F 9070 34 02 9075 34 02 9076 A6 80 9080 81 2D 9082 27 F8 9084 81 2D 9084 81 2B 9086 27 F4 9088 81 24 9088 81 24 9088 81 24	L9065	LEAX STX LDA SUBA BEQ LDA CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ	TEMPTR VARPTR+1 TEMPTR+1 VD9 L9ØA9 ,X #SPACE L9Ø65 #'*' L9Ø65 A ,X+ #'-' L9Ø7C \$'\$'	MOVE BUFFER POINTER UP ONE SAVE BUFFER POINTER IN TEMPTR * GET ADDRESS OF DECIMAL POINT IN BUFFER, SUBTRACT * CURRENT POSITION AND SUBTRACT LEFT DIGIT COUNTER - * THE RESULT WILL BE ZERO WHEN TEMPTR+1 IS POINTING * TO THE FIRST DIGIT OF THE FORMAT STRING RETURN IF NO DIGITS TO LEFT OF THE DECIMAL POINT GET THE CURRENT BUFFER CHARACTER SPACE? YES - ADVANCE POINTER A ZERO ON THE STACK IS END OF DATA POINTER PUSH A CHARACTER ONTO THE STACK GET NEXT CHARACTER FROM BUFFER MINUS SIGN? YES DOLLAR SIGN? YES DOLLAR SIGN? YES
2077 2078 2080 2081 2081 2082 2083 2084 2085 2086 2087 2099 2090 2091 2092 2093 2094 2095 2096 2097 2098	9065 30 01 9067 9F 0F 9069 96 3A 9068 90 10 906D 90 D9 906F 27 38 9071 A6 84 9073 81 20 9075 27 EE 9077 81 2A 9079 27 EA 9079 4F 9070 34 02 9078 4F 9070 34 02 9080 81 2D 9082 27 F8 9086 27 F4 9088 81 24 9088 27 F4 9088 81 24 9088 81 24 9088 81 24 9088 81 24 9088 81 24 9088 81 24 9088 81 24	L9065	LEAX STX LDA SUBA SUBA BEQ CDPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA	TEMPTR VARPTR+1 TEMPTR+1 VD9 L90A9 ,X #SPACE L9065 #'*' L9065 A ,X+ #'-' L907C #'+' L907C #'97C #'8' L907C	MOVE BUFFER POINTER UP ONE SAVE BUFFER POINTER IN TEMPTR * GET ADDRESS OF DECIMAL POINT IN BUFFER, SUBTRACT * CURRENT POSITION AND SUBTRACT LEFT DIGIT COUNTER - * THE RESULT WILL BE ZERO WHEN TEMPTR+1 IS POINTING * TO THE FIRST DIGIT OF THE FORMAT STRING RETURN IF NO DIGITS TO LEFT OF THE DECIMAL POINT GET THE CURRENT BUFFER CHARACTER SPACE? YES - ADVANCE POINTER A ZERO ON THE STACK IS END OF DATA POINTER PUSH A CHARACTER ONTO THE STACK GET NEXT CHARACTER FROM BUFFER MINUS SIGN? YES PLUS SIGN? YES DOLLAR SIGN? YES ZERO?
2077 2078 2089 2089 2081 2082 2083 2084 2085 2086 2087 2099 2091 2092 2093 2094 2095 2096 2097 2098	9065 30 01 9067 9F 0F 9069 96 3A 9068 90 10 9068 97 10 9067 27 38 9071 A6 84 9073 81 20 9075 27 EE 9077 81 2A 9079 27 EA 9078 4F 9077 A6 80 9080 81 2D 9082 27 F8 9084 81 2B 9086 27 F4 9088 81 24 9088 81 24 9088 81 24 9088 27 F0 9088 81 24 9088 27 F0	L9065	LEAX STX LDA SUBA BEQ LDA CMPA BEQ B BEQ B BEQ B B B B B B B B B B B B	TEMPTR VARPTR+1 TEMPTR+1 VD9 L90A9 ,X #SPACE L9065 #'*' L9065 A ,X+ #'-' L907C #'+' L907C \$'\$' L907C #'#' L907C	MOVE BUFFER POINTER UP ONE SAVE BUFFER POINTER IN TEMPTR * GET ADDRESS OF DECIMAL POINT IN BUFFER, SUBTRACT * CURRENT POSITION AND SUBTRACT LEFT DIGIT COUNTER - * THE RESULT WILL BE ZERO WHEN TEMPTR+1 IS POINTING * TO THE FIRST DIGIT OF THE FORMAT STRING RETURN IF NO DIGITS TO LEFT OF THE DECIMAL POINT GET THE CURRENT BUFFER CHARACTER SPACE? YES - ADVANCE POINTER ASTERISK? YES - ADVANCE POINTER A ZERO ON THE STACK IS END OF DATA POINTER PUSH A CHARACTER ONTO THE STACK GET NEXT CHARACTER FROM BUFFER MINUS SIGN? YES PLUS SIGN? YES DULLAR SIGN? YES ZERO? NO - ERROR
2077 2078 2089 2081 2082 2083 2084 2085 2086 2087 2091 2092 2093 2094 2095 2097 2098 2097 2098 2099 2099 2099 2099 2099 2099 2099	9065 30 01 9067 9F 0F 9069 96 3A 906B 90 10 906F 27 38 9071 A6 84 9073 81 20 9075 27 EE 9077 81 2A 9079 27 EA 9078 4F 9070 34 02 9070 34 02 9080 81 2D 9082 27 F8 9084 81 2B 9086 27 F4 9088 81 2B 9088 81 2B	L9065	LEAX STX LDA SUBA SUBA BEQ CMPA BEQ CLRA CMPA BEQ B B B B B B B B B B B B B B B B B B	TEMPTR VARPTR+1 TEMPTR+1 VD9 L9ØA9 ,X #SPACE L9Ø65 #'*' L9Ø65 A ,X+ #'-' L9Ø7C \$'\$' \$'\$' L9Ø7C \$'\$' L9Ø7C \$'\$' L9Ø7C #'ø' L9Ø9E \$\vec{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{	MOVE BUFFER POINTER UP ONE SAVE BUFFER POINTER IN TEMPTR * GET ADDRESS OF DECIMAL POINT IN BUFFER, SUBTRACT * CURRENT POSITION AND SUBTRACT LEFT DIGIT COUNTER - * THE RESULT WILL BE ZERO WHEN TEMPTR+1 IS POINTING * TO THE FIRST DIGIT OF THE FORMAT STRING RETURN IF NO DIGITS TO LEFT OF THE DECIMAL POINT GET THE CURRENT BUFFER CHARACTER SPACE? YES - ADVANCE POINTER A ZERO ON THE STACK IS END OF DATA POINTER PUSH A CHARACTER ONTO THE STACK GET NEXT CHARACTER FROM BUFFER MINUS SIGN? YES DOLLAR SIGN? YES DOLLAR SIGN? YES ZERO? NO - ERROR GET CHARACTER FOLLOWING ZERO
2077 2078 2080 2081 2082 2083 2084 2085 2086 2087 2090 2091 2092 2093 2094 2095 2096 2097 2098 2099 2099 2099 2099 2099 2099 2099	9065 30 01 9067 9F 0F 9069 96 3A 9068 90 10 906D 90 D9 906F 27 38 9071 A6 84 9073 81 20 9075 27 EE 9077 81 2A 9078 4F 9070 34 02 9078 4F 9070 34 02 9080 81 2D 9082 27 F8 9080 81 2D 9082 27 F4 9088 81 2D 9088 81 24 9088 81 24 9088 81 24 9088 81 24 9088 81 24 9088 81 24 9088 81 30 9080 81 30 9080 81 30 9080 81 30 9080 81 30 9080 81 30 9080 81 30 9080 86 01	L9065	LEAX STX LDA SUBA SUBA SUBA BEQ LDA CMPA BEQ CMPA BNE BNE SR	TEMPTR VARPTR+1 TEMPTR+1 VD9 L90A9 ,X #SPACE L9065 #'*' L9065 A ,X+ #'-' L907C #'+' L907C #'9' \$'\$' L908E \$\delta_1, X L908A	MOVE BUFFER POINTER UP ONE SAVE BUFFER POINTER IN TEMPTR * GET ADDRESS OF DECIMAL POINT IN BUFFER, SUBTRACT * CURRENT POSITION AND SUBTRACT LEFT DIGIT COUNTER - * THE RESULT WILL BE ZERO WHEN TEMPTR+1 IS POINTING * TO THE FIRST DIGIT OF THE FORMAT STRING RETURN IF NO DIGITS TO LEFT OF THE DECIMAL POINT GET THE CURRENT BUFFER CHARACTER SPACE? YES - ADVANCE POINTER A ZERO ON THE STACK IS END OF DATA POINTER PUSH A CHARACTER ONTO THE STACK GET NEXT CHARACTER FROM BUFFER MINUS SIGN? YES DULLAR SIGN? YES DULLAR SIGN? YES ZERO? NO - ERROR GET CHARACTER FOLLOWING ZERO CLEAR CARRY IF NUMERIC
2077 2078 2080 2081 2082 2083 2084 2085 2086 2087 2099 2091 2092 2093 2094 2095 2096 2097 2099 2099 2099 2091 2092 2091 2092 2093 2094 2095 2096 2097 2099 2099 2099 2099 2099 2099 2099	9055 30 01 9067 9F 0F 9069 96 3A 9068 90 10 9068 90 D9 906F 27 38 9071 A6 84 9073 81 20 9075 27 EE 9077 81 2A 9079 27 EA 9078 4F 9077 A6 80 9080 81 2D 9082 27 F8 9084 81 2B 9086 27 F4 9088 81 24 9088 81 24 9088 81 24 9088 27 F0 9088 81 24 9088 81 24 9088 81 24 9088 81 24 9088 81 24 9088 81 24 9088 81 30 9088 26 06 9090 A6 01 9092 8D 16	L9065	LEAX STX LDA SUBA SUBA SUBA SUBA BEQ LDA CMPA BEQ CLRA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ LDA BNE LDA BNE LDA BNE LDA BNE BLO	TEMPTR VARPTR+1 TEMPTR+1 VD9 L90A9 ,X #SPACE L9065 #'*' L9065 A ,X+ #'-' L907C #'+' L907C #'f' L909E \$01,X L909E	MOVE BUFFER POINTER UP ONE SAVE BUFFER POINTER IN TEMPTR * GET ADDRESS OF DECIMAL POINT IN BUFFER, SUBTRACT * CURRENT POSITION AND SUBTRACT LEFT DIGIT COUNTER - * THE RESULT WILL BE ZERO WHEN TEMPTR+1 IS POINTING * TO THE FIRST DIGIT OF THE FORMAT STRING RETURN IF NO DIGITS TO LEFT OF THE DECIMAL POINT GET THE CURRENT BUFFER CHARACTER SPACE? YES - ADVANCE POINTER A ZERO ON THE STACK IS END OF DATA POINTER PUSH A CHARACTER ONTO THE STACK GET NEXT CHARACTER FROM BUFFER MINUS SIGN? YES PLUS SIGN? YES DULLAR SIGN? YES ZERO? NO - ERROR GET CHARACTER FOLLOWING ZERO CLEAR CARRY IF NUMERIC BRANCH IF NOT A NUMERIC CHARACTER - ERROR
2077 2078 2080 2081 2082 2083 2084 2085 2086 2087 2090 2091 2092 2093 2094 2095 2096 2097 2098 2099 2099 2099 2099 2099 2099 2099	9065 30 01 9067 9F 0F 9069 96 3A 9068 90 10 906D 90 D9 906F 27 38 9071 A6 84 9073 81 20 9075 27 EE 9077 81 2A 9078 4F 9070 34 02 9078 4F 9070 34 02 9080 81 2D 9082 27 F8 9080 81 2D 9082 27 F4 9088 81 2D 9088 81 24 9088 81 24 9088 81 24 9088 81 24 9088 81 24 9088 81 24 9088 81 30 9080 81 30 9080 81 30 9080 81 30 9080 81 30 9080 81 30 9080 81 30 9080 86 01	L9065	LEAX STX LDA SUBA SUBA SUBA SUBA BEQ LDA CMPA BEQ CLRA PSHS BEQ CMPA DEQ CMPA BEQ CMPA BEQ DHULS	TEMPTR VARPTR+1 TEMPTR+1 VD9 L90A9 ,X #SPACE L9065 #'*' L9065 A ,X+ #'-' L907C \$'\$' L907C \$'\$' L907C #'0' L909E \$01,X L908AA L909E A	MOVE BUFFER POINTER UP ONE SAVE BUFFER POINTER IN TEMPTR * GET ADDRESS OF DECIMAL POINT IN BUFFER, SUBTRACT * CURRENT POSITION AND SUBTRACT LEFT DIGIT COUNTER - * THE RESULT WILL BE ZERO WHEN TEMPTR+1 IS POINTING * TO THE FIRST DIGIT OF THE FORMAT STRING RETURN IF NO DIGITS TO LEFT OF THE DECIMAL POINT GET THE CURRENT BUFFER CHARACTER SPACE? YES - ADVANCE POINTER A ZERO ON THE STACK IS END OF DATA POINTER PUSH A CHARACTER ONTO THE STACK GET NEXT CHARACTER FROM BUFFER MINUS SIGN? YES DULLAR SIGN? YES DULLAR SIGN? YES ZERO? NO - ERROR GET CHARACTER FOLLOWING ZERO CLEAR CARRY IF NUMERIC
2077 2078 2089 2081 2082 2083 2084 2085 2086 2087 2091 2092 2093 2094 2095 2096 2097 2098 2097 2098 2099 2090 2091 2091 2092 2093 2094 2095 2096 2097 2098 2099 2090 2091 2092 2093 2094 2095 2096 2097 2098 2097 2098 2099 2099 2090 2091 2092 2093 2094 2095 2096 2097 2098 2097 2098 2099 2099 2099 2099 2099 2099 2099	9065 30 01 9067 9F 0F 9069 96 3A 906B 90 10 906F 27 38 9071 A6 84 9073 81 20 9075 27 EE 9077 81 2A 9079 27 EA 9078 4F 9070 34 02 9070 34 02 9080 81 2D 9082 27 F8 9084 81 2B 9086 27 F4 9088 81 2B 9086 27 F4 9088 81 24 9088 81 25 9088 81 30 9088 81 25 9088 81 30 9088 81 3	L9065	LEAX STX LDA SUBA SUBA BEQ LDA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BNE BEQ STA STA	TEMPTR VARPTR+1 TEMPTR+1 VD9 L90A9 ,X #SPACE L9065 #'*' L9065 A ,X+ #'-' L907C \$'\$' L907C \$'\$' L907C #'0' L909E \$01,X L908AA L909E A	MOVE BUFFER POINTER UP ONE SAVE BUFFER POINTER IN TEMPTR * GET ADDRESS OF DECIMAL POINT IN BUFFER, SUBTRACT * CURRENT POSITION AND SUBTRACT LEFT DIGIT COUNTER - * THE RESULT WILL BE ZERO WHEN TEMPTR+1 IS POINTING * TO THE FIRST DIGIT OF THE FORMAT STRING RETURN IF NO DIGITS TO LEFT OF THE DECIMAL POINT GET THE CURRENT BUFFER CHARACTER SPACE? YES - ADVANCE POINTER A ZERO ON THE STACK IS END OF DATA POINTER PUSH A CHARACTER ONTO THE STACK GET NEXT CHARACTER FROM BUFFER MINUS SIGN? YES DOLLAR SIGN? YES DOLLAR SIGN? YES ZERO? NO - ERROR GET CHARACTER FOLLOWING ZERO CLEAR CARRY IF NUMERIC BRANCH IF NOT A NUMERIC CHARACTER - ERROR * PULL A CHARACTER OFF OF THE STACK
2077 2078 2089 2081 2082 2083 2084 2085 2086 2087 2091 2092 2093 2094 2095 2096 2097 2098 2099 2090 2101 2101 2102 2103 2104 2104 2105 2106	9065 30 01 9067 9F 0F 9069 96 3A 906B 90 10 906F 27 38 9071 A6 84 9073 81 20 9075 27 EE 9077 81 2A 9078 4F 9070 34 02 9078 4F 9070 34 02 9080 81 2D 9082 27 F8 9084 81 2B 9086 27 F4 9088 81 2B 9086 27 F4 9088 81 24 9088 81 24 9088 81 24 9088 81 24 9088 81 24 9088 81 24 9088 81 24 9088 81 24 9088 81 24 9088 81 24 9088 81 24 9088 81 24 9088 81 24 9088 81 30 9088 87 82 9098 87 82 9098 87 82	L907C	LEAX STX LDA SUBA SUBA SUBA BEQ LDA BEQ CMPA BEQ	TEMPTR VARPTR+1 TEMPTR+1 VD9 L90A9 ,X #SPACE L9065 #'*' L9065 A ,X+ #'-' L907C *'s' *'s' L907C #'g' L909E \$40, X L909A L909E \$40, X L908A L909E A ,-X L9086	MOVE BUFFER POINTER UP ONE SAVE BUFFER POINTER UP ONE SAVE BUFFER POINTER IN TEMPTR * GET ADDRESS OF DECIMAL POINT IN BUFFER, SUBTRACT * CURRENT POSITION AND SUBTRACT LEFT DIGIT COUNTER - * THE RESULT WILL BE ZERO WHEN TEMPTR+1 IS POINTING * TO THE FIRST DIGIT OF THE FORMAT STRING RETURN IF NO DIGITS TO LEFT OF THE DECIMAL POINT GET THE CURRENT BUFFER CHARACTER SPACE? YES - ADVANCE POINTER A ZERO ON THE STACK IS END OF DATA POINTER PUSH A CHARACTER ONTO THE STACK GET NEXT CHARACTER FROM BUFFER MINUS SIGN? YES DOLLAR SIGN? YES DOLLAR SIGN? YES ZERO? NO - ERROR GET CHARACTER FOLLOWING ZERO CLEAR CARRY IF NUMERIC BRANCH IF NOT A NUMERIC CHARACTER - ERROR * AND PUT IT BACK IN THE STRING BUFFER * KEEP GOING UNTIL ZERO FLAG KEEP CLEANING UP THE IMPUT BUFFER
2077 2078 2089 2089 2081 2082 2083 2084 2085 2086 2087 2099 2091 2092 2093 2094 2095 2096 2097 2098 2099 2100 2101 2102 2103 2104 2105 2105 2105 2105 2105 2105 2105 2105	9055 30 01 9067 9F 0F 9069 96 3A 9068 90 10 9068 90 D9 906F 27 38 9071 A6 84 9073 81 20 9075 27 EE 9077 81 2A 9079 27 EA 9078 4F 9077 A4 02 9077 A6 80 9080 81 2D 9082 27 F8 9084 81 2B 9086 27 F4 9088 81 24 9080 27 F4 9080 81 24 9080 81 24 9080 81 29 9080 81 24 9080 81 24 9080 81 24 9080 81 30	L9065	LEAX STX SUBA SUBA BEQ LDA BEQ CMPA BEQ BEQ BEQ BEQ BEQ BEQ BEQ BEQ BEQ BEQ	TEMPTR VARPTR+1 TEMPTR+1 VD9 L90A9 ,X #SPACE L9065 #'*' L9065 A ,X+ #'-' L907C *'s' *'s' L907C #'g' L909E \$40, X L909A L909E \$40, X L908A L909E A ,-X L9086	MOVE BUFFER POINTER UP ONE SAVE BUFFER POINTER IN TEMPTR * GET ADDRESS OF DECIMAL POINT IN BUFFER, SUBTRACT * CURRENT POSITION AND SUBTRACT LEFT DIGIT COUNTER - * THE RESULT WILL BE ZERO WHEN TEMPTR+1 IS POINTING * TO THE FIRST DIGIT OF THE FORMAT STRING RETURN IF NO DIGITS TO LEFT OF THE DECIMAL POINT GET THE CURRENT BUFFER CHARACTER SPACE? YES - ADVANCE POINTER ASTERISK? YES - ADVANCE POINTER A ZERO ON THE STACK IS END OF DATA POINTER PUSH A CHARACTER ONTO THE STACK GET NEXT CHARACTER FROM BUFFER MINUS SIGN? YES PLUS SIGN? YES DOLLAR SIGN? YES DOLLAR SIGN? YES CERO? NO - ERROR GET CHARACTER FOLLOWING ZERO CLEAR CARRY IF NUMERIC BRANCH IF NOT A NUMERIC CHARACTER - ERROR * PULL A CHARACTER OFF OF THE STACK * AND PUL IT BACK IN THE STRING BUFFER * KEEP GOING UNTIL ZERO FLAG KEEP CLEANING UP THE INPUT BUFFER = REMOVE THE CHARACTERS ON
2077 2078 2089 2081 2082 2083 2084 2085 2086 2087 2099 2091 2092 2093 2094 2095 2096 2097 2098 2099 2100 2101 2102 2103 2104 2105 2106 2107 2106 2107 2106 2107 2108	9055 30 01 9067 9F 0F 9069 96 3A 9068 90 10 906D 90 D9 906F 27 38 9071 A6 84 9073 81 20 9075 27 EE 9077 81 2A 9079 27 EA 9078 4F 9077 34 02 9078 A6 80 9080 81 2D 9082 27 F8 9084 81 2B 9086 27 F4 9088 81 24 9088 27 F0 9088 27 F0 9088 27 F0 9088 27 F0 9088 31 24 9088 27 F0 9088 28 F0 9088 27 F0 9088 28 F0 908	L907C	LEAX STX LDA SUBA SUBA SUBA BEQ LDA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BNE LDA BNE LDA BNE LDA BNE LDA BNE LDA BNE LDA STA BNE BNE STA BNE STA	TEMPTR VARPTR+1 TEMPTR+1 VD9 L90A9 ,X #SPACE L9065 #'*' L9065 A ,X+ #'-' L907C *'\$' L907C #'0' L907C #'0' L908A L908A L908A L9086 A ,-X L9086 A ,-X L9086 A	MOVE BUFFER POINTER UP ONE SAVE BUFFER POINTER IN TEMPTR * GET ADDRESS OF DECIMAL POINT IN BUFFER, SUBTRACT * CURRENT POSITION AND SUBTRACT LEFT DIGIT COUNTER - * THE RESULT WILL BE ZERO WHEN TEMPTR+1 IS POINTING * TO THE FIRST DIGIT OF THE FORMAT STRING RETURN IF NO DIGITS TO LEFT OF THE DECIMAL POINT GET THE CURRENT BUFFER CHARACTER SPACE? YES - ADVANCE POINTER A ZERO ON THE STACK IS END OF DATA POINTER PUSH A CHARACTER ONTO THE STACK GET NEXT CHARACTER FROM BUFFER MINUS SIGN? YES PLUS SIGN? YES DOLLAR SIGN? YES ZERO? NO - ERROR GET CHARACTER FOLLOWING ZERO CLEAR CARRY IF NUMERIC BRANCH IF NOT A NUMERIC CHARACTER - ERROR * PULL A CHARACTER OFF OF THE STACK * AND PUT IT BACK IN THE STRING BUFFER * KEEP GOING UNTIL ZERO FLAG KEEP CLEANING UP THE INPUT BUFFER REMOVE THE CHARACTER SON = THE STACK AND EXIT WHEN
2077 2078 2080 2081 2082 2083 2084 2085 2086 2087 2091 2092 2093 2094 2095 2096 2097 2098 2099 2100 2101 2102 2103 2104 2106 2106 2107 2108	9065 30 01 9067 9F 0F 9069 96 3A 906B 90 10 906F 27 38 9071 A6 84 9073 81 20 9075 27 EE 9077 81 2A 9078 4F 9070 34 02 9088 81 2D 9082 27 F8 9088 81 2D 9082 27 F4 9088 81 2B 9086 27 F4 9088 81 24 908	L907C	LEAX STX SUBA SUBA BEQ LDA CMPA BEQ CHPA BEQ CHPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ SMPA BEQ SMPA BEQ SMPA BEQ SMPA BEQ SMPA BEQ SMPA BEQ SMPA BEQ SMPA BEQ SMPA BEQ SMPA SMPA SMPA SMPA SMPA SMPA SMPA SMPA	TEMPTR VARPTR+1 TEMPTR+1 VD9 L90A9 ,X #SPACE L9065 #'*' L9065 A ,X+ #'-' L907C *'\$' \$'\$' L907C *'\$' L909E \$01,1X L90AA L909E A ,-X L9096 A L9096 L9096	MOVE BUFFER POINTER UP ONE SAVE BUFFER POINTER IN TEMPTR * GET ADDRESS OF DECIMAL POINT IN BUFFER, SUBTRACT * CURRENT POSITION AND SUBTRACT LEFT DIGIT COUNTER - * THE RESULT WILL BE ZERO WHEN TEMPTR+1 IS POINTING * TO THE FIRST DIGIT OF THE FORMAT STRING RETURN IF NO DIGITS TO LEFT OF THE DECIMAL POINT GET THE CURRENT BUFFER CHARACTER SPACE? YES - ADVANCE POINTER A ZERO ON THE STACK IS END OF DATA POINTER PUSH A CHARACTER ONTO THE STACK GET NEXT CHARACTER FROM BUFFER MINUS SIGN? YES DOLLAR SIGN? YES DOLLAR SIGN? YES DOLLAR ARACTER FOLLOWING ZERO CLEAR CARRY IF NUMERIC BRANCH IF NOT A NUMERIC CHARACTER - ERROR * AND PUT IT BACK IN THE STACK * AND PUT IT BACK IN THE STRING BUFFER * KEEP GOING UNTIL ZERO FLAG KEEP CLEANING UP THE IMPUT BUFFER = REMOVE THE CHARACTERS ON THE STACK AND EXIT WHEN = ZERO FLAG FOUND
2077 2078 2089 2089 2081 2082 2083 2084 2085 2086 2087 2099 2091 2092 2093 2094 2095 2096 2097 2098 2099 2100 2102 2103 2104 2105 2107 2108 2108 2108 2108 2108 2108 2108 2108	9065 30 01 9067 9F 0F 9069 96 3A 9068 90 10 9068 90 10 9066 27 38 9071 A6 84 9073 81 20 9075 27 EE 9077 81 2A 9079 27 EA 9078 4F 9077 81 2A 9078 4F 9077 A6 80 9080 81 2D 9082 27 F8 9084 81 2D 9082 27 F8 9084 81 2B 9086 27 F4 9082 81 30 9082 81 30 9088 81 24 9088 87 60 9088 87 60 9089 87 60 9089 87 60 9089 87 82 9099 87 82 9099 87 82 9099 87 82 9098 87 82 9098 87 82 9098 87 82 9098 87 82 9088 98 67 9088 87 82	L907C	LEAX STX STX SUBA SUBA SUBA SUBA BEQ CMPA BEQ CLRA BEQ CMPA BEQ BEQ CMPA BEQ BEQ CMPA BEQ BEQ CMPA BEQ BEQ BEQ CMPA BEQ	TEMPTR VARPTR+1 TEMPTR+1 VD9 L90A9 ,X #SPACE L9065 #'*' L9065 A ,X+ #'-' L907C #'+' L907C #'+' L907C #'+' L907C #' L909E A L909E A L909E A L909E TEMPTR	MOVE BUFFER POINTER UP ONE SAVE BUFFER POINTER IN TEMPTR * GET ADDRESS OF DECIMAL POINT IN BUFFER, SUBTRACT * CURRENT POSITION AND SUBTRACT LEFT DIGIT COUNTER - * THE RESULT WILL BE ZERO WHEN TEMPTR+1 IS POINTING * TO THE FIRST DIGIT OF THE FORMAT STRING RETURN IF NO DIGITS TO LEFT OF THE DECIMAL POINT GET THE CURRENT BUFFER CHARACTER SPACE? YES - ADVANCE POINTER ASTERISK? YES - ADVANCE POINTER A ZERO ON THE STACK IS END OF DATA POINTER PUSH A CHARACTER ONTO THE STACK GET NEXT CHARACTER FROM BUFFER MINUS SIGN? YES PLUS SIGN? YES PLUS SIGN? YES ZERO? NO - ERROR GET CHARACTER FOLLOWING ZERO CLEAR CARPY IF NUMERIC BRANCH IF NOT A NUMERIC CHARACTER - ERROR * PULL A CHARACTER OFF OF THE STACK * AND PUT IT BACK IN THE STRING BUFFER * KEEP GOING UNTIL ZERO FLAG KEEP CLEANING UP THE INPUT BUFFER = REMOVE THE CHARACTERS ON = THE STACK AND EXIT WHEN = ZERO FLAG FOUND GET THE STRING BUFFER START POINTER
2077 2078 2089 2089 2081 2082 2083 2084 2085 2086 2087 2099 2091 2092 2093 2094 2095 2096 2097 2098 2099 2100 2102 2103 2104 2105 2107 2108 2108 2108 2108 2108 2108 2108 2108	9065 30 01 9067 9F 0F 9069 96 3A 906B 90 10 906F 27 38 9071 A6 84 9073 81 20 9075 27 EE 9077 81 2A 9078 4F 9070 34 02 9088 81 2D 9082 27 F8 9088 81 2D 9082 27 F4 9088 81 2B 9086 27 F4 9088 81 24 908	L907C	LEAX STX SUBA SUBA BEQ LDA CMPA BEQ CHPA BEQ CHPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ SMPA BEQ SMPA BEQ SMPA BEQ SMPA BEQ SMPA BEQ SMPA BEQ SMPA BEQ SMPA BEQ SMPA BEQ SMPA SMPA SMPA SMPA SMPA SMPA SMPA SMPA	TEMPTR VARPTR+1 TEMPTR+1 VD9 L90A9 ,X #SPACE L9065 #'*' L9065 A A ,X+ #'-' L907C \$'\$' '\$' L907C #'0' L907C #'0' L909E A ,-X L909AA L909E A ,-X L9086 L9096 A L9096 L9065 A	MOVE BUFFER POINTER UP ONE SAVE BUFFER POINTER IN TEMPTR * GET ADDRESS OF DECIMAL POINT IN BUFFER, SUBTRACT * CURRENT POSITION AND SUBTRACT LEFT DIGIT COUNTER - * THE RESULT WILL BE ZERO WHEN TEMPTR+1 IS POINTING * TO THE FIRST DIGIT OF THE FORMAT STRING RETURN IF NO DIGITS TO LEFT OF THE DECIMAL POINT GET THE CURRENT BUFFER CHARACTER SPACE? YES - ADVANCE POINTER A ZERO ON THE STACK IS END OF DATA POINTER PUSH A CHARACTER ONTO THE STACK GET NEXT CHARACTER FROM BUFFER MINUS SIGN? YES DOLLAR SIGN? YES DOLLAR SIGN? YES DOLLAR SIGN? YES DOLLAR ARACTER FOLLOWING ZERO CLEAR CARRY IF NUMERIC BRANCH IF NOT A NUMERIC CHARACTER - ERROR * MO PUT IT BACK IN THE STRING BUFFER * KEEP GOING UNTIL ZERO FLAG KEEP CLEANING UP THE IMPUT BUFFER * KEEP GOING UNTIL ZERO FLAG KEEP CLEANING UP THE IMPUT BUFFER = REMOVE THE CHARACTERS ON THE STACK AND EXIT WHEN = ZERO FLAG FOUND

2113 2114	9ØA9 39	L9ØA9 *	RTS		
2115				IF NUMERIC	
2116 2117	90AA 81 30 90AC 25 04	L9ØAA		#'0' L90B2	ASCII ZERO RETURN IF ACCA < ASCII Ø
2118	90AE 80 3A		SUBA	#'9'+1	*
2119 2120	90B0 80 C6 90B2 39	L9ØB2	SUBA RTS	#-('9'+1)	*CARRY CLEAR IF NUMERIC
2121	3882 03	*			
2122 2123	90B3 96 D8	* PROCES L9ØB3		ITEM-LIST WHICH IS < 100,000,000 VD8	GET RIGHT DIGIT COUNTER
2124	9ØB5 27 Ø1		BEQ	L9ØB8	BRANCH IF NO FORMATTED DIGITS TO THE RIGHT OF DECIMAL PT
2125 2126	90B7 4A 90B8 9B 47	L9ØB8	DECA ADDA	V47	SUBTRACT ONE FOR DECIMAL POINT *ADD THE BASE 10 EXPONENT OFFSET - ACCA CONTAINS THE
2127		*			*NUMBER OF SHIFTS REQUIRED TO ADJUST FPAØ TO THE SPECIFIED
2128 2129	90BA 2B 01	*	BMI	L9ØBD	*NUMBER OF DIGITS TO THE RIGHT OF THE DECIMAL POINT IF ACCA >= Ø THEN NO SHIFTS ARE REQUIRED
2130	9ØBC 4F	L9ØBD	CLRA		FORCE SHIFT COUNTER = Ø SAVE INITIAL SHIFT COUNTER ON THE STACK
2131 2132	90BD 34 02 90BF 2A 0A	L90BF	PSHS BPL	L9ØCB	EXIT ROUTINE IF POSITIVE
2133 2134	90C1 34 02 90C3 BD BB 82		PSHS	A LBB82	SAVE SHIFT COUNTER ON STACK
2135	90C6 35 02		PULS		DIVIDE FPAØ BY 10 - SHIFT ONE DIGIT TO RIGHT GET SHIFT COUNTER FROM THE STACK
2136 2137	90C8 4C 90C9 20 F4		INCA	L9ØBF	BUMP SHIFT COUNTER UP BY ONE CHECK FOR FURTHER DIVISION
2138	90CB 96 47	L9ØCB		V47	* GET BASE 10 EXPONENT OFFSET, ADD INITIAL SHIFT COUNTER
2139 2140	90CD A0 E0 90CF 97 47		SUBA STA	,S+ V47	* AND SAVE NEW BASE 10 EXPONENT OFFSET - BECAUSE * FPA0 WAS SHIFTED ABOVE
2141	90D1 8B 09		ADDA	#\$09	=ADD NINE (SIGNIFICANT PLACES) AND BRANCH IF THERE ARE NO
2142 2143	90D3 2B 19 90D5 96 D9		BMI LDA	L9ØEE VD9	=ZEROES TO THE LEFT OF THE DECIMAL POINT IN THIS PRINT ITEM *DETERMINE HOW MANY FILLER ZEROES TO THE LEFT OF THE DECIMAL
2144	90D7 80 09		SUBA	#\$Ø9	*POINT. GET THE NUMBER OF FORMAT PLACES TO LEFT OF DECIMAL
2145 2146	90D9 90 47 90DB 8D 0D		SUBA BSR	V47 L9ØEA	*POINT, SUBTRACT THE BASE 10 EXPONENT OFFSET AND THE CONSTANT 9 *(UNNORMALIZATION)-THEN OUTPUT THAT MANY ZEROES TO THE BUFFER
2147	9ØDD BD 92 63		JSR	L9263	INITIALIZE DECIMAL POINT AND COMMA COUNTERS
2148 2149	90E0 20 1D	*	BRA	L90FF	PROCESS THE REMAINDER OF THE PRINT ITEM
2150	0050 04 00) ASCII ZEROES IN BUFFER	CAME TERM COMMETER
2151 2152	90E2 34 02 90E4 86 30	L9ØE2	PSHS LDA		SAVE ZERO COUNTER * INSERT A ZERO INTO
2153 2154	90E6 A7 C0 90E8 35 02		STA PULS		* THE BUFFER RESTORE ZERO COUNTER
2155	30E0 33 02		FULS	A	RESTORE ZERO COUNTER
2156 2157	90EA 4A	* PUT AC L9ØEA	CCA AS DECA	CII ZEROES INTO THE BUFFER	DECREMENT ZERO COUNTER
2158	9ØEB 2A F5	LJULA	BPL	L9ØE2	BRANCH IF NOT DONE
2159 2160	9ØED 39		RTS		
2161	9ØEE 96 D9	L9ØEE	LDA	VD9	* GET THE LEFT DIGIT COUNTER AND PUT
2162 2163	90F0 8D F8 90F2 BD 92 4D		BSR JSR	L90EA L924D	* THAT MANY ZEROES IN THE STRING BUFFER PUT THE DECIMAL POINT IN THE STRING BUFFER
2164	9ØF5 86 F7		LDA	#-9	*DETERMINE HOW MANY FILLER ZEROES BETWEEN THE DECIMAL POINT
2165 2166	90F7 90 47 90F9 8D EF		SUBA BSR	V47 L90EA	*AND SIGNIFICANT DATA. SUBTRACT BASE 10 EXPONENT FROM -9 *(UNNORMALIZATION) AND OUTPUT THAT MANY ZEROES TO BUFFER
2167	90FB 0F 45		CLR	V45	CLEAR THE DECIMAL POINT COUNTER - SUPPRESS THE DECIMAL POINT
2168 2169	90FD 0F D7 90FF BD 92 02	L9ØFF	CLR JSR	VD7 L9202	CLEAR THE COMMA COUNTER - SUPPRESS COMMAS DECODE FPAØ INTO A DECIMAL ASCII STRING
2170	9102 96 D8		LDA	VD8	GET THE RIGHT DIGIT COUNTER
2171 2172	9104 26 02 9106 DE 39		BNE LDU	L9108 VARPTR	BRANCH IF RIGHT DIGIT COUNTER <> Ø RESET BUFFER PTR TO THE DECIMAL POINT IF NO DIGITS TO RIGHT
2173 2174	9108 9B 47	L91Ø8 *	ADDA	V47	*ADD BASE 10 EXPONENT - A POSITIVE ACCA WILL CAUSE THAT MANY
2175		*			*FILLER ZEROES TO BE OUTPUT TO THE RIGHT OF LAST SIGNIFICANT DATA *SIGNIFICANT DATA
2176 2177	910A 16 FF 43	*	LBRA	L9050	INSERT LEADING ASTERISKS, FLOATING DOLLAR SIGN, ETC
2178			THE N	UMERIC OUTPUT FORMAT TO BE EXPONE	
2179 2180	91ØD 96 4F 91ØF 34 Ø2	L91ØD	LDA PSHS	FPØEXP A	* GET EXPONENT OF FPAØ AND * SAVE IT ON THE STACK
2181	9111 27 Ø3		BEQ	L9116	BRANCH IF FPAØ = Ø
2182 2183	9113 BD 91 CD	*	JSR	L91CD	*CONVERT FPAØ INTO A NUMBER WITH 9 SIGNIFICANT *DIGITS TO THE LEFT OF THE DECIMAL POINT
2184	9116 96 D8	L9116	LDA		GET THE RIGHT DIGIT COUNTER
	9118 27 Ø1 911A 4A		DECA	L911B	BRANCH IF NO FORMATTED DIGITS TO THE RIGHT SUBTRACT ONE FOR THE DECIMAL POINT
	911B 9B D9	L911B	ADDA		ADD TO THE LEFT DIGIT COUNTER CLEAR BUFFER BYTE AS TEMPORARY STORAGE LOCATION
2188	911D 7F Ø3 DA 912Ø D6 DA			STRBUF+3 VDA	* GET THE STATUS BYTE FOR A
219Ø 2191	9122 C4 Ø4 9124 26 Ø3			#\$Ø4 L9129	* POST-BYTE FORCE; BRANCH IF * A POST-BYTE FORCE
2192	9126 73 Ø3 DA			STRBUF+3	TOGGLE BUFFER BYTE TO -1 IF NO POST-BYTE FORCE
2193 2194	9129 BB Ø3 DA 912C 8Ø Ø9	L9129		STRBUF+3 #\$Ø9	SUBTRACT 1 IF NO POST BYTE FORCE *SUBTRACT 9 (DUE TO THE CONVERSION TO 9
2195		*			*SIGNIFICANT DIGITS TO LEFT OF DECIMAL POINT)
2196 2197	912E 34 Ø2	*	PSHS	A	=SAVE SHIFT COUNTER ON THE STACK - ACCA CONTAINS THE NUMBER =OF SHIFTS REQUIRED TO ADJUST FPAØ FOR THE NUMBER OF
2198		*			=FORMATTED PLACES TO THE RIGHT OF THE DECIMAL POINT.
2199 2200	9130 2A ØA 9132 34 Ø2	L913Ø	BPL PSHS	L913C A	NO MORE SHIFTS WHEN ACCA >= Ø SAVE SHIFT COUNTER
2201	9134 BD BB 82		JSR	LBB82	DIVIDE FPAØ BY 10 - SHIFT TO RIGHT ONE
	9137 35 Ø2 9139 4C		PULS INCA		RESTORE THE SHIFT COUNTER ADD 1 TO SHIFT COUNTER
2204	913A 2Ø F4	10120	BRA	L9130	CHECK FOR FURTHER SHIFTING (DIVISION)
22Ø5 22Ø6	913C A6 E4 913E 2B Ø1	L913C	LDA BMI	,S L9141	*GET THE INITIAL VALUE OF THE SHIFT COUNTER *AND BRANCH IF SHIFTING HAS TAKEN PLACE
2207	9140 4F	10141	CLRA		RESET ACCA IF NO SHIFTING HAS TAKEN PLACE
22Ø8	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
	9144 4C		INCA		*PLUS ONE AND THE POST-BYTE POSITION, IF USED
	9145 BB Ø3 DA			STRBUF+3	*
	9148 97 45		STA		SAVE DECIMAL POINT COUNTER
	914A ØF D7		CLR		CLEAR COMMA COUNTER - NO COMMAS INSERTED
	914C BD 92 Ø2			L9202	CONVERT FPAØ INTO ASCII DECIMAL STRING
	914F 35 Ø2		PULS		=GET THE INITIAL VALUE OF SHIFT COUNTER AND
2216	9151 BD 92 81		JSR	L9281	=INSERT THAT MANY ZEROES INTO THE BUFFER
	9154 96 D8		LDA		*GET THE RIGHT DIGIT COUNTER AND BRANCH
2218	9156 26 Ø2		BNE	L915A	*IF NOT ZERO
2219	9158 33 5F		LEAU	\$-01,U	MOVE BUFFER POINTER BACK ONE
2220					
2221		* CALCUL	LATE V	ALUE OF EXPONENT AND PUT IN STRIM	NG BUFFER
2222	915A E6 EØ	L915A	LDB	,S+	GET ORIGINAL EXPONENT OF FPAØ
	915C 27 Ø9			L9167	BRANCH IF EXPONENT = Ø
	915E D6 47		LDB		GET BASE 10 EXPONENT
	9160 CB 09			#\$Ø9	ADD 9 FOR 9 SIGNIFICANT DIGIT CONVERSION
	9162 DØ D9		SUBB		SUBTRACT LEFT DIGIT COUNTER
	9164 FØ Ø3 DA			STRBUF+3	ADD ONE TO EXPONENT IF POST-SIGN FORCE
	9167 86 2B	L9167	LDA	#'+'	PLUS SIGN
	9169 5D		TSTB	10165	TEST EXPONENT
	916A 2A Ø3			L916F	BRANCH IF POSITIVE EXPONENT
	916C 86 2D			#'-'	MINUS SIGN
	916E 5Ø 916F A7 41	L916F	NEGB	\$Ø1,U	CONVERT EXPONENT TO POSITIVE NUMBER PUT SIGN OF EXPONENT IN STRING BUFFER
	9171 86 45	LJIOF		#'E'	* PUT AN E (EXPONENTIATION FLAG) IN
	9173 A7 C1			,U++	* BUFFER AND SKIP OVER THE SIGN
	9175 86 2F			#'0'-1	INITIALIZE TENS DIGIT TO ASCII ZERO MINUS ONE
2237	3173 00 21	*CONVERT		RY EXPONENT IN ACCB TO ASCII VALU	
	9177 4C	L9177		THE TABLE	ADD ONE TO TENS DIGIT COUNTER
2239	9178 CØ ØA		SUBB	#10	*SUBTRACT 10 FROM EXPONENT AND ADD ONE TO TENS
2240	917A 24 FB			L9177	* DIGIT IF NO CARRY. TENS DIGIT DONE IF THERE IS A CARRY
2241	917C CB 3A			#'9'+1	ADD ASCII BIAS TO UNITS DIGIT
2242	917E ED C1			,U++	SAVE EXPONENT IN BUFFER
2243	918Ø 6F C4		CLR	,Ü	CLEAR FINAL BYTE IN BUFFER - PRINT TERMINATOR
2244	9182 7E 9Ø 54		JMP	Ĺ9Ø54	INSERT ASTERISK PADDING, FLOATING DOLLAR SIGN, ETC.
2245					
2246		* INSERT	T ASTE	RISK PADDING, FLOATING \$ AND PRE-	-SIGN
2247	9185 8E Ø3 DB	L9185		#STRBUF+4	POINT X TO START OF PRINT ITEM BUFFER
	9188 E6 84		LDB		* GET SIGN BYTE OF ITEM-LIST BUFFER
	918A 34 Ø4		PSHS		* AND SAVE IT ON THE STACK
	918C 86 20			#SPACE	DEFAULT PAD WITH BLANKS
	918E D6 DA		LDB		* GET STATUS BYTE AND CHECK FOR
	9190 C5 20			#\$20	* ASTERISK LEFT PADDING
	9192 35 Ø4		PULS		GET SIGN BYTE AGAIN
	9194 27 Ø8			L919E	BRANCH IF NO PADDING
	9196 86 2A			#'*'	PAD WITH ASTERISK
	9198 C1 2Ø			#SPACE	WAS THE FIRST BYTE A BLANK (POSITIVE)?
	919A 26 Ø2			L919E	NO
	919C 1F 89 919E 34 Ø4	10105	TFR		TRANSFER PAD CHARACTER TO ACCB
		L919E L91AØ			SAVE FIRST CHARACTER ON STACK STORE PAD CHARACTER IN BUFFER
	91AØ A7 8Ø 91A2 E6 84	L9IAW	STA LDB	, X , X	GET NEXT CHARACTER IN BUFFER
	91A4 27 1Ø			,^ L91B6	INSERT A ZERO IF END OF BUFFER
	91A6 C1 45			#'E'	* CHECK FOR AN E AND
	91A8 27 ØC			L91B6	* PUT A ZERO BEFORE IT
	91AA C1 3Ø			#'0'	= REPLACE LEADING ZEROES WITH
	91AC 27 F2			L91AØ	= PAD CHARACTERS
	91AE C1 2C			#','	* REPLACE LEADING COMMAS
2268	91BØ 27 EE		BEQ	L91AØ	* WITH PAD CHARACTERS
2269	91B2 C1 2E		CMPB	#'.'	= CHECK FOR DECIMAL POINT
2270	91B4 26 Ø4		BNE	L91BA	= AND DON T PUT A ZERO BEFORE IT
2271	91B6 86 30	L91B6	LDA		* REPLACE PREVIOUS CHARACTER
2272	91B8 A7 82			, - X	* WITH A ZERO
2273	91BA 96 DA	L91BA		VDA	= GET STATUS BYTE, CHECK
2274	91BC 85 1Ø			#\$10	= FOR FLOATING \$
	91BE 27 Ø4			L91C4	= BRANCH IF NO FLOATING \$
	91CØ C6 24			#'\$' '	* STORE A \$ IN * BUFFER
	91C2 E7 82 91C4 84 Ø4	L91C4	STB	,-X #\$04	* BUFFER CHECK PRE-SIGN FLAG
	9104 84 04	LJ104	PULS		GET SIGN CHARACTER
	9108 26 02				RETURN IF POST-SIGN REQUIRED
	91CA E7 82		STB		STORE FIRST CHARACTER
	91CC 39	L91CC	RTS	,	
2283		*			
2284			RT FPA	Ø INTO A NUMBER OF THE FORM - NN	N,NNN,NNN X 10**M.
2285				T M WILL BE RETURNED IN V47 (BASE	
2286	91CD 34 40	L91CD	PSHS		SAVE BUFFER POINTER
2287	91CF 4F		CLRA		INITIAL EXPONENT OFFSET = Ø
	91DØ 97 47	L91DØ	STA		SAVE EXPONENT OFFSET
2289	91D2 D6 4F		LDB	FPØEXP	GET EXPONENT OF FPAØ
	91D4 C1 8Ø			#\$80	* COMPARE TO EXPONENT OF .5
	91D6 22 11		BHI	L91E9	* AND BRANCH IF FPAØ > = 1.0
2292			. ~	a wu	T. TO
2293		* 11 11		.Ø, MULTIPLY IT BY 1E+Ø9 UNTIL IT	
	91D8 8E BD CØ			#LBDCØ	POINT X TO FP NUMBER (1E+Ø9)
	91DB BD BA CA			LBACA	MULTIPLY FPAØ BY 1E+Ø9
	91DE 96 47		LDA		GET EXPONENT OFFSET
	91EØ 8Ø Ø9			#\$Ø9	SUBTRACT 9 (BECAUSE WE MULTIPLIED BY 1E+09 ABOVE)
	91E2 20 EC	10154		L91DØ	CHECK TO SEE IF > 1.0
2299	91E4 BD BB 82 91E7 ØC 47	L91E4	JSR INC	LBB82	DIVIDE FPAØ BY 10
	91E/ 0C 4/ 91E9 8E BD BB	L91E9	TINC	#LBDBB	INCREMENT EXPONENT OFFSET POINT X TO FP NUMBER (999,999,999)
	91EC BD BC AØ	LJIEŸ		#LBDBB	COMPARE FPAØ TO X
	91EF 2E F3			L91E4	BRANCH IF FPAØ > 999,999,999
		L91F1		#LBDB6	POINT X TO FP NUMBER (99,999,999.9)
2304		-5111	237		

2305 2306 2307 2308 2309 2310 2311	91F4 BD BC AØ 91F7 ZE Ø7 91F9 BD BB 6A 91FC ØA 47 91FE 2Ø F1 9200 35 CØ	BGT JSR DEC BRA	LBCA0 L9200 LBB6A V47 L91F1 U,PC	COMPARE FPAØ TO X RETURN IF 999,999 > FPAØ > 99,999,999.9 MULTIPLY FPAØ BY 1Ø DECREMENT EXPONENT OFFSET KEEP UNNORMALIZING RESTORE BUFFER POINTER AND RETURN
2312 2313 2314 2315 2316 2317 2318	9202 34 40 9204 BD B9 B4 9207 BD BC C8 9207A 35 40	* INTO A DECI L9202 PSHS JSR	LB9B4 LBCC8	SAVE BUFFER POINTER ADD .5 TO FPAØ (ROUND OFF) CONVERT FPAØ TO INTEGER FORMAT RESTORE BUFFER POINTER
2319 2320	920C 8E BE C5 920F C6 80	LDX LDB * BIT 7 SET I * IS NEGATIVE * THE SAME AS	Ø INTO A DECIMAL ASCII STRING #LBEC5 #S8Ø S USED TO INDICATE THAT THE POWEF . WHEN YOU ADD A NEGATIVE MANTI SUBTRACTING A POSITIVE ONE AND E ROUTINE KNOWS THAT A SUBTRACTIC	SSA, IT IS BIT 7 OF ACCB
2326 2327 2328 2329 2330 2331 2332 2333	9211 80 36 9213 96 53 9215 AB 83 9217 97 53 9219 96 52 9218 A9 82 9210 97 52 9217 96 51 9221 A9 81 9223 97 51	L9211 BSR L9213 LDA ADDA STA LDA ADCA STA LDA ADCA	L9249 FPA0+3 \$03,X FPA0+3 FPA0+2 \$02,X FPA0+2 FPA0+1 \$01,X FPA0+1	CHECK FOR COMMA INSERTION * ADD A POWER OF 10 MANTISSA TO FPA0. * IF THE MANTISSA IS NEGATIVE, A SUBTRACTION * WILL BE WHAT REALLY TAKES PLACE. * * * * * * * * * * * * *
2336 2337 2338 2339 2340 2341 2342 2343 2344	9225 96 50 9227 A9 84 9229 97 50 9228 5C 922C 56 922D 59 922E 28 E3 9230 24 03 9232 C0 0B	LDA ADCA STA INCB RORB ROLB BVC BCC SUBB	FPAØ ,X FPAØ L9213 L9235 #10+1	* * * * * * * * * * * * *
2346 2347 2348 2349 2350 2351 2352 2353 2354	9234 50 9235 CB 2F 9237 30 04 9239 1F 98 9238 84 7F 923D A7 C0 923F 53 9240 C4 80 9242 8C BE E9 9245 26 CA 9247 6F C4	LEAX TFR ANDA STA COMB ANDB CMPX	#'0'-1 \$04,X B,A #\$7F ,U+ #\$80 #LBEE9 L9211	* IF ADDING MANTISSA ADD IN ASCII OFFSET MOVE TO NEXT POWER OF 10 MANTISSA SAVE DIGIT IN ACCA MASK OFF ADD/SUBTRACT FLAG (BIT 7) STORE DIGIT IN BUFFER TOGGLE ADD/SUBTRACT FLAG MASK OFF EVERYTHING BUT ADD/SUB FLAG COMPARE TO END OF UNNORMALIZED POWERS OF 10 BRANCH IF NOT DONE PUT A ZERO AT END OF INTEGER
2356 2357 2358 2359 2360 2361 2362 2363 2364	9249 ØA 45 9248 26 Ø9 9240 DF 39 9247 86 2E 9251 A7 CØ 9253 ØF D7	L9249 DEC BNE L924D STU LDA STA CLR	ECIMAL POINT COUNTER AND CHECK FC V45 L9256 VARPTR #'.' ,U+ VD7	OR COMMA INSERTION DECREMENT DECIMAL POINT COUNTER NOT TIME FOR DECIMAL POINT SAVE BUFFER POINTER-POSITION OF THE DECIMAL POINT * STORE A DECIMAL * POINT IN THE OUTPUT BUFFER =-CLEAR COMMA COUNTER - NOW IT WILL TAKE 255 =-DECREMENTS BEFORE ANOTHER COMMA WILL BE INSERTED
2366 2367 2368 2369 2370 2371 2372	9255 39 9258 0A D7 9258 26 08 925A 86 03 925C 97 D7 925E 86 2C 9260 A7 C0 9262 39	BNE LDA STA LDA	VD7 L9262 #\$Ø3 VD7 #','	DECREMENT COMMA COUNTER RETURN IF NOT TIME FOR COMMA * RESET COMMA COUNTER TO 3; THREE * DIGITS BETWEEN COMMAS = PUT A COMMA INTO = THE BUFFER
2376 2377 2378 2379	9263 96 47 9265 8B ØA 9267 97 45 9269 4C 926A 8Ø Ø3 926C 24 FC	L9263 LDA ADDA STA INCA L926A SUBA	DECIMAL POINT AND COMMA COUNTERS V47 #10 V45 #\$03 L926A	GET THE BASE 10 EXPONENT OFFSET * ADD 10 (FPA0 WAS NORMALIZED TO 9 PLACES LEFT * OF DECIMAL POINT) - SAVE IN DECIMAL POINT COUNTER ADD ONE FOR THE DECIMAL POINT = DIVIDE DECIMAL POINT COUNTER BY 3; LEAVE = THE REMAINDER IN ACCA
2381 2382 2383 2384 2385 2386 2387	926E 8B Ø5 927Ø 97 D7 9272 96 DA 9274 84 4Ø 9276 26 Ø2 9278 97 D7 927A 39	ADDA STA LDA ANDA BNE	#\$05 VD7 VDA #\$40 L927A VD7	CONVERT REMAINDER INTO A NUMBER FROM 1-3 SAVE COMMA COUNTER GET STATUS BYTE CHECK FOR COMMA FLAG BRANCH IF COMMA FLAG ACTIVE CLEAR COMMA COUNTER - 255 DIGITS OUTPUT BEFORE A COMMA
2391 2392 2393 2394 2395 2396 2397 2398 2399	927B 34 02 927F 35 02 9281 4A 9282 2B 0A 9284 34 02 9286 86 30 9288 A7 C0 928A 7 C0 928C 26 ED 928E 39	L927B PSHS BSR PULS L9281 DECA BMI PSHS LDA STA LDA	L9249 A L928E	SAVE ZEROES COUNTER CHECK FOR DECIMAL POINT RESTORE ZEROES COUNTER * DECREMENT ZEROES COUNTER AND * RETURN IF < Ø SAVE ZEROES COUNTER * PUT A ZERO INTO * THE BUFFER RESTORE THE ZEROES COUNTER BRANCH IF NOT DONE

```
2401
2402
                                      ***** GRAPHICS PACKAGE ******
24Ø3
24Ø4
                                      * GET THE ADDRESS OF THE ROUTINE WHICH
                                      * WILL CONVERT HOR & VER COORDINATES INTO
* AN ABSOLUTE RAM ADDRESS AND PIXEL MASK
2405
2406
2407
                                      * DEPENDING UPON THE CURRENT PMODE AND
* RETURN THE ADDRESS IN U.
24Ø8
2409
                                      1928F
                                                                                                 JUMP TABLE ADDRESS TO U
2410
         928F CF 92 9C
                                                 I DII
                                                       #1 9290
                                                                                                  GET PMODE VALUE
         9292 96 B6
                                                 LDA
                                                       PMODE
                                                                                                 MUL ACCA X2 - 2 BYTES PER ADDRESS
2412
         9294 48
                                                 ASIA
                                                                                                 GET JUMP ADDRESS
2413
         9295 EE C6
                                                 LDU
                                                       A,U
2414
         9297 39
                                                 RTS
2415
                                      * CONVERT VER COORD (VERBEG) & NOR COORD (HORBEG) INTO * ABSOLUTE SCREEN ADDR IN X AND PIXEL MASK IN ACCA.
2416
2417
         9298 8D F5
929A 6E C4
                                                 BSR
2418
                                      L9298
                                                       L928F
                                                                                                 GO GET JUMP ADDRESS
2419
                                                 JMP
                                                                                                 GO TO IT
2420
                                      * JUMP TABLE
                                                       -- HOR, VER COORD CONVERSION
2421
2422
         929C 92 A6
929E 92 C2
                                      L929C
                                                 FDB
                                                       L92A6
                                                                                                  PMODE Ø
2423
                                      L929E
                                                 FDB
                                                        L92C2
                                                                                                  PMODE 1
2424
         92AØ 92 A6
                                      L92AØ
                                                        L92A6
                                                                                                  PMODE 2
                                                 FDB
2425
         92A2 92 C2
                                      L92A2
                                                 FDB
                                                        L92C2
                                                                                                  PMODE 3
         92A4 92 A6
2426
                                      L92A4
                                                        L92A6
                                                                                                  PMODE 4
                                                 FDB
2427
                                      * HOR, VER COORD CONVERSION ROUTINE FOR 2
2428
2429
                                      * COLOR HIRES GRAPHICS MODES
                                                                                                 SAVE REGISTERS
         92A6 34 44
                                                 PSHS U.B
2430
                                      L92A6
2431
         92A8 D6 B9
                                                        HORBYT
                                                                                                  GET NUMBER BYTES/HOR GRAPHIC ROW
2432
         92AA 96 CØ
                                                 LDA
                                                       VERBEG+1
                                                                                                 GET VERTICAL COORDINATE
         92AC 3D
                                                                                                  CALCULATE VERTICAL BYTE OFFSET
                                                 MUL
                                                 ADDD REGGRP
                                                                                                 ADD IN START OF GRAPHIC PAGE SAVE TEMP VALUE IN X REG
2434
         92AD D3 BA
2435
         92AF 1F Ø1
                                                 TFR
                                                        D,X
                                                                                                 GET HORIZONTAL COORDINATE
* THREE LSRBS EQUALS DIVIDE BY 8 -
2436
         92B1 D6 BE
                                                 I DR
                                                        HORBEG+1
2437
         92B3 54
                                                 LSRB
                                                                                                 * IN THE TWO COLOR MODE THERE ARE

* 8 PIXELS/BYTE
2438
         92B4 54
                                                 LSRB
2439
         92B5 54
                                                 LSRB
2440
         92B6 3A
                                                                                                  ADD HOR BYTE OFFSET
                                                        HORBEG+1
                                                                                                 GET HORIZONTAL COORDINATE
*KEEP ONLY BITS 0-2, WHICH CONTAIN THE NUMBER
*OF THE PIXEL IN THE BYTE
2441
         92B7 96 BE
                                                 LDA
2442
                                                 ANDA
                                                       #$Ø7
2443
                                                                                                 *POINT U TO MASK LOOKUP TABLE

*GET PIXEL MASK - THE MASK WILL HAVE ONE BIT SET WHICH

*CORRESPONDS TO THE PIXEL SELECTED.
         92BB CE 92 DD
                                                 LDU
                                                        #L92DD
2445
         92BE A6 C6
                                                 LDA
                                                       A,U
2446
2447
         92CØ 35 C4
                                                 PULS B,U,PC
                                                                                                  RESTORE REGISTERS
2448
2449
                                      * HOR, VER COORDINATE CONVERSION ROUTINE
                                      * FOR 4 COLOR HI RES GRAPHICS MODES
2450
                                                                                                 SAVE REGISTERS
GET NUMBER BYTES/HOR GRAPHIC ROW
GET VERTICAL COORDINATE
CALCULATE VERTICAL OFFSET
         92C2 34 44
                                                 PSHS U,B
                                                        HORBYT
2452
         92C4 D6 B9
                                                 LDB
2453
         92C6 96 CØ
                                                        VERBEG+1
                                                 LDA
         92C8 3D
2454
                                                 MUL
         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
                                                        HORBEG+1
                                                                                                  GET HORIZONTAL COORDINATE
                                                 LDB
                                                                                                 TWO LSRBS = DIVIDE BY 4; IN THE 4
COLOR MODE THERE ARE 4 PIXELS/BYTE
2458
         92CF 54
                                                 LSRR
2459
         92DØ 54
                                                 LSRB
246Ø
2461
         92D1 3A
                                                                                                 ADD HORIZONTAL BYTE OFFSET
GET HORIZONTAL COORDINATE
                                                 ABX
                                                        HORBEG+1
         92D2 96 BE
                                                 LDA
                                                                                                  *KEEP ONLY BITS Ø,1 WHICH CONTAIN THE NUMBER OF THE PIXEL
*TO CHANGE (4 COLOR)
         92D4 84 Ø3
                                                 ANDA
                                                       #$Ø3
2462
2463
                                                       #L92E5
2464
         92D6 CE 92 E5
                                                 LDU
                                                                                                  POINT U TO MASK LOOKUP TABLE
                                                                                                  GET THE MASK FOR THE PROPER PIXEL
2465
         92D9 A6 C6
                                                 I DA
                                                 PULS B,U,PC
                                                                                                  RESTORE REGISTERS AND RETURN
2466
         92DB 35 C4
2467
2468
                                      * 2 COLOR MODE PIXEL MASKS
         92DD 80 40 20 10 08 04 L92DD
92E3 02 01
                                                FCB $80,$40,$20,$10,$08,$04
FCB $02,$01
2469
2470
2471
                                       4 COLOR MODE PIXEL MASKS
2472
2473
         92E5 CØ 3Ø ØC Ø3
                                                 FCB
                                                       $CØ,$3Ø,$ØC,$Ø3
                                      L92E5
2474
2475
                                      * MOVE X REG DOWN ONE GRAPHIC ROW
2476
         92E9 D6 B9
                                      L92E9
                                                 LDB
                                                       HORBYT
                                                                                                 GET NUMBER BYTES/HOR ROW
                                                                                                 ADD TO ABSOLUTE SCREEN POSITION
2477
         92EB 3A
                                                 ABX
2478
2479
         92EC 39
                                      * ENTER W/ABSOLUTE SCREEN POSITION IN X, THE PIXEL * MASK IN ACCA - ADJUST X AND ACCA TO THE NEXT * PIXEL TO THE RIGHT IN THE TWO COLOR MODE.
2480
2481
2482
2483
         92ED 44
                                                                                                  SHIFT ONE BIT TO RIGHT
                                                 LSRA
                                      L92ED
2484
         92EE 24 Ø3
                                                                                                 BRANCH IF IN SAME BYTE
                                                       L92F3
                                                                                                 IF YOU HAVE MOVED TO NEXT BYTE, SET BIT 7 IN ACCA
                                                 RORA
2485
         92FØ 46
2486
         92F1 3Ø Ø1
                                                 LEAX $Ø1,X
                                                                                                  AND ADD ONE TO X.
2487
         92F3 39
                                      192F3
                                                 RTS
2488
2489
2490
                                      * MOVE ABSOLUTE SCREEN ADDRESS OF CURRENT
2491
                                      * HOR, VER COORD ONE TO RIGHT AND ADJUST
* THE PIXEL MASK FOR THE 4 COLOR MODE
2492
         92F4 44
92F5 24 F6
                                                                                                 SHIFT MASK ONE BIT TO RIGHT
SHIFT RIGHT AGAIN IF SAME BYTE
2493
                                      L92F4
                                                 LSRA
                                                 BCC L92ED
2494
                                                                                                 SET PIXEL #3 IF NEW BYTE
ADD ONE TO ABS SCREEN POSITION
2495
         92F7 86 CØ
                                                        #$CØ
         92F9 3Ø Ø1
                                                 LEAX $01.X
2496
```

2497	92FB 39		RTS		
2498	32FB 33	*	KIS		
2499				EXPRESSIONS - PUT THE FIRST	
2500 2501				COORD) IN HORBEG AND THE COORD) IN VERBEG.	
2502	92FC BD B7 34	L92FC		LB734	*EVALUATE TWO EXPRESSIONS FROM THE BASIC LINE -
25Ø3 25Ø4	92FF 10 8E 00 BD	*	ı ny	#HORBEG	*RETURN WITH THE 1ST VALUE IN BINVAL AND THE 2ND IN ACCB POINT Y TO TEMP STORAGE LOC
2505	93Ø3 C1 CØ		CMPB		IS VERT COORD > 191?
2506	9305 25 02			L93Ø9	NO
25Ø7 25Ø8	9307 C6 BF 9309 4F		CLRA	#191	FORCE VER COORD TO 191 HIGH ORDER BYTE OF VER COORD
2509	93ØA ED 22	9	STD	\$02,Y	SAVE VERTICAL COORDINATE
251Ø 2511	930C DC 2B 930E 10 83 01 00		LDD CMPD	BINVAL #256	GET RAW HORIZONTAL COORDINATE IS IT WITHIN RANGE?
2512	9312 25 Ø3			L9317	YES YES
2513	9314 CC ØØ FF		LDD		FORCE IT TO 255 IF NOT IN RANGE
2514 2515	9317 ED A4 9319 39		STD RTS	, Ү	SAVE HORIZONTAL COORDINATE
2516					
2517 2518				ORIZONTAL AND VERTICAL COORDINATE ALIZED VALUES IN (HORBEG,VERBEG)	S FOR THE PROPER PHODE
	> 931A BD 92 FC	L931A	JSR	L92FC	GO GET HOR & VER COORDINATES
	931D CE ØØ BD			#HORBEG	POINT U TO HOR & VER COORDS
	932Ø 96 B6 9322 81 Ø2			PMODE #\$Ø2	GET PHODE CHECK MODE
2523	9324 24 Ø6	E	BCC	L932C	BRANCH IF > 1
	9326 EC 42 9328 44		LDD LSRA	\$Ø2,U	GET THE VERT COORD *DIVIDE ACCD BY TWO SINCE
	9329 56		RORB		*PMODES Ø&1 HAVE ONLY 96 VERT BLOCKS
	932A ED 42			\$Ø2,U	SAVE NEW VERT COORD
	932C 96 B6 932E 81 Ø4			PMODE #\$Ø4	GET PMODE CHECK PMODE
2530	9330 24 06	E	BCC	L9338	BRANCH IF PMODE = 4
	9332 EC C4 9334 44		LDD LSRA	,U	GET HOR COORD * DIVIDE HORIZONTAL COORDINATE
	9335 56		RORB		* BY 2-PMODES Ø,1,2,3 HAVE ONLY 128 HOR BLOCKS
	9336 ED C4		STD	, U	SAVE NEW HOR COORD
2535 2536	9338 39	L9338 F	RTS		
2537		* PPOINT			
	> 9339 BD 93 B2 > 933C BD 93 1D	PPOINT 3		L93B2 L931D	EVAL TWO EXPRESSIONS, RETURN VALUES IN (HORBEG, VERBEG) NORMALIZE EXPRESSIONS FOR PROPER PMODE
	933F BD 92 98			L9298	CONVERT COORDS INTO ABS SCREEN POSITION & PIXEL MASK
	9342 A4 84		ANDA		AND PIXEL MASK WITH CONTENTS OF SCREEN
	9344 D6 B6 9346 56		RORB	PMODE	GET CURRENT PMODE SHIFT RIGHT
2544	9347 24 12	E	BCC	L935B	BRANCH IF PMODE Ø,2,4 (2 COLOR)
2545 2546	9349 81 Ø4			#\$Ø4 L9351	* IS THE ON PIXEL IN THE 2 RIGHTMOST BITS?
2546	934B 25 Ø4 934D 46		RORA	L9351	BRANCH IF IT IS =SHIFT RIGHT
	934E 46		RORA		=ONE PIXEL
	934F 2Ø F8 9351 4C		BRA INCA	L9349	KEEP CHECKING ADD 1 TO COLOR - BASIC USES 1 TO 4, NOT Ø TO 3
2551	9352 48	į.	ASLA		TIMES 2
2552 2553	9353 9B C1 9355 44		ADDA LSRA	CSSVAL	ADD COLOR SET (Ø OR 8) DIVIDE ACCB BY TWO - COLORS RANGE FROM Ø - 8
2554	9356 1F 89		TFR	A,B	*TRANSFER COLOR INFO INTO ACCB
2555	9358 7E B4 F3	j	JMP	LB4F3	*CONVERT ACCB TO FP NUMBER
2556 2557	935B 4D 935C 27 F8		TSTA RFO	L9356	IS 2 COLOR PIXEL ON?
2558	935E 4F		CLRA	25000	* FORCE THE TWO-COLOR ON VALUE TO BE
2559 2560	935F 20 F0	E	BRA	L9351	* ONE OR FIVE (DEPENDING ON CSS).
2561		* PSET			
2562			LDA		PSET FLAG
2563 2564	9363 20 01	E	BKA	L9366	
2565		* PRESET			
2566 2567	9365 4F 9366 97 C2	PRESET (SETFLG	PRESET FLAG STORE FLAG Ø = PRESET, 1 = PSET
2568	9368 BD B2 6A			LB26A	SYNTAX CHECK FOR (
2569 >	936B BD 93 1A			L931A	EVAL HOR & VER COORDS AND NORMALIZE
257Ø 2571	936E BD 95 81	*	U2K	L9581	*EVALUATE COLOR - RETURN IN WCOLOR; ALLCOL *WILL BE ONE BYTE WITH ALL PIXELS SET TO THAT COLOR
2572	9371 BD B2 67			LB267	SYNTAX CHECK FOR)
2573 2574	9374 BD 92 98	*	JSR	L9298	*CALCULATE THE ABSOLUTE ADDRESS OF THE *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 MA	SK IN ACCA) TO THE COLOR
2579		* IN ALLC	0L.	SET CHGFLG \Leftrightarrow Ø IF THE PIXEL COL	OR IS CHANGED.
258Ø 2581	9377 E6 84 9379 34 Ø4		LDB PSHS		GET BYTE FROM THE SCREEN SAVE IT ON STACK
2582	937B 1F 89		TFR		PUT PIXEL MASK IN ACCB
2583	937D 43	(COMA	,	INVERT PIXEL MASK
2584 2585	937E A4 84	F	ANDA	, X	* AND WITH SCREEN DATA - KEEP ALL PIXELS * EXCEPT THE ONE TO MODIFY
2586	938Ø D4 B5			ALLCOL	CONVERT PIXEL IN THE PIXEL MASK TO THE PROPER COLOR
2587 2588	9382 34 Ø4 9384 AA EØ		PSHS ORA		SAVE IT ON STACK OR IT INTO THE REST OF THE PIXELS
2588 2589	9386 A7 84		STA		PUT IT ON SCREEN
2590	9388 AØ EØ	9	SUBA	,S+	SUBTRACT OLD BYTE FROM NEW BYTE; ACCA=0 IF NEW BYTE = OLD BYTE
2591 2592	938A 9A DB 938C 97 DB			CHGFLG CHGFLG	OR DIFFERENCE WITH CHANGE FLAG SAVE IT - CHGFLG WILL BE = Ø IF THE GRAPHIC BYTE IS UNCHANGED
		•		-	The state of the s

2593	938E 39		RTS		
2594		* EVALUA		SETS OF COORDINATES SEPARATED B	Y A MINUS
2595				ST SET OF COORDS AT (HORBEG, VERB	
2596				ID, VEREND). IF NOTHING BEFORE MI	
2597				F) AT (HORBEG, VERBEG)	100 014H, 101
2598	938F 9E C7		LDX F		GET LAST HORIZ END POINT
2599	9391 9F BD	25501	STX F		PUT IN START POINT STORAGE LOC
2600	9393 9E C9		LDX \		GET LAST VERT END POINT
			STX \		
2601	9395 9F BF				PUT IN START POINT VERT STORAGE LOC
2602	9397 81 AC		CMPA #		TOKEN FOR MINUS SIGN
2603	9399 27 Ø3		BEQ L		BRANCH IF NO STARTING COORDINATES GIVEN
	939B BD 93 B2		JSR L		GO GET STARTING COORDINATES
2605		L939E	LDB #		TOKEN FOR MINUS SIGN
2606	93AØ BD B2 6F		JSR L		GO DO A SYNTAX CHECK
2607	93A3 BD B2 6A		JSR L		SYNTAX CHECK FOR A (
2608	93A6 BD B7 34		JSR L		EVALUATE 2 EXPRESSIONS
2609	93A9 10 8E 00 C3		LDY #	#HOREND	TEMP STORAGE LOCATION FOR END COORDINATES OF LINE COMMAND
2610	93AD BD 93 Ø3		JSR L	_9303	GET END POINT COORDS
2611	93BØ 2Ø Ø6		BRA L	_93B8	CHECK SYNTAX FOR)
2612		L93B2	JSR L		SYNTAX CHECK FOR (
2613	93B5 BD 92 FC		JSR L		EVALUATE 2 EXPRESSIONS
2614	93B8 7E B2 67	L93B8		_B267	SYNTAX CHECK FOR) AND RETURN
2615	3030 72 32 07	23000		-5207	OTHER PORT OF THE REPORT
2616		* LINE			
2617	93BB 81 89	LINE	CMPA #	4¢00	INPUT TOKEN
		LINE			
	93BD 10 27 F5 FF		LBEQ L		GO DO LINE INPUT COMMAND
2619	93C1 81 28		CMPA #		CHECK FOR (
2620	93C3 27 Ø9		BEQ L		GO LOOK FOR START AND END POINTS
2621	93C5 81 AC		CMPA #		CHECK TOKEN FOR MINUS SIGN
2622	93C7 27 Ø5		BEQ L		GO GET START AND END POINTS
2623	93C9 C6 4Ø		LDB #		CHECK FOR @ SIGN
2624	93CB BD B2 6F		JSR L	_B26F	DO A SYNTAX CHECK
2625 >	93CE BD 93 8F	L93CE	JSR L	_938F	GET STARTING AND ENDING COORDINATES
2626	93D1 9E C3		LDX F	HOREND	GET ENDING HORIZ COORDINATE
2627	93D3 9F C7		STX F		PUT IN LAST USED HOR END POINT
2628	93D5 9E C5		LDX \		GET ENDING VER COORD
2629	93D7 9F C9		STX \		PUT IN LAST USED VER END POINT
2630	93D9 BD B2 6D			SYNCOMMA	SYNTAX CHECK FOR COMMA
	93DC 81 BE		CMPA #		PRESET TOKEN?
2631					
2632	93DE 27 Ø9		BEQ L		YES
2633	93EØ 81 BD		CMPA #		PSET TOKEN?
	93E2 10 26 1E 91		LBNE L		SYNTAX ERROR IF NOT PSET OR PRESET
	93E6 C6 Ø1		LDB #		PSET FLAG
2636	93E8 86	L93E8	FCB S	SKP1LD	SKIP ONE BYTE, LOAD ACCA WITH \$5F
2637	93E9 5F	L93E9	CLRB		PRESET FLAG
2638	93EA 34 Ø4		PSHS E	3	SAVE PSET/PRESET FLAG
2639	93EC 9D 9F		JSR G	GETNCH	GET ANOTHER CHAR
	93EE BD 94 20		JSR L		NORMALIZE START/END COORDS
2641	93F1 35 Ø4		PULS E		GET PSET/PRESET FLAG
2642	93F3 D7 C2		STB S		SAVE IT
	93F5 BD 95 9A		JSR L		SET ACTIVE COLOR BYTE
2644	93F8 9D A5		JSR 0		GET ANOTHER CHARACTER
2645	93FA 10 27 00 A3		LBEQ L		BRANCH IF NO BOX TO BE DRAWN
2646	93FE BD B2 6D			SYNCOMMA	SYNTAX CHECK FOR COMMA
2647	94Ø1 C6 42		LDB #		BOX ?
2648	94Ø3 BD B2 6F		JSR L		GO DO A SYNTAX CHECK FOR A B
2649	9406 26 21		BNE L	_9429	FOUND A B AND SOMETHING FOLLOWS
2650	94Ø8 8D 3A		BSR L	_9444	DRAW A HORIZ LINE
2651	94ØA 8D 62		BSR L	_946E	DRAW A VERTICAL LINE
2652	94ØC 9E BD		LDX F	HORBEG	GET HOR START COORD
2653	94ØE 34 1Ø		PSHS >	(SAVE IT ON STACK
2654	941Ø 9E C3		LDX F		* GET HORIZONTAL END COORDINATE AND
2655	9412 9F BD		STX F		* PUT THEM IN HORIZONTAL START COORDINATE
2656	9414 8D 58		BSR L		DRAW VERTICAL LINE
2657	9416 35 10		PULS)		GET THE PREVIOUS HORIZONTAL START COORDINATE
2658	9418 9F BD		STX F		RESTORE IT
2659	941A 9E C5		LDX \		GET VER END COORD
2660	941C 9F BF		STX \		PUT INTO START
2661	941C 9F BF 941E 20 24		BRA L		DRAW HORIZ LINE
	J-T1L CD C4	* NODMAI			
2662	9420 BD 93 1D			RT COORDS IN (HORBEG, VERBEG) &	
2663		L942Ø	JSR L		NORMALIZE COORDS IN (HORBEG, VERBEG)
	9423 CE ØØ C3			#HOREND	- HODMALIZE COORDS IN (HODEND MESSAGE)
	9426 7E 93 2Ø		JMP L		= NORMALIZE COORDS IN (HOREND, VEREND)
	9429 C6 46	L9429	LDB #		*
	942B BD B2 6F		JSR L		*GO DO A SYNTAX CHECK FOR AN F
	942E 2Ø Ø4		BRA L		FILL THE BOX
		L943Ø	LEAX \$		MOVE VER COORD UP ONE
2670	9432 9F BF	L9432	STX \	/ERBEG	SAVE NEW VERTICAL START COORDINATE
2671					
2672		* DRAW A	A SERIES	OF HORIZONTAL LINES FROM VER S	TART TO VER END
	9434 BD 94 44	L9434	JSR L		DRAW A HORIZ LINE
	9437 9E BF		LDX \		GET START VER COORD
	9439 9C C5		CMPX \		COMPARE TO END VER COORD
	943B 27 Ø6		BEQ L		RETURN IF EQUAL
	943D 24 F1		BCC L		BRANCH IF START HOR > END HOR
	943F 3Ø Ø1		LEAX \$		MOVE HOR COORD DOWN ONE
2679	9441 20 EF	1011	BRA L	_943Z	KEEP DRAWING LINES
2680	9443 39	L9443	RTS		WASTED BYTE - SHOULD USE L946B INSTEAD
2681				NTAL LINE FROM HOREND TO HORBEG	
2682				VERBEG; COLOR IN ALLCOL	
	9444 9E BD	L9444	LDX F		GET STARTING COORDS
	9446 34 10		PSHS >	(SAVE EM
				_971D	GET ABSOLUTE VALUE OF HOREND - HORBEG (HORIZONTAL COORDINATE)
2003	9448 BD 97 1D		JOK I		
			BCC I		BRANCH IF END > START
2686	9448 BD 97 1D 944B 24 Ø4		BCC L	_9451	
2686 2687	9448 BD 97 1D			_9451 HOREND	BRANCH IF END > START * TRANSFER END COORD TO START *

89	9451 1F Ø2	L9451	TFR D,Y	SAVE DIFFERENCE IN Y
9Ø 91	9453 31 21 9455 BD 92 98		LEAY \$01,Y JSR L9298	ADD ONE TO DIFFERENCE - TURN ON STARTING & ENDING COORD GET ABS SCREEN POS TO X AND PIXEL MASK TO ACCA
92	9458 35 40		PULS U	GET START COORDS
	945A DF BD 945C 8D 36		STU HORBEG BSR L9494	RESTORE THEM POINT U TO ROUTINE TO MOVE PIXEL POINTERS TO RIGHT
95	945E 97 D7	L945E	STA VD7	SAVE PIXEL MASK
96 97	9460 BD 93 77 9463 96 D7		JSR L9377 LDA VD7	TURN ON PIXEL GET OLD PIXEL MASK
98	9465 AD C4		JSR ,U	MOVE TO NEXT ONE TO RIGHT
99 ØØ	9467 31 3F 9469 26 F3		LEAY \$-01,Y BNE L945E	DEC COUNTER LOOP IF NOT DONE
Ø1	946B 39	L946B	RTS	LOUI II NOI DONE
Ø2 Ø3	946C 35 Ø6	L946C	PULS A,B	CLEAN UP STACK
Ø4			A VERTICAL LINE FROM VEREND TO	
	946E DC BF 947Ø 34 Ø6	L946E	LDD VERBEG PSHS B,A	GET END COORDS SAVE THEM
Ø7	9472 BD 97 10		JSR L9710	CALCULATE ABSOLUTE VALUE OF VEREND-VERBEG
	9475 24 Ø4 9477 9E C5		BCC L947B LDX VEREND	BRANCH IF END COORD > START COORD
	9479 9F BF		STX VERBEG	*SWITCH VER COORDS IF END COORD IS TO RIGHT OF START
	947B 1F Ø2	L947B	TFR D,Y	LENGTH OF LINE TO Y
	947D 31 21 947F BD 92 98		LEAY \$01,Y JSR L9298	SET BOTH START AND END COORDS GET ABSOLUTE SCREEN POS TO X, MASK TO ACCA
	9482 35 40		PULS U	GET END COORD
15 16	9484 DF BF 9486 8D 15		STU VERBEG BSR L949D	RESTORE THEM POINT U TO ROUTINE TO MOVE DOWN ONE ROW
17	9488 20 D4		BRA L945E	DRAW VERT LINE
18 19		* JUMP	TABLE OF ADDRESSES OF ROUTINES	WHICH WILL MOVE THE
20	0404 02 FP	* ABSOL	UTE SCREEN ADDRESS POINTER ONE	PIXEL TO THE RIGHT.
	948A 92 ED 948C 92 F4	L948A L948C	FDB L92ED FDB L92F4	PMODE Ø PMODE 1
23	948E 92 ED	L948E	FDB L92ED	PMODE 2
	949Ø 92 F4 9492 92 ED	L949Ø L9492	FDB L92F4 FDB L92ED	PMODE 3 PMODE 4
26				
27 28	9494 CE 94 8A	* POINT	U TO ROUTINE WHICH WILL MOVE P LDU #L948A	IXEL ONE TO RIGHT POINT TO JUMP TABLE
29	9497 D6 B6		LDB PMODE	GET PMODE VALUE
3Ø 31	9499 58 949A EE C5		ASLB LDU B,U	X2 GET JUMP ADDRESS
32	949C 39		RTS	
33 34	949D CE 92 E9 94AØ 39	L949D	LDU #L92E9 RTS	POINT U TO ROUTINE TO MOVE ABS POS DOWN ONE ROW
35				
36 37	94A1 10 8E 95 0D		LINE FROM (HORBEG.VERBEG) TO (H LDY #L95ØD	OREND, VEREND) POINT Y TO INCR VERBEG
38	94A5 BD 97 10		JSR L9710	CALCULATE VEREND - VERBEG (VERTICAL DIFFERENCE)
	94A8 10 27 FF 98 94AC 24 04		LBEQ L9444 BCC L94B2	DRAW A HORIZONTAL LINE IF DELTA V = Ø BRANCH IF VER END COORD > VER START COORD
41	94AE 10 8E 95 1B		LDY #L951B	POINT Y TO DECR VER COORD (VERBEG)
	94B2 34 Ø6 94B4 CE 95 Ø6	L94B2	PSHS B,A LDU #L9506	SAVE DELTA V POINT U TO INCR HOR COORD
44	94B7 BD 97 1D		JSR L971D	CALCULATE HOREND-HORBEG (HOR DIFFERENCE)
	94BA 27 BØ 94BC 24 Ø3		BEQ L946C BCC L94C1	DRAW A VERTICAL LINE IF DELTA H = Ø BRANCH IF HOR END COORD > HOR START COORD
47	94BE CE 95 14		LDU #L9514	POINT U TO DECR HOR COORD
	94C1 10 A3 E4	L94C1	CMPD ,S	COMPARE DELTA H TO DELTA V
49 5Ø	94C4 35 1Ø 94C6 24 Ø4		PULS X BCC L94CC	PUT DELTA V IN X BRANCH IF DELTA H > DELTA V
51	94C8 1E 32		EXG U,Y	SWAP CHANGE HOR & CHANGE VER ADDRESS
	94CA 1E Ø1 94CC 34 46	L94CC	EXG D,X PSHS U,B,A	EXCHANGE DELTA HOR & DELTA VER *SAVE THE LARGER OF DELTA V, DELTA H
54		*		*AND THE INCREMENT/DECREMENT ADDRESS
	94CE 34 Ø6 94DØ 44		PSHS B,A LSRA	SAVE WHICHEVER IS LARGER OF DELTA V, DELTA H *
57	94D1 56		RORB	* DIVIDE BY 2, SHIFT ACCD RIGHT ONE BIT
58 50	94D2 25 Ø9 94D4 11 83 95 ØF		BLO L94DD CMPU #L95ØD+1	BRANCH IF ODD NUMBER SEE IF INCR OR DECR
6Ø	94D4 11 83 95 ØE 94D8 25 Ø3		BLO L94DD	BRANCH IF INCR
	94DA 83 ØØ Ø1	10400	SUBD #1	SUBTRACT 1 IF DECREMENT
62 63	94DD 34 16	L94UU	PSHS X,B,A	*SAVE SMALLEST DELTA (X) AND INITIAL MINOR COORDINATE *INCREMENT COUNTER WHICH IS 1/2 OF LARGEST DELTA
64 65	94DF BD 92 8F		JSR L928F	POINT U TO PROPER COORDINATE TO SCREEN CONVERSION ROUT
66		* DRAW	THE LINE HERE - AT THIS POINT T	HE STACK HAS THE DRAW DATA ON IT
67 68		* 01 1 0	=MINOR COORDINATE INCREMENT COU	NTER
69		* 2 3,9	=ASSOLUTE VALUE OF THE SMALLEST	DELTA COORDINATE
7Ø			=ABSOLUTE VALUE OF THE LARGEST	
71 72		* 6 /,5	=LARGEST COORDINATE COUNTER (HO LOOP. INITIALLY SET TO ABSOLUT	
73			=ADDRESS OF THE ROUTINE WHICH W	
74 75		*	THE LARGEST DELTA COORDINATE	
76	94E2 AD C4	L94E2	JSR ,U	CONVERT (X,Y) COORDINATES TO ABSOLUTE SCREEN ADDRESS
77	94E4 BD 93 77 94E7 AE 66		JSR L9377 LDX \$06,S	TURN ON A PIXEL GET DISTANCE COUNTER
	94E9 27 17		BEQ L9502	BRANCH IF LINE IS COMPLETELY DRAWN
78	3 123 27 27		LEAX \$-Ø1,X	DECR ONE
78 79 8Ø	94EB 3Ø 1F			CAVE IT
78 79 8Ø 81			STX \$06,S JSR [\$08,S]	SAVE IT INCR/DECR COORDINATE-WHICH HAS THE LARGEST DELTA
78 79 8Ø 81 82 83	94EB 3Ø 1F 94ED AF 66		STX \$06,S	

2785	94F6 ED E4		STD	,\$	SAVE NEW MINOR COORDINATE INCREMENT COUNTER
2786 2787	94F8 A3 64 94FA 25 E6			\$Ø4,S L94E2	*SUBTR OUT THE LARGEST DIFFERENCE AND *BRANCH IF RESULT NOT > LARGEST DIFFERENCE
2788	94FC ED E4		STD		IF >=, THEN STORE NEW MIHOR COORDINATE INCREMENT
2789	94FE AD A4		JSR	,Υ	INCREMENT/DECREMENT COORDINATE WHICH HAS THE SMALLEST DELTA
2790	9500 20 E0	10500		L94E2	KEEP GOING
2791 2792	9502 35 10 9504 35 F6	L95Ø2	PULS PULS	A,B,X,Y,U,PC	*CLEAN UP THE STACK AND RETURN
2793 2794		* * TUECE	DOUTT	NES ARE USED TO INCREMENT OR DECR	DEMENT THE
2795				& VERTICAL COORDINATES. THEY NEED	
2796		* IN THI	S ORD	ER (INCR,INCR,DECR,DECR).	
2797 2798		* INCR H	IORBEG	(HOR COORD)	
2799	9506 9E BD			HORBEG	GET COORDINATE
2800	9508 30 01			\$01,X	ADD ONE SAVE COORDINATE
28Ø1 28Ø2	950A 9F BD 950C 39		RTS	HORBEG	SAVE COURDINATE
28Ø3		* INCR V		(VER COORD)	
28Ø4	95ØD 9E BF	L95ØD		VERBEG	GET COORDINATE
28Ø5 28Ø6	950F 30 01 9511 9F BF			\$Ø1,X VERBEG	ADD ONE SAVE COORDINATE
2807	9513 39		RTS	TENSEG	one occupant
28Ø8	0544 05 00			(HOR COORD)	05T 0000DYNATE
28Ø9 281Ø	9514 9E BD 9516 30 1F	L9514		HORBEG \$-Ø1,X	GET COORDINATE SUBTRACT ONE
2811	9518 9F BD			HORBEG	SAVE COORDINATE
2812	951A 39		RTS	(UED 000DD)	
2813 2814	951B 9E BF			(VER COORD) VERBEG	GET COORDINATE
2815	951D 30 1F	25510		\$-01,X	SUBTRACT ONE
2816	951F 9F BF			VERBEG	SAVE COORDINATE
2817 2818	9521 39	*	RTS		
2819		* GET MA	XIMUM	VALUE OF HOR/VER COORDINATES	
2820				FOR PROPER PMODE. RETURN VALUES	
2821 2822	9522 CE ØØ D3	* HOR = L9522		VER = VD5 #VD3	POINT U TO TEMP STORAGE AREA (VD3)
	9525 8E ØØ FF	LJJLL	LDX		MAXIMUM VALUE HORIZONTAL COORD (255)
2824	9528 AF C4			, U	SAVE IT
2825 2826	952A 8E ØØ BF 952D AF 42		LDX STX	#191 \$02,U	MAXIMUM VALUE VERTICAL COORD (191) SAVE IT
2827	952F 7E 93 2Ø			L932Ø	GO CONVERT THEM TO PROPER PMODE
2828		+ 0010			
2829 283Ø	9532 27 ØE	* PCLS PCLS	BEO	L9542	CLEAR TO BACKGROUND COLOR IF NO ARGUMENT
	9534 8D 24	. 020		L955A	EVALUATE EXPRESSION, CONVERT TO PROPER COLOR CODE
	9536 86 55	L9536		#\$55	CONSIDER EACH BYTE AS 4 GROUPS OF 2 BIT SUB-NIBBLES
2833 2834	9538 3D 9539 9E BA		MUL LDX	BEGGRP	MULT BY COLOR GET STARTING ADOR
2835	953B E7 8Ø	L953B		, X+	SET BYTE TO PROPER COLOR
2836	953D 9C B7			ENDGRP	AT END OF GRAPHIC PAGE?
2837 2838	953F 26 FA 9541 39		BNE RTS	L953B	NO .
2839	9542 D6 B3	L9542		BAKCOL	GET BACKGROUND COLOR
2840	9544 20 F0		BRA	L9536	
2841 2842		* COLOR			
2843	9546 81 2C	COLOR	CMPA	#','	*CHECK FOR COMMA AND
2844	9548 27 Ø8			L9552	*BRANCH IF FOREGROUND COLOR ARGUMENT MISSING
2845 2846	954A 8D ØE 954C D7 B2			L955A FORCOL	EVALUATE FIRST ARGUMENT STORE IN FOREGROUND LOCATION
2847	954E 9D A5			GETCCH	GET NEXT INPUT CHARACTER
2848	9550 27 07			L9559	RETURN IF NONE
2849 285Ø	9552 BD B2 6D 9555 8D Ø3	L9552	JSR BSR	SYNCOMMA L955A	SYNTAX CHECK FOR COMMMA EVALUATE LAST ARGUMENT
2851	9557 D7 B3		STB	BAKCOL	STORE IN BACKGROUND COLOR
2852	9559 39	L9559	RTS		
2853 2854		* FVALIIA	TF AN	EXPRESSION AND CONVERT IT TO A F	PROPER COLOR CODE
2855				N THE PMODE AND CSS; ILLEGAL FUNC	
2856				R VALUE IN ACCB; CSS VALUE IN ACC	
2857 2858	955A BD B7 ØB 955D C1 Ø9	L955A L955D		EVALEXPB #\$Ø9	EVALUATE EXPRESSION ONLY ALLOW Ø-8
2859	955F 10 24 1E E7	L3330		LB44A	ILLEGAL FUNCTION CALL IF BAD COLOR
2860	9563 4F		CLRA		VDG CSS VALUE FOR FIRST COLOR SET
2861 2862	9564 C1 Ø5 9566 25 Ø4			#\$Ø5 L956C	FIRST OR SECOND COLOR SET? BRANCH IF FIRST SET
2863	9568 86 Ø8			#\$Ø8	VDG CSS VALUE FOR SECOND COLOR SET
2864	956A CØ Ø4			#\$04	MAKE 5-8 BECOME 1-4
2865 2866	956C 34 Ø2 956E 96 B6	L956C	PSHS	A PMODE	SAVE VDG CSS VALUE ON THE STACK GET PMODE
2867	9570 46		RORA		4 COLOR OR 2 COLOR
2868	9571 24 Ø8			L957B	2 COLOR
2869 287Ø	9573 5D 9574 26 Ø2		TSTB BNF	L9578	WAS COLOR = Ø NO
2871	9576 C6 Ø4	L9576		#\$04	IF SO, MAKE IT 4
2872	9578 5A	L9578	DECB	A DC	CONVERT 1-4 TO Ø-3
2873 2874	9579 35 82 957B 56	L9579 L957B	PULS RORB	A,PC	PUT VDG CSS VALUE IN ACCA AND RETURN CHECK ONLY THE LSB OF COLOR IF IN 2 COLOR MODE
2875	957C 25 F8	23370	BLO	L9576	BRANCH IF ODD - FORCE ACCB TO 3
2876	957E 5F		CLRB		FORCE ACCB = Ø IF EVEN
2877 2878	957F 2Ø F8		DКА	L9579	RETURN
2879				RENT ACTIVE COLOR AND ALL PIXEL E	SYTE
2880		* TO FOR	EGROU	ND/BACKGROUND COLOR DEPENDING ON	

2881 2882 2883 > 9581 BD 95 9A 2884 9584 9D A5 2885 9586 27 10 2886 9588 81 29 2887 9588 27 70 2888 958C BD 82 6D 2889 958F 81 2C 2890 9591 27 05 2891 > 9593 BD 95 5A 2892 9596 BD 0A 2893 9598 0E A5	* PSET, PRESET IF NO EXPRESSION ,) OR * , . OTHERWISE EVALUATE THE EXPRESSION L9581 JSR L959A JSR GETCCH BEQ L9598 CMPA #')' BEQ L9598 JSR SYNCOMMA CMPA #',' BEQ L9598 JSR L955A BSR L955A BSR L95A2 L9598 JMP GETCCH	GET THE COLOR OF A BYTE CHECK CURRENT INPUT CHARACTER BRANCH IF NONE * CHECK FOR) AND BRANCH IF * NO MORE ARGUMENTS SYNTAX CHECK FOR COMMA WAS NEXT CHARACTER A COMMA? YES EVALUATE EXPRESSION, RETURN COLOR IN ACCB TEMP STORE COLOR AND ALL PIXEL BYTE CHECK INPUT CHARACTER AND RETURN
2895 959A D6 B2 2897 959C ØD C2 2898 959E 26 Ø2 2899 95AØ D6 B3 29ØØ 95AØ D7 B4 29Ø1 95AØ 3D 29Ø3 95AØ 3D 29Ø3 95AØ 3D 29Ø3 95AØ 3D 29Ø3 95AØ 29Ø5 95AØ 29Ø6 29Ø7 29Ø8	* SET THE ACTIVE COLOR BYTE AND THE ALL AND L959A LDB FORCOL TST SETFLG BNE L95A2 LDB BAKCOL L95A2 STB WCOLOR LDA #\$55 MUL STB ALLCOL RTS L95AA BNE L95CF * THIS CODE WILL RESET THE DISPLAY PAGE R * SAM CHIP TO 2 (\$400) AND RESET THE SAM S	GET FOREGROUND COLOR CHECK PSET/PRESET FLAG BRANCH IF PSET GET BACKGROUND COLOR TEMP STORE COLOR CONSIDER A BYTE AS 4 PIXELS SET COLOR ON ALL 4 PIXELS SAVE BYTE WITH ALL PIXELS TURNED ON BRANCH IF GRAPHIC MODE, OTHERWISE SET UP ALPHA GRAPHIC MODE EGISTER IN THE
2989 2910 2911 2912 2913 395AC 34 16 2914 95AE 8E FF C8 2915 95B1 A7 ØA 2916 95B3 A7 Ø8 2917 95B5 A7 Ø6 2918 95B7 A7 Ø4 2919 95B9 A7 Ø2 2920 95B9 A7 Ø2 2920 95B8 A7 Ø1 2921 95B0 A7 1E 2922 95BF A7 1C 2923 95C1 A7 1A 2924 95C3 A7 18 2925 95C5 86 FF 22 2926 95C8 84 Ø7 2927 95CA B7 FF 22 2928 95C0 35 96	* REGISTER TO Ø (ALPHA-NUMERICS). IN ADDI' * RESET THE VDG CONTROL PINS TO ALPHA-GRA * SET UP THE SAM AND VDG TO GRAPHICS MODE L95AC PSHS X,B,A LDX #SAM+B STA 10,X STA \$06,X STA \$06,X STA \$04,X STA \$02,X STA \$01,X STA \$02,X STA \$01,X STA \$-02,X STA \$-04,X STA \$-04,X STA \$-04,X STA \$-04,X STA \$-04,X STA \$-04,X STA \$-05,X STA \$-0	
2929 2930 95CF 34 16 2931 95D1 96 B6 2932 95D3 8B Ø3 2933 95D5 C6 10 2934 95D7 3D 2935 95D8 CA 8Ø 2936 95DA DA C1 2937 95DC B6 FF 22 2938 95DF 84 Ø7 2939 95E1 34 Ø2 2940 95E3 EA EØ 2941 95E5 F7 FF 22 2942 95E8 96 BA 2944 2944 2944 2945 > 95EB BD 96 ØF 2946 95EE 96 B6 2947 95FØ 8B Ø3 2948 95FØ 8B Ø3 2948 95FØ 8B Ø3 2948 95FØ 8B Ø3 2948 95FØ 4A 2949 95FØ 6A 295FØ 95FØ 6A	L95CF PSHS X,B,A LDA PMODE ADDA #\$03 LDB #\$10 MUL ORB #\$80 ORB CSSVAL LDA PIA1+2 ANDA #\$07 PSHS A ORB ,S+ STB PIA1+2 LDA BEGGRP LSRA * JSR L960F LDA PMODE ADDA #\$03 CMPA #\$07 BNE L95F7 DECA L95F7 BSR L95FB	GET CURRENT PMODE VALUE ADD 3 - NON 3-7 ONLY 5 OF 8 POSSIBLE MODES USED \$10 OFFSET BETWEEN PMODES GET PMODE VALUES FOR VOG GMØ, GM1, GM2 FORCE BIT 7 HIGH (VDG A/G CONTROL) OR IN THE VDG CSS DATA GET PIAL, PORT B MASK OFF THE VDG CONTROL DATA SAVE IT OR IT WITH THE VDG VALUES CALCULATED ABOVE STORE IT INTO THE PIA GET MSB OF START OF GRAPHIC PAGE *DIVIDE BY 2 - ACCA CONTAINS HOW MANY 512 BYTE *BLOCKS IN STARTING ADDR GO SET SAM CONTROL REGISTER GET PMODE VALUE ADD IN BIAS TO ADJUST TO PMODE THE SAM REGISTER WANTS WAS PMODE 4? NO DECREMENT ACCA IF PMODE 4 (SAME VDG AS PMODE3) SET THE SAM S VDG REGISTER
2952 95F9 35 96 2953 2954 95FB C6 Ø3 2955 2956 95FD 8E FF CØ 2957 96ØØ 46 2958 96Ø1 24 Ø4 2959 96Ø3 A7 Ø1 296Ø 96Ø5 2Ø Ø2 2961 96Ø7 A7 84 2962 96Ø9 3Ø Ø2 2963 96Ø9 3Ø Ø2 2963 96Ø8 5A 2964 96ØC 26 F2 2965 9ØØE 39	PULS A,B,X,PC L95FB LDB #\$03 * ENTER WITH DATA TO GO IN VDG REGISTER II LDX #SAM L9600 RORA BCC L9607 STA \$01,X BRA L9609 L9607 STA ,X L9609 LEAX \$02,X DECB BNE L9600 RTS	RESTORE REGISTERS AND RETURN 3 BITS IN SAM VDG CONTROL REGISTER N BOTTOM 3 BITS OF ACCA POINT X TO SAM CONTROL REGISTER PUT A BIT INTO CARRY FLAG BRANCH IF BIT WAS A ZERO SET SAM REGISTER BIT DO NEXT BIT CLEAR SAM REGISTER NEXT BIT IN REGISTER DONE ALL BITS? NO
2966 960F C6 07 2967 9611 8E FF C6 2968 9614 20 EA 2969 9616 86 FF 22 2970 9618 9A C1 2971 9618 9A C1 2972 9610 B7 FF 22 2973 9620 39 2974 2975 2976 9621 81 2C	L960F LDB #\$07 LDX #SAM+6 BRA L9600 L9616 LDA PIA1+2 ANDA #\$F7 ORA CSSVAL STA PIA1+2 RTS * PMODE PMOD CMPA #','	7 BITS IN SAM DISPLAY PAGE REGISTER POINT X TO SAM DISPLAY PAGE REGISTER GO SET THE REGISTER GET PIA1, PORT B MASK OFF VDG CSS CONTROL BIT OR IN CSS COLOR DATA RESTORE IT IN PIA1 CHECK FOR COMMA - FIRST ARGUMENT MAY BE MISSING

2077					
	0692 27 20		DEO	10658	IT IC A COMMA
2977 2978	9623 27 2B 9625 BD B7 ØB			L965Ø EVALEXPB	IT IS A COMMA
2978	9628 C1 Ø5		CMPB		EVALUATE EXPRESSION > 4?
2979	962A 24 41				
2981				L966D	YES, ILLEGAL FUNCTION CALL
	962C 96 BC	10005		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			#L97Ø6+1	LOOKUP TABLE
2985	9634 AB C5		ADDA		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		*GET THE NUMBER OF BYTES/HORIZONTAL LINE
2991	964Ø 97 B9			HORBYT	*AND SAVE IT IN HORBYT
2992	9642 54		LSRB	none i	RESTORE PMODE VALUE
2993	9643 D7 B6		STB	PMODE	SAVE IT
	9645 4F			FRIODE	BACKGROUND COLOR
2994			CLRA	DAKCOL	
2995	9646 97 B3			BAKCOL	SET BACKGROUND COLOR TO ZERO
2996	9648 86 Ø3		LDA		FOREGROUND COLOR
2997	964A 97 B2			FORCOL	SET FOREGROUND COLOR
2998	964C 9D A5			GETCCH	IS THERE A STARTING PAGE NUMBER?
2999	964E 27 1C		BEQ	L966C	NO 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	#\$06	EACH GRAPHIC PAGE = 6 X 256 (1.5K)
3005	9659 3D		MUL	•	MULT BY PAGE NUMBER
3006	965A DB BC			GRPRAM	ADD IN START OF GRAPHIC RAM
3007	965C 34 Ø4		PSHS		SAVE TEMP START ADDR
3008	965E DB B7			ENDGRP	ADD CURRENT END ADDR
3009	9660 DØ BA			BEGGRP	SUB OUT CURRENT START ADDR - (ADDS THE SIZE OF ONE GRAPHIC PAGE)
3010	9662 D1 19			TXTTAB	IS IT > CURRENT START OF BASIC PROGRAM
3Ø11	9664 22 Ø7			L966D	YES! ILLEGAL FUNCTION CALL
3Ø12	9666 D7 B7			ENDGRP	SAVE AS END OF GRAPHIC PAGE
3Ø13	9668 35 Ø4		PULS		GET TEMP START ADOR
3Ø14	966A D7 BA		STB	BEGGRP	SAVE AS START OF GRAPHIC PAGE
3Ø15	966C 39	L966C	RTS		
3Ø16	966D 7E B4 4A	L966D	JMP	LB44A	ILLEGAL FUNCTION CALL'
3Ø17					
3Ø18		* SCREEN	ı		
3Ø19	967Ø 81 2C	SCREEN	CMPA	#'.'	CHECK FOR A COMMA
3020	9672 27 ØB			L967F	BRANCH IF COMMA - FIRST ARGUMENT MISSING
3021	9674 BD B7 ØB			EVALEXPB	EVALUATE EXPRESSION
3022	9677 5D		TSTB	ETALEAID	ZERO FLAG SET IF ALPHA, NOT SET IF GRAPHIC SCREEN
				LOFAA	
3023	9678 BD 95 AA			L95AA	SET UP THE SAM & VDG FOR PROPER GRAPHIC MODE
3Ø24	967B 9D A5		JSR	GETCCH	GET NEXT CHARACTER
3Ø25	967D 27 ED			L966C	RETURN IF NOTHING ELSE ON LINE
3Ø26	967F BD B7 38	L967F	JSR	LB738	CHECK FOR COMMA AND EVALUATE EXPRESSION
3Ø27	9682 5D		TSTB		SET FLAGS
3Ø28	9683 27 Ø2		BEQ	L9687	BRANCH IF COLOR SET ZERO
3Ø29	9685 C6 Ø8		LDB	#\$Ø8	VALUE FOR COLOR SET ONE
3Ø3Ø	9687 D7 C1	L9687	STB	CSSVAL	SAVE IN VDG CSS RAM IMAGE
				L9616	GO SET IT INTO PIA
3Ø31	9689 20 8B				
3Ø31 3Ø32	9689 2Ø 8B				
3Ø32	9689 20 8B	* DC1EAG			
3Ø32 3Ø33		* PCLEAR	₹	EVALEYDD	EVALUATE EVADESSION DETUDN VALUE IN ACCO
3Ø32 3Ø33 3Ø34	968B BD B7 ØB	* PCLEAR PCLEAR	R JSR	EVALEXPB	EVALUATE EXPRESSION, RETURN VALUE IN ACCB
3032 3033 3034 3035	968B BD B7 ØB 968E 5D		JSR TSTB		SET FLAGS
3032 3033 3034 3035 3036	968B BD B7 ØB 968E 5D 968F 27 DC		JSR TSTB BEQ	L966D	SET FLAGS BRANCH IF PCLEARØ - FC ERROR
3032 3033 3034 3035 3036 3037	968B BD B7 ØB 968E 5D 968F 27 DC 9691 C1 Ø9		JSR TSTB BEQ CMPB	L966D #\$09	SET FLAGS BRANCH IF PCLEARØ - FC ERROR TRYING TO CLEAR MORE THAN 8 PAGES?
3032 3033 3034 3035 3036 3037 3038	968B BD B7 ØB 968E 5D 968F 27 DC 9691 C1 Ø9 9693 24 D8		JSR TSTB BEQ CMPB BCC	L966D #\$Ø9 L966D	SET FLAGS BRANCH IF PCLEARØ - FC ERROR TRYING TO CLEAR MORE THAN 8 PAGES? YES ILLEGAL FUNCTION CALL
3032 3033 3034 3035 3036 3037	968B BD B7 ØB 968E 5D 968F 27 DC 9691 C1 Ø9		JSR TSTB BEQ CMPB BCC	L966D #\$09	SET FLAGS BRANCH IF PCLEARØ - FC ERROR TRYING TO CLEAR MORE THAN 8 PAGES? YES ILLEGAL FUNCTION CALL 6 X 256 (1.5K) PER GRAPHIC PAGE
3032 3033 3034 3035 3036 3037 3038	968B BD B7 ØB 968E 5D 968F 27 DC 9691 C1 Ø9 9693 24 D8		JSR TSTB BEQ CMPB BCC LDA MUL	L966D #\$89 L966D #\$86	SET FLAGS BRANCH IF PCLEARØ - FC ERROR TRYING TO CLEAR MORE THAN 8 PAGES? YES ILLEGAL FUNCTION CALL
3032 3033 3034 3035 3036 3037 3038 3039	968B BD B7 ØB 968E 5D 968F 27 DC 9691 C1 Ø9 9693 24 DB 9695 86 Ø6		JSR TSTB BEQ CMPB BCC LDA MUL	L966D #\$Ø9 L966D	SET FLAGS BRANCH IF PCLEARØ - FC ERROR TRYING TO CLEAR MORE THAN 8 PAGES? YES ILLEGAL FUNCTION CALL 6 X 256 (1.5K) PER GRAPHIC PAGE
3032 3033 3034 3035 3036 3037 3038 3039 3040	968B BD B7 ØB 968E 5D 968F 27 DC 9691 C1 Ø9 9693 24 DB 9695 86 Ø6 9697 3D		JSR TSTB BEQ CMPB BCC LDA MUL	L966D #\$89 L966D #\$96 GRPRAM	SET FLAGS BRANCH IF PCLEARØ - FC ERROR TRYING TO CLEAR MORE THAN 8 PAGES? YES ILLEGAL FUNCTION CALL 6 X 256 (1.5K) PER GRAPHIC PAGE MULT BY NUMBER OF PAGES
3032 3033 3034 3035 3036 3037 3038 3039 3040 3041	9688 BD B7 ØB 968E 5D 968F 27 DC 9691 C1 Ø9 9693 24 DB 9693 86 Ø6 9697 3D 9698 DB BC		JSR TSTB BEQ CMPB BCC LDA MUL ADDB	L966D #\$09 L966D #\$06 GRPRAM B,A	SET FLAGS BRANCH IF PCLEARØ - FC ERROR TRYING TO CLEAR MORE THAN 8 PAGES? YES ILLEGAL FUNCTION CALL 6 X 256 (1.5K) PER GRAPHIC PAGE MULT BY NUMBER OF PAGES ADD IN START OF GRAPHIC RAM
3032 3033 3034 3035 3036 3037 3038 3039 3040 3041 3042	9688 BD 87 ØB 968E 5D 968F 27 DC 9691 C1 Ø9 9693 24 DB 9695 86 Ø6 9697 3D 9698 DB BC 9694 1F 98		JSR TSTB BEQ CMPB BCC LDA MUL ADDB TFR	L966D #\$09 L966D #\$06 GRPRAM B,A	SET FLAGS BRANCH IF PCLEARØ - FC ERROR TRYING TO CLEAR MORE THAN 8 PAGES? YES ILLEGAL FUNCTION CALL 6 X 256 (1.5K) PER GRAPHIC PAGE MULT BY NUMBER OF PAGES ADD IN START OF GRAPHIC RAM MOVE B TO MSB OF REG ACCD
3032 3033 3034 3035 3036 3037 3038 3039 3040 3041 3042 3043	968B BD B7 ØB 968E 5D 968F 27 DC 9691 C1 Ø9 9693 24 D8 9695 86 Ø6 9697 3D 9698 DB BC 9698 DB BC 9694 IF 98		JSR TSTB BEQ CMPB BCC LDA MUL ADDB TFR LDB	L966D #\$09 L966D #\$06 GRPRAM B,A	SET FLAGS BRANCH IF PCLEARØ - FC ERROR TRYING TO CLEAR MORE THAN 8 PAGES? YES ILLEGAL FUNCTION CALL 6 X 256 (1.5K) PER GRAPHIC PAGE MULT BY NUMBER OF PAGES ADD IN START OF GRAPHIC RAM MOVE B TO MSB OF REG ACCD REG D NOW CONTAINS TOP OF PCLEARED SPACE +1
3032 3033 3034 3035 3036 3037 3038 3039 3040 3041 3042 3043 3044	968B BD B7 ØB 968E 5D 968F 27 DC 9691 C1 Ø9 9693 24 D8 9695 86 Ø6 9697 3D 9698 DB BC 9698 DB BC 9696 C6 Ø1 9696 LF Ø8	PCLEAR	JSR TSTB BEQ CMPB BCC LDA MUL ADDB TFR LDB TFR CMPD	L966D #\$09 L966D #\$06 GRPRAM B,A #\$01 D,Y ENDGRP	SET FLAGS BRANCH IF PCLEARØ - FC ERROR TRYING TO CLEAR MORE THAN 8 PAGES? YES ILLEGAL FUNCTION CALL 6 X 256 (1.5K) PER GRAPHIC PAGE MULT BY NUMBER OF PAGES ADD IN START OF GRAPHIC RAM MOVE B TO MSB OF REG ACCD REG D NOW CONTAINS TOP OF PCLEARED SPACE +1 SAVE IN Y
3032 3033 3034 3035 3036 3037 3038 3049 3041 3042 3043 3044 3044 3045 3046	968B BD B7 ØB 968E 5D 968F 27 DC 9691 C1 Ø9 9693 24 DB 9695 86 Ø6 9697 3D 9698 DB BC 969A IF 98 969C C6 Ø1 969E 1F Ø2 96AØ 1Ø 93 B7	PCLEAR	JSR TSTB BEQ CMPB BCC LDA MUL ADDB TFR LDB TFR CMPD	L966D #\$09 L966D #\$06 GRPRAM B,A #\$01 D,Y ENDGRP FFLECTS THE INFAMOUS PCLEAR BUG	SET FLAGS BRANCH IF PCLEARØ - FC ERROR TRYING TO CLEAR MORE THAN 8 PAGES? YES ILLEGAL FUNCTION CALL 6 X 256 (1.5K) PER GRAPHIC PAGE MULT BY NUMBER OF PAGES ADD IN START OF GRAPHIC RAM MOVE B TO MSB OF REG ACCD REG D NOW CONTAINS TOP OF PCLEARED SPACE +1 SAVE IN Y COMPARE TOP OF PCLEARED SPACE TO END OF CURRENT GRAPHIC PAGE
3032 3033 3034 3035 3036 3037 3038 3049 3044 3042 3043 3044 3045 3044 3044	968B BD B7 ØB 968E 5D 968F 27 DC 9691 C1 Ø9 9693 24 D8 9695 86 Ø6 9697 3D 9698 DB BC 9698 DB BC 9698 1F 98 969C C6 Ø1 969E 1F Ø2 96AØ 1Ø 93 B7	PCLEAR	JSR TSTB BEQ CMPB BCC LDA MUL ADDB TFR LDB TFR CMPD CODE RI BLO	L966D #\$09 L966D #\$06 GRPRAM B,A #\$01 D,Y ENDGRP EFLECTS THE INFAMOUS PCLEAR BUG L966D	SET FLAGS BRANCH IF PCLEARØ - FC ERROR TRYING TO CLEAR MORE THAN 8 PAGES? YES ILLEGAL FUNCTION CALL 6 X 256 (1.5K) PER GRAPHIC PAGE MULT BY NUMBER OF PAGES ADD IN START OF GRAPHIC RAM MOVE B TO MSB OF REG ACCD REG D NOW CONTAINS TOP OF PCLEARED SPACE +1 SAVE IN Y COMPARE TOP OF PCLEARED SPACE TO END OF CURRENT GRAPHIC PAGE FC ERROR IF TRYING TO CLEAR < END OF GRAPHIC RAM
3032 3033 3034 3035 3036 3037 3040 3041 3042 3043 3044 3045 3046 3047 3048	968B BD B7 ØB 968E 5D 968F 27 DC 9691 CI 09 9693 24 DB 9695 86 Ø6 9697 3D 9698 DB BC 969A 1F 98 969C C6 Ø1 969E 1F Ø2 96AØ 1Ø 93 B7	PCLEAR	JSR TSTB BEQ CMPB BCC LDA MUL ADDB TFR LDB TFR CMPD CODE RI BLO SUBD	L966D #\$09 L966D #\$06 GRPRAM B,A #\$01 D,Y ENDGRP FLECTS THE INFAMOUS PCLEAR BUG L966D	SET FLAGS BRANCH IF PCLEARØ - FC ERROR TRYING TO CLEAR MORE THAN 8 PAGES? YES ILLEGAL FUNCTION CALL 6 X 256 (1.5K) PER GRAPHIC PAGE MULT BY NUMBER OF PAGES ADD IN START OF GRAPHIC RAM MOVE B TO MSB OF REG ACCD REG D NOW CONTAINS TOP OF PCLEARED SPACE +1 SAVE IN Y COMPARE TOP OF PCLEARED SPACE TO END OF CURRENT GRAPHIC PAGE FC ERROR IF TRYING TO CLEAR < END OF GRAPHIC RAM SUBTRACT START OF BASIC PROGRAM
3032 3033 3034 3035 3036 3037 3040 3041 3042 3043 3044 3045 3046 3047 3048	968B BD B7 ØB 968E 5D 968F 27 DC 9691 C1 Ø9 9693 24 D8 9695 86 Ø6 9697 3D 9698 DB BC 9698 DF BC	PCLEAR	JSR TSTB BEQ CMPB BCC LDA MUL ADDB TFR LDB TFR CMPD CODE RI BLO ADDD	L966D #\$09 L966D #\$06 GRPRAM B,A #\$01 D,Y ENDGRP FLECTS THE INFAMOUS PCLEAR BUG L966D TXTTAB	SET FLAGS BRANCH IF PCLEARØ - FC ERROR TRYING TO CLEAR MORE THAN 8 PAGES? YES ILLEGAL FUNCTION CALL 6 X 256 (1.5K) PER GRAPHIC PAGE MULT BY NUMBER OF PAGES ADD IN START OF GRAPHIC RAM MOVE B TO MSB OF REG ACCD REG D NOW CONTAINS TOP OF PCLEARED SPACE +1 SAVE IN Y COMPARE TOP OF PCLEARED SPACE TO END OF CURRENT GRAPHIC PAGE FC ERROR IF TRYING TO CLEAR < END OF GRAPHIC RAM SUBTRACT START OF BASIC PROGRAM ADD END OF BASIC PROGRAM
3032 3033 3034 3035 3036 3038 3040 3041 3042 3043 3044 3044 3045 3046 3047 3048 3049 3050	9688 BD B7 ØB 968E 5D 968F 27 DC 9691 C1 Ø9 9693 24 D8 9697 3D 9698 DB BC 9698 DB BC 9690 1F 98 9690 C6 Ø1 9696 1F Ø2 96AØ 1Ø 93 B7 96A3 25 C8 96A7 D3 1B 96A9 F Ø1	PCLEAR	JSR TSTB BEQ CMPB BCC LDA MUL ADDB TFR LDB TFR CMPD CODE RI BLO SUBD ADDD TFR	L966D #\$09 L966D #\$06 GRPRAM B,A #\$01 D,Y ENDGRP FLECTS THE INFAMOUS PCLEAR BUG L966D TXTTAB	SET FLAGS BRANCH IF PCLEARØ - FC ERROR TRYING TO CLEAR MORE THAN 8 PAGES? YES ILLEGAL FUNCTION CALL 6 X 256 (1.5K) PER GRAPHIC PAGE MULT BY NUMBER OF PAGES ADD IN START OF GRAPHIC RAM MOVE B TO MSB OF REG ACCD REG D NOW CONTAINS TOP OF PCLEARED SPACE +1 SAVE IN Y COMPARE TOP OF PCLEARED SPACE TO END OF CURRENT GRAPHIC PAGE FC ERROR IF TRYING TO CLEAR < END OF GRAPHIC RAM SUBTRACT START OF BASIC PROGRAM ADD END OF BASIC PROGRAM X=TOP OF PCLEARED SPACE + LENGTH OF BASIC PROGRAM
3032 3033 3034 3035 3036 3037 3038 3049 3041 3042 3043 3044 3045 3046 3047 3048 3049 3051	968B BD B7 ØB 968E 5D 968F 27 DC 9691 CI Ø9 9693 24 DB 9695 86 Ø6 9697 3D 9698 DB BC 969A 1F 98 969C C6 Ø1 969E 1F Ø2 96AØ 1Ø 93 B7 96AS 25 C8 96A5 25 C8 96A5 703 1B 96A9 1F Ø1 96A8 1F	PCLEAR	JSR TSTB BEQ CMPB BCC LDA MUL ADDB TFR LDB TFR CMPD CODE RI BLO SUBD ADDD TFR INCA	L966D #\$89 L966D #\$\$6 GRPRAM B, A #\$\$1 D,Y ENDGRP EFLECTS THE INFAMOUS PCLEAR BUG L966D TXTTAB VARTAB D,X	SET FLAGS BRANCH IF PCLEARØ - FC ERROR TRYING TO CLEAR MORE THAN 8 PAGES? YES ILLEGAL FUNCTION CALL 6 X 256 (1.5K) PER GRAPHIC PAGE MULT BY NUMBER OF PAGES ADD IN START OF GRAPHIC RAM MOVE B TO MSB OF REG ACCD REG D NOW CONTAINS TOP OF PCLEARED SPACE +1 SAVE IN Y COMPARE TOP OF PCLEARED SPACE TO END OF GRAPHIC RAM SUBTRACT START OF BASIC PROGRAM ADD END OF BASIC PROGRAM X=TOP OF PCLEARED SPACE + LENGTH OF BASIC PROGRAM ADD 256 - LEAVE SOME ROOM FOR THE STACK
3/832 3/833 3/834 3/835 3/836 3/837 3/838 3/839 3/844 3/844 3/844 3/844 3/844 3/844 3/844 3/844 3/849 3/850 3/850 3/852	968B BD B7 ØB 968E 5D 968F 27 DC 9691 C1 Ø9 9693 24 DB 9697 3D 9698 DB BC 9697 3D 9698 DB BC 9698 1F 98 9696 1F Ø2 96AØ 1Ø 93 B7 96AØ 1Ø 93 B7	PCLEAR	JSR TSTB BEQ CMPB BCC LDA MUL ADDB TFR LDB TFR CMPD CODE RI BLO SUBD ADDD TFR INCA SUBD	L966D #\$09 L966D #\$06 GRPRAM B,A #\$01 D,Y ENDGRP EFLECTS THE INFAMOUS PCLEAR BUG L966D TXTTAB VARTAB D,X	SET FLAGS BRANCH IF PCLEARØ - FC ERROR TRYING TO CLEAR MORE THAN 8 PAGES? YES ILLEGAL FUNCTION CALL 6 X 256 (1.5K) PER GRAPHIC PAGE MULT BY NUMBER OF PAGES ADD IN START OF GRAPHIC RAM MOVE B TO MSB OF REG ACCD REG D NOW CONTAINS TOP OF PCLEARED SPACE +1 SAVE IN Y COMPARE TOP OF PCLEARED SPACE TO END OF CURRENT GRAPHIC PAGE FC ERROR IF TRYING TO CLEAR < END OF GRAPHIC RAM SUBTRACT START OF BASIC PROGRAM ADD END OF BASIC PROGRAM X=TOP OF PCLEARED SPACE + LENGTH OF BASIC PROGRAM ADD 256 - LEAVE SOME ROOM FOR THE STACK SUBTRACT OUT TOP OF CLEARED SPACE
3032 3033 3034 3035 3036 3037 3038 3049 3041 3042 3043 3044 3045 3047 3048 3050 3051 3051 3052 3053	9688 BD B7 ØB 968E 5D 968F 27 DC 9691 C1 Ø9 9693 24 D8 9697 3D 9698 DB BC 9697 3D 9698 DB BC 9696 1F 98 969C C6 Ø1 969E 1F Ø2 96AØ 1Ø 93 B7 96A3 25 C8 96A5 93 19 96A7 D3 1B 96A9 1F Ø1 96A8 4C 96AC 93 21	PCLEAR	JSR TSTB BEQ CMPB BCC LDA MUL ADDB TFR LDB TFR CMPD CODE RI BLO SUBD ADDD TFR INCA SUBD BCC	L966D #\$09 L966D #\$06 GRPRAM B,A #\$51 D,Y ENDGRP FFLECTS THE INFAMOUS PCLEAR BUG L966D TXTTAB VARTAB D,X FRETOP L966D	SET FLAGS BRANCH IF PCLEARØ - FC ERROR TRYING TO CLEAR MORE THAN 8 PAGES? YES ILLEGAL FUNCTION CALL 6 X 256 (1.5K) PER GRAPHIC PAGE MULT BY NUMBER OF PAGES ADD IN START OF GRAPHIC RAM MOVE B TO MSB OF REG ACCD REG D NOW CONTAINS TOP OF PCLEARED SPACE +1 SAVE IN Y COMPARE TOP OF PCLEARED SPACE TO END OF CURRENT GRAPHIC PAGE FC ERROR IF TRYING TO CLEAR < END OF GRAPHIC RAM SUBTRACT START OF BASIC PROGRAM ADD END OF BASIC PROGRAM X=TOP OF PCLEARED SPACE + LENGTH OF BASIC PROGRAM ADD 256 - LEAVE SOME ROOM FOR THE STACK SUBTRACT OUT TOP OF CLEARED SPACE FC ERROR - NO ROOM LEFT
3/832 3/833 3/834 3/835 3/836 3/837 3/848 3/849 3/841 3/844 3/845 3/846 3/846 3/847 3/848 3/849 3/851 3/852	968B BD B7 ØB 968E 5D 968F 27 DC 9691 CI Ø9 9693 24 DB 9695 86 Ø6 9697 3D 9698 DB BC 9696 1F 98 969C C6 Ø1 9696 1F Ø2 96AØ 1Ø 93 B7 96A3 25 C8 96A5 73 19 96A7 D3 1B 96A9 1F Ø1 96A8 4C 96AC 93 21 96AE 24 BD	PCLEAR	JSR TSTB BEQ CMPB BCC LDA MUL ADDB TFR LDB TFR CMPD CODE RI BLO SUBD TFR INCA SUBD BCC JSR	L966D #\$09 L966D #\$06 GRPRAM B,A #\$01 D,Y ENDGRP EFLECTS THE INFAMOUS PCLEAR BUG L966D TXTTAB VARTAB D,X	SET FLAGS BRANCH IF PCLEARØ - FC ERROR TRYING TO CLEAR MORE THAN 8 PAGES? YES ILLEGAL FUNCTION CALL 6 X 256 (1.5K) PER GRAPHIC PAGE MULT BY NUMBER OF PAGES ADD IN START OF GRAPHIC RAM MOVE B TO MSB OF REG ACCD REG D NOW CONTAINS TOP OF PCLEARED SPACE +1 SAVE IN Y COMPARE TOP OF PCLEARED SPACE TO END OF GRAPHIC RAM SUBTRACT START OF BASIC PROGRAM ADD END OF BASIC PROGRAM X=TOP OF PCLEARED SPACE + LENGTH OF BASIC PROGRAM ADD 256 - LEAVE SOME ROOM FOR THE STACK SUBTRACT OUT TOP OF CLEARED SPACE FC ERROR - NO ROOM LEFT ADJUST BASIC S INPUT POINTER
3/832 3/833 3/834 3/835 3/836 3/837 3/838 3/839 3/841 3/842 3/843 3/844 3/845 3/849 3/859	968B BD B7 ØB 968E 5D 968F 27 DC 9691 C1 Ø9 9693 24 D8 9695 86 Ø6 9697 3D 9698 DB BC 9698 1F 98 969C C6 Ø1 969E 1F Ø2 96AØ 1Ø 93 B7 96AØ 32 C8 96AØ 1Ø 93 B7 96AØ 19 96AØ 19 96AØ 19 96AØ 19 96AØ 10 96AØ 10 96AØ 10 96AØ 10	PCLEAR	JSR JSR TSTB BEQ CMPB BCC LDA MUL ADDB TFR LDB CMPD BCO CODE RI BLO SUBD TFR INCA SUBD BCC INCA NOP	L966D #\$09 L966D #\$06 GRPRAM B,A #\$01 D,Y ENDGRP FILECTS THE INFAMOUS PCLEAR BUG L966D TXTTAB VARTAB D,X FRETOP L966D L80D0	SET FLAGS BRANCH IF PCLEARØ - FC ERROR TRYING TO CLEAR MORE THAN 8 PAGES? YES ILLEGAL FUNCTION CALL 6 X 256 (1.5K) PER GRAPHIC PAGE MULT BY NUMBER OF PAGES ADD IN START OF GRAPHIC RAM MOVE B TO MSB OF REG ACCD REG D NOW CONTAINS TOP OF PCLEARED SPACE +1 SAVE IN Y COMPARE TOP OF PCLEARED SPACE TO END OF CURRENT GRAPHIC PAGE FC ERROR IF TRYING TO CLEAR < END OF GRAPHIC RAM SUBTRACT START OF BASIC PROGRAM ADD END OF BASIC PROGRAM X=TOP OF PCLEARED SPACE + LENGTH OF BASIC PROGRAM ADD 256 - LEAVE SOME ROOM FOR THE STACK SUBTRACT OUT TOP OF CLEARED SPACE FC ERROR - NO ROOM LEFT ADJUST BASIC S INPUT POINTER SPACE FILLER FOR EXBAS 1.1
3032 3033 3034 3035 3036 3037 3038 3049 3041 3042 3043 3044 3045 3047 3050 3051 3050 3051 3053 3054 3053 3054 3055	9688 BD B7 ØB 968E 5D 968F 27 DC 9691 C1 Ø9 9693 24 D8 9697 3D 9698 DB BC 9697 3D 9698 DB BC 9696 1F 98 969C C6 Ø1 969E 1F Ø2 96AØ 1Ø 93 B7 96AS 25 C8 96AF 93 19 96AF 93 19 96AF 93 19 96AF 93 21 96AE 24 BD 96BB BØ DØ 96BB DØ 96BB DØ	PCLEAR	JSR TSTB BEQ CMPB BCC CMPB LDA ADDB TFR CMPD LDB LOS SUBD ADDD TFR SUBD BCC JSR SUBD BCC JSR SUBD BCC LDU	L966D #\$09 L966D #\$06 GRPRAM B,A #\$01 D,Y ENDGRP EFLECTS THE INFAMOUS PCLEAR BUG L966D TXTTAB VARTAB D,X FRETOP L966D L80D0	SET FLAGS BRANCH IF PCLEARØ - FC ERROR TRYING TO CLEAR MORE THAN 8 PAGES? YES ILLEGAL FUNCTION CALL 6 X 256 (1.5K) PER GRAPHIC PAGE MULT BY NUMBER OF PAGES ADD IN START OF GRAPHIC RAM MOVE B TO MSB OF REG ACCD REG D NOW CONTAINS TOP OF PCLEARED SPACE +1 SAVE IN Y COMPARE TOP OF PCLEARED SPACE TO END OF CURRENT GRAPHIC PAGE FC ERROR IF TRYING TO CLEAR < END OF GRAPHIC RAM SUBTRACT START OF BASIC PROGRAM ADD END OF BASIC PROGRAM X=TOP OF PCLEARED SPACE + LENGTH OF BASIC PROGRAM ADD END OF BASIC PROGRAM ADD END OF BASIC POORGAM SUBTRACT OUT TOP OF CLEARED SPACE FC ERROR - NO ROOM LEFT ADJUST BASIC S INPUT POINTER SPACE FILLER FOR EXBAS 1.1 GET END OF BASIC PROGRAM
3/832 3/833 3/834 3/835 3/836 3/837 3/849 3/841 3/842 3/843 3/844 3/844 3/844 3/845 3/845 3/845 3/855 3/855 3/855 3/855 3/855 3/855	9688 BD B7 ØB 968E 5D 968F 27 DC 9691 CI Ø9 9693 24 DB 9695 86 Ø6 9697 3D 9698 DB BC 9690 CG Ø1 9692 IF Ø2 9640 IØ 93 B7	PCLEAR	JSR TSTB BEQ CMPB BEC LDA MUL ADDB TFR LDB TFR INCA SUBD TFR INCA SUBD SUBD SUBD SUBD SUBD SUBD SUBD SUBD	L966D #\$89 L966D #\$96 GRPRAM B,A #\$81 D,Y ENDGRP EFLECTS THE INFAMOUS PCLEAR BUG L966D TXTTAB VARTAB D,X FRETOP L966D L966D L966D VARTAB	SET FLAGS BRANCH IF PCLEARØ - FC ERROR TRYING TO CLEAR MORE THAN 8 PAGES? YES ILLEGAL FUNCTION CALL 6 X 256 (1.5K) PER GRAPHIC PAGE MULT BY NUMBER OF PAGES ADD IN START OF GRAPHIC RAM MOVE B TO MSB OF REG ACCD REG D NOW CONTAINS TOP OF PCLEARED SPACE +1 SAVE IN Y COMPARE TOP OF PCLEARED SPACE TO END OF GRAPHIC RAM SUBTRACT START OF BASIC PROGRAM ADD END OF BASIC PROGRAM X=TOP OF PCLEARED SPACE + LENGTH OF BASIC PROGRAM ADD 256 - LEAVE SOME ROOM FOR THE STACK SUBTRACT OUT TOP OF CLEARED SPACE FC ERROR IN TOP OF CLEARED SPACE FC ERROR IN STACK SUBTRACT OUT TOP OF CLEARED SPACE FC ERROR IN STACK SUBTRACT OUT TOP OF CLEARED SPACE FC ERROR - NO ROOM LEFT ADJUST BASIC S INPUT POINTER SPACE FILLER FOR EXBAS 1.1 GET END OF BASIC PROGRAM STORE NEW END OF BASIC PROGRAM
3032 3033 3034 3035 3036 3037 3038 3049 3041 3042 3043 3044 3045 3047 3050 3051 3050 3051 3053 3054 3053 3054 3055	9688 BD B7 ØB 968E 5D 968F 27 DC 9691 C1 Ø9 9693 24 D8 9697 3D 9698 DB BC 9697 3D 9698 DB BC 9696 1F 98 969C C6 Ø1 969E 1F Ø2 96AØ 1Ø 93 B7 96AS 25 C8 96AF 93 19 96AF 93 19 96AF 93 19 96AF 93 21 96AE 24 BD 96BB BØ DØ 96BB DØ 96BB DØ	PCLEAR	JSR TSTB BEQ CMPB BCC LDA MUL ADDB TFR LDB TFR INCA SUBD TFR INCA SUBD SUBD SUBD SUBD SUBD SUBD SUBD SUBD	L966D #\$09 L966D #\$06 GRPRAM B,A #\$01 D,Y ENDGRP EFLECTS THE INFAMOUS PCLEAR BUG L966D TXTTAB VARTAB D,X FRETOP L966D L80D0	SET FLAGS BRANCH IF PCLEARØ - FC ERROR TRYING TO CLEAR MORE THAN 8 PAGES? YES ILLEGAL FUNCTION CALL 6 X 256 (1.5K) PER GRAPHIC PAGE MULT BY NUMBER OF PAGES ADD IN START OF GRAPHIC RAM MOVE B TO MSB OF REG ACCD REG D NOW CONTAINS TOP OF PCLEARED SPACE +1 SAVE IN Y COMPARE TOP OF PCLEARED SPACE TO END OF CURRENT GRAPHIC PAGE FC ERROR IF TRYING TO CLEAR < END OF GRAPHIC RAM SUBTRACT START OF BASIC PROGRAM ADD END OF BASIC PROGRAM X=TOP OF PCLEARED SPACE + LENGTH OF BASIC PROGRAM ADD END OF BASIC PROGRAM ADD END OF BASIC POORGAM SUBTRACT OUT TOP OF CLEARED SPACE FC ERROR - NO ROOM LEFT ADJUST BASIC S INPUT POINTER SPACE FILLER FOR EXBAS 1.1 GET END OF BASIC PROGRAM
3/832 3/833 3/834 3/835 3/836 3/837 3/849 3/841 3/842 3/843 3/844 3/844 3/844 3/845 3/845 3/845 3/855 3/855 3/855 3/855 3/855 3/855	9688 BD B7 ØB 968E 5D 968F 27 DC 9691 CI Ø9 9693 24 DB 9695 86 Ø6 9697 3D 9698 DB BC 9690 CG Ø1 9692 IF Ø2 9640 IØ 93 B7	PCLEAR	JSR TSTB BEQ CMPB BCC CMPB BCC LDA MUL ADDB TFR LDB TFR DODG BCC SUBD ADDD SUBD BCC SUBD BCC SUBD BCC SUBD BCC SUBD BCC SUBD BCC STX CMPU	L966D #\$89 L966D #\$96 GRPRAM B,A #\$81 D,Y ENDGRP EFLECTS THE INFAMOUS PCLEAR BUG L966D TXTTAB VARTAB D,X FRETOP L966D L966D L966D VARTAB	SET FLAGS BRANCH IF PCLEARØ - FC ERROR TRYING TO CLEAR MORE THAN 8 PAGES? YES ILLEGAL FUNCTION CALL 6 X 256 (1.5K) PER GRAPHIC PAGE MULT BY NUMBER OF PAGES ADD IN START OF GRAPHIC RAM MOVE B TO MSB OF REG ACCD REG D NOW CONTAINS TOP OF PCLEARED SPACE +1 SAVE IN Y COMPARE TOP OF PCLEARED SPACE TO END OF GRAPHIC RAM SUBTRACT START OF BASIC PROGRAM ADD END OF BASIC PROGRAM X=TOP OF PCLEARED SPACE + LENGTH OF BASIC PROGRAM ADD 256 - LEAVE SOME ROOM FOR THE STACK SUBTRACT OUT TOP OF CLEARED SPACE FC ERROR IN TOP OF CLEARED SPACE FC ERROR IN STACK SUBTRACT OUT TOP OF CLEARED SPACE FC ERROR IN STACK SUBTRACT OUT TOP OF CLEARED SPACE FC ERROR - NO ROOM LEFT ADJUST BASIC S INPUT POINTER SPACE FILLER FOR EXBAS 1.1 GET END OF BASIC PROGRAM STORE NEW END OF BASIC PROGRAM
3/832 3/833 3/834 3/835 3/836 3/837 3/838 3/839 3/841 3/842 3/843 3/844 3/845 3/849 3/859 3/859 3/859 3/853 3/855 3/855 3/856 3/856 3/856 3/858	968B BD B7 ØB 968E 5D 968F 27 DC 9691 CI Ø9 9693 24 DB 9695 86 Ø6 9697 3D 9698 DB BC 9698 IF 98 969C C6 Ø1 969E IF Ø2 96AØ 1Ø 93 B7 96AØ 1Ø 93 B7 96AØ 1Ø 93 B7 96AØ 1Ø 93 B7 96AØ 1Ø 90 BP 96AØ 1Ø 10 BP 96AØ 10	PCLEAR	S JSR TSTB BEQ CMPB BCC LDA MUL LDB TFR LDB CODE RI INCA ADDD TFR INCA ADDD STX DBCC JSR CMPU BCC CMPU BCC CMPU BCC	L966D #\$89 L966D #\$86 GRPRAM B,A #\$\$1 D,Y ENDGRP EFLECTS THE INFAMOUS PCLEAR BUG L966D TXTTAB VARTAB D,X FRETOP L966D L80DØ VARTAB VARTAB VARTAB VARTAB VARTAB VARTAB L9604	SET FLAGS BRANCH IF PCLEARØ - FC ERROR TRYING TO CLEAR MORE THAN 8 PAGES? YES ILLEGAL FUNCTION CALL 6 X 256 (1.5K) PER GRAPHIC PAGE MULT BY NUMBER OF PAGES ADD IN START OF GRAPHIC RAM MOVE B TO MSB OF REG ACCD REG D NOW CONTAINS TOP OF PCLEARED SPACE +1 SAVE IN Y COMPARE TOP OF PCLEARED SPACE TO END OF CURRENT GRAPHIC PAGE FC ERROR IF TRYING TO CLEAR < END OF GRAPHIC RAM SUBTRACT START OF BASIC PROGRAM ADD END OF BASIC PROGRAM X=TOP OF PCLEARED SPACE + LENGTH OF BASIC PROGRAM ADD 256 - LEAVE SOME ROOM FOR THE STACK SUBTRACT OUT TOP OF CLEARED SPACE FC ERROR - NO ROOM LEFT ADJUST BASIC S INPUT POINTER SPACE FILLER FOR EXBAS 1.1 GET END OF BASIC PROGRAM STORE NEW BND OF BASIC PROGRAM STORE NEW BND OF BASIC PROGRAM STORE NEW BND OF BASIC PROGRAM COMPARE OLD END TO NEW END
3/832 3/833 3/834 3/835 3/836 3/837 3/849 3/841 3/844 3/844 3/845 3/847 3/846 3/847 3/845 3/845 3/855 3/855 3/855 3/855 3/855 3/856	9688 BD B7 ØB 968E 5D 968F 27 DC 9691 CI Ø9 9693 24 DB 9695 86 Ø6 9697 3D 9698 DB BC 9696 IF Ø2 9640 10 93 B7	PCLEAR * THIS C	R JSR TSTB BEQ CMPB BCC LDA MUL BCC LDA SUBD TFR INCA SUBD JSR NOPU LSTX CMPU BCC CMPU BCC LDA	L966D #\$09 L966D #\$\$06 GRPRAM B,A #\$01 D,Y ENDGRP EFLECTS THE INFAMOUS PCLEAR BUG L966D TXTTAB VARTAB D,X FRETOP L966D L966D VARTAB	SET FLAGS BRANCH IF PCLEARØ - FC ERROR TRYING TO CLEAR MORE THAN 8 PAGES? YES ILLEGAL FUNCTION CALL 6 X 256 (1.5K) PER GRAPHIC PAGE MULT BY NUMBER OF PAGES ADD IN START OF GRAPHIC RAM MOVE B TO MSB OF REG ACCD REG D NOW CONTAINS TOP OF PCLEARED SPACE +1 SAVE IN Y COMPARE TOP OF PCLEARED SPACE TO END OF GRAPHIC RAM SUBTRACT START OF BASIC PROGRAM ADD END OF BASIC PROGRAM X=TOP OF PCLEARED SPACE + LENGTH OF BASIC PROGRAM ADD 256 - LEAWE SOME ROOM FOR THE STACK SUBTRACT OUT TOP OF CLEARED SPACE FC ERROR IN TRYING TO CLEAR < SUBTRACT STACK SUBTRACT START OF BASIC PROGRAM ADD 256 - LEAWE SOME ROOM FOR THE STACK SUBTRACT OUT TOP OF CLEARED SPACE FC ERROR - NO ROOM LEFT ADJUST BASIC S INPUT POINTER SPACE FILLER FOR EXBAS 1.1 GET END OF BASIC PROGRAM STORE NEW END OF BASIC PROGRAM COMPARE OLD END TO NEW END BRANCH IF OLD END > NEW END BRANCH IF OLD D PROGRAM
3/832 3/833 3/834 3/835 3/836 3/837 3/838 3/839 3/841 3/842 3/843 3/844 3/845 3/849 3/850 3/850 3/855 3/855 3/855 3/856 3/855 3/856 3/858	968B BD B7 ØB 968E 5D 968F 27 DC 9691 C1 Ø9 9693 24 DB 9695 86 Ø6 9697 3D 9698 DB BC 9698 IF 98 969C C6 Ø1 969E IF Ø2 96AØ 1Ø 93 B7 96AØ 1Ø 98 96AØ 1F Ø1 96AØ 1Ø 98	PCLEAR * THIS C	S JSR TSTB BEQ CMPB BECC LDA MUL LDB TFR LDB TFR CMPD SUBD ADDD TFR SUBD ADDD TFR SUBD BCC NOP LDU STX CMPU BCC LDA STA STA STA STA STA STA STA STA STA ST	L966D #\$89 L966D #\$86 GRPRAM B,A #\$81 D,Y ENDGRP FLECTS THE INFAMOUS PCLEAR BUG L966D TXTTAB VARTAB D,X FRETOP L966D L880Ø VARTAB VARTAB VARTAB VARTAB VARTAB UARTAB VARTAB UARTAB VARTAB UARTAB VARTAB UARTAB	SET FLAGS BRANCH IF PCLEARØ - FC ERROR TRYING TO CLEAR MORE THAN 8 PAGES? YES ILLEGAL FUNCTION CALL 6 X 256 (1.5K) PER GRAPHIC PAGE MULT BY NUMBER OF PAGES ADD IN START OF GRAPHIC RAM MOVE B TO MSB OF REG ACCD REG D NOW CONTAINS TOP OF PCLEARED SPACE +1 SAVE IN Y COMPARE TOP OF PCLEARED SPACE TO END OF CURRENT GRAPHIC PAGE FC ERROR IF TRYING TO CLEAR < END OF GRAPHIC RAM SUBTRACT START OF BASIC PROGRAM ADD END OF BASIC PROGRAM MX-TOP OF PCLEARED SPACE + LENGTH OF BASIC PROGRAM ADD 256 - LEAVE SOME ROOM FOR THE STACK SUBTRACT OUT TOP OF CLEARED SPACE FC ERROR - NO ROOM LEFT ADJUST BASIC S INPUT POINTER SPACE FILLER FOR EXBAS 1.1 GET END OF BASIC PROGRAM STORE NEW END OF BASIC PROGRAM COMPARE OLD END TO NEW END BRANCH IF OLD END > NEW END BRANCH IF OLD END > NEW END GET BYTE FROM OLD PROGRAM MOVE TO NEW PROGRAM
3/832 3/833 3/834 3/835 3/836 3/837 3/849 3/844 3/845 3/845 3/845 3/859 3/856 3/856 3/857 3/856 3/857 3/856 3/857 3/856 3/857 3/856 3/857 3/856 3/857 3/856 3/857 3/856 3/857 3/856 3/857 3/856 3/857 3/856 3/857 3/856 3/857 3/856 3/857 3/856 3/857 3/856 3/857 3/856 3/856 3/857 3/856 3/857 3/856 3/856 3/857 3/856 3/857 3/856	968B BD B7 ØB 968E 5D 968F 27 DC 9691 CI 09 9693 24 DB 9695 86 Ø6 9697 3D 9698 DB BC 969A 1F 98 999C C6 Ø1 969B 1F Ø2 96AØ 1Ø 93 B7 96AØ 25 CB 96AØ 33 19 96AF 03 1B 96AF 03 1B 96AF 03 1B 96AF 03 21 96AF 03 1B 96AF 03 1B 96AF 1B 96	PCLEAR * THIS C	R JSR TSTB BEQ CMPB BCC LDA MUL ADDB TFR LDB TFR LDB LTFR LDB LTFR LDB LTFR LDB LTFR LDB LDC LDA LDD LDC LDA LDC	L966D #\$89 L966D #\$86 GRPRAM B,A #\$\$1 D,Y ENDGRP EFLECTS THE INFAMOUS PCLEAR BUG L966D TXTTAB VARTAB D,X FRETOP L966D L80DØ VARTAB VARTAB VARTAB VARTAB L9604 ,-U ,-X TXTTAB	SET FLAGS BRANCH IF PCLEARØ - FC ERROR TRYING TO CLEAR MORE THAN 8 PAGES? YES ILLEGAL FUNCTION CALL 6 X 256 (1.5K) PER GRAPHIC PAGE MULT BY NUMBER OF PAGES ADD IN START OF GRAPHIC RAM MOVE B TO MSB OF REG ACCD REG D NOW CONTAINS TOP OF PCLEARED SPACE +1 SAVE IN Y COMPARE TOP OF PCLEARED SPACE TO END OF CURRENT GRAPHIC PAGE FC ERROR IF TRYING TO CLEAR < END OF GRAPHIC RAM SUBTRACT START OF BASIC PROGRAM ADD END OF BASIC PROGRAM X=TOP OF PCLEARED SPACE + LENGTH OF BASIC PROGRAM ADD END OF BASIC PROGRAM X=TOP OF PCLEARED SPACE + LENGTH OF BASIC PROGRAM ADD END OF BASIC PROGRAM SUBTRACT OUT TOP OF CLEARED SPACE FC ERROR - NO ROOM LEFT ADJUST BASIC S INPUT POINTER SPACE FILLER FOR EMBAS 1.1 GET END OF BASIC PROGRAM STORE NEW END OF BASIC PROGRAM STORE NEW END OF BASIC PROGRAM COMPARE OLD END TO NEW END BRANCH IF OLD END > NEW END GET BYTE FROM OLD PROGRAM MOVE TO NEW PROGRAM LOCATION AT THE BEGINNING OF OLD PROGRAM?
3/832 3/833 3/834 3/835 3/836 3/837 3/838 3/839 3/849 3/844 3/845 3/846 3/847 3/847 3/847 3/845 3/855	9688 BD B7 ØB 9688 5D 9687 27 DC 9697 27 DC 9693 24 DB 9693 24 DB 9695 86 Ø6 9697 3D 9698 DB BC 9690 16 Ø2 9690 17 Ø2 9640 10 93 B7 9640 10 9640 BB 9640 10 9640 BB 9640 10 9640 BB 9650 10 9660 BB 9660 91 BB 9668 11 93 1B 9668 11 93 1B 9688 11 93 1B 9688 11 93 1B 9688 24 17 9680 A6 C2 9687 A7 B2 9661 11 93 19	PCLEAR * THIS C	R JSR TSTB BEQ MPB BCC LDA ADDB TFR CMPD BCC ADDD TFR SUBD BCC CMPU BNE BNE BNE	L966D #\$89 L966D #\$98 L966D #\$96 GRPRAM B,A #\$81 D,Y ENDGRP EFLECTS THE INFAMOUS PCLEAR BUG L966D TXTTAB VARTAB D,X FRETOP L966D L8000 VARTAB VARTAB VARTAB VARTAB L9604 ,-U ,-X TXTTAB L966D	SET FLAGS BRANCH IF PCLEARØ - FC ERROR TRYING TO CLEAR MORE THAN 8 PAGES? YES ILLEGAL FUNCTION CALL 6 X 256 (1.5K) PER GRAPHIC PAGE MULT BY NUMBER OF PAGES ADD IN START OF GRAPHIC RAM MOVE B TO MSB OF REG ACCD REG D NOW CONTAINS TOP OF PCLEARED SPACE +1 SAVE IN Y COMPARE TOP OF PCLEARED SPACE TO END OF GRAPHIC RAM SUBTRACT START OF BASIC PROGRAM ADD END OF BASIC PROGRAM X=TOP OF PCLEARED SPACE + LENGTH OF BASIC PROGRAM ADD 256 - LEAVE SOME ROOM FOR THE STACK SUBTRACT OUT TOP OF CLEARED SPACE FC ERROR IF TRYING TO CLEAR < SND ROOM STACK SUBTRACT START OF BASIC PROGRAM STOP OF PCLEARED SPACE + LENGTH OF BASIC PROGRAM ADD 256 - LEAVE SOME ROOM FOR THE STACK SUBTRACT OUT TOP OF CLEARED SPACE FC ERROR - NO ROOM LEFT ADJUST BASIC S INPUT POINTER SPACE FILLER FOR EXBAS 1.1 GET END OF BASIC PROGRAM STORE NEW END OF BASIC PROGRAM COMPARE OLD END TO NEW END BRANCH IF OLD END > NEW END BRANCH IF OLD PROGRAM MOVE TO NEW PROGRAM LOCATION AT THE BEGINNING OF OLD PROGRAM? NO
3/832 3/833 3/834 3/835 3/836 3/837 3/838 3/839 3/841 3/842 3/843 3/844 3/845 3/849 3/856 3/856 3/855 3/856 3/855 3/856 3/856 3/856 3/858	968B BD B7 ØB 968E 5D 968F 27 DC 9691 C1 Ø9 9693 24 DB 9695 86 Ø6 9697 3D 9698 DB BC 9698 IF 98 969C C6 Ø1 9696 IF Ø2 96AØ 1Ø 93 B7	PCLEAR * THIS C	R JSR TSTB BEQ CMPB BCC LDA ADDB TFR LDB LDB SUBD BCC SUBD BCC JSR CMPU BCC LDA CMPU BCC LDA CMPU BCC LDA CMPU BCC LDA BCC STA CMPU BNE STY	L966D #\$89 L966D #\$86 GRPRAM B,A #\$91 D,Y ENDGRP FLECTS THE INFAMOUS PCLEAR BUG L966D TXTTAB VARTAB D,X FRETOP L966D L800D VARTAB VARTAB VARTAB UARTAB	SET FLAGS BRANCH IF PCLEARØ - FC ERROR TRYING TO CLEAR MORE THAN 8 PAGES? YES ILLEGAL FUNCTION CALL 6 X 256 (1.5K) PER GRAPHIC PAGE MULT BY NUMBER OF PAGES ADD IN START OF GRAPHIC RAM MOVE B TO MSB OF REG ACCD REG D NOW CONTAINS TOP OF PCLEARED SPACE +1 SAVE IN Y COMPARE TOP OF PCLEARED SPACE TO END OF CURRENT GRAPHIC PAGE FC ERROR IF TRYING TO CLEAR < END OF GRAPHIC RAM SUBTRACT START OF BASIC PROGRAM ADD END OF BASIC PROGRAM X=TOP OF PCLEARED SPACE + LENGTH OF BASIC PROGRAM ADD 256 - LEAVE SOME ROOM FOR THE STACK SUBTRACT OUT TOP OF CLEARED SPACE FC ERROR - NO ROOM LEFT ADJUST BASIC S INPUT POINTER SPACE FILLER FOR EXBAS 1.1 GET END OF BASIC PROGRAM COMPARE OLD END SOME END BRANCH IF OLD END > NEW END BRANCH IF OLD END > NEW END BRANCH IF OLD END > NEW END GET BYTE FROM OLD PROGRAM MOVE TO NEW PROGRAM HOVE TO NEW PROGRAM MOVE TO NEW PROGRAM HOVE TO NEW PROGRAM MOVE TO NEW PROGRAM HOVE TO NEW PROGRAM MOVE TO NEW PROGRAM LOCATION AT THE BEGINNING OF OLD PROGRAM? NO SAVE NEW STARTING ADDRESS
3/832 3/833 3/834 3/835 3/836 3/837 3/848 3/849 3/844 3/845 3/846 3/847 3/848 3/849 3/851 3/851 3/852 3/853 3/855	968B BD B7 ØB 968E 5D 968F 27 DC 969F 27 DC 9699 24 DB 9699 34 DB 9699 36 Ø6 9697 3D 9698 DB BC 9698 IF 98 969C C6 Ø1 9691 IF 92 96AØ 1Ø 93 B7 96AØ 25 C8 96A5 93 19 96A7 D3 1B 96A9 IF Ø1 96A8 4C 96AC 93 21 96A8 4C 96AC 93 21 96A8 9B BØ DØ 96B8 BD DØ 96	* THIS C	R JSR TSTB BEQ CMPB BCC LDA MUL ADDB TFR LDB TFR LDB TFR SUBD SUBD STR CMPD STX CMPU STX CMPU BEC LDA STX CMPU BNC STA CMPU BNC STA CTPU STA	L966D #\$89 L966D #\$\$96 GRPRAM B, A #\$\$1 D,Y ENDGRP EFLECTS THE INFAMOUS PCLEAR BUG L966D TXTTAB VARTAB D,X FRETOP L966D L88DØ VARTAB V	SET FLAGS BRANCH IF PCLEARØ - FC ERROR TRYING TO CLEAR MORE THAN 8 PAGES? YES ILLEGAL FUNCTION CALL 6 X 256 (1.5K) PER GRAPHIC PAGE MULT BY NUMBER OF PAGES ADD IN START OF GRAPHIC RAM MOVE B TO MSB OF REG ACCD REG D NOW CONTAINS TOP OF PCLEARED SPACE +1 SAVE IN Y COMPARE TOP OF PCLEARED SPACE TO END OF GRAPHIC RAM SUBTRACT START OF BASIC PROGRAM ADD END OF BASIC PROGRAM X=TOP OF PCLEARED SPACE + LENGTH OF BASIC PROGRAM ADD 256 - LEAVE SOME ROOM FOR THE STACK SUBTRACT OUT TOP OF CLEARED SPACE FC ERROR - NO ROOM LEFT ADJUST BASIC S INPUT POINTER SPACE FILLER FOR EXBAS 1.1 GET END OF BASIC PROGRAM STORE NEW END OF BASIC PROGRAM STORE NEW END OF BASIC PROGRAM OUMPARE OLD END TO NEW END BRANCH IF OLD END > NEW END GET BYTE FROM OLD PROGRAM MOVE TO NEW PROGRAM HOVE TO NEW PROGRAM NOVE TO NEW PROGRAM OF DID PROGRAM? NO SAVE NEW STARTING ADDRESS CLEAR BYTE JUST BEFORE PROGRAM
3/832 3/833 3/834 3/835 3/836 3/837 3/838 3/849 3/844 3/844 3/845 3/847 3/847 3/847 3/855	9688 BD B7 ØB 9688 ED B7 ØB 9688 ED B7 9689 27 DC 9699 11 Ø9 9693 24 DB 9695 86 Ø6 9697 3D 9698 DB BC 9690 16 Ø3 BB 9690 16 Ø3 BB 9690 16 Ø3 BB 9690 17 Ø3 B7 9690 18 BB 9680 18 BB 9680 18 BB 9680 18 BB 9680 BB	PCLEAR * THIS C	S JSR TSTB BEQ TSTB BEQ BCC LDA ADDB TFR CMPD BCC CMPD SUBD BCC CMPD BCC CMPU BCC CM	L966D #\$89 L966D #\$96 GRPRAM B,A #\$81 D,Y ENDGRP EFLECTS THE INFAMOUS PCLEAR BUG L966D TXTTAB VARTAB D,X FRETOP L966D L8000 VARTAB VARTAB VARTAB VARTAB VARTAB VARTAB L96D4 ,-U ,-X TXTTAB L96BD TXTTAB L96BL TXTAB LACEF	SET FLAGS BRANCH IF PCLEARØ - FC ERROR TRYING TO CLEAR MORE THAN 8 PAGES? YES ILLEGAL FUNCTION CALL 6 X 256 (1.5K) PER GRAPHIC PAGE MULT BY NUMBER OF PAGES ADD IN START OF GRAPHIC RAM MOVE B TO MSB OF REG ACCD REG D NOW CONTAINS TOP OF PCLEARED SPACE +1 SAVE IN Y COMPARE TOP OF PCLEARED SPACE TO END OF GRAPHIC RAM SUBTRACT START OF BASIC PROGRAM ADD END OF BASIC PROGRAM X=TOP OF PCLEARED SPACE + LENGTH OF BASIC PROGRAM ADD 256 - LEAVE SOME ROOM FOR THE STACK SUBTRACT OUT TOP OF CLEARED SPACE FC ERROR - NO ROOM LEFT ADJUST BASIC SIMULT POINTER SPACE FILLER FOR EXBAS 1.1 GET END OF BASIC PROGRAM STORE NEW END OF BASIC PROGRAM COMPARE OLD END TO NEW END BRANCH IF OLD END NEW END BRANCH IF OLD END NEW END BRANCH IF OLD PROGRAM OVE TO NEW PROGRAM LOCATION AT THE BEGINNING OF OLD PROGRAM? NO SAVE NEW STARTING ADDRESS CLEAR BYTE JUST BEFORE PROGRAM PUT CORRECT ADDRESSES IN FIRST 2 BYTES OF EACH LINE
3/832 3/833 3/834 3/835 3/836 3/837 3/838 3/839 3/841 3/842 3/843 3/844 3/845 3/845 3/845 3/855 3/856 3/855 3/856	968B BD B7 ØB 968E 5D 968F 27 DC 9691 C1 Ø9 9693 24 DB 9695 86 Ø6 9697 3D 9698 DB BC 9698 1F 98 969C C6 Ø1 9696 1Ø 93 B7 9640 1Ø 98 DØ 9658 10 9668 11 93 1B 9668 24 17 9660 18 97 19 9660 18 97 19 9660 67 3F 9660 67 3F 9660 67 3F	* THIS C	R JSR TSTB BEQ CMPB BCC LDA ADDB TFR LDB LDB SUBD SUBD BCC JSR DEC LDA CMPU BCC LDA	L966D #\$89 L966D #\$96 GRPRAM B,A #\$91 D,Y ENDGRP FLECTS THE INFAMOUS PCLEAR BUG L966D TXTTAB VARTAB D,X FRETOP L966D L800D VARTAB VARTAB VARTAB UARTAB	SET FLAGS BRANCH IF PCLEARØ - FC ERROR TRYING TO CLEAR MORE THAN 8 PAGES? YES ILLEGAL FUNCTION CALL 6 X 256 (1.5K) PER GRAPHIC PAGE MULT BY NUMBER OF PAGES ADD IN START OF GRAPHIC RAM MOVE B TO MSB OF REG ACCD REG D NOW CONTAINS TOP OF PCLEARED SPACE +1 SAVE IN Y COMPARE TOP OF PCLEARED SPACE TO END OF CURRENT GRAPHIC PAGE FC ERROR IF TRYING TO CLEAR < END OF GRAPHIC RAM SUBTRACT START OF BASIC PROGRAM ADD END OF BASIC PROGRAM X=TOP OF PCLEARED SPACE + LENGTH OF BASIC PROGRAM ADD 256 - LEAVE SOME ROOM FOR THE STACK SUBTRACT OUT TOP OF CLEARED SPACE FC ERROR - NO ROOM LEFT ADJUST BASIC S INPUT POINTER SPACE FILLER FOR EXBAS 1.1 GET END OF BASIC PROGRAM COMPARE OLD END SANCE PROGRAM COMPARE OLD END SNEW END BRANCH IF OLD END > NEW END STORE NEW END OF BASIC PROGRAM MOVE TO NEW PROGRAM HOVE TO NEW PROGRAM HOVE TO NEW PROGRAM LOCATION AT THE BEGINNING OF OLD PROGRAM? NO SAVE NEW STARTING ADDRESS CLEAR BYTE JUST BEFORE PROGRAM PUT CORRECT ADDRESSES IN FIRST 2 BYTES OF EACH LINE DO PART OF A NEW
3/832 3/833 3/834 3/835 3/836 3/837 3/848 3/849 3/844 3/844 3/844 3/844 3/845 3/846 3/847 3/855 3/855 3/855 3/855 3/856 3/855 3/856	968B BD B7 ØB 968E 5D 968F 27 DC 969F 27 DC 9699 324 DB 9699 324 DB 9699 369 BB BC 9697 3D 9698 DB BC 9696 1F 92 9640 1Ø 93 B7 9643 25 CB 9645 32 19 9647 D3 1B 9648 4C 9646 24 BD 9658 BB W DØ 9668 12 24 BD 9668 BB W DØ 9688 11 93 1B 9689 FB BB 9689 FB 9688 BB W DØ	* THIS C	S JSR TSTB BEQ BEQ BEQ BEQ BEQ BEQ BEQ BEQ BEQ BE	L966D #\$89 L966D #\$99 L966D #\$96 GRPRAM B, A #\$91 D, Y EFNOGRP EFLECTS THE INFAMOUS PCLEAR BUG L966D TXTTAB VARTAB	SET FLAGS BRANCH IF PCLEARØ - FC ERROR TRYING TO CLEAR MORE THAN 8 PAGES? YES ILLEGAL FUNCTION CALL 6 X 256 (1.5K) PER GRAPHIC PAGE MULT BY NUMBER OF PAGES ADD IN START OF GRAPHIC RAM MOVE B TO MSB OF REG ACCD REG D NOW CONTAINS TOP OF PCLEARED SPACE +1 SAVE IN Y COMPARE TOP OF PCLEARED SPACE TO END OF GRAPHIC RAM SUBTRACT START OF BASIC PROGRAM ADD END OF BASIC PROGRAM ADD END OF BASIC PROGRAM ADD 256 - LEAVE SOME ROOM FOR THE STACK SUBTRACT OUT TOP OF CLEARED SPACE FC ERROR - NO ROOM LEFT ADJUST BASIC SIMPUT POINTER SPACE FILLER FOR EXBAS 1.1 GET END OF BASIC PROGRAM STORE NEW END OF BASIC PROGRAM STORE NEW END OF BASIC PROGRAM OUT OF BASIC PROGRAM STORE NEW END OF BASIC PROGRAM ASTORE NEW END OF BASIC PROGRAM STORE NEW END OF BASIC PROGRAM STORE NEW END OF BASIC PROGRAM OUT OF BASIC PROGRAM STORE NEW END OF BASIC PROGRAM STORE NEW END OF BASIC PROGRAM OUT TO NEW PROGRAM LOCATION AT THE BEGINNING OF OLD PROGRAM? NO SAVE NEW STARTING ADDRESS CLEAR BYTE JUST BEFORE PROGRAM PUT CORRECT ADDRESSES IN FIRST 2 BYTES OF EACH LINE DO PART OF A NEW GO BACK TO BASIC S MAIN LOOP
3/832 3/833 3/834 3/835 3/836 3/837 3/838 3/839 3/841 3/842 3/843 3/844 3/845 3/846 3/847 3/848 3/849 3/855 3/856	9688 BD B7 ØB 9688 5D 9687 57 DC 9697 10 Ø9 9693 24 DB 9697 3D 9698 BB BC 9698 1F 98 969C C6 Ø1 9698 17 Ø2 9640 10 93 B7 9640 11 93 B9 9640 12 B9 9650 14 B9 9650 15 B9 9660 17 B9 9660 18 B9 9660 19 F9 9660 19 F9 9660 6F 3F 9660 6F 3F 9660 BD AC EF 9660 BD AC EF 9660 BD AC EF	* THIS C	S JSR TSTB BEQ TSTB BEQ BCC LDA ADDB TFR CMPD BCO CMPD RINCA SUBD BCC CMPU	L966D #\$89 L966D #\$96 #\$96 GRPRAM B,A #\$81 D,Y ENDGRP EFLECTS THE INFAMOUS PCLEAR BUG L966D TXTTAB VARTAB	SET FLAGS BRANCH IF PCLEARØ - FC ERROR TRYING TO CLEAR MORE THAN 8 PAGES? YES ILLEGAL FUNCTION CALL 6 X 256 (1.5K) PER GRAPHIC PAGE MULT BY NUMBER OF PAGES ADD IN START OF GRAPHIC RAM MOVE B TO MSB OF REG ACCD REG D NOW CONTAINS TOP OF PCLEARED SPACE +1 SAVE IN Y COMPARE TOP OF PCLEARED SPACE TO END OF GRAPHIC RAM SUBTRACT START OF BASIC PROGRAM ADD END OF BASIC PROGRAM X=TOP OF PCLEARED SPACE + LENGTH OF BASIC PROGRAM ADD 256 - LEAVE SOME ROOM FOR THE STACK SUBTRACT OUT TOP OF CLEARED SPACE FC ERROR - NO ROOM LEFT ADJUST BASIC S MUDIT POINTER SPACE FILLER FOR EXBAS 1.1 GET END OF BASIC PROGRAM STORE NEW END OF BASIC PROGRAM COMPARE OLD END TO NEW END BRANCH IF OLD END > NEW END BRANCH IF OLD END > NEW END GET BYTE FROM OLD PROGRAM MOVE TO NEW PROGRAM LOCATION AT THE BEGINNING OF OLD PROGRAM? NO SAVE NEW STARTING ADDRESS CLEAR BYTE JUST BEFORE PROGRAM PUT CORRECT ADDRESSES IN FIRST 2 BYTES OF EACH LINE DO PART OF A NEW GO BACK TO BASIC S MAIN LOOP GET START OF BASIC PROGRAM
3/832 3/833 3/834 3/835 3/836 3/837 3/838 3/839 3/841 3/842 3/843 3/844 3/845 3/845 3/845 3/855 3/856	968B BD B7 ØB 968E 5D 968F 27 DC 9691 CI Ø9 9693 24 DB 9695 86 Ø6 9697 3D 9698 DB BC 9698 IF 98 969C CE Ø1 9696 IØ 93 B7 9688 BB RC 9684 BB RC 9684 BB RC 9685 BB RC 9686 BB RC 9686 BB RC 9686 BB RC 9686 BB RC 9688 BB RC	* THIS C	STY	L966D #\$89 L966D #\$96 GRPRAM B,A #\$91 D,Y ENDGRP FLECTS THE INFAMOUS PCLEAR BUG L966D TXTTAB VARTAB D,X FRETOP L966D L8000 VARTAB VARTAB VARTAB UARTAB VARTAB UARTAB VARTAB UARTAB VARTAB UARTAB UARTAB VARTAB UARTAB UARTAB UARTAB VARTAB UARTAB	SET FLAGS BRANCH IF PCLEARØ - FC ERROR TRYING TO CLEAR MORE THAN 8 PAGES? YES ILLEGAL FUNCTION CALL 6 X 256 (1.5K) PER GRAPHIC PAGE MULT BY NUMBER OF PAGES ADD IN START OF GRAPHIC RAM MOVE B TO MSB OF REG ACCD REG D NOW CONTAINS TOP OF PCLEARED SPACE +1 SAVE IN Y COMPARE TOP OF PCLEARED SPACE TO END OF CURRENT GRAPHIC PAGE FC ERROR IF TRYING TO CLEAR < END OF GRAPHIC RAM SUBTRACT START OF BASIC PROGRAM A=TOP OF PCLEARED SPACE + LENGTH OF BASIC PROGRAM ADD END OF BASIC PROGRAM MX=TOP OF PCLEARED SPACE + LENGTH OF BASIC PROGRAM ADD 256 - LEAVE SOME ROOM FOR THE STACK SUBTRACT OUT TOP OF CLEARED SPACE FC ERROR - NO ROOM LEFT ADJUST BASIC S INPUT POINTER SPACE FILLER FOR EXBAS 1.1 GET END OF BASIC PROGRAM COMPARE OLD END SHORE STORE NEW END OF BASIC PROGRAM COMPARE OLD END > NEW END BRANCH IF OLD END > NEW END BRANCH IF OLD END > NEW END GET BYTE FROM OLD PROGRAM MOVE TO NEW PROGRAM HOVE TO NEW PROGRAM MOVE TO NEW PROGRAM LOCATION AT THE BEGINNING OF OLD PROGRAM? NO SAVE NEW STARTING ADDRESS CLEAR BYTE JUST BEFORE PROGRAM PUT CORRECT ADDRESSES IN FIRST 2 BYTES OF EACH LINE DO PART OF A NEW GO BACK TO BASIC S MAIN LOOP GET START OF BASIC PROGRAM STORE NEW STARTING ADDRE
3/832 3/833 3/834 3/835 3/836 3/837 3/849 3/841 3/844 3/844 3/844 3/845 3/846 3/847 3/855	968B BD B7 ØB 968E 5D 968F 27 DC 969F 27 DC 9699 324 DB 9695 86 Ø6 9697 3D 9698 DB BC 9697 BD 9698 DB BC 9690 10 93 B7 9690 16 92 9640 10 93 B7 9648 4C 9640 93 21 9648 4C 9640 93 21 9648 4C 9640 93 21 9648 11 93 19 9658 40 BB BØ DØ 9688 11 93 18 9689 BB BØ DØ 9688 11 93 18 9689 BB BØ DØ 9688 11 93 18 9688 11 93 18 9688 11 93 18 9688 11 93 18 9688 11 93 18 9688 67 FB 9688 11 93 18 9688 67 FB 9688 11 93 18 9688 11 93 18 9688 11 93 18 9688 11 93 18 9688 11 93 18 9688 11 93 18 9688 11 93 18 9688 11 93 18 9688 BD AC EF 9668 BD AC EF	* THIS C	S JSR TSTB BEQP BEQC LDA ADDB TFR BLO COMP RISE BLO STY COMP LDU STX CMPU BNE CMPU BNE CMPU BNE CMPU BNE CMPU BNE STY LDU STX CMPU BNE STY JSR JSR JSR JSR JSR JSR LDU STA CMPU BNE STY LDU STY CLR STY LDU CLR STY LDU STY CLR STY LDU STY CLR STY LDU CLR STY LDU CLR STY LDU CLR STY LDU CLR STY CLR STY LDU CLR STY CLR ST	L966D #\$89 L966D #\$98 L966D #\$96 GRPRAM B, A #\$81 D,Y ENDGRP EFLECTS THE INFAMOUS PCLEAR BUG L966D TXTTAB VARTAB L960D TXTTAB S-81,Y LACEF LAD26 LAD96 TXTTAB LAD26 LAD96 TXTTAB S-61,Y	SET FLAGS BRANCH IF PCLEARØ - FC ERROR TRYING TO CLEAR MORE THAN 8 PAGES? YES ILLEGAL FUNCTION CALL 6 X 256 (1.5K) PER GRAPHIC PAGE MULT BY NUMBER OF PAGES ADD IN START OF GRAPHIC RAM MOVE B TO MSB OF REG ACCD REG D NOW CONTAINS TOP OF PCLEARED SPACE +1 SAVE IN Y COMPARE TOP OF PCLEARED SPACE TO END OF GRAPHIC RAM SUBTRACT START OF BASIC PROGRAM ADD END OF BASIC PROGRAM ADD END OF BASIC PROGRAM ADD 256 - LEAVE SOME ROOM FOR THE STACK SUBTRACT OUT TOP OF CLEARED SPACE FC ERROR I F TRYING TO CHARE STACK SUBTRACT OUT TOP OF CLEARED SPACE FC ERROR - NO ROOM LEFT ADJUST BASIC S INPUT POINTER SPACE FILLER FOR EXBAS 1.1 GET END OF BASIC PROGRAM STORE NEW END OF BASIC PROGRAM COMPARE OLD END TO NEW END BRANCH IF OLD END > NEW END BRANCH IF OLD END > NEW END GET BYTE FROM OLD PROGRAM MOVE TO NEW PROGRAM LOCATION AT THE BEGINNING OF OLD PROGRAM? NO SAVE NEW STARTING ADDRESS LICAR BYTE JUST BEFORE PROGRAM PUT CORRECT ADDRESSES IN FIRST 2 BYTES OF EACH LINE DO PART OF A NEW GO BACK TO BASIC PROGRAM STORE NEW STARTING ADDRESS STORE NEW STARTING ADDRESS STORE NEW STARTING ADDRESS STORE NEW STARTING ADDRESS CLEAR BYTE JUST BEFORE PROGRAM PUT CORRECT ADDRESSES IN FIRST 2 BYTES OF EACH LINE DO PART OF A NEW GO BACK TO BASIC PROGRAM STORE NEW STARTING ADDRESS STORE NEW STARTING ADDRESS STORE NEW STARTING ADDRESS STORE NEW STARTING ADDRESS CLEAR BYTE JUST BEFORE PROGRAM STORE NEW STARTING ADDR
3/832 3/833 3/834 3/835 3/836 3/837 3/838 3/839 3/841 3/842 3/843 3/844 3/845 3/845 3/845 3/855 3/856	968B BD B7 ØB 968E 5D 968F 27 DC 9691 CI Ø9 9693 24 DB 9695 86 Ø6 9697 3D 9698 DB BC 9698 IF 98 969C CE Ø1 9696 IØ 93 B7 9688 BB RC 9684 BB RC 9684 BB RC 9685 BB RC 9686 BB RC 9686 BB RC 9686 BB RC 9686 BB RC 9688 BB RC	* THIS C	STY	L966D #\$89 L966D #\$98 L966D #\$96 GRPRAM B, A #\$81 D,Y ENDGRP EFLECTS THE INFAMOUS PCLEAR BUG L966D TXTTAB VARTAB L960D TXTTAB S-81,Y LACEF LAD26 LAD96 TXTTAB LAD26 LAD96 TXTTAB S-61,Y	SET FLAGS BRANCH IF PCLEARØ - FC ERROR TRYING TO CLEAR MORE THAN 8 PAGES? YES ILLEGAL FUNCTION CALL 6 X 256 (1.5K) PER GRAPHIC PAGE MULT BY NUMBER OF PAGES ADD IN START OF GRAPHIC RAM MOVE B TO MSB OF REG ACCD REG D NOW CONTAINS TOP OF PCLEARED SPACE +1 SAVE IN Y COMPARE TOP OF PCLEARED SPACE TO END OF CURRENT GRAPHIC PAGE FC ERROR IF TRYING TO CLEAR < END OF GRAPHIC RAM SUBTRACT START OF BASIC PROGRAM A=TOP OF PCLEARED SPACE + LENGTH OF BASIC PROGRAM ADD END OF BASIC PROGRAM MX=TOP OF PCLEARED SPACE + LENGTH OF BASIC PROGRAM ADD 256 - LEAVE SOME ROOM FOR THE STACK SUBTRACT OUT TOP OF CLEARED SPACE FC ERROR - NO ROOM LEFT ADJUST BASIC S INPUT POINTER SPACE FILLER FOR EXBAS 1.1 GET END OF BASIC PROGRAM COMPARE OLD END SHORE STORE NEW END OF BASIC PROGRAM COMPARE OLD END > NEW END BRANCH IF OLD END > NEW END BRANCH IF OLD END > NEW END GET BYTE FROM OLD PROGRAM MOVE TO NEW PROGRAM HOVE TO NEW PROGRAM MOVE TO NEW PROGRAM LOCATION AT THE BEGINNING OF OLD PROGRAM? NO SAVE NEW STARTING ADDRESS CLEAR BYTE JUST BEFORE PROGRAM PUT CORRECT ADDRESSES IN FIRST 2 BYTES OF EACH LINE DO PART OF A NEW GO BACK TO BASIC S MAIN LOOP GET START OF BASIC PROGRAM STORE NEW STARTING ADDRE

3Ø73	96DD A7 AØ	STA	, Y+	MOVE TO NEW PROG LOCATION
3074 3075	96DF 10 9C 1B 96E2 26 F7		Y VARTAB L96DB	AT END OF OLD PROGRAM?
3Ø76 3Ø77	96E4 2Ø E5	BRA	L96CB	GO RESET SOME POINTERS
3Ø78 3Ø79	96E6 C6 1E		TION ROUTINE FOR EXBAS GRAPHICS V #\$1E	ARIABLES *
3080 3081	96E8 D7 19 96EA 86 Ø6	STB LDA	TXTTAB #\$06	* SET START OF BASIC PROG TO 1E00
3Ø82 3Ø83	96EC 97 BC	L96EC STA	GRPRAM	 -CONSTANT OFFSET OF \$600 START OF GRAPHICS PAGE TO \$600
3Ø84 3Ø85	96EE 97 BA 96FØ 4F	CLRA	4	PMODE = Ø
3Ø86 3Ø87	96F1 97 B6 96F3 86 10	LDA	PMODE #\$10 HORBYT	SET PMODE TO Ø 16 BYTES/HOR GRAPHIC ROW SAVE IT
3Ø88 3Ø89	96F5 97 B9 96F7 86 Ø3 96F9 97 B2		#\$Ø3 FORCOL	SET FOREGROUND COLOR TO 3 SET FOREGROUND COLOR TO 3
3Ø9Ø	96FB 86 ØC	LDA	#\$ØC	* * * * SET END OF GRAPHICS PAGE TO \$CØØ
3Ø91 3Ø92	96FD 97 B7 96FF 9E 19	STA LDX	TXTTAB	GET START OF PROGRAM
3Ø93 3Ø94	9701 6F 1F 9703 7E AD 19	JMP	\$-01,X LAD19	CLEAR ONE BYTE JUST BEFORE PROGRAM GO DO A NEW
3Ø95 3Ø96			OW MANY BYTES/GRAPHIC ROW AND HOW	
3Ø97 3Ø98	0705 40 05	* BYTES (IN	RES SCREEN FOR THE PMODES. ROWS 256 BYTE BLOCKS) SECOND.	
3099 3100	9706 10 06 9708 20 0C	L97Ø8 FCB	\$10,\$06 \$20,\$0C	PMODE Ø PMODE 1
3101	970A 10 0C 970C 20 18	L97ØA FCB L97ØC FCB	\$10,\$0C \$20,\$18	PMODE 2 PMODE 3
31Ø3 31Ø4	97ØE 2Ø 18	L97ØE FCB	\$20,\$18	PMODE 4
31Ø5 31Ø6	9710 DC C5	L971Ø LDD	EREND - VERBEG) VEREND	GET VERTICAL ENDING ADDRESS
31Ø7 31Ø8	9712 93 BF 9714 24 3B	L9714 BCC	VERBEG L9751	SUBTRACT OUT VERTICAL BEGINNING ADDRESS RETURN IF END >= START CAME STATUS (MINISTRACTOR)
31Ø9 311Ø	9716 34 Ø1 9718 BD 9D C3	JSR	S CC L9DC3	SAVE STATUS (WHICH COORDINATE IS GREATER) NEGATE ACCD IF START COORD > END COORD
3111 3112	971B 35 81	* CALC ABS(H	S CC,PC OREND - HORBEG)	RESTORE STATUS AND RETURN
3113 3114 3115	971D DC C3 971F 93 BD 9721 20 F1		HOREND HORBEG L9714	GET HORIZONTAL END COORD SUB OUT HORIZONTAL START COORD GET ABSOLUTE VALUE
3116 3117	3721 20 11	* PCOPY	23714	del absolute value
3118 3119	9723 8D 1A	PCOPY BSR	L973F	*EVALUATE SOURCE PAGE NUMBER AND RETURN MSB OF *ADDRESS OF START OF PAGE IN ACCD
312Ø 3121	9725 34 Ø6 9727 C6 A5	PSHS	S B,A #\$A5	SAVE PAGE 1 OFFSET TOKEN FOR TO
3122 3123	9727 CO AS 9729 BD B2 6F 972C 8D 11	JSR	LB26F L973F	SYNTAX CHECK FOR TO EVALUATE PAGE NUMBER
3124 3125	972E 35 10 973Ø 1F Ø3		5 X	SET ADDRESS OF SOURCE PAGE TO X ADDRESS OF DESTINATION PAGE TO U
3126 3127	9732 10 8E 03 00 9736 EC 81	LDY L9736 LDD	#\$3ØØ ,X++	MOVE \$300 PAIRS OF BYTES (ONE GRAPHIC PAGE) GET TWO BYTES FROM SOURCE
3128 3129	9738 ED C1 973A 31 3F	STD	,U++ Y \$-01,Y	PUT INTO DESTINATION PAGE DECREMENT COUNTER
313Ø 3131	973C 26 F8 973E 39	BNE RTS	L9736	NOT DONE YET
3132 3133	973F BD B7 ØB		EVALEXPB	EVALUATE EXPRESSION
3134 3135	9742 5D 9743 27 ØD	TSTE BEQ		PAGE ZERO? YES - ILLEGAL FUNCTION CALL
3136 3137	3743 E7 90		FLAKEY ERROR CHECK - IT WILL LET	
3138 3139	9745 D1 19	* THE TOP OF	THE BASIC PROGRAM IN SOME INSTAN 3 TXTTAB	
314Ø 3141	9747 22 Ø9 9749 5A		L9752	FC ERROR IF SO - BAD ERROR CHECK *BUMP PAGE NUMBER DOWN 1, BASIC STARTS AT 1, THIS
3142 3143	974A 86 Ø6	* LDA		*ROUTINE STARTS AT ZERO 6*256 (1.5K) PER GRAPHIC PAGE
3144 3145	974C 3D 974D DB BC	MUL	B GRPRAM	GET OFFSET OF THIS PAGE NUMBER *GET START OF GRAPHIC RAM- ACCB NOW CONTAINS
3146 3147	974F 1E 89	* EXG	А,В	*MSB OF ADDRESS OF THIS PAGE NON ACCD HAS ADDRESS OF PAGE START
3148 3149	9751 39 9752 7E B4 4A	L9751 RTS L9752 JMP	LB44A	ILLEGAL FUNCTION CALL
315Ø 3151		* GET		
3152 3153	9755 5F 9756 20 02	GET CLRE BRA	B L975A	GET FLAG THIS SHOULD BE FCB SKP2 - IT WOULD SAVE A BYTE
3154 3155		* PUT		
3156 3157	9758 C6 Ø1 975A D7 D8		#\$01 VD8	PUT FLAG SAVE GET/PUT FLAG
3158 3159	975C BD Ø1 AØ 975F 81 4Ø	JSR	RVEC22 A #'@'	HOOK INTO RAM CHECK FOR @ SIGN
316Ø 3161	9761 26 Ø2 9763 9D 9F	BNE	L9765 GETNCH	NO @ SIGN GO GET NEXT INPUT CHARACTER
3162 3163	9765 BD 93 8F		L938F	GO EVALUATE START AND END POINTS - STORE START AT (HORBEG, VERSEG), END AT (HOREND, VEREND)
3164 3165	9768 BD B2 6D 976B BD 98 CC	JSR JSR	SYNCOMMA L98CC	SYNTAX CHECK FOR COMMA GET POINTER TO ARRAY DESCRIPTOR
3166 3167	976E 1F 1Ø 977Ø EE 84	TFR LDU	X,D,X	SAVE DESCRIPTOR + 2 IN ACCD SAVE OFFSET TO NEXT ARRAY IN U
3168	9772 33 5E		J \$-02,U	POINT U TO START OF DESCRIPTOR

3169					
	9774 33 CB		LEAU		POINT U TO END OF ARRAY
3170	9776 DF D1		STU		SAVE END OF DATA (END OF ARRAY)
3171	9778 30 02			\$02,X	POINT X TO NUMBER OF DIMENSIONS AND
3172	977A E6 84		LDB	, Х	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		SAVE START OF DATA (START OF ARRAY DATA)
3176	978Ø 96 Ø6			VALTYP	CHECK VARIABLE TYPE
3177	9782 26 CE		BNE	L9752	FC ERROR IF STRING VARIABLE
3178	9784 ØF D4		CLR	VD4	GET/PUT GRAPHIC/ACTION FLAG
3179	9786 9D A5			GETCCH	GET CURRENT INPUT CHAR
3180	9788 27 2D			L97B7	BRANCH IF END OF LINE
3181	978A Ø3 D4		COM	VD4	TOGGLE GET/PUT GRAPHIC/ACTION FLAG
3182	978C BD B2 6D		JSR	SYNCOMMA	SYNTAX CHECK FOR COMMA
3183	978F ØD D8		TST	VD8	CHECK GET/PUT FLAG
3184	9791 26 Ø7			L979A	BRANCH IF PUT
3185	9793 C6 47			#'G'	CHECK FOR FULL GRAPHIC OPTION
3186	9795 BD B2 6F			LB26F	DO A SYNTAX CHECK FOR A G
3187	9798 20 30			L97CA	SKIP AROUND THE NO G OPTION CODE
	979A C6 Ø5	L979A		#\$Ø5	FIVE LEGAL TOKENS AT END OF PUT
3189	979C 8E 98 39			#L9839	POINT X TO LOOK UP TABLE
3190	979F EE 81	L979F		,X++	GET CLEAR BIT ACTION ROUTINE ADDRESS
3191	97A1 10 AE 81		LDY	, X++	GET SET BIT ACTION ROUTINE ADDRESS
3192	97A4 A1 8Ø		CMPA		CHECK FOR ONE OF THE FIVE LEGAL TOKENS
3193	97A6 27 Ø6			L97AE	FOUND ONE
3194	97A8 5A		DECB	10705	CHECKED ALL FIVE?
3195	97A9 26 F4			L979F	NO - KEEP GOING
3196 3197	97AB 7E B2 77	L97AE		LB277	SYNTAX ERROR IF TOKEN NOT FOUND ARRAY SET BIT ACTION ROUTINE ADDRESS
3198	97AE 10 9F D5 97B1 DF D9	L3/AL		VD9	ARRAY CLEAR BIT ACTION ROUTINE ADDRESS
3199	97B1 DF D9			GETNCH	GET INPUT CHAR FROM BASIC
				L97CA	
3200 3201	97B5 2Ø 13		BRA	EJ/UN	SKIP AROUND THE NO G OPTION CODE
3201		* NO C	ОРТТ	ON OR ACTION SPECIFIED BY BASIC :	INPIIT I INF
3202	97B7 C6 F8	^ NO G L97B7		#\$F8	BOTTOM 3 BITS MASK (8 PIXELS/BYTE)
3203	97B9 96 B6	LJ/D/		PMODE	GET CURRENT PMODE
3204	97BB 46		RORA		BIT Ø TO CARRY
3206	97BC 24 Ø2			L97CØ	BRANCH IF PMODE Ø,2,4 (2 COLOR)
3207	97BE C6 FC			#\$FC	BOTTOM 2 BITS MASK (4 COLOR MODE - 4 PIXELS/BYTE)
3208	97CØ 1F 98	L97CØ	TFR		SAVE ACCB IN ACCA
3209	97C2 D4 BE	L3700		HORBEG+1	*
3210	97C4 D7 BE			HORBEG+1	* MASK THE PIXEL COUNTER (BITS Ø,1=
3211	97C6 94 C4			HOREND+1	* 4 COLOR, BITS Ø-2=2 COLOR) BITS OFF
	97C8 97 C4			HOREND+1	* THE HORIZONTAL DIFFERENCE
	97CA BD 97 1D	L97CA		L971D	CALC HORIZ DIFFERENCE ABS(HOREND-HORBEG)
3214	97CD 24 Ø4			L97D3	BRANCH IF END > START
3215	97CF 9E C3			HOREND	* MAKE START = END IF
3216	97D1 9F BD			HORBEG	* START > END
3217	97D3 DD C3	L97D3		HOREND	SAVE HORIZ DIFFERENCE
	97D5 BD 97 10			L971Ø	CALC VERT DIFFERENCE ABS (VEREND-VERBEG)
3219	97D8 24 Ø4				BRANCH IF END > START
3220	97DA 9E C5			VEREND	* MAKE START = END IF
3221	97DC 9F BF		STX	VERBEG	* START > END
3221 3222	97DC 9F BF 97DE DD C5	L97DE	STX STD		* START > END SAVE VERT DIFFERENCE
	97DC 9F BF 97DE DD C5 97EØ 96 B6	L97DE		VERBEG VEREND PMODE	* START > END SAVE VERT DIFFERENCE * GET PMODE BIT Ø
3222	97DE DD C5	L97DE	STD	VEREND	SAVE VERT DIFFERENCE
3222 3223	97DE DD C5 97EØ 96 B6	L97DE	STD LDA RORA	VEREND	SAVE VERT DIFFERENCE * GET PMODE BIT Ø
3222 3223 3224	97DE DD C5 97EØ 96 B6 97E2 46	L97DE	STD LDA RORA LDD	VEREND PMODE	SAVE VERT DIFFERENCE * GET PMODE BIT Ø * TO THE CARRY FLAG
3222 3223 3224 3225 3226 3227	97DE DD C5 97EØ 96 B6 97E2 46 97E3 DC C3	L97DE	STD LDA RORA LDD BCC	VEREND PMODE HOREND L97EB HOREND	SAVE VERT DIFFERENCE * GET PMODE BIT Ø * TO THE CARRY FLAG GET HORIZ DIFFERENCE BRANCH IF PMODE = Ø,2,4(2 COLOR) * DOUBLE HORIZ DIFF - 2X AS MANY BYTES FOR
3222 3223 3224 3225 3226 3227 3228	97DE DD C5 97EØ 96 86 97E2 46 97E3 DC C3 97E5 24 Ø4 97E7 D3 C3 97E9 DD C3		STD LDA RORA LDD BCC ADDD STD	VEREND PMODE HOREND L97EB HOREND HOREND	SAVE VERT DIFFERENCE * GET PMODE BIT Ø * TO THE CARRY FLAG GET HORIZ DIFFERENCE BRANCH IF PMODE = 0,2,4(2 COLOR) * DOUBLE HORIZ DIFF - 2X AS MANY BYTES FOR * NUMBER OF PIXELS IN PMODES 1,3
3222 3223 3224 3225 3226 3227 3228 3229	97DE DD C5 97E0 96 86 97E2 46 97E3 DC C3 97E5 24 04 97E7 D3 C3 97E9 DD C3 97EB BD 94 20	L97DE L97EB	STD LDA RORA LDD BCC ADDD STD JSR	VEREND PMODE HOREND L97EB HOREND HOREND L942Ø	SAVE VERT DIFFERENCE * GET PMODE BIT Ø * TO THE CARRY FLAG GET HORIZ DIFFERENCE BRANCH IF PMODE = Ø,2,4(2 COLOR) * DOUBLE HORIZ DIFF - 2X AS MANY BYTES FOR * NUMBER OF PIXELS IN PMODES 1,3 NORMALIZE DIFFERENCES
3222 3223 3224 3225 3226 3227 3228 3229 3230	970E DD C5 97E0 96 B6 97E2 46 97E3 DC C3 97E5 24 04 97E7 D3 C3 97E9 DD C3 97EB BD 94 20 97EB CC C3		STD LDA RORA LDD BCC ADDD STD JSR LDD	VEREND PMODE HOREND L97EB HOREND HOREND L942Ø HOREND	SAVE VERT DIFFERENCE * GET PMODE BIT Ø * TO THE CARRY FLAG GET HORIZ DIFFERENCE BRANCH IF PMODE = 0,2,4(2 COLOR) * DOUBLE HORIZ DIFF - 2X AS MANY BYTES FOR * NUMBER OF PIXELS IN PMODES 1,3
3222 3223 3224 3225 3226 3227 3228 3229 3230 3231	97DE DD C5 97E0 96 86 97E2 46 97E3 DC C3 97E5 24 04 97E7 D3 C3 97E9 DD C3 97EB BD 94 20 97EB DC C3 97EB DC C3		STD LDA RORA LDD BCC ADDD STD JSR LDD LDX	VEREND PMODE HOREND L97EB HOREND HOREND L942Ø HOREND VEREND	SAVE VERT DIFFERENCE * GET PMODE BIT Ø * TO THE CARRY FLAG GET HORIZ DIFFERENCE BRANCH IF PMODE = Ø,2,4(2 COLOR) * DOUBLE HORIZ DIFF - 2X AS MANY BYTES FOR * NUMBER OF PIXELS IN PMODES 1,3 NORMALIZE DIFFERENCES
3222 3223 3224 3225 3226 3227 3228 3229 3230 3231 3232	97DE DD C5 97E0 96 86 97E3 DC C3 97E5 24 04 97E7 D3 C3 97E9 DD C3 97EB BD 94 20 97EB C C3 97EB C C3 97E9 DC C3		STD LDA RORA LDD BCC ADDD STD JSR LDD LDX LEAX	VEREND PMODE HOREND L97EB HOREND HOREND L942Ø HOREND VEREND \$01, X	SAVE VERT DIFFERENCE * GET PMODE BIT Ø * TO THE CARRY FLAG GET HORIZ DIFFERENCE BRANCH IF PMODE = Ø,2,4(2 COLOR) * DOUBLE HORIZ DIFF - 2X AS MANY BYTES FOR * NUMBER OF PIXELS IN PMODES 1,3 NORMALIZE DIFFERENCES GET HORIZ DIFFERENCE * *
3222 3223 3224 3225 3226 3227 3228 3229 3230 3231 3232 3233	97DE DD C5 97E0 96 B6 97E2 46 97E3 DC C3 97E5 24 04 97E7 D3 C3 97E9 DD C3 97EB BD 94 20 97EE DC C3 97F0 9E C5 97F2 30 01 97F4 9F C5		STD LDA RORA LDD BCC ADDD STD JSR LDD LDX LEAX STX	VEREND PMODE HOREND L97EB HOREND HOREND L9420 HOREND VEREND \$401,X VEREND	SAVE VERT DIFFERENCE * GET PMODE BIT Ø * TO THE CARRY FLAG GET HORIZ DIFFERENCE BRANCH IF PMODE = Ø,2,4(2 COLOR) * DOUBLE HORIZ DIFF - 2X AS MANY BYTES FOR * NUMBER OF PIXELS IN PMODES 1,3 NORMALIZE DIFFERENCES GET HORIZ DIFFERENCE * * ADD 1 TO VERT DIFFERENCE
3222 3223 3224 3225 3226 3227 3228 3229 3230 3231 3232 3233 3234	97DE DD C5 97E0 96 86 97E2 46 97E3 DC C3 97E5 24 04 97E7 DD C3 97E9 DD C3 97E9 DC C3 97E8 BD 94 20 97E8 CC C3 97F0 9E C5 97F2 30 01 97F4 9F C5		STD LDA RORA LDD BCC ADDD STD JSR LDD LDX LEAX STX TST	VEREND PMODE HOREND L97EB HOREND HOREND L942Ø HOREND VEREND \$Ø1, X VEREND VD4	SAVE VERT DIFFERENCE * GET PMODE BIT Ø * TO THE CARRY FLAG GET HORIZ DIFFERENCE BRANCH IF FMODE = 0,2,4(2 COLOR) * DOUBLE HORIZ DIFF - 2X AS MANY BYTES FOR * NUMBER OF PIXELS IN PMODES 1,3 NORMALIZE DIFFERENCES GET HORIZ DIFFERENCE * * * ADD 1 TO VERT DIFFERENCE = CHECK FOR G OPTION OR GET ACTION
3222 3223 3224 3225 3226 3227 3228 3229 3230 3231 3232 3233 3234 3235	97DE DD C5 97E0 96 86 97E3 DC C3 97E5 24 04 97E7 D3 C3 97E5 DD C3 97EB DD C3 97EB BD 94 20 97EE DC C3 97E0 C5 97F0 30 01 97F4 9F C5 97F6 0D D4 97F8 26 58		STD LDA RORA LDD BCC ADDD STD JSR LDD LDX LEAX STX TST BNE	VEREND PMODE HOREND L97EB HOREND HOREND L9420 HOREND VEREND \$401,X VEREND	SAVE VERT DIFFERENCE * GET PMODE BIT Ø * TO THE CARRY FLAG GET HORIZ DIFFERENCE BRANCH IF PMODE = Ø,2,4(2 COLOR) * DOUBLE HORIZ DIFF - 2X AS MANY BYTES FOR * NUMBER OF PIXELS IN PMODES 1,3 NORMALIZE DIFFERENCES GET HORIZ DIFFERENCE * * ADD 1 TO VERT DIFFERENCE
3222 3223 3224 3225 3226 3227 3228 3229 3230 3231 3232 3233 3234 3235 3236	97DE DD C5 97E0 96 B6 97E2 46 97E3 DC C3 97E5 24 04 97E7 D3 C3 97E9 DD C3 97EB BD 94 20 97EE DC C3 97F0 9E C5 97F2 30 01 97F4 9F C5 97F6 0D D4 97F8 26 58 97F4 44		STD LDA RORA LDD BCC ADDD STD JSR LDD LDX LEAX STX TST BNE LSRA	VEREND PMODE HOREND L97EB HOREND HOREND L942Ø HOREND VEREND \$Ø1, X VEREND VD4	SAVE VERT DIFFERENCE * GET PMODE BIT Ø * TO THE CARRY FLAG GET HORIZ DIFFERENCE BRANCH IF FMODE = 0,2,4(2 COLOR) * DOUBLE HORIZ DIFF - 2X AS MANY BYTES FOR * NUMBER OF PIXELS IN PMODES 1,3 NORMALIZE DIFFERENCES GET HORIZ DIFFERENCE * * * ADD 1 TO VERT DIFFERENCE = CHECK FOR G OPTION OR GET ACTION
3222 3223 3224 3225 3226 3227 3228 3229 3230 3231 3232 3233 3234 3235 3236 3237	97DE DD C5 97E0 96 86 97E2 46 97E3 DC C3 97E5 24 04 97E7 DD C3 97E9 DD C3 97E9 DC C3 97E9 DC C3 97E9 DC C3 97E9 DC C3 97F0 9E C5 97F2 30 01 97F4 9F C5 97F6 0D D4 97F8 26 58 97FA 44		STD LDA RORA LDD BCC ADDD STD JSR LDD LDX LEAX STX TST BNE LSRA RORB	VEREND PMODE HOREND L97EB HOREND HOREND L942Ø HOREND VEREND \$Ø1, X VEREND VD4	SAVE VERT DIFFERENCE * GET PMODE BIT Ø * TO THE CARRY FLAG GET HORIZ DIFFERENCE BRANCH IF FMODE = 0,2,4(2 COLOR) * DOUBLE HORIZ DIFF - 2X AS MANY BYTES FOR * NUMBER OF PIXELS IN PMODES 1,3 NORMALIZE DIFFERENCES GET HORIZ DIFFERENCE * * * ADD 1 TO VERT DIFFERENCE = CHECK FOR G OPTION OR GET ACTION
3222 3223 3224 3225 3226 3227 3238 3231 3232 3233 3234 3235 3236 3237 3238	970E DD C5 97E0 96 86 97E2 46 97E3 DC C3 97E5 24 04 97E7 D3 C3 97E9 DD C3 97EB DD 94 20 97EE DC C3 97F0 9E C5 97F6 49F C5 97F6 0D D4 97F8 44 97FB 56 97FC 44		STD LDA RORA LDD BCC ADDD STD JSR LDD LDX LEAX STX TST BNE LSRA RORB LSRA	VEREND PMODE HOREND L97EB HOREND HOREND L942Ø HOREND VEREND \$Ø1, X VEREND VD4	SAVE VERT DIFFERENCE * GET PMODE BIT Ø * TO THE CARRY FLAG GET HORIZ DIFFERENCE BRANCH IF FMODE = 0,2,4(2 COLOR) * DOUBLE HORIZ DIFF - 2X AS MANY BYTES FOR * NUMBER OF PIXELS IN PMODES 1,3 NORMALIZE DIFFERENCES GET HORIZ DIFFERENCE * * * ADD 1 TO VERT DIFFERENCE = CHECK FOR G OPTION OR GET ACTION
3222 3223 3224 3225 3226 3227 3228 3229 3230 3231 3232 3233 3234 3235 3236 3237 3238 3238	97DE DD C5 97E0 96 B6 97E2 46 97E3 DC C3 97E5 24 04 97E7 DD C3 97E9 DD C3 97EB DD 94 20 97EE DC C3 97F0 9E C5 97F2 30 01 97F4 9F C5 97F6 0D D4 97F8 26 58 97FA 44 97FB 56		STD LDA RORA LDD BCC ADDD STD JSR LDD LDX LEAX STX TST BNE LSRA RORB LSRA RORB	VEREND PMODE HOREND L97EB HOREND HOREND L942Ø HOREND VEREND \$Ø1, X VEREND VD4	SAVE VERT DIFFERENCE * GET PMODE BIT Ø * TO THE CARRY FLAG GET HORIZ DIFFERENCE BRANCH IF FMODE = 0,2,4(2 COLOR) * DOUBLE HORIZ DIFF - 2X AS MANY BYTES FOR * NUMBER OF PIXELS IN PMODES 1,3 NORMALIZE DIFFERENCES GET HORIZ DIFFERENCE * * * ADD 1 TO VERT DIFFERENCE = CHECK FOR G OPTION OR GET ACTION
3222 3223 3224 3225 3226 3227 3238 3231 3232 3233 3234 3235 3236 3237 3238 3239 3240	97DE DD C5 97E0 96 86 97E2 46 97E3 DC C3 97E5 24 04 97E7 DD C3 97E9 DD C3 97E9 DC C3 97E9 CC C3 97F0 9E C5 97F2 30 01 97E4 9F C5 97F6 DD D4 97F8 26 58 97FA 44 97FB 56 97FC 44		STD LDA RORA LDD BCC ADDD STD JSR LDD LDX LEAX STX TST BNE LSRA RORB LSRA RORB LSRA	VEREND PMODE HOREND L97EB HOREND HOREND L942Ø HOREND VEREND \$Ø1,X VEREND VD4	SAVE VERT DIFFERENCE * GET PMODE BIT Ø * TO THE CARRY FLAG GET HORIZ DIFFERENCE BRANCH IF FMODE = 0,2,4(2 COLOR) * DOUBLE HORIZ DIFF - 2X AS MANY BYTES FOR * NUMBER OF PIXELS IN PMODES 1,3 NORMALIZE DIFFERENCES GET HORIZ DIFFERENCE * * ADD 1 TO VERT DIFFERENCE = CHECK FOR G OPTION OR GET ACTION = AND BRANCH IF GIVEN * * * * * * * * * * * * *
3222 3223 3224 3225 3226 3227 3230 3231 3232 3233 3234 3235 3236 3237 3238 3239 3240 3240 3241	970E DD C5 97E0 96 86 97E2 46 97E3 DC C3 97E5 24 04 97E7 D3 C3 97E9 DD C3 97EB DD 4 20 97EE DC C3 97F0 9E C5 97F6 0D D4 97F8 26 58 97FA 44 97FB 56 97FE 44 97FF 56		STD LDA RORA LDD BCC ADDD STD JSR LDD LDX LEAX STX TST BNE LSRA RORB LSRA RORB	VEREND PMODE HOREND L97EB HOREND HOREND L9420 HOREND VEREND \$601,X VEREND VD4 L9852	SAVE VERT DIFFERENCE * GET PMODE BIT Ø * TO THE CARRY FLAG GET HORIZ DIFFERENCE BRANCH IF PMODE = Ø,2,4(2 COLOR) * DOUBLE HORIZ DIFF - 2X AS MANY BYTES FOR * NUMBER OF PIXELS IN PMODES 1,3 NORMALIZE DIFFERENCES GET HORIZ DIFFERENCE * * ADD 1 TO VERT DIFFERENCE = CHECK FOR G OPTION OR GET ACTION = AND BRANCH IF GIVEN * * * DIVIDE HORIZONTAL DIFFERENCE BY 8
3222 3223 3224 3225 3226 3227 3228 3239 3231 3232 3233 3234 3235 3236 3237 3238 3239 3240 3241 3241	97DE DD C5 97E0 96 B6 97E2 46 97E3 DC C3 97E5 24 04 97E7 D3 C3 97E9 DD C3 97EB DD 94 20 97EE DC C3 97F0 9E C5 97F2 30 01 97F4 9F C5 97F6 0D D4 97F8 26 58 97FA 44 97FB 56 97FC 44 97FB 56 97FF 56 9880 C3 00 01		STD LDA RORA LDD BCC ADDD STD JSR LDX LEAX STX TST BNE LSRA RORB LSRA RORB LSRA RORB LSRA RORB	VEREND PMODE HOREND L97EB HOREND HOREND L9420 HOREND VEREND \$01, X VEREND VD4 L9852	SAVE VERT DIFFERENCE * GET PMODE BIT Ø * TO THE CARRY FLAG GET HORIZ DIFFERENCE BRANCH IF PMODE = Ø,2,4(2 COLOR) * DOUBLE HORIZ DIFF - 2X AS MANY BYTES FOR * NUMBER OF PIXELS IN PMODES 1,3 NORMALIZE DIFFERENCES GET HORIZ DIFFERENCE * * ADD 1 TO VERT DIFFERENCE = CHECK FOR G OPTION OR GET ACTION = AND BRANCH IF GIVEN * * * DIVIDE HORIZONTAL DIFFERENCE BY 8 ADD ONE TO QUOTIENT
3222 3223 3224 3225 3226 3227 3238 3239 3231 3232 3233 3234 3235 3236 3237 3238 3239 3240 3241 3242 3243	97DE DD C5 97E0 96 86 97E2 46 97E3 DC C3 97E5 24 04 97E7 DD C3 97E9 DD C3 97E9 DC C3 97E9 CC C3 97F0 9E C5 97F2 30 01 97E4 9F C5 97F6 0D D4 97F8 26 58 97FA 44 97FB 56 97FC 44 97FF 56 97FE 44 97FF 56 9800 C3 00 01		STD LDA RORA LDD BCC ADDD STD JSR LDD LDX LEAX STX TST BNE LSRA RORB LSRA RORB LSRA RORB ADDD STD	VEREND PMODE HOREND L97EB HOREND HOREND L942Ø HOREND VEREND \$Ø1, X VEREND VD4 L9852	SAVE VERT DIFFERENCE * GET PMODE BIT Ø * TO THE CARRY FLAG GET HORIZ DIFFERENCE BRANCH IF FMODE = 0,2,4(2 COLOR) * DOUBLE HORIZ DIFF - 2X AS MANY BYTES FOR * NUMBER OF PIXELS IN PMODES 1,3 NORMALIZE DIFFERENCES GET HORIZ DIFFERENCE * * ADD 1 TO VERT DIFFERENCE = CHECK FOR G OPTION OR GET ACTION = AND BRANCH IF GIVEN * * DIVIDE HORIZONTAL DIFFERENCE BY 8 ADD ONE TO QUOTIENT SAVE NEW HOR DIFFERENCE
3222 3223 3224 3225 3226 3227 3228 3239 3231 3232 3233 3234 3235 3236 3237 3238 3239 3240 3241 3242 3243 3244	97DE DD C5 97E0 96 B6 97E2 46 97E3 DC C3 97E5 24 04 97E7 D3 C3 97E9 DD C3 97EB DD 94 20 97EE DC C3 97F0 9E C5 97F2 30 01 97F4 9F C5 97F6 0D D4 97F8 26 58 97FA 44 97FB 56 97FC 44 97FB 56 97FF 56 9880 C3 00 01		STD LDA RORA LDD BCC ADDD STD JSR LDD LDX LEAX STX TST BNE LSRA RORB LSRA RORB LSRA RORB ADDD STD	VEREND PMODE HOREND L97EB HOREND HOREND L9420 HOREND VEREND \$01, X VEREND VD4 L9852	SAVE VERT DIFFERENCE * GET PMODE BIT Ø * TO THE CARRY FLAG GET HORIZ DIFFERENCE BRANCH IF PMODE = Ø,2,4(2 COLOR) * DOUBLE HORIZ DIFF - 2X AS MANY BYTES FOR * NUMBER OF PIXELS IN PMODES 1,3 NORMALIZE DIFFERENCES GET HORIZ DIFFERENCE * * ADD 1 TO VERT DIFFERENCE = CHECK FOR G OPTION OR GET ACTION = AND BRANCH IF GIVEN * * DIVIDE HORIZONTAL DIFFERENCE BY 8 ADD ONE TO QUOTIENT SAVE NEW HOR DIFFERENCE *CONVERT (HORBEG, VERSEG) INTO ABSOLUTE SCREEN
3222 3223 3224 3225 3226 3227 3238 3231 3232 3233 3234 3235 3236 3237 3238 3239 3240 3241 3242 3243 3244 3244 3244	97DE DD C5 97E0 96 B6 97E2 46 97E3 DC C3 97E5 24 04 97E7 DG C3 97E9 DD C3 97EB DD 94 20 97EE DC C3 97F0 9E C5 97F2 30 01 97F4 9F C5 97F6 0D D4 97F8 26 58 97FA 44 97FB 56 97FC 44 97FB 56 97FE 44 97FB 56 97FE 44 97FB 56 9800 C3 00 01 9803 DD C3 9805 BD 92 98		STD LDA RORA LDD BCC ADDD JSR LDD LDX LEAX STX TST BNE LSRA RORB LSRA	VEREND PMODE HOREND L97EB HOREND HOREND L942Ø HOREND VEREND \$Ø1, X VEREND VD4 L9852	SAVE VERT DIFFERENCE * GET PMODE BIT Ø * TO THE CARRY FLAG GET HORIZ DIFFERENCE BRANCH IF FMODE = 0,2,4(2 COLOR) * DOUBLE HORIZ DIFF - 2X AS MANY BYTES FOR * NUMBER OF PIXELS IN PMODES 1,3 NORMALIZE DIFFERENCES GET HORIZ DIFFERENCE * * ADD 1 TO VERT DIFFERENCE = CHECK FOR G OPTION OR GET ACTION = AND BRANCH IF GIVEN * * DIVIDE HORIZONTAL DIFFERENCE BY 8 ADD ONE TO QUOTIENT SAVE NEW HOR DIFFERENCE
3222 3223 3224 3225 3226 3227 3238 3239 3231 3232 3233 3234 3235 3236 3237 3238 3240 3241 3242 3243 3244 3244 3244 3244	97DE DD C5 97E0 96 86 97E2 46 97E3 DC C3 97E5 24 04 97E7 DD C3 97E9 DD C3 97E9 DC C3 97E9 DC C3 97F0 9E C5 97F2 30 01 97E4 9F C5 97F6 0D D4 97F8 26 58 97FA 44 97FB 56 97FC 44 97FB 56 97FC 44 97FF 56 97FE 44 97FF 56 9800 C3 00 01 9803 DD C3 9805 BD 92 98	L97EB	STD LDA RORA RORA BCC ADDD BCC ADDD LDX LEAX STX TST BNE LSRA RORB LSRA RORB ADDD STD JSR LDB	VEREND PMODE HOREND L97EB HOREND HOREND L942Ø HOREND VEREND \$Ø1, X VEREND VD4 L9852 #1 HOREND L9298 HOREND+1	SAVE VERT DIFFERENCE * GET PMODE BIT Ø * TO THE CARRY FLAG GET HORIZ DIFFERENCE BRANCH IF PMODE = Ø,2,4(2 COLOR) * DOUBLE HORIZ DIFF - 2X AS MANY BYTES FOR * NUMBER OF PIXELS IN PMODES 1,3 NORMALIZE DIFFERENCES GET HORIZ DIFFERENCE * * ADD 1 TO VERT DIFFERENCE = CHECK FOR G OPTION OR GET ACTION = AND BRANCH IF GIVEN * * * * * DIVIDE HORIZONTAL DIFFERENCE BY 8 ADD ONE TO QUOTIENT SAVE NEW HOR DIFFERENCE * *CONVERT (HORBEG, VERSEG) INTO ABSOLUTE SCREEN *POS (X) AND PIXEL MASK (ACCA)
3222 3223 3224 3225 3226 3227 3238 3231 3232 3233 3234 3235 3236 3237 3238 3249 3249 3241 3242 3243 3244 3245 3245 3246 3247 3248	97DE DD C5 97E0 96 B6 97E2 46 97E3 DC C3 97E5 24 04 97E7 DC C3 97E9 DD C3 97E9 DD C3 97EB DD 94 20 97EE DC C3 97F0 9E C5 97F2 30 01 97F4 9F C5 97F6 0D D4 97F8 26 58 97FC 44 97FB 56 97FC 44 97FB 56 97FC 44 97FF 56 97FC 44 97FF 56 9800 C3 00 01 9803 DD C3 9805 BD 92 98 9808 BC C4 980A 34 10 980C 0D D8	L97EB	STD LDA RORA LDD BCC ADDD JSR LDD LDX LEAX STX TST BNE LSRA RORB LSRA	VEREND PMODE HOREND L97EB HOREND HOREND L942Ø HOREND VEREND \$Ø1,X VEREND VD4 L9852 #1 HOREND L9298 HORENDL	SAVE VERT DIFFERENCE * GET PMODE BIT Ø * TO THE CARRY FLAG GET HORIZ DIFFERENCE BRANCH IF PMODE = Ø,2,4(2 COLOR) * DOUBLE HORIZ DIFF - 2X AS MANY BYTES FOR * NUMBER OF PIXELS IN PMODES 1,3 NORMALIZE DIFFERENCE * * ADD 1 TO VERT DIFFERENCE = CHECK FOR G OPTION OR GET ACTION = AND BRANCH IF GIVEN * * * * DIVIDE HORIZONTAL DIFFERENCE BY 8 ADD ONE TO QUOTIENT SAVE NEW HOR DIFFERENCE *CONVERT (HORBEG, VERSEG) INTO ABSOLUTE SCREEN *POS (X) AND PIXEL MASK (ACCA) GET HORIZ DIFFERENCE SAVE SCREEN POSITION CHECK THE GET/PUT FLAG
3222 3223 3224 3225 3226 3227 3238 3231 3232 3233 3234 3235 3236 3237 3238 3240 3241 3242 3243 3244 3244 3244 3244 3244	970E DD C5 97E0 96 86 97E2 46 97E3 DC C3 97E5 24 04 97E7 DG C3 97E9 DD C3 97E9 DC C3 97E9 DC C3 97F0 9E C5 97F2 30 01 97E4 9F C5 97F6 0D D4 97F8 26 58 97FA 44 97FB 56 97FC 44 97FB 56 97FC 44 97FF 56 97FE 44 97FF 56 9800 C3 00 01 9803 DD C3 9805 BD 92 98 9808 D6 C4 980A 34 10 980C 0D D8 980C 27 21	L97EB L98Ø8	STD LDA RORA LDD BCC ADDD JSR LDD JSR LDD LEAX STX TST BNE LSRA RORB LSRA RORB LSRA ROBD STD JSR LDB PSHS STD JSR LDB PSHS STD JSR LDB PSHS STD STD BEQ	VEREND PMODE HOREND L97EB HOREND HOREND L942Ø HOREND VEREND \$Ø1, X VEREND VD4 L9852 #1 HOREND L9298 HOREND+1 X VD8 L9831	SAVE VERT DIFFERENCE * GET PMODE BIT Ø * TO THE CARRY FLAG GET HORIZ DIFFERENCE BRANCH IF PMODE = Ø,2,4(2 COLOR) * DOUBLE HORIZ DIFF - 2X AS MANY BYTES FOR * NUMBER OF PIXELS IN PMODES 1,3 NORMALIZE DIFFERENCES GET HORIZ DIFFERENCE * * ADD 1 TO VERT DIFFERENCE = CHECK FOR G OPTION OR GET ACTION = AND BRANCH IF GIVEN * * DIVIDE HORIZONTAL DIFFERENCE BY 8 ADD ONE TO QUOTIENT SAVE NEW HOR DIFFERENCE *CONVERT (HORBEG, VERSEG) INTO ABSOLUTE SCREEN *POS (X) AND PIXEL MASK (ACCA) GET HORIZ DIFFERENCE SAVE SCREEN POSITION CHECK THE GET/PUT FLAG BRANCH IF GET
3222 3223 3224 3225 3227 3228 3229 3230 3231 3232 3233 3234 3235 3236 3237 3238 3241 3242 3243 3241 3242 3243 3244 3245 3244 3245 3246 3247 3248 3247 3248 3247 3248 3258	97DE DD C5 97E0 96 B6 97E0 96 B6 97E2 46 97E3 DC C3 97E5 24 04 97E7 D3 C3 97E9 DD C3 97EB DD 94 20 97EE DC C3 97F0 9E C5 97F2 30 01 97F4 9F C5 97F6 0D D4 97F8 26 58 97FA 44 97FB 56 97FC 44 97FD 56 9800 C3 00 01 9803 DD C3 9805 BD 92 98 9808 D6 C4 9800 D D8 9806 0D D8	L97EB L98Ø8	STD LDA RORA LDD BCC ADDD STD JSR LDX LEAX STX BNE LSRA RORB LSRA RORB LSRA RORB LSRA ESTD JSR LSRA BDDD STD LSRA BDDD STD STD BEG BSR BSR	VEREND PMODE HOREND L97EB HOREND HOREND L9420 HOREND VEREND \$Ø1,X VEREND VD4 L9852 #1 HOREND L9298 HOREND L9298 HOREND+1 X VD8 L9831 L9823	SAVE VERT DIFFERENCE * GET PMODE BIT Ø * TO THE CARRY FLAG GET HORIZ DIFFERENCE BRANCH IF PMODE = Ø,2,4(2 COLOR) * DOUBLE HORIZ DIFF - 2X AS MANY BYTES FOR * NUMBER OF PIXELS IN PMODES 1,3 NORMALIZE DIFFERENCES GET HORIZ DIFFERENCE * * ADD 1 TO VERT DIFFERENCE = CHECK FOR G OPTION OR GET ACTION = AND BRANCH IF GIVEN * * DIVIDE HORIZONTAL DIFFERENCE BY 8 ADD ONE TO QUOTIENT SAVE NEW HOR DIFFERENCE *CONVERT (HORBEG, VERSEG) INTO ABSOLUTE SCREEN *POS (X) AND PIXEL MASK (ACCA) GET HORIZ DIFFERENCE SAVE SCREEN POSITION CHECK THE GET/PUT FLAG BRANCH IF GET INCREMENT ARRAY DATA POINTER
3222 3223 3224 3225 3226 3227 3238 3231 3232 3233 3234 3235 3236 3237 3240 3241 3242 3243 3244 3245 3246 3247 3248 3249 3259 3259	97DE DD C5 97E0 96 B6 97E2 46 97E3 DC C3 97E5 24 04 97E7 D3 C3 97E9 DD C3 97EB DD 94 20 97EE DC C3 97F0 9E C5 97F2 30 01 97F4 9F C5 97F6 0D D4 97F8 26 58 97FC 44 97FB 56 9800 C3 00 01 9803 DD C3 9808 DG C4	L97EB L98Ø8	STD LDA LDD BCC STD JSR LDD LDX LSTX TST BNE LSRA RORB RORB RORB RORB RORB RORB RORB RO	VEREND PMODE HOREND L97EB HOREND HOREND L9420 HOREND VEREND \$01, X VEREND VD4 L9852 #1 HOREND L9298 HOREND L9298 HOREND L9298 HOREND L9298 L9831 L9823 ,U	SAVE VERT DIFFERENCE * GET PMODE BIT Ø * TO THE CARRY FLAG GET HORIZ DIFFERENCE BRANCH IF PMODE = Ø,2,4(2 COLOR) * DOUBLE HORIZ DIFF - 2X AS MANY BYTES FOR * NUMBER OF PIXELS IN PMODES 1,3 NORMALIZE DIFFERENCE SECT HORIZ DIFFERENCE * * ADD 1 TO VERT DIFFERENCE = CHECK FOR G OPTION OR GET ACTION = AND BRANCH IF GIVEN * * * * * * * * * * * * *
3222 3223 3224 3225 3226 3227 3238 3231 3232 3233 3234 3235 3236 3237 3238 3241 3242 3243 3244 3244 3245 3246 3247 3248 3259 3250 3250	97DE DD C5 97E0 96 86 97E2 46 97E3 DC C3 97E5 24 04 97E7 D3 C3 97E9 DD C3 97EB BD 94 20 97EE DC C3 97F0 9E C5 97F6 0D D4 97F8 26 58 97FA 44 97FB 56 98F0 C3 97FF 44 97FD 56 9860 C3 00 01 9865 BD 92 98 9808 DG C4 980A 34 10 980C 0D D8 980E C7 21 9810 8D 11 9812 A6 C4 9814 A7 80	L97EB L98Ø8 L98ØC	STD LDA LDD BCC STD JSR LDD LDX LEAX TST BNE LSRA RORB RORB RORB RORB RORB RORB RORB RO	VEREND PMODE HOREND L97EB HOREND HOREND L9420 HOREND VEREND \$Ø1,X VEREND VD4 L9852 #1 HOREND L9298 HOREND+1 X VD8 L9831 L9823 ,U ,X++	SAVE VERT DIFFERENCE * GET PMODE BIT Ø * TO THE CARRY FLAG GET HORIZ DIFFERENCE BRANCH IF PMODE = Ø,2,4(2 COLOR) * DOUBLE HORIZ DIFF - 2X AS MANY BYTES FOR * NUMBER OF PIXELS IN PMODES 1,3 NORMALIZE DIFFERENCES GET HORIZ DIFFERENCE * * ADD 1 TO VERT DIFFERENCE - CHECK FOR G OPTION OR GET ACTION - AND BRANCH IF GIVEN * * DIVIDE HORIZONTAL DIFFERENCE BY 8 ADD ONE TO QUOTIENT SAVE NEW HOR DIFFERENCE *CONVERT (HORBEG, VERSEG) INTO ABSOLUTE SCREEN *POS (X) AND PIXEL MASK (ACCA) GET HORIZ DIFFERENCE SAVE SCREEN POSITION CHECK THE GET/PUT FLAG BRANCH IF GET INCREMENT ARRAY DATA POINTER GET DATA FROM ARRAY PUT IT ON THE SCREEN
3222 3223 3224 3225 3227 3228 3239 3231 3232 3233 3234 3235 3236 3237 3241 3242 3243 3241 3242 3243 3244 3245 3246 3257 3258	97DE DD C5 97E0 96 B6 97E0 96 B6 97E2 46 97E3 DC C3 97E5 24 04 97E7 D3 C3 97E9 DD C3 97EB BD 94 20 97EE DC C3 97F0 9E C5 97F6 0D D4 97F8 26 58 97FA 44 97FB 56 97FC 44 97FD 56 9800 C3 00 01 9803 DD C3 9805 BD 92 98 9808 D6 C4 9804 A7 80 9806 C7 21 9810 8D 11 9812 A6 C4 9814 A7 80 9816 5A	L97EB L98Ø8	STD LDA LDD BCC ADDD STD JSR LDD LDX LEAX TST ST EST BEQ LSRA ADDD STD JSR LSRA RORB LOB LSRA RORB LOB LSRA RORB LOB LSRA RORB LOB LOB LOB LOB LOB LOB LOB LOB LOB LO	VEREND PMODE HOREND L97EB HOREND HOREND L9420 HOREND VEREND \$Ø1,X VEREND VEREND VEREND VEREND VD4 L9852 #1 HOREND L9298 HOREND+1 X VD8 L9831 L9823 ,U ,X+	SAVE VERT DIFFERENCE * GET PMODE BIT Ø * TO THE CARRY FLAG GET HORIZ DIFFERENCE BRANCH IF PMODE = Ø,2,4(2 COLOR) * DOUBLE HORIZ DIFF - 2X AS MANY BYTES FOR * NUMBER OF PIXELS IN PMODES 1,3 NORMALIZE DIFFERENCE SET HORIZ DIFFERENCE * * ADD 1 TO VERT DIFFERENCE - CHECK FOR G OPTION OR GET ACTION - AND BRANCH IF GIVEN * * DIVIDE HORIZONTAL DIFFERENCE BY 8 ADD ONE TO QUOTIENT SAVE NEW HOR DIFFERENCE *CONVERT (HORBEG, VERSEG) INTO ABSOLUTE SCREEN *POS (X) AND PIXEL MASK (ACCA) GET HORIZ DIFFERENCE SAVE SCREEN POSITION CHECK THE GET/PUT FLAG BRANCH IF GET INCREMENT ARRAY DATA POINTER GET DATA FROM ARRAY PUT IT ON THE SCREEN DECREMENT HORIZ DIFFERENCE
3222 3223 3224 3225 3226 3227 3238 3231 3232 3233 3234 3235 3236 3237 3240 3241 3242 3243 3244 3245 3246 3247 3248 3249 3253 3251 3252 3253	97DE DD C5 97E0 96 B6 97E0 96 B6 97E2 46 97E3 DC C3 97E5 24 04 97E7 D3 C3 97E9 DD C3 97EB DD 94 20 97EE DC C3 97F0 9E C5 97F2 30 01 97F4 9F C5 97F6 0D D4 97F8 26 58 97FC 44 97FB 56 97FC 44 97FB 56 97FC 44 97FF 56 9800 C3 00 01 9803 DD C3 9805 BD 92 98 9808 DG C4 9804 34 10 9806 QD D8 9808 C7 21 9810 8D 11 9810 8D 11 9812 A6 C4 9814 A7 80 9816 5A	L97EB L98Ø8 L98ØC	STD LDA LDD BCC STD JSR LDD LDX STST BNE LSRA RORB LSRA RORB LSRA RORB LSRA ROBD STD JSR LDB LSRA ROBD STD JSR LDB STD JSR LDB STD JSR BEQ BSR BBE LDB STA BEG BBR BBE	VEREND PMODE HOREND L97EB HOREND HOREND L9420 HOREND VEREND \$01, X VEREND VD4 L9852 #1 HOREND L9298 HOREND L9298 HOREND+1 X VD8 L9831 L9823 ,U ,X+ L980C	SAVE VERT DIFFERENCE * GET PMODE BIT Ø * TO THE CARRY FLAG GET HORIZ DIFFERENCE BRANCH IF PMODE = Ø,2,4(2 COLOR) * DOUBLE HORIZ DIFF - 2X AS MANY BYTES FOR * NUMBER OF PIXELS IN PMODES 1,3 NORMALIZE DIFFERENCE SECT HORIZ DIFFERENCE * * ADD 1 TO VERT DIFFERENCE = CHECK FOR G OPTION OR GET ACTION = AND BRANCH IF GIVEN * * * * DIVIDE HORIZONTAL DIFFERENCE BY 8 ADD ONE TO QUOTIENT SAVE NEW HOR DIFFERENCE *CONVERT (HORBEG, VERSEG) INTO ABSOLUTE SCREEN *POS (X) AND PIXEL MASK (ACCA) GET HORIZ DIFFERENCE SAVE SCREEN POSITION CHECK THE GET/PUT FLAG BRANCH IF GET INCREMENT ARRAY DATA POINTER GET DATA FROM ARRAY PUT IT ON THE SCREEN DECREMENT HORIZ DIFFERENCE BRANCH IF NOT AT END OF HORIZ LINE
3222 3223 3224 3225 3226 3227 3238 3231 3232 3233 3234 3235 3236 3237 3238 3241 3242 3243 3244 3245 3246 3247 3248 3259 3250 3251 3252 3253 3252 3253 3255	97DE DD C5 97E0 96 86 97E2 46 97E3 DC C3 97E5 24 04 97E7 D3 C3 97E9 DD C3 97EB BD 94 20 97EB DC C3 97F0 9E C5 97F6 0D D4 97F8 26 58 97FA 44 97FB 56 97FC 44 97FD 56 9800 C3 00 01 9805 BD 92 98 9808 BD 92 98 9808 DC C4 980A 34 10 980C 0D D8 980E C7 21 9810 8D 11 9812 A6 C4 9814 A7 80 9816 5A 9817 26 F3	L97EB L98Ø8 L98ØC	STD LDA RORA LDD BCC ADDD STD JSR LDD LDX STX TST ST	VEREND PMODE HOREND L97EB HOREND HOREND L9420 HOREND VEREND \$601, X VEREND VD4 L9852 #1 HOREND L9298 HOREND+1 X VD8 L9823 ,U ,X+ L980C X	SAVE VERT DIFFERENCE * GET PMODE BIT Ø * TO THE CARRY FLAG GET HORIZ DIFFERENCE BRANCH IF PMODE = Ø,2,4(2 COLOR) * DOUBLE HORIZ DIFF - 2X AS MANY BYTES FOR * NUMBER OF PIXELS IN PMODES 1,3 NORMALIZE DIFFERENCES GET HORIZ DIFFERENCE * * ADD 1 TO VERT DIFFERENCE - CHECK FOR G OPTION OR GET ACTION - AND BRANCH IF GIVEN * * DIVIDE HORIZONTAL DIFFERENCE BY 8 ADD ONE TO QUOTIENT SAVE NEW HOR DIFFERENCE *CONVERT (HORBEG, VERSEG) INTO ABSOLUTE SCREEN *POS (X) AND PIXEL MASK (ACCA) GET HORIZ DIFFERENCE SAVE SCREEN POSITION CHECK THE GET TINCEMENT ARRAY DATA POINTER GET DATA FROM ARRAY PUT IT ON THE SCREEN DECREMENT HORIZ DIFFERENCE BRANCH IF GET INCREMENT ARRAY DATA POINTER GET DATA FROM ARRAY PUT IT ON THE SCREEN DECREMENT HORIZ DIFFERENCE BRANCH IF NOT AT END OF HORIZ LINE GET SCREEN POSITION BACK
3222 3223 3224 3225 3227 3228 3239 3231 3232 3233 3234 3235 3236 3237 3241 3242 3243 3241 3242 3243 3241 3245 3246 3257 3258 3251 3258 3259 3259 3259 3259 3259 3259 3259 3259	97DE DD C5 97E0 96 B6 97E0 96 B6 97E2 46 97E3 DC C3 97E5 24 04 97E7 D3 C3 97E9 DD C3 97EB BD 94 20 97EE DC C3 97F0 9E C5 97F6 0D D4 97F8 26 58 97FA 44 97FB 56 97FC 44 97FD 56 9800 C3 00 01 9803 DD C3 9808 BD 92 98 9808 D6 C4 9804 A7 80 9806 C7 21 9810 8D 11 9812 A6 C4 9814 A7 80 9815 A 9817 26 F3 9819 35 10 9818 BD 92 E9	L97EB L98Ø8 L98ØC	STD LDA LDD BCC ADDD STD JSR LDD LDX LEAX TST ST BNE LSRA RORB BME LSRA RORB BEQ BBE BBE LDA STD BEQ BBE BBE BBE BBE LSRA RORB BBE LSRA RORB BBE BBE LSRA RORB BBE LSRA RORB BBE BBE BBE BBE BBE BBE BBE BBE BBE B	VEREND PMODE HOREND L97EB HOREND HOREND L9420 HOREND VEREND \$Ø1,X VEREND VD4 L9852 #1 HOREND L9298 HOREND+1 X VD8 L9831 L9831 L9823 ,U ,X+ L980C X L92E9	SAVE VERT DIFFERENCE * GET PMODE BIT Ø * TO THE CARRY FLAG GET HORIZ DIFFERENCE BRANCH IF PMODE = Ø,2,4(2 COLOR) * DOUBLE HORIZ DIFF - 2X AS MANY BYTES FOR * NUMBER OF PIXELS IN PMODES 1,3 NORMALIZE DIFFERENCE SET HORIZ DIFFERENCE * * ADD 1 TO VERT DIFFERENCE = CHECK FOR G OPTION OR GET ACTION = AND BRANCH IF GIVEN * * DIVIDE HORIZONTAL DIFFERENCE BY 8 ADD ONE TO QUOTIENT SAVE NEW HOR DIFFERENCE *CONVERT (HORBEG, VERSEG) INTO ABSOLUTE SCREEN *>DOS (X) AND PIXEL MASK (ACCA) GET HORIZ DIFFERENCE SAVE SCREEN POSITION CHECK THE GET/PUT FLAG BRANCH IF GET INCREMENT ARRAY DATA POINTER GET DATA FROM ARRAY PUT IT ON THE SCREEN DECREMENT HORIZ DIFFERENCE BRANCH IF NOT AT END OF HORIZ LINE GET SCREEN POSITION BACK MOVE ABS POSITION DOWN ONE ROW
3222 3223 3224 3225 3226 3227 3238 3231 3232 3233 3234 3235 3236 3237 3248 3249 3241 3242 3243 3244 3245 3245 3246 3247 3248 3251 3252 3252 3252 3252 3252 3252 3252	970E DD C5 97E0 96 B6 97E2 46 97E3 DC C3 97E5 24 04 97E7 DG C3 97E9 DD C3 97E9 DD C3 97E8 DD 94 20 97EE DC C3 97F0 9E C5 97F2 30 01 97F4 9F C5 97F6 0D D4 97F8 26 58 97FC 44 97FB 56 97FC 44 97FB 56 97FC 44 97FF 56 9800 C3 00 01 9803 DD C3 9805 BD 92 98 9808 DG C4 9804 34 10 9805 0D D8 9806 27 21 9810 8D 11 9812 AG C4 9814 A7 80 9816 5A 9817 26 F3 9819 35 10 9818 BD 92 E9 9818 BD 92 E9 9818 BD 92 E9	L97EB L98Ø8 L98ØC	STD LDA LDD BCC STD JSR LDD LDX STX TST BNE LSRA RORB LSRA RORB LSRA RORB LSRA RORB LSRA ROB BCSB LSRA ROB BCSB BCSB BCSB BCSB BCC BCCB BCCB BCC	VEREND PMODE HOREND L97EB HOREND HOREND L9420 HOREND VEREND \$01, X VEREND VD4 L9852 #1 HOREND L9298 HOREND L9298 HOREND+1 X VD8 L9831 L9823 ,U ,X+ L980C X L9259 VEREND+1	SAVE VERT DIFFERENCE * GET PMODE BIT Ø * TO THE CARRY FLAG GET HORIZ DIFFERENCE BRANCH IF PMODE = Ø,2,4(2 COLOR) * DOUBLE HORIZ DIFF - 2X AS MANY BYTES FOR * NUMBER OF PIXELS IN PMODES 1,3 NORMALIZE DIFFERENCE SECT HORIZ DIFFERENCE * * ADD 1 TO VERT DIFFERENCE = CHECK FOR G OPTION OR GET ACTION = AND BRANCH IF GIVEN * * * * DIVIDE HORIZONTAL DIFFERENCE BY 8 ADD ONE TO QUOTIENT SAVE NEW HOR DIFFERENCE *CONVERT (HORBEG, VERSEG) INTO ABSOLUTE SCREEN *POS (X) AND PIXEL MASK (ACCA) GET HORIZ DIFFERENCE SAVE SCREEN POSITION CHECK THE GET/PUT FLAG BRANCH IF GET INCREMENT ARRAY DATA POINTER GET DATA FROM ARRAY PUT IT ON THE SCREEN DECREMENT HORIZ DIFFERENCE BRANCH IF NOT AT END OF HORIZ LINE GET SCREEN POSITION BACK MOYE ABS POSITION DAWN ONE ROW DECREMENT VERTICAL DIFFERENCE
3222 3223 3224 3225 3227 3228 3239 3231 3232 3233 3234 3235 3236 3237 3241 3242 3243 3244 3245 3246 3247 3248 3259 3259 3259 3251 3252 3253 3255 3256 3257 3258	970E DD C5 97E0 96 B6 97E2 46 97E3 DC C3 97E5 24 04 97E7 D3 C3 97E9 DD C3 97EB DD 94 20 97EE DC C3 97F6 9E C5 97F6 0D D4 97F8 26 58 97FA 44 97FB 56 97FC 44 97FB 56 97FE 44 97FB 56 9800 C3 00 01 9803 DD C3 9805 BD 92 98 9808 D6 C4 980A 34 10 980C 0D D8 980E 0D D8	L98Ø8 L98ØC L9816	STD LDA LDD BCC ADDD STD JSR LDD LDX STX TST ST ST ST ST ST LDB PSHS TST SE BCQ BSR LDB BCD BCB BNE BCB BCB BCB BCB BCB BCB BCB BCB BCB BC	VEREND PMODE HOREND L97EB HOREND HOREND L9420 HOREND VEREND \$Ø1,X VEREND VD4 L9852 #1 HOREND L9298 HOREND+1 X VD8 L9831 L9831 L9823 ,U ,X+ L980C X L92E9	SAVE VERT DIFFERENCE * GET PMODE BIT Ø * TO THE CARRY FLAG GET HORIZ DIFFERENCE BRANCH IF PMODE = Ø,2,4(2 COLOR) * DOUBLE HORIZ DIFF - 2X AS MANY BYTES FOR * NUMBER OF PIXELS IN PMODES 1,3 NORMALIZE DIFFERENCE SET HORIZ DIFFERENCE * * ADD 1 TO VERT DIFFERENCE = CHECK FOR G OPTION OR GET ACTION = AND BRANCH IF GIVEN * * DIVIDE HORIZONTAL DIFFERENCE BY 8 ADD ONE TO QUOTIENT SAVE NEW HOR DIFFERENCE *CONVERT (HORBEG, VERSEG) INTO ABSOLUTE SCREEN *>DOS (X) AND PIXEL MASK (ACCA) GET HORIZ DIFFERENCE SAVE SCREEN POSITION CHECK THE GET/PUT FLAG BRANCH IF GET INCREMENT ARRAY DATA POINTER GET DATA FROM ARRAY PUT IT ON THE SCREEN DECREMENT HORIZ DIFFERENCE BRANCH IF NOT AT END OF HORIZ LINE GET SCREEN POSITION BACK MOVE ABS POSITION DOWN ONE ROW
3222 3223 3224 3225 3227 3228 3239 3231 3232 3233 3234 3235 3236 3237 3238 3241 3242 3243 3241 3245 3246 3257 3258 3251 3251 3251 3252 3253 3254 3255 3256 3257 3256 3257 3256 3257 3256 3257 3256 3257 3256 3257 3256 3257	970E DD C5 97E0 96 B6 97E2 46 97E3 DC C3 97E5 24 04 97E7 D3 C3 97E9 DD C3 97EB DD 94 20 97EE DC C3 97F6 9E C5 97F6 0D D4 97F8 26 58 97FA 44 97FB 56 97FC 44 97FB 56 97FE 44 97FB 56 9800 C3 00 01 9803 DD C3 9805 BD 92 98 9808 D6 C4 980A 34 10 980C 0D D8 980E 0D D8	L97EB L98Ø8 L98ØC	STD LDA LDD BCC STD JSR LDD LDX STX TST BNE LSRA RORB LSRA RORB LSRA RORB LSRA RORB LSRA ROB BCSB LSRA ROB BCSB BCSB BCSB BCSB BCC BCCB BCCB BCC	VEREND PMODE HOREND L97EB HOREND HOREND L9420 HOREND VEREND \$01, X VEREND VD4 L9852 #1 HOREND L9298 HOREND L9298 HOREND+1 X VD8 L9831 L9823 ,U ,X+ L980C X L9259 VEREND+1	SAVE VERT DIFFERENCE * GET PMODE BIT Ø * TO THE CARRY FLAG GET HORIZ DIFFERENCE BRANCH IF PMODE = Ø,2,4(2 COLOR) * DOUBLE HORIZ DIFF - 2X AS MANY BYTES FOR * NUMBER OF PIXELS IN PMODES 1,3 NORMALIZE DIFFERENCE SECT HORIZ DIFFERENCE * * ADD 1 TO VERT DIFFERENCE = CHECK FOR G OPTION OR GET ACTION = AND BRANCH IF GIVEN * * * * DIVIDE HORIZONTAL DIFFERENCE BY 8 ADD ONE TO QUOTIENT SAVE NEW HOR DIFFERENCE *CONVERT (HORBEG, VERSEG) INTO ABSOLUTE SCREEN *POS (X) AND PIXEL MASK (ACCA) GET HORIZ DIFFERENCE SAVE SCREEN POSITION CHECK THE GET/PUT FLAG BRANCH IF GET INCREMENT ARRAY DATA POINTER GET DATA FROM ARRAY PUT IT ON THE SCREEN DECREMENT HORIZ DIFFERENCE BRANCH IF NOT AT END OF HORIZ LINE GET SCREEN POSITION BACK MOYE ABS POSITION DAWN ONE ROW DECREMENT VERTICAL DIFFERENCE
3222 3223 3224 3225 3226 3227 3238 3231 3232 3233 3234 3235 3236 3237 3240 3241 3242 3243 3244 3245 3245 3246 3247 3248 3252 3253 3252 3253 3252 3253 3253 325	97DE DD C5 97E0 96 B6 97E2 46 97E3 DC C3 97E5 24 04 97E7 D3 C3 97E9 DD C3 97EB BD 94 20 97EE DC C3 97FB 30 01 97FB 26 58 97FC 40 97FB 26 58 97FC 44 97FB 56 97FC 44 97FB 66 97	L9808 L980C L9816	STD LDA LDD BCC STD JSR LDD LDX STST BNE LSRA RORB LSRA RORB LSRA RORB LSRA RORB LSRA ROB ADD STD JSR LDB BEQ BSR BEQ BSR BRE PULS JSR ROB ROB ROB BRE RTS	VEREND PMODE HOREND L97EB HOREND HOREND L9420 HOREND VEREND \$01, X VEREND VD4 L9852 #1 HOREND L9298 HOREND L9298 HOREND+1 X VD8 L9831 L9823 ,U ,X+ L980C X L92E9 VEREND+1 L9808	SAVE VERT DIFFERENCE * GET PMODE BIT Ø * TO THE CARRY FLAG GET HORIZ DIFFERENCE BRANCH IF PMODE = Ø,2,4(2 COLOR) * DOUBLE HORIZ DIFF - 2X AS MANY BYTES FOR * NUMBER OF PIXELS IN PMODES 1,3 NORMALIZE DIFFERENCE SECT HORIZ DIFFERENCE * * ADD 1 TO VERT DIFFERENCE = CHECK FOR G OPTION OR GET ACTION = AND BRANCH IF GIVEN * * * * * * DIVIDE HORIZONTAL DIFFERENCE BY 8 ADD ONE TO QUOTIENT SAVE NEW HOR DIFFERENCE *CONVERT (HORBEG, VERSEG) INTO ABSOLUTE SCREEN *POS (X) AND PIXEL MASK (ACCA) GET HORIZ DIFFERENCE SAVE SCREEN POSITION CHECK THE GET/PUT FLAG BRANCH IF GET INCREMENT ARRAY DATA POINTER GET DATA FROM ARRAY PUT IT ON THE SCREEN DECREMENT HORIZ DIFFERENCE BRANCH IF NOT AT END OF HORIZ LINE GET SCREEN POSITION BACK MOVE ABS POSITION DOWN ONE ROW DECREMENT VERTICAL DIFFERENCE BRANCH IF NOT AT END OF HORIZ LINE GET SCREEN POSITION BACK MOVE ABS POSITION DOWN ONE ROW DECREMENT VERTICAL DIFFERENCE BRANCH IF NOT DONE BRANCH IF NOT DONE
3222 3223 3224 3225 3227 3228 3230 3231 3232 3233 3234 3235 3236 3237 3241 3242 3243 3244 3245 3244 3245 3250 3251 3252 3253 3254 3256 3257 3256 3257 3256 3257 3256 3257 3256 3257 3256 3257 3256 3257 3256 3257 3256 3257 3256 3257 3256 3257 3256 3257 3256 3257 3256 3257 3256 3257 3256 3257 3257 3257 3257 3257 3257 3257 3257	970E DD C5 97E0 96 B6 97E2 46 97E3 DC C3 97E5 24 04 97E7 D3 C3 97E9 DD C3 97EB BD 94 20 97EE DC C3 97F6 9E C5 97F6 0D D4 97F8 26 58 97FA 44 97FB 56 97FE 44 97FB 56 9800 C3 00 01 9803 DD C3 9808 BD 92 98 9808 D6 C4 9808 C4 DB 9808 C7 21 9810 8D 11 9812 A6 C4 9814 A7 80 9816 5A 9817 26 F3 9819 35 10 9818 BD 92 E9	L98Ø8 L98ØC L9816	STD LDA LDD BCC ADDD STD JSR LDD LDX STX TST ST ST LSRA RORB ADDD STD BEQ BSR LDB LSRA RORB ADDD STD BEQ BSR LDB BCQ BSR BSR BCQ B	VEREND PMODE HOREND L97EB HOREND HOREND L942Ø HOREND VEREND \$Ø1,X VEREND VD4 L9852 #1 HOREND L9298 HOREND+1 X VD8 L9831 L9823 ,U ,X+ L9269 VEREND+1 L9808 VCF	SAVE VERT DIFFERENCE * GET PMODE BIT Ø * TO THE CARRY FLAG GET HORIZ DIFFERENCE BRANCH IF PMODE = Ø,2,4(2 COLOR) * DOUBLE HORIZ DIFF - 2X AS MANY BYTES FOR * NUMBER OF PIXELS IN PMODES 1,3 NORMALIZE DIFFERENCE SECT HORIZ DIFFERENCE * * ADD 1 TO VERT DIFFERENCE = CHECK FOR G OPTION OR GET ACTION = AND BRANCH IF GIVEN * * * * DIVIDE HORIZONTAL DIFFERENCE BY 8 ADD ONE TO QUOTIENT SAVE NEW HOR DIFFERENCE *CONVERT (HORBEG, VERSEG) INTO ABSOLUTE SCREEN *POS (X) AND PIXEL MASK (ACCA) GET HORIZ DIFFERENCE SAVE SCREEN POSITION CHECK THE GET/PUT FLAG BRANCH IF GET INCREMENT ARRAY DATA POINTER GET DATA FROM ARRAY PUT IT ON THE SCREEN DECREMENT HORIZ DIFFERENCE BRANCH IF NOT AT END OF HORIZ LINE GET SCREEN POSITION BACK MOYE ABS POSITION DAWN ONE ROW DECREMENT VERTICAL DIFFERENCE
3222 3223 3224 3225 3227 3228 3239 3231 3232 3233 3234 3235 3236 3237 3238 3241 3242 3243 3244 3245 3246 3251 3251 3252 3253 3254 3256 3257 3256 3257 3256 3257 3256 3257 3256 3257 3256 3257 3256 3257 3256 3257 3256 3257 3256 3257 3256 3257 3256 3257 3256 3257 3256 3257 3256 3257 3256 3257 3257 3258 3257 3258 3257 3258 3257 3258 3257 3258 3259 3259 3259 3259 3259 3259 3259 3259	970E DD C5 97E0 96 B6 97E2 46 97E3 DC C3 97E5 24 04 97E7 D3 C3 97E9 DD C3 97EB BD 94 20 97EE DC C3 97F0 9E C5 97F2 30 01 97F4 9F C5 97F6 0D D4 97F8 26 58 97FA 44 97FB 56 97FE 44 97FD 56 97FE 44 97FD 56 9800 C3 00 01 9803 DD C3 9805 BD 92 98 9808 D6 C4 9806 0D D8 9808 D6 C4 9814 A7 80 9816 5A 9817 26 F3 9819 35 10 9818 BD 92 E9 9818 0A C6 9820 26 E6 9822 39 9823 DE CF 9825 33 41	L9808 L980C L9816	STD LDA LDD BCC ADDD STD JSR LDD LDX TST STST BNE LSRA RORB LSRA RORB LSRA ROB LSRA ROB LSRA ROB LSRA ROB BLSRA ROB BLSRA ROB BLSRA ROB BLSRA ROB BLSRA ROB BLSRA ROB BNE PULS BSR LDA STA DECB BNE PULS BNE PULS BNE LDA STA LDA STA LDB BRE RTS LDA LDA LDB BRE RTS LDA	VEREND PMODE HOREND L97EB HOREND HOREND L9420 HOREND VEREND \$01,X VEREND	SAVE VERT DIFFERENCE * GET PMODE BIT Ø * TO THE CARRY FLAG GET HORIZ DIFFERENCE BRANCH IF PMODE = Ø,2,4(2 COLOR) * DOUBLE HORIZ DIFF - 2X AS MANY BYTES FOR * NUMBER OF PIXELS IN PMODES 1,3 NORMALIZE DIFFERENCE SET HORIZ DIFFERENCE * * ADD 1 TO VERT DIFFERENCE = CHECK FOR G OPTION OR GET ACTION AND BRANCH IF GIVEN * * * DIVIDE HORIZONTAL DIFFERENCE BY 8 ADD ONE TO QUOTIENT SAVE NEW HOR DIFFERENCE *CONVERT (HORBEG, VERSEG) INTO ABSOLUTE SCREEN *POS (X) AND PIXEL MASK (ACCA) GET HORIZ DIFFERENCE SAVE SCREEN POSITION CHECK THE GET/PUT FLAG BRANCH IF GET INCREMENT ARRAY DATA POINTER GET DATA FROM ARRAY PUT IT ON THE SCREEN DECREMENT HORIZ DIFFERENCE BRANCH IF NOT AT END OF HORIZ LINE GET SCREEN POSITION BACK MOVE ABS POSITION DOWN ONE ROW DECREMENT VERTICAL DIFFERENCE BRANCH IF NOT DONE
3222 3223 3224 3225 3227 3228 3230 3231 3232 3233 3234 3235 3236 3237 3241 3242 3243 3244 3245 3244 3245 3250 3251 3252 3253 3254 3256 3257 3256 3257 3256 3257 3256 3257 3256 3257 3256 3257 3256 3257 3256 3257 3256 3257 3256 3257 3256 3257 3256 3257 3256 3257 3256 3257 3256 3257 3256 3257 3257 3257 3257 3257 3257 3257 3257	970E DD C5 97E0 96 B6 97E2 46 97E3 DC C3 97E5 24 04 97E7 D3 C3 97E9 DD C3 97EB BD 94 20 97EE DC C3 97F6 9E C5 97F6 0D D4 97F8 26 58 97FA 44 97FB 56 97FE 44 97FB 56 9800 C3 00 01 9803 DD C3 9808 BD 92 98 9808 D6 C4 9808 C4 DB 9808 C7 21 9810 8D 11 9812 A6 C4 9814 A7 80 9816 5A 9817 26 F3 9819 35 10 9818 BD 92 E9	L9808 L980C L9816	STD LDA LDD BCC ADDD STD JSR LDD LDX STX TST ST ST LSRA RORB ADDD STD BEQ BSR LDB LSRA RORB ADDD STD BEQ BSR LDB BCQ BSR BSR BCQ B	VEREND PMODE HOREND L97EB HOREND HOREND L9420 HOREND VEREND \$01, X VEREND VD4 L9852 #1 HOREND L9298 HOREND L9298 HOREND+1 X VD8 L9831 L9823 ,U ,X+ L980C X L92E9 VEREND+1 L9808	SAVE VERT DIFFERENCE * GET PMODE BIT Ø * TO THE CARRY FLAG GET HORIZ DIFFERENCE BRANCH IF PMODE = Ø,2,4(2 COLOR) * DOUBLE HORIZ DIFF - 2X AS MANY BYTES FOR * NUMBER OF PIXELS IN PMODES 1,3 NORMALIZE DIFFERENCE SET HORIZ DIFFERENCE * * ADD 1 TO VERT DIFFERENCE = CHECK FOR G OPTION OR GET ACTION = AND BRANCH IF GIVEN * * DIVIDE HORIZONTAL DIFFERENCE BY 8 ADD ONE TO QUOTIENT SAVE NEW HOR DIFFERENCE *CONVERT (HORBEG, VERSEG) INTO ABSOLUTE SCREEN *POS (X) AND PIXEL MASK (ACCA) GET HORIZ DIFFERENCE SAVE SCREEN POSITION CHECK THE GET/PUT FLAG BRANCH IF GET INCREMENT ARRAY DATA POINTER GET DATA FROM ARRAY PUT IT ON THE SCREEN DECREMENT HORIZ DIFFERENCE BRANCH IF NOT AT END OF HORIZ LINE GET SCREEN POSITION BACK MOVE ABS POSITION DOWN ONE ROW DECREMENT VERTICAL DIFFERENCE BRANCH IF NOT AT TEND OF HORIZ LINE GET SCREEN POSITION BACK MOVE ABS POSITION DOWN ONE ROW DECREMENT VERTICAL DIFFERENCE BRANCH IF NOT DONE

3265	982C 26 F4		BNE	L9822	RETURN IF NOT AT END
3266 3267	982E 7E B4 4A	L982E	JMP	LB44A	ILLEGAL FUNCTION CALL
3268	9831 A6 8Ø	L9831	LDA	, X+	GET DATA FROM SCREEN
3269 327Ø	9833 8D EE 9835 A7 C4		BSR STA	L9823 ,U	INCREMENT ARRAY DATA POINTER STORE IN ARRAY
3271	9837 20 DD	*		Ĺ9816	KEEP LOOPING TILL DONE
3272 3273	9839 98 94 98 9B	L9839	FDB	L9894,L989B	
3274 3275	983D BD 983E 98 9B 98 94	L983D L983E	FCB FDB	\$BD L989B,L9894	TOKEN FOR PSET
3276	9842 BE	L9842	FCB	\$BE	TOKEN FOR PRESET
3277 3278	9843 98 B1 98 9B 9847 B1	L9843 L9847	FDB FCB	L98B1,L989B \$B1	TOKEN FOR OR
3279	9848 98 94 98 B1	L9848	FDB	L9894,L98B1	
328Ø 3281	984C BØ 984D 98 A1 98 A1	L984C L984D	FCB FDB	198A1,L98A1	TOKEN FOR AND
3282 3283	9851 A8	L9851	FCB	\$A8	TOKEN FOR NOT
3284	2052 22 22 24			H 'G' OPTION SPECIFIED	400 AND TO HART DEFERENCE
3285 3286	9852 C3 ØØ Ø1 9855 DD C3	L9852	ADDD STD	#1 HOREND	ADD ONE TO HORIZ DIFFERENCE AND SAVE IT
3287 3288	9857 96 D8 9859 26 Ø9		LDA	VD8 L9864	*CHECK GET/PUT FLAG AND *BRANCH IF PUT
3289	985B DE D1		LDU	VD1	GET END OF ARRAYS
329Ø 3291	985D A7 C2 985F 11 93 CF	L985D	STA CMPU	,-U VCF	*THIS CODE WILL *ZERO OUT THE ENTIRE
3292 3293	9862 22 F9 9864 BD 92 98	10064		L985D	*'GET' ARRAY
3294	3004 BD 32 30	L9864 *		L9298	=CONVERT (HORBEG, VERBEG) INTO ABSOLUTE SCREEN POSITION =(X) AND PIXEL MASK (ACCA)
3295 3296	9867 D6 B6 9869 56		LDB RORB	PMODE	GET CURRENT PMODE BIT Ø TO CARRY
3297	986A 24 Ø2		BCC	L986E	BRANCH IF PMODE Ø,2,4 (2 COLOR)
3298 3299	986C 84 AA 986E C6 Ø1	L986E	LDB	#\$AA #\$Ø1	USE \$AA AS THE PIXEL MASK IN 4 COLOR MODE INITIALIZE SHIFT CTR
3300 3301	9870 10 9E CF 9873 34 12	L9873	LDY PSHS		POINT Y TO ARRAY DATA SAVE PIXEL MASK (ACCA) AND ABS SCRN POS (X) ON STACK
3302	9875 DE C3		LDU	HOREND	GET THE HORIZONTAL DIFFERENCE
33Ø3 33Ø4	9877 34 42 9879 54	L9877	PSHS LSRB	U,A	SAVE PIXEL MASK AND HORIZ DIFF SHIFT BIT CTR RIGHT
33Ø5 33Ø6	987A 24 Ø8 987C 56		BCC RORB	L9884	BRANCH IF ALL 8 SHIFTS NOT DONE SHIFT CARRY BACK INTO ACCB
33Ø7	987D 31 21		LEAY	\$01,Y	INCREMENT ARRAY DATA POINTER
33Ø8 33Ø9	987F 10 9C D1 9882 27 AA		CMPY BEQ	VD1 L982E	COMPARE TO END OF ARRAY FC ERROR IF AT END
331Ø 3311	9884 ØD D8 9886 27 1F	L9884	TST	VD8 L98A7	CHECK THE GET/PUT FLAG AND BRANCH IF GET
3312	9888 E5 A4		BITB	,Υ	TEST A BIT IN ARRAY DATA
3313 3314	988A 27 Ø4 988C 6E 9F ØØ D5		BEQ JMP	L9890 [VD5]	BRANCH IF ZERO JUMP TO ACTION ROUTINE FOR ARRAY BIT SET
3315 3316	9890 6E 9F 00 D9 9894 43	L989Ø L9894	JMP COMA	[VD9]	JUMP TO ACTION ROUTINE FOR ARRAY BIT CLEAR *MASK SOURCE DATA
3317	9895 A4 84	L3034	${\tt ANDA}$		*OFF OF SCREEN DATA
3318 3319	9897 A7 84 9899 20 16		STA BRA	, X L98B1	SAVE TO SCREEN
3320	989B AA 84	L989B	ORA	,Х	OR SOURCE DATA WITH SCREEN
3321 3322	989D A7 84 989F 20 10		STA BRA	,X L98B1	SAVE TO SCREEN
3323 3324	98A1 A8 84 98A3 A7 84	L98A1	EORA STA	, X , X	INVERT THE PIXEL SAVE TO SCREEN
3325	98A5 20 0A		BRA	L98B1	
3326 3327	98A7 A5 84 98A9 27 Ø6	L98A7	BITA BEQ	,X L98B1	TEST THE PIXEL BRANCH IF IT IS OFF
3328 3329	98AB 1F 98 98AD AA A4		TFR ORA	B,A ,Y	PUT SHIFT CTR IN ACCA TURN ON PROPER BIT IN
3330	98AF A7 A4		STA	,Υ	THE ARRAY DATA
	98B1 35 42 98B3 BD 92 ED	L98B1	PULS JSR	A,U L92ED	RESTORE PIXEL MASK AND HOR DIFF MOVE SCRN POS & PIXEL MASK ONE TO RIGHT (TWO COLOR MODE)
	98B6 33 5F 98B8 11 93 8A		LEAU	\$-Ø1,U ZERO	* * DECD HODI7 DIEEEDENCE AND
3335	98BB 26 BA		BNE	L9877	* DECR HORIZ DIFFERENCE AND * BRANCH IF NOT ZERO
3336 3337	98BD AE 61 98BF 96 B9			\$Ø1,S HORBYT	GET ABS SCRN POS FROM STACK GET NUMBER BYTES/GRAPHIC ROW
3338	98C1 3Ø 86		LEAX	A,X	MOVE SCRN POS DOWN ONE ROW
	98C3 35 Ø2 98C5 32 62		PULS LEAS	A \$02,S	PULL PIXEL MASK OFF THE STACK GET X OFF THE STACK
3341 3342	98C7 ØA C6 98C9 26 A8			VEREND+1 L9873	DECR VERT ROW CTR BRANCH IF NOT DONE
3343	98CB 39		RTS	25073	RETURN FROM GET/PUT COMMAND
3344 3345	98CC BD B3 57	L98CC	JSR	LB357	EVAL ALPHA EXPR, RETURN DESCRIPTOR PTR IN X
3346 3347	98CF E6 82 98D1 A6 82		LDB LDA		*STRIP OFF THE VARIABLE *NAME (2 ALPHA-NUMERIC CHARACTERS) AND
3348	98D3 1F Ø3		TFR	D,U	*STORE THEM IN U
	98D5 9E 1D 98D7 9C 1F	L98D7		ARYTAB ARYEND	GET START OF ARRAYS COMPARE TO END OF ARRAYS
3351	98D9 10 27 1B 6D		LBEQ	LB44A	FC ERROR IF UNDEFINED ARRAY
3352 3353	98DD 11 A3 84 98EØ 27 Ø6			L98E8	COMPARE TARGET NAME TO ARRAY NAME RETURN IF CORRECT ARRAY FOUND
3354 3355	98E2 EC Ø2 98E4 3Ø 8B			\$Ø2,X	* GET OFFSET TO NEXT ARRAY AND * ADD TO POINTER
3356	98E6 20 EF		BRA	L98D7	KEEP SEARCHING FOR MATCH
3357 3358	98E8 30 02 98EA 39	L98E8	LEAX RTS	\$02,X	MOVE POINTER TO OFFSET TO NEXT ARRAY WASTED BYTE
3359 336Ø	98EB 39	L98EB	RTS		
3300					

3361	0050 01 40	* PAINT	CMDA	#!@!	CHECK FOR W CICK
3362 3363	98EC 81 40 98EE 26 02	PAINT	CMPA BNE	# W L98F2	CHECK FOR @ SIGN SKIP IF NOT
3364	98FØ 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 3367	98F5 BD 93 1D	•	JSR	L931D	*SAVE HOR COORD IN HORSES, VER COORD IN VERSES NORMALIZE THE HOR, VER COORDINATES
3368	98F8 86 Ø1		LDA	#\$01	PSET VALUE
3369	98FA 97 C2			SETFLG	SET PSET/PRESET FLAG TO PSET
337Ø 3371	98FC BD 95 81 98FF DC B4			L9581 WCOLOR	GET PAINT COLOR CODE & SET THE ACTIVE COLOR AND ALL PIXEL BYTES GET THEM
3371	9901 34 06		PSHS		SAVE THEM ON STACK
3373	99Ø3 9D A5			GÉTCCH	GET CURRENT CHARACTER FROM INPUT LINE
3374	9905 27 03		BEQ	L99ØA	BRANCH IF NONE LEFT - DEFAULT BORDER COLOR TO FOREGROUND,
3375 3376	9907 BD 95 81		JSR	L9581	PAINT COLOR TO BACKGROUND EVALUATE THE BORDER COLOR
3377	99ØA 96 B5	L99ØA		ALLCOL	GET BORDER COLOR ALL PIXEL BYTE
3378	99ØC 97 D8		STA		TEMP SAVE IT
3379 338Ø	99ØE 35 Ø6 991Ø DD B4		PULS	A,B WCOLOR	GET PAINT ACTIVE COLORS BACK RESAVE IT
3381	9912 4F		CLRA	WOOLOK	* 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 3385	9915 BD 95 22	*	JSR	L9522	*USED AS A FLAG TO EXIT THE PAINT ROUTINE GET NORMALIZED MAX HOR/VER VALUES - RETURN RESULT IN VD3, VD5
3386	9918 BD 92 8F			L928F	POINT U TO THE ROUTINE WHICH WILL SELECT A PIXEL
3387		*			
3388 3389	991B DF D9	* 'PAINI	STU	FIRST HORIZONTAL LINE FROM THE S	START COURDINATES SAVE IT
3390	991D BD 99 DF			L99DF	PAINT FROM CURRENT HOR COORD TO ZERO
3391	9920 27 ØF		BEQ	L9931	BRANCH IF NO PAINTING DONE - HIT BORDER INSTANTLY
3392 3393	9922 BD 99 CB			L99CB #\$Ø1	PAINT TOWARD MAX HOR COORD
3393 3394	9925 86 Ø1 9927 97 D7			#\$01 VD7	* UP/DN FLAG UP=1; DOWN=\$FF
3395	9929 BD 99 BA		JSR	L99BA	SAVE POSITIVE GOING LINE INFO ON STACK
3396	992C ØØ D7			VD7	SET UP/ON FLAG = FF
3397 3398	992E BD 99 BA 9931 10 DF DC	L9931		L99BA TMPSTK	SAVE NEGATIVE GOING LINE INFO ON STACK TEMP STORE STACK POINTER
3399	9934 ØD DB	L9934		CHGFLG	SEE IF PAINTED COLOR IS DIFFERENT THAN ORIGINAL COLOR
3400	9936 26 Ø3			L993B	BRANCH IF DATA HAS BEEN MODIFIED
34Ø1 34Ø2	9938 10 DE DC 993B 35 56	L993B		TMPSTK A,B,X,U	GET STACK POINTER BACK GET DATA FOR THE NEXT LINE SEGMENT TO CHECK FROM THE STACK
3403	993D ØF DB	L9930		CHGFLG	CLEAR CHANGE FLAG
34Ø4	993F 10 DF DC		STS	TMPSTK	TEMP SAVE STACK
3405	9942 30 01			\$Ø1,X	ADD ONE TO START HOR COORD - 1
34Ø6 34Ø7	9944 9F BD 9946 DF D1			HORBEG VD1	PUT IT AT CURRENT HOR COORD ADDR LENGTH OF PARENT LINE
3408	9948 97 D7			VD7	SAVE UP/DN FLAG
34Ø9	994A 27 9F			L98EB	EXIT ROUTINE IF UP/DN FLAG = Ø
341Ø 3411	994C 2B Ø6	* CUECN		L9954 ELOW CURRENT DATA	BRANCH IF UP/DN FLAG = DOWN
3412	994E 5C	CIILCK	INCB	DATA	INCREMENT VER COORD
3413	994F D1 D6		CMPB		COMPARE TO MAXIMUM VER COORD
3414 3415	9951 23 Ø5 9953 5F		BLS CLRB	L9958	BRANCH IF NOT GREATER - PROCESS LINE 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 3420	9957 5A	•	DECB		*VER COORD = Ø OR ABOVE MAX VERTICAL COORD DEC VER COORD - WE ARE TESTING UP/DN FLAG = UP IF THERE
3421			SS A HO	R LINE THAT WAS STORED ON STACK	- LIMIT CHECKS HAVE BEEN DONE
3422	9958 D7 CØ	L9958		VERBEG+1	SAVE CURRENT VER COORD
3423 3424	995A BD 99 DF 995D 27 ØF			L99DF L996E	PAINT FROM HOR COORD TO ZERO OR BORDER IF NUMBER OF PAINTED PIXELS = Ø, COMPLEMENT LENGTH
3425	995F 10 83 00 03		CMPD		*SEE IF < 3 PIXELS WERE PAINTED - IF FEWER THAN
3426		*			*THREE PIXELS WERE PAINTED THEN THERE IS NO NEED TO
3427 3428		*			*CHECK FOR MORE DATA TO PAINT ON THE LINE TO THE *LEFT OF THE CURRENT POSITION IN THE OPPOSITE
3429		*			*DIRECTION THAT THE UP/DN FLAG IS CURRENTLY SET TO
3430	9963 25 Ø4			L9969	BRANCH IF NO NEED TO CHECK FOR PAINTABLE DATA
3431 3432	9965 30 1E 9967 8D 38			\$-02,X L99A1	MOVE THE HORIZONTAL COORDINATE TWO PIXELS TO THE LEFT *SAVE A BLOCK OF PAINT DATA IN THE DIRECTION
3433	3307 OD 30	*	אכם	2371	*OPPOSITE TO THE UP/DN FLAG
3434 >	9969 BD 99 CB	L9969		L99CB	CONTINUE PAINTING LINE TO THE RIGHT
3435 3436	996C 8D 4C	L996C	BSR	L99BA	*SAVE A BLOCK OF PAINT DATA IN THE SAME *DIRECTION AS THE UP/DN FLAG
3437		*			*DIRECTION AS THE OP/DN FLAG
3438		* THIS C	CODE WI	LL INSURE THAT THE CURRENT LINE	IS
3439				THE RIGHT FOR 'PAINTABLE' PIXELS	
344Ø 3441	996E 43	* A LENG	COMA	AL TO THE LENGTH OF THE 'PARENT'	LINE
3442	996F 53	LJJUL	COMB		* COMPLEMENT LENGTH OF LINE JUST PAINTED
3443	997Ø D3 D1	L997Ø	ADDD		ADD TO LENGTH OF PARENT LINE
3444	9972 DD D1		STD	VD1 L998C	SAVE DIFFERENCE OF LINE JUST PAINTED AND PARENT LINE
3445 3446	9974 2F 16 9976 BD 95 Ø6			L9506	BRANCH IF PARENT LINE IS SHORTER GO INC HOR COORD
3447	9979 BD 9A 12		JSR	L9A12	CHECK FOR BORDER COLOR
3448	9970 26 Ø5			L9983	NOT BORDER COLOR -
	997E CC FF FF		LDD BRA	#-1 L997Ø	* GO DECREMENT ONE FROM LENGTH OF DIFFERENCE * LINE AND KEEP LOOKING FOR NON BORDER COLOR
3449 345Ø	9981 20 ED				
345Ø 3451	9981 20 ED 9983 BD 95 14	L9983	JSR	L9514	GO DEC HOR COORD
3450 3451 3452	9983 BD 95 14 9986 8D 3E	L9983	JSR BSR	L99C6	GET AND SAVE HOR COORD
345Ø 3451	9983 BD 95 14	L9983	JSR BSR BSR		
3450 3451 3452 3453	9983 BD 95 14 9986 8D 3E 9988 8D 5E	*	JSR BSR BSR BRA	L99C6 L99E8	GET AND SAVE HOR COORD PAINT FORWARD TO MAX HOR COORD OR BORDER SAVE A BLOCK OF PAINT DATA AND KEEP CHECKING

```
* THE RIGHT THAN THE PARENT LINE AND PUT A BLOCK OF * PAINT DATA ON THE STACK IF IT IS MORE THAN 2 PIXELS
3457
3458
                                          * PAST THE END OF THE PARENT LINE
L998C JSR L9506
LEAX D,X
STX HORBEG
3459
346Ø
          998C BD 95 Ø6
                                                                                                            INC CURRENT HOR COORD
         998F 3Ø 8B
9991 9F BD
                                                                                                            * POINT X TO THE RIGHT END OF THE PARENT LINE * AND SAVE IT AS THE CURRENT HORIZ COORDINATE
3461
3462
3463
          9993 43
                                                       COMA
                                                                                                            = ACCD CONTAINS A NEGATIVE NUMBER CORRESPONDING TO
= THE NUMBER OF PIXELS THE CURRENT LINE EXTENDS
3464
          9994 53
                                                       COMB
          9995 83 ØØ Ø1
                                                       SUBD #1
                                                                                                            = PAST THE RIGHT END OF THE PARENT LINE. CONVERT
= TO A POSITIVE NUMBER AND BRANCH IF THE LINE DOESN T EXTEND
3465
3466
          9998 2F Ø4
                                                       RIF
                                                              1999F
                                                                                                             *SAVE THE PORTION OF THE LINE TO THE RIGHT OF THE PARENT LINE
3467
          999A 1F Ø1
                                                       TFR
                                                              D,X
3468
                                                                                                             *AS THE LENGTH
                                                                                                             =SAVE A BLOCK OF PAINT DATA IN THE DIRECTION OPPOSITE THE
3469
         999C 8D Ø3
                                                       BSR
                                                              L99A1
3470
                                                                                                            =CURRENT UP/DN FLAG
PROCESS MORE PAINT DATA BLOCKS
3471 > 999E 7E 99 34
                                          L999E
                                                      JMP
                                                             L9934
3472
3473
                                          *
* BLOCKS OF PAINT DATA ARE STORED ON THE STACK SO THAT PAINT
* CAN REMEMBER WHERE IT SHOULD GO BACK TO PAINT UP OR DOWN
* FROM THE CURRENT LINE IT IS PAINTING. THESE BLOCKS OF DATA
* REPRESENT HORIZONTAL LINES ABOVE OR BELOW THE CURRENT LINE
* BEING PAINTED AND REQUIRE SIX BYTES OF STORAGE ON THE STACK.
* THE DATA ARE STORED AS FOLLOWS: ,S=UP/DOWN FLAG; 1,S=VER COORD
* OF LINE; 2 3,S=LEFTMOST HOR COORD OF LINE; 4 5,S=LENGTH OF LINE
3474
3475
3476
3477
3478
3479
3480
                                          * SAVE A BLOCK OF PAINT DATA FOR A LINE IN THE * OPPOSITE DIRECTION OF THE CURRENT UP/DN FLAG
3481
3482
3483
          99A1 DD CB
                                                       STD
                                                       STD VCB
PULS Y
                                                                                                            SAVE NUMBER PIXELS PAINTED PUT RETURN ADDR IN Y
3484
          99A3 35 2Ø
3485
          99A5 DC BD
                                                       LDD
                                                              HORBEG
                                                                                                             GET HORIZONTAL START COORDINATE
                                                       PSHS X,B,A
          99A7 34 16
3486
                                                                                                            PUT ON STACK
3487
          99A9 96 D7
                                                               VĎ7
                                                                                                             GET U/D FLAG
                                                       LDA
3488
          99AB 4Ø
                                                       NEGA
                                                                                                            REVERSE THE UP/DN FLAG
                                                              VERBEG+1
                                                                                                             GET VERTICAL START COORDINATE
          99AC D6 CØ
                                           L99AC
                                                       LDB
                                                       PSHS B,A
3490
          99AF 34 Ø6
                                                                                                            SAVE VERTICAL START COORDINATE AND U/D FLAG
3491
          99BØ 34 2Ø
                                                                                                             PUT RETURN ADDR BACK ON STACK
                                                       PSHS
3492
3493
                                             CODE BELOW CHECKS FOR ABILITY TO STORE FOUR BYTES IN
                                          * FREE RAM, HOWEVER THE PAINT ROUTINE WILL STORE SIX
* BYTES IN FREE RAM - FIRST INSTRUCTION SHOULD BE LDB #3
3494
3495
                                                                                                            * CHECK TO SEE IF THERE S ENOUGH FREE
* RAM FOR 4 BYTES TEMP STORAGE
3496
          99B2 C6 Ø2
                                                       LDB
3497
          99B4 BD AC 33
                                                       JSR
                                                              LAC33
3498
          99B7 DC CB
                                                                                                            GET LENGTH OF RIGHT PAINTED LINE
                                                               VCB
3499
          9989 39
                                                       RTS
3500
                                          * SAVE A BLOCK OF PAINT DATA FOR A LINE IN * THE SAME DIRECTION AS THE CURRENT UP/DN FLAG
3501
3502
          99BA DD CB
99BC 35 20
                                                                                                            SAVE LENGTH OF RIGHT HOR PAINTED LINE SAVE RETURN ADDRESS IN Y
3503
                                          L99BA
                                                       STD
                                                              VCB
                                                       PULS Y
35Ø4
                                                                                                            HORIZONTAL START COORDINATE
SAVE HORIZONTAL START COORDINATE AND LENGTH
35Ø5
          99BE DC C3
                                                       LDD
                                                              HOREND
          99CØ 34 16
                                                       PSHS
3506
                                                              X,B,A
                                                                                                            SAVE THE PAINT DATA ON THE STACK
GET CURRENT HOR COORD
3507
          99C2 96 D7
                                                               VD7
3508
          99C4 20 E6
                                                       BRA
                                                              L99AC
          99C6 9E BD
35Ø9
                                           L99C6
                                                       LDX
                                                               HORBEG
                                                                                                             SAVE IT
3510
          99C8 9F C3
                                                       STX
                                                              HOREND
          99CA 39
3512
                                           * GO HERE TO FINISH PAINTING TO RIGHT AFTER YOU HAVE PAINTED TO THE LEFT
          99CB DD CD
                                                      STD
                                                                                                            SAVE COUNT OF THE NUMBER OF PIXELS PAINTED
3513
                                          L99CB
                                                              VCD
                                                                                                            GET LAST HOR START COORD
*SAVE CURRENT HOR COORD - NOW HOREND CONTAINS COORDINATE
3514
          99CD 10 9E C3
                                                       IDY
                                                              HOREND
          99DØ 8D F4
3515
                                                              L99C6
                                                       BSR
                                                                                                            START PAINTING TO RIGHT FROM THE LEFT PAINT START COORD
3516
          99D2 10 9F BD
                                                       STY
                                                              HORBEG
3517
          99D5 8D 11
                                                              L99E8
                                                                                                             PAINT TOWARDS THE RIGHT
3518
                                                                                                             GET THE NUMBER OF PIXELS PAINTED WHEN GOING TOWARDS LEFT PIXELS
3519
          99D7 9E CD
                                                       LDX
                                                              VCD
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
                                          * PAINT FROM HOR COORD TO ZERO OR HIT BORDER
3524
                                          * RETURN WITH Z = 1 IF NO PAINTING DONE L99DF JSR L99C6
3525
3526 > 99DF BD 99 C6
                                                                                                             PUT STARTING HOR COORD IN HOREND
                                                              #L9514
L99EE
                                                                                                            (DECR HOR COORD ADDRESS) TO Y
3527
          99E2 10 8E 95 14
                                                       LDY
3528
          99E6 2Ø Ø6
                                                       BRA
                                           * PAINT FROM HOR COORD TO MAX HOR COORD OR HIT
* BORDER-RETURN Z=1 IF NO PAINTING DONE
3530
                                                                                                            PUT INCR HOR COORD ADDR IN Y
INCR HOR COORD - THE LEFT PAINT ROUTINE PAINTED THE FIRST COORD
ZERO U REG - INITIAL PIXEL PAINT COUNTER
GET HOR COORD
BRANCH IF HORIZONTAL COORDINATE IS > 87F OR < Ø
3531
          99E8 10 8E 95 06
                                           L99E8
                                                       LDY
                                                              #L95Ø6
3532
          99EC AD A4
                                                       JSR
          99EE DE 8A
                                                               ZERO
3533
                                          L99EE
                                                       LDU
          99FØ 9E BD
99F2 2B 17
3534
                                                       LDX
                                                              HORBEG
3535
                                                       BMI
                                                               L9AØB
                                                                                                            COMPARE CURRENT HOR COORD TO MAX VALUE
BRANCH IF > MAX
SAVE PAINT COUNTER, INC/DEC POINTER
CHECK FOR BORDER PIXEL
3536
         99F4 9C D3
99F6 22 13
                                                       CMPX
                                                              VD3
3537
                                                              L9AØB
                                                       BHI
                                                      PSHS U,Y
BSR L9A12
3538
          99F8 34 6Ø
          99FA 8D 16
3539
3540
          99FC 27 ØB
                                                               L9AØ9
                                                                                                            HIT BORDER
SET PIXEL TO PAINT COLOR - PAINTING IS DONE HERE
          99FE BD 93 77
3541
                                                       JSR
                                                              L9377
3542
          9AØ1 35 6Ø
                                                                                                             RESTORE PAINT COUNTER AND INC/DEC POINTER
                                                               Y,U
3543
          9403 33 41
                                                       LEAU $Ø1.U
                                                                                                             ADD ONE TO PAINT COUNTER
          9AØ5 AD A4
                                                                                                             INCR OR DECR HOR COORD DEPENDING ON CONTENTS OF Y
3544
                                                       JSR
                                                                                                            KEEP PAINTING THE LINE
RESTORE PAINT COUNTER AND INC/DEC POINTER
3545
          9AØ7 2Ø E9
                                                       RRA
                                                              199F2
3546
          9AØ9 35 6Ø
                                           L9AØ9
                                                       PULS
                                                              Y,U
          9AØB 1F 3Ø
9AØD 1F Ø1
3547
                                          L9AØB
                                                       TFR
TFR
                                                              U,D
                                                                                                            SAVE PAINT COUNTER IN ACCD SAVE PAINT COUNTER IN X
3548
3549
          9AØF 93 8A
                                                       SUBD ZÉRO
                                                                                                             SET FLAGS ACCORDING TO CONDITION OF PAINT COUNTER
3550
          9A11 39
                                                       RTS
                                          * CHECK FOR BORDER COLOR - ENTER W/VD9 CONTAINING * ADDRESS OF ROUTINE TO GET ABS SCREEN ADDRESS
3552
```

3553 3554	9A12 AD 9F ØØ D9	* AND PI L9A12	XEL MASK - E JSR [VD9]	XIT WITH Z = 1 IF HIT BORDER	R COLOR PIXEL GET THE ADDR AND PIXEL MASK
3555	9A16 1F 89	LJAIZ	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 3559	9A1C A4 84 9A1E A1 61		ANDA ,X CMPA \$Ø1,S		* PUT CURRENT PIXEL DATA INTO ACCB AND * COMPARE IT TO BORDER COLOR; Z FLAG = 1 IF MATCH
3560	9A2Ø 35 86		PULS A,B,P		RESTORE MASK AND BORDER PIXEL - THEN RETURN
3561					
3562	0400 05 04	* PLAY	1.DV 7.EDO	_	ADDECAULT WALLES FOR LENGTH OF DLAY COMMAND AND ADDRESS
3563 3564	9A22 9E 8A 9A24 C6 Ø1	PLAY	LDX ZERO LDB #\$Ø1		DEFAULT VALUES FOR LENGTH OF PLAY COMMAND AND ADDRESS FOF 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	E	EVALUATE EXPRESSION
3567 3568	9A2B 5F 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	0.105 00 00	*			FOF STRING INTO ACCB
3572 3573	9A35 2Ø Ø2 9A37 35 14	L9A37	BRA L9A39 PULS B,X		INEFFICIENT – SHOULD BE FCB SKP2 GET PLAY STRING START AND LENGTH
3574	9A39 D7 D8	L9A39	STB VD8		ENGTH 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 3578	9A3F 10 27 0F 31 9A43 0D D8	L9A43	LBEQ LA974 TST VD8		DISABLE ANA MUX AND RETURN IF X = Ø SEE IF LENGTH OF STRING = Ø
3579	9A45 27 FØ		BEQ L9A37		SET NEW DATA IF SO
3580	9A47 BD 9B 98		JSR L9B98		GET A COMMAND CHARACTER IF NOT
3581 3582	9A4A 81 3B 9A4C 27 F5		CMPA #';' BEQ L9A43		SUB COMMAND TERMINATED 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 3587	9A54 10 27 01 B2 9A58 8D 02		LBEQ L9CØA BSR L9A5C		GO PROCESS SUB COMMAND CHECK FOR OTHER COMMANDS
3588	9A5A 2Ø E7		BRA L9A43		KEEP GOING THROUGH INTERPRETATION LOOP
3589		* OCTAVE			
3590	9A5C 81 4F	L9A5C	CMPA #'0'		ADJUST OCTAVE?
3591 3592	9A5E 26 ØD 9A6Ø D6 DE		BNE L9A6D LDB OCTAVI		NO 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 3597	9A66 C1 Ø4 9A68 22 63		CMPB #\$Ø4 BHI L9ACD		MAXIMUM VALUE OF 4 FC ERROR
3598	9A6A D7 DE		STB OCTAVI		SAVE NEW VALUE OF OCTAVE
3599	9A6C 39		RTS		
3600 3601	9A6D 81 56	* VOLUME L9A6D	CMPA #'V'	Α.	ADJUST VOLUME?
3602	9A6F 26 1A	LYAOD	BNE L9A8B		NO
3603	9A71 D6 DF		LDB VOLHI	G	SET CURRENT HIGH VOLUME LIMIT
3604	9A73 54		LSRB		SHIFT 2 BITS TO RIGHT; DA IS ONLY 6 BITS (BIT 2 - BIT 7) -
36Ø5 36Ø6	9A74 54 9A75 CØ 1F		LSRB SUBB #31		FTO MANIPULATE THE DATA IT MUST BE IN BITS Ø-5 SUBTRACT OUT MID VALUE OFFSET
3607	9A77 8D 47		BSR L9ACØ		MODIFIER CHECK
36Ø8	9A79 C1 1F		CMPB #31	M	MAXIMUM ALLOWED RANGE IS 31
3609	9A7B 22 5Ø		BHI L9ACD	_	FC ERROR
361Ø 3611	9A7D 58 9A7E 58		ASLB 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 #\$7E7I		PUT MID VALUE IN HIGH AND LOW LIMIT
3614 3615	9A84 AB E4 9A86 EØ EØ		ADDA ,S SUBB ,S+		ADD NEW VOLUME TO HIGH LIMIT SUBTR NEW VOLUME FROM LOW LIMIT
3616	9A88 DD DF		STD VOLHI		SAVE NEW VOLUME LIMITS
3617	9A8A 39		RTS		
3618	9A8B 81 4C	* NOTE L			PET NOTE LENGTHS
3619 3620	9A8D 26 23	L9A8B	CMPA #'L' BNE L9AB2		SET NOTE LENGTH? NO
3621	9A8F D6 E1		LDB NOTELI	I G	GET CURRENT LENGTH
3622	9A91 8D 2D		BSR L9ACØ	M *	MODIFIER CHECK
3623 3624	9A93 5D 9A94 27 37		TSTB BEQ L9ACD		FC ERROR IF LENGTH = Ø
3625	9A96 D7 E1		STB NOTELI		SAVE NEW NOTE LENGTH
3626	9A98 ØF E5		CLR DOTVA		RESET NOTE TIMER SCALE FACTOR
3627	9A9A 8D Ø3	L9A9A	BSR L9A9F		CHECK FOR A DOTTED NOTE
3628 3629	9A9C 24 FC 9A9E 39		BCC L9A9A RTS	В	BRANCH IF DOTTED NOTE
3630	31.32 03	* SCALE	FACTOR - DO	TTED NOTE	
3631	9A9F ØD D8	L9A9F	TST VD8		CHECK COMMAND LENGTH
3632	9AA1 27 ØA 9AA3 BD 9B 98		JSR L9B98		IT S EMPTY
3633 3634	9AA5 BD 9B 96 9AA6 81 2E		CMPA #'.'		GET COMMAND CHARACTER CHECK FOR DOTTED NOTE
3635	9AA8 27 Ø5		BEQ L9AAF	В	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 3638	9AAD 43	L9AAD	COMA		COMMAND LENGTH SET CARRY FLAG
3639	9AAE 39	LJAND	RTS	3	DEL CANAL LEAU
3640	9AAF ØC E5	L9AAF	INC DOTVA	. A	ADD ONE TO NOTE TIMER SCALE FACTOR
3641	9AB1 39	# TEMP	RTS		
3642 3643	9AB2 81 54	* TEMPO L9AB2	CMPA #'T'	М	NODIFY TEMPO?
3644	9AB4 26 ØD	LJNUL	BNE L9AC3		100 FT TEMPO:
3645	9AB6 D6 E2		LDB TEMPO		GET CURRENT TEMPO
3646 3647	9AB8 8D Ø6 9ABA 5D		BSR L9ACØ TSTB		EVALUATE MODIFIER SET FLAGS
3648	9ABB 27 10		BEQ L9ACD		FC ERROR IF IT S Ø

3649	9ABD D7 E2	STB TEMPO	SAVE NEW TEMPO
3650	9ABF 39	RTS	
3651	9ACØ 7E 9B AC	L9ACØ 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 3658	9ACB 26 Ø3	BNE L9ADØ L9ACD JMP LB44A	* BRANCH IF PAUSE VALUE <> Ø FC ERROR IF PAUSE <> Ø
3659	9ACD 7E B4 4A 9ADØ 96 E5	L9ACD JMP LB44A L9ADØ 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 = Ø
3664	9ADA 97 EØ	STA VOLLOW	*
3665	9ADC ØF E5	CLR DOTVAL	RESET NOTE TIMER SCALE FACTOR
3666	9ADE 8D Ø7	BSR L9AE7	GO PLAY A NOTE OF Ø VOLUME
3667 3668	9AEØ 35 12 9AE2 97 E5	PULS A,X STA DOTVAL	*RESTORE VALUE OF VOLUME
3669	9AE4 9F DF	STX VOLHI	*AND NOTE TIMER SCALE
3670	9AE6 39	RTS	AND HOTE THEN GOVEE
3671	9AE7 6F E2	L9AE7 CLR ,-S	PUSH NOTE NUMBER Ø ONTO STACK
3672	9AE9 2Ø 4Ø	BRA Ĺ9B2B	GO PLAY IT
3673		* NOTE	
3674	9AEB 81 4E	L9AEB CMPA #'N'	LETTER N BEFORE THE NUMBER OF A NOTE?
3675	9AED 26 Ø3	BNE L9AF2	NO - IT S OPTIONAL
3676	9AEF BD 9B 98	JSR L9B98	GET NEXT COMMAND CHARACTER
3677 3678	9AF2 81 41 9AF4 25 Ø4	L9AF2 CMPA #'A' BLO L9AFA	CHECK FOR NOTE A BELOW
3679	9AF6 81 47	CMPA #'G'	CHECK FOR NOTE B
3680	9AF8 23 Ø5	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 8Ø 41	L9AFF SUBA #'A'	MASK OFF ASCII
3685	9BØ1 8E 9C 5B	LDX #L9C5B	LOAD X WITH NOTE JUMP TABLE GET NOTE
3686 3687	9BØ4 E6 86 9BØ6 ØD D8	LDB A,X TST VD8	ANY COMMAND CHARACTERS LEFT?
3688	9BØ8 27 18	BEQ L9B22	NO
3689	9BØA BD 9B 98	JSR L9B98	GET COMMAND CHARACTER
3690	9BØD 81 23	CMPA #'#'	SHARP NOTE?
3691	9BØF 27 Ø4	BEQ L9B15	YES
3692	9B11 81 2B	CMPA #'+'	SHARP NOTE?
3693	9B13 26 Ø3	BNE L9B18	NO
3694	9B15 5C	L9B15 INCB	ADD 1 TO NOTE NUMBER (SHARP)
3695	9B16 2Ø ØA	BRA L9B22	PROCESS NOTE
3696 3697	9B18 81 2D 9B1A 26 Ø3	L9B18 CMPA #'-' BNE L9B1F	FLAT NOTE? NO
3698	9B1C 5A	DECB	SUBTR 1 FROM NOTE NUMBER (FLAT)
3699	9B1D 2Ø Ø3	BRA L9B22	PROCESS NOTE
3700	9B1F BD 9B E2	L9B1F JSR L9BE2	*MOVE COMMAND STRING PTR BACK ONE AND ADD ONE
37Ø1		*	*TO COMMAND LENGTH CTR
37Ø2	9B22 5A	L9B22 DECB	=ADJUST NOTE NUMBER, BASIC USES NOTE NUMBERS 1-12, INTERNALLY
37Ø3		*	=COMPUTER USES Ø-11
3704	9B23 C1 ØB	CMPB #12-1	MAXIMUM NOTE VALUE
37Ø5 37Ø6	9B25 22 A6 9B27 34 Ø4	BHI L9ACD PSHS B	FC ERROR IF > 11 SAVE NOTE VALUE
3707	9B29 D6 E1	LDB NOTELN	GET NOTE LENGTH
37Ø8	9B2B 96 E2	L9B2B LDA TEMPO	GET TEMPO
37Ø9	9B2D 3D	MUL	CALCULATE NOTE DURATION
3710	9B2E DD D5	STD VD5	SAVE NOTE DURATION
3711			
3712		* THE IRQ INTERRUPT IS USED TO PROVIDE A MA	STER TIMING REFERENCE FOR
0740		* THE PLAY COMMAND. WHEN A NOTE IS DONE,	STER TIMING REFERENCE FOR THE IRQ SERVICING
3713	OB30 33 61	* THE PLAY COMMAND. WHEN A NOTE IS DONE, * ROUTINE WILL RETURN CONTROL TO THE MAIN	STER TIMING REFERENCE FOR THE IRQ SERVICING PLAY COMMAND INTERPRETATION LOOP
3714	9B30 33 61	* THE PLAY COMMAND. WHEN A NOTE IS DONE,	STER TIMING REFERENCE FOR THE IRQ SERVICING PLAY COMMAND INTERPRETATION LOOP *LOAD U W/CURRENT VALUE OF (STACK POINTER+1) SO THAT THE STACK
	9B3Ø 33 61	* THE PLAY COMMAND. WHEN A NOTE IS DONE, * ROUTINE WILL RETURN CONTROL TO THE MAIN	STER TIMING REFERENCE FOR THE IRQ SERVICING PLAY COMMAND INTERPRETATION LOOP
3714 3715	9B32 96 DE	* THE PLAY COMMAND. WHEN A NOTE IS DONE, * ROUTINE WILL RETURN CONTROL TO THE MAIN LEAU \$01,S * LDA OCTAVE	STER TIMING REFERENCE FOR THE IRQ SERVICING PLAY COMMAND INTERPRETATION LOOP *LOAD U W/CURRENT VALUE OF (STACK POINTER+1) SO THAT THE STACK *POINTER WILL BE PROPERLY RESET WHEN IRQ VECTORS
3714 3715 3716 3717 3718	9B32 96 DE 9B34 81 Ø1	* THE PLAY COMMAND. WHEN A NOTE IS DONE, * ROUTINE WILL RETURN CONTROL TO THE MAIN LEAU \$01,S * LDA OCTAVE CMPA #\$01	STER TIMING REFERENCE FOR THE IRQ SERVICING PLAY COMMAND INTERPRETATION LOOP *LOAD U W/CURRENT VALUE OF (STACK POINTER+1) SO THAT THE STACK *POINTER WILL BE PROPERLY RESET WHEN IRQ VECTORS *YOU OUT OF THE PLAY TIMING ROUTINES BELOW GET CURRENT OCTAVE *
3714 3715 3716 3717 3718 3719	9B32 96 DE	* THE PLAY COMMAND. WHEN A NOTE IS DONE, * ROUTINE WILL RETURN CONTROL TO THE MAIN LEAU \$01,S * LDA OCTAVE CMPA #\$01 BHI L9864	STER TIMING REFERENCE FOR THE IRQ SERVICING PLAY COMMAND INTERPRETATION LOOP *LOAD U W/CURRENT VALUE OF (STACK POINTER+1) SO THAT THE STACK *POINTER WILL BE PROPERLY RESET WHEN IRQ VECTORS *YOU OUT OF THE PLAY TIMING ROUTINES BELOW GET CURRENT OCTAVE * * BRANCH IF OCTAVE > 1
3714 3715 3716 3717 3718 3719 3720	9B32 96 DE 9B34 81 Ø1 9B36 22 2C	* THE PLAY COMMAND. WHEN A NOTE IS DONE, * ROUTINE WILL RETURN CONTROL TO THE MAIN LEAU \$01,S * LDA OCTAVE CMPA #\$01 BHI L9B64 * OCTAVES I AND 2 USE A TWO BYTE DELAY TO S	STER TIMING REFERENCE FOR THE IRQ SERVICING PLAY COMMAND INTERPRETATION LOOP *LOAD U W/CURRENT VALUE OF (STACK POINTER+1) SO THAT THE STACK *POINTER WILL BE PROPERLY RESET WHEN IRQ VECTORS *YOU OUT OF THE PLAY TIMING ROUTINES BELOW GET CURRENT OCTAVE * * BRANCH IF OCTAVE > 1 SET THE PROPER FREQUENCY
3714 3715 3716 3717 3718 3719 3720 3721	9B32 96 DE 9B34 81 Ø1 9B36 22 2C 9B38 8E 9C 62	* THE PLAY COMMAND. WHEN A NOTE IS DONE, * ROUTINE WILL RETURN CONTROL TO THE MAIN LEAU \$01,S * LDA OCTAVE CMPA #\$01 BHI L9864 * OCTAVES 1 AND 2 USE A TWO BYTE DELAY TO S LDX #L9662	STER TIMING REFERENCE FOR THE IRQ SERVICING PLAY COMMAND INTERPRETATION LOOP *LOAD U W/CURRENT VALUE OF (STACK POINTER+1) SO THAT THE STACK *POINTER WILL BE PROPERLY RESET WHEN IRQ VECTORS *YOU OUT OF THE PLAY TIMING ROUTINES BELOW GET CURRENT OCTAVE * * BRANCH IF OCTAVE > 1 ET THE PROPER FREQUENCY POINT TO DELAY TABLE
3714 3715 3716 3717 3718 3719 3720 3721 3722	9B32 96 DE 9B34 81 Ø1 9B36 22 2C 9B38 8E 9C 62 9B3B C6 18	* THE PLAY COMMAND. WHEN A NOTE IS DONE, * ROUTINE WILL RETURN CONTROL TO THE MAIN LEAU \$01,S * LDA OCTAVE CMPA #\$01 BHI L9B64 * OCTAVES 1 AND 2 USE A TWO BYTE DELAY TO S LDX #1962 LDB #2*12	ISTER TIMING REFERENCE FOR THE IRQ SERVICING PLAY COMMAND INTERPRETATION LOOP *LOAD U W/CURRENT VALUE OF (STACK POINTER+1) SO THAT THE STACK *POINTER WILL BE PROPERLY RESET WHEN IRQ VECTORS *YOU OUT OF THE PLAY TIMING ROUTINES BELOW GET CURRENT OCTAVE * * BRANCH IF OCTAVE > 1 IET THE PROPER FREQUENCY POINT TO DELAY TABLE 24 BYTES DATA/OCTAVE
3714 3715 3716 3717 3718 3719 3720 3721 3722 3723	9B32 96 DE 9B34 81 01 9B36 22 2C 9B38 8E 9C 62 9B38 C6 18 9B3D 3D	* THE PLAY COMMAND. WHEN A NOTE IS DONE, * ROUTINE WILL RETURN CONTROL TO THE MAIN LEAU \$01,S * LDA OCTAVE CMPA #\$01 BHI L9B64 * OCTAVES 1 AND 2 USE A TWO BYTE DELAY TO S LDX #L9C62 LDB #2*12 MUL	ISTER TIMING REFERENCE FOR THE IRQ SERVICING PLAY COMMAND INTERPRETATION LOOP *LOAD U W/CURRENT VALUE OF (STACK POINTER+1) SO THAT THE STACK *POINTER WILL BE PROPERLY RESET WHEN IRQ VECTORS *YOU OUT OF THE PLAY TIMING ROUTINES BELOW GET CURRENT OCTAVE * * BRANCH IF OCTAVE > 1 SET THE PROPER FREQUENCY POINT TO DELAY TABLE 24 BYTES DATA/OCTAVE CALC OCTAVE TABLE OFFSET
3714 3715 3716 3717 3718 3719 3720 3721 3722	9B32 96 DE 9B34 81 01 9B36 22 2C 9B38 8E 9C 62 9B38 C6 18 9B3D 3D	* THE PLAY COMMAND. WHEN A NOTE IS DONE, * ROUTINE WILL RETURN CONTROL TO THE MAIN LEAU \$01,S * LDA OCTAVE CMPA #\$01 BHI L9B64 * OCTAVES 1 AND 2 USE A TWO BYTE DELAY TO S LDX #1962 LDB #2*12	ISTER TIMING REFERENCE FOR THE IRQ SERVICING PLAY COMMAND INTERPRETATION LOOP *LOAD U W/CURRENT VALUE OF (STACK POINTER+1) SO THAT THE STACK *POINTER WILL BE PROPERLY RESET WHEN IRQ VECTORS *YOU OUT OF THE PLAY TIMING ROUTINES BELOW GET CURRENT OCTAVE * * BRANCH IF OCTAVE > 1 IET THE PROPER FREQUENCY POINT TO DELAY TABLE 24 BYTES DATA/OCTAVE
3714 3715 3716 3717 3718 3719 3720 3721 3722 3723 3724	9B32 96 DE 9B34 81 Ø1 9B36 22 2C 9B38 8E 9C 62 9B3B C6 18 9B3D 3D 9B3E 3A	* THE PLAY COMMAND. WHEN A NOTE IS DONE, * ROUTINE WILL RETURN CONTROL TO THE MAIN LEAU \$01,S * LDA OCTAVE CMPA #\$01 BHI L9864 * OCTAVES 1 AND 2 USE A TWO BYTE DELAY TO S LDX #L9C62 LDB #2*12 MUL ABX	STER TIMING REFERENCE FOR THE IRQ SERVICING PLAY COMMAND INTERPRETATION LOOP *LOAD U W/CURRENT VALUE OF (STACK POINTER+1) SO THAT THE STACK *POINTER WILL BE PROPERLY RESET WHEN IRQ VECTORS *YOU OUT OF THE PLAY TIMING ROUTINES BELOW GET CURRENT OCTAVE * * BRANCH IF OCTAVE > 1 SET THE PROPER FREQUENCY POINT TO DELAY TABLE 24 BYTES DATA/OCTAVE CALC OCCTAVE TABLE OFFSET POINT TO CORRECT OCTAVE TABLE
3714 3715 3716 3717 3718 3719 3720 3721 3722 3723 3724 3725 3726 3727	9B32 96 DE 9B34 81 Ø1 9B36 22 2C 9B38 8E 9C 62 9B38 C6 18 9B3D 3D 9B3E 3A 9B3F 35 Ø4 9B41 58 9B42 3A	* THE PLAY COMMAND. WHEN A NOTE IS DONE, * ROUTINE WILL RETURN CONTROL TO THE MAIN LEAU \$01,S * LDA OCTAVE CMPA #\$01 BHI L9864 * OCTAVES 1 AND 2 USE A TWO BYTE DELAY TO S LDX #L9C62 LDB #2*12 MUL ABX PULS B ASLB ABX ABX ABX ABX ABX ABX ABX ABX ABX AB	ISTER TIMING REFERENCE FOR THE IRQ SERVICING PLAY COMMAND INTERPRETATION LOOP *LOAD U W/CURRENT VALUE OF (STACK POINTER+1) SO THAT THE STACK *POINTER WILL BE PROPERLY RESET WHEN IRQ VECTORS *YOU OUT OF THE PLAY TIMING ROUTINES BELOW GET CURRENT OCTAVE * * BRANCH IF OCTAVE > 1 SET THE PROPER FREQUENCY POINT TO DELAY TABLE 24 BYTES DATA/OCTAVE CALC OCTAVE TABLE CALC OCTAVE TABLE OFFICE OFFICE FOINT TO CORRECT OCTAVE TABLE GET NOTE VALUE BACK X 2 - 2 BYTES/NOTE POINT TO CORRECT NOTE
3714 3715 3716 3717 3718 3719 3720 3721 3722 3723 3724 3725 3726 3727 3728	9B32 96 DE 9B34 81 Ø1 9B36 22 2C 9B38 8E 9C 62 9B3B C6 18 9B3D 3D 9B3E 3A 9B3F 35 Ø4 9B41 58 9B42 3A	* THE PLAY COMMAND. WHEN A NOTE IS DONE, * ROUTINE WILL RETURN CONTROL TO THE MAIN LEAU \$01,S * LDA OCTAVE CMPA #\$01 BHI L9B64 * OCTAVES 1 AND 2 USE A TWO BYTE DELAY TO S LDX #19662 LDB #2*12 MUL ABX PULS B ASLB ABX LEAY,X	STER TIMING REFERENCE FOR THE IRQ SERVICING PLAY COMMAND INTERPRETATION LOOP *LOAD U W/CURRENT VALUE OF (STACK POINTER+1) SO THAT THE STACK *POINTER WILL BE PROPERLY RESET WHEN IRQ VECTORS *YOU OUT OF THE PLAY TIMING ROUTINES BELOW GET CURRENT OCTAVE * * BRANCH IF OCTAVE > 1 SET THE PROPER FREQUENCY POINT TO DELAY TABLE 24 BYTES DATA/OCTAVE CALC OCTAVE TABLE OFFSET POINT TO CORRECT OCTAVE TABLE GET NOTE VALUE BACK X 2 - 2 BYTES/NOTE POINT TO CORRECT NOTE GET POINTER TO Y REG (TFR X,Y)
3714 3715 3716 3717 3718 3719 3720 3721 3722 3723 3724 3725 3726 3727 3728 3729	9832 96 DE 9834 81 Ø1 9836 22 2C 9838 8E 9C 62 9838 C6 18 9830 3D 9835 3A 9837 35 Ø4 9841 58 9842 3A 9843 31 84 9843 31 84	* THE PLAY COMMAND. WHEN A NOTE IS DONE, * ROUTINE WILL RETURN CONTROL TO THE MAIN LEAU \$01,S * LDA OCTAVE CMPA #\$01 BHI L9B64 * OCTAVES 1 AND 2 USE A TWO BYTE DELAY TO S LDX #L9C62 LDB #2*12 MUL ABX PULS B ASLB ABX LEAY ,X BSR L9BBC	ISTER TIMING REFERENCE FOR THE IRQ SERVICING PLAY COMMAND INTERPRETATION LOOP *LOAD U W/CURRENT VALUE OF (STACK POINTER+1) SO THAT THE STACK *POINTER WILL BE PROPERLY RESET WHEN IRQ VECTORS *YOU OUT OF THE PLAY TIMING ROUTINES BELOW GET CURRENT OCTAVE * * BRANCH IF OCTAVE > 1 SET THE PROPER FREQUENCY POINT TO DELAY TABLE 24 BYTES DATA/OCTAVE CALC OCTAVE TABLE OFFSET POINT TO CORRECT OCTAVE TABLE GET NOTE VALUE BACK X 2 - 2 BYTES/NOTE POINT TO CORRECT NOTE GET POINTER TO Y REG (TFR X,Y) CALCULATE NOTE TIMER VALUE
3714 3715 3716 3717 3718 3719 3721 3722 3723 3724 3725 3726 3727 3728 3729 3730	9832 96 DE 9834 81 Ø1 9836 22 2C 9838 8E 9C 62 9838 C6 18 9830 3D 9835 3A 9837 35 Ø4 9841 58 9842 3A 9843 31 84 9843 80 45	* THE PLAY COMMAND. WHEN A NOTE IS DONE, * ROUTINE WILL RETURN CONTROL TO THE MAIN LEAU \$01,S * LDA OCTAVE CMPA #\$01 BHI L9864 * OCTAVES 1 AND 2 USE A TWO BYTE DELAY TO S LDX #L9C62 LDB #2*12 MUL ABX PULS B ASLB ABX LEAY ,X BSR L9B8C STD PLYTMR	ISTER TIMING REFERENCE FOR THE IRQ SERVICING PLAY COMMAND INTERPRETATION LOOP *LOAD U W/CURRENT VALUE OF (STACK POINTER+1) SO THAT THE STACK *POINTER WILL BE PROPERLY RESET WHEN IRQ VECTORS *YOU OUT OF THE PLAY TIMING ROUTINES BELOW GET CURRENT OCTAVE * * BRANCH IF OCTAVE > 1 SET THE PROPER FREQUENCY POINT TO DELAY TABLE 24 BYTES DATA/OCTAVE CALC OCTAVE TABLE OFFSET POINT TO CORRECT OCTAVE TABLE GET NOTE VALUE BACK X 2 - 2 BYTES/NOTE POINT TO CORRECT OCTAVE GET POINTER TO Y REG (TFR X,Y) CALCULATE NOTE TIMER VALUE SAVE IT
3714 3715 3716 3717 3718 3719 3720 3721 3722 3723 3724 3725 3726 3727 3728 3727 3738 3739 3730	9832 96 DE 9834 81 Ø1 9836 22 2C 9838 8E 9C 62 9838 C6 18 9830 3D 9835 3A 9837 35 Ø4 9841 58 9842 3A 9843 31 84 9843 31 84	* THE PLAY COMMAND. WHEN A NOTE IS DONE, * ROUTINE WILL RETURN CONTROL TO THE MAIN LEAU \$01,S * LDA OCTAVE CMPA #\$01 BHI L9B64 * OCTAVES 1 AND 2 USE A TWO BYTE DELAY TO S LDX #L9C62 LDB #2*12 MUL ABX PULS B ASLB ABX LEAY ,X BSR L9BBC STD PLYTMR * MAIN SOUND GENERATION LOOP - ONLY THE IRC	ISTER TIMING REFERENCE FOR THE IRQ SERVICING PLAY COMMAND INTERPRETATION LOOP *LOAD U W/CURRENT VALUE OF (STACK POINTER+1) SO THAT THE STACK *POINTER WILL BE PROPERLY RESET WHEN IRQ VECTORS *YOU OUT OF THE PLAY TIMING ROUTINES BELOW GET CURRENT OCTAVE * * BRANCH IF OCTAVE > 1 SET THE PROPER FREQUENCY POINT TO DELAY TABLE 24 BYTES DATA/OCTAVE CALC OCTAVE TABLE OFFSET POINT TO CORRECT OCTAVE TABLE GET NOTE VALUE BACK X 2 - 2 BYTES/NOTE POINT TO CORRECT OCTAVE GET POINTER TO Y REG (TFR X,Y) CALCULATE NOTE TIMER VALUE SAVE IT
3714 3715 3716 3717 3718 3719 3721 3722 3723 3724 3725 3726 3727 3728 3729 3730	9832 96 DE 9834 81 Ø1 9836 22 2C 9838 8E 9C 62 9838 C6 18 9830 3D 9835 3A 9837 35 Ø4 9841 58 9842 3A 9843 31 84 9843 31 84	* THE PLAY COMMAND. WHEN A NOTE IS DONE, * ROUTINE WILL RETURN CONTROL TO THE MAIN LEAU \$01,S * LDA OCTAVE CMPA #\$01 BHI L9864 * OCTAVES 1 AND 2 USE A TWO BYTE DELAY TO S LDX #L9C62 LDB #2*12 MUL ABX PULS B ASLB ABX LEAY ,X BSR L9B8C STD PLYTMR	ISTER TIMING REFERENCE FOR THE IRQ SERVICING THE IRQ SERVICING LAY COMMAND INTERPRETATION LOOP *LOAD U W/CURRENT VALUE OF (STACK POINTER+1) SO THAT THE STACK *POINTER WILL BE PROPERLY RESET WHEN IRQ VECTORS *YOU OUT OF THE PLAY TIMING ROUTINES BELOW GET CURRENT OCTAVE * * BRANCH IF OCTAVE > 1 SET THE PROPER FREQUENCY POINT TO DELAY TABLE 24 BYTES DATA/OCTAVE CALC OCTAVE TABLE OFFSET POINT TO CORRECT OCTAVE TABLE GET NOTE VALUE BACK X 2 - 2 BYTES/NOTE POINT TO CORRECT OCTAVE TABLE GET POINTER TO Y REG (TFR X,Y) CALCULATE NOTE TIMER VALUE SAVE IT I SERVICE WILL GET YOU OUT
3714 3715 3716 3717 3718 3719 3720 3721 3722 3723 3724 3725 3726 3727 3728 3729 3731 3731	9832 96 DE 9834 81 Ø1 9836 22 2C 9838 8E 9C 62 9838 06 18 9830 3D 9835 3A 9837 35 Ø4 9841 58 9842 3A 9843 31 84 9843 31 84 9845 8D 45 9847 DD E3	* THE PLAY COMMAND. WHEN A NOTE IS DONE, * ROUTINE WILL RETURN CONTROL TO THE MAIN LEAU \$01,S * LDA OCTAVE CMPA #\$01 BHI L9B64 * OCTAVES 1 AND 2 USE A TWO BYTE DELAY TO S LDX #L9C62 LDB #2*12 MUL ABX PULS B ASLB ABX LEAY ,X BSR L9B8C STD PLYTMR * MAIN SOUND GENERATION LOOP - ONLY THE IRQ * OF THIS LOOP (OCTAVES 1 AND 2)	ISTER TIMING REFERENCE FOR THE IRQ SERVICING PLAY COMMAND INTERPRETATION LOOP *LOAD U W/CURRENT VALUE OF (STACK POINTER+1) SO THAT THE STACK *POINTER WILL BE PROPERLY RESET WHEN IRQ VECTORS *YOU OUT OF THE PLAY TIMING ROUTINES BELOW GET CURRENT OCTAVE * * BRANCH IF OCTAVE > 1 SET THE PROPER FREQUENCY POINT TO DELAY TABLE 24 BYTES DATA/OCTAVE CALC OCTAVE TABLE OFFSET POINT TO CORRECT OCTAVE TABLE GET NOTE VALUE BACK X 2 - 2 BYTES/NOTE POINT TO CORRECT OCTAVE GET POINTER TO Y REG (TFR X,Y) CALCULATE NOTE TIMER VALUE SAVE IT
3714 3715 3716 3717 3718 3719 3720 3721 3722 3723 3724 3725 3726 3727 3728 3739 3731 3732 3733	9B32 96 DE 9B34 81 Ø1 9B36 22 2C 9B38 8E 9C 62 9B38 C6 18 9B30 3D 9B35 3A 9B37 35 Ø4 9B41 58 9B42 3A 9B43 31 84 9B45 8D Ø5 9B49 8D ØC	* THE PLAY COMMAND. WHEN A NOTE IS DONE, * ROUTINE WILL RETURN CONTROL TO THE MAIN LEAU \$01,S * LDA OCTAVE CMPA #\$01 BHI L9B64 * OCTAVES 1 AND 2 USE A TWO BYTE DELAY TO S LDX #L9C62 LDB #2*12 MUL ABX PULS B ASLB ABX LEAY, X BSR L9BBC STD PLYTMR * MAIN SOUND GENERATION LOOP - ONLY THE IRC * OF THIS LOOP (OCTAVES 1 AND 2) L9B49 BSR L9B57	ISTER TIMING REFERENCE FOR THE IRQ SERVICING PLAY COMMAND INTERPRETATION LOOP *LOAD U W/CURRENT VALUE OF (STACK POINTER+1) SO THAT THE STACK *POINTER WILL BE PROPERLY RESET WHEN IRQ VECTORS *YOU OUT OF THE PLAY TIMING ROUTINES BELOW GET CURRENT OCTAVE * * BRANCH IF OCTAVE > 1 SET THE PROPER FREQUENCY POINT TO DELAY TABLE 24 BYTES DATA/OCTAVE CALC OCTAVE TABLE OFFSET POINT TO CORRECT OCTAVE TABLE GET NOTE VALUE BACK X 2 - 2 BYTES/NOTE POINT TO CORRECT OCTAVE GET POINTER TO Y REG (TFR X,Y) CALCULATE NOTE TIMER VALUE SAVE IT SERVICE WILL GET YOU OUT MID VALUE TO DA AND WAIT
3714 3715 3716 3717 3718 3719 3720 3721 3722 3723 3724 3725 3726 3727 3728 3729 3730 3731 3732 3733 3734 3733	9B32 96 DE 9B34 81 01 9B36 22 2C 9B38 8E 9C 62 9B38 C6 18 9B30 3D 9B3E 3A 9B3F 35 04 9B41 58 9B42 3A 9B43 31 84 9B43 31 84 9B45 8D 45 9B47 DD E3 9B49 8D 0C 9B48 96 DF 9B40 8D 0B 9B4F 8D 06	* THE PLAY COMMAND. WHEN A NOTE IS DONE, * ROUTINE WILL RETURN CONTROL TO THE MAIN LEAU \$01,S * LDA OCTAVE CMPA #\$01 BHI L9B64 * OCTAVES 1 AND 2 USE A TWO BYTE DELAY TO S LDX #L9C62 LDB #2*12 MUL ABX PULS B ASLB ABX LEAY ,X BSR 19BBC STD PLYTMR * MAIN SOUND GENERATION LOOP - ONLY THE IRC * OF THIS LOOP (OCTAVES 1 AND 2) L9B49 BSR L9B57 LDA VOLHI BSR L9B5A BSR L9B5A BSR L9B57	ISTER TIMING REFERENCE FOR THE IRQ SERVICING PLAY COMMAND INTERPRETATION LOOP *LOAD U W/CURRENT VALUE OF (STACK POINTER+1) SO THAT THE STACK *POINTER WILL BE PROPERLY RESET WHEN IRQ VECTORS *YOU OUT OF THE PLAY TIMING ROUTINES BELOW GET CURRENT OCTAVE * * BRANCH IF OCTAVE > 1 SET THE PROPER FREQUENCY POINT TO DELAY TABLE 24 BYTES DATA/OCTAVE CALC OCTAVE TABLE OFFSET POINT TO CORRECT OCTAVE TABLE GET NOTE VALUE BACK X 2 - 2 BYTES/NOTE POINT TO CORRECT NOTE GET POINTER TO Y REG (TFR X,Y) CALCULATE NOTE TIMER VALUE SAVE IT 1 SERVICE WILL GET YOU OUT MID VALUE TO DA AND WAIT MID VALUE TO DA AND WAIT MID VALUE TO DA AND WAIT
3714 3715 3716 3717 3718 3719 3720 3721 3722 3723 3724 3725 3726 3727 3728 3729 3731 3732 3731 3732 3733 3734 3735	9B32 96 DE 9B34 81 Ø1 9B36 22 2C 9B38 8E 9C 62 9B38 C6 18 9B3D 3D 9B3E 3A 9B3F 35 Ø4 9B41 58 9B42 3A 9B43 31 84 9B45 8D 45 9B47 DD E3 9B49 8D ØC 9B48 96 DF 9B40 8D ØB 9B4F 8D ØB	* THE PLAY COMMAND. WHEN A NOTE IS DONE, * ROUTINE WILL RETURN CONTROL TO THE MAIN LEAU \$01,S * LDA OCTAVE CMPA #\$01 BHI L9B64 * OCTAVES 1 AND 2 USE A TWO BYTE DELAY TO S LDX #L9C62 LDB #2*12 MUL ABX PULS B ASLB ABX PULS B ASLB ABX LEAY ,X BSR L9BBC STD PLYTMR * MAIN SOUND GENERATION LOOP - ONLY THE IRO * OF THIS LOOP (OCTAVES 1 AND 2) L9B49 BSR L9B57 LDA VOLHU BSR L9B57 LDA VOLHU BSR L9B57 LDA VOLHOW	ISTER TIMING REFERENCE FOR THE IRQ SERVICING THE IRQ SERVICING LAY COMMAND INTERPRETATION LOOP *LOAD U W/CURRENT VALUE OF (STACK POINTER+1) SO THAT THE STACK *POINTER WILL BE PROPERLY RESET WHEN IRQ VECTORS *YOU OUT OF THE PLAY TIMING ROUTINES BELOW GET CURRENT OCTAVE * BRANCH IF OCTAVE > 1 SET THE PROPER FREQUENCY POINT TO DELAY TABLE 24 BYTES DATA/OCTAVE CALC OCTAVE TABLE OFFSET POINT TO CORRECT OCTAVE TABLE GET NOTE VALUE BACK X 2 - 2 BYTES/NOTE POINT TO CORRECT NOTE GET POINTER TO Y REG (TFR X,Y) CALCULATE NOTE TIMER VALUE SAVE IT SERVICE WILL GET YOU OUT MID VALUE TO DA AND WAIT GET HIGH VALUE STORE TO DA AND WAIT GET LOW VALUE
3714 3715 3716 3717 3718 3720 3721 3722 3723 3724 3725 3726 3727 3728 3731 3732 3731 3732 3733 3734 3735 3736 3737 3738	9832 96 DE 9834 81 01 9836 22 2C 9838 8E 9C 62 9838 C6 18 9830 3D 9835 3A 9837 35 04 9847 58 9842 3A 9843 31 84 9843 31 84 9845 8D 45 9847 DD E3 9849 8D 0C 9849 96 DF 9840 8D 0B 9847 8D 06 9851 96 E0 9853 8D 05	* THE PLAY COMMAND. WHEN A NOTE IS DONE, * ROUTINE WILL RETURN CONTROL TO THE MAIN LEAU \$01,S * LDA OCTAVE CMPA #\$01 BHI L9B64 * OCTAVES 1 AND 2 USE A TWO BYTE DELAY TO S LDX #L9C62 LDB #2*12 MUL ABX PULS B ASLB ABX LEAY ,X BSR L9B8C STD PLYTMR * MAIN SOUND GENERATION LOOP - ONLY THE IRQ * OF THIS LOOP (OCTAVES 1 AND 2) L9B49 BSR L9B57 LDA VOLHI BSR L9B57 LDA VOLLOW BSR L9B57 LDA VOLLOW BSR L9B55	ISTER TIMING REFERENCE FOR THE IRQ SERVICING THE IRQ SERVICING LAY COMMAND INTERPRETATION LOOP *LOAD U W/CURRENT VALUE OF (STACK POINTER+1) SO THAT THE STACK *POINTER WILL BE PROPERLY RESET WHEN IRQ VECTORS *YOU OUT OF THE PLAY TIMING ROUTINES BELOW GET CURRENT OCTAVE * BRANCH IF OCTAVE > 1 SET THE PROPER FREQUENCY POINT TO DELAY TABLE 24 BYTES DATA/OCTAVE CALC OCTAVE TABLE OFFSET POINT TO CORRECT OCTAVE TABLE GET NOTE VALUE BACK X 2 - 2 BYTES/NOTE POINT TO CORRECT NOTE GET POINTER TO Y REG (TFR X,Y) CALCULATE NOTE TIMER VALUE SAVE IT ISERVICE WILL GET YOU OUT MID VALUE TO DA AND WAIT GET HIGH VALUE STORE TO DA AND WAIT GET LOW VALUE STORE TO DA AND WAIT GET LOW VALUE STORE
3714 3715 3716 3717 3718 3720 3721 3722 3723 3724 3725 3726 3727 3728 3729 3730 3731 3732 3733 3734 3735 3736 3737 3738	9832 96 DE 9834 81 01 9836 22 2C 9838 8E 9C 62 9838 C6 18 9830 3D 9835 3A 9837 35 04 9841 58 9842 3A 9843 31 84 9843 31 84 9843 31 84 9845 8D 45 9847 DD E3 9849 8D 0C 9848 96 DF 9840 8D 0B 9851 96 E0 9853 8D 05 9853 8D 05 9855 20 F2	* THE PLAY COMMAND. WHEN A NOTE IS DONE, * ROUTINE WILL RETURN CONTROL TO THE MAIN LEAU \$01,S * LDA OCTAVE CMPA #\$01 BHI L9B64 * OCTAVES 1 AND 2 USE A TWO BYTE DELAY TO S LDX #L9C62 LDB #2*12 MUL ABX PULS B ASLB ABX LEAY ,X BSR L9BBC STD PLYTMR * MAIN SOUND GENERATION LOOP - ONLY THE IRC * OF THIS LOOP (OCTAVES 1 AND 2) L9B49 BSR L9B57 LDA VOLHT BSR L9B5A BSR L9B57 LDA VOLLOW BSR L9B5A BSR L9B5BA	ISTER TIMING REFERENCE FOR THE IRQ SERVICING PLAY COMMAND INTERPRETATION LOOP *LOAD U W/CURRENT VALUE OF (STACK POINTER+1) SO THAT THE STACK *POINTER WILL BE PROPERLY RESET WHEN IRQ VECTORS *YOU OUT OF THE PLAY TIMING ROUTINES BELOW GET CURRENT OCTAVE * * BRANCH IF OCTAVE > 1 SET THE PROPER FREQUENCY POINT TO DELAY TABLE 24 BYTES DATA/OCTAVE CALC OCTAVE TABLE OFFSET POINT TO CORRECT OCTAVE TABLE GET NOTE VALUE BACK X 2 - 2 BYTES/NOTE POINT TO CORRECT NOTE GET POINTER TO Y REG (TFR X,Y) CALCULATE NOTE TIMER VALUE SAVE IT 1 SERVICE WILL GET YOU OUT MID VALUE TO DA AND WAIT GET HIGH VALUE STORE TO DA AND WAIT GET LOW VALUE STORE STORE KEEP LOOPING
3714 3715 3716 3717 3718 3719 3720 3721 3722 3723 3724 3725 3726 3727 3728 3729 3731 3732 3731 3732 3733 3734 3735 3736	9B32 96 DE 9B34 81 01 9B36 22 2C 9B38 8E 9C 62 9B38 C6 18 9B3D 3D 9B3E 3A 9B3F 35 04 9B41 58 9B42 3A 9B43 31 84 9B45 8D 45 9B47 DD E3 9B49 8D 0C 9B48 96 DF 9B40 8D 0B 9B4F 8D 06 9B51 96 E0 9B53 8D 05 9B55 20 F2 9B57 86 7E	* THE PLAY COMMAND. WHEN A NOTE IS DONE, * ROUTINE WILL RETURN CONTROL TO THE MAIN LEAU \$01,S * LDA OCTAVE CMPA #\$01 BHI L9B64 * OCTAVES 1 AND 2 USE A TWO BYTE DELAY TO S LDX #L9C62 LDB #2*12 MUL ABX PULS B ASLB ABX LEAY ,X BSR L9B8C STD PLYTMR * MAIN SOUND GENERATION LOOP - ONLY THE IRO * OF THIS LOOP (OCTAVES 1 AND 2) L9B49 BSR L9B57 LDA VOLHI BSR L9B57 LDA VOLHO BSR L9B57 LDA VOLHO BSR L9B58 BSR L9B57 LDA VOLLOW BSR L9B59 LDA #857E	ISTER TIMING REFERENCE FOR THE IRQ SERVICING PLAY COMMAND INTERPRETATION LOOP *LOAD U W/CURRENT VALUE OF (STACK POINTER+1) SO THAT THE STACK *POINTER WILL BE PROPERLY RESET WHEN IRQ VECTORS *YOU OUT OF THE PLAY TIMING ROUTINES BELOW GET CURRENT OCTAVE * BRANCH IF OCTAVE > 1 SET THE PROPER FREQUENCY POINT TO DELAY TABLE 24 BYTES DATA/OCTAVE CALC OCTAVE TABLE OFFSET POINT TO CORRECT OCTAVE TABLE GET NOTE VALUE BACK X 2 - 2 BYTES/NOTE POINT TO CORRECT NOTE GET POINTER TO Y REG (TFR X,Y) CALCULATE NOTE TIMER VALUE SAVE IT SERVICE WILL GET YOU OUT MID VALUE TO DA AND WAIT GET HIGH VALUE STORE TO DA AND WAIT GET HIGH VALUE STORE KEEP LOOPING DA MID VALUE AND RS 232 MARKING
3714 3715 3716 3717 3718 3720 3721 3722 3723 3724 3725 3726 3727 3728 3729 3730 3731 3732 3733 3734 3735 3736 3737 3738	9832 96 DE 9834 81 01 9836 22 2C 9838 8E 9C 62 9838 06 18 9830 3D 9835 3A 9837 35 04 9841 58 9842 3A 9843 31 84 9843 31 84 9845 8D 45 9847 DD E3 9849 8D 0C 9848 96 DF 9840 8D 0B 9847 8D 06 9851 96 E0 9853 8D 05 9855 20 F2 9857 86 7E 9859 12	* THE PLAY COMMAND. WHEN A NOTE IS DONE, * ROUTINE WILL RETURN CONTROL TO THE MAIN LEAU \$01,S * LDA OCTAVE CMPA #\$01 BHI L9B64 * OCTAVES 1 AND 2 USE A TWO BYTE DELAY TO S LDX #L9C62 LDB #2*12 MUL ABX PULS B ASLB ABX LEAY ,X BSR L9BBC STD PLYTMR * MAIN SOUND GENERATION LOOP - ONLY THE IRC * OF THIS LOOP (OCTAVES 1 AND 2) L9B49 BSR L9B57 LDA VOLHT BSR L9B5A BSR L9B57 LDA VOLLOW BSR L9B5A BSR L9B5BA	ISTER TIMING REFERENCE FOR THE IRQ SERVICING PLAY COMMAND INTERPRETATION LOOP *LOAD U W/CURRENT VALUE OF (STACK POINTER+1) SO THAT THE STACK *POINTER WILL BE PROPERLY RESET WHEN IRQ VECTORS *YOU OUT OF THE PLAY TIMING ROUTINES BELOW GET CURRENT OCTAVE * * BRANCH IF OCTAVE > 1 SET THE PROPER FREQUENCY POINT TO DELAY TABLE 24 BYTES DATA/OCTAVE CALC OCTAVE TABLE OFFSET POINT TO CORRECT OCTAVE TABLE GET NOTE VALUE BACK X 2 - 2 BYTES/NOTE POINT TO CORRECT NOTE GET POINTER TO Y REG (TFR X,Y) CALCULATE NOTE TIMER VALUE SAVE IT 1 SERVICE WILL GET YOU OUT MID VALUE TO DA AND WAIT GET HIGH VALUE STORE TO DA AND WAIT GET LOW VALUE STORE STORE KEEP LOOPING
3714 3715 3716 3717 3718 3720 3721 3722 3723 3724 3725 3726 3727 3728 3731 3732 3731 3732 3733 3734 3735 3736 3737 3738 3739 3740	9B32 96 DE 9B34 81 Ø1 9B36 22 2C 9B38 8E 9C 62 9B38 C6 18 9B3D 3D 9B3E 3A 9B3F 35 Ø4 9B41 58 9B42 3A 9B43 31 84 9B45 8D 45 9B47 DD E3 9B49 8D ØC 9B49 8D ØC 9B49 8D ØB 9B47 BD ØB 9B47 BD ØB 9B47 BD ØB 9B48 96 DF 9B49 8D ØB 9B51 96 EØ 9B53 8D Ø5 9B57 86 7E 9B57 86 7E 9B58 B7 FF 2Ø	* THE PLAY COMMAND. WHEN A NOTE IS DONE, * ROUTINE WILL RETURN CONTROL TO THE MAIN LEAU \$01,S * LDA OCTAVE CMPA #\$01 BHI L9B64 * OCTAVES 1 AND 2 USE A TWO BYTE DELAY TO S LDX #L9C62 LDB #2*12 MUL ABX PULS B ASLB ABX LEAY ,X BSR L9B8C STD PLYTMR * MAIN SOUND GENERATION LOOP - ONLY THE IRQ * OF THIS LOOP (OCTAVES 1 AND 2) L9B49 BSR L9B57 LDA VOLHI BSR L9B57 LDA VOLHI BSR L9B57 LDA VOLLOW BSR L9B58 BSR L9B57 LDA VOLLOW BSR L9B59 LDA WSTE NOP L9B5A STA PIA1 LDX , Y	ISTER TIMING REFERENCE FOR THE IRQ SERVICING PLAY COMMAND INTERPRETATION LOOP *LOAD U W/CURRENT VALUE OF (STACK POINTER+1) SO THAT THE STACK *POINTER WILL BE PROPERLY RESET WHEN IRQ VECTORS *YOU OUT OF THE PLAY TIMING ROUTINES BELOW GET CURRENT OCTAVE * * BRANCH IF OCTAVE > 1 SET THE PROPER FREQUENCY POINT TO DELAY TABLE 24 BYTES DATA/OCTAVE CALC OCTAVE TABLE GET NOTE VALUE BACK X 2 - 2 BYTES/NOTE POINT TO CORRECT OCTAVE TABLE GET POINTER TO Y REG (TFR X,Y) CALCULATE NOTE TIMER VALUE SAVE IT SERVICE WILL GET YOU OUT MID VALUE TO DA AND WAIT GET HIGH VALUE STORE TO DA AND WAIT GET LOW VALUE STORE KEEP LOOPING DA MID VALUE AND RS 232 MARKING DELAY SOME - FINE TUNE PLAY FREQUENCY
3714 3715 3716 3717 3718 3720 3721 3722 3723 3724 3725 3726 3727 3728 3729 3730 3731 3732 3733 3734 3735 3736 3737 3738 3739 3737 3738	9B32 96 DE 9B34 81 Ø1 9B36 22 2C 9B38 8E 9C 62 9B38 C6 18 9B3D 3D 9B3E 3A 9B3F 35 Ø4 9B41 58 9B42 3A 9B43 31 84 9B45 8D 45 9B47 DD E3 9B49 8D ØC 9B49 8D ØC 9B49 8D ØB 9B47 BD ØB 9B47 BD ØB 9B47 BD ØB 9B48 96 DF 9B49 8D ØB 9B51 96 EØ 9B53 8D Ø5 9B57 86 7E 9B57 86 7E 9B58 B7 FF 2Ø	* THE PLAY COMMAND. WHEN A NOTE IS DONE, * ROUTINE WILL RETURN CONTROL TO THE MAIN LEAU \$01,S * LDA OCTAVE CMPA #\$01 BHI L9B64 * OCTAVES 1 AND 2 USE A TWO BYTE DELAY TO S LDX #L9C62 LDB #2*12 MUL ABX PULS B ASLB ABX LEAY ,X BSR L9B8C STD PLYTMR * MAIN SOUND GENERATION LOOP - ONLY THE IRQ * OF THIS LOOP (OCTAVES 1 AND 2) L9B49 BSR L9B57 LDA VOLHI BSR L9B57 LDA VOLHI BSR L9B57 LDA VOLLOW BSR L9B57 LDA VOLLOW BSR L9B57 LDA VOLLOW BSR L9B57 LDA WOLLOW BSR L9B55 BRA L9B49 L9B57 LDA #\$7E NOP L9B5A STA PIA1	ISTER TIMING REFERENCE FOR THE IRQ SERVICING PLAY COMMAND INTERPRETATION LOOP *LOAD U W/CURRENT VALUE OF (STACK POINTER+1) SO THAT THE STACK *POINTER WILL BE PROPERLY RESET WHEN IRQ VECTORS *YOU OUT OF THE PLAY TIMING ROUTINES BELOW GET CURRENT OCTAVE * * BRANCH IF OCTAVE > 1 SET THE PROPER FREQUENCY POINT TO DELAY TABLE 24 BYTES DATA/OCTAVE CALC OCTAVE TABLE OFFSET POINT TO CORRECT OCTAVE TABLE GET NOTE VALUE BACK X 2 - 2 BYTES/NOTE POINT TO CORRECT NOTE GET POINTER TO Y REG (TFR X,Y) CALCULATE NOTE TIMER VALUE SAVE IT SERVICE WILL GET YOU OUT MID VALUE TO DA AND WAIT GET HIGH VALUE STORE TO DA AND WAIT GET LOW VALUE STORE KEEP LOOPING DA MID VALUE AND RS 232 MARKING DELAY SOME - FINE TUNE PLAY FREQUENCY STORE TO DA CONVERTER

3745	9B61 26 FC		BNE L9B5F	*COUNT X TO ZERO - PROGRAMMABLE DELAY
3746	9B63 39		RTS	
3747 3748	9B64 8E 9C 7A		S 3,4 AND 5 USE A ONE BYTE DELAY TO LDX #L9C92-2*12	D SET THE PROPER FREQUENCY POINT TO DELAY TABLE
3749	9B67 C6 ØC	L3004	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 3754	9B6D 3A		ABX	POINT TO CORRECT NOTE
3755	9B6E 8D 1C 9B7Ø DD E3		BSR L9B8C STD PLYTMR	CALCULATE NOTE TIMER VALUE 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 LDA VOLLOW	MID VALUE TO DA AND WAIT
376Ø 3761	9B7A 96 EØ 9B7C 8D Ø5		BSR L9B83	GET LOW VALUE STORE TO DA AND WAIT
3762	9B7E 2Ø F2		BRA L9B72	KEEP GOING
3763			D VALUE TO DA CONVERTER AND WAIT A	
3764	9B8Ø 86 7E	L9B8Ø	LDA #\$7E	DA CONVERTER MID VALUE AND KEEP RS 232 OUTPUT MARKING
3765 3766	9B82 12 9B83 B7 FF 20	L9B83	NOP STA PIA1	DELAY SOME - FINE TUNE PLAY FREQUENCY STORE IN DA CONVERTER
3767	9B86 A6 84	L9003	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	+ CALCIII	RTS	VALUE IN ACCD
3771 3772			ATE NOTE TIMER VALUE - RETURN WITH RGER ACCD IS, THE LONGER THE NOTE W	
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 3777	9B92 8B Ø2 9B94 3D		ADDA #\$Ø2 MUL	ADD IN CONSTANT TIMER SCALE FACTOR 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		+ CET NE	VT COMMAND DETUDN VALUE IN ACCA	
	9B98 34 1Ø		XT COMMAND - RETURN VALUE IN ACCA 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 3788	9BAØ A6 8Ø 9BA2 9F D9		LDA ,X+ STX VD9	GET COMMAND SAVE NEW ADDRESS
3789	9BA4 ØA D8		DEC VD8	DECREMENT COMMAND CTR
	9BA6 81 20		CMPA #SPACE	CHECK FOR BLANK
	9BA8 27 FØ		BEQ L9B9A	IGNORE BLANKS
	9BAA 35 9Ø		PULS X,PC	RESTORE X RESISTER AND RETURN
3793 3794		* FVAI 11A	TE THE >,<,+,-,= OPERATORS - ENTER	WITH THE VALUE TO
3795			RATED ON IN ACCB, RETURN NEW VALUE	
3796	9BAC 8D EA	L9BAC	BSR L9B98	GET A COMMAND CHARACTER
3797			CMPA #'+'	ADD ONES
	9BAE 81 2B			ADD ONE?
3798	9BBØ 27 3C		BEQ L9BEE	YES
	9BBØ 27 3C 9BB2 81 2D			
3798 3799	9BBØ 27 3C		BEQ L9BEE CMPA #'-'	YES SUBTRACT ONE?
3798 3799 3800 3801 3802	9BBØ 27 3C 9BB2 81 2D 9BB4 27 3C 9BB6 81 3E 9BB8 27 42		BEQ L9BEE CMPA #'-' BEQ L9BF2 CMPA #'>' BEQ L9BFC	YES SUBTRACT ONE? YES MULTIPLY BY TWO? YES
3798 3799 3800 3801 3802 3803	9BBØ 27 3C 9BB2 81 2D 9BB4 27 3C 9BB6 81 3E 9BB8 27 42 9BBA 81 3C		BEQ L9BEE CMPA #'-' BEQ L9BF2 CMPA #'>' BEQ L9BFC CMPA # < '	YES SUBTRACT ONE? YES MULTIPLY BY TWO? YES DIVIDE BY TWO?
3798 3799 3800 3801 3802 3803 3804	9BBØ 27 3C 9BB2 81 2D 9BB4 27 3C 9BB6 81 3E 9BB8 27 42 9BBA 81 3C 9BBC 27 39		BEQ L98EE CMPA #'-' BEQ L98F2 CMPA #'>' BEQ L98FC CMPA #'<' BEQ L98FC	YES SUBTRACT ONE? YES MULTIPLY BY TWO? YES DIVIDE BY TWO? YES
3798 3799 3800 3801 3802 3803	9BBØ 27 3C 9BB2 81 2D 9BB4 27 3C 9BB6 81 3E 9BB8 27 42 9BBA 81 3C		BEQ L9BEE CMPA #'-' BEQ L9BF2 CMPA #'>' BEQ L9BFC CMPA # < '	YES SUBTRACT ONE? YES MULTIPLY BY TWO? YES DIVIDE BY TWO?
3798 3799 3800 3801 3802 3803 3804 3805	98BØ 27 3C 98B2 81 2D 98B4 27 3C 98B6 81 3E 98B8 27 42 98BA 81 3C 98BC 27 39 98BE 81 3D	L9BBE *	BEQ L9BEE CMPA #'-' BEQ L9BF2 CMPA #'>' BEQ L9BFC CMPA #'<' BEQ L9BF7 CMPA #'='	YES SUBTRACT ONE? YES MULTIPLY BY TWO? YES DIVIDE BY TWO? YES *CHECK FOR VARIABLE EQUATE - BRANCH IF SO; ACCB WILL BE
3798 3799 3800 3801 3802 3803 3804 3805 3806 3807 3808	98BØ 27 3C 98B2 81 2D 98B4 27 3C 98B6 81 3E 98B8 27 42 98B8 81 3C 98BC 27 39 98BE 81 3D 98CØ 27 3F		BEQ L9BEE CMPA #'-' BEQ L9BF2 CMPA #'>' BEQ L9BFC CMPA #'<' BEQ L9BF7 CMPA #'=' BEQ L9CØ1	YES SUBTRACT ONE? YES MULTIPLY BY TWO? YES DIVIDE BY TWO? YES *CHECK FOR VARIABLE EQUATE - BRANCH IF SO; ACCB WILL BE *SET TO THE VALUE OF THE BASIC VARIABLE IN THE COMMAND *STRING WHICH MUST BE NUMERIC, LESS THAN 256 *AND THE VARIABLE MUST BE FOLLOWED BY A SEMICOLON.
3798 3799 3800 3801 3802 3803 3804 3805 3806 3807 3808 3809	98B0 27 3C 98B2 27 3C 98B4 27 3C 98B6 81 3E 98B8 27 42 98BA 81 3C 98BC 27 39 98BC 27 39 98C2 7 3F	L9BBE *	BEQ L98EE CMPA #'-' BEQ L98F2 CMPA #'>' BEQ L98FC CMPA #'<' BEQ L98F7 CMPA #'=' BEQ L9CØ1 JSR L9ØAA	YES SUBTRACT ONE? YES MULTIPLY BY TWO? YES DIVIDE BY TWO? YES *CHECK FOR VARIABLE EQUATE - BRANCH IF SO; ACCB WILL BE *SET TO THE VALUE OF THE BASIC VARIABLE IN THE COMMAND *STRING WHICH MUST BE NUMERIC, LESS THAN 256 *AND THE VARIABLE MUST BE FOLLOWED BY A SEMICOLON. CLEAR CARRY IF NUMERIC
3798 3799 3800 3801 3802 3803 3804 3805 3806 3807 3808	98BØ 27 3C 98B2 81 2D 98B4 27 3C 98B6 81 3E 98B8 27 42 98B8 81 3C 98BC 27 39 98BE 81 3D 98CØ 27 3F	L9BBE *	BEQ L9BEE CMPA #'-' BEQ L9BF2 CMPA #'>' BEQ L9BFC CMPA #'<' BEQ L9BF7 CMPA #'=' BEQ L9CØ1	YES SUBTRACT ONE? YES MULTIPLY BY TWO? YES DIVIDE BY TWO? YES *CHECK FOR VARIABLE EQUATE - BRANCH IF SO; ACCB WILL BE *SET TO THE VALUE OF THE BASIC VARIABLE IN THE COMMAND *STRING WHICH MUST BE NUMERIC, LESS THAN 256 *AND THE VARIABLE MUST BE FOLLOWED BY A SEMICOLON.
3798 3799 3800 3801 3802 3803 3804 3805 3806 3807 3808 3809 3810 3811	98BØ 27 3C 98B2 81 2D 98B4 27 3C 98B6 81 3E 98B8 27 42 98BA 81 3C 98BC 27 39 98BE 81 3D 98CØ 27 3F 98CØ 27 3F	L9BBE * * * * * * * * * * * * *	BEQ L9BEE CMPA #'-' BEQ L9BF2 CMPA #'>' BEQ L9BFC CMPA #'<' BEQ L9BF7 CMPA #'=' BEQ L9C01 JSR L9C0A BLO L9BEB CLRB A DECIMAL ASCII VALUE OFF OF THE CO	YES SUBTRACT ONE? YES MULTIPLY BY TWO? YES DIVIDE BY TWO? YES *CHECK FOR VARIABLE EQUATE - BRANCH IF SO; ACCB WILL BE *SET TO THE VALUE OF THE BASIC VARIABLE IN THE COMMAND *STRING WHICH MUST BE NUMERIC, LESS THAN 256 *AND THE VARIABLE MUST BE FOLLOWED BY A SEMICOLON. CLEAR CARRY IF NUMERIC FC ERROR IF NON NUMERIC UNITS DIGIT = 0
3798 3799 3800 3801 3802 3803 3804 3805 3806 3807 3810 3811 3811 3812 3813	98BØ 27 3C 98B2 81 2D 98B4 27 3C 98B6 81 3E 98B8 27 42 98BA 81 3C 98BC 27 39 98BE 81 3D 98CØ 27 3F 98CØ 27 3F	L9BBE * * * STRIP * AND RE	BEQ L9BEE CMPA #'-' BEQ L9BF2 CMPA #'>' BEQ L9BFC CMPA #'<' BEQ L9BF7 CMPA #'=' BEQ L9CØ1 JSR L9ØAA BLO L9BEB CLRB A DECIMAL ASCII VALUE OFF OF THE CO	YES SUBTRACT ONE? YES MULTIPLY BY TWO? YES DIVIDE BY TWO? YES *CHECK FOR VARIABLE EQUATE - BRANCH IF SO; ACCB WILL BE *SET TO THE VALUE OF THE BASIC VARIABLE IN THE COMMAND *STRING WHICH MUST BE NUMERIC, LESS THAN 256 *AND THE VARIABLE MUST BE FOLLOWED BY A SEMICOLON. CLEAR CARRY IF NUMERIC FC ERROR IF NON NUMERIC UNITS DIGIT = Ø DMMAND STRING
3798 3799 3800 3801 3802 3803 3804 3805 3806 3807 3809 3810 3811 3812 3813 3814	98B0 27 3C 98B2 21 2D 98B4 27 3C 98B6 81 3E 98B8 27 42 98BA 81 3C 98BC 27 39 98BC 81 3D 98C0 27 3F 98C2 BD 90 AA 98C5 25 24 98C7 5F	L9BBE * * * * * * * * * * * * *	BEQ L9BEE CMPA #'-' BEQ L9BF2 CMPA #'>' BEQ L9BFC CMPA #'<' BEQ L9BF7 CMPA #'=' BEQ L9CØ1 JSR L9ØAA BLO L9BEB CLRB A DECIMAL ASCII VALUE OFF OF THE CO	YES SUBTRACT ONE? YES MULTIPLY BY TWO? YES DIVIDE BY TWO? YES *CHECK FOR VARIABLE EQUATE - BRANCH IF SO; ACCB WILL BE *SET TO THE VALUE OF THE BASIC VARIABLE IN THE COMMAND *STRING WHICH MUST BE NUMERIC, LESS THAN 256 *AND THE VARIABLE MUST BE FOLLOWED BY A SEMICOLON. CLEAR CARRY IF NUMERIC FC ERROR IF NOM NUMERIC UNITS DIGIT = Ø DMMAND STRING MASK OFF ASCII
3798 3799 3809 3801 3802 3803 3804 3805 3806 3809 3810 3811 3812 3813 3814	98BØ 27 3C 98B2 81 2D 98B4 27 3C 98B6 81 3E 98B8 27 42 98BA 81 3C 98BC 27 39 98BE 81 3D 98CØ 27 3F 98CØ 27 3F	L9BBE * * * STRIP * AND RE	BEQ L9BEE CMPA #'-' BEQ L9BF2 CMPA #'>' BEQ L9BFC CMPA #'<' BEQ L9BF7 CMPA #'=' BEQ L9CØ1 JSR L9ØAA BLO L9BEB CLRB A DECIMAL ASCII VALUE OFF OF THE CO	YES SUBTRACT ONE? YES MULTIPLY BY TWO? YES DIVIDE BY TWO? YES *CHECK FOR VARIABLE EQUATE - BRANCH IF SO; ACCB WILL BE *SET TO THE VALUE OF THE BASIC VARIABLE IN THE COMMAND *STRING WHICH MUST BE NUMERIC, LESS THAN 256 *AND THE VARIABLE MUST BE FOLLOWED BY A SEMICOLON. CLEAR CARRY IF NUMERIC FC ERROR IF NON NUMERIC UNITS DIGIT = Ø DMMAND STRING
3798 3799 3800 3801 3802 3803 3804 3805 3806 3807 3808 3810 3811 3812 3813 3814 3815 3815	98BØ 27 3C 98B8 27 3C 98B6 81 3E 98B8 27 42 98B6 81 3C 98BC 27 39 98BE 81 3D 98CØ 27 3F 98CØ 80 90 AA 98C5 25 24 98C7 5F 98C8 80 30 98CA 97 D7 98CC 86 ØA 98CE 3D	L9BBE * * * STRIP * AND RE	BEQ L9BEE CMPA #'-' BEQ L9BF2 CMPA #'>' BEQ L9BFC CMPA #'<' BEQ L9BF7 CMPA #'=' BEQ L9C01 JSR L9C0A BLO L9BEB CLRB A DECIMAL ASCII VALUE OFF OF THE COUNTY TURN BINARY VALUE IN ACCB SUBA #'0' STA VD7	YES SUBTRACT ONE? YES MULTIPLY BY TWO? YES DIVIDE BY TWO? YES *CHECK FOR VARIABLE EQUATE - BRANCH IF SO; ACCB WILL BE *SET TO THE VALUE OF THE BASIC VARIABLE IN THE COMMAND *STRING WHICH MUST BE NUMERIC, LESS THAN 256 *AND THE VARIABLE MUST BE FOLLOWED BY A SEMICOLON. CLEAR CARRY IF NUMERIC FC ERROR IF NON NUMERIC UNITS DIGIT = Ø DMMAND STRING MASK OFF ASCII SAVE VALUE TEMPORARILY
3798 3798 3800 3801 3802 3803 3804 3805 3806 3807 3810 3811 3812 3813 3814 3815 3816 3817 3818	98BØ 27 3C 98B8 21 2D 98B4 27 3C 98B6 81 3E 98B8 27 42 98BA 81 3C 98BC 27 39 98BC 27 39 98BC 81 3D 98CØ 27 3F 98C2 8D 9Ø AA 98C5 25 24 98C7 5F 98C8 8Ø 3Ø 98CA 97 D7 98CC 86 ØA 98CE 3D	L9BBE * * * STRIP * AND RE	BEQ L9BEE CMPA #'-' BEQ L9BF2 CMPA #'>' BEQ L9BFC CMPA #'<' BEQ L9BF7 CMPA #'=' BEQ L9C01 JSR L9C0A BLO L9BEB CLRB A DECIMAL ASCII VALUE OFF OF THE COUNTY TURN BINARY VALUE IN ACCB SUBA #'0' STA VO7 LDA #10 MUL TSTA	YES SUBTRACT ONE? YES MULTIPLY BY TWO? YES DIVIDE BY TWO? YES *CHECK FOR VARIABLE EQUATE - BRANCH IF SO; ACCB WILL BE *SET TO THE VALUE OF THE BASIC VARIABLE IN THE COMMAND *STRING WHICH MUST BE NUMERIC, LESS THAN 256 *AND THE VARIABLE MUST BE FOLLOWED BY A SEMICOLON. CLEAR CARRY IF NUMERIC FC ERROR IF NON NUMERIC UNITS DIGIT = Ø MMAND STRING MASK OFF ASCII SAVE VALUE TEMPORARILY BASE 106 MULT BY DIGIT *
3798 3799 3800 3801 3802 3803 3804 3805 3807 3808 3810 3811 3812 3813 3814 3815 3816 3817 3818	9880 27 3C 9882 81 2D 9884 27 3C 9886 81 3E 9888 27 42 988A 81 3C 98BC 27 39 98BE 81 3D 98CØ 27 3F 98CØ 27 3F 98CØ 80 9Ø AA 98C5 25 24 98C7 5F 98C8 8Ø 3Ø 98CA 97 D7 98CC 86 ØA 98CE 3D 98CF 4D 98DØ 26 19	L9BBE * * * STRIP * AND RE	BEQ L9BEE CMPA #'-' BEQ L9BF2 CMPA #'-' BEQ L9BFC CMPA #'-' BEQ L9BF7 CMPA #'-' BEQ L9C01 JSR L9GAA BLO L9BEB CLRB CLRB A DECIMAL ASCII VALUE OFF OF THE CO TURN BINARY VALUE IN ACCB SUBA #'0' STA VD7 LDA #10 MUL TSTA BNE L9BEB	YES SUBTRACT ONE? YES MULTIPLY BY TWO? YES DIVIDE BY TWO? YES *CHECK FOR VARIABLE EQUATE - BRANCH IF SO; ACCB WILL BE *SET TO THE VALUE OF THE BASIC VARIABLE IN THE COMMAND *STRING WHICH MUST BE NUMERIC, LESS THAN 256 *AND THE VARIABLE MUST BE FOLLOWED BY A SEMICOLON. CLEAR CARRY IF NUMERIC FC ERROR IF NON NUMERIC UNITS DIGIT = Ø DMMAND STRING MASK OFF ASCII SAVE VALUE TEMPORARILY BASE 10 MULT BY DIGIT * * FC ERROR IF RESULT > 255
3798 3799 3809 3801 3802 3803 3804 3805 3806 3807 3808 3811 3812 3813 3814 3815 3816 3817 3818 3819	9880 27 3C 9882 81 2D 9884 27 3C 9886 81 3E 9888 27 42 9888 81 3C 988C 27 39 988E 81 3D 98CØ 27 3F 98CØ 27 3F 98CØ 80 90 AA 98CØ 25 24 98C7 5F 98C8 80 30 98CA 97 D7 98CC 86 ØA 98CE 3D 98CF 4D 98DØ 26 19 98DØ 26 19	L9BBE * * * STRIP * AND RE	BEQ L9BEE CMPA #'-' BEQ L9BF2 CMPA #'-' BEQ L9BFC CMPA #'-' BEQ L9BF7 CMPA #'-' BEQ L9C01 JSR L9C0A BEQ L9C01 JSR L9C0A BLO L9BEB CLRB A DECIMAL ASCII VALUE OFF OF THE CO STA VO7 LDA #10 MUL TSTA BNE L9BEB ADDB V07	YES SUBTRACT ONE? YES MULTIPLY BY TWO? YES DIVIDE BY TWO? YES *CHECK FOR VARIABLE EQUATE - BRANCH IF SO; ACCB WILL BE *SET TO THE VALUE OF THE BASIC VARIABLE IN THE COMMAND *STRING WHICH MUST BE NUMERIC, LESS THAN 256 *AND THE VARIABLE MUST BE FOLLOWED BY A SEMICOLON. CLEAR CARRY IF NUMERIC FC ERROR IF NON NUMERIC UNITS DIGIT = Ø MMASK OFF ASCII SAVE VALUE TEMPORARILY BASE 10 MULT BY DIGIT * * FC ERROR IF RESULT > 255 GET TEMPORARY VALUE
3798 3798 3800 3801 3802 3803 3804 3805 3807 3810 3811 3812 3813 3814 3815 3816 3817 3816 3817 3818 3819 3820	9880 27 3C 9882 81 2D 9884 27 3C 9886 81 3E 9888 27 42 9886 81 3C 988C 27 39 988C 27 39 988C 82 37 98C2 8D 90 AA 98C5 25 24 98C7 5F 98C8 80 30 98CA 80 AB 98CA 80 AB 98CA 80 BA 98CB BB 98CB	L9BBE * * * STRIP * AND RE	BEQ L9BEE CMPA #'-' BEQ L9BF2 CMPA #'>' BEQ L9BFC CMPA #'<' BEQ L9BF7 CMPA #'=' BEQ L9C01 JSR L9C0A BLO L9BEB CLRB A DECIMAL ASCII VALUE OFF OF THE COUNTY TURN BINARY VALUE IN ACCB SUBA #'0' STA VD7 LDA #10 MUL TSTA BNE L9BEB ADDB VD7 BLO L9BEB	YES SUBTRACT ONE? YES MULTIPLY BY TWO? YES DIVIDE BY TWO? YES *CHECK FOR VARIABLE EQUATE - BRANCH IF SO; ACCB WILL BE *SET TO THE VALUE OF THE BASIC VARIABLE IN THE COMMAND *STRING WHICH MUST BE NUMERIC, LESS THAN 256 *AND THE VARIABLE MUST BE FOLLOWED BY A SEMICOLON. CLEAR CARRY IF NUMERIC FC ERROR IF NON NUMERIC UNITS DIGIT = Ø DMMAND STRING MASK OFF ASCII SAVE VALUE TEMPORARILY BASE 10 MULT BY DIGIT * * FC ERROR IF RESULT > 255
3798 3798 3800 3801 3802 3803 3804 3805 3806 3807 3808 3811 3812 3813 3814 3815 3816 3817 3818 3819 3820 3821 3822 3823	9880 27 3C 9882 81 2D 9884 27 3C 9886 81 3E 9888 27 42 9888 81 3C 988C 27 39 988E 81 3D 98CØ 27 3F 98C2 BD 9Ø AA 98C5 25 24 98C7 5F 98C8 8Ø 3Ø 98CA 97 D7 98CC 86 ØA 98CE 3D 98CF 4D 98DØ 26 19 98DØ 26 19 98DØ 26 19 98DØ 26 19 98DØ 27 17	L9BBE * * * STRIP * AND RE	BEQ L9BEE CMPA #'-' BEQ L9BF2 CMPA #'-' BEQ L9BFC CMPA #'-' BEQ L9BFT CMPA #'-' BEQ L9C01 JSR L9C0A BEB L9C0B BEB L9C0B BEB L9BEB BED L9BEB BED L9BEB BED L9BEB BEQ L9BF1	YES SUBTRACT ONE? YES MULTIPLY BY TWO? YES DIVIDE BY TWO? YES *CHECK FOR VARIABLE EQUATE - BRANCH IF SO; ACCB WILL BE *SET TO THE VALUE OF THE BASIC VARIABLE IN THE COMMAND *STRING WHICH MUST BE NUMERIC, LESS THAN 256 *AND THE VARIABLE MUST BE FOLLOWED BY A SEMICOLON. CLEAR CARRY IF NUMERIC UNITS DIGIT = Ø DMMAND STRING MASK OFF ASCII SAVE VALUE TEMPORARILY BASE 10 MULT BY DIGIT * * FC ERROR IF RESULT > 255 GET TEMPORARY VALUE FC ERROR IF RESULT > 255 * RETURN IF NO COMMANDS LEFT
3798 3798 3800 3801 3802 3803 3804 3806 3807 3810 3811 3812 3813 3814 3815 3816 3817 3818 3819 3821 3822 3823	9880 27 3C 9882 81 2D 9884 27 3C 9886 81 3E 9888 27 42 9886 81 3C 9886 27 39 9886 27 37 9860 27 3F 9862 80 90 AA 9865 25 24 9867 5F 9868 80 30 9860 80 AB 9868 3D 9860 26 19 9800 26 19 9801 26 19 9802 08 07 9804 25 15 9806 00 08 9808 27 17 > 980A 80 98 98	L9BBE * * * STRIP * AND RE	BEQ L9BEE CMPA #'-' BEQ L9BF2 CMPA #'>' BEQ L9BFC CMPA #'-' BEQ L9BF7 CMPA #'=' BEQ L9C01 JSR L9C0A BEQ L9C0 BEQ L9C	YES SUBTRACT ONE? YES MULTIPLY BY TWO? YES DIVIDE BY TWO? YES *CHECK FOR VARIABLE EQUATE - BRANCH IF SO; ACCB WILL BE *SET TO THE VALUE OF THE BASIC VARIABLE IN THE COMMAND *STRING WHICH MUST BE NUMERIC, LESS THAN 256 *AND THE VARIABLE MUST BE FOLLOWED BY A SEMICOLON. CLEAR CARRY IF NUMERIC UNITS DIGIT = 0 MMAND STRING MASK OFF ASCII SAVE VALUE TEMPORARILY BASE 10 MULT BY DIGIT * * FC ERROR IF RESULT > 255 GET TEMPORARY VALUE FC ERROR IF RESULT > 255 * * RETURN IF NO COMMANDS LEFT GET ANOTHER COMMAND
3798 3799 3800 3801 3802 3803 3804 3805 3806 3807 3808 3810 3811 3812 3813 3814 3815 3816 3817 3818 3819 3820 3821 3822 3823 3824 3824	9880 27 3C 9882 81 2D 9884 27 3C 9886 81 3E 9888 87 42 9888 81 3C 988C 27 39 98C2 8D 90 AA 98C5 25 24 98C7 5F 98C8 80 30 98CA 97 D7 98CA 97 D7 98CC 80 0A 98CE 3D 98CF 4D 98C	L9BBE * * * STRIP * AND RE	BEQ L9BEE CMPA #'-' BEQ L9BF2 CMPA #'-' BEQ L9BFC CMPA #'-' BEQ L9BFT CMPA #'-' BEQ L9CØ1 JSR L9ØAA BLO L9BEB CLRB A DECIMAL ASCII VALUE OFF OF THE COTTURN BINARY VALUE IN ACCB SUBA #'0' STA V07 LDA #10 MUL TSTA BNE L9BEB ADDB VD7 BLO L9BEB ADDB VD7 BLO L9BEB TST V08 BEQ L9BF1 JSR L9ØAA	YES SUBTRACT ONE? YES MULTIPLY BY TWO? YES DIVIDE BY TWO? YES *CHECK FOR VARIABLE EQUATE - BRANCH IF SO; ACCB WILL BE *SET TO THE VALUE OF THE BASIC VARIABLE IN THE COMMAND *STRING WHICH MUST BE NUMERIC, LESS THAN 256 *AND THE VARIABLE MUST BE FOLLOWED BY A SEMICOLON. CLEAR CARRY IF NUMERIC UNITS DIGIT = 0 MMASK OFF ASCII SAVE VALUE TEMPORARILY BASE 10 MULT BY DIGIT * * FC ERROR IF RESULT > 255 GET TEMPORARY VALUE FC ERROR IF RESULT > 255 * * RETURN IF NO COMMANDS LEFT GET ANOTHER COMMAND CLEAR CARRY IF NUMERIC
3798 3799 3800 3801 3802 3803 3804 3805 3807 3808 3810 3811 3812 3813 3814 3815 3816 3817 3816 3817 3816 3817 3820 3820 3821 3822 3823 3824 3823	9880 27 3C 9882 81 2D 9884 27 3C 9886 81 3E 9888 27 42 9886 81 3C 9886 27 39 9886 27 37 9860 27 3F 9862 80 90 AA 9865 25 24 9867 5F 9868 80 30 9860 80 AB 9868 3D 9860 26 19 9800 26 19 9801 26 19 9802 08 07 9804 25 15 9806 00 08 9808 27 17 > 980A 80 98 98	L9BBE * * * STRIP * AND RE	BEQ L9BEE CMPA #'-' BEQ L9BF2 CMPA #'>' BEQ L9BFC CMPA #'-' BEQ L9BF7 CMPA #'=' BEQ L9C01 JSR L9C0A BEQ L9C0 BEQ L9C	YES SUBTRACT ONE? YES MULTIPLY BY TWO? YES DIVIDE BY TWO? YES *CHECK FOR VARIABLE EQUATE - BRANCH IF SO; ACCB WILL BE *SET TO THE VALUE OF THE BASIC VARIABLE IN THE COMMAND *STRING WHICH MUST BE NUMERIC, LESS THAN 256 *AND THE VARIABLE MUST BE FOLLOWED BY A SEMICOLON. CLEAR CARRY IF NUMERIC UNITS DIGIT = 0 MMAND STRING MASK OFF ASCII SAVE VALUE TEMPORARILY BASE 10 MULT BY DIGIT * * FC ERROR IF RESULT > 255 GET TEMPORARY VALUE FC ERROR IF RESULT > 255 * * RETURN IF NO COMMANDS LEFT GET ANOTHER COMMAND
3798 3798 3800 3801 3802 3803 3804 3806 3807 3810 3811 3812 3813 3814 3815 3816 3817 3818 3819 3821 3822 3823 3821 3822 3823	9880 27 3C 9882 81 2D 9884 27 3C 9886 81 3E 9888 27 42 9886 81 3C 988C 27 39 988E 81 3D 98CØ 27 3F 98CØ 27 3F 98CØ 28D 9Ø AA 98CØ 25 24 98C7 5F 98C8 8Ø 3Ø 98CA 97 D7 98CC 86 ØA 98CØ 20 98CF 4D 98DØ 26 19 98DØ 27 17 > 98DA 8D 9B 98 98DD 8D 9Ø AA 98EØ 24 E6	L9BBE * * * STRIP * AND RE L9BC8	BEQ L9BEE CMPA #'-' BEQ L9BF2 CMPA #'-' BEQ L9BFC CMPA #'-' BEQ L9BFT CMPA #'-' BEQ L9C01 JSR L90AA BLO L9BEB CLRB A DECIMAL ASCII VALUE OFF OF THE COTURN BINARY VALUE IN ACCB SUBA #'0' STA VD7 LDA #10 MUL TSTA BNE L9BEB ADDB VD7 BLO L9BEB TST VD8 BEQ L9BF1 JSR L9BF3 JSR L9BAA BCC L9BCB	YES SUBTRACT ONE? YES MULTIPLY BY TWO? YES DIVIDE BY TWO? YES *CHECK FOR VARIABLE EQUATE - BRANCH IF SO; ACCB WILL BE *SET TO THE VALUE OF THE BASIC VARIABLE IN THE COMMAND *STRING WHICH MUST BE NUMERIC, LESS THAN 256 *AND THE VARIABLE MUST BE FOLLOWED BY A SEMICOLON. CLEAR CARRY IF NUMERIC UNITS DIGIT = Ø DMMAND STRING MASK OFF ASCII SAVE VALUE TEMPORARILY BASE 10 MULT BY DIGIT * * FC ERROR IF RESULT > 255 GET TEMPORARY VALUE FC ERROR IF RESULT > 255 * * RETURN IF NO COMMANDS LEFT GET ANOTHER COMMAND CLEAR CARRY IF NUMERIC BRANCH IF MORE NUMERIC DATA
3798 3798 3798 3800 3801 3802 3803 3804 3805 3806 3807 3808 3811 3812 3813 3814 3815 3816 3817 3818 3819 3820 3821 3822 3823 3824 3825 3826 3827 3828	98BØ 27 3C 98B8 27 3C 98B6 81 3E 98B8 27 42 98B6 81 3C 98B6 27 39 98B8 81 3C 98BC 27 39 98B2 81 3D 98C0 27 3F 98C2 8D 9Ø AA 98C5 25 24 98C7 5F 98C8 8Ø 3Ø 98CA 97 D7 98CC 86 ØA 98CB 20 BD 98CF 4D 98DØ 26 19 98DØ 26 19 98DØ 26 19 98DØ 26 19 98DØ 27 17 > 98DA 8D 9B 98 98DB 8D 9Ø AA 98EØ 3D 98 98DB 8D 9Ø AA 98EØ 24 E6 98E2 ØC B8 98E4 9E D9 98E6 3Ø 1F	L9BBE * * * STRIP * AND RE L9BC8	BEQ L98EE CMPA #'-' BEQ L98F2 CMPA #'-' BEQ L98FC CMPA #'-' BEQ L98F7 CMPA #'-' BEQ L90FT CMPA #'-' JSR L90AA BLO L98EB CHRB A DECIMAL ASCII VALUE OFF OF THE CO TURN BINARY VALUE IN ACCB SUBA #'0' STA VD7 LDA #10 MUL TSTA BNE L98EB ADDB VD7 BLO L98EB TST VD8 BEQ L98F1 JSR L989B JSR L98AA BCC L98CB INC VD8 LDX VD9 LEAX \$-01,X	YES SUBTRACT ONE? YES MULTIPLY BY TWO? YES DIVIDE BY TWO? YES *CHECK FOR VARIABLE EQUATE - BRANCH IF SO; ACCB WILL BE *SET TO THE VALUE OF THE BASIC VARIABLE IN THE COMMAND *STRING WHICH MUST BE NUMERIC, LESS THAN 256 *AND THE VARIABLE MUST BE FOLLOWED BY A SEMICOLON. CLEAR CARRY IF NUMERIC UNITS DIGIT = Ø MMAND STRING MASK OFF ASCII SAVE VALUE TEMPORARILY BASE 10 MULT BY DIGIT * * FC ERROR IF RESULT > 255 GET TEMPORARY VALUE FC ERROR IF RESULT > 255 * * RETURN IF NO COMMANDS LEFT GET ANOTHER COMMAND CLEAR CARRY IF NUMERIC BRANCH IF MORE NUMERIC BRANCH IF MORE NUMERIC BRANCH IF MORE NUMERIC DATA *ADD ONE TO COMMAND STRING BACK ONE * *
3798 3798 3800 3801 3802 3803 3804 3807 3808 3810 3811 3812 3813 3814 3815 3816 3817 3818 3819 3821 3822 3823 3821 3822 3823 3824 3825 3825 3826 3827 3828 3828 3828 3828	9880 27 3C 9882 81 2D 9884 27 3C 9886 81 3E 9888 27 42 9886 81 3C 9886 27 39 9886 27 37 9887 37 9888 81 3D 988	L9BBE * * * STRIP * AND RE L9BC8	BEQ L98EE CMPA #'-' BEQ L98F2 CMPA #'>' BEQ L98F6 CMPA #'-' BEQ L98F7 CMPA #'-' BEQ L96F7 CMPA #'-' BEQ L9001 JSR L900A BLO L9BEB CLRB A DECIMAL ASCII VALUE OFF OF THE COUNTY TURN BINARY VALUE IN ACCB SUBA #'0' STA V07 LDA #10 MUL TSTA BNE L9BEB ADDB V07 BLO L9BEB STY V08 BEQ L98F1 JSR L989B JSR L980A BCC L9BCB INC V08 LDX V09 LEAX \$-01,X STX V09	YES SUBTRACT ONE? YES MULTIPLY BY TWO? YES DIVIDE BY TWO? YES *CHECK FOR VARIABLE EQUATE - BRANCH IF SO; ACCB WILL BE *SET TO THE VALUE OF THE BASIC VARIABLE IN THE COMMAND *STRING WHICH MUST BE NUMERIC, LESS THAN 256 *AND THE VARIABLE MUST BE FOLLOWED BY A SEMICOLON. CLEAR CARRY IF NUMERIC FC ERROR IF NON NUMERIC UNITS DIGIT = Ø MMASK OFF ASCII SAVE VALUE TEMPORARILY BASE 10 MULT BY DIGIT * * FC ERROR IF RESULT > 255 GET TEMPORARY VALUE FC ERROR IF RESULT > 255 * * RETURN IF NO COMMANDS LEFT GET ANOTHER COMMAND CLEAR CARRY IF NUMERIC BRANCH IF MORE NUMERIC DATA *ADD ONE TO COMMAND STRING BACK ONE
3798 3798 3809 3801 3802 3803 3804 3805 3806 3807 3808 3811 3812 3813 3814 3815 3816 3817 3818 3819 3820 3821 3822 3823 3824 3825 3826 3827 3826 3827 3829	9880 27 3C 9882 81 2D 9884 27 3C 9886 81 3E 9888 27 42 9888 81 3C 988C 27 39 98C2 27 3F 98C2 8D 90 AA 98C5 25 24 98C7 5F 98C8 80 30 98CA 97 D7 98CC 86 ØA 98CA 97 D7 98CC 86 ØA 98CF 4D 98CB 9D 90 AA 98CB 3D 98CF 4D 98DB D7 98DB D7 98DB D7 98DB D7 98DB D8 98DB D7 98DB D7 98DB D8 98DB D7 98DB D8 98DB D7 98DB D9 B9 98CB 27 17 > 98DA BD 98 98 98DB D9 90 98CB 30 98CB 20 98CB 30 98CB 20 98CB 30	L9BBE * * * STRIP * AND RE L9BC8	BEQ L98EE CMPA #'-' BEQ L9BF2 CMPA #'-' BEQ L9BFC CMPA #'-' BEQ L9BFT CMPA #'-' BEQ L9BFT CMPA #'-' BEQ L9C01 JSR L90AA BLO L9BEB CLRB A DECIMAL ASCII VALUE OFF OF THE CO TURN BINARY VALUE IN ACCB SUBA #'0' STA VD7 LDA #10 MUL TSTA BNE L9BEB ADDB V07 BLO L9BEB ADDB V07 BLO L9BEB TST V08 BEQ L9BF1 JSR L9B9A BCC L9BC8 INC V08 LOX V09 LEAX \$-01,X STX VD9 LEAX VD9 RTS	YES SUBTRACT ONE? YES MULTIPLY BY TWO? YES DIVIDE BY TWO? YES *CHECK FOR VARIABLE EQUATE - BRANCH IF SO; ACCB WILL BE *SET TO THE VALUE OF THE BASIC VARIABLE IN THE COMMAND *STRING WHICH MUST BE NUMERIC, LESS THAN 256 *AND THE VARIABLE MUST BE FOLLOWED BY A SEMICOLON. CLEAR CARRY IF NUMERIC FC ERROR IF NON NUMERIC UNITS DIGIT = Ø MMASK OFF ASCII SAVE VALUE TEMPORARILY BASE 1Ø MULT BY DIGIT * * FC ERROR IF RESULT > 255 GET TEMPORARY VALUE FC ERROR IF RESULT > 255 * * RETURN IF NO COMMANDS LEFT GET ANOTHER COMMAND CLEAR CARRY IF NUMERIC BRANCH IF MORE NUMERIC DATA *ADD ONE TO COMMAND COUNTER AND * ** ** ** ** ** ** ** ** ** ** ** ** *
3798 3798 3798 3800 3801 3802 3803 3804 3805 3806 3807 3808 3811 3812 3813 3814 3815 3816 3817 3818 3819 3820 3821 3822 3823 3824 3825 3826 3827 3828 3829 3830 3831 3830	9880 27 3C 9882 81 2D 9884 27 3C 9886 81 3E 9888 27 42 9886 81 3C 9886 27 39 9886 27 37 9887 37 9888 81 3D 988	L9BBE * * * STRIP * AND RE L9BC8	BEQ L98EE CMPA #'-' BEQ L98F2 CMPA #'>' BEQ L98F6 CMPA #'-' BEQ L98F7 CMPA #'-' BEQ L96F7 CMPA #'-' BEQ L9001 JSR L900A BLO L9BEB CLRB A DECIMAL ASCII VALUE OFF OF THE COUNTY TURN BINARY VALUE IN ACCB SUBA #'0' STA V07 LDA #10 MUL TSTA BNE L9BEB ADDB V07 BLO L9BEB STY V08 BEQ L98F1 JSR L989B JSR L980A BCC L9BCB INC V08 LDX V09 LEAX \$-01,X STX V09	YES SUBTRACT ONE? YES MULTIPLY BY TWO? YES DIVIDE BY TWO? YES *CHECK FOR VARIABLE EQUATE - BRANCH IF SO; ACCB WILL BE *SET TO THE VALUE OF THE BASIC VARIABLE IN THE COMMAND *STRING WHICH MUST BE NUMERIC, LESS THAN 256 *AND THE VARIABLE MUST BE FOLLOWED BY A SEMICOLON. CLEAR CARRY IF NUMERIC UNITS DIGIT = Ø MMAND STRING MASK OFF ASCII SAVE VALUE TEMPORARILY BASE 10 MULT BY DIGIT * * FC ERROR IF RESULT > 255 GET TEMPORARY VALUE FC ERROR IF RESULT > 255 * * RETURN IF NO COMMANDS LEFT GET ANOTHER COMMAND CLEAR CARRY IF NUMERIC BRANCH IF MORE NUMERIC BRANCH IF MORE NUMERIC BRANCH IF MORE NUMERIC DATA *ADD ONE TO COMMAND STRING BACK ONE * *
3798 3798 3809 3801 3802 3803 3804 3805 3806 3807 3808 3811 3812 3813 3814 3815 3816 3817 3818 3819 3820 3821 3822 3823 3824 3825 3826 3827 3828 3829 3829 3831 3822 3823 3824 3825 3826 3827 3828 3829 3838	9880 27 3C 9882 81 2D 9884 27 3C 9886 81 3E 9888 27 42 9888 81 3C 988C 27 39 98E8 81 3D 98C0 27 3F 98C2 8D 90 AA 98C5 25 24 98C7 5F 98C8 80 30 98CA 97 D7 98CC 86 ØA 98CE 3D 96CF 4D 98D0 26 19 98D1 26 19 98D2 B D7 98D4 25 15 98D8 D7 98D8 D8 D7 98D8 D8 D7 98D8 27 17 99BDA D9 98 98D8 27 17 99BDA D9 98 98D8 27 17 99BDA BD 98 98D8 27 17 99BBA BD 98 98BB 78 BP D9 98BC 39 98BC 39 98BC 39 98BC 39 98BC 78 9	L9BBE * * STRIP * AND RE L9BC8 L9BE2 L9BEE L9BEB	BEQ L98EE CMPA #'-' BEQ L9BF2 CMPA #'-' BEQ L9BFC CMPA #'-' BEQ L9BFC CMPA #'-' BEQ L9BFT CMPA #'-' BEQ L9C01 JSR L90AA BLO L9BEB CLRB A DECIMAL ASCII VALUE OFF OF THE CO TURN BINARY VALUE IN ACCB SUBA #'0' STA VD7 LDA #10 MUL TSTA BNE L9BEB ADDB VD7 BLO L9BEB ADDB V07 BLO L9BEB TST V08 BEQ L9BF1 JSR L989AA BCC L9BC8 INC V08 LOX V09 LEAX \$-01,X STX VD9 LEAX \$-01,X STX VD9 RTS JMP L844A INCB BEQ L9BEB	YES SUBTRACT ONE? YES MULTIPLY BY TWO? YES DIVIDE BY TWO? YES *CHECK FOR VARIABLE EQUATE - BRANCH IF SO; ACCB WILL BE *SET TO THE VALUE OF THE BASIC VARIABLE IN THE COMMAND *STRING WHICH MUST BE NUMERIC, LESS THAN 256 *AND THE VARIABLE MUST BE FOLLOWED BY A SEMICOLON. CLEAR CARRY IF NUMERIC UNITS DIGIT = Ø DMMAND STRING MASK OFF ASCII SAVE VALUE TEMPORARILY BASE 10 MULT BY DIGIT * * FC ERROR IF RESULT > 255 GET TEMPORARY VALUE FC ERROR IF RESULT > 255 * * RETURN IF NO COMMANDS LEFT GET ANOTHER COMMAND CLEAR CARRY IF NUMERIC BRANCH IF MORE NUMERIC DATA *ADD ONE TO COMMAND COUNTER AND *MOVE COMMAND STRING BACK ONE * * FC ERROR
3798 3798 3798 3800 3801 3802 3803 3804 3805 3806 3807 3808 3811 3812 3813 3814 3815 3816 3817 3818 3819 3820 3821 3822 3823 3824 29 3830 3831 3834 3832 3833 3834 3833	9880 27 3C 9882 81 2D 9884 27 3C 9886 81 3E 9888 27 42 9886 81 3C 9886 27 39 9882 81 3D 9865 27 3F 9862 8D 90 AA 9865 25 24 9867 5F 9868 80 30 986A 97 D7 986C 86 ØA 986A 97 D7 986C 86 ØA 986B 97 D7 980A D8 D9 986A 97 D7 980A D8 D7 980B 26 19 980B 27 17 > 980A BD 98 98 980B 27 17 > 980B 80 19 980B	L9BBE * * * STRIP * AND RE L9BC8 L9BE2 L9BEB L9BEE L9BEE	BEQ L98EE CMPA #'-' BEQ L98F2 CMPA #'-' BEQ L98FC CMPA #'-' BEQ L98FC CMPA #'-' BEQ L98FT CMPA #'-' BEQ L90FT CMPA #'-' BEQ L90F1 JSR L90AA BLO L9BEB A DECIMAL ASCII VALUE OFF OF THE CONTROLOGY STA VD7 LDA #10 MUL TSTA BNE L9BEB ADDB V07 BLO L98EB TST V08 BEQ L98F1 JSR L98P8 JSR L90AA BCC L98C8 INC V08 LDX VD9 LEAX \$-01, X STX VD9 RTS JMP L844A INCB BEQ L98EB RTS L98EB BEQ L98EB	YES SUBTRACT ONE? YES MULTIPLY BY TWO? YES DIVIDE BY TWO? YES **CHECK FOR VARIABLE EQUATE - BRANCH IF SO; ACCB WILL BE **SET TO THE VALUE OF THE BASIC VARIABLE IN THE COMMAND **STRING WHICH MUST BE NUMERIC, LESS THAN 256 **AND THE VARIABLE MUST BE FOLLOWED BY A SEMICOLON. CLEAR CARRY IF NUMERIC UNITS DIGIT = 0 MMAND STRING MASK OFF ASCII SAVE VALUE TEMPORARILY BASE 10 MULT BY DIGIT * * FC ERROR IF RESULT > 255 GET TEMPORARY VALUE FC ERROR IF RESULT > 255 * * RETURN IF NO COMMANDS LEFT GET ANOTHER COMMAND CLEAR CARRY IF NUMERIC BRANCH IF MORE NUMERIC DATA **ADD ONE TO COMMAND COUNTER AND **MOVE COMMAND STRING BACK ONE * * FC ERROR ADD ONE TO PARAMETER
3798 3798 3800 3801 3802 3803 3804 3807 3808 3810 3811 3812 3813 3814 3815 3816 3817 3818 3819 3821 3822 3823 3824 3825 3826 3827 3828 3828 3828 3830 3831 3831 3833 3834 3833 3834 3833 3834 3833 3834 3833 3834 3833 3834 3836	9880 27 3C 9882 81 2D 9884 27 3C 9886 81 3E 9888 27 42 9888 81 3C 9886 27 39 9882 27 37 9882 80 37 9882 80 37 9882 80 38 9883 80 38 9884 80 38 9885 25 24 9887 5F 9888 80 38 98	L9BBE * * STRIP * AND RE L9BC8 L9BE2 L9BEE L9BEB	BEQ L98EE CMPA #'-' BEQ L98F2 CMPA #'-' BEQ L98FC CMPA #'-' BEQ L98F7 CMPA #'-' BEQ L96F7 CMPA #'-' BEQ L9601 JSR L960A BLO L98EB CLRB A DECIMAL ASCII VALUE OFF OF THE CONTROL OF THE CO	YES SUBTRACT ONE? YES MULTIPLY BY TWO? YES DIVIDE BY TWO? YES **CHECK FOR VARIABLE EQUATE - BRANCH IF SO; ACCB WILL BE **SET TO THE VALUE OF THE BASIC VARIABLE IN THE COMMAND **STRING WHICH MUST BE NUMERIC, LESS THAN 256 **AND THE VARIABLE MUST BE FOLLOWED BY A SEMICOLON. CLEAR CARRY IF NUMERIC UNITS DIGIT = Ø MMAND STRING MASK OFF ASCII SAVE VALUE TEMPORARILY BASE 10 MULT BY DIGIT * * FC ERROR IF RESULT > 255 GET TEMPORARY VALUE FC ERROR IF RESULT > 255 * * RETURN IF NO COMMANDS LEFT GET ANOTHER COMMAND CLEAR CARRY IF NUMERIC BRANCH IF MORE NUMERIC DATA **ADD ONE TO COMMAND COUNTER AND **MOVE COMMAND STRING BACK ONE * * * * FC ERROR ADD ONE TO PARAMETER FC ERROR IF ADDING 1 TO 255 * *
3798 3798 3809 3801 3802 3803 3804 3805 3806 3807 3808 3811 3812 3813 3814 3815 3816 3817 3818 3817 3818 3829 3824 3825 3826 3827 3828 3829 3821 3822 3823 3824 3825 3826 3827 3828 3829 3838	9880 27 3C 9882 81 2D 9884 27 3C 9886 81 3E 9888 27 42 9886 81 3C 9886 27 39 9882 81 3D 9865 27 3F 9862 8D 90 AA 9865 25 24 9867 5F 9868 80 30 986A 97 D7 986C 86 ØA 986A 97 D7 986C 86 ØA 986B 97 D7 980A D8 D9 986A 97 D7 980A D8 D7 980B 26 19 980B 27 17 > 980A BD 98 98 980B 27 17 > 980B 80 19 980B	L9BBE * * * STRIP * AND RE L9BC8 L9BE2 L9BEB L9BEE L9BEE	BEQ L98EE CMPA #'-' BEQ L98F2 CMPA #'-' BEQ L98FC CMPA #'-' BEQ L98FC CMPA #'-' BEQ L98FT CMPA #'-' BEQ L90FT CMPA #'-' BEQ L90F1 JSR L90AA BLO L9BEB A DECIMAL ASCII VALUE OFF OF THE CONTROLOGY STA VD7 LDA #10 MUL TSTA BNE L9BEB ADDB V07 BLO L98EB TST V08 BEQ L98F1 JSR L98P8 JSR L90AA BCC L98C8 INC V08 LDX VD9 LEAX \$-01, X STX VD9 RTS JMP L844A INCB BEQ L98EB RTS L98EB BEQ L98EB	YES SUBTRACT ONE? YES MULTIPLY BY TWO? YES DIVIDE BY TWO? YES **CHECK FOR VARIABLE EQUATE - BRANCH IF SO; ACCB WILL BE **SET TO THE VALUE OF THE BASIC VARIABLE IN THE COMMAND **STRING WHICH MUST BE NUMERIC, LESS THAN 256 **AND THE VARIABLE MUST BE FOLLOWED BY A SEMICOLON. CLEAR CARRY IF NUMERIC UNITS DIGIT = 0 MMAND STRING MASK OFF ASCII SAVE VALUE TEMPORARILY BASE 10 MULT BY DIGIT * * FC ERROR IF RESULT > 255 GET TEMPORARY VALUE FC ERROR IF RESULT > 255 * * RETURN IF NO COMMANDS LEFT GET ANOTHER COMMAND CLEAR CARRY IF NUMERIC BRANCH IF MORE NUMERIC DATA **ADD ONE TO COMMAND COUNTER AND **MOVE COMMAND STRING BACK ONE * * FC ERROR ADD ONE TO PARAMETER
3798 3798 3800 3801 3802 3803 3804 3807 3808 3810 3811 3812 3813 3814 3815 3816 3817 3818 3821 3822 3823 3824 3825 3826 3827 3828 3828 3830 3831 3831 3833 3834 3833 3834 3833 3834 3833 3834 3833 3834 3833 3834 3833 3834 3833 3834 3838 3839	9880 27 3C 9882 81 2D 9884 27 3C 9886 81 3E 9888 27 42 9888 81 3C 988C 27 39 98C2 8D 98 AA 98C5 25 24 98C7 5F 98C8 80 30 98CA 97 D7 98C8 80 AA 98C6 3D 98CF 4D 98C6 80 A 98C7 5F 98C8 80 AA 98C7 5F 98C8 80 AA 98C8 3D 98C7 4D 98C8 80 AA 98C8 3D 98C7 4D 98C8 80 AA 98C8 3D 98C7 4D 98C8 80 AA 98C8 3D 98C8 80 AA 98C8 5C 98C8 80 AA 98C8 5C 98C8 80 AA 98C8 5C 98C8 80 AA 98C8 9C 98C8 80 AA 98C8 9C	L9BBE * * STRIP * AND RE L9BC8 L9BE2 L9BEE L9BEE L9BF1 L9BF2	BEQ L9BEE CMPA #'-' BEQ L9BF2 CMPA #'-' BEQ L9BFC CMPA #'-' BEQ L9BFC CMPA #'-' BEQ L9BFT CMPA #'' BEQ L9CØ1 JSR L9ØAA BLO L9BEB CLRB A DECIMAL ASCII VALUE OFF OF THE CO TURN BINARY VALUE IN ACCB SUBA #'0' STA V07 LDA #10 MUL TSTA BNE L9BEB ADDB VD7 BLO L9BEB ADDB VD7 BLO L9BEB TST V08 BEQ L9BF1 JSR L9ØAA BCC L9BC8 LDX VD9 LEAX \$-Ø1, X STX VD9 RTS JMP LB44A INCB BEQ L9BEB RTS TSTB BEQ L9BEB	YES SUBTRACT ONE? YES MULTIPLY BY TWO? YES DIVIDE BY TWO? YES **CHECK FOR VARIABLE EQUATE - BRANCH IF SO; ACCB WILL BE **SET TO THE VALUE OF THE BASIC VARIABLE IN THE COMMAND **STRING WHICH MUST BE NUMERIC, LESS THAN 256 **AND THE VARIABLE MUST BE FOLLOWED BY A SEMICOLON. CLEAR CARRY IF NUMERIC UNITS DIGIT = Ø MMAND STRING MASK OFF ASCII SAVE VALUE TEMPORARILY BASE 10 MULT BY DIGIT * * FC ERROR IF RESULT > 255 GET TEMPORARY VALUE FC ERROR IF RESULT > 255 ** **RETURN IF NO COMMANDS LEFT GET ANOTHER COMMAND CLEAR CARRY IF NUMERIC BRANCH IF MORE NUMERIC DATA **ADD ONE TO COMMAND COUNTER AND **MOVE COMMAND STRING BACK ONE * * * FC ERROR ADD ONE TO PARAMETER FC ERROR IF ADDING 1 TO 255 * * FC ERROR IF TRYING TO DECREMENT Ø SUBTRACT ONE FROM PARAMETER
3798 3798 3798 3800 3801 3802 3803 3804 3807 3808 3810 3811 3812 3813 3814 3815 3816 3817 3828 3829 3820 3821 3822 3823 3824 325 3826 3827 3828 3829 3830 3831 3834 3835 3836 3837 3838	9880 27 3C 9882 81 2D 9884 27 3C 9886 81 3E 9888 27 42 9886 81 3C 9886 27 39 9882 81 3D 9862 27 3F 9862 8D 90 AA 9865 25 24 9867 5F 9868 80 30 986A 97 D7 986C 86 0A 986C 3D 986C 80 D9 986C 3D 986C 19 986C 40 D8 986C 40 D8 986D 40 D8 986D 40 D8 986D 40 D8 986D 80 D9 986D 80	L9BBE * * * STRIP * AND RE L9BC8 L9BE2 L9BEB L9BEE L9BEE	BEQ L98EE CMPA #'-' BEQ L98F2 CMPA #'-' BEQ L98FC CMPA #'-' BEQ L98FC CMPA #'-' BEQ L98FT CMPA #'-' BEQ L90FT CMPA #'-' BEQ L90F1 JSR L90AA BLO L98EB CLRB A DECIMAL ASCII VALUE OFF OF THE CO TURN BINARY VALUE IN ACCB SUBA #'0 STA VD7 LDA #10 MUL TSTA BNE L98EB ADDB V07 BLO L98EB TST V08 BEQ L98F1 JSR L98P8 JSR L98AA BCC L98C8 INC V08 LDX VD9 LEAX \$-01, X STX VD9 RTS STY VD9 RTS RTS JMP L844A INCB BEQ L98EB BCQ L98EB	YES SUBTRACT ONE? YES MULTIPLY BY TWO? YES DIVIDE BY TWO? YES *CHECK FOR VARIABLE EQUATE - BRANCH IF SO; ACCB WILL BE *SET TO THE VALUE OF THE BASIC VARIABLE IN THE COMMAND *STRING WHICH MUST BE NUMERIC, LESS THAN 256 *AND THE VARIABLE MUST BE FOLLOWED BY A SEMICOLON. CLEAR CARRY IF NUMERIC UNITS DIGIT = Ø MMASK OFF ASCII SAVE VALUE TEMPORARILY BASE 10 MULT BY DIGIT * * FC ERROR IF RESULT > 255 GET TEMPORARY VALUE FC ERROR IF RESULT > 255 * * RETURN IF NO COMMANDS LEFT GET ANOTHER COMMAND CLEAR CARRY IF NUMERIC BRANCH IF MORE NUMERIC DATA *ADD ONE TO COMMAND COUNTER AND *MOVE COMMAND STRING BACK ONE * * * * * FC ERROR ADD ONE TO PARAMETER FC ERROR IF TRYING TO DECREMENT Ø

	9BF8 27 F	1			L9BEB	* FC ERROR IF DIVIDING BY ZERO
	9BFA 54 9BFB 39			LSI		DIVIDE BY TWO
	9BFC 5D		LS	RT: BBFC TS		*
3845	9BFD 2B E	:C			L9BEB	* FC ERROR IF RESULT WOULD BE > 255
	9BFF 58			AS		MULTIPLY BY TWO
	9CØØ 39 9CØ1 34 6	ia	10	RT: 9001 PSI	S HS U,Y	SAVE U, Y REGISTERS
	9CØ3 8D 1				R L9C1B	INTERPRET COMMAND STRING AS IF IT WERE A BASIC VARIABLE
	9CØ5 BD B				R LB7ØE	CONVERT FPAØ TO AN INTEGER VALUE IN ACCB
	9CØ8 35 E 9CØA BD 9		1.0		.S Y,U,PC R L9C1B	RESTORE U,Y REGISTERS AND RETURN EVALUATE AN EXPRESSION IN THE COMMAND STRING
	9CØD C6 Ø		L		3 #2	=
	9CØF BD A				R LAC33	=ROOM FOR 4 BYTES ON STACK?
	9C12 D6 D				3 VD8 (VD9	* GET THE CURRENT COMMAND LENGTH AND POINTER AND * SAVE THEM ON THE STACK
	9016 34 1				IS X,B	* SAVE THEM ON THE STACK
3858	9C18 7E 9				P L9A32	GO INTERPRET AND PROCESS THE NEW PLAY SUB COMMAND
3859 386Ø			*	INTEDDDET	THE PRESENT COMMAND STRING AS I	E IT WEDE A RASIC VADIABLE
	9C1B 9E D	19			(VD9	GET COMMAND POINTER
	9C1D 34 1				IS X	SAVE IT
	9C1F BD 9 9C22 BD B				R L9B98 R LB3A2	GET A COMMAND CHARACTER SET CARRY IF NOT ALPHA
	9025 25 0				L9BEB	FC ERROR IF NOT ALPHA - ILLEGAL VARIABLE NAME
	9C27 BD 9		LS		R L9B98	GET A COMMAND CHARACTER
	9C2A 81 3				PA #';' E L9C27	CHECK FOR SEMICOLON - COMMAND SEPARATOR BRANCH UNTIL FOUND
	9C2E 35 1				_S X	GET SAVED COMMAND POINTER
	9C3Ø DE A				J CHARAD	GET BASIC S INPUT POINTER
	9C32 34 4 9C34 9F A				HS U (CHARAD	SAVE IT PUT PLAY COMMAND POINTER IN PLACE OF BASIC S INPUT POINTER
	9C36 BD B				R LB284	EVALUATE AN ALPHA EXPRESSION P GET NEW STRING DESCRIPTOR
3874	9039 35 1	LØ			_S X	* RESTORE BASIC S INPUT POINTER
	9C3B 9F A 9C3D 39	۱6		ST: RT:	(CHARAD	*
3877	9630 39			KI.	•	
3878			*	MORE OF E	XTENDED BASIC S IRQ ROUTINE	
3879 388Ø	9C3E 4F		10	OC3E CL	84	CLEAR ACCA
	9C3F 1F 8	3B			R A,DP	SET THE DIRECT PAGE TO ZERO
	9C41 DC E) PLYTMR	GET THE PLAY TIMER
	9C43 10 2 9C47 93 D				EQ LA9BB BD VD5	BRANCH TO COLOR BASIC S IRQ ROUTINE IF ZERO SUBTRACT OUT PLAY INTERVAL
	9C49 DD E) PLYTMR	SAVE THE NEW TIMER VALUE
	9C4B 22 Ø				L9C5A	BRANCH IF PLAY COMMAND NOT DONE
	9C4D ØF E				R PLYTMR R PLYTMR+1	RESET MSB OF PLAY TIMER IF DONE RESET LSB OF PLAY TIMER
	9051 35 0				S A	GET THE CONDITION CODE REG
	9C53 10 E	E 67			\$ \$07,S	*LOAD THE STACK POINTER WITH THE CONTENTS OF THE U REGISTER
3891 3892	9056 84 7	/ E	*	ANI	OA #\$7F	*WHICH WAS STACKED WHEN THE INTERRUPT WAS HONORED. CLEAR E FLAG - MAKE COMPUTER THINK THIS WAS AN FIRQ
	9058 34 0				IS A	SAVE CONDITION CODE
3894						THE DIT HALL NOW NOT DETHING TO HIS DE AT HAD
3895 3896			*			THE RTI WILL NOW NOT RETURN TO WHERE IT WAS INTERRUPTED FROM - IT WILL RETURN TO THE MAIN PLAY
3897			*			COMMAND INTERPRETATION LOOP.
3898						
3899) CEA DT		DETUDN
	9C5A 3B		LS	OC5A RT		RETURN
3900 3901	9C5A 3B				: NUMERICAL NOTE VALUES FOR LETTER	
3900 3901 3902	9C5B ØA Ø	IC Ø1 Ø3	*	TABLE OF		
3900 3901		IC Ø1 Ø3	*	TABLE OF	NUMERICAL NOTE VALUES FOR LETTER	RNOTES
3900 3901 3902 3903 3904 3905	9C5B ØA Ø 9C61 Ø8		* Ø5 Ø6 L9 *	TABLE OF 9C5B FCI	NUMERICAL NOTE VALUES FOR LETTER 10,12,1,3,5,6,8 DELAYS FOR OCTAVE 1	RNOTES
3900 3901 3902 3903 3904 3905 3906	9C5B ØA Ø 9C61 Ø8	A8 Ø1 9Ø	* Ø5 Ø6 L9 * Ø1 7A L9	TABLE OF PC5B FCI TABLE OF PC62 FDI	NUMERICAL NOTE VALUES FOR LETTER 3 10,12,1,3,5,6,8 DELAYS FOR OCTAVE 1 3 \$01A8,\$0190,\$017A	RNOTES
3900 3901 3902 3903 3904 3905	9C5B ØA Ø 9C61 Ø8	A8 Ø1 9Ø 54 Ø1 5Ø	# Ø5 Ø6 L9 * * Ø1 7A L9 Ø1 3D	TABLE OF 9C5B FCI	NUMERICAL NOTE VALUES FOR LETTER 3 10,12,1,3,5,6,8 DELAYS FOR OCTAVE 1 3 \$01A8,\$0190,\$017A 3 \$0164,\$0150,\$013D	NOTES NOTES A,B,C,D,E,F,G
3900 3901 3902 3903 3904 3905 3906 3907 3908 3909	9C5B ØA Ø 9C61 Ø8 9C62 Ø1 A 9C68 Ø1 6	A8 Ø1 9Ø 54 Ø1 5Ø 2B Ø1 1A	# Ø5 Ø6 L9 * # Ø1 7A L9 Ø1 3D Ø1 ØA	TABLE OF PC5B FCI TABLE OF PC62 FDI FDI	NUMERICAL NOTE VALUES FOR LETTER 3 10,12,1,3,5,6,8 DELAYS FOR OCTAVE 1 3 \$01A8,\$0190,\$017A 3 \$0164,\$0150,\$013D 3 \$0128,\$011A,\$010A	NOTES NOTES A,B,C,D,E,F,G
3900 3901 3902 3903 3904 3905 3906 3907 3908 3909 3910	9C5B ØA Ø 9C61 Ø8 9C62 Ø1 A 9C68 Ø1 6 9C6E Ø1 2	A8 Ø1 9Ø 54 Ø1 5Ø 2B Ø1 1A	# Ø5 Ø6 L9 * * * * * * * * * * * * * * * * * *	TABLE OF PC5B FCI TABLE OF PC62 FDI FDI FDI	NUMERICAL NOTE VALUES FOR LETTER 3 10,12,1,3,5,6,8 DELAYS FOR OCTAVE 1 3 \$01A8,\$0190,\$017A 3 \$0164,\$0150,\$013D 3 \$012B,\$011A,\$010A 3 \$00FB,\$00ED,\$00DF	NOTES NOTES A,B,C,D,E,F,G
3900 3901 3902 3903 3904 3905 3906 3907 3908 3909	9C5B ØA Ø 9C61 Ø8 9C62 Ø1 A 9C68 Ø1 6 9C6E Ø1 2	A8 Ø1 9Ø 54 Ø1 5Ø 2B Ø1 1A FB ØØ ED	# Ø5 Ø6 L9 * * * * * * * * * * * * * * * * * *	TABLE OF TABLE OF TABLE OF FDI FDI TABLE OF	NUMERICAL NOTE VALUES FOR LETTER 3 10,12,1,3,5,6,8 DELAYS FOR OCTAVE 1 3 \$01A8,\$0190,\$017A 3 \$0164,\$0150,\$013D 3 \$012B,\$011A,\$010A 3 \$00FB,\$00ED,\$00DF DELAYS FOR OCTAVE 2	NOTES NOTES A,B,C,D,E,F,G
3900 3901 3902 3903 3904 3905 3906 3907 3908 3910 3911 3912 3913	9C5B ØA Ø 9C61 Ø8 9 9C62 Ø1 A 9C68 Ø1 6 9C6E Ø1 2 9C74 ØØ F	A8 Ø1 9Ø 54 Ø1 5Ø 2B Ø1 1A FB ØØ ED O3 ØØ C7 81 ØØ A6	# # # # # # # # # # # # # # # # # # #	TABLE OF POINT FOR TABLE OF TABLE OF POINT FOR TABLE OF POINT FOR TABLE OF POINT FOR TABLE OF TABLE O	NUMERICAL NOTE VALUES FOR LETTER 3 10,12,1,3,5,6,8 DELAYS FOR OCTAVE 1 3 \$01A8,\$0190,\$017A 3 \$0164,\$0150,\$013D 3 \$012B,\$011A,\$010A 3 \$00FB,\$00ED,\$00DF DELAYS FOR OCTAVE 2 8 \$00D3,\$00C7,\$00BB 3 \$000B1,\$00A6,\$009D	R NOTES NOTES A,B,C,D,E,F,G DELAYS FOR OCTAVE 1
3900 3901 3902 3903 3904 3905 3906 3907 3908 3909 3910 3911 3912 3913 3914	9C5B ØA Ø 9C61 Ø8 9C62 Ø1 A 9C68 Ø1 6 9C74 ØØ F 9C74 ØØ F	A8 Ø1 9Ø 54 Ø1 5Ø 2B Ø1 1A FB ØØ ED 03 ØØ C7 81 ØØ A6 94 ØØ 8B	* * * * * * * * * * * * * * * * * * *	TABLE OF FOIL FOIL FOIL FOIL FOIL FOIL FOIL F	NUMERICAL NOTE VALUES FOR LETTER 3 10,12,1,3,5,6,8 DELAYS FOR OCTAVE 1 3 \$01A8,\$0190,\$017A 3 \$0164,\$0150,\$013D 3 \$0128,\$011A,\$010A 3 \$00FB,\$00ED,\$00DF DELAYS FOR OCTAVE 2 3 \$0003,\$00C7,\$008B 3 \$0081,\$00A6,\$009D 3 \$0094,\$008B,\$0083	R NOTES NOTES A,B,C,D,E,F,G DELAYS FOR OCTAVE 1
3900 3901 3902 3903 3904 3905 3906 3907 3908 3909 3910 3911 3912 3913 3914 3915	9C5B ØA Ø 9C61 Ø8 9 9C62 Ø1 A 9C68 Ø1 6 9C6E Ø1 2 9C74 ØØ F	A8 Ø1 9Ø 54 Ø1 5Ø 2B Ø1 1A FB ØØ ED 03 ØØ C7 81 ØØ A6 94 ØØ 8B	* * * * * * * * * * * * * * * * * * *	TABLE OF POINT FOR TABLE OF TABLE OF POINT FOR TABLE OF POINT FOR TABLE OF POINT FOR TABLE OF TABLE O	NUMERICAL NOTE VALUES FOR LETTER 3 10,12,1,3,5,6,8 DELAYS FOR OCTAVE 1 3 \$01A8,\$0190,\$017A 3 \$0164,\$0150,\$013D 3 \$0128,\$011A,\$010A 3 \$00FB,\$00ED,\$00DF DELAYS FOR OCTAVE 2 3 \$0003,\$00C7,\$008B 3 \$0081,\$00A6,\$009D 3 \$0094,\$008B,\$0083	R NOTES NOTES A,B,C,D,E,F,G DELAYS FOR OCTAVE 1
3900 3901 3902 3903 3904 3906 3907 3908 3910 3911 3911 3913 3914 3913 3914 3915	9C5B 8A 8 9C61 88 9C62 81 A 9C68 81 6 9C6E 01 2 9C74 80 F 9C7A 80 C 9C88 80 9 9C86 80 9 9C86 80 7	A8 Ø1 9Ø 54 Ø1 5Ø 2B Ø1 1A -B ØØ ED 03 ØØ C7 81 ØØ A6 94 ØØ 8B 7C ØØ 75	% % % % % % % % % % % % % % % % % % %	TABLE OF FOR TABLE OF FOR TABLE OF FOR FOR FOR FOR FOR FOR FOR FOR FOR	NUMERICAL NOTE VALUES FOR LETTER 3 10,12,1,3,5,6,8 DELAYS FOR OCTAVE 1 3 \$01A8,\$0190,\$017A 3 \$0164,\$0150,\$013D 3 \$012B,\$011A,\$010A 3 \$00FB,\$00ED,\$00DF DELAYS FOR OCTAVE 2 3 \$0003,\$00C7,\$00BB 3 \$000B,\$00G,\$00DB DELAYS FOR OCTAVE 2 3 \$0003,\$00C7,\$00BB 3 \$0007,\$00FB,\$00B3 3 \$007C,\$0075,\$00BB DELAYS FOR OCTAVES 3,4,5	R NOTES NOTES A,B,C,D,E,F,G DELAYS FOR OCTAVE 1 DELAYS FOR OCTAVE 2
3900 3901 3902 3903 3904 3906 3906 3908 3909 3910 3912 3913 3912 3913 3915 3916 3917 3918	9C5B 0A 0 9C61 08 0 9C62 01 A 9C68 01 6 9C6E 01 2 9C74 00 F 9C74 00 F 9C80 00 9 9C86 00 9 9C86 00 9	A8 Ø1 9Ø 64 Ø1 5Ø 2B Ø1 1A 7B ØØ ED 03 ØØ C7 81 ØØ A6 64 ØØ 8B 7C ØØ 75	# * * * * * * * * * * * * * * * * * * *	TABLE OF FOR TABLE OF TABLE OF TABLE OF TABLE OF TABLE OF FOR TABLE OF	NUMERICAL NOTE VALUES FOR LETTER 3 10,12,1,3,5,6,8 DELAYS FOR OCTAVE 1 3 \$0164,\$0190,\$017A 3 \$0164,\$0150,\$013D 3 \$0128,\$011A,\$010A 3 \$0078,\$000F DELAYS FOR OCTAVE 2 3 \$0003,\$0007,\$000B 3 \$0001,\$0007,\$000B 3 \$0001,\$0006,\$000D DELAYS FOR OCTAVE 3,4,5 5 \$0076,\$0075,\$006E DELAYS FOR OCTAVES 3,4,5 5 \$46,\$90,993,\$88,\$83,\$78	R NOTES NOTES A,B,C,D,E,F,G DELAYS FOR OCTAVE 1
3900 3901 3902 3903 3904 3906 3907 3908 3910 3911 3911 3913 3914 3913 3914 3915	9C5B 8A 8 9C61 88 9C62 81 A 9C68 81 6 9C6E 01 2 9C74 80 F 9C7A 80 C 9C88 80 9 9C86 80 9 9C86 80 7	A8 Ø1 90 64 Ø1 5Ø 88 Ø1 1A FB ØØ ED 03 ØØ C7 81 ØØ A6 84 ØØ 88 7C ØØ 75 9C 93 8B 5D 67 61	* * * * * * * * * * * * * * * * * * *	TABLE OF FOIL TA	NUMERICAL NOTE VALUES FOR LETTER 3 10,12,1,3,5,6,8 DELAYS FOR OCTAVE 1 3 \$0164,\$0190,\$017A 3 \$0164,\$0150,\$013D 3 \$012B,\$011A,\$010A 3 \$00FB,\$00ED,\$00DF DELAYS FOR OCTAVE 2 3 \$0003,\$00C7,\$00BB 3 \$00091,\$000B,\$000B 3 \$0094,\$008B,\$0083 3 \$007C,\$0075,\$006E DELAYS FOR OCTAVE 3,4,5 5 \$066,\$0990	R NOTES NOTES A,B,C,D,E,F,G DELAYS FOR OCTAVE 1 DELAYS FOR OCTAVE 2
3900 3901 3902 3903 3904 3906 3906 3908 3909 3910 3912 3913 3915 3916 3916 3917 3918 3919 3919 3919 3919 3919 3919 3919	9C5B 0A 0 9C61 08 01 9C62 01 A 09 05 01 00 05 01 00 05 01 00 05 01 00 05 00 05 00 05 00 05 00 05 00 05 00 05 00 05 00 05 00 05 05	A8 Ø1 90 54 Ø1 5Ø 2B Ø1 1A FB ØØ ED 03 ØØ C7 81 ØØ A6 64 ØØ 8B 7C ØØ 75 00 93 8B 60 67 61 61 67 61 61 67 63 61 67 67 63 61 67 67 67 67 67 67 67 67 67 67 67 67 67	* * * * * * * * * * * * * * * * * * *	TABLE OF FOR FOR FOR FOR FOR FOR FOR FOR FOR	NUMERICAL NOTE VALUES FOR LETTER 3 10,12,1,3,5,6,8 DELAYS FOR OCTAVE 1 3 \$0164,\$0190,\$017A 3 \$0164,\$0150,\$013D 3 \$0128,\$011A,\$010A 3 \$0078,\$000F DELAYS FOR OCTAVE 2 3 \$0003,\$0007,\$000B 3 \$0001,\$0007,\$000B 3 \$0001,\$0007,\$000B 3 \$0007,\$000B,\$0003 3 \$0077,\$000B,\$0003 3 \$0077,\$000B,\$0003 3 \$0077,\$000B,\$0003 3 \$0077,\$0078,\$000B,\$0003 3 \$0077,\$000B,\$	R NOTES NOTES A,B,C,D,E,F,G DELAYS FOR OCTAVE 1 DELAYS FOR OCTAVE 2
3900 3901 3902 3903 3905 3906 3906 3908 3909 3911 3912 3914 3915 3916 3917 3916 3917 3918 3919 3919 3916 3917 3918 3918 3918 3918 3918 3918 3918 3918	9C58 0A 0 9C61 08 9C62 01 A 9C68 01 C 9C68 01 C 9C74 00 F 9C74 00 F 9C80 00 B 9C80 00 7 9C92 A6 9 9C92 A6 9 9C95 51 4 9C94 37 3 9C44 37 3	A8 Ø1 9Ø 44 Ø1 5Ø 85 Ø1 16 86 Ø1 16 87 Ø1 5Ø 88 Ø1 16 88 Ø1 16 88 Ø1 16 88 Ø2 75 89 Ø3 88 80 67 61 60 47 43 81 31 28 82 3 21 1F	* * * * * * * * * * * * * * * * * * *	TABLE OF FDI FDI FDI FDI FDI FDI FDI FDI FDI FD	NUMERICAL NOTE VALUES FOR LETTER 3 10,12,1,3,5,6,8 DELAYS FOR OCTAVE 1 3 \$01AB,\$0190,\$017A 3 \$0164,\$0150,\$013D 3 \$012B,\$011A,\$010A 3 \$00FB,\$00ED,\$00DF DELAYS FOR OCTAVE 2 3 \$0003,\$00C7,\$00BB 3 \$0001,\$00AC7,\$00BB 3 \$00	R NOTES NOTES A,B,C,D,E,F,G DELAYS FOR OCTAVE 1 DELAYS FOR OCTAVE 2
3900 3901 3902 3903 3904 3906 3906 3908 3909 3910 3912 3913 3915 3916 3916 3917 3918 3919 3919 3919 3919 3919 3919 3919	9C5B 0A 0 9C61 08 01 9C62 01 A 09 05 01 00 05 01 00 05 01 00 05 01 00 05 00 05 00 05 00 05 00 05 00 05 00 05 00 05 00 05 00 05 05	A8 Ø1 9Ø 44 Ø1 5Ø 85 Ø1 16 86 Ø1 16 87 Ø1 5Ø 88 Ø1 16 88 Ø1 16 88 Ø1 16 88 Ø2 75 89 Ø3 88 80 67 61 60 47 43 81 31 28 82 3 21 1F	* * * * * * * * * * * * * * * * * * *	TABLE OF FDI FDI FDI FDI FDI FDI FDI FDI FDI FD	NUMERICAL NOTE VALUES FOR LETTER 3 10,12,1,3,5,6,8 DELAYS FOR OCTAVE 1 3 \$0164,\$0190,\$017A 3 \$0164,\$0150,\$013D 3 \$0128,\$011A,\$010A 3 \$0078,\$000F DELAYS FOR OCTAVE 2 3 \$0003,\$0007,\$000B 3 \$0001,\$0007,\$000B 3 \$0001,\$0007,\$000B 3 \$0007,\$000B,\$0003 3 \$0077,\$000B,\$0003 3 \$0077,\$000B,\$0003 3 \$0077,\$000B,\$0003 3 \$0077,\$0078,\$000B,\$0003 3 \$0077,\$000B,\$	R NOTES NOTES A,B,C,D,E,F,G DELAYS FOR OCTAVE 1 DELAYS FOR OCTAVE 2
3900 3901 3902 3903 3905 3906 3906 3909 3911 3912 3913 3914 3915 3916 3917 3918 3919 3920 3921 3922 3923 3924	9C58 0A 2 9C61 08 9C62 01 A 9C68 01 2 9C74 00 E 9C80 00 9 9C8C 00 7 9C92 A6 9C95 51 4 9CA4 37 3 9CA4 26 9CA6 19 1	A8 Ø1 9Ø 64 Ø1 5Ø 62 Ø1 1A 63 ØØ ED 63 ØØ C7 61 ØØ 88 62 ØØ 75 62 Ø3 88 65 67 61 64 44 43 64 31 2E 64 32 21 1F 68 16 14	* 05 06 LS * 01 7A LS 01 3D 01 0A 00 DF * 00 8B LS 00 9D 00 83 00 6E * 83 7B LS 3F 3B LS 3F 3B LS 2B 2B 2B 13 12	TABLE OF FDI FDI FDI FDI FDI FDI FDI FDI FDI FD	NUMERICAL NOTE VALUES FOR LETTER 3 10,12,1,3,5,6,8 DELAYS FOR OCTAVE 1 3 \$01A8,\$0190,\$017A 3 \$0164,\$0150,\$013D 3 \$0128,\$011A,\$010A 3 \$00FB,\$00ED,\$00DF DELAYS FOR OCTAVE 2 3 \$0003,\$00C7,\$00BB 3 \$0001,\$00ADF DELAYS FOR OCTAVE 2 3 \$0003,\$00C7,\$00BB 3 \$00075,\$00BB 3 \$00075,\$00BB 3 \$00075,\$00BB 3 \$0094,\$00BB,\$00BB 3 \$0076,\$0075,\$00BE DELAYS FOR OCTAVES 3,4,5 DELAYS FOR OCTAVES 3,4,5 3 \$1 \$1,\$00,\$07,\$01,\$18,\$16,\$16,\$16,\$16,\$11,\$18,\$18,\$18,\$18,\$18,\$18,\$18,\$18,\$18	NOTES NOTES A,B,C,D,E,F,G DELAYS FOR OCTAVE 1 DELAYS FOR OCTAVE 2 DELAYS FOR OCTAVES 3,4,5
3900 3901 3902 3903 3904 3906 3906 3908 3910 3911 3912 3913 3914 3915 3916 3917 3918 3917 3918 3919 3920 3921 3922 3923 3924 3924 3925	9C5B ØA Ø 9C61 Ø8 9 9C62 Ø1 A 9C68 Ø1 6 9C74 ØØ F 9C74 ØØ F 9C80 Ø0 7 9C80 Ø0 7 9C92 A6 9 9C92 A6 9 9C98 74 6 9C92 A6 9 9C98 74 6 9C92 A6 9 9C98 74 6 9C98 74 6	A8 Ø1 9Ø 54 Ø1 5Ø 88 Ø1 1A 88 Ø2 Ø2 Ø3 80 Ø2 Ø3 80 Ø3 Ø3 Ø3 Ø4 84 Ø3 Ø3 Ø4 85 Ø3 Ø4 86 Ø3 Ø4 86 Ø4 87 Ø4 88	* 05 06 LS * 01 7A LS 01 3D 01 0A 00 DF * 00 8B LS 00 9D 00 83 00 6E * 83 7B LS 3F 3B LS 3F 3B LS 2B 2B 2B 13 12	TABLE OF FOR FOR FOR FOR FOR FOR FOR FOR FOR	NUMERICAL NOTE VALUES FOR LETTER 3 10,12,1,3,5,6,8 DELAYS FOR OCTAVE 1 3 \$91A8,\$9199,\$917A 3 \$9164,\$9159,\$913D 3 \$9128,\$911A,\$919A 3 \$90FB,\$00ED,\$00DF DELAYS FOR OCTAVE 2 3 \$0003,\$90C7,\$90BB 3 \$000B1,\$00A6,\$009D 3 \$907C,\$0075,\$00BB 3 \$007C,\$0075,\$00BB 3 \$007C,\$00BB,	R NOTES NOTES A,B,C,D,E,F,G DELAYS FOR OCTAVE 1 DELAYS FOR OCTAVE 2 DELAYS FOR OCTAVES 3,4,5 * X=0. ACCB=1; END OF DRAW COMMAND LINE VALUES -
3900 3901 3902 3903 3905 3906 3906 3909 3911 3912 3913 3914 3915 3916 3917 3918 3919 3920 3921 3922 3923 3924	9C58 0A 2 9C61 08 9C62 01 A 9C68 01 2 9C74 00 E 9C80 00 9 9C8C 00 7 9C92 A6 9C95 51 4 9CA4 37 3 9CA4 26 9CA6 19 1	A8 Ø1 9Ø 64 Ø1 5Ø 88 Ø1 10 88 Ø1 60 89 Ø1 ED 30 ØØ C7 81 ØØ A6 84 Ø0 88 87 ØØ 75 80 93 88 80 67 61 64 47 43 84 31 2E 83 21 1F 88 16 14	* 05 06 LS * 01 7A LS 01 3D 01 0A 00 DF * 00 8B LS 00 9D 00 83 00 6E * 83 7B LS 3F 3B LS 3F 3B LS 2B 2B 2B 13 12	TABLE OF FOR FOR FOR FOR FOR FOR FOR FOR FOR	NUMERICAL NOTE VALUES FOR LETTER 3 10,12,1,3,5,6,8 DELAYS FOR OCTAVE 1 3 \$01A8,\$0190,\$017A 3 \$0164,\$0150,\$013D 3 \$0128,\$011A,\$010A 3 \$00FB,\$00ED,\$00DF DELAYS FOR OCTAVE 2 3 \$0003,\$00C7,\$00BB 3 \$0001,\$00ADF DELAYS FOR OCTAVE 2 3 \$0003,\$00C7,\$00BB 3 \$00075,\$00BB 3 \$00075,\$00BB 3 \$00075,\$00BB 3 \$0094,\$00BB,\$00BB 3 \$0076,\$0075,\$00BE DELAYS FOR OCTAVES 3,4,5 DELAYS FOR OCTAVES 3,4,5 3 \$1 \$1,\$00,\$07,\$01,\$18,\$16,\$16,\$16,\$16,\$11,\$18,\$18,\$18,\$18,\$18,\$18,\$18,\$18,\$18	NOTES NOTES A,B,C,D,E,F,G DELAYS FOR OCTAVE 1 DELAYS FOR OCTAVE 2 DELAYS FOR OCTAVES 3,4,5
3900 3901 3902 3903 3904 3906 3906 3908 3910 3911 3912 3913 3914 3915 3916 3917 3918 3917 3918 3919 3920 3921 3922 3923 3924 3925 3926 3927 3927 3927 3928	9C5B ØA Ø 9C61 Ø8 9 9C62 Ø1 A 9C68 Ø1 6 9C74 ØØ F 9C74 ØØ F 9C80 Ø0 7 9C80 Ø0 7 9C92 A6 9 9C92 A6 9 9C83 A7 9 9C84 A7 9 9C86 A	A8 Ø1 9Ø 54 Ø1 58 88 Ø1 98 88 Ø1 98 88 Ø1 98 80 ED 33 ØØ C7 81 Ø0 A6 84 Ø0 88 87 Ø0 75 80 93 88 80 67 61 61 C4 47 43 84 31 2E 23 21 1F 88 16 14 88 84	* 05 06 LS * 01 7A LS 01 3D 01 0A 00 DF * 00 8B LS 00 9D 00 83 00 6E * 83 7B LS 3F 3B LS 3F 3B LS 2B 2B 2B 13 12	TABLE OF FOR FOR FOR FOR FOR FOR FOR FOR FOR	NUMERICAL NOTE VALUES FOR LETTER 3 10,12,1,3,5,6,8 DELAYS FOR OCTAVE 1 3 \$91A8,\$9199,\$917A 3 \$9164,\$9159,\$913D 3 \$9128,\$911A,\$919A 3 \$90FB,\$00ED,\$00DF DELAYS FOR OCTAVE 2 3 \$0003,\$90C7,\$90BB 3 \$000B1,\$00A6,\$009D 3 \$907C,\$0075,\$00BB 3 \$007C,\$0075,\$00BB 3 \$007C,\$0075,\$00BB 3 \$007C,\$0075,\$00BB 3 \$007C,\$0075,\$00BB 3 \$007C,\$0075,\$00BB 3 \$1,450,\$00BB	* X=Ø. ACCB=1; END OF DRAW COMMAND LINE VALUES - * WHEN THESE VALUES ARE PULLED OFF THE * STACK, THE DRAW COMMAND WILL END SET PSET/PRESET FLAG TO PSET
3900 3901 3902 3903 3904 3906 3907 3908 3909 3910 3912 3913 3914 3915 3916 3917 3918 3919 3920 3921 3922 3923 3924 3925 3927 3928 3928 3928 3928 3928 3928 3928 3928	9C5B 0A 0 9C61 08 01 9C62 01 A 09 05 9C68 01 6 9C74 00 F 9C74 00 F 9C78 00 7 9C86 00 7 9C92 A6 9 9C86 00 7 9C92 A6 9 9C95 51 4 9C44 37 3 9C44 37 3 9C46 00 5 9C56 01 7 9C66 01 7 9C66 01 7 9C68 01 7 9C78	A8 Ø1 9Ø 54 Ø1 5Ø 28 Ø1 1A 66 ØØ ED 33 ØØ C7 51 ØØ A6 64 Ø0 88 7C ØØ 75 8C 93 8B 6C 97 88 6C	* 05 06 LS * 01 7A LS 01 3D 01 0A 00 DF * 00 8B LS 00 9D 00 83 00 6E * 83 7B LS 3F 3B LS 3F 3B LS 2B 2B 2B 13 12	TABLE OF FOR FOR FOR FOR FOR FOR FOR FOR FOR	NUMERICAL NOTE VALUES FOR LETTER 3 10,12,1,3,5,6,8 DELAYS FOR OCTAVE 1 3 \$01A8,\$0190,\$017A 3 \$0164,\$0150,\$013D 3 \$0128,\$011A,\$010A 3 \$00FB,\$00ED,\$00DF DELAYS FOR OCTAVE 2 3 \$0003,\$00C7,\$00BB 3 \$00094,\$000B,\$000D DELAYS FOR OCTAVE 2 3 \$0003,\$00C7,\$00BB 3 \$0007,\$000BB,\$0003 3 \$007C,\$0075,\$006E DELAYS FOR OCTAVES 3,4,5 5 \$046,\$9C,\$93,\$8B,\$83,\$7B 3 \$074,\$60,\$67,\$61,\$58,\$56 3 \$37,\$34,\$31,\$2E,\$2B,\$2B 3 \$26,\$23,\$21,\$1F,\$1D,\$1B 3 \$19,\$18,\$16,\$14,\$13,\$12 (C ZERO 3 #\$01 SX ,B 3 SETFLG (VD5	* NOTES NOTES A,B,C,D,E,F,G DELAYS FOR OCTAVE 1 DELAYS FOR OCTAVE 2 DELAYS FOR OCTAVES 3,4,5 * X=Ø. ACCB=1; END OF DRAW COMMAND LINE VALUES - * WHEN THESE VALUES ARE PULLED OFF THE * STACK, THE DRAW COMMAND WILL END SET PSET/PRESET FLAG TO PSET CLEAR UPDATE FLAG AND DRAW FLAG
3900 3901 3902 3903 3904 3906 3906 3908 3910 3911 3912 3913 3914 3915 3916 3917 3918 3917 3918 3919 3920 3921 3922 3923 3924 3925 3926 3927 3927 3927 3928	9C5B ØA Ø 9C61 Ø8 9 9C62 Ø1 A 9C68 Ø1 6 9C74 ØØ F 9C74 ØØ F 9C80 Ø0 7 9C80 Ø0 7 9C92 A6 9 9C92 A6 9 9C83 A7 9 9C84 A7 9 9C86 A	A8 Ø1 9Ø 64 Ø1 5Ø 62 Ø1 1A 63 ØØ C7 61 ØØ 8B 67 61 16 64 ØØ 8B 67 60 67 61 66 47 43 68 48 48 12 68 48 68 68 68 68 68 68 68 68 68 68 68 68 68	* 05 06 LS * 01 7A LS 01 3D 01 0A 00 DF * 00 8B LS 00 9D 00 83 00 6E * 83 7B LS 3F 3B LS 3F 3B LS 2B 2B 2B 13 12	TABLE OF FOR FOR FOR FOR FOR FOR FOR FOR FOR	NUMERICAL NOTE VALUES FOR LETTER 3 10,12,1,3,5,6,8 DELAYS FOR OCTAVE 1 3 \$01A8,\$0190,\$017A 3 \$0164,\$0150,\$013D 3 \$0128,\$011A,\$010A 3 \$00FB,\$00ED,\$00DF DELAYS FOR OCTAVE 2 3 \$0003,\$00C7,\$008B 3 \$0003,\$00C7,\$008B 3 \$0003,\$00C7,\$008B 3 \$0007,\$0075,\$00BE DELAYS FOR OCTAVE 2 3 \$0003,\$00C7,\$00BB 3 \$0007,\$0075,\$00BE DELAYS FOR OCTAVES 3,4,5 3 \$007C,\$0075,\$00BE DELAYS FOR OCTAVES 3,4,5 3 \$19,\$10,\$10,\$10,\$10,\$10,\$10,\$10,\$10,\$10,\$10	* X=Ø. ACCB=1; END OF DRAW COMMAND LINE VALUES - * WHEN THESE VALUES ARE PULLED OFF THE * STACK, THE DRAW COMMAND WILL END SET PSET/PRESET FLAG TO PSET
3900 3901 3902 3903 3905 3906 3908 3909 3910 3911 3912 3914 3915 3916 3917 3916 3917 3918 3919 3920 3921 3922 3923 3923 3923 3925 3926 3927 3928 3929 3929 3929 3929 3929 3929 3929	9C58 0A 0 9C61 08 9 9C62 01 A 9C68 01 6 9C68 01 6 9C74 00 F 9C80 00 7 9C80 00 7 9C92 A6 9 9C95 1 4 9C95 1 4 9C96 01 9 1	A8 Ø1 9Ø 64 Ø1 5Ø 62 Ø1 1A 63 ØØ C7 61 ØØ 8B 64 Ø0 8B 65 ØØ 75 66 Ø0 75 67 Ø0 75 68 Ø0 67 61 68 Ø0 67 61 69 Ø0 67 60 93 88 60 67 61 61 64 64 64 64 64 64 65 65 65 66 66 66 66 66 66 66 66 66 66 66 66 6	* 05 06 L9 * 01 7A L9 01 10 A 00 DF * 00 BB L9 00 9B 00 9B 00 8B 00 6E * 83 7B L9 5B 5B 5B 5B 5B 5B 5B 10 1B 13 12	TABLE	NUMERICAL NOTE VALUES FOR LETTER 3 10,12,1,3,5,6,8 DELAYS FOR OCTAVE 1 3 \$01AB,\$0190,\$017A 3 \$01AB,\$0190,\$017A 3 \$01C4,\$0150,\$013D 3 \$012B,\$011A,\$010A 3 \$00FB,\$00ED,\$00DF DELAYS FOR OCTAVE 2 3 \$0003,\$00C7,\$00BB 3 \$0003,\$00C7,\$00BB 3 \$0001,\$00AC5,\$00BD 3 \$00094,\$00AB,\$00BD 3 \$00094,\$00AB,\$00BD 3 \$0094,\$00AB,\$00BB 3 \$0094,\$00AB,\$00BB 3 \$0094,\$00AB,\$00BB 3 \$0094,\$00AB,\$00BB 3 \$00B,\$00AC5,\$00BB 3 \$00B,\$00AC5,\$00BB 3 \$00B,\$00AC5,\$00BB 3 \$00B,\$00AC5,\$00BB 3 \$00B,\$00AC5,\$00BB 3 \$00B,\$00AC5,\$00BB 3 \$19,\$18,\$16,\$1,\$51,\$51,\$51 3 \$19,\$18,\$16,\$14,\$13,\$12 (C ZERO 3 #\$01 15 X,B 3 SEFFLG (VD5 4 L959A 4 LB156 4 LB156 4 LB156 6 LB654	* X=0. ACCB=1; END OF DRAW COMMAND LINE VALUES - * WHEN THESE VALUES ARE PULLED OFF THE * STACK, THE DRAW COMMAND WILL END SET PSET/PRESET FLAG TO PSET CLEAR UPDATE FLAG AND DRAW FLAG SET ACTIVE COLOR BYTE EVALUATE EXPRESSION GET THE LENGTH AND ADDRESS OF THE COMMAND STRING
3900 3901 3902 3903 3905 3906 3906 3909 3910 3911 3912 3913 3914 3915 3917 3918 3919 3920 3921 3922 3923 3924 3925 3926 3927 3928 3929 3939 3932 3932 3932 3932 3932	9C5B 0A 0 9C62 01 A 9C68 01 6 9C68 01 6 9C74 00 F 9C74 00 F 9C80 00 9 9C80 00 7 9C92 A6 9 9C92 A6 9 9C92 51 4 9C94 37 3 9C4A 26 2 9C80 19 1 9C80 07 7 9C80 07 7 9C80 07 7 9C80 08 7 9C80 08 8 8 8 9 9C80 08 9 9C80	A8 Ø1 9Ø 64 Ø1 5Ø 62 Ø1 1A 63 ØØ C7 61 ØØ 8B 67 C ØØ 75 60 93 8B 60 67 61 61 C 47 43 63 21 1F 68 16 14 68 16 14 68 16 14 68 16 14 68 16 14 68 16 14	* * 05 06 LS * 01 7A LS 01 01 0A 00 DF * 00 8B LS 00 6E * 83 7B LS 3F 3B 2B 28 1D 1B 13 12	TABLE OF FDI FDI FDI FDI FDI FDI FDI FDI FDI FD	NUMERICAL NOTE VALUES FOR LETTER 3 10,12,1,3,5,6,8 DELAYS FOR OCTAVE 1 3 \$01A8,\$0190,\$017A 3 \$01A8,\$0190,\$017A 3 \$0128,\$011A,\$010A 3 \$00FB,\$00ED,\$00DF DELAYS FOR OCTAVE 2 3 \$0003,\$00C7,\$00BB 3 \$0001,\$00A6,\$00D 3 \$007C,\$0075,\$00BB 3 \$007C,\$0075,\$00BB DELAYS FOR OCTAVE 2 3 \$0007,\$0075,\$00BB 3 \$007C,\$0075,\$00BB 4 \$00BB,\$00BB,\$00BB 5 \$00BB,\$00BB,\$00BB 5 \$00BB,\$00BB,\$00BB 5 \$00BB,\$00BB,\$00BB 5 \$00BB,\$00BB,\$00BB 6 \$00BB,\$00BB,\$00BB,\$00BB 6 \$00BB,\$00BB,\$00BB,\$00BB 6 \$00BB,\$00BB,\$00BB,\$00BB 6 \$00BB,\$00BB,\$00BB,\$00BB 6 \$00BB,\$	* NOTES NOTES A,B,C,D,E,F,G DELAYS FOR OCTAVE 1 DELAYS FOR OCTAVE 2 DELAYS FOR OCTAVES 3,4,5 * X=Ø. ACCB=1; END OF DRAW COMMAND LINE VALUES - * WHEN THESE VALUES ARE PULLED OFF THE * STACK, THE DRAW COMMAND WILL END SET PSET/PRESET FLAG TO PSET CLEAR UPDATE FLAG AND DRAW FLAG SET ACTIVE COLOR BYTE EVALUATE EXPRESSION GET THE LENGTH AND ADDRESS OF THE COMMAND STRING INTERPRET THE COMMAND STRING
3900 3901 3902 3903 3905 3906 3908 3909 3910 3911 3912 3914 3915 3916 3917 3916 3917 3918 3919 3920 3921 3922 3923 3923 3923 3925 3926 3927 3928 3929 3929 3929 3929 3929 3929 3929	9C58 0A 0 9C61 08 9 9C62 01 A 9C68 01 6 9C68 01 6 9C74 00 F 9C80 00 7 9C80 00 7 9C92 A6 9 9C95 1 4 9C95 1 4 9C96 01 9 1	A8 Ø1 9Ø 54 Ø1 5Ø 88 Ø1 1A 88 Ø2 Ø1 1A 88 Ø3 Ø0 C7 81 Ø0 A6 84 Ø0 8B 87 Ø0 75 80 93 8B 87 Ø0 75 80 93 8B 81 Ø1 A6 84 Ø1 14 84 Ø1 8B 84 Ø1 14 85 Ø1 16 14 86 Ø1 16 16 16 86 §1 Ø2 Ø2 Ø3	* * 05 06 LS * 01 7A LS 01 01 0A 00 DF * 00 8B LS 00 6E * 83 7B LS 3F 3B 2B 28 1D 1B 13 12	TABLE	NUMERICAL NOTE VALUES FOR LETTER 3 10,12,1,3,5,6,8 DELAYS FOR OCTAVE 1 3 \$01A8,\$0190,\$017A 3 \$0164,\$0150,\$013D 3 \$0128,\$011A,\$013A 3 \$0078,\$000F DELAYS FOR OCTAVE 2 3 \$0003,\$00C7,\$000BB 3 \$000B1,\$000A6,\$000D DELAYS FOR OCTAVE 2 3 \$000A6,\$000B,\$000B 3 \$0007C,\$0075,\$006E DELAYS FOR OCTAVES 3,4,5 3 \$007C,\$0075,\$006E DELAYS FOR OCTAVES 3,4,5 3 \$007C,\$0075,\$008B 3 \$007C,\$0075,\$008B 3 \$007C,\$0075,\$008B 3 \$1,500,\$008B,\$0083 3 \$007C,\$0075,\$008B 3 \$1,500,\$008B,\$0083 3 \$007C,\$0075,\$008B 3 \$1,500,\$008B,\$0083 3 \$1,500,\$008B,\$0	* X=0. ACCB=1; END OF DRAW COMMAND LINE VALUES - * WHEN THESE VALUES ARE PULLED OFF THE * STACK, THE DRAW COMMAND WILL END SET PSET/PRESET FLAG TO PSET CLEAR UPDATE FLAG AND DRAW FLAG SET ACTIVE COLOR BYTE EVALUATE EXPRESSION GET THE LENGTH AND ADDRESS OF THE COMMAND STRING

2027			
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			SET COMMAND LINE ADDRESS
	9CD7 9F D9		
3941	9CD9 10 27 00 EA	LBEQ L9DC7	EXIT ROUTINE IF ADDRESS = Ø
3942	9CDD ØD D8	L9CDD TST VD8	TEST COMMAND LENGTH CTR
3943	9CDF 27 FØ	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 Ø4	BNE L9CF4	NO
3951	9CFØ Ø3 D5	COM VD5	TOGGLE UPDATE FLAG Ø = UPDATE, FF = NO UPDATE
3952	9CF2 2Ø E9	BRA L9CDD	GET NEXT COMMAND
3953	9CF4 81 42	L9CF4 CMPA #'B'	CHECK DRAW FLAG?
3954	9CF6 26 Ø4	BNE L9CFC	NO
3955	9CF8 Ø3 D6	COM VD6	TOGGLE DRAW FLAG Ø = DRAW LINE, FF = DON T DRAW LINE
3956	9CFA 2Ø 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	9DØ2 81 4D	CMPA #'M'	MOVE THE DRAW POSITION ?
3960	9DØ4 1Ø 27 Ø1 2A	LBEQ L9E32	YES; GO MOVE IT
3961	9DØ8 34 Ø2	PSHS A	SAVE CURRENT COMMAND
3962	9DØA C6 Ø1	LDB #\$Ø1	DEFAULT VALUE IF NO NUMBER FOLLOWS COMMAND
3963	9DØC ØD D8	TST VD8	CHECK COMMAND LENGTH CTR
3964	9DØE 27 11	BEQ L9D21	BRANCH IF NO COMMANDS LEFT
3965	9D1Ø BD 9B 98	JSR L9B98	GET A COMMAND CHAR
3966	9D13 BD B3 A2	JSR LB3A2	SET CARRY IF NOT ALPHA
3967	9D16 34 Ø1	PSHS CC	SAVE CARRY FLAG
3968	9D18 BD 9B E2	JSR L9BE2	MOVE COMMAND POINTER BACK ONE
3969			
	9D1B 35 Ø1	PULS CC	RESTORE CARRY FLAG
3970	9D1D 24 Ø2	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 Ø2	L9D21 PULS A	GET CURRENT COMMAND BACK
3973	9D23 81 43	CMPA #'C'	CHANGE COLOR?
3974	9D25 27 28	BEQ L9D4F	YES
3975		CMPA #'A'	
	9D27 81 41		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 8Ø 45	SUBA #'E'	MASK OFF ASCII FOR LETTER E-H COMMAND CHECKS
3988	9D41 27 2F	BEQ L9D72	BRANCH IF E (45 DEGREES)
			BRANCH IF E (45 DEGREES)
3989	9D43 4A	DECA	10041011 75 5 (405 0500550)
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)
	9D4C 7E B4 4A	L9D4C JMP LB44A	FC ERROR IF ILLEGAL COMMAND
	304C /L D4 4A	LJDTC UIII LDTTA	TO ERROR IT TELEGAE COMMAND
3995			
3996			
3996 3997		* CHANGE COLOR	
3996	9D4F BD 95 5D	L9D4F JSR L955D	ADJUST COLOR CODE FOR PROPER PMODE
3996 3997	9D4F BD 95 5D 9D52 D7 B2		ADJUST COLOR CODE FOR PROPER PMODE SAVE NEW FOREGROUND COLOR
3996 3997 3998	9D52 D7 B2	L9D4F JSR L955D STB FORCOL	SAVE NEW FOREGROUND COLOR
3996 3997 3998 3999 4000	9D52 D7 B2 9D54 BD 95 9A	L9D4F JSR L955D STB FORCOL JSR L959A	SAVE NEW FOREGROUND COLOR SET COLOR BYTES (WCOLOR,ALLCOL)
3996 3997 3998 3999 4000 4001	9D52 D7 B2	L9D4F JSR L955D STB FORCOL JSR L959A L9D57 BRA L9CDD	SAVE NEW FOREGROUND COLOR
3996 3997 3998 3999 4000 4001 4002	9D52 D7 B2 9D54 BD 95 9A 9D57 2Ø 84	L9D4F JSR L955D STB FORCOL JSR L959A L9D57 BRA L9CDD * CHANGE ANGLE	SAVE NEW FOREGROUND COLOR SET COLOR BYTES (WCOLOR, ALLCOL) GO PROCESS ANOTHER COMMAND
3996 3997 3998 3999 4000 4001 4002 4003	9D52 D7 B2 9D54 BD 95 9A 9D57 2Ø 84 9D59 C1 Ø4	L9D4F JSR L955D STB FORCOL JSR L959A L9D57 BRA L9CDD * CHANGE ANGLE L9D59 CMPB #\$04	SAVE NEW FOREGROUND COLOR SET COLOR BYTES (WCOLOR, ALLCOL) GO PROCESS ANOTHER COMMAND ONLY 0-3 ARE LEGAL
3996 3997 3998 3999 4000 4001 4002 4003 4004	9D52 D7 B2 9D54 BD 95 9A 9D57 20 84 9D59 C1 04 9D58 24 EF	L9D4F JSR L955D STB FORCOL JSR L959A L9D57 BRA L9CDD * CHANGE ANGLE L9D59 CMPB #\$Ø4 BCC L9D4C	SAVE NEW FOREGROUND COLOR SET COLOR BYTES (WCOLOR,ALLCOL) GO PROCESS ANOTHER COMMAND ONLY Ø-3 ARE LEGAL FC ERROR IF ANGLE NUMBER > 3
3996 3997 3998 3999 4000 4001 4002 4003 4004 4005	9D52 D7 B2 9D54 BD 95 9A 9D57 20 84 9D59 C1 04 9D58 24 EF 9D5D D7 E8	L9D4F JSR L955D STB FORCOL JSR L959A L9D57 BRA L9CDD * CHANGE ANGLE L9D59 CMPB #\$04 BCC L9D4C STB ANGLE	SAVE NEW FOREGROUND COLOR SET COLOR BYTES (WCOLOR,ALLCOL) GO PROCESS ANOTHER COMMAND ONLY Ø-3 ARE LEGAL FC ERROR IF ANGLE NUMBER > 3 SAVE DRAW ANGLE
3996 3997 3998 3999 4000 4001 4002 4003 4004 4005 4006	9D52 D7 B2 9D54 BD 95 9A 9D57 20 84 9D59 C1 04 9D58 24 EF	L9D4F JSR L955D STB FORCOL JSR L959A L9D57 BRA L9CDD * CHANGE ANGLE L9D59 CMPB #\$04 BCC L9D4C STB ANGLE BRA L9D57	SAVE NEW FOREGROUND COLOR SET COLOR BYTES (WCOLOR,ALLCOL) GO PROCESS ANOTHER COMMAND ONLY Ø-3 ARE LEGAL FC ERROR IF ANGLE NUMBER > 3
3996 3997 3998 3999 4000 4001 4002 4003 4004 4005	9D52 D7 B2 9D54 BD 95 9A 9D57 20 84 9D59 C1 04 9D58 24 EF 9D5D D7 E8	L9D4F JSR L955D STB FORCOL JSR L959A L9D57 BRA L9CDD * CHANGE ANGLE L9D59 CMPB #\$04 BCC L9D4C STB ANGLE	SAVE NEW FOREGROUND COLOR SET COLOR BYTES (WCOLOR,ALLCOL) GO PROCESS ANOTHER COMMAND ONLY Ø-3 ARE LEGAL FC ERROR IF ANGLE NUMBER > 3 SAVE DRAW ANGLE
3996 3997 3998 3999 4000 4001 4002 4003 4004 4005 4006	9D52 D7 B2 9D54 BD 95 9A 9D57 20 84 9D59 C1 04 9D58 24 EF 9D5D D7 E8	L9D4F JSR L955D STB FORCOL JSR L959A L9D57 BRA L9CDD * CHANGE ANGLE L9D59 CMPB #\$04 BCC L9D4C STB ANGLE BRA L9D57	SAVE NEW FOREGROUND COLOR SET COLOR BYTES (WCOLOR,ALLCOL) GO PROCESS ANOTHER COMMAND ONLY Ø-3 ARE LEGAL FC ERROR IF ANGLE NUMBER > 3 SAVE DRAW ANGLE
3996 3997 3998 3999 4000 4001 4002 4003 4004 4005 4006 4007 4008	9052 D7 B2 9054 B0 95 9A 9057 20 84 9059 C1 04 9058 24 EF 905D D7 E8 905F 20 F6 9061 C1 3F	L9D4F JSR L955D STB FORCOL JSR L959A L9D57 BRA L9CDD * CHANGE ANGLE L9D59 CMPB #\$Ø4 BCC L9D4C STB ANGLE BRA L9D57 * CHANGE SCALE L9D61 CMPB #63	SAVE NEW FOREGROUND COLOR SET COLOR BYTES (WCOLOR, ALLCOL) GO PROCESS ANOTHER COMMAND ONLY Ø-3 ARE LEGAL FC ERROR IF ANGLE NUMBER > 3 SAVE DRAW ANGLE GO PROCESS ANOTHER COMMAND ONLY Ø-63 ARE LEGAL
3996 3997 3998 3999 4000 4001 4002 4003 4004 4005 4006 4007 4008 4009	9052 D7 B2 9054 B0 95 9A 9057 20 84 9059 C1 04 9058 24 EF 9050 D7 E8 905F 20 F6 9061 C1 3F 9063 24 E7	L9D4F JSR L955D STB FORCOL JSR L959A L9D57 BRA L9CDD * CHANGE ANGLE L9D59 CMPB #\$04 BCC L9D4C STB ANGLE BRA L9D57 * CHANGE SCALE L9D61 CMPB #63 BCC L9D4C	SAVE NEW FOREGROUND COLOR SET COLOR BYTES (WCOLOR, ALLCOL) GO PROCESS ANOTHER COMMAND ONLY Ø-3 ARE LEGAL FC ERROR IF ANGLE NUMBER > 3 SAVE DRAW ANGLE GO PROCESS ANOTHER COMMAND ONLY Ø-63 ARE LEGAL FC ERROR IF SCALE > 63
3996 3997 3998 3999 4000 4001 4002 4003 4004 4005 4006 4007 4008 4009 4010	9052 07 82 9054 80 95 9A 9057 20 84 9059 C1 04 9058 24 EF 9050 07 E8 9057 20 F6 9061 C1 3F 9063 24 E7 9065 07 E9	L9D4F JSR L955D STB FORCOL JSR L959A L9D57 BRA L9CDD * CHANGE ANGLE L9D59 CMPB #\$04 BCC L9D4C STB ANGLE BRA L9D57 * CHANGE SCALE L9D61 CMPB #63 BCC L9D4C STB SCALE	SAVE NEW FOREGROUND COLOR SET COLOR BYTES (WCOLOR,ALLCOL) GO PROCESS ANOTHER COMMAND ONLY Ø-3 ARE LEGAL FC ERROR IF ANGLE NUMBER > 3 SAVE DRAW ANGLE GO PROCESS ANOTHER COMMAND ONLY Ø-63 ARE LEGAL FC ERROR IF SCALE > 63 SAVE DRAW SCALE
3996 3997 3998 3999 4000 4001 4002 4003 4004 4005 4006 4007 4008 4008 4010 4011	9052 D7 B2 9054 B0 95 9A 9057 20 84 9059 C1 04 9058 24 EF 9050 D7 E8 905F 20 F6 9061 C1 3F 9063 24 E7	L9D4F JSR L955D STB FORCOL JSR L959A L9D57 BRA L9CDD * CHANGE ANGLE L9D59 CMPB #\$04 BCC L9D4C STB ANGLE BRA L9D57 * CHANGE SCALE L9D61 CMPB #63 BCC L9D4C	SAVE NEW FOREGROUND COLOR SET COLOR BYTES (WCOLOR, ALLCOL) GO PROCESS ANOTHER COMMAND ONLY Ø-3 ARE LEGAL FC ERROR IF ANGLE NUMBER > 3 SAVE DRAW ANGLE GO PROCESS ANOTHER COMMAND ONLY Ø-63 ARE LEGAL FC ERROR IF SCALE > 63
3996 3997 3998 3999 4000 4001 4002 4003 4004 4005 4006 4007 4008 4009 4011 4011	9052 07 82 9054 80 95 9A 9057 20 84 9059 C1 04 9058 24 EF 9050 07 E8 9057 20 F6 9061 C1 3F 9063 24 E7 9065 07 E9	L9D4F JSR L955D STB FORCOL JSR L959A L9557 BRA L9CDD * CHANGE ANGLE L9D59 CMPB #\$04 BCC L9D4C STB ANGLE BRA L9D57 * CHANGE SCALE L9D61 CMPB #63 BCC L9D4C STB SCALE BRA L9D57	SAVE NEW FOREGROUND COLOR SET COLOR BYTES (WCOLOR,ALLCOL) GO PROCESS ANOTHER COMMAND ONLY Ø-3 ARE LEGAL FC ERROR IF ANGLE NUMBER > 3 SAVE DRAW ANGLE GO PROCESS ANOTHER COMMAND ONLY Ø-63 ARE LEGAL FC ERROR IF SCALE > 63 SAVE DRAW SCALE
3996 3997 3998 4000 4001 4002 4003 4004 4005 4006 4007 4009 4010 4011 4012 4013	9052 07 82 9054 80 95 9A 9057 20 84 9059 C1 04 9058 24 EF 9050 07 E8 9057 20 F6 9061 C1 3F 9063 24 E7 9065 07 E9 9067 20 EE	L9D4F JSR L955D STB FORCOL JSR L959A L9D57 BRA L9CDD * CHANGE ANGLE L9D59 CMPB #\$04 BCC L9D4C STB ANGLE BRA L9D57 * CHANGE SCALE L9D61 CMPB #63 BCC L9D4C STB SCALE L9D61 SCALE L9D61 SCALE L9D61 SCALE L9D63 BCC L9D4C STB SCALE BRA L9D57	SAVE NEW FOREGROUND COLOR SET COLOR BYTES (WCOLOR, ALLCOL) GO PROCESS ANOTHER COMMAND ONLY Ø-3 ARE LEGAL FC ERROR IF ANGLE NUMBER > 3 SAVE DRAW ANGLE GO PROCESS ANOTHER COMMAND ONLY Ø-63 ARE LEGAL FC ERROR IF SCALE > 63 SAVE DRAW SCALE GO PROCESS ANOTHER COMMAND
3996 3997 3998 3999 4000 4001 4002 4003 4004 4005 4006 4007 4008 4009 4011 4011	9052 D7 B2 9054 B0 95 9A 9057 20 84 9059 C1 04 9058 24 EF 905D D7 E8 905F 20 F6 9061 C1 3F 9063 24 E7 9065 D7 E9 9067 20 EE	L9D4F JSR L955D STB FORCOL JSR L959A L9D57 BRA L9CDD * CHANGE ANGLE L9D59 CMPB #\$04 BCC L9D4C STB ANGLE BRA L9D57 * CHANGE SCALE L9D61 CMPB #63 BCC L9D4C STB SCALE L9D61 CMPB #03 BCC L9D4C STB SCALE L9D69 CLRA	SAVE NEW FOREGROUND COLOR SET COLOR BYTES (WCOLOR,ALLCOL) GO PROCESS ANOTHER COMMAND ONLY Ø-3 ARE LEGAL FC ERROR IF ANGLE NUMBER > 3 SAVE DRAW ANGLE GO PROCESS ANOTHER COMMAND ONLY Ø-63 ARE LEGAL FC ERROR IF SCALE > 63 SAVE DRAW SCALE
3996 3997 3998 4000 4001 4002 4003 4004 4005 4006 4007 4009 4010 4011 4012 4013	9052 07 82 9054 80 95 9A 9057 20 84 9059 C1 04 9058 24 EF 9050 07 E8 9057 20 F6 9061 C1 3F 9063 24 E7 9065 07 E9 9067 20 EE	L9D4F JSR L955D STB FORCOL JSR L959A L9D57 BRA L9CDD * CHANGE ANGLE L9D59 CMPB #\$04 BCC L9D4C STB ANGLE BRA L9D57 * CHANGE SCALE L9D61 CMPB #63 BCC L9D4C STB SCALE L9D61 SCALE L9D61 SCALE L9D61 SCALE L9D63 BCC L9D4C STB SCALE BRA L9D57	SAVE NEW FOREGROUND COLOR SET COLOR BYTES (WCOLOR, ALLCOL) GO PROCESS ANOTHER COMMAND ONLY Ø-3 ARE LEGAL FC ERROR IF ANGLE NUMBER > 3 SAVE DRAW ANGLE GO PROCESS ANOTHER COMMAND ONLY Ø-63 ARE LEGAL FC ERROR IF SCALE > 63 SAVE DRAW SCALE GO PROCESS ANOTHER COMMAND
3996 3997 3998 4000 4001 4002 4003 4004 4005 4006 4007 4008 4009 4010 4011 4011 4012 4013	9052 D7 B2 9054 B0 95 9A 9057 20 84 9059 C1 04 9058 24 EF 905D D7 E8 905F 20 F6 9061 C1 3F 9063 24 E7 9065 D7 E9 9067 20 EE	L9D4F JSR L955D STB FORCOL JSR L959A L9D57 BRA L9CDD * CHANGE ANGLE L9D59 CMPB #\$04 BCC L9D4C STB ANGLE BRA L9D57 * CHANGE SCALE L9D61 CMPB #63 BCC L9D4C STB SCALE L9D61 CMPB #03 BCC L9D4C STB SCALE L9D69 CLRA	SAVE NEW FOREGROUND COLOR SET COLOR BYTES (WCOLOR, ALLCOL) GO PROCESS ANOTHER COMMAND ONLY Ø-3 ARE LEGAL FC ERROR IF ANGLE NUMBER > 3 SAVE DRAW ANGLE GO PROCESS ANOTHER COMMAND ONLY Ø-63 ARE LEGAL FC ERROR IF SCALE > 63 SAVE DRAW SCALE GO PROCESS ANOTHER COMMAND
3996 3997 3998 3999 4000 4001 4002 4003 4004 4005 4006 4007 4008 4009 4010 4011 4011 4013 4014 4015	9052 D7 B2 9054 B0 95 9A 9057 20 84 9059 C1 04 9058 24 EF 905D D7 E8 905F 20 F6 9061 C1 3F 9063 24 E7 9065 D7 E9 9067 20 EE	L9D4F JSR L955D STB FORCOL JSR L959A L9D57 BRA L9CDD * CHANGE ANGLE L9D59 CMPB #\$04 BCC L9D4C STB ANGLE BRA L9D57 * CHANGE SCALE L9D61 CMPB #63 BCC L9D4C STB SCALE L9D61 CMPB #63 BCC L9D4C STB SCALE L9D61 CMPB #63 STB SCALE BRA L9D57 * 315 DEGREES L9D69 CLRA BSR L9DC4 L9D6C FCB SKP1	SAVE NEW FOREGROUND COLOR SET COLOR BYTES (WCOLOR, ALLCOL) GO PROCESS ANOTHER COMMAND ONLY Ø-3 ARE LEGAL FC ERROR IF ANGLE NUMBER > 3 SAVE DRAW ANGLE GO PROCESS ANOTHER COMMAND ONLY Ø-63 ARE LEGAL FC ERROR IF SCALE > 63 SAVE DRAW SCALE GO PROCESS ANOTHER COMMAND *NEGATE ACCD - MAKE HORIZONTAL *DIFFERENCE NEGATIVE
3996 3997 3998 3999 4000 4001 4002 4003 4004 4005 4006 4007 4008 4009 4010 4011 4012 4013 4014 4015 4016 4016 4016 4016 4017	9052 D7 B2 9054 B0 95 9A 9057 20 84 9059 C1 04 9058 24 EF 905D D7 E8 905F 20 F6 9061 C1 3F 9063 24 E7 9065 D7 E9 9067 20 EE	L9D4F JSR L955D STB FORCOL JSR L959A L9D57 BRA L9CDD * CHANGE ANGLE L9D59 CMPB #\$04 BCC L9D4C STB ANGLE BRA L9D57 * CHANGE SCALE L9D61 CMPB #63 BCC L9D4C STB SCALE L9D61 CMPB #63 BCC L9D4C STB SCALE L9D61 CMPB #63 BCC L9D4C STB SCALE L9D66 FGB SKP1 * 315 DEGREES	SAVE NEW FOREGROUND COLOR SET COLOR BYTES (WCOLOR, ALLCOL) GO PROCESS ANOTHER COMMAND ONLY Ø-3 ARE LEGAL FC ERROR IF ANGLE NUMBER > 3 SAVE DRAW ANGLE GO PROCESS ANOTHER COMMAND ONLY Ø-63 ARE LEGAL FC ERROR IF SCALE > 63 SAVE DRAW SCALE GO PROCESS ANOTHER COMMAND *NEGATE ACCD - MAKE HORIZONTAL *DIFFERENCE NEGATIVE SKIP ONE BYTE - KEEP HORIZONTAL DIFFERENCE NEGATIVE
3996 3998 3999 48001 4801 4802 4803 4804 4805 4806 4807 4809 4811 4812 4813 4814 4815 4816 4817 4818	9052 D7 B2 9054 B0 95 9A 9057 20 84 9059 C1 04 9058 C1 04 9050 D7 E8 9050 D7 E8 9051 C1 3F 9063 C4 E7 9065 D7 E9 9067 20 EE	L9D4F JSR L955D STB FORCOL JSR L959A L9557 BRA L9CDD * CHANGE ANGLE L9D59 CMPB #\$04 BCC L9D4C STB ANGLE BRA L9D57 * CHANGE SCALE L9D61 CMPB #63 BCC L9D4C STB SCALE L9D61 CMPB #63 BCC L9D4C STB SCALE L9D61 CMPB #63 BCC L9D4C STB SCALE L9D60 FCB SCALE L9D60 CLRA L9D67	SAVE NEW FOREGROUND COLOR SET COLOR BYTES (WCOLOR, ALLCOL) GO PROCESS ANOTHER COMMAND ONLY Ø-3 ARE LEGAL FC ERROR IF ANGLE NUMBER > 3 SAVE DRAW ANGLE GO PROCESS ANOTHER COMMAND ONLY Ø-63 ARE LEGAL FC ERROR IF SCALE > 63 SAVE DRAW SCALE GO PROCESS ANOTHER COMMAND *NEGATE ACCD - MAKE HORIZONTAL *DIFFERENCE NEGATIVE SKIP ONE BYTE - KEEP HORIZONTAL DIFFERENCE NEGATIVE CLEAR MS BYTE OF HORIZONTAL DIFFERENCE
3996 3997 3998 3999 4000 4001 4002 4003 4004 4005 4006 4007 4008 4010 4011 4011 4011 4014 4014 4015 4014 4017 4018	9052 D7 B2 9054 B0 95 9A 9057 20 84 9059 C1 04 9058 24 EF 9050 D7 E8 905F 20 F6 9061 C1 3F 9063 24 E7 9065 D7 E9 9067 20 EE 9069 4F 9068 8D 58 906C 21 9060 4F 9060 4F 9060 4F 9060 4F	L9D4F JSR L955D STB FORCOL JSR L959A L9D57 BRA L9CDD * CHANGE ANGLE L9D59 CMPB #\$04 BCC L9D4C STB ANGLE BRA L9D57 * CHANGE SCALE L9D61 CMPB #63 BCC L9D4C STB SCALE L9D61 CMPB #63 BCC L9D4C STB SCALE L9D61 CMPB #63 BCC L9D4C STB SCALE BRA L9D57 * 315 DEGREES L9D69 CLRA BSR 19DC4 L9D6C FCB SKP1 * 135 DEGREES L9D6D CLRA TFR D, X	SAVE NEW FOREGROUND COLOR SET COLOR BYTES (WCOLOR, ALLCOL) GO PROCESS ANOTHER COMMAND ONLY Ø-3 ARE LEGAL FC ERROR IF ANGLE NUMBER > 3 SAVE DRAW ANGLE GO PROCESS ANOTHER COMMAND ONLY Ø-63 ARE LEGAL FC ERROR IF SCALE > 63 SAVE DRAW SCALE GO PROCESS ANOTHER COMMAND *NEGATE ACCD - MAKE HORIZONTAL *DIFFERENCE NEGATIVE SKIP ONE BYTE - KEEP HORIZONTAL DIFFERENCE NEGATIVE CLEAR MS BYTE OF HORIZONTAL DIFFERENCE COPY HORIZONTAL DIFFERENCE
3996 3998 3999 48001 4801 4802 4803 4804 4805 4806 4807 4809 4811 4812 4813 4814 4815 4816 4817 4818	9052 D7 B2 9054 B0 95 9A 9057 20 84 9059 C1 04 9058 C1 04 9050 D7 E8 9050 D7 E8 9051 C1 3F 9063 C4 E7 9065 D7 E9 9067 20 EE	L9D4F JSR L955D STB FORCOL JSR L959A L9557 BRA L9CDD * CHANGE ANGLE L9D59 CMPB #\$04 BCC L9D4C STB ANGLE BRA L9D57 * CHANGE SCALE L9D61 CMPB #63 BCC L9D4C STB SCALE L9D61 CMPB #63 BCC L9D4C STB SCALE L9D61 CMPB #63 BCC L9D4C STB SCALE L9D60 FCB SCALE L9D60 CLRA L9D67	SAVE NEW FOREGROUND COLOR SET COLOR BYTES (WCOLOR, ALLCOL) GO PROCESS ANOTHER COMMAND ONLY Ø-3 ARE LEGAL FC ERROR IF ANGLE NUMBER > 3 SAVE DRAW ANGLE GO PROCESS ANOTHER COMMAND ONLY Ø-63 ARE LEGAL FC ERROR IF SCALE > 63 SAVE DRAW SCALE GO PROCESS ANOTHER COMMAND *NEGATE ACCD - MAKE HORIZONTAL *DIFFERENCE NEGATIVE SKIP ONE BYTE - KEEP HORIZONTAL DIFFERENCE NEGATIVE CLEAR MS BYTE OF HORIZONTAL DIFFERENCE
3996 3997 3998 3999 4000 4001 4002 4003 4004 4005 4006 4007 4008 4010 4011 4011 4011 4014 4014 4015 4014 4017 4018	9052 D7 B2 9054 B0 95 9A 9057 20 84 9059 C1 04 9058 24 EF 9050 D7 E8 905F 20 F6 9061 C1 3F 9063 24 E7 9065 D7 E9 9067 20 EE 9069 4F 9068 8D 58 906C 21 9060 4F 9060 4F 9060 4F 9060 4F	L9D4F JSR L955D STB FORCOL JSR L959A L9D57 BRA L9CDD * CHANGE ANGLE L9D59 CMPB #\$04 BCC L9D4C STB ANGLE BRA L9D57 * CHANGE SCALE L9D61 CMPB #63 BCC L9D4C STB SCALE L9D61 CMPB #63 BCC L9D4C STB SCALE L9D61 CMPB #63 BCC L9D4C STB SCALE BRA L9D57 * 315 DEGREES L9D69 CLRA BSR 19DC4 L9D6C FCB SKP1 * 135 DEGREES L9D6D CLRA TFR D, X	SAVE NEW FOREGROUND COLOR SET COLOR BYTES (WCOLOR, ALLCOL) GO PROCESS ANOTHER COMMAND ONLY Ø-3 ARE LEGAL FC ERROR IF ANGLE NUMBER > 3 SAVE DRAW ANGLE GO PROCESS ANOTHER COMMAND ONLY Ø-63 ARE LEGAL FC ERROR IF SCALE > 63 SAVE DRAW SCALE GO PROCESS ANOTHER COMMAND *NEGATE ACCD - MAKE HORIZONTAL *DIFFERENCE NEGATIVE SKIP ONE BYTE - KEEP HORIZONTAL DIFFERENCE NEGATIVE CLEAR MS BYTE OF HORIZONTAL DIFFERENCE COPY HORIZONTAL DIFFERENCE
3996 3997 3998 3999 4000 4001 4002 4003 4004 4005 4006 4007 4008 4007 4010 4011 4011 4011 4011 4011 4014 4015 4016 4017 4018 4018 4019 4018 4019 4018 4019 4019 4019 4019 4019 4019 4019 4019	9052 D7 B2 9054 B0 95 9A 9057 20 84 9059 C1 04 9058 24 EF 9050 D7 E8 905F 20 F6 9061 C1 3F 9063 24 E7 9065 D7 E9 9067 20 EE 9069 4F 9068 8D 58 906C 21 9060 4F 9060 4F 9060 4F 9060 4F	L9D4F JSR L955D STB FORCOL JSR L959A L9557 BRA L9CDD * CHANGE ANGLE L9D59 CMPB #\$04 BCC L9D4C STB ANGLE L9D61 CMPB #63 BCC L9D4C STB SCALE L9D61 CMPB #63 BCC L9D4C STB SCALE L9D67 * 315 DEGREES L9D60 CLRA BSR L9D64 L9D6C FCB SKP1 * 135 DEGREES L9D60 CLRA TFR D, X BRA L9DCB	SAVE NEW FOREGROUND COLOR SET COLOR BYTES (WCOLOR, ALLCOL) GO PROCESS ANOTHER COMMAND ONLY Ø-3 ARE LEGAL FC ERROR IF ANGLE NUMBER > 3 SAVE DRAW ANGLE GO PROCESS ANOTHER COMMAND ONLY Ø-63 ARE LEGAL FC ERROR IF SCALE > 63 SAVE DRAW SCALE GO PROCESS ANOTHER COMMAND *NEGATE ACCD - MAKE HORIZONTAL *DIFFERENCE NEGATIVE SKIP ONE BYTE - KEEP HORIZONTAL DIFFERENCE NEGATIVE CLEAR MS BYTE OF HORIZONTAL DIFFERENCE COPY HORIZONTAL DIFFERENCE
3996 3997 3998 3999 4000 4001 4002 4003 4004 4005 4006 4007 4008 4010 4011 4011 4012 4013 4014 4017 4016 4017 4018 4019 4019 4019 4019 4019 4019 4019 4019	9052 D7 B2 9054 B0 95 9A 9057 20 84 9059 C1 04 9058 24 EF 9050 D7 E8 9057 20 F6 9061 C1 3F 9063 24 E7 9065 D7 E9 9067 20 EE 9069 4F 9068 8D 58 906C 21 9060 4F 9060 4F 9060 4F 9060 4F 9060 59 9070 20 59	L9D4F JSR L955D STB FORCOL JSR L959A L9D57 BRA L9CDD * CHANGE ANGLE L9D59 CMPB #\$04 BCC L9D4C STB ANGLE BRA L9D57 * CHANGE SCALE L9D61 CMPB #63 BCC L9D4C STB SCALE L9D61 CMPB #63 BCC L9D4C STB SCALE L9D61 CMPB #63 BCC L9D4C STB SCALE L9D61 CMPB #63 STB SCALE BRA L9D57 * 315 DEGREES L9D69 CLRA BSR 19DC4 L9D6C FCB SKP1 * 135 DEGREES L9D6D CLRA TFR D,X BRA L9DCB * 45 DEGREES L9D72 CLRA	SAVE NEW FOREGROUND COLOR SET COLOR BYTES (WCOLOR, ALLCOL) GO PROCESS ANOTHER COMMAND ONLY Ø-3 ARE LEGAL FC ERROR IF ANGLE NUMBER > 3 SAVE DRAW ANGLE GO PROCESS ANOTHER COMMAND ONLY Ø-63 ARE LEGAL FC ERROR IF SCALE > 63 SAVE DRAW SCALE GO PROCESS ANOTHER COMMAND *NEGATE ACCD - MAKE HORIZONTAL *DIFFERENCE NEGATIVE SKIP ONE BYTE - KEEP HORIZONTAL DIFFERENCE NEGATIVE CLEAR MS BYTE OF HORIZONTAL DIFFERENCE GO MOVE THE DRAW POSITION CLEAR MS BYTE OF HORIZONTAL DIFFERENCE
3996 3997 3998 3999 4000 4001 4002 4003 4004 4005 4006 4007 4008 4007 4010 4011 4011 4011 4011 4011 4011	9052 D7 B2 9054 B0 95 9A 9057 20 84 9059 C1 04 9058 24 EF 9050 D7 E8 9057 20 F6 9061 C1 3F 9063 24 E7 9065 D7 E9 9067 20 EE 9069 4F 9060 4F 9060 21 9060 4F 9060 4F 9060 4F 9060 4F 9060 1F 01 9070 20 59	L9D4F JSR L955D STB FORCOL JSR L959A L9557 BRA L9CDD * CHANGE ANGLE L9D59 CMPB #\$04 BCC L9D4C STB ANGLE L9D61 CMPB #63 BCC L9D4C STB SCALE L9D61 CMPB #63 BCC L9D4C STB SCALE L9D67 SSR L9D67 * 315 DEGREES L9D60 CLRA BSR L9D64 FCB SKP1 * 135 DEGREES L9D60 CLRA TFR D,X BRA L9DCB * 45 DEGREES L9D72 CLRA TFR D,X TFR D,X TFR D,X	SAVE NEW FOREGROUND COLOR SET COLOR SYTES (WCOLOR, ALLCOL) GO PROCESS ANOTHER COMMAND ONLY Ø-3 ARE LEGAL FC ERROR IF ANGLE NUMBER > 3 SAVE DRAW ANGLE GO PROCESS ANOTHER COMMAND ONLY Ø-63 ARE LEGAL FC ERROR IF SCALE > 63 SAVE DRAW SCALE GO PROCESS ANOTHER COMMAND *NEGATE ACCD - MAKE HORIZONTAL *DIFFERENCE NEGATIVE SKIP ONE BYTE - KEEP HORIZONTAL DIFFERENCE NEGATIVE CLEAR MS BYTE OF HORIZONTAL DIFFERENCE GO MOVE THE DRAW POSITION CLEAR MS BYTE OF HORIZONTAL DIFFERENCE COPY HORIZONTAL DIFFERENCE COPY HORIZONTAL DIFFERENCE
3996 3997 3998 3999 4000 4001 4002 4003 4004 4005 4006 4007 4008 4010 4011 4011 4015 4016 4017 4018 4019 4019 4019 4019 4019 4022 4022 4023 4024	9052 D7 B2 9054 B0 95 9A 9057 20 84 9059 C1 04 9058 24 EF 9050 D7 E8 9057 20 F6 9061 C1 3F 9063 24 E7 9065 D7 E9 9067 20 EE 9069 4F 9068 8D 58 906C 21 9060 4F 9068 1F 01 9070 20 59 9072 4F 9073 1F 01 9075 8D 4D	L9D4F JSR L955D STB FORCOL JSR L959A L9557 BRA L9CDD * CHANGE ANGLE L9D59 CMPB #\$04 BCC L9D4C STB ANGLE L9D61 CMPB #63 BCC L9D4C STB SCALE L9D60 CMPB #63 L9D67 CLRA L9D67 CLRA L9D68 FCB SKP1 * 135 DEGREES L9D60 CLRA TFR D,X BRA L9DCB * 45 DEGREES L9D60 CLRA TFR D,X BRA L9DCB	SAVE NEW FOREGROUND COLOR SET COLOR BYTES (WCOLOR, ALLCOL) GO PROCESS ANOTHER COMMAND ONLY Ø-3 ARE LEGAL FC ERROR IF ANGLE NUMBER > 3 SAVE DRAW ANGLE GO PROCESS ANOTHER COMMAND ONLY Ø-63 ARE LEGAL FC ERROR IF SCALE > 63 SAVE DRAW SCALE GO PROCESS ANOTHER COMMAND *NEGATE ACCD - MAKE HORIZONTAL *DIFFERENCE NEGATIVE SKIP ONE BYTE - KEEP HORIZONTAL DIFFERENCE NEGATIVE CLEAR MS BYTE OF HORIZONTAL DIFFERENCE COPY HORIZONTAL DIFFERENCE GO MOVE THE DRAW POSITION CLEAR MS BYTE OF HORIZONTAL DIFFERENCE COPY HORIZONTAL DIFFERENCE NEGATE ACCD - MAKE HORIZONTAL DIFFERENCE NEGATE ACCD - MAKE HORIZONTAL DIFFERENCE NEGATE ACCD - MAKE HORIZONTAL DIFFERENCE
3996 3997 3998 3999 4000 4001 4002 4003 4004 4005 4006 4007 4008 4010 4011 4011 4012 4013 4014 4017 4016 4017 4018 4019 4020 4020 4022 4023 4025	9052 D7 B2 9054 B0 95 9A 9057 20 84 9059 C1 04 9058 24 EF 9050 D7 E8 9057 20 F6 9061 C1 3F 9063 24 E7 9065 D7 E9 9067 20 EE 9069 4F 9068 8D 58 906C 21 9060 4F 9066 1F 01 9070 20 59 9072 4F 9073 1F 01 9077 1E 01	L9D4F JSR L955D STB FORCOL JSR L959A L9D57 BRA L9CDD * CHANGE ANGLE L9D59 CMPB #\$04 BCC L9D4C STB ANGLE L9D61 CMPB #63 BCC L9D4C STB SCALE BRA L9D57 * 315 DEGREES L9D69 CLRA BSR L9DC4 L9D6C FCB SKP1 * 135 DEGREES L9D6D CLRA TFR D,X BRA L9DCB * 45 DEGREES L9D72 CLRA TFR D,X BSR L9DCB L9D72 CLRA TFR D,X BSR L9DCB L9D674 EXG D,X	SAVE NEW FOREGROUND COLOR SET COLOR BYTES (WCOLOR, ALLCOL) GO PROCESS ANOTHER COMMAND ONLY Ø-3 ARE LEGAL FC ERROR IF ANGLE NUMBER > 3 SAVE DRAW ANGLE GO PROCESS ANOTHER COMMAND ONLY Ø-63 ARE LEGAL FC ERROR IF SCALE > 63 SAVE DRAW SCALE GO PROCESS ANOTHER COMMAND *NEGATE ACCD - MAKE HORIZONTAL *DIFFERENCE NEGATIVE SKIP ONE BYTE - KEEP HORIZONTAL DIFFERENCE NEGATIVE CLEAR MS BYTE OF HORIZONTAL DIFFERENCE COPY HORIZONTAL DIFFERENCE GO MOVE THE DRAW POSITION CLEAR MS BYTE OF HORIZONTAL DIFFERENCE COPY HORIZONTAL DIFFERENCE COPY HORIZONTAL DIFFERENCE COPY HORIZONTAL DIFFERENCE COPY HORIZONTAL DIFFERENCE NEGATE ACCD - MAKE HORIZONTAL DIFFERENCE EXCHANGE HORIZONTAL AND VERTICAL DIFFERENCE
3996 3997 3998 3999 4000 4001 4002 4003 4006 4007 4006 4007 4010 4011 4012 4013 4014 4015 4016 4017 4018 4017 4018 4017 4018 4017 4018 4020 4021 4023 4024 4023 4024 4026	9052 D7 B2 9054 B0 95 9A 9057 20 84 9059 C1 04 9058 24 EF 9050 D7 E8 9057 20 F6 9061 C1 3F 9063 24 E7 9065 D7 E9 9067 20 EE 9069 4F 9068 8D 58 906C 21 9060 4F 9068 1F 01 9070 20 59 9072 4F 9073 1F 01 9075 8D 4D	L9D4F JSR L955D STB FORCOL JSR L959A L9557 BRA L9CDD * CHANGE ANGLE L9D59 CMPB #\$04 BCC L9D4C STB ANGLE L9D61 CMPB #63 BCC L9D4C STB SCALE L9D61 CMPB #63 BCC L9D4C STB SCALE L9D67 CMPB #63 BCC L9D4C STB SCALE L9D60 FR SCALE L9D67 CLRA BRA L9DC4 L9D6C FCB SKP1 * 135 DEGREES L9D6D CLRA TFR D,X BRA L9DCB * 45 DEGREES L9D72 CLRA TFR D,X BSR L9DC6 STB D,X BSR L9DC6	SAVE NEW FOREGROUND COLOR SET COLOR BYTES (WCOLOR, ALLCOL) GO PROCESS ANOTHER COMMAND ONLY Ø-3 ARE LEGAL FC ERROR IF ANGLE NUMBER > 3 SAVE DRAW ANGLE GO PROCESS ANOTHER COMMAND ONLY Ø-63 ARE LEGAL FC ERROR IF SCALE > 63 SAVE DRAW SCALE GO PROCESS ANOTHER COMMAND *NEGATE ACCD - MAKE HORIZONTAL *DIFFERENCE NEGATIVE SKIP ONE BYTE - KEEP HORIZONTAL DIFFERENCE NEGATIVE CLEAR MS BYTE OF HORIZONTAL DIFFERENCE COPY HORIZONTAL DIFFERENCE GO MOVE THE DRAW POSITION CLEAR MS BYTE OF HORIZONTAL DIFFERENCE COPY HORIZONTAL DIFFERENCE NEGATE ACCD - MAKE HORIZONTAL DIFFERENCE NEGATE ACCD - MAKE HORIZONTAL DIFFERENCE NEGATE ACCD - MAKE HORIZONTAL DIFFERENCE
3996 3997 3998 3999 4000 4001 4002 4003 4004 4005 4006 4007 4008 4010 4011 4011 4012 4013 4014 4017 4016 4017 4018 4019 4020 4020 4022 4023 4025	9052 D7 B2 9054 B0 95 9A 9057 20 84 9059 C1 04 9058 24 EF 9050 D7 E8 9057 20 F6 9061 C1 3F 9063 24 E7 9065 D7 E9 9067 20 EE 9069 4F 9068 8D 58 906C 21 9060 4F 9066 1F 01 9070 20 59 9072 4F 9073 1F 01 9077 1E 01	L9D4F JSR L955D STB FORCOL JSR L959A L9D57 BRA L9CDD * CHANGE ANGLE L9D59 CMPB #\$04 BCC L9D4C STB ANGLE L9D61 CMPB #63 BCC L9D4C STB SCALE BRA L9D57 * 315 DEGREES L9D69 CLRA BSR L9DC4 L9D6C FCB SKP1 * 135 DEGREES L9D6D CLRA TFR D,X BRA L9DCB * 45 DEGREES L9D72 CLRA TFR D,X BSR L9DCB L9D72 CLRA TFR D,X BSR L9DCB L9D674 EXG D,X	SAVE NEW FOREGROUND COLOR SET COLOR BYTES (WCOLOR, ALLCOL) GO PROCESS ANOTHER COMMAND ONLY Ø-3 ARE LEGAL FC ERROR IF ANGLE NUMBER > 3 SAVE DRAW ANGLE GO PROCESS ANOTHER COMMAND ONLY Ø-63 ARE LEGAL FC ERROR IF SCALE > 63 SAVE DRAW SCALE GO PROCESS ANOTHER COMMAND *NEGATE ACCD - MAKE HORIZONTAL *DIFFERENCE NEGATIVE SKIP ONE BYTE - KEEP HORIZONTAL DIFFERENCE NEGATIVE CLEAR MS BYTE OF HORIZONTAL DIFFERENCE COPY HORIZONTAL DIFFERENCE GO MOVE THE DRAW POSITION CLEAR MS BYTE OF HORIZONTAL DIFFERENCE COPY HORIZONTAL DIFFERENCE COPY HORIZONTAL DIFFERENCE COPY HORIZONTAL DIFFERENCE COPY HORIZONTAL DIFFERENCE NEGATE ACCD - MAKE HORIZONTAL DIFFERENCE EXCHANGE HORIZONTAL AND VERTICAL DIFFERENCE
3996 3997 3998 3999 4000 4001 4002 4003 4006 4007 4006 4007 4010 4011 4012 4013 4014 4015 4016 4017 4018 4017 4018 4017 4018 4017 4018 4020 4021 4023 4024 4023 4024 4026	9052 D7 B2 9054 B0 95 9A 9057 20 84 9059 C1 04 9058 24 EF 9050 D7 E8 9057 20 F6 9061 C1 3F 9063 24 E7 9065 D7 E9 9067 20 EE 9069 4F 9068 8D 58 906C 21 9060 4F 9066 1F 01 9070 20 59 9072 4F 9073 1F 01 9077 1E 01	L9D4F JSR L955D STB FORCOL JSR L959A L9557 BRA L9CDD * CHANGE ANGLE L9D59 CMPB #\$04 BCC L9D4C STB ANGLE L9D61 CMPB #63 BCC L9D4C STB SCALE L9D61 CMPB #63 BCC L9D4C STB SCALE L9D67 CMPB #63 BCC L9D4C STB SCALE L9D60 FR SCALE L9D67 CLRA BRA L9DC4 L9D6C FCB SKP1 * 135 DEGREES L9D6D CLRA TFR D,X BRA L9DCB * 45 DEGREES L9D72 CLRA TFR D,X BSR L9DC6 STB D,X BSR L9DC6	SAVE NEW FOREGROUND COLOR SET COLOR BYTES (WCOLOR, ALLCOL) GO PROCESS ANOTHER COMMAND ONLY Ø-3 ARE LEGAL FC ERROR IF ANGLE NUMBER > 3 SAVE DRAW ANGLE GO PROCESS ANOTHER COMMAND ONLY Ø-63 ARE LEGAL FC ERROR IF SCALE > 63 SAVE DRAW SCALE GO PROCESS ANOTHER COMMAND *NEGATE ACCD - MAKE HORIZONTAL *DIFFERENCE NEGATIVE SKIP ONE BYTE - KEEP HORIZONTAL DIFFERENCE NEGATIVE CLEAR MS BYTE OF HORIZONTAL DIFFERENCE COPY HORIZONTAL DIFFERENCE GO MOVE THE DRAW POSITION CLEAR MS BYTE OF HORIZONTAL DIFFERENCE COPY HORIZONTAL DIFFERENCE COPY HORIZONTAL DIFFERENCE COPY HORIZONTAL DIFFERENCE COPY HORIZONTAL DIFFERENCE NEGATE ACCD - MAKE HORIZONTAL DIFFERENCE EXCHANGE HORIZONTAL AND VERTICAL DIFFERENCE
3996 3997 3998 3999 4000 4001 4002 4003 4006 4007 4008 4007 4008 4010 4011 4012 4013 4014 4017 4016 4017 4019 4020 4020 4020 4022 4023 4026 4026 4026 4028	9052 D7 B2 9054 B0 95 9A 9057 20 84 9059 C1 04 9058 24 EF 9050 D7 E8 9057 20 F6 9061 C1 3F 9063 24 E7 9065 D7 E9 9067 20 EE 9069 4F 9068 8D 58 906C 21 9060 4F 9060 4F 9060 4F 9060 4F 9060 59 9072 4F 9073 1F 01 9073 1F 01 9075 8D 4D 9077 1E 01 9079 20 50 9078 4F	L9D4F JSR L955D STB FORCOL JSR L959A L9557 BRA L9CDD * CHANGE ANGLE L9D59 CMPB #\$04 BCC L9D4C STB ANGLE L9D61 CMPB #63 BCC L9D4C STB SCALE L9D62 CLRA BSR L9D65 * 315 DEGREES L9D60 CLRA BSR L9D64 SKP1 * 135 DEGREES L9D60 CLRA TFR D,X BRA L9DCB * 45 DEGREES L9D72 CLRA TFR D,X BSR L9DC4 EXG D,X BRA L9DC4 EXG D,X BRA L9DC6 * 225 DEGREES L9D7B CLRA * 225 DEGREES L9D7B CLRA * 225 DEGREES L9D7B CLRA	SAVE NEW FOREGROUND COLOR SET COLOR BYTES (WCOLOR, ALLCOL) GO PROCESS ANOTHER COMMAND ONLY Ø-3 ARE LEGAL FC ERROR IF ANGLE NUMBER > 3 SAVE DRAW ANGLE GO PROCESS ANOTHER COMMAND ONLY Ø-63 ARE LEGAL FC ERROR IF SCALE > 63 SAVE DRAW SCALE GO PROCESS ANOTHER COMMAND *NEGATE ACCD - MAKE HORIZONTAL *DIFFERENCE NEGATIVE SKIP ONE BYTE - KEEP HORIZONTAL DIFFERENCE NEGATIVE CLEAR MS BYTE OF HORIZONTAL DIFFERENCE COPY HORIZONTAL DIFFERENCE GO MOVE THE DRAW POSITION CLEAR MS BYTE OF HORIZONTAL DIFFERENCE COPY HORIZONTAL DIFFERENCE COPY HORIZONTAL DIFFERENCE COPY HORIZONTAL DIFFERENCE OF WORLD THE DRAW POSITION CLEAR MS BYTE OF HORIZONTAL DIFFERENCE NEGATE ACCD - MAKE HORIZONTAL DIFFERENCE OMOVE THE DRAW POSITION CLEAR MS BYTE OF HORIZONTAL DIFFERENCE
3996 33997 33998 3999 40001 4002 4003 4004 4006 4007 4006 4007 4008 4010 4011 4012 4013 4014 4015 4016 4017 4018 4017 4018 4017 4018 4020 4021 4022 4023 4024 4024 4027 4026 4027 4029	9052 D7 B2 9054 B0 95 9A 9057 20 84 9059 C1 04 9058 C1 04 9050 D7 E8 9050 D7 E8 9051 C1 3F 9063 24 E7 9065 D7 E9 9067 20 EE 9069 4F 9068 BD 58 906C 21 9060 4F 9060 4F 9060 59 9072 4F 9073 1F 01 9075 8D 4D 9077 1E 01 9079 20 50 9078 4F 9076 1F 01	L9D4F JSR L955D STB FORCOL JSR L959A L9D57 BRA L9CDD * CHANGE ANGLE L9D59 CMPB #\$04 BCC L9D4C STB ANGLE L9D61 CMPB #63 BCC L9D4C STB SCALE L9D60 CMPB #63 BCC L9D4C STB SCALE L9D57 * 315 DEGREES L9D60 CLRA TFR D,X BSR L9DCB * 45 DEGREES L9D72 CLRA TFR D,X BSR L9DCB * 45 DEGREES L9D72 CLRA TFR D,X BSR L9DCB * 225 DEGREES L9D7B CLRA TFR D,X BRA L9DCB * 225 DEGREES L9D7B CLRA TFR D,X BRA L9DCB	SAVE NEW FOREGROUND COLOR SET COLOR BYTES (WCOLOR, ALLCOL) GO PROCESS ANOTHER COMMAND ONLY Ø-3 ARE LEGAL FC ERROR IF ANGLE NUMBER > 3 SAVE DRAW ANGLE GO PROCESS ANOTHER COMMAND ONLY Ø-63 ARE LEGAL FC ERROR IF SCALE > 63 SAVE DRAW SCALE GO PROCESS ANOTHER COMMAND *NEGATE ACCD - MAKE HORIZONTAL *DIFFERENCE NEGATIVE SKIP ONE BYTE - KEEP HORIZONTAL DIFFERENCE COPY HORIZONTAL DIFFERENCE TO VERTICAL DIFFERENCE GO MOVE THE DRAW POSITION CLEAR MS BYTE OF HORIZONTAL DIFFERENCE COPY HORIZONTAL DIFFERENCE TO VERTICAL DIFFERENCE NEGATE ACCD - MAKE HORIZONTAL DIFFERENCE NEGATIVE EXCHANGE HORIZONTAL AND VERTICAL DIFFERENCE GO MOVE THE DRAW POSITION CLEAR MS BYTE OF HORIZONTAL DIFFERENCE COPY HORIZONTAL DIFFERENCE COPY HORIZONTAL DIFFERENCE
3996 3998 3998 4000 4001 4002 4003 4004 4005 4006 4007 4009 4010 4011 4011 4015 4016 4017 4018 4019 4021 4022 4023 4024 4025 4026 4027 4028 4028 4028 4028	9052 D7 B2 9054 B0 95 9A 9057 20 84 9059 C1 04 9058 24 EF 9050 D7 E8 9057 20 F6 9061 C1 3F 9063 24 E7 9065 D7 E9 9067 20 EE 9069 4F 9068 BD 58 906C 21 9060 4F 9068 1F 01 9070 20 59 9072 4F 9073 1F 01 9075 80 4D 9077 1E 01 9077 20 50 9078 4F 9076 1F 01 9077 20 50	L9D4F JSR L955D STB FORCOL JSR L959A L9557 BRA L9CDD * CHANGE ANGLE L9D59 CMPB #\$04 BCC L9D4C STB ANGLE L9D61 CMPB #63 BCC L9D4C STB SCALE L9D61 CMPB #63 BCC L9D4C STB SCALE L9D61 CMPB #63 BCC L9D4C STB SCALE L9D60 CLRA BSR L9D57 * 315 DEGREES L9D60 CLRA TFR D,X BRA L9DCB * 45 DEGREES L9D6D CLRA TFR D,X BRA L9DCB * 45 DEGREES L9D6B CLRA TFR D,X BRA L9DCB * 45 DEGREES L9D72 CLRA TFR D,X BRA L9DCB * 225 DEGREES L9DCB * 26 DCC * 19DCB * 27 DCC * 19DCB * 28 DCC * 19DCB * 19DCB * 28 DCC * 19DCB * 19DCB * 28 DCC * 19DCB * 19DCB * 28 DCC * 19DCB * 19DCB * 19DCB	SAVE NEW FOREGROUND COLOR SET COLOR BYTES (WCOLOR, ALLCOL) GO PROCESS ANOTHER COMMAND ONLY Ø-3 ARE LEGAL FC ERROR IF ANGLE NUMBER > 3 SAVE DRAW ANGLE GO PROCESS ANOTHER COMMAND ONLY Ø-63 ARE LEGAL FC ERROR IF SCALE > 63 SAVE DRAW SCALE GO PROCESS ANOTHER COMMAND *NEGATE ACCD - MAKE HORIZONTAL *DIFFERENCE NEGATIVE SKIP ONE BYTE - KEEP HORIZONTAL DIFFERENCE NEGATIVE CLEAR MS BYTE OF HORIZONTAL DIFFERENCE COPY HORIZONTAL DIFFERENCE GO MOVE THE DRAW POSITION CLEAR MS BYTE OF HORIZONTAL DIFFERENCE COPY HORIZONTAL DIFFERENCE OF MAKE HORIZONTAL DIFFERENCE SECHANGE HORIZONTAL AND VERTICAL DIFFERENCE OF MOVE THE DRAW POSITION CLEAR MS BYTE OF HORIZONTAL DIFFERENCE COPY HORIZONTAL DIFFERENCE
3996 3997 3998 3999 4000 4001 4002 4003 4004 4005 4006 4007 4008 4010 4011 4012 4013 4014 4017 4016 4017 4019 4020 4020 4020 4020 4020 4020 4022 4023 4025 4026 4027 4026 4027 4028 4029 4031	9052 D7 B2 9054 B0 95 9A 9057 20 84 9059 C1 04 9058 C1 04 9050 D7 E8 9050 D7 E8 9051 C1 3F 9063 24 E7 9065 D7 E9 9067 20 EE 9069 4F 9068 BD 58 906C 21 9060 4F 9060 4F 9060 59 9072 4F 9073 1F 01 9075 8D 4D 9077 1E 01 9079 20 50 9078 4F 9076 1F 01	L9D4F JSR L955D STB FORCOL JSR L959A L9557 BRA L9CDD * CHANGE ANGLE L9D59 CMPB #\$04 BCC L9D4C STB ANGLE L9D61 CMPB #63 BCC L9D4C STB SCALE L9D62 CMPB #63 BCC L9D4C STB SCALE L9D67 CMPB #63 BCC L9D4C STB SCALE L9D67 CMPB #63 BCC L9D4C STB SCALE L9D67 CMPB #63 BCC L9D4C STB SCALE L9D60 LPA TFR D,X BCA L9DCB * 45 DEGREES L9D72 CLRA TFR D,X BCA L9DCB * 225 DEGREES L9D78 CLRA TFR D,X BCA L9DCB * 225 DEGREES L9D78 CLRA TFR D,X BCA L9DCB * 225 DEGREES L9D78 CLRA TFR D,X BCA L9DCB * 225 DEGREES L9D78 CLRA TFR D,X BCA L9DCB * 225 DEGREES L9D78 CLRA TFR D,X BCA L9DCB * 225 DEGREES L9D78 CLRA TFR D,X BCA L9DCB	SAVE NEW FOREGROUND COLOR SET COLOR BYTES (WCOLOR, ALLCOL) GO PROCESS ANOTHER COMMAND ONLY Ø-3 ARE LEGAL FC ERROR IF ANGLE NUMBER > 3 SAVE DRAW ANGLE GO PROCESS ANOTHER COMMAND ONLY Ø-63 ARE LEGAL FC ERROR IF SCALE > 63 SAVE DRAW SCALE GO PROCESS ANOTHER COMMAND *NEGATE ACCD - MAKE HORIZONTAL *DIFFERENCE NEGATIVE SKIP ONE BYTE - KEEP HORIZONTAL DIFFERENCE COPY HORIZONTAL DIFFERENCE TO VERTICAL DIFFERENCE GO MOVE THE DRAW POSITION CLEAR MS BYTE OF HORIZONTAL DIFFERENCE COPY HORIZONTAL DIFFERENCE TO VERTICAL DIFFERENCE NEGATE ACCD - MAKE HORIZONTAL DIFFERENCE NEGATIVE EXCHANGE HORIZONTAL AND VERTICAL DIFFERENCE GO MOVE THE DRAW POSITION CLEAR MS BYTE OF HORIZONTAL DIFFERENCE COPY HORIZONTAL DIFFERENCE COPY HORIZONTAL DIFFERENCE
3996 3998 3998 4000 4001 4002 4003 4004 4005 4006 4007 4009 4010 4011 4011 4015 4016 4017 4018 4019 4021 4022 4023 4024 4025 4026 4027 4028 4028 4028 4028	9052 D7 B2 9054 B0 95 9A 9057 20 84 9059 C1 04 9058 24 EF 9050 D7 E8 9057 20 F6 9061 C1 3F 9063 24 E7 9065 D7 E9 9067 20 EE 9069 4F 9068 BD 58 906C 21 9060 4F 9068 1F 01 9070 20 59 9072 4F 9073 1F 01 9075 80 4D 9077 1E 01 9077 20 50 9078 4F 9076 1F 01 9077 20 50	L9D4F JSR L955D STB FORCOL JSR L959A L9557 BRA L9CDD * CHANGE ANGLE L9D59 CMPB #\$04 BCC L9D4C STB ANGLE L9D61 CMPB #63 BCC L9D4C STB SCALE L9D61 CMPB #63 BCC L9D4C STB SCALE L9D61 CMPB #63 BCC L9D4C STB SCALE L9D60 CLRA BSR L9D57 * 315 DEGREES L9D60 CLRA TFR D,X BRA L9DCB * 45 DEGREES L9D6D CLRA TFR D,X BRA L9DCB * 45 DEGREES L9D6B CLRA TFR D,X BRA L9DCB * 45 DEGREES L9D72 CLRA TFR D,X BRA L9DCB * 225 DEGREES L9DCB * 26 DCC * 19DCB * 27 DCC * 19DCB * 28 DCC * 19DCB * 19DCB * 28 DCC * 19DCB * 19DCB * 28 DCC * 19DCB * 19DCB * 28 DCC * 19DCB * 19DCB * 19DCB	SAVE NEW FOREGROUND COLOR SET COLOR BYTES (WCOLOR, ALLCOL) GO PROCESS ANOTHER COMMAND ONLY Ø-3 ARE LEGAL FC ERROR IF ANGLE NUMBER > 3 SAVE DRAW ANGLE GO PROCESS ANOTHER COMMAND ONLY Ø-63 ARE LEGAL FC ERROR IF SCALE > 63 SAVE DRAW SCALE GO PROCESS ANOTHER COMMAND *NEGATE ACCD - MAKE HORIZONTAL *DIFFERENCE NEGATIVE SKIP ONE BYTE - KEEP HORIZONTAL DIFFERENCE NEGATIVE CLEAR MS BYTE OF HORIZONTAL DIFFERENCE COPY HORIZONTAL DIFFERENCE GO MOVE THE DRAW POSITION CLEAR MS BYTE OF HORIZONTAL DIFFERENCE COPY HORIZONTAL DIFFERENCE OF MAKE HORIZONTAL DIFFERENCE SECHANGE HORIZONTAL AND VERTICAL DIFFERENCE OF MOVE THE DRAW POSITION CLEAR MS BYTE OF HORIZONTAL DIFFERENCE COPY HORIZONTAL DIFFERENCE

4033	9D82 4F	L9D82	CLRA	CLEAR MS BYTE OF HORIZONTAL DIFFERENCE
4034	9D83 9E 8A	L9D83	LDX ZERO	$X = \emptyset$; VERT DIFF = \emptyset
4035	9D85 2Ø 44	* GO LE	BRA L9DCB	GO MOVE THE DRAW POSITION
4036 4037	9D87 4F	L9D87	CLRA	*NEGATE ACCD - MAKE THE HORIZONTAL
4038	9D88 8D 3A	23007	BSR L9DC4	*DIFFERENCE NEGATIVE
4039	9D8A 2Ø F7		BRA L9D83	MAKE VERTICAL DIFFERENCE ZERO AND MOVE THE DRAW POSITION
4040		* GO DO		
4041	9D8C 4F	L9D8C	CLRA	CLEAR MS BYTE OF HORIZONTAL DIFFERENCE
4042 4043	9D8D 20 03	*	BRA L9D92	*MAKE VERTICAL DIFFERENCE = Ø, EXCHANGE HORIZONTAL AND *VERTICAL DIFFERENCES AND MOVE THE DRAW POSTION
4043		* GO UP		*VERTICAL DIFFERENCES AND MOVE THE DRAW POSITON
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	$X = \emptyset$; HORZ DIFF = \emptyset
4048	9D94 1E 1Ø		EXG X,D	EXCHANGE THE HORIZONTAL AND VERTICAL DIFFERENCES
4049 4050	9D96 2Ø 33	* EAEUII.	BRA L9DCB TE A COMMAND SUB STRING	GO MOVE THE DRAW POSITION
4051	9D98 BD 9C 1B	L9D98	JSR L9C1B	INTERPRET CURRENT COMMAND AS IF IT WERE A BASIC VARIABLE
4052	9D9B C6 Ø2		LDB #\$Ø2	=
4Ø53	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
4Ø56 4Ø57	9DA4 34 14 9DA6 7E 9C C6		PSHS X,B JMP L9CC6	* AND SAVE THEM ON THE STACK EVALUATE NUMERICAL VALUE IN COMMAND LINE
4058	30NO 7E 30 00	* MULTI	PLY HOR OR VER DIFFERENCE BY SCALE FAC	
4059			E PRODUCT BY 4 AND RETURN VALUE IN ACC	
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 4063	9DAD 4F 9DAE 1E Ø1		CLRA EXG D,X	CLEAR MS BYTE 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 4070	9DBB 44 9DBC 56		LSRA RORB	*
4070	9DBD 44	L9DBD	LSRA	*
4072	9DBE 56	23000	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	0000 40		E ACCUMULATOR D	
4076 4077	9DC3 4Ø 9DC4 5Ø	L9DC3 L9DC4	NEGA NEGB	*
4077	9DC5 82 ØØ	L3004	SBCA #\$00	* 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			THE DRAW POSITION - ADD THE ORTHOGONAL	DIFFERENCES
4Ø83 4Ø84			CD (HORIZONTAL) AND X (VERTICAL) TO URRENT 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
4Ø87	9DCF 35 10		PULS X	GET HORIZ DIFFERENCE
4088	9DD1 34 Ø6		PSHS B,A	SAVE VERT DIFFERENCE
4Ø89 4Ø9Ø	9DD3 8D D4 9DD5 35 10		BSR L9DA9 PULS X	APPLY SCALE FACTOR TO HORIZONTAL GET VERT DIFFERENCE
4091	9DD7 10 9E E8		LDY ANGLE	* GET THE DRAW ANGLE AND SCALE AND SAVE THEM ON
4092	9DDA 34 20		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 4097	9DE2 8D DF 9DE4 6A E4		BSR L9DC3 DEC ,S	NEGATE ACCD DECR ANGLE
	9DE6 20 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
	9DEE 2B Ø2		BMI L9DF2	HORIZ COORD = Ø IF RESULT IS NEG
	9DEE 2B Ø2 9DFØ 1F Ø3	L9DF2	BMI L9DF2 TFR D,U	HORIZ COORD = Ø IF RESULT IS NEG SAVE HOR END POSITION IN U
4100	9DEE 2B Ø2	L9DF2	BMI L9DF2	HORIZ COORD = Ø IF RESULT IS NEG
4106	9DEE 2B Ø2 9DFØ 1F Ø3 9DF2 1F 1Ø	L9DF2	BMI L9DF2 TFR D,U TFR X,D	HORIZ COORD = Ø IF RESULT IS NEG SAVE HOR END POSITION IN U PUT VERT DIFFERENCE IN ACCD
4107	9DEE 2B 02 9DF0 1F 03 9DF2 1F 10 9DF4 9E 8A 9DF6 D3 C9 9DF8 2B 02	L9DF2	BMI L9DF2 TFR D,U TFR X,D LDX ZERO ADDD VERDEF BMI L9DFC	HORIZ COORD = Ø IF RESULT IS NEG SAVE HOR END POSITION IN U PUT VERT DIFFERENCE IN ACCD X = Ø; DEFAULT VER END POSITION = Ø ADD DIFFERENCE TO VER START VER COORD = Ø IF RESULT IS NEG
4107 4108	9DEE 2B 02 9DF0 1F 03 9DF2 1F 10 9DF4 9E 8A 9DF6 D3 C9	L9DF2	BMI L9DF2 TFR D,U TFR X,D LDX ZERO ADDD VERDEF	HORIZ COORD = Ø IF RESULT IS NEG SAVE HOR END POSITION IN U PUT VERT DIFFERENCE IN ACCD X = Ø; DEFAULT VER END POSITION = Ø ADD DIFFERENCE TO VER START
4107 4108 4109	9DEE 2B 02 9DF0 1F 03 9DF2 1F 10 9DF4 9E 8A 9DF6 D3 C9 9DF8 2B 02		BMI L9DF2 TFR D,U TFR X,D LDX ZERO ADDD VERDEF BMI L9DFC TFR D,X	HORIZ COORD = Ø IF RESULT IS NEG SAVE HOR END POSITION IN U PUT VERT DIFFERENCE IN ACCD X = Ø; DEFAULT VER END POSITION = Ø ADD DIFFERENCE TO VER START VER COORD = Ø IF RESULT IS NEG SAVE VERT END POSITION IN X
4107 4108	9DEE 2B 02 9DF0 1F 03 9DF2 1F 10 9DF4 9E 8A 9DF6 D3 C9 9DF8 2B 02	* MOVE	BMI L9DF2 TFR D,U TFR X,D LDX ZERO ADDD VERDEF BMI L9DFC	HORIZ COORD = Ø IF RESULT IS NEG SAVE HOR END POSITION IN U PUT VERT DIFFERENCE IN ACCD X = Ø; DEFAULT VER END POSITION = Ø ADD DIFFERENCE TO VER START VER COORD = Ø IF RESULT IS NEG SAVE VERT END POSITION IN X HORIZONTAL POSITION
4107 4108 4109 4110 4111 4112	9DEE 28 02 9DF0 1F 03 9DF2 1F 10 9DF4 9E 8A 9DF6 D3 C9 9DF8 28 02 9DFA 1F 01 9DFC 11 83 01 00	* MOVE	BMI L9DF2 TFR D,U TFR X,D LDX ZERO ADDD VERDEF BMI L9DFC TFR D,X THE DRAW POSITION; ENTER WITH ABSOLUTE REGISTER AND ABSOLUTE VERTICAL POSITIO CMPU #256	HORIZ COORD = Ø IF RESULT IS NEG SAVE HOR END POSITION IN U PUT VERT DIFFERENCE IN ACCD X = Ø; DEFAULT VER END POSITION = Ø ADD DIFFERENCE TO VER START VER COORD = Ø IF RESULT IS NEG SAVE VERT END POSITION IN X HORIZONTAL POSITION N IN X REGISTER. IS HORIZ COORD WITHIN RANGE?
4107 4108 4109 4110 4111 4112 4113	9DEE 2B 02 9DF0 1F 03 9DF2 1F 10 9DF4 9E 8A 9DF6 03 C9 9DF8 2B 02 9DFA 1F 01 9DFC 11 83 01 00 9E00 25 03	* MOVE '	BMI L9DF2 TFR D,U TFR X,D LDX ZERO ADDD VERDEF BMI L9DFC TFR D,X THE DRAW POSITION; ENTER WITH ABSOLUTE REGISTER AND ABSOLUTE VERTICAL POSITIO CMPU #256 BLO L9FØ5	HORIZ COORD = Ø IF RESULT IS NEG SAVE HOR END POSITION IN U PUT VERT DIFFERENCE IN ACCD X = Ø; DEFAULT VER END POSITION = Ø ADD DIFFERENCE TO VER START VER COORD = Ø IF RESULT IS NEG SAVE VERT END POSITION IN X HORIZONTAL POSITION N IN X REGISTER. IS HORIZ COORD WITHIN RANGE? YES
4107 4108 4109 4110 4111 4112 4113 4114	9DEE 28 02 9DF0 1F 03 9DF2 1F 10 9DF4 9E 8A 9DF6 03 C9 9DF8 1F 01 9DFC 11 83 01 00 9DFC 11 83 01 00 9DFC 25 03 9E00 25 03 9E00 CE 00 FF	* MOVE * * IN U L9DFC	BMI L9DF2 TFR D,U TFR X,D LDX ZERO ADDD VERDEF BMI L9DFC TFR D,X THE DRAW POSITION; ENTER WITH ABSOLUTE REGISTER AND ABSOLUTE VERTICAL POSITIO CMPU #256 BLO L9EØ5 LDU #255	HORIZ COORD = Ø IF RESULT IS NEG SAVE HOR END POSITION IN U PUT VERT DIFFERENCE IN ACCD X = Ø; DEFAULT VER END POSITION = Ø ADD DIFFERENCE TO VER START VER COORD = Ø IF RESULT IS NEG SAVE VERT END POSITION IN X HORIZONTAL POSITION N IN X REGISTER. IS HORIZ COORD WITHIN RANGE? YES NO - FORCE TO MAX VALUE
4107 4108 4109 4110 4111 4112 4113 4114 4115	9DEE 28 02 9DF0 1F 03 9DF2 1F 10 9DF4 9E 8A 9DF6 D3 C9 9DF8 28 02 9DFA 1F 01 9DFC 11 83 01 00 9E00 25 03 9E02 CE 00 FF 99E05 8C 00 C0	* MOVE '	BMI L9DF2 TFR D,U TFR X,D LDX ZERO ADDD VERDEF BMI L9DFC TFR D,X THE DRAW POSITION; ENTER WITH ABSOLUTE REGISTER AND ABSOLUTE VERTICAL POSITIO CMPU #256 BLO L9EØ5 LDU #255 CMPX #192	HORIZ COORD = Ø IF RESULT IS NEG SAVE HOR END POSITION IN U PUT VERT DIFFERENCE IN ACCD X = Ø; DEFAULT VER END POSITION = Ø ADD DIFFERENCE TO VER START VER COORD = Ø IF RESULT IS NEG SAVE VERT END POSITION IN X HORIZONTAL POSITION N IN X REGISTER. IS HORIZ COORD WITHIN RANGE? YES NO - FORCE TO MAX VALUE IS VERT COORD WITHIN RANGE?
4107 4108 4109 4110 4111 4112 4113 4114 4115 4116	9DEE 28 02 9DF0 1F 03 9DF2 1F 10 9DF4 9E 8A 9DF6 03 C9 9DF8 1F 01 9DFC 11 83 01 00 9DFC 11 83 01 00 9DFC 25 03 9E00 25 03 9E00 CE 00 FF	* MOVE * * IN U L9DFC	BMI L9DF2 TFR D,U TFR X,D LDX ZERO ADDD VERDEF BMI L9DFC TFR D,X THE DRAW POSITION; ENTER WITH ABSOLUTE REGISTER AND ABSOLUTE VERTICAL POSITIO CMPU #256 BLO L9EØ5 LDU #255	HORIZ COORD = Ø IF RESULT IS NEG SAVE HOR END POSITION IN U PUT VERT DIFFERENCE IN ACCD X = Ø; DEFAULT VER END POSITION = Ø ADD DIFFERENCE TO VER START VER COORD = Ø IF RESULT IS NEG SAVE VERT END POSITION IN X HORIZONTAL POSITION N IN X REGISTER. IS HORIZ COORD WITHIN RANGE? YES NO - FORCE TO MAX VALUE
4107 4108 4109 4110 4111 4112 4113 4114 4115 4116 4117	9DEE 2B 02 9DF0 1F 03 9DF2 1F 10 9DF4 9E 8A 9DF6 03 C9 9DF8 2B 02 9DFA 1F 01 9DFC 11 83 01 00 9E00 25 03 9E02 CE 00 FF 9E05 8C 00 C0 9E08 25 03	* MOVE * * IN U L9DFC	BMI L9DF2 TFR D,U TFR X,D LDX ZERO ADDD VERDEF BMI L9DFC TFR D,X THE DRAW POSITION; ENTER WITH ABSOLUTE REGISTER AND ABSOLUTE VERTICAL POSITIO CMPU #256 LDU #255 CMPX #192 BLO L9FØD	HORIZ COORD = Ø IF RESULT IS NEG SAVE HOR END POSITION IN U PUT VERT DIFFERENCE IN ACCD X = Ø; DEFAULT VER END POSITION = Ø ADD DIFFERENCE TO VER START VER COORD = Ø IF RESULT IS NEG SAVE VERT END POSITION IN X HORIZONTAL POSITION N IN X REGISTER. IS HORIZ COORD WITHIN RANGE? YES NO - FORCE TO MAX VALUE IS VERT COORD WITHIN RANGE? YES
4107 4108 4109 4110 4111 4112 4113 4114 4115 4116 4117 4118 4119	9DEE 28 02 9DF0 1F 03 9DF2 1F 10 9DF4 9E 8A 9DF6 03 C9 9DF8 28 02 9DFA 1F 01 9DFC 11 83 01 00 9E00 25 03 9E00 25 03 9E05 8C 00 C0 9E08 25 03 9E08 25 03 9E08 25 03 9E08 8C 00 BF 9E0D DC C7 9EFF DD BD	* MOVE * IN U L9DFC	BMI L9DF2 TFR D,U TFR X,D LDX ZERO ADDD VERDEF BMI L9DFC TFR D,X THE DRAW POSITION; ENTER WITH ABSOLUTE REGISTER AND ABSOLUTE VERTICAL POSITIO CMPU #255 LDU #255 CMPX #192 BLO L9FØD LDX #191 LDD HORDEF STD HORBEG	HORIZ COORD = Ø IF RESULT IS NEG SAVE HOR END POSITION IN U PUT VERT DIFFERENCE IN ACCD X = Ø; DEFAULT VER END POSITION = Ø ADD DIFFERENCE TO VER START VER COORD = Ø IF RESULT IS NEG SAVE VERT END POSITION IN X HORIZONTAL POSITION N IN X REGISTER. IS HORIZ COORD WITHIN RANGE? YES NO - FORCE TO MAX VALUE IS VERT COORD WITHIN RANGE? YES NO - FORCE TO MAX VALUE *
4107 4108 4109 4110 4111 4112 4113 4114 4115 4116 4117 4118 4119 4120	9DEE 28 02 9DF0 1F 03 9DF2 1F 10 9DF4 9E 8A 9DF6 03 C9 9DF8 1B 02 9DFA 1F 01 9DFC 11 83 01 00 9E00 25 03 9E00 25 03 9E02 CE 00 FF 9E08 8C 00 C0 9E08 8C 00 C0 9E08 8C 00 C7 9E07 DD BD 9E11 DC C9	* MOVE * IN U L9DFC	BMI L9DF2 TFR D,U TFR X,D LDX ZERO ADDD VERDEF BMI L9DFC TFR D,X THE DRAW POSITION; ENTER WITH ABSOLUTE REGISTER AND ABSOLUTE VERTICAL POSITIO CMPU #256 BLO L9E05 LDU #255 CMPX #192 BLO L9E0D LDX #191 LDD HORDEF STD HORBEG LDD VERDEF	HORIZ COORD = Ø IF RESULT IS NEG SAVE HOR END POSITION IN U PUT VERT DIFFERENCE IN ACCD X = Ø; DEFAULT VER END POSITION = Ø ADD DIFFERENCE TO VER START VER COORD = Ø IF RESULT IS NEG SAVE VERT END POSITION IN X HORIZONTAL POSITION N IN X REGISTER. IS HORIZ COORD WITHIN RANGE? YES NO - FORCE TO MAX VALUE IS VERT COORD WITHIN RANGE? YES NO - FORCE TO MAX VALUE * * * COPY THE HOR AND VER POINTERS
4107 4108 4109 4110 4111 4112 4113 4114 4115 4116 4117 4118 4119 4120 4121	9DEE 28 02 9DF0 1F 03 9DF2 1F 10 9DF4 9E 8A 9DF6 D3 C9 9DF8 28 02 9DFA 1F 01 9DFC 11 83 01 00 9E00 25 03 9E00 26 00 FF 9E05 8C 00 C0 9E08 25 03 9E0A 8E 00 BF 9E0D DC C7 9E0F DD BD 9E11 DC C9 9E13 DD BF	* MOVE * IN U L9DFC	BMI L9DF2 TFR D,U TFR X,D LDX ZERO ADDD VERDEF BMI L9DFC TFR D,X THE DRAW POSITION; ENTER WITH ABSOLUTE REGISTER AND ABSOLUTE VERTICAL POSITIO CMPU #256 BLO L9E05 LDU #255 CMPX #192 BLO L9E0D LDX #191 LDD HORDEF STD HORBEG LDD VERBEG	HORIZ COORD = Ø IF RESULT IS NEG SAVE HOR END POSITION IN U PUT VERT DIFFERENCE IN ACCD X = Ø; DEFAULT VER END POSITION = Ø ADD DIFFERENCE TO VER START VER COORD = Ø IF RESULT IS NEG SAVE VERT END POSITION IN X HORIZONTAL POSITION N IN X REGISTER. IS HORIZ COORD WITHIN RANGE? YES NO - FORCE TO MAX VALUE IS VERT COORD WITHIN RANGE? YES NO - FORCE TO MAX VALUE *
4107 4108 4109 4110 4111 4112 4113 4114 4115 4116 4117 4118 4119 4120 4121 4122	9DEE 2B 02 9DF0 1F 03 9DF2 1F 10 9DF4 9E 8A 9DF6 03 C9 9DF8 2B 02 9DFA 1F 01 9DFC 11 83 01 00 9E00 25 03 9E00 25 03 9E05 8C 00 C0 9E08 25 03 9E08 DC C7 9E10 DB D 9E11 DC C9 9E13 DD BF 9E15 9F C5	* MOVE * IN U L9DFC	BMI L9DF2 TFR D,U TFR X,D LDX ZERO ADDD VERDEF BMI L9DFC TFR D,X THE DRAW POSITION; ENTER WITH ABSOLUTE REGISTER AND ABSOLUTE VERTICAL POSITIO CMPU #256 LDU #255 LDU #255 CMPX #192 BLO L9FØD LDX #191 LDD HORDEF STD HORBEG LDD VERDEF STD VERBEG STX VEREND	HORIZ COORD = Ø IF RESULT IS NEG SAVE HOR END POSITION IN U PUT VERT DIFFERENCE IN ACCD X = Ø; DEFAULT VER END POSITION = Ø ADD DIFFERENCE TO VER START VER COORD = Ø IF RESULT IS NEG SAVE VERT END POSITION IN X HORIZONTAL POSITION N IN X REGISTER. IS HORIZ COORD WITHIN RANGE? YES NO - FORCE TO MAX VALUE IS VERT COORD WITHIN RANGE? YES NO - FORCE TO MAX VALUE * * COPY THE HOR AND VER POINTERS * INTO THE DRAW LINE START POSITION =
4107 4108 41109 4111 4111 4112 4113 4114 4115 4116 4117 4118 4119 4120 4121 4122 4123	9DEE 28 02 9DF0 1F 03 9DF2 1F 10 9DF4 9E 8A 9DF6 D3 C9 9DF8 28 02 9DFA 1F 01 9DFC 11 83 01 00 9E00 25 03 9E00 26 00 FF 9E05 8C 00 C0 9E08 25 03 9E0A 8E 00 BF 9E0D DC C7 9E0F DD BD 9E11 DC C9 9E13 DD BF	* MOVE * IN U L9DFC	BMI L9DF2 TFR D,U TFR X,D LDX ZERO ADDD VERDEF BMI L9DFC TFR D,X THE DRAW POSITION; ENTER WITH ABSOLUTE REGISTER AND ABSOLUTE VERTICAL POSITIO CMPU #256 BLO L9E05 LDU #255 CMPX #192 BLO L9E0D LDX #191 LDD HORDEF STD HORBEG LDD VERBEG	HORIZ COORD = Ø IF RESULT IS NEG SAVE HOR END POSITION IN U PUT VERT DIFFERENCE IN ACCD X = Ø; DEFAULT VER END POSITION = Ø ADD DIFFERENCE TO VER START VER COORD = Ø IF RESULT IS NEG SAVE VERT END POSITION IN X HORIZONTAL POSITION N IN X REGISTER. IS HORIZ COORD WITHIN RANGE? YES NO - FORCE TO MAX VALUE IS VERT COORD WITHIN RANGE? YES NO - FORCE TO MAX VALUE * * * COPY THE HOR AND VER POINTERS
4107 4108 4110 4111 4111 4112 4113 4114 4115 4116 4117 4118 4119 4120 4121 4122 4123 4124 4125	9DEE 28 02 9DF0 1F 03 9DF2 1F 10 9DF4 9E 8A 9DF6 03 C9 9DF8 28 02 9DFA 1F 01 9DFC 11 83 01 00 9E00 25 03 9E00 25 03 9E02 CE 00 FF 9E05 8C 00 C0 9E08 25 03 9E08 25 03 9E09 DC C7 9E19 DD BD 9E11 DC C9 9E13 DD BF 9E15 9F C5 9E17 DF C3 9E19 00 D5 9E18 26 04	* MOVE * IN U L9DFC	BMI L9DF2 TFR D,U TFR X,D LDX ZERO ADDD VERDEF BMI L9DFC TFR D,X THE DRAW POSITION; ENTER WITH ABSOLUTE REGISTER AND ABSOLUTE VERTICAL POSITIO CMPU #256 LDU #255 CMPX #192 BLO L9EØD LDX #191 LDD HORDEF STD HORBEG LDD VERDEF STD VERBEG STX VEREND STU HOREND TST VD5 BME L9E21	HORIZ COORD = Ø IF RESULT IS NEG SAVE HOR END POSITION IN U PUT VERT DIFFERENCE IN ACCD X = Ø; DEFAULT VER END POSITION = Ø ADD DIFFERENCE TO VER START VER COORD = Ø IF RESULT IS NEG SAVE VERT END POSITION IN X HORIZONTAL POSITION N IN X REGISTER. IS HORIZ COORD WITHIN RANGE? YES NO - FORCE TO MAX VALUE IS VERT COORD WITHIN RANGE? YES NO - FORCE TO MAX VALUE * * COPY THE HOR AND VER POINTERS * INTO THE DRAW LINE START POSITION = SET THE DRAW LINE END POSITION
4107 4108 4110 4111 4112 4113 4114 4115 4116 4117 4118 4120 4121 4122 4123 4124 4125 4126	9DEE 28 02 9DF0 1F 03 9DF2 1F 10 9DF4 9E 8A 9DF6 03 C9 9DFA 1F 01 9DFC 11 83 01 00 9E00 25 03 9E00 25 03 9E00 25 03 9E00 85 00 C0 9E00 85 03 9E00 BC C7 9E0F DD BD 9E11 DC C9 9E13 DD BF 9E15 9F C5 9E17 DF C3 9E19 0D D5 9E18 26 04 9E19 0D D5 9E19 0D D5 9E19 0D D5 9E19 0D D5	* MOVE * IN U L9DFC	BMI L9DF2 TFR D,U TFR X,D LDX ZERO ADDD VERDEF BMI L9DFC TFR D,X THE DRAW POSITION; ENTER WITH ABSOLUTE REGISTER AND ABSOLUTE VERTICAL POSITIO CMPU #256 LDU #255 CMPX #192 BLO L9EØ5 LDX #191 LDD HORDEF STD HORBEG LDD VERDEF STD VERBEG STX VEREND STU HOREND TST UDS BNE L9E21 STX VERDEF	HORIZ COORD = Ø IF RESULT IS NEG SAVE HOR END POSITION IN U PUT VERT DIFFERENCE IN ACCD X = Ø; DEFAULT VER END POSITION = Ø ADD DIFFERENCE TO VER START VER COORD = Ø IF RESULT IS NEG SAVE VERT END POSITION IN X HORIZONTAL POSITION N IN X REGISTER. IS HORIZ COORD WITHIN RANGE? YES NO - FORCE TO MAX VALUE IS VERT COORD WITHIN RANGE? YES NO - FORCE TO MAX VALUE * * COPY THE HOR AND VER POINTERS * INTO THE DRAW LINE START POSITION = SET THE DRAW LINE END POSITION CHECK UPDATE FLAG BRANCH IF NO UPDATE *
4107 4108 4110 4111 4112 4113 4114 4115 4116 4117 4118 4119 4121 4122 4123 4124 4125 4126 4127	9DEE 28 02 9DF0 1F 03 9DF2 1F 10 9DF4 9E 8A 9DF6 D3 C9 9DF8 28 02 9DFA 1F 01 9DFC 11 83 01 00 9E00 25 03 9E00 25 03 9E08 25 03 9E08 25 00 9E08 25 00 9E08 DC 7 9E0F DD BD 9E11 DC C9 9E13 DD BF 9E15 9F C5 9E17 DF C3 9E19 00 D5 9E18 26 04 9E10 9F C9 9E18 PC C9 9E19 DD BD	* MOVE * IN U L9DFC L9EØ5	BMI L9DF2 TFR D,U TFR X,D LDX ZERO ADDD VERDEF BMI L9DFC TFR D,X THE DRAW POSITION; ENTER WITH ABSOLUTE REGISTER AND ABSOLUTE VERTICAL POSITIO CMPU #256 LDU #255 CMPX #192 BLO L9EØ5 LDX #191 LDD HORDEF STD HORBEG LDD VERDEF STD VERBEG STX VEREND STU HOREND TST VDS BNE L9E21 STX VERDEF	HORIZ COORD = Ø IF RESULT IS NEG SAVE HOR END POSITION IN U PUT VERT DIFFERENCE IN ACCD X = Ø; DEFAULT VER END POSITION = Ø ADD DIFFERENCE TO VER START VER COORD = Ø IF RESULT IS NEG SAVE VERT END POSITION IN X HORIZONTAL POSITION N IN X REGISTER. IS HORIZ COORD WITHIN RANGE? YES NO - FORCE TO MAX VALUE IS VERT COORD WITHIN RANGE? YES NO - FORCE TO MAX VALUE * * * COPY THE HOR AND VER POINTERS * * INTO THE DRAW LINE START POSITION = SET THE DRAW LINE END POSITION CHECK UPDATE FLAG BRANCH IF NO UPDATE * * UPDATE POSITION OF DRAW POINTER
4107 4108 4110 4111 4112 4113 4114 4115 4116 4117 4118 4119 4121 4122 4123 4124 4125 4126 4127	9DEE 28 02 9DF0 1F 03 9DF2 1F 10 9DF4 9E 8A 9DF6 03 C9 9DFA 1F 01 9DFC 11 83 01 00 9E00 25 03 9E00 25 03 9E00 25 03 9E00 85 00 C0 9E00 85 03 9E00 BC C7 9E0F DD BD 9E11 DC C9 9E13 DD BF 9E15 9F C5 9E17 DF C3 9E19 0D D5 9E18 26 04 9E19 0D D5 9E19 0D D5 9E19 0D D5 9E19 0D D5	* MOVE * IN U L9DFC	BMI L9DF2 TFR D,U TFR X,D LDX ZERO ADDD VERDEF BMI L9DFC TFR D,X THE DRAW POSITION; ENTER WITH ABSOLUTE REGISTER AND ABSOLUTE VERTICAL POSITIO CMPU #256 LDU #255 CMPX #192 BLO L9EØ5 LDX #191 LDD HORDEF STD HORBEG LDD VERDEF STD VERBEG STX VEREND STU HOREND TST UDS BNE L9E21 STX VERDEF	HORIZ COORD = Ø IF RESULT IS NEG SAVE HOR END POSITION IN U PUT VERT DIFFERENCE IN ACCD X = Ø; DEFAULT VER END POSITION = Ø ADD DIFFERENCE TO VER START VER COORD = Ø IF RESULT IS NEG SAVE VERT END POSITION IN X HORIZONTAL POSITION N IN X REGISTER. IS HORIZ COORD WITHIN RANGE? YES NO - FORCE TO MAX VALUE IS VERT COORD WITHIN RANGE? YES NO - FORCE TO MAX VALUE * * COPY THE HOR AND VER POINTERS * INTO THE DRAW LINE START POSITION = SET THE DRAW LINE END POSITION CHECK UPDATE FLAG BRANCH IF NO UPDATE *

4129	9E24 ØD D6		TST	VD6	GET DRAW FLAG
413Ø 4131	9E26 26 Ø3 9E28 BD 94 A1		BNE JSR	L9E2B L94A1	BRANCH IF NO DRAW DRAW A LINE FROM (HORBEG, VERBEG) TO (HOREND, VEREND)
4132	9E2B ØF D5	L9E2B	CLR	VD5	RESET UPDATE FLAG
4133 4134	9E2D ØF D6 9E2F 7E 9C DD		CLR JMP	VD6 L9CDD	RESET DRAW FLAG GO GET ANOTHER COMMAND
4135	3121 7L 30 00		OHI	29000	do del Anomen command
4136 4137	9E32 BD 9B 98			W POSITION	CET A CHAR EROM COMMAND LINE
4138	9E35 34 Ø2	L9E32	PSHS	L9B98 A	GET A CHAR FROM COMMAND LINE SAVE CHARACTER
	> 9E37 BD 9E 5E			L9E5E	EVALUATE HORIZ DIFFERENCE
414Ø 4141	9E3A 34 Ø6 9E3C BD 9B 98			B,A L9B98	SAVE IT ON STACK GET A CHAR FROM COMMAND LINE
4142	9E3F 81 2C		CMPA	#','	CHECK FOR COMMA
4143	9E41 10 26 FF 07 > 9E45 BD 9E 5B			L9D4C L9E5B	FC ERROR IF NO COMMA EVALUATE VERT DIFFERENCE
4145	9E48 1F Ø1			D, X	SAVE VERT DIFFERENCE IN X
4146	9E4A 35 4Ø		PULS PULS		GET HORIZ DIFFERENCE IN U
4147 4148	9E4C 35 Ø2 9E4E 81 2B			#'+'	GET FIRST COMMAND CHARACTER *IF FIRST COMMAND CHAR WAS EITHER + OR -, TREAT
4149	9E5Ø 27 Ø4			L9E56	*THE VALUES IN U & X AS DIFFERENCES AND MOVE
415Ø 4151	9E52 81 2D 9E54 26 A6			#'-' L9DFC	*POINTER, OTHERWISE TREAT U & X AS AN ABSOLUTE *POSITION AND MOVE THE CURRENT POSITION THERE.
4152	9E56 1F 3Ø	L9E56	TFR	U,D	PUT HORIZ DIFFERENCE IN ACCD
4153 4154	9E58 7E 9D CB		JMP	L9DCB	GO MOVE THE DRAW POSITION
4155	9E5B BD 9B 98	L9E5B		L9B98	GET A CHAR FROM COMMAND LINE
4156 4157	9E5E 81 2B 9E6Ø 27 Ø7	L9E5E		#'+' L9E69	*CHECK FOR A LEADING PLUS SIGN (RELATIVE MOTION) *AND BRANCH IF RELATIVE
4158	9E62 81 2D		CMPA	#'-'	=CHECK FOR A LEADING MINUS SIGN (RELATIVE MOTION)
4159 4160	9E64 27 Ø4 9E66 BD 9B E2			L9E6A L9BE2	=AND BRANCH IF RELATIVE 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		SAVE ADD/SUB FLAG
4163 4164	9E6C BD 9C CB	*	JSK	L9CCB	EVALUATE DECIMAL NUMBER IN COMMAND STRING - RETURN VALUE IN ACCB
4165 4166	9E6F 35 Ø2 9E71 4D		PULS TSTA		GET ADD/SUB FLAG
4167	9E72 27 Ø4			L9E78	CHECK IT, Ø:ADD, ◇ Ø:SUB RETURN IF ADD
4168	9E74 4F		CLRA		*
4169 4170	9E75 5Ø 9E76 82 ØØ		NEGB SBCA	#\$00	*NEGATE ACCB INTO A TWO BYTE SIGNED VALUE IN ACCD
4171	9E78 39	L9E78 *	RTS		
4172 4173		* TABLE	OF SI	NES AND COSINES FOR CIRCLE	
4174 4175	9E79 ØØ ØØ ØØ Ø1	* L9E79	FDB	\$0000,\$0001	SUBARC Ø
4176	9E7D FE C5 19 19	L9E7D		\$FEC5,\$1919	SUBARC 1
4176 4177	9E81 FB 16 31 F2	L9E81	FDB	\$FB16,\$31F2	SUBARC 1 SUBARC 2
4176 4177 4178 4179	9E81 FB 16 31 F2 9E85 F4 FB 4A 51 9E89 EC 84 61 F9	L9E81 L9E85 L9E89	FDB FDB FDB	\$FB16,\$31F2 \$F4FB,\$4A51 \$EC84,\$61F9	SUBARC 1
4176 4177 4178 4179 4180	9E81 FB 16 31 F2 9E85 F4 FB 4A 51 9E89 EC 84 61 F9 9E8D E1 C7 78 AE	L9E81 L9E85 L9E89 L9E8D	FDB FDB FDB FDB	\$FB16,\$31F2 \$F4FB,\$4A51 \$EC84,\$61F9 \$E1C7,\$78AE	SUBARC 1 SUBARC 2 SUBARC 3 SUBARC 4 SUBARC 5
4176 4177 4178 4179	9E81 FB 16 31 F2 9E85 F4 FB 4A 51 9E89 EC 84 61 F9	L9E81 L9E85 L9E89	FDB FDB FDB	\$FB16,\$31F2 \$F4FB,\$4A51 \$EC84,\$61F9	SUBARC 1 SUBARC 2 SUBARC 3 SUBARC 4
4176 4177 4178 4179 4180 4181 4182 4183	9E81 FB 16 31 F2 9E85 F4 FB 4A 51 9E89 EC 84 61 F9 9E8D E1 C7 78 AE 9E91 D4 DC 8E 3B	L9E81 L9E85 L9E89 L9E8D L9E91	FDB FDB FDB FDB FDB	\$FB16,\$31F2 \$F4FB,\$4A51 \$EC84,\$61F9 \$E1C7,\$78AE \$D4DC,\$8E3B	SUBARC 1 SUBARC 2 SUBARC 3 SUBARC 4 SUBARC 5 SUBARC 5 SUBARC 5
4176 4177 4178 4179 4180 4181 4182 4183 4184 4185	9E81 FB 16 31 F2 9E85 F4 FB 4A 51 9E89 EC 84 61 F9 9E8D E1 C7 78 AE 9E91 D4 DC 8E 3B 9E95 C5 E5 A2 69	L9E81 L9E85 L9E89 L9E8D L9E91 L9E95 L9E99 *	FDB FDB FDB FDB FDB FDB	\$FB16,\$31F2 \$F4FB,\$4A51 \$EC64,\$61F9 \$E1C7,\$78AE \$D4DC,\$8E3B \$C5E5,\$A269 \$B506,\$B506	SUBARC 1 SUBARC 2 SUBARC 3 SUBARC 4 SUBARC 5 SUBARC 6 SUBARC 7
4176 4177 4178 4179 4180 4181 4182 4183 4184 4185 4186	9E81 FB 16 31 F2 9E85 F4 FB 4A 51 9E89 EC 84 61 F9 9E8D E1 C7 78 AE 9E91 D4 DC 8E 3B 9E95 C5 E5 A2 69	L9E81 L9E85 L9E89 L9E8D L9E91 L9E95 L9E99 * * CIRCL	FDB FDB FDB FDB FDB FDB FDB	\$FB16,\$31F2 \$F4FB,\$4A51 \$EC84,\$61F9 \$E1C7,\$78AE \$D4DC,\$8E3B \$C5E5,\$A2E9 \$B506,\$B506	SUBARC 1 SUBARC 2 SUBARC 3 SUBARC 4 SUBARC 5 SUBARC 6 SUBARC 7 SUBARC 7
4176 4177 4178 4179 4180 4181 4182 4183 4184 4185 4186 4187 4188	9881 FB 16 31 F2 9885 F4 FB 4A 51 9889 EC 84 61 F9 9880 E1 C7 78 AE 9991 D4 DC 88 3B 9995 C5 E5 A2 69 9899 B5 Ø6 B5 Ø6	L9E81 L9E85 L9E89 L9E8D L9E91 L9E95 L9E99 * * CIRCL	FDB FDB FDB FDB FDB EDB FDB	\$FB16,\$31F2 \$F4FB,\$4A51 \$EC64,\$61F9 \$E1C7,\$78AE \$D4DC,\$883B \$C5E5,\$A269 \$B506,\$B506 IS ACTUALLY DRAWN AS A 64 SIDED T IS COMPOSED OF 64 LINE COMMA #'@'	SUBARC 1 SUBARC 2 SUBARC 3 SUBARC 4 SUBARC 5 SUBARC 6 SUBARC 7 SUBARC 8 NDS CHECK FOR @ SIGN
4176 4177 4178 4179 4180 4181 4182 4183 4184 4185 4186 4187 4188 4189	9E81 FB 16 31 F2 9E85 F4 FB 4A 51 9E89 EC 84 61 F9 9E8D E1 C7 78 AE 9E91 D4 DC 8E 3B 9E95 C5 E5 A2 69 9E99 B5 Ø6 B5 Ø6 9E9D 81 4Ø 9E9F 26 Ø2	L9E81 L9E85 L9E89 L9E80 L9E91 L9E95 * * * CIRCL * THE C	FDB FDB FDB FDB FDB FDB FDB FDB CIRCLE GON. I CMPA BNE	\$FB16,\$31F2 \$F4FB,\$4A51 \$EC84,\$61F9 \$E1C7,\$78AE \$D4DC,\$8E3B \$C5E5,\$A2E9 \$B5Ø6,\$B5Ø6 IS ACTUALLY DRAWN AS A 64 SIDED T IS COMPOSED OF 64 LINE COMMA #'@' L9EA3	SUBARC 1 SUBARC 2 SUBARC 3 SUBARC 4 SUBARC 5 SUBARC 6 SUBARC 7 SUBARC 8 NDS CHECK FOR @ SIGN SKIP IF NOT
4176 4177 4178 4179 4180 4181 4182 4183 4184 4185 4186 4187 4188	9881 FB 16 31 F2 9885 F4 FB 4A 51 9889 EC 84 61 F9 9880 E1 C7 78 AE 9991 D4 DC 88 3B 9995 C5 E5 A2 69 9899 B5 Ø6 B5 Ø6	L9E81 L9E85 L9E89 L9E80 L9E91 L9E95 * * * CIRCL * THE C	FDB FDB FDB FDB FDB FDB FDB CIRCLE GON. I CMPA BNE JSR	\$FB16,\$31F2 \$F4FB,\$4A51 \$EC64,\$61F9 \$E1C7,\$78AE \$D4DC,\$883B \$C5E5,\$A269 \$B506,\$B506 IS ACTUALLY DRAWN AS A 64 SIDED T IS COMPOSED OF 64 LINE COMMA #'@'	SUBARC 1 SUBARC 2 SUBARC 3 SUBARC 4 SUBARC 5 SUBARC 6 SUBARC 7 SUBARC 8 NDS CHECK FOR @ SIGN
4176 4177 4178 4179 4180 4181 4182 4183 4184 4185 4186 4187 4188 4189 4190 4191 4192	9E81 FB 16 31 F2 9E85 F4 FB 4A 51 9E89 EC 84 61 F9 9E80 E1 C7 78 AE 9E91 D4 DC 8E 3B 9E95 C5 E5 A2 69 9E99 B5 Ø6 B5 Ø6 9E90 81 4Ø 9E9F 26 Ø2 9EA1 9D 9F 9EA3 BD 95 22 9EA6 BD 93 B2	L9E81 L9E85 L9E89 L9E8D L9E91 L9E95 L9E99 * * CIRCL * THE C	FDB FDB FDB FDB FDB FDB EIRCLE GON. I CMPA BNE JSR JSR	\$FB16,\$31F2 \$F4FB,\$4A51 \$EC84,\$61F9 \$E1C7,\$78AE \$D4DC,\$8E3B \$C5E5,\$A2E9 \$B5Ø6,\$B5Ø6 IS ACTUALLY DRAWN AS A 64 SIDED T IS COMPOSED OF 64 LINE COMMA #'@' L9EA3 GETNCH L9522 L9382	SUBARC 1 SUBARC 2 SUBARC 3 SUBARC 4 SUBARC 5 SUBARC 6 SUBARC 7 SUBARC 8 NDS CHECK FOR @ SIGN SKIP IF NOT GET ANOTHER CHARACTER FROM BASIC GET MAX HOR & VER COORD VALUES AND PUT THEM IN VD3 AND VD5 GET HOX & VER CENTER COORDS AND PUT IN HORBEG, VERBEG
4176 4177 4178 4179 4180 4181 4182 4183 4184 4185 4186 4187 4188 4189 4190 4191	9881 FB 16 31 F2 9885 F4 FB 4A 51 9889 EC 84 61 F9 9880 E1 C7 78 AE 9991 D4 DC 88 3B 9595 C5 E5 A2 69 9699 B5 Ø6 B5 Ø6 9699 B1 4Ø 969F 26 Ø2 96A1 9D 9F 96A3 BD 95 22	L9E81 L9E85 L9E89 L9E8D L9E91 L9E95 L9E99 * * CIRCL * THE C	FDB FDB FDB FDB FDB FDB EIRCLE GON. I CMPA BNE JSR JSR	\$FB16,\$31F2 \$F4FB,\$4A51 \$EC64,\$61F9 \$E1C7,\$78AE \$D4DC,\$8E3B \$C5E5,\$A269 \$B506,\$B506 IS ACTUALLY DRAWN AS A 64 SIDED T IS COMPOSED OF 64 LINE COMMA #'@' L9EA3 GETNCH	SUBARC 1 SUBARC 2 SUBARC 3 SUBARC 4 SUBARC 5 SUBARC 6 SUBARC 7 SUBARC 8 NDS CHECK FOR @ SIGN SKIP IF NOT GET ANOTHER CHARACTER FROM BASIC GET MAX HOR & VER COORD VALUES AND PUT THEM IN VD3 AND VD5
4176 4177 4178 4189 4181 4182 4183 4184 4185 4186 4187 4188 4189 4191 4192 4193 4194 4195	9E81 FB 16 31 F2 9E85 F4 FB 4A 51 9E89 EC 84 61 F9 9E80 E1 C7 78 AE 9E91 D4 DC 8E 3B 9E95 C5 E5 A2 69 9E99 B5 Ø6 B5 Ø6 9E90 81 4Ø 9E9F 26 Ø2 9EA1 9D 9F 9EA3 BD 95 22 9EA4 BD 93 B2 9EA9 BD 93 1D 9EAC AE C4 9EAE 9F CB	L9E81 L9E85 L9E89 L9E8D L9E91 L9E95 L9E99 * * CIRCL * THE C	FDB FDB FDB FDB FDB FDB FDB FDB SIRCLE GON. I CMPA BNE JSR JSR JSR JSR JSR JSR	\$FB16,\$31F2 \$F4FB,\$4A51 \$EC84,\$61F9 \$E1C7,\$78AE \$D4DC,\$8E3B \$C5E5,\$A2E9 \$B5Ø6,\$B5Ø6 IS ACTUALLY DRAWN AS A 64 SIDED T IS COMPOSED OF 64 LINE COMMA #'@' L9EA3 GETNCH L9522 L9382 L9310 ,U VCB	SUBARC 1 SUBARC 2 SUBARC 3 SUBARC 4 SUBARC 5 SUBARC 6 SUBARC 6 SUBARC 7 SUBARC 8 NDS CHECK FOR @ SIGN SKIP IF NOT GET ANOTHER CHARACTER FROM BASIC GET MAX HOR & VER CENTER COORD VALUES AND PUT THEM IN VD3 AND VD5 GET HOR & VER CENTER COORDS AND PUT IN HORBEG, VERBEG NORMALIZE START COORDS FOR PROPER PMODE GET HOR COORD SAVE IT
4176 4177 4178 4189 4181 4182 4183 4184 4185 4186 4187 4199 4199 4199 4191 4192 4193 4194 4195	9881 FB 16 31 F2 9885 F4 FB 4A 51 9880 EC 84 61 F9 9880 E1 C7 78 AE 9891 D4 DC 8E 3B 9899 B5 Ø6 B5 Ø6 9899 B5 Ø6 B5 Ø6 9899 B5 Ø6 B5 Ø6 9899 B7 B8	L9E81 L9E85 L9E89 L9E8D L9E91 L9E95 L9E99 * * CIRCL * THE C	FDB FDB FDB FDB FDB FDB EIRCLE GON. I CMPA BNE JSR JSR JSR JSR JSR JSR JSR	\$FB16,\$31F2 \$F4FB,\$4A51 \$EC64,\$61F9 \$E1C7,\$78AE \$D4DC,\$8E3B \$C5E5,\$A2E9 \$B506,\$B506 IS ACTUALLY DRAWN AS A 64 SIDED T IS COMPOSED OF 64 LINE COMMA #'@' L9EA3 GETNCH L9522 L931D ,U VCB \$82,U	SUBARC 1 SUBARC 2 SUBARC 3 SUBARC 4 SUBARC 5 SUBARC 6 SUBARC 7 SUBARC 7 SUBARC 8 NDS CHECK FOR @ SIGN SKIP IF NOT GET ANOTHER CHARACTER FROM BASIC GET ANOTHER CHARACTER FROM BASIC GET MAX HOR & VER COORD VALUES AND PUT THEM IN VD3 AND VD5 GET HOR & VER CENTER COORDS AND PUT IN HORBEG, VERBEG NORMALIZE START COORDS FOR PROPER PMODE GET HOR COORD
4176 4177 4178 4179 4180 4181 4182 4183 4184 4185 4186 4187 4190 4190 4191 4192 4193 4194 4195 4196 4197	9E81 FB 16 31 F2 9E85 F4 FB 4A 51 9E89 EC 84 61 F9 9E80 E1 C7 78 AE 9E91 D4 DC 8E 3B 9E95 C5 E5 A2 69 9E99 B5 Ø6 B5 Ø6 9E90 81 4Ø 9E9F 26 Ø2 9EA1 90 9F 9EA3 BD 95 22 9EA6 BD 93 B2 9EA9 BD 93 1D 9EAC AE C4 9EB2 9F CB	L9E81 L9E85 L9E89 L9E8D L9E91 L9E95 L9E99 * * CIRCL * THE C	FDB FDB FDB FDB FDB EIRCLE SON. I CMPA BNE JSR JSR JSR JSR JSR JSR JSR JSR JSR JSR	\$FB16,\$31F2 \$F4FB,\$4A51 \$EC84,\$61F9 \$E1C7,\$78AE \$D4DC,\$8E3B \$C5E5,\$A2E9 \$B5Ø6,\$B5Ø6 IS ACTUALLY DRAWN AS A 64 SIDED T IS COMPOSED OF 64 LINE COMMA #'@' L9EA3 GETNCH L9522 L93B2 L931D ,U VCB \$Ø2,U VCD SYNCOMMA	SUBARC 1 SUBARC 2 SUBARC 3 SUBARC 4 SUBARC 5 SUBARC 6 SUBARC 6 SUBARC 7 SUBARC 8 NDS CHECK FOR @ SIGN SKIP IF NOT GET ANOTHER CHARACTER FROM BASIC GET MAX HOR & VER CONTEN VALUES AND PUT THEM IN VD3 AND VD5 GET HOR & VER CENTER COORDS AND PUT IN HORBEG, VERBEG NORMALIZE START COORDS FOR PROPER PMODE GET HOR COORD SAVE IT GET VERT COORD SAVE IT STYNTAX CHECK FOR COMMA
4176 4177 4178 4179 4180 4181 4182 4183 4184 4185 4186 4187 4191 4192 4193 4194 4195 4196 4197 4198	9881 FB 16 31 F2 9885 F4 FB 4A 51 9880 EC 84 61 F9 9880 E1 C7 78 AE 9981 D4 DC 8E 3B 9695 C5 E5 A2 69 9699 B5 06 B5 06 9899 B5 06 B5 06 9899 B5 06 B5 06 9899 B5 06 B5 06	L9E81 L9E85 L9E89 L9E8D L9E91 L9E95 L9E99 * * CIRCL * THE C	FDB FDB FDB FDB FDB FDB FDB FDB SIRCLE SON. I CMPA BNE JSR JSR JSR JSR LDX STX LDX STX LDX STX JSR	\$FB16,\$31F2 \$F4FB,\$4A51 \$EC84,\$61F9 \$E1C7,\$78AE \$D4DC,\$8E3B \$C5E5,\$A269 \$B506,\$B506 IS ACTUALLY DRAWN AS A 64 SIDED T IS COMPOSED OF 64 LINE COMMA #'@' L9EA3 GETNCH L9522 L9382 L9310 ,U VCB \$82,U VCD \$SYNCOMMA LB73D	SUBARC 1 SUBARC 2 SUBARC 3 SUBARC 4 SUBARC 5 SUBARC 6 SUBARC 6 SUBARC 7 SUBARC 8 NDS CHECK FOR @ SIGN SKIP IF NOT GET ANOTHER CHARACTER FROM BASIC GET MAX HOR & VER COORD VALUES AND PUT THEM IN VD3 AND VD5 GET HOR & VER CENTER COORDS AND PUT IN HORBEG, VERBEG NORMALIZE START COORDS FOR PROPER PMODE GET HOR COORD SAVE IT GET VERT COORD SAVE IT SYNTAX CHECK FOR COMMA EVALUATE EXPRESSION RETURN VALUE IN X
4176 4177 4178 4179 4180 4181 4182 4183 4184 4185 4186 4187 4188 4199 4191 4191 4191 4194 4195 4196 4197 4198 4199 4199 4199 4201	9E81 FB 16 31 F2 9E85 F4 FB 4A 51 9E89 EC 84 61 F9 9E80 E1 C7 78 AE 9E91 D4 DC 8E 3B 9E95 C5 E5 A2 69 9E99 B5 Ø6 B5 Ø6 9E90 81 4Ø 9E97 26 Ø2 9EA1 90 9F 9EA3 BD 95 22 9EA6 BD 93 B2 9EA9 BD 93 1D 9EAC AE C4 9EB2 9F CB 9EB4 BD E2 GD 9EB4 BD B2 GD 9EB4 BD B2 GD 9EB4 BD B2 GD 9EB7 BD B7 AD 9EBA CE ØØ CF	L9E81 L9E85 L9E89 L9E8D L9E91 L9E95 L9E99 * * CIRCL * THE C	FDB FDB FDB FDB FDB FDB FDB FDB FDB FDB	\$FB16,\$31F2 \$F4FB,\$4A51 \$EC84,\$61F9 \$E1C7,\$78AE \$D4DC,\$8E3B \$C5E5,\$A2E9 \$B5Ø6,\$B5Ø6 IS ACTUALLY DRAWN AS A 64 SIDED T IS COMPOSED OF 64 LINE COMMA #'@' L9EA3 GETNCH L9522 L93B2 L931D ,U VCB \$Ø2,U VCD \$YNCOMMA LB73D #VCF ,U	SUBARC 1 SUBARC 2 SUBARC 3 SUBARC 4 SUBARC 5 SUBARC 6 SUBARC 6 SUBARC 7 SUBARC 8 NDS CHECK FOR @ SIGN SKIP IF NOT GET ANOTHER CHARACTER FROM BASIC GET MAX HOR & VER CENTER COORDS AND PUT THEM IN VD3 AND VD5 GET HOR & VER CENTER COORDS AND PUT IN HORBEG, VERBEG NORMALIZE START COORDS FOR PROPER PMODE GET HOR COORD SAVE IT GET VERT COORD SAVE IT SYNTAX CHECK FOR COMMA EVALUATE EXPRESSION RETURN VALUE IN X POINT U TO TEMP DATA STORAGE SAVE POTOR TO THE POTO TO THE POTO TO TO THE POTO TO
4176 4177 4178 4179 4180 4181 4182 4183 4184 4185 4186 4187 4191 4192 4193 4194 4195 4196 4197 4198 4199 4200 4201	9881 FB 16 31 F2 9885 F4 FB 4A 51 9880 EC 84 61 F9 9880 E1 C7 78 AE 9981 D4 DC 8E 3B 9695 C5 E5 A2 69 9699 B5 Ø6 B5 Ø6 9899 B5 Ø6 B5 Ø6 9899 B6 Ø2 9891 B6 Ø2 9891 B7 Ø2 9891 B7 Ø2 9891 B7 Ø2 9891 B7 Ø3	L9E81 L9E85 L9E89 L9E8D L9E91 L9E95 L9E99 * * CIRCL * THE C	FDB	\$FB16,\$31F2 \$F4FB,\$4A51 \$EC84,\$61F9 \$E1C7,\$78AE \$D4DC,\$8E3B \$C5E5,\$A2E9 \$B506,\$B506 IS ACTUALLY DRAWN AS A 64 SIDED T IS COMPOSED OF 64 LINE COMMA #'@' L9EA3 GETNCH L9522 L9382 L9310 ,U VCB \$802,U VCD \$802,U VCD \$YNCOMMA LB73D #VCF ,U L9320	SUBARC 1 SUBARC 2 SUBARC 3 SUBARC 4 SUBARC 5 SUBARC 6 SUBARC 6 SUBARC 7 SUBARC 8 NDS CHECK FOR @ SIGN SKIP IF NOT GET ANOTHER CHARACTER FROM BASIC GET MAX HOR & VER COORD VALUES AND PUT THEM IN VD3 AND VD5 GET HOR & VER CENTER COORDS AND PUT IN HORBEG, VERBEG NORMALIZE START COORDS FOR PROPER PMODE GET HOR COORD SAVE IT GET VERT COORD SAVE IT SYNTAX CHECK FOR COMMA EVALUATE EXPRESSION RETURN VALUE IN X POINT U TO TEMP DATA STORAGE SAVE RADIUS NORMALIZE RADIUS
4176 4177 4178 4179 4180 4181 4182 4183 4184 4185 4186 4187 4188 4199 4191 4191 4191 4194 4195 4196 4197 4198 4199 4199 4199 4201	9E81 FB 16 31 F2 9E85 F4 FB 4A 51 9E89 EC 84 61 F9 9E80 E1 C7 78 AE 9E91 D4 DC 8E 3B 9E95 C5 E5 A2 69 9E99 B5 Ø6 B5 Ø6 9E90 81 4Ø 9E97 26 Ø2 9EA1 90 9F 9EA3 BD 95 22 9EA6 BD 93 B2 9EA9 BD 93 1D 9EAC AE C4 9EB2 9F CB 9EB4 BD E2 GD 9EB4 BD B2 GD 9EB4 BD B2 GD 9EB4 BD B2 GD 9EB7 BD B7 AD 9EBA CE ØØ CF	L9E81 L9E85 L9E89 L9E8D L9E91 L9E95 L9E99 * * CIRCL * THE C	FDB	\$FB16,\$31F2 \$F4FB,\$4A51 \$EC84,\$61F9 \$E1C7,\$78AE \$D4DC,\$8E3B \$C5E5,\$A2E9 \$B5Ø6,\$B5Ø6 IS ACTUALLY DRAWN AS A 64 SIDED T IS COMPOSED OF 64 LINE COMMA #'@' L9EA3 GETNCH L9522 L93B2 L931D ,U VCB \$Ø2,U VCD \$YNCOMMA LB73D #VCF ,U	SUBARC 1 SUBARC 2 SUBARC 3 SUBARC 4 SUBARC 5 SUBARC 6 SUBARC 6 SUBARC 7 SUBARC 8 NDS CHECK FOR @ SIGN SKIP IF NOT GET ANOTHER CHARACTER FROM BASIC GET MAX HOR & VER CENTER COORDS AND PUT THEM IN VD3 AND VD5 GET HOR & VER CENTER COORDS AND PUT IN HORBEG, VERBEG NORMALIZE START COORDS FOR PROPER PMODE GET HOR COORD SAVE IT GET VERT COORD SAVE IT SYNTAX CHECK FOR COMMA EVALUATE EXPRESSION RETURN VALUE IN X POINT U TO TEMP DATA STORAGE SAVE POTOR TO THE POTO TO THE POTO TO TO THE POTO TO
4176 4177 4178 4179 4180 4181 4182 4183 4184 4185 4186 4187 4198 4191 4192 4193 4194 4195 4197 4198 4199 4202 4203 4204 4205	9881 FB 16 31 F2 9885 F4 FB 4A 51 9880 EC 84 61 F9 9880 E1 C7 78 AE 9981 D4 DC 88 3B 9595 C5 E5 A2 69 9599 B5 Ø6 B5 Ø6 9599 B7 CØ2 9564 BD 93 B2 9564 BD 93 B2 9568 AF C4 9588 B7 CB 9588 B7 CB 9588 B7 CB 9588 B7 B7 B7 9588 CE ØØ CF 9580 AF C4 9588 B7 C2 9586 B0 B7 S1	L9E81 L9E85 L9E89 L9E8D L9E91 L9E95 L9E99 * * CIRCL * THE C	FDB	\$FB16,\$31F2 \$F4FB,\$4A51 \$EC64,\$61F9 \$E1C7,\$78AE \$D4DC,\$8E3B \$C5E5,\$A2E9 \$B506,\$B506 IS ACTUALLY DRAWN AS A 64 SIDED T IS COMPOSED OF 64 LINE COMMA #'@' L9EA3 GETNCH L9522 L9382 L931D ,U VCB \$02,U VCD \$802,U VCD \$YNCOMMA LB730 #VCF ,U L9320 #\$01 \$ETFLG L9521	SUBARC 1 SUBARC 2 SUBARC 3 SUBARC 4 SUBARC 5 SUBARC 6 SUBARC 6 SUBARC 7 SUBARC 8 NDS CHECK FOR @ SIGN SKIP IF NOT GET ANOTHER CHARACTER FROM BASIC GET MAX HOR & VER COORD VALUES AND PUT THEM IN VD3 AND VD5 GET HOR & VER CENTER COORDS AND PUT IN HORBEG, VERBEG NORMALIZE START COORDS FOR PROPER PMODE GET HOR COORD SAVE IT GET VERT COORD SAVE IT SYNTAX CHECK FOR COMMA EVALUATE EXPRESSION RETURN VALUE IN X POINT U TO TEMP DATA STORAGE SAVE RADIUS SET TO PSET SAVE POSET/PRESET FLAG GO EVALUATE COLOR EXPRESSION AND SAVE IN WCOLOR
4176 4177 4178 4179 4180 4181 4182 4183 4184 4185 4186 4190 4191 4192 4193 4194 4195 4196 4197 4198 4199 4201 4201 4202 4203 4204	9881 FB 16 31 F2 9885 F4 FB 4A 51 9880 EC 84 61 F9 9880 E1 C7 78 AE 9891 D4 DC 8E 3B 9899 B5 Ø6 B5 Ø6 9899 B7 B0 B7 9843 BD 95 B2 9848 BD 93 B2 9849 BD 93 B2 9849 BD 93 B2 9849 BD 93 B2 9848 B7 CB 9888 B7 B8 B7 B7 9888 B8 B8 B8 9888 B8 B8 9888 B8 B8 9888 B8 B8 9888 B8 98	L9E81 L9E85 L9E89 L9E8D L9E91 L9E95 L9E99 * * CIRCL * THE C	FDB	\$FB16,\$31F2 \$F4FB,\$4A51 \$EC84,\$61F9 \$E1C7,\$78AE \$D4DC,\$8E3B \$C5E5,\$A2E9 \$B506,\$B506 IS ACTUALLY DRAWN AS A 64 SIDED T IS COMPOSED OF 64 LINE COMMA #'@' L9EA3 GETNCH L9522 L93B2 L931D ,U VCB \$82,U VCD SYNCOMMA LB73D #VCF ,U L9320 #\$01 SETFLG	SUBARC 1 SUBARC 2 SUBARC 3 SUBARC 4 SUBARC 5 SUBARC 6 SUBARC 6 SUBARC 7 SUBARC 8 NDS CHECK FOR @ SIGN SKIP IF NOT GET ANOTHER CHARACTER FROM BASIC GET MAX HOR & VER CENTER COORDS AND PUT THEM IN VD3 AND VD5 GET HOR & VER CENTER COORDS AND PUT IN HORBEG, VERBEG NORMALIZE START COORDS FOR PROPER PMODE GET HOR COORD SAVE IT GET VERT COORD SAVE IT SYNTAX CHECK FOR COMMA EVALUATE EXPRESSION RETURN VALUE IN X POINT U TO TEMP DATA STORAGE SAVE RADIUS NORMALIZE RADIUS SET TO PSET SAVE PSET/PRESET FLAG
4176 4177 4178 4179 4180 4181 4182 4183 4184 4185 4186 4187 4191 4192 4193 4194 4195 4197 4199 4200 4201 4202 4203 4204 4205 4206 4207 4208	9881 FB 16 31 F2 9885 F4 FB 4A 51 9880 EC 84 61 F9 9880 E1 C7 78 AE 9981 D4 DC 88 3B 9999 B5 Ø6 B5 Ø6 9899 B5 Ø6 B5 Ø6 9899 B5 Ø6 B5 Ø6 9899 B1 4Ø 9997 C6 Ø2 9840 B0 93 B2 9840 B0 B0 B1 9840 B0 B1 9840 B0 B1 9840 B0 B2 9840 B0 B7 9840 B0 B0 B0 9840 B0 B0 9840 B0 B0 9840 B0	L9E81 L9E85 L9E89 L9E8D L9E91 L9E95 L9E99 * * CIRCL * THE C	FDB	\$FB16,\$31F2 \$F4FB,\$4A51 \$EC84,\$61F9 \$E1C7,\$78AE \$D4DC,\$8E3B \$C5E5,\$A2E9 \$B506,\$B506 IS ACTUALLY DRAWN AS A 64 SIDED T IS COMPOSED OF 64 LINE COMMA #'@' L9EA3 GETNCH L9522 L9382 L931D ,U VCB \$W02,U VCD SYNCOMMA LB73D #VCF ,U L9220 #\$01 ESTFLG L9521 #501 ESTFLG L9551 #\$100 GETCCH L95EP	SUBARC 1 SUBARC 2 SUBARC 3 SUBARC 4 SUBARC 5 SUBARC 6 SUBARC 6 SUBARC 7 SUBARC 8 NDS CHECK FOR @ SIGN SKIP IF NOT GET ANOTHER CHARACTER FROM BASIC GET MAX HOR & VER COORD VALUES AND PUT THEM IN VD3 AND VD5 GET HOR & VER CENTER COORDS AND PUT IN HORBEG, VERBEG NORMALIZE START COORDS FOR PROPER PMODE GET HOR COORD SAVE IT GET VERT COORD SAVE IT SYNTAX CHECK FOR COMMA EVALUATE EXPRESSION RETURN VALUE IN X POINT U TO TEMP DATA STORAGE SAVE RADIUS NORMALIZE RADIUS SET TO PSET SAVE PSET/PRESET FLAG GO EVALUATE COLOR EXPRESSION AND SAVE IN WCOLOR HEIGHT/WIOTH RATIO DEFAULT VALUE GET AN INPUT CHARACTER FROM BASIC BRANCH IF NONE
4176 4177 4178 4179 4180 4181 4182 4183 4184 4185 4186 4187 4194 4199 4191 4192 4193 4194 4195 4196 4197 4198 4199 4201 4202 4203 4204 4205 4206 4207	9881 FB 16 31 F2 9885 F4 FB 4A 51 9880 EC 84 61 F9 9880 E1 C7 78 AE 9891 D4 DC 8E 3B 9899 B5 Ø6 B5 Ø6 9899 B7 B0 B7 9843 BD 95 B2 9848 BD 93 B2 9849 BD 93 B2 9849 BD 93 B2 9849 BD 93 B2 9848 B7 CB 9888 B7 B8 B7 B7 9888 B8 B8 B8 9888 B8 B8 9888 B8 B8 9888 B8 B8 9888 B8 98	L9E81 L9E85 L9E89 L9E8D L9E91 L9E95 L9E99 * * CIRCL * THE C	FDB	\$FB16,\$31F2 \$F4FB,\$4A51 \$EC64,\$61F9 \$E1C7,\$78AE \$D4DC,\$8E3B \$C5E5,\$A2E9 \$B506,\$B506 IS ACTUALLY DRAWN AS A 64 SIDED T IS COMPOSED OF 64 LINE COMMA #'@' L9EA3 GETNCH L9522 L931D ,U VCB \$02,U VCB \$02,U VCB \$\$VCB \$\$VCF,U L9320 #\$01 ETFLE L9581 #\$100 GETCCH	SUBARC 1 SUBARC 2 SUBARC 3 SUBARC 4 SUBARC 5 SUBARC 6 SUBARC 6 SUBARC 7 SUBARC 8 NDS CHECK FOR @ SIGN SKIP IF NOT GET ANOTHER CHARACTER FROM BASIC GET MAX HOR & VER COURD VALUES AND PUT THEM IN VD3 AND VD5 GET HOR & VER CENTER COORDS AND PUT IN HORBEG, VERBEG NORMALIZE START COORDS FOR PROPER PMODE GET HOR COORD SAVE IT SYNTAX CHECK FOR COMMA EVALUATE EXPRESSION RETURN VALUE IN X POINT U TO TEMP DATA STORAGE SAVE RADIUS NORMALIZE RADIUS SET TO PSET SAVE PSET/PRESET FLAG GO EVALUATE COLOR EXPRESSION AND SAVE IN WCOLOR HEIGHT/WIDTH RATIO DEFAULT VALUE GET AN INPUT CHARACTER FROM BASIC
4176 4177 4178 4179 4180 4181 4182 4183 4184 4185 4186 4187 4198 4191 4192 4193 4194 4195 4196 4197 4198 4202 4203 4204 4205 4206 4207 4208 4209 4210	9881 FB 16 31 F2 9885 F4 FB 4A 51 9880 EC 84 61 F9 9880 E1 C7 78 AE 9981 D4 DC 88 3B 9999 B5 Ø6 B5 Ø6 9899 B5 Ø6 B5 Ø6 9899 B5 Ø6 B5 Ø6 9899 B1 4Ø 9997 C6 Ø2 9840 B0 93 B2 9840 B0 B0 B1 9840 B0 B7 3D 9840 B7 B7 9840 B7 9850 B	L9E81 L9E85 L9E89 L9E8D L9E91 L9E95 L9E99 * * CIRCL * THE C	FDB	\$FB16,\$31F2 \$F4FB,\$4A51 \$EC84,\$61F9 \$E1C7,\$78AE \$D4DC,\$8E3B \$C5E5,\$A269 \$B506,\$B506 IS ACTUALLY DRAWN AS A 64 SIDED T IS COMPOSED OF 64 LINE COMMA #'@' L9EA3 GETNCH L9522 L93B2 L93B2 L931D ,U VCB SYNCOMMA LB73D #VCF ,U L9220 #\$01 SETFLG L9528 #\$100 GETCCH L9529 F\$100 GETCCH L9521 F\$100	SUBARC 1 SUBARC 2 SUBARC 3 SUBARC 4 SUBARC 5 SUBARC 6 SUBARC 6 SUBARC 7 SUBARC 8 NDS CHECK FOR @ SIGN SKIP IF NOT GET ANOTHER CHARACTER FROM BASIC GET MAX HOR & VER COORD VALUES AND PUT THEM IN VD3 AND VD5 GET HOR 8 VER CENTER COORDS AND PUT IN HORBEG, VERBEG NORMALIZE START COORDS FOR PROPER PMODE GET HOR COORD SAVE IT GET VERT COORD SAVE IT SYNTAX CHECK FOR COMMA EVALUATE EXPRESSION RETURN VALUE IN X POINT U TO TEMP DATA STORAGE SAVE RADIUS NORMALIZE RADIUS SET TO PSET SAVE PSET/PRESET FLAG GO EVALUATE COLOR EXPRESSION AND SAVE IN WCOLOR HEIGHT/WIDTH RATIO DEFAULT VALUE GET AN INPUT CHARACTER FROM BASIC BRANCH IF NONE SYNTAX CHECK FOR COMMA EVALUATE EXPRESSION *6ET FPAØ EXPONENT, ADD 8 TO IT AND RESAVE IT - THIS
4176 4177 4178 4179 4181 4182 4183 4184 4185 4186 4187 4198 4199 4191 4195 4195 4196 4197 4198 4199 4201 4202 4203 4204 4205 4206 4207 4209 4210 4210 4211 4212	9E81 FB 16 31 F2 9E85 F4 FB 4A 51 9E89 EC 84 61 F9 9E80 E1 C7 78 AE 9E91 D4 DC 82 3B 9E99 B5 06 B5 06 9E99 B5 06 B5 06 9E90 81 40 9E97 26 02 9EA1 9D 9F 9EA3 BD 93 B2 9EA8 BD 93 B2 9EA9 BD 93 ID 9EAC AE C4 9EAE 9F CB 9EB0 AE 42 9EB2 9F CD 9EB4 BD 82 6D 9EB7 BD 87 3D 9EBA CB 02 9EB7 BD 87 3D 9EBA CB 03 9EBA CB 04 9EB7 BD 87 3D 9EBA CB 04 9EB7 BD 87 3D 9EBA CB 06 9EB7 BD 87 3D 9EBA CB 07 9EB8 BD 93 20 9EC2 86 01 9EC4 97 C2 9EC6 BD 95 81 9EC9 8E 01 00 9EC0 9D A5 9EC0 9D A5 9EC0 9D BB 26 D 9ED3 BD B2 6D	L9E81 L9E85 L9E89 L9E8D L9E91 L9E95 L9E99 * * CIRCL * THE C	FDB	\$FB16,\$31F2 \$F4FB,\$4A51 \$EC84,\$61F9 \$E1C7,\$78AE \$D4DC,\$8E3B \$C5E5,\$A269 \$B506,\$B506 IS ACTUALLY DRAWN AS A 64 SIDED I S COMPOSED OF 64 LINE COMMA #'@' L9EA3 GETNCH L9522 L93B2 L931D ,U VCB \$02,U VCB \$W2,U VCB \$YNCOMMA LB73D #YCF ,U L9320 #\$501 SETFLG L9581 #\$100 GETCCH L9EDF SYNCOMMA LB141 FPØEXP #\$08	SUBARC 1 SUBARC 2 SUBARC 3 SUBARC 4 SUBARC 5 SUBARC 6 SUBARC 6 SUBARC 7 SUBARC 8 NDS CHECK FOR @ SIGN SKIP IF NOT GET ANOTHER CHARACTER FROM BASIC GET MAX HOR & VER COURD VALUES AND PUT THEM IN VD3 AND VD5 GET HOR & VER CENTER COORDS AND PUT IN HORBEG, VERBEG NORMALIZE START COORDS FOR PROPER PMODE GET HOR COORD SAVE IT GET VERT COORD SAVE IT SYNTAX CHECK FOR COMMA EVALUATE EXPRESSION RETURN VALUE IN X POINT U TO TEMP DATA STORAGE SAVE RADIUS NORMALIZE RADIUS SET TO PSET SAVE PRADIUS SET TO PSET SAVE PRADIUS GO EVALUATE COLOR EXPRESSION AND SAVE IN WCOLOR HEIGHT/WIDTH RATIO DEFAULT VALUE GET AN INPUT CHARACTER FROM BASIC BRANCH IF NONE SYNTAX CHECK FOR COMMA EVALUATE EXPRESSION
4176 4177 4178 4179 4180 4181 4182 4183 4184 4185 4186 4187 4188 4189 4191 4192 4193 4194 4195 4196 4197 4198 4202 4203 4204 4205 4206 4207 4208 4209 4211 4212 4213 4214	9E81 FB 16 31 F2 9E85 F4 FB 4A 51 9E89 EC 84 61 F9 9E80 E1 C7 78 AE 9E91 D4 DC 88 3B 9E95 C5 E5 A2 69 9E99 B5 Ø6 B5 Ø6 9E99 B1 4Ø 9E97 E6 Ø2 9EA1 9D 9F 9EA3 BD 95 22 9EA6 BD 93 B2 9EA9 BD 95 B2 9EA9 BD 97 B2 9EA9 BD 97 B2 9EA9 BD 97 B2 9EA9 BD 87 3D 9EBA CE Ø2 9EBA BD 93 B2 9EBA BD 95 CB 9EBA BD 93 B2 9EA9 BD 97 CB 9EBA BD 82 GD 9EBA CE Ø2 9EBA BD 87 3D 9EBA CE Ø2 9EBA BD 87 3D 9EBA CE Ø2 9EBA BD 87 3D 9EBA CE Ø3 CF 9EBB AF C4 9EBB AF C4 9EBB AF C4 9EBF BD 87 3D 9EBA CE Ø3 CF 9EBB AF C4 9EBF BD 87 3D 9EBA CE Ø3 CF 9EBB BF 3B 9ECC BA Ø1 9ECC BD AF 9ECC BD BB B1 41 9ECB BB BB BB BB 9EDA 97 4F 9EDB BB BB Ø8 9EDA 97 4F	L9EB1 L9EB9 L9EB9 L9EB0 L9E91 L9E95 L9E99 * CIRCL * THE C * POLYG CIRCLE	FDB	\$FB16,\$31F2 \$F4FB,\$4A51 \$EC84,\$61F9 \$E1C7,\$78AE \$D4DC,\$8E3B \$C5E5,\$A269 \$B506,\$B506 IS ACTUALLY DRAWN AS A 64 SIDED T IS COMPOSED OF 64 LINE COMMA #'@' L9EA3 GETNCH L9522 L93B2 L93B2 L931D ,U VCB \$VCD SYNCOMMA LB73D #VCF ,U L9220 #\$01 ETFLG L9521 #\$100 GETCCH L9522 #\$01 ETFLG L9525 #\$01 ETFLG L9521 #\$100 FFCCH L9527 #\$01 ETFLG L9528 #\$100 FFCCH L9529 #\$01 ETFLG L9528 #\$100 FFCCH L9529 #\$100 FFCCH L9529 #\$100 FFCCH L9529 #\$200 FFCCH L9520 #\$200 FFCCH L9520 F	SUBARC 1 SUBARC 2 SUBARC 3 SUBARC 4 SUBARC 5 SUBARC 6 SUBARC 6 SUBARC 7 SUBARC 8 NDS CHECK FOR @ SIGN SKIP IF NOT GET ANOTHER CHARACTER FROM BASIC GET MAX HOR & VER COORD VALUES AND PUT THEM IN VD3 AND VD5 GET HOR 8 VER CENTER COORDS AND PUT IN HORBEG, VERBEG NORMALIZE START COORDS FOR PROPER PMODE GET HOR COORD SAVE IT GET VERI COORD SAVE IT SYNTAX CHECK FOR COMMA EVALUATE EXPRESSION RETURN VALUE IN X POINT U TO TEMP DATA STORAGE SAVE RADIUS NORMALIZE RADIUS SET TO PSET SAVE PSET/PRESET FLAG GO EVALUATE COLOR EXPRESSION AND SAVE IN WCOLOR HEIGHT/WIDTH RATIO DEFAULT VALUE GET AN INPUT CHARACTER FROM BASIC BRANCH IF NONE SYNTAX CHECK FOR COMMA EVALUATE EXPRESSION **ET PFAØ EXPONENT, ADD 8 TO IT AND RESAVE IT - THIS **WILL EFFECTIVELY MULTIPLY FPAØ BY 256. * EVALUATE EXPRESSION, RETURN VALUE IN X
4176 4177 4178 4179 4180 4181 4182 4183 4184 4185 4186 4187 4188 4199 4190 4191 4192 4193 4194 4195 4196 4197 4202 4203 4204 4206 4207 4208 4209 4210 4211 4212 4213 4214	9E81 FB 16 31 F2 9E85 F4 FB 4A 51 9E89 EC 84 61 F9 9E80 E1 C7 78 AE 9E91 D4 DC 8E 3B 9E95 C5 66 B5 Ø6 9E99 B5 Ø6 B5 Ø6 9E99 B5 Ø6 B5 Ø6 9E99 B7 46 Ø2 9EA1 9D 9F 9EA3 BD 93 B2 9EA9 BD 93 B2 9EA9 BD 93 ID 9EAC AE C4 9EAE 9F CB 9EB0 AE 42 9EB2 PF CD 9EB4 BD B2 6D 9EB7 BD B7 3D 9EBA CB Ø2 9EB7 BD B7 3D 9EBA CB Ø2 9EB8 BD 93 2Ø 9EB8 BD 93 2Ø 9EB8 BD 93 2Ø 9EB8 BD 95 B1 9ECC 8D 95 B1 9ECC 9D A5 9ECC 9T OF 9EB0 BD B2 6D 9ED3 BD B1 41 9ED6 96 4F 9ED8 BB Ø8 9EDA 97 4F 9EDB BB Ø8 9EDA 97 4F 9EDB BB 77 4Ø 9EDF BB BB 77 4Ø 9EDF BB BB 77 4Ø 9EDF BB BB 77 4Ø 9EDB BB Ø8 9EDA 97 4F 9EDB BB 77 4Ø 9EDF BB BB 77 4Ø	L9E81 L9E85 L9E89 L9E8D L9E91 L9E95 L9E99 * * CIRCL * THE C	FDB	\$FB16,\$31F2 \$F4FB,\$4A51 \$EC84,\$61F9 \$E1C7,\$78AE \$D4DC,\$8E3B \$C5E5,\$A269 \$B506,\$B506 IS ACTUALLY DRAWN AS A 64 SIDED I S COMPOSED OF 64 LINE COMMA #'@' 19EA3 GETNCH 19522 193B2 1931D 10 VCB \$02,U VCB \$02,U VCB \$VCD SYNCOMMA 1B73D #VCF ,U 19320 #\$\$01 SETFLG 19581 #\$100 GETCCH 19EDF \$YNCOMMA 1B141 FPØEXP #\$08 FPØEXP #\$08 FPØEXP #\$08 FPØEXP #\$08 FPØEXP 1B740 PMODE	SUBARC 1 SUBARC 2 SUBARC 3 SUBARC 4 SUBARC 5 SUBARC 6 SUBARC 6 SUBARC 7 SUBARC 8 NDS CHECK FOR @ SIGN SKIP IF NOT GET ANOTHER CHARACTER FROM BASIC GET MAX HOR & VER COURD VALUES AND PUT THEM IN VD3 AND VD5 GET HOR & VER CENTER COORDS AND PUT IN HORBEG, VERBEG NORMALIZE START COORDS FOR PROPER PMODE GET HOR GORD SAVE IT GET VERT COORD SAVE IT SVITAX CHECK FOR COMMA EVALUATE EXPRESSION RETURN VALUE IN X POINT UT OT TEMP DATA STORAGE SAVE RADIUS NORMALIZE RADIUS SET TO PSET SAVE PSET/PRESET FLAG GO EVALUATE COLOR EXPRESSION AND SAVE IN WCOLOR HEIGHT/MIDTH RATIO DEFAULT VALUE GET AN INPUT CHARACTER FROM BASIC BRANCH IF NONE SYNTAX CHECK FOR COMMA EVALUATE EXPRESSION *GET FPAØ EXPONENT, ADD 8 TO IT AND RESAVE IT - THIS **WILL EFFECTIVELY MULTIPLY FPAØ BY 256. ** **WILL EFFECTIVELY MULTIPLY FPAØ BY 256. ** EVALUATE EXPRESSION, RETURN VALUE IN X GET CURRENT PMODE VALUE
4176 4177 4178 4179 4180 4181 4182 4183 4184 4185 4186 4187 4188 4189 4191 4192 4193 4194 4195 4196 4197 4198 4202 4203 4204 4205 4206 4207 4208 4209 4211 4212 4213 4214	9E81 FB 16 31 F2 9E85 F4 FB 4A 51 9E89 EC 84 61 F9 9E80 E1 C7 78 AE 9E91 D4 DC 88 3B 9E95 C5 E5 A2 69 9E99 B5 Ø6 B5 Ø6 9E99 B1 4Ø 9E97 C6 W2 9E91 B9 F C8 9E98 B9 B9 B2 9EA3 BD 95 22 9EA6 BD 93 B2 9EA9 BD 93 1D 9EAC AE C4 9EAE 9F CB 9EB0 BF CB 9EB0 BF 3D 9EBA CE ØØ CF 9EB0 BF 3D 9EBA CE ØØ CF 9EB0 AF C4 9EBF BD B7 3D 9EBA CE ØØ CF 9EBD AF C4 9EBF BD S7 3D 9EBA CE ØØ CF 9EBD AF C4 9EBF BD S7 3D 9EBA CE ØØ CF 9EBD AF C4 9EBF BD S7 3D 9EBA C6 ØØ CF 9EBD BF 3D 9ECC B6 Ø1 9ECC B7 B5 9ECC B7 B5 9ECØ BD B7 4B 9ECØ BD B7 9ECØ BD B	L9EB1 L9EB9 L9EB9 L9EB0 L9E91 L9E95 L9E99 * CIRCL * THE C * POLYG CIRCLE	FDB	\$FB16,\$31F2 \$F4FB,\$4A51 \$EC84,\$61F9 \$E1C7,\$78AE \$D4DC,\$8E3B \$C5E5,\$A269 \$B506,\$B506 IS ACTUALLY DRAWN AS A 64 SIDED T IS COMPOSED OF 64 LINE COMMA #'@' L9EA3 GETNCH L9522 L93B2 L93B2 L931D ,U VCB \$VCD SYNCOMMA LB73D #VCF ,U L9220 #\$01 ETFLG L9521 #\$100 GETCCH L9522 #\$01 ETFLG L9525 #\$01 ETFLG L9521 #\$100 FFCCH L9527 #\$01 ETFLG L9528 #\$100 FFCCH L9529 #\$01 ETFLG L9528 #\$100 FFCCH L9529 #\$100 FFCCH L9529 #\$100 FFCCH L9529 #\$200 FFCCH L9520 #\$200 FFCCH L9520 F	SUBARC 1 SUBARC 2 SUBARC 3 SUBARC 4 SUBARC 5 SUBARC 6 SUBARC 6 SUBARC 7 SUBARC 8 NDS CHECK FOR @ SIGN SKIP IF NOT GET ANOTHER CHARACTER FROM BASIC GET MAX HOR & VER COORD VALUES AND PUT THEM IN VD3 AND VD5 GET HOR 8 VER CENTER COORDS AND PUT IN HORBEG, VERBEG NORMALIZE START COORDS FOR PROPER PMODE GET HOR COORD SAVE IT GET VERI COORD SAVE IT SYNTAX CHECK FOR COMMA EVALUATE EXPRESSION RETURN VALUE IN X POINT U TO TEMP DATA STORAGE SAVE RADIUS NORMALIZE RADIUS SET TO PSET SAVE PSET/PRESET FLAG GO EVALUATE COLOR EXPRESSION AND SAVE IN WCOLOR HEIGHT/WIDTH RATIO DEFAULT VALUE GET AN INPUT CHARACTER FROM BASIC BRANCH IF NONE SYNTAX CHECK FOR COMMA EVALUATE EXPRESSION **ET PFAØ EXPONENT, ADD 8 TO IT AND RESAVE IT - THIS **WILL EFFECTIVELY MULTIPLY FPAØ BY 256. * EVALUATE EXPRESSION, RETURN VALUE IN X
4176 4177 4178 4179 4180 4181 4182 4183 4184 4185 4186 4187 4188 4199 4191 4191 4192 4193 4194 4195 4196 4197 4201 4202 4203 4204 4205 4206 4207 4208 4209 4210 4211 4212 4213 4214 4215 4216 4217 4218	9881 FB 16 31 F2 9885 F4 FB 4A 51 9880 EC 84 61 F9 9880 E1 C7 78 AE 9891 D4 DC 8E 3B 9899 B5 Ø6 B5 Ø6 9899 B5 Ø6 B5 Ø6 9899 B5 Ø6 B5 Ø6 9899 B7 Ø6 B7 Ø6 9890 B1 4Ø 9897 E6 Ø2 9841 B0 97 9843 BD 95 22 9848 BD 93 1D 9840 BB 8D 93 1D 9840 BB 8D 93 1D 9840 BB 96 B2 6D 9887 BD B7 6B 9888 BD 93 2Ø 99888 BD 94 80 99888 BB 88 9989 97 4F 9888 BB Ø8 9988 988 988 9988 988 988 9889 988 988	L9EB1 L9EB9 L9EB9 L9EB0 L9E91 L9E95 L9E99 * CIRCL * THE C * POLYG CIRCLE	FDB	\$FB16,\$31F2 \$F4FB,\$4A51 \$EC64,\$61F9 \$E1C7,\$78AE \$D4DC,\$8E3B \$C5E5,\$A2E9 \$B506,\$B506 IS ACTUALLY DRAWN AS A 64 SIDED T IS COMPOSED OF 64 LINE COMMA #'@' L9EA3 GETNCH L9522 L931D ,U VCB \$02,U VCB \$02,U VCB \$102,U VCB \$102,U VCB \$103,U VCB \$104,U VCB \$105,U VCB \$106,U VCB \$107,U VCB \$108,U VCB \$108	SUBARC 1 SUBARC 2 SUBARC 3 SUBARC 4 SUBARC 5 SUBARC 6 SUBARC 6 SUBARC 7 SUBARC 8 NDS CHECK FOR @ SIGN SKIP IF NOT GET ANOTHER CHARACTER FROM BASIC GET MAX HOR & VER COURD VALUES AND PUT THEM IN VD3 AND VD5 GET HOR & VER CENTER COORDS AND PUT IN HORBEG, VERBEG NORMALIZE START COORDS FOR PROPER PMODE GET HOR GOORD SAVE IT GET VERT COORD SAVE IT SYNTAX CHECK FOR COMMA EVALUATE EXPRESSION RETURN VALUE IN X POINT U TO TEMP DATA STORAGE SAVE RADIUS NORMALIZE RADIUS SET TO PSET SAVE PSET/PRESET FLAG GO EVALUATE COLOR EXPRESSION AND SAVE IN WCOLOR HEIGHT/WIDTH RATIO DEFAULT VALUE GET AN INPUT CHARACTER FROM BASIC BRANCH IF NONE SYNTAX CHECK FOR COMMA EVALUATE EXPRESSION *GET FPAØ EXPONENT, ADD 8 TO IT AND RESAVE IT - THIS ************************************
4176 4177 4178 4179 4180 4181 4182 4183 4184 4185 4186 4187 4188 4199 4190 4191 4192 4193 4194 4195 4196 4201 4202 4203 4204 4207 4208 4211 4212 4213 4214 4215 4216	9E81 FB 16 31 F2 9E85 F4 FB 4A 51 9E89 EC 84 61 F9 9E80 E1 C7 78 AE 9E91 D4 DC 88 3B 9E95 C5 E5 A2 69 9E99 B5 Ø6 B5 Ø6 9E99 B1 4Ø 9E97 C6 W2 9E91 B9 F C8 9E98 B9 B9 B2 9EA3 BD 95 22 9EA6 BD 93 B2 9EA9 BD 93 1D 9EAC AE C4 9EAE 9F CB 9EB0 BF CB 9EB0 BF 3D 9EBA CE ØØ CF 9EB0 BF 3D 9EBA CE ØØ CF 9EB0 AF C4 9EBF BD B7 3D 9EBA CE ØØ CF 9EBD AF C4 9EBF BD S7 3D 9EBA CE ØØ CF 9EBD AF C4 9EBF BD S7 3D 9EBA CE ØØ CF 9EBD AF C4 9EBF BD S7 3D 9EBA C6 ØØ CF 9EBD BF 3D 9ECC B6 Ø1 9ECC B7 B5 9ECC B7 B5 9ECØ BD B7 4B 9ECØ BD B7 9ECØ BD B	L9EB1 L9EB9 L9EB9 L9EB0 L9E91 L9E95 L9E99 * CIRCL * THE C * POLYG CIRCLE	FDB	\$FB16,\$31F2 \$F4FB,\$4A51 \$EC64,\$61F9 \$E1C7,\$78AE \$D4DC,\$8E3B \$C5E5,\$A269 \$B506,\$B506 IS ACTUALLY DRAWN AS A 64 SIDED T IS COMPOSED OF 64 LINE COMMA #'@' L9EA3 GETNCH L9522 L9382 L931D ,U VCB \$YNCOMMA LB73D #VCF ,U USPECH L9528 #\$01 SETFLG L9581 #\$100 GETCCH L95PF SYNCOMMA LB141 FFP0EXP #\$08 FP0EXP #\$08 FP0EXP #\$08 PPOEXP L9740 PMODE #\$02 L95E9 X,D D,X VD1	SUBARC 1 SUBARC 2 SUBARC 3 SUBARC 4 SUBARC 5 SUBARC 6 SUBARC 6 SUBARC 7 SUBARC 8 NDS CHECK FOR @ SIGN SKIP IF NOT GET ANOTHER CHARACTER FROM BASIC GET MAX HOR & VER COORD VALUES AND PUT THEM IN VD3 AND VD5 GET HOR 8 VER CENTER COORDS AND PUT IN HORBEG, VERBEG NORMALIZE START COORDS FOR PROPER PMODE GET HOR COORD SAVE IT SYNTAX CHECK FOR COMMA EVALUATE EXPRESSION RETURN VALUE IN X POINT U TO TEMP DATA STORAGE SAVE RADIUS NORMALIZE RADIUS SET TO PSET SAVE PSET/PRESET FLAG GO EVALUATE COLOR EXPRESSION AND SAVE IN WCOLOR HEIGHT/WIDTH RATIO DEFAULT VALUE GET AN INPUT CHARACTER FROM BASIC BRANCH IF NONE SYNTAX CHECK FOR COMMA EVALUATE EXPRESSION *6ET FPAØ EXPONENT, ADD 8 TO IT AND RESAVE IT - THIS *WILL EFFECTIVELY MULTIPLY FPAØ BY 256. * EVALUATE EXPRESSION, RETURN VALUE IN X GET CURRENT PMODE VALUE TEST FOR PMODE Ø,1,4 BRANCH IF SO
4176 4177 4178 4179 4180 4181 4182 4183 4184 4185 4186 4187 4188 4199 4191 4191 4192 4193 4194 4295 4206 4207 4208 4207 4208 4207 4208 4210 4211 4212 4213 4214 4215 4216 4217 4218	9881 FB 16 31 F2 9885 F4 FB 4A 51 9880 EC 84 61 F9 9880 EC 84 61 F9 9880 E1 C7 78 AE 9999 B5 Ø6 B5 Ø6 9899 B7 Ø6 B7 Ø6 9890 B1 4Ø 9897 E6 Ø2 9840 B0 93 ID 9840 BB 95 CB 9880 BB 60 9880 BB	L9E81 L9E85 L9E80 L9E91 L9E95 L9E95 * * CIRCL * THE C * POLYS CIRCLE	FDB	\$FB16,\$31F2 \$F4FB,\$4A51 \$EC64,\$61F9 \$E1C7,\$78AE \$D4DC,\$8E3B \$C5E5,\$A2E9 \$B506,\$B506 IS ACTUALLY DRAWN AS A 64 SIDED T IS COMPOSED OF 64 LINE COMMA #'@' L9EA3 GETNCH L9522 L931D ,U VCB \$02,U VCB \$02,U VCB \$\$70,U VCB \$\$10,U VCB \$\$2,U VCB \$\$2,U VCB \$\$2,U VCB \$\$2,U VCB \$\$300 #\$501 #\$100	SUBARC 1 SUBARC 2 SUBARC 3 SUBARC 4 SUBARC 5 SUBARC 6 SUBARC 6 SUBARC 7 SUBARC 8 NDS CHECK FOR @ SIGN SKIP IF NOT GET ANOTHER CHARACTER FROM BASIC GET MAX HOR & VER COURD VALUES AND PUT THEM IN VD3 AND VD5 GET HOR & VER CENTER COORDS AND PUT IN HORBEG, VERBEG NORMALIZE START COORDS FOR PROPER PMODE GET HOR COORD SAVE IT GET VERT COORD SAVE IT SYNTAX CHECK FOR COMMA EVALUATE EXPRESSION RETURN VALUE IN X POINT U TO TEMP DATA STORAGE SAVE RADIUS NORMALIZE RADIUS SET TO PSET SAVE PSET/PRESET FLAG GO EVALUATE COLOR EXPRESSION AND SAVE IN WCOLOR HEIGHT/MIDTH RATIO DEFAULT VALUE GET AN INPUT CHARACTER FROM BASIC BRANCH IF NONE SYNTAX CHECK FOR COMMA EVALUATE EXPRESSION **GET CURRENT PMODE VALUE TEST FOR PMODES 0,1,4 BRANCH IF SO ** MULT X BY 2 -FOR PMODES 2,3 THE HOR PIXELS ARE 2X AS LONG AS ** PMODES Ø,1,4; MULT HW RATIO BY 2 TO COMPENSATE SAVE HW RATIO **
4176 4177 4178 4179 4180 4181 4182 4183 4184 4185 4186 4187 4188 4199 4190 4191 4192 4193 4194 4195 4196 4201 4202 4203 4204 4207 4208 4211 4212 4213 4214 4215 4216	9E81 FB 16 31 F2 9E85 F4 FB 4A 51 9E89 EC 84 61 F9 9E80 E1 C7 78 AE 9E91 D4 DC 88 3B 9E95 C5 E5 A2 69 9E99 B5 Ø6 B5 Ø6 9E99 B1 4Ø 9E97 C6 W2 9E91 P9 F 9EA3 BD 95 22 9EA6 BD 93 B2 9EA9 BD 93 1D 9EAC AE C4 9EAE 9F CB 9EB2 9F CD 9EB4 BD 87 3D 9EBA CE ØØ CF 9EBB AF C4 9EBF BD S7 3D 9EBA C6 ØØ CF 9EBB AF C4 9EBF BD B7 3D 9EBA C6 ØØ CF 9EBB AF C4 9EBF BD B7 3D 9EBA C6 ØØ CF 9EBB AF C4 9EBF BD AF C4 9	L9E81 L9E85 L9E80 L9E91 L9E95 L9E95 * * CIRCL * THE C * POLYS CIRCLE	FDB	\$FB16,\$31F2 \$F4FB,\$4A51 \$EC64,\$61F9 \$E1C7,\$78AE \$D4DC,\$8E3B \$C5E5,\$A269 \$B506,\$B506 IS ACTUALLY DRAWN AS A 64 SIDED T IS COMPOSED OF 64 LINE COMMA #'@' L9EA3 GETNCH L9522 L9382 L931D ,U VCB \$YNCOMMA LB73D #VCF ,U USPECH L9528 #\$01 SETFLG L9581 #\$100 GETCCH L95PF SYNCOMMA LB141 FFP0EXP #\$08 FP0EXP #\$08 FP0EXP #\$08 PPOEXP L9740 PMODE #\$02 L95E9 X,D D,X VD1	SUBARC 1 SUBARC 2 SUBARC 3 SUBARC 4 SUBARC 5 SUBARC 6 SUBARC 6 SUBARC 7 SUBARC 8 NDS CHECK FOR @ SIGN SKIP IF NOT GET ANOTHER CHARACTER FROM BASIC GET MAX HOR & VER COURD VALUES AND PUT THEM IN VD3 AND VD5 GET HOR & VER CENTER COORDS AND PUT IN HORBEG, VERBEG NORMALIZE START COORDS FOR PROPER PMODE GET HOR COORD SAVE IT SYNTAX CHECK FOR COMMA EVALUATE EXPRESSION RETURN VALUE IN X POINT U TO TEMP DATA STORAGE SAVE RADIUS NORMALIZE RADIUS SET TO PSET SAVE PRADIUS NORMALIZE RADIUS SET TO PSET SAVE PRESSION FROM BASIC BEANCH IF NONE SYNTAX CHECK FOR COMMA EVALUATE EXPRESSION AND SAVE IN WCOLOR HEIGHT/WIDTH RATIO DEFAULT VALUE GET AN INPUT CHARACTER FROM BASIC BRANCH IF NONE SYNTAX CHECK FOR COMMA EVALUATE EXPRESSION *GET FPAØ EXPONENT, ADD 8 TO IT AND RESAVE IT - THIS *WILL EFFECTIVELY MULTIPLY FPAØ BY 256. * WULL SEPPENDENT, ADD 8 TO IT AND RESAVE IT - THIS *WILL EFFECTIVELY MULTIPLY FPAØ BY 256. * EVALUATE EXPRESSION, RETURN VALUE IN X GET CURRENT PMODE VALUE TEST FOR PMODE Ø,1,4 BRANCH IF SO * MULT X BY 2 - FOR PMODES 2,3 THE HOR PIXELS ARE 2X AS LONG AS * PMODES Ø,1,4; MULT HW RATIO BY 2 TO COMPENSATE

4225	9EF4 34 Ø6		PSHS		SAVE START POINT
4226	9EF6 BD 9F E2			L9FE2	EVALUATE CIRCLE END POINT (OCTANT, SUBARC)
4227	9EF9 DD D9		STD		SAVE END POINT
4228	9EFB 35 Ø6		PULS		GET START POINT
4229	9EFD 34 Ø6	L9EFD	PSHS		STORE CURRENT CIRCLE POSITION
4230	9EFF 9E C3				* MOVE HOR, VER COORDS FROM HOREND, VEREND TO
4231	9FØ1 9F BD			HORBEG	* HORBEG, VERBEG R MOVE OLD END COORDINATES
4232 4233	9FØ3 9E C5			VEREND	* NEW START COORDINATES *
	9FØ5 9F BF			VERBEG	
	9FØ7 CE 9E 7B 9FØA 84 Ø1			#L9E79+2	POINT TO TABLE OF SINES & COSINES =GET OCTANT NUMBER
			ANDA		
	9FØC 27 Ø3			L9F11	=BRANCH IF EVEN
4237 4238	9FØE 5Ø 9FØF CB Ø8		NEGB ADDB	## ao	
4239	9F11 58	L9F11	ASLB	#400	*CONVERT Ø-7 TO 8-1 FOR ODD OCTANT NUMBERS
4240	9F12 58	Larii	ASLB		=FOUR BYTES/TABLE ENTRY
4241	9F13 33 C5		LEAU	RII	POINT U TO CORRECT TABLE ENTRY
4242	9F15 34 4Ø		PSHS		SAVE SIN/COS TABLE ENTRY
4243	9F17 BD 9F A7			L9FA7	CALCULATE HORIZ OFFSET
	9F1A 35 4Ø		PULS		GET SIN/COS TABLE PTR
4245	9F1C 33 5E				MOVE TO COSINE (VERT)
4246	9F1E 34 10		PSHS		SAVE HORIZ OFFSET
4247	9F2Ø BD 9F A7			L9FA7	CALCULATE VERT OFFSET
4248	9F23 35 2Ø		PULS		PUT HORIZ OFFSET IN Y
4249	9F25 A6 E4		LDA		*
4250	9F27 84 Ø3		ANDA		*
4251	9F29 27 Ø6			L9F31	*BRANCH IF OCTANT 0,3,4,7
	9F2B 81 Ø3		CMPA		*
	9F2D 27 Ø2			L9F31	*BRANCH IF OCTANT Ø,3,4,7
4254	9F2F 1E 12		EXG		SWAP HOR AND VERT OFFSETS
4255		L9F31			SAVE HORIZ OFFSET
4256				WILL ONLY MODIFY THE VERT COORD	
	9F33 1F 21		TFR		LOAD X WITH THE CALCULATED VERT OFFSET
	9F35 DC D1		LDD		GET HW RATIO
	→ 9F37 BD 9F B5			L9FB5	MULT VERT OFFSET BY HW RATIO
	9F3A 1F 2Ø		TFR		TRANSFER THE PRODUCT TO ACCD
	9F3C 4D		TSTA	.,-	CHECK OVERFLOW FLAG AND GET MSB RESULT
	9F3D 10 26 15 09			LB44A	FC ERROR IF RESULT > 255
	9F41 D7 C5		STB	LB44A VEREND	SAVE DELTA VER MBS
4264	9F43 1F 3Ø		TFR		LSB RESULT TO ACCA
4265	9F45 97 C6			VEREND+1	SAVE DELTA VER LSB
4266	9F47 A6 E4		LDA		*
4267	9F49 81 Ø2		CMPA	#\$02	* BRANCH IF OCTANT = Ø,1,6,7 (SUBARC HOR END
4268	9F4B 25 ØE			L9F5B	* POINT >= HOR CENTER)
	9F4D 81 Ø6		CMPA		= BRANCH IF OCTANT = Ø,1,6,7 (SUSARC HOR END
4270	9F4F 24 ØA		BCC		= POINT >= HOR CENTER)
4271	9F51 DC CB		LDD		GET HOR COORD OF CENTER
4272	9F53 93 C3			HOREND	SUBTRACT HORIZONTAL DIFFERENCE
4273	9F55 24 11			L9F68	BRANCH IF NO UNDERFLOW
4274	9F57 4F		CLRA		*
4275	9F58 5F		CLRB		* IF NEW HOR < Ø, FORCE IT TO BE Ø
4276	9F59 20 ØD			L9F68	SAVE NEW COORD
4277	9F5B DC CB	L9F5B	LDD		GET HOR COORD OF CENTER
4278	9F5D D3 C3			HOREND	ADD HORIZONTAL DIFFERENCE
4279	9F5F 25 Ø5			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 SUBARC END COORD
4284	9F6A A6 E4		LDA	,\$	*
4285	9F6C 81 Ø4		CMPA	#\$Ø4	* BRANCH IF OCTANT = Ø,1,2,3 (SUBARC VERT END
4286	9F6E 25 ØA		BL0	L9F7A	* POINT >= VERT CENTER)
4287	9F7Ø DC CD		LDD	VCD	GET VER COORD OF CENTER
	9F72 93 C5			VEREND	SUBTRACT VERTICAL DIFFERENCE
4289	9F74 24 11			L9F87	BRANCH IF NO UNDERFIOW
	9F76 4F		CLRA		*
4291	9F77 5F		CLRB		*IF NEW VERT < Ø, FORCE IT TO BE Ø
4292				L9F87	SAVE NEW COORD
	9F78 20 0D				
4293	9F7A DC CD	L9F7A	LDD		GET VER COORD OF CENTER
4293 4294	9F7A DC CD 9F7C D3 C5	L9F7A	LDD ADDD	VEREND	ADD VERTICAL DIFFERENCE
4293 4294 4295	9F7A DC CD 9F7C D3 C5 9F7E 25 Ø5	L9F7A	LDD ADDD BLO	VEREND L9F85	ADD VERTICAL DIFFERENCE BRANCH IF OVERFLOW
4293 4294 4295 4296	9F7A DC CD 9F7C D3 C5 9F7E 25 Ø5 9F8Ø 1Ø 93 D5	L9F7A	LDD ADDD BLO CMPD	VEREND L9F85 VD5	ADD VERTICAL DIFFERENCE BRANCH IF OVERFLOW COMPARE TO MAX VERT COORD
4293 4294 4295 4296 4297	9F7A DC CD 9F7C D3 C5 9F7E 25 Ø5 9F8Ø 1Ø 93 D5 9F83 25 Ø2		ADDD BLO CMPD BLO	VEREND L9F85 VD5 L9F87	ADD VERTICAL DIFFERENCE BRANCH IF OVERFLOW COMPARE TO MAX VERT COORD BRANCH IF < MAX VER
4293 4294 4295 4296 4297 4298	9F7A DC CD 9F7C D3 C5 9F7E 25 Ø5 9F8Ø 1Ø 93 D5 9F8Ø 25 Ø2 9F85 DC D5	L9F85	LDD ADDD BLO CMPD BLO LDD	VEREND L9F85 VD5 L9F87 VD5	ADD VERTICAL DIFFERENCE BRANCH IF OVERFLOW COMPARE TO MAX VERT COORD BRANCH IF < MAX VER GET MAX VERT COORD
4293 4294 4295 4296 4297 4298 4299	9F7A DC CD 9F7C D3 C5 9F7E 25 Ø5 9F80 1Ø 93 D5 9F83 DC D5 9F87 DD C5		LDD ADDD BLO CMPD BLO LDD STD	VEREND L9F85 VD5 L9F87 VD5 VEREND	ADD VERTICAL DIFFERENCE BRANCH IF OVERFLOW COMPARE TO MAX VERT COORD BRANCH IF < MAX VER GET MAX VERT COORD SAVE NEW VERT SUSARC END COORD
4293 4294 4295 4296 4297 4298 4299 4300	9F7A DC CD 9F7C D3 C5 9F7E 25 Ø5 9F8Ø 1Ø 93 D5 9F83 25 Ø2 9F85 DC D5 9F87 DD C5 9F89 ØD D8	L9F85	LDD ADDD BLO CMPD BLO LDD STD TST	VEREND L9F85 VD5 L9F87 VD5 VEREND VD8	ADD VERTICAL DIFFERENCE BRANCH IF OVERFLOW COMPARE TO MAX VERT COORD BRANCH IF < MAX VER GET MAX VERT COORD SAVE NEW VERT SUSARC END COORD CHECK FIRST TIME FLAG
4293 4294 4295 4296 4297 4298 4299 4300 4301	9F7A DC CD 9F7C D3 C5 9F7E 25 Ø5 9F80 1Ø 93 D5 9F83 DC D5 9F87 DD C5	L9F85 L9F87	LDD ADDD BLO CMPD BLO LDD STD	VEREND L9F85 VD5 L9F87 VD5 VEREND	ADD VERTICAL DIFFERENCE BRANCH IF OVERFLOW COMPARE TO MAX VERT COORD BRANCH IF < MAX VER GET MAX VERT COORD SAVE NEW VERT SUSARC END COORD CHECK FIRST TIME FLAG *DO NOT DRAW A LINE FIRST TIME THRU -
4293 4294 4295 4296 4297 4298 4299 4300 4301 4302	9F7A DC CD 9F7C D3 C5 9F7E 25 Ø5 9F8Ø 1Ø 93 D5 9F83 25 Ø2 9F85 DC D5 9F87 DD C5 9F89 ØD D8	L9F85	LDD ADDD BLO CMPD BLO LDD STD TST	VEREND L9F85 VD5 L9F87 VD5 VEREND VD8	ADD VERTICAL DIFFERENCE BRANCH IF OVERFLOW COMPARE TO MAX VERT COORD BRANCH IF < MAX VER GET MAX VERT COORD SAVE NEW VERT SUSARC END COORD CHECK FIRST TIME FLAG *DO NOT DRAW A LINE FIRST TIME THRU - *BECAUSE THE FIRST TIME YOU WOULD DRAW A LINE
4293 4294 4295 4296 4297 4298 4299 4300 4301 4302 4303	9F7A DC CD 9F7C D3 C5 9F7E 25 Ø5 9F8Ø 1Ø 93 D5 9F83 25 Ø2 9F85 DC D5 9F87 DD C5 9F89 ØD D8 9F88 26 Ø2	L9F85 L9F87 *	LDD ADDD BLO CMPD BLO LDD STD TST BNE	VEREND L9F85 VD5 L9F87 VD5 VEREND VD8 L9F8F	ADD VERTICAL DIFFERENCE BRANCH IF OVERFLOW COMPARE TO MAX VERT COORD BRANCH IF < MAX VER GET MAX VERT COORD SAVE NEW VERT SUSARC END COORD CHECK FIRST TIME FLAG *DO NOT DRAW A LINE FIRST TIME THRU - *BECAUSE THE FIRST TIME YOU WOULD DRAW A LINE *FROM THE CENTER TO THE FIRST POINT ON THE CIRCLE
4293 4294 4295 4296 4297 4298 4299 4300 4301 4302 4303 4304	9F7A DC CD 9F7C D3 C5 9F7E 25 Ø5 9F8Ø 1Ø 93 D5 9F83 25 Ø2 9F85 DC D5 9F87 DD C5 9F89 ØD D8 9F88 26 Ø2	L9F85 L9F87 *	ADDD BLO CMPD BLO LDD STD TST BNE	VEREND L9F85 VD5 L9F87 VD5 VEREND VD8 L9F8F	ADD VERTICAL DIFFERENCE BRANCH IF OVERFLOW COMPARE TO MAX VERT COORD BRANCH IF < MAX VER GET MAX VERT COORD SAVE NEW VERT SUSARC END COORD CHECK FIRST TIME FLAG *DO NOT DRAW A LINE FIRST TIME THRU - *BECAUSE THE FIRST TIME YOU WOULD DRAW A LINE *FROM THE CENTER TO THE FIRST POINT ON THE CIRCLE DRAW A LINE
4293 4294 4295 4296 4297 4298 4299 4300 4301 4302 4303 4304 4305	9F7A DC CD 9F7C D3 C5 9F7E 25 Ø5 9F8Ø 1Ø 93 D5 9F83 25 Ø2 9F85 DC D5 9F87 DD C5 9F89 ØD D8 9F8B 26 Ø2 9F8B 8D 5Ø 9F8F 35 Ø6	L9F85 L9F87 *	ADDD BLO CMPD BLO LDD STD TST BNE BSR PULS	VEREND L9F85 VD5 L9F87 VD5 VEREND VD8 L9F8F	ADD VERTICAL DIFFERENCE BRANCH IF OVERFLOW COMPARE TO MAX VER GET MAX VER GET MAX VER GET MAX VER GET MAX VERT COORD SAVE NEW VERT SUSARC END COORD CHECK FIRST TIME FLAG *DO NOT DRAW A LINE FIRST TIME THRU - *BECAUSE THE FIRST TIME YOU WOULD DRAW A LINE *FROM THE CENTER TO THE FIRST POINT ON THE CIRCLE DRAW A LINE GET END COORDS
4293 4294 4295 4296 4297 4298 4299 4300 4301 4302 4303 4304 4305 4306	9F7A DC CD 9F7C D3 C5 9F7E 25 Ø5 9F8Ø 1Ø 93 D5 9F83 25 Ø2 9F85 DC D5 9F87 DD C5 9F89 ØD D8 9F88 26 Ø2 9F8B 8D 5Ø 9F8F 35 Ø6 9F91 Ø4 D8	L9F85 L9F87 *	LDD ADDD BLO CMPD BLO LDD STD TST BNE BSR PULS LSR	VEREND L9785 VD5 L9787 VD5 VEREND VD8 L978F	ADD VERTICAL DIFFERENCE BRANCH IF OVERFLOW COMPARE TO MAX VERT COORD BRANCH IF < MAX VER GET MAX VERT COORD SAVE NEW VERT SUSARC END COORD CHECK FIRST TIME FLAG *DO NOT DRAW A LINE FIRST TIME THRU - *BECAUSE THE FIRST TIME YOU WOULD DRAW A LINE *FROM THE CENTER TO THE FIRST POINT ON THE CIRCLE DRAW A LINE GET END COORDS SHIFT FIRST TIME FLAG
4293 4294 4295 4296 4297 4298 4299 4300 4301 4302 4303 4304 4305 4305 4306 4307	9F7A DC CD 9F7C D3 C5 9F7E 25 Ø5 9F8Ø 1Ø 93 D5 9F83 25 Ø2 9F85 DC D5 9F87 DD C5 9F89 ØD D8 9F8B 26 Ø2 9F8B 8D 5Ø 9F8B 35 Ø6 9F91 Ø4 D8 9F93 25 Ø5	L9F85 L9F87 *	LDD ADDD BLO CMPD BLO LDD STD TST BNE BSR PULS LSR BLO	VEREND L9F85 VD5 L9F87 VD5 VEREND VD8 L9F8F	ADD VERTICAL DIFFERENCE BRANCH IF OVERFLOW COMPARE TO MAX VERT COORD BRANCH IF < MAX VER GET MAX VERT COORD SAVE NEW VERT COORD CHECK FIRST TIME FLAG *DO NOT DRAW A LINE FIRST TIME THRU - *BECAUSE THE FIRST TIME YOU WOULD DRAW A LINE *FROM THE CENTER TO THE FIRST POINT ON THE CIRCLE DRAW A LINE GET END COORDS SHIFT FIRST TIME FLAG DO NOT CHECK FOR END POINT AFTER DRAWING FIRST ARC
4293 4294 4295 4296 4297 4298 4299 4300 4301 4302 4303 4304 4305 4306 4307 4308	9F7A DC CD 9F7C D3 C5 9F7E 25 Ø5 9F8Ø 1Ø 93 D5 9F83 25 Ø2 9F85 DC D5 9F87 DD C5 9F89 ØD D8 9F88 26 Ø2 9F8D 8D 5Ø 9F8F 35 Ø6 9F91 Ø4 D8 9F93 25 Ø5 9F95 1Ø 93 D9	L9F85 L9F87 *	LDD ADDD BLO CMPD BLO LDD STD TST BNE BSR PULS LSR BLO CMPD	VEREND L9F85 VD5 L9F87 VD5 VEREND VD8 L9F8F L9F0F A,B VD8 L9F9A VD9	ADD VERTICAL DIFFERENCE BRANCH IF OVERFLOW COMPARE TO MAX VERT COORD BRANCH IF < MAX VER GET MAX VERT COORD SAVE NEW VERT SUSARC END COORD CHECK FIRST TIME FLAG *DO NOT DRAW A LINE FIRST TIME THRU - *BECAUSE THE FIRST TIME YOU WOULD DRAW A LINE *FROM THE CENTER TO THE FIRST POINT ON THE CIRCLE DRAW A LINE GET END COORDS SHIFT FIRST TIME FLAG DO NOT CHECK FOR END POINT AFTER DRAWING FIRST ARC COMPARE CURRENT POSITION TO END POINT
4293 4294 4295 4296 4297 4298 4299 4300 4301 4302 4303 4304 4305 4306 4307 4308 4308	9F7A DC CD 9F7C D3 C5 9F7E 25 Ø5 9F8Ø 1Ø 93 D5 9F83 25 Ø2 9F85 DC D5 9F87 DD C5 9F89 ØD D8 9F8B 26 Ø2 9F8B 8D 5Ø 9F8B 35 Ø6 9F91 Ø4 D8 9F93 25 Ø5	L9F85 L9F87 * * L9F8F	LDD ADDD BLO CMPD BLO LDD TST BNE BSR PULS LSR BLO CMPD BEQ	VEREND L9F85 VD5 L9F87 VD5 VEREND VD8 L9F8F L9F0F A,B VD8 L9F9A VD9 L9FA6	ADD VERTICAL DIFFERENCE BRANCH IF OVERFLOW COMPARE TO MAX VERT COORD BRANCH IF < MAX VER GET MAX VERT COORD SAVE NEW VERT SUSARC END COORD CHECK FIRST TIME FLAG *DO NOT DRAW A LINE FIRST TIME THRU - *BECAUSE THE FIRST TIME YOU WOULD DRAW A LINE *FROM THE CENTER TO THE FIRST POINT ON THE CIRCLE DRAW A LINE GET END COORDS SHIFT FIRST TIME FLAG DO NOT CHECK FOR END POINT AFTER DRAWING FIRST ARC COMPARE CURRENT POSITION TO END POINT CIRCLE DRAWING FINISHED
4293 4294 4295 4296 4297 4299 4300 4301 4302 4303 4304 4305 4306 4307 4308 4308 4308	9F7A DC CD 9F7C D3 C5 9F7C D3 C5 9F80 10 93 D5 9F83 25 02 9F85 DC D5 9F87 DD C5 9F89 0D D8 9F88 26 02 9F8D 8D 50 9F8D 8D 50 9F8D 8D 50 9F8T 35 06 9F91 04 D8 9F93 25 05 9F95 10 93 D9 9F98 27 0C	L9F85 L9F87 * * L9F8F	LDD ADDD BLO CMPD BLO STD TST BNE BSR PULS LSR BLO CMPD BEQ MENT S	VEREND L9F85 VD5 L9F87 VD5 VEREND VD8 L9F8F L9F0F A,B VD8 L9F9A VD9	ADD VERTICAL DIFFERENCE BRANCH IF OVERFLOW COMPARE TO MAX VER GET MAX VER GET MAX VER GET COORD SAVE NEW VERT SUSARC END COORD CHECK FIRST TIME FLAG *DO NOT DRAW A LINE FIRST TIME THRU - *BECAUSE THE FIRST TIME YOU WOULD DRAW A LINE *FROM THE CENTER TO THE FIRST POINT ON THE CIRCLE DRAW A LINE GET END COORDS SHIFT FIRST TIME FLAG DO NOT CHECK FOR END POINT AFTER DRAWING FIRST ARC COMPARE CURRENT POSITION TO END POINT CIRCLE DRAWING FINISHED T CTR
4293 4294 4295 4296 4297 4299 4300 4301 4302 4303 4304 4305 4306 4307 4308 4309 4310 4311	9F7A DC CD 9F7C D3 C5 9F7E 25 Ø5 9F8Ø 1Ø 93 D5 9F83 25 Ø2 9F85 DC D5 9F87 DD C5 9F89 ØD D8 9F88 26 Ø2 9F8D 8D 5Ø 9F8F 35 Ø6 9F91 Ø4 D8 9F93 25 Ø5 9F95 1Ø 93 D9 9F98 27 ØC	L9F85 L9F87 * * L9F8F	LDD ADDD BLO CMPD BLO LDD STD TST BNE BSR PULS LSR BLO CMPD BEQ MENT S INCB	VEREND L9F85 VD5 L9F87 VD5 VEREND VD8 L9F8F L9FDF A,B VD8 L9F9A VD9 L9FA6 JBARC CTR, IF > 7 THEN INCR OCTAN	ADD VERTICAL DIFFERENCE BRANCH IF OVERFLOW COMPARE TO MAX VER GET MAX VERT COORD BRANCH IF < MAX VER GET MAX VERT COORD SAVE NEW VERT SUSARC END COORD CHECK FIRST TIME FLAG *DO NOT DRAW A LINE FIRST TIME THRU - *BECAUSE THE FIRST TIME YOU WOULD DRAW A LINE *FROM THE CENTER TO THE FIRST POINT ON THE CIRCLE DRAW A LINE GET END COORDS SHIFT FIRST TIME FLAG DO NOT CHECK FOR END POINT AFTER DRAWING FIRST ARC COMPARE CURRENT POSITION TO END POINT CIRCLE DRAWING FINISHED T CTR INCR SUBARC CTR
4293 4294 4295 4296 4297 4299 4300 4301 4302 4303 4304 4305 4306 4307 4308 4309 4311 4311 4312	9F7A DC CD 9F7C D3 C5 9F7C D3 C5 9F8M 10 93 D5 9F83 25 02 9F85 DC D5 9F87 DD C5 9F89 0D D8 9F88 26 02 9F8D 8D 50 9F8B 35 06 9F91 04 D8 9F93 25 05 9F95 10 93 D9 9F98 27 0C 9F9B 5C 09	L9F85 L9F87 * * L9F8F	LDD ADDD BLO CMPD BLO LDD STD TST BNE BSR PULS LSR BLO CMPD BEQ MENT S INCB	VEREND L9F85 VD5 L9F87 VD5 VEREND VD8 L9F8F L9FDF A,B VD8 L9F9A VD9 L9FA6 JBARC CTR, IF > 7 THEN INCR OCTAN #\$08	ADD VERTICAL DIFFERENCE BRANCH IF OVERFLOW COMPARE TO MAX VER COORD BRANCH IF < MAX VER GET MAX VERT COORD SAVE NEW VERT COORD CHECK FIRST TIME FLAG *DO NOT DRAW A LINE FIRST TIME THRU - *BECAUSE THE FIRST TIME YOU WOULD DRAW A LINE *FROM THE CENTER TO THE FIRST POINT ON THE CIRCLE DRAW A LINE GET END COORDS SHIFT FIRST TIME FLAG DO NOT CHECK FOR END POINT AFTER DRAWING FIRST ARC COMPARE CURRENT POSITION TO END POINT CIRCLE DRAWING FINISHED T CTR INCR SUBARC CTR > 7?
4293 4294 4295 4296 4297 4298 4299 4300 4301 4302 4303 4304 4307 4307 4308 4307 4310 4311 4312 4313	9F7A DC CD 9F7C D3 C5 9F7C D3 C5 9F80 10 93 D5 9F83 25 02 9F85 DC D5 9F87 DD C5 9F89 0D D8 9F88 26 02 9F8D 8D 50 9F8B 35 06 9F91 04 D8 9F93 25 05 9F95 10 93 D9 9F98 27 0C 9F98 5C 9F98 C1 08 9F9D 26 04	L9F85 L9F87 * * L9F8F	LDD ADDD BLO CMPD BLO LDD STD TST BNE BSR PULS LSR BLO CMPD BEQ MENT S INCB EMBE BNE	VEREND L9F85 VD5 L9F87 VD5 VEREND VD8 L9F8F L9FDF A,B VD8 L9F9A VD9 L9FA6 JBARC CTR, IF > 7 THEN INCR OCTAN	ADD VERTICAL DIFFERENCE BRANCH IF OVERFLOW COMPARE TO MAX VER GET MAX VER GET COORD BRANCH IF < MAX VER GET MAX VERT COORD SAVE NEW VERT SUSARC END COORD CHECK FIRST TIME FLAG *DO NOT DRAW A LINE FIRST TIME THRU - *BECAUSE THE FIRST TIME YOU WOULD DRAW A LINE *FROM THE CENTER TO THE FIRST POINT ON THE CIRCLE DRAW A LINE GET END COORDS SHIFT FIRST TIME FLAG DO NOT CHECK FOR END POINT AFTER DRAWING FIRST ARC COMPARE CURRENT POSITION TO END POINT CIRCLE DRAWING FINISHED T CTR INCR SUBARC CTR > 7? NO
4293 4294 4295 4296 4297 4298 4299 4301 4302 4303 4305 4306 4307 4308 4308 4311 4312 4313 4314	9F7A DC CD 9F7C D3 C5 9F7E 25 Ø5 9F8Ø 1Ø 93 D5 9F83 25 Ø2 9F85 DC D5 9F87 DD C5 9F89 ØD D8 9F88 26 Ø2 9F8D 8D 5Ø 9F8F 35 Ø6 9F91 Ø4 D8 9F93 25 Ø5 9F95 1Ø 93 D9 9F98 27 ØC 9F9A 5C 9F9B C1 Ø8 9F9D 26 Ø4	L9F85 L9F87 * * L9F8F	LDD ADDD BLO CMPD BLO LDD TST BNE BSR PULS LSR BLO CMPD BEQ MENT S INCB CMPD BNE INCB	VEREND L9F85 VD5 L9F87 VD5 VEREND VD8 L9F8F L9FDF A,B VD8 L9F9A VD9 L9FA6 JBARC CTR, IF > 7 THEN INCR OCTAN #\$08	ADD VERTICAL DIFFERENCE BRANCH IF OVERFLOW COMPARE TO MAX VER GET MAX VERT COORD BRANCH IF < MAX VER GET MAX VERT COORD SAVE NEW VERT SUSARC END COORD CHECK FIRST TIME FLAG *DO NOT DRAW A LINE FIRST TIME THRU - *BECAUSE THE FIRST TIME YOU WOULD DRAW A LINE *FROM THE CENTER TO THE FIRST POINT ON THE CIRCLE DRAW A LINE GET END COORDS SHIFT FIRST TIME FLAG DO NOT CHECK FOR END POINT AFTER DRAWING FIRST ARC COMPARE CURRENT POSITION TO END POINT CIRCLE DRAWING FINISHED T CTR INCR SUBARC CTR > 77 NO INCR OCTANT CTR
4293 4294 4295 4296 4297 4298 4299 4301 4302 4303 4304 4305 4307 4306 4307 4310 4311 4312 4313 4314 4315	9F7A DC CD 9F7C D3 C5 9F7C D3 C5 9F8M 10 93 D5 9F83 25 02 9F8B DC D5 9F87 DD C5 9F89 0D D8 9F8B 26 02 9F8D 8D 50 9F8T 35 06 9F91 04 D8 9F93 25 05 9F95 10 93 D9 9F98 27 0C 9F9B 5C 08 9F9B C1 08 9F9B C1 08 9F9F 4C 9F9F 4C	L9F85 L9F87 * * L9F8F	LDD ADDD BLO CMPD BLO LDD STD TST BNE BSR PULS LSR BLO CMPD BEQ MENT S INCB CMPB BNE INCA CLRB	VEREND L9F85 VD5 L9F87 VD5 VEREND VD8 L9F8F L9F0F A,B VD8 L9F9A VD9 L9FA6 JBARC CTR, IF > 7 THEN INCR OCTAN #\$08 L9FA3	ADD VERTICAL DIFFERENCE BRANCH IF OVERFLOW COMPARE TO MAX VER COORD BRANCH IF < MAX VER GET MAX VERT COORD SAVE NEW VERT SUSARC END COORD CHECK FIRST TIME FLAG *DO NOT DRAW A LINE FIRST TIME THRU - *BECAUSE THE FIRST TIME YOU WOULD DRAW A LINE *FROM THE CENTER TO THE FIRST POINT ON THE CIRCLE DRAW A LINE GET END COORDS SHIFT FIRST TIME FLAG DO NOT CHECK FOR END POINT AFTER DRAWING FIRST ARC COMPARE CURRENT POSITION TO END POINT CIRCLE DRAWING FINISHED T CTR INCR SUBARC CTR > 7? NO INCR OCTANT CTR RESET SUBARC CTR
4293 4294 4295 4296 4297 4298 4299 4301 4302 4303 4305 4306 4307 4308 4308 4311 4312 4313 4314	9F7A DC CD 9F7C D3 C5 9F7E 25 Ø5 9F8Ø 1Ø 93 D5 9F83 25 Ø2 9F85 DC D5 9F87 DD C5 9F89 ØD D8 9F88 26 Ø2 9F8D 8D 5Ø 9F8F 35 Ø6 9F91 Ø4 D8 9F93 25 Ø5 9F95 1Ø 93 D9 9F98 27 ØC 9F9A 5C 9F9B C1 Ø8 9F9D 26 Ø4	L9F85 L9F87 * * L9F8F	LDD ADDD BLO CMPD BLO LDD STD TST BNE BSR PULS LSR BLO CMPD BEQ MENT S INCB CMPB BNE INCA CLRB	VEREND L9F85 VD5 L9F87 VD5 VEREND VD8 L9F8F L9FDF A,B VD8 L9F9A VD9 L9FA6 JBARC CTR, IF > 7 THEN INCR OCTAN #\$08	ADD VERTICAL DIFFERENCE BRANCH IF OVERFLOW COMPARE TO MAX VER GET MAX VERT COORD BRANCH IF < MAX VER GET MAX VERT COORD SAVE NEW VERT SUSARC END COORD CHECK FIRST TIME FLAG *DO NOT DRAW A LINE FIRST TIME THRU - *BECAUSE THE FIRST TIME YOU WOULD DRAW A LINE *FROM THE CENTER TO THE FIRST POINT ON THE CIRCLE DRAW A LINE GET END COORDS SHIFT FIRST TIME FLAG DO NOT CHECK FOR END POINT AFTER DRAWING FIRST ARC COMPARE CURRENT POSITION TO END POINT CIRCLE DRAWING FINISHED T CTR INCR SUBARC CTR > 7? NO INCR OCTANT CTR RESET SUBARC CTP *KEEP IN RANGE OF 0-7; ONCE ACCA = B, THIS WILL MAKE ACCA = 0,
4293 4294 4295 4296 4297 4298 4299 4300 4301 4302 4303 4304 4307 4307 4308 4307 4310 4311 4312 4313 4314 4315 4316	9F7A DC CD 9F7C D3 C5 9F7C D3 C5 9F8M 10 93 D5 9F83 25 02 9F8B DC D5 9F87 DD C5 9F89 0D D8 9F8B 26 02 9F8D 8D 50 9F8T 35 06 9F91 04 D8 9F93 25 05 9F95 10 93 D9 9F98 27 0C 9F9B 5C 08 9F9B C1 08 9F9B C1 08 9F9F 4C 9F9F 4C	L9F85 L9F87 * * L9F8F * INCRE L9F9A	LDD ADDD BLO CMPD BLO LDD STD TST BNE BSR PULS LSR BLO CMPD BEQ MENT S INCB CMPB BNE INCA CLRB	VEREND L9F85 VD5 L9F87 VD5 VEREND VD8 L9F8F L9F0F A,B VD8 L9F9A VD9 L9FA6 JBARC CTR, IF > 7 THEN INCR OCTAN #\$08 L9FA3	ADD VERTICAL DIFFERENCE BRANCH IF OVERFLOW COMPARE TO MAX VER COORD BRANCH IF < MAX VER GET MAX VERT COORD SAVE NEW VERT SUSARC END COORD CHECK FIRST TIME FLAG *DO NOT DRAW A LINE FIRST TIME THRU - *BECAUSE THE FIRST TIME YOU WOULD DRAW A LINE *FROM THE CENTER TO THE FIRST POINT ON THE CIRCLE DRAW A LINE GET END COORDS SHIFT FIRST TIME FLAG DO NOT CHECK FOR END POINT AFTER DRAWING FIRST ARC COMPARE CURRENT POSITION TO END POINT CIRCLE DRAWING FINISHED T CTR INCR SUBARC CTR > 7? NO INCR OCTANT CTR RESET SUBARC CTR
4293 4294 4295 4296 4297 4298 4299 4301 4302 4303 4305 4306 4307 4311 4312 4313 4314 4315 4316 4317	9F7A DC CD 9F7C D3 C5 9F7E 25 Ø5 9F8Ø 1Ø 93 D5 9F83 25 Ø2 9F85 DC D5 9F87 DD C5 9F89 ØD D8 9F88 26 Ø2 9F8D 8D 5Ø 9F8F 35 Ø6 9F91 Ø4 D8 9F93 25 Ø5 9F95 1Ø 93 D9 9F98 C1 Ø8 9F98 C1 Ø8 9F99 C6 Ø4 9F9F 4C 9FAØ 5F 9FAI 84 Ø7	L9F85 L9F87 * * L9F8F * INCRE L9F9A	LDD ADDD BLO CMPD BLO LDD TST BNE BSR PULS LSR BLO CMPD BEQ MENT S INCB CMPD ENC	VEREND L9F85 VD5 L9F87 VD5 VEREND VD8 L9F8F L9F0F A,B VD8 L9F9A VD9 L9FA6 JBARC CTR, IF > 7 THEN INCR OCTAN #\$08 L9FA3	ADD VERTICAL DIFFERENCE BRANCH IF OVERFLOW COMPARE TO MAX VER GET MAX VERT COORD BRANCH IF < MAX VER GET MAX VERT COORD SAVE NEW VERT SUSARC END COORD CHECK FIRST TIME FLAG *DO NOT DRAW A LINE FIRST TIME THRU - *BECAUSE THE FIRST TIME YOU WOULD DRAW A LINE *FROM THE CENTER TO THE FIRST POINT ON THE CIRCLE DRAW A LINE GET END COORDS SHIFT FIRST TIME FLAG DO NOT CHECK FOR END POINT AFTER DRAWING FIRST ARC COMPARE CURRENT POSITION TO END POINT CIRCLE DRAWING FINISHED T CTR INCR SUBARC CTR > 7? NO INCR OCTANT CTR RESET SUBARC CTR *KEEP IN RANGE OF Ø-7; ONCE ACCA = B, THIS WILL MAKE ACCA = Ø, *SO THE END POINT WILL BE (Ø,Ø) AND THE CIRCLE ROUTINE WILL END.
4293 4294 4295 4296 4297 4298 4299 4301 4302 4303 4304 4305 4307 4308 4310 4311 4312 4313 4314 4315 4316 4317 4318	9F7A DC CD 9F7C D3 C5 9F8C D3 C5 9F8M 10 93 D5 9F83 25 02 9F8B DC D5 9F87 DD C5 9F89 0D D8 9F8B 26 02 9F8B 8D 50 9F8B 35 06 9F91 04 D8 9F93 25 05 9F95 10 93 D9 9F98 C1 08 9F9B C1 08	L9F85 L9F87 * * L9F8F * INCRE L9F9A	LDD ADDD BLO CMPD BLO LDD STD TST BNE BSR PULS LSR BLO BEQ MENT S INCB BNE INCA CMPD BNE INCA CMPD BNE CLRB ANDA JMP RTS	VEREND L9F85 VD5 L9F87 VD5 VEREND VD8 L9F8F L9F0F A,B VD8 L9F9A VD9 L9FA6 JBARC CTR, IF > 7 THEN INCR OCTAN #\$08 L9FA3	ADD VERTICAL DIFFERENCE BRANCH IF OVERFLOW COMPARE TO MAX VER GET MAX VERT COORD BRANCH IF < MAX VER GET MAX VERT COORD SAVE NEW VERT SUSARC END COORD CHECK FIRST TIME FLAG *DO NOT DRAW A LINE FIRST TIME THRU - *BECAUSE THE FIRST TIME YOU WOULD DRAW A LINE *FROM THE CENTER TO THE FIRST POINT ON THE CIRCLE DRAW A LINE GET END COORDS SHIFT FIRST TIME FLAG DO NOT CHECK FOR END POINT AFTER DRAWING FIRST ARC COMPARE CURRENT POSITION TO END POINT CIRCLE DRAWING FINISHED T CTR INCR SUBARC CTR > 7? NO INCR OCTANT CTR RESET SUBARC CTR **SO THE END POINT WILL BE (0,0) AND THE CIRCLE ROUTINE WILL END. KEEP DRAWING CIRCLE EXIT CIRCLE ROUTINE

EXTENDED BASIC UNRAVELLED II APPENDIX B ORIGIN:SPECTRAL ASSOC DISASSEMBLY OF EXTENDED BASIC 1.1 REVISED:12/26/1999 WALTER K ZYDHEK

4321	9FA7 9E CF	L9FA7	LDX	VCF	GET RADIUS
4322	9FA9 EC C4		LDD	,U	GET SIN/COS TABLE MODIFIER
4323	9FAB 27 Ø7			Ĺ9FB4	BRANCH IF = Ø - OFFSET = RADIUS
4324	9FAD 83 ØØ Ø1		SUBD		SUBTR 1
4325	9FBØ 8D Ø3			L9FB5	
					MULT RADIUS BY SIN/COS
4326	9FB2 1F 21		TFR	Υ,Χ	RETURN RESULT IN X REG
4327	9FB4 39	L9FB4	RTS		
4328		* MULTI	PLY (U	NSIGNED) TWO 16 BIT NUMBERS TOGE	THER -
4329		* FNTFR	WITH	ONE NUMBER IN ACCD, THE OTHER IN	X
4330				BYTE PRODUCT WILL BE STORED IN 4	
4331				N THE STACK). I.E. (AA AB) X (XH	
4332				6*(AA*XL+AB*XH)+AB*XL. THE 2 BYT	
4333		* MULTI	PLIER	AND MULTIPLICAND ARE TREATED AS	A 1
4334		* BYTE	INTEGE	R PART (MSB) WITH A 1 BYTE FRACT	IONAL 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			\$Ø4,S	RESET OVERFLOW FLAG
4337	9FB9 A6 63			\$Ø3,S	=
4338	9FBB 3D		MUL	400,0	=
4339	9FBC ED 66			*#* *	
				\$Ø6,S	= CALCULATE ACCB*XL, STORE RESULT IN 6,S
4340	9FBE EC 61			\$Ø1,S	*
4341	9FCØ 3D		MUL		* CALCULATE ACCB*XH
4342	9FC1 EB 66		ADDB	\$Ø6,S	=
4343	9FC3 89 ØØ		ADCA	#\$00	=
4344	9FC5 ED 65		STD	\$Ø5,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			\$Ø3,S	*
				¥W3,3	
4347	9FCB 3D		MUL		* CALCULATE ACCA*XL
4348	9FCC E3 65			\$Ø5,S	=
4349	9FCE ED 65		STD	\$Ø5,S	= ADD RESULT TO TOTAL OF 2 PREVIOUS MULTS
4350	9FDØ 24 Ø2		BCC	L9FD4	BRANCH IF NO OVERFLOW
4351	9FD2 6C 64		INC	\$Ø4,S	SET OVERFLOW FLAG (ACCD > \$FFFF)
4352	9FD4 A6 E4	L9FD4	LDA	,S	*
4353	9FD6 E6 62	25.5.		\$Ø2,S	*
4354	9FD8 3D		MUL	¥02,3	* CALCULATE ACCA*XH
				ta1 c	= CALCULATE ACCA"AN
4355	9FD9 E3 64			\$Ø4,S	
4356	9FDB ED 64			\$Ø4,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		* CALCU	LATE S	TART OR END POINT WHICH IS A NUM	BER FROM
4361				ED AS AN OCTANT NUMBER (Ø-7) AND	
4362	9FE2 5F	L9FE2	CLRB	ED NO AN OCTANT NOTBER (B 7) AND	DEFAULT VALUE OF ZERO
		LJFEZ		OFTOCII	
4363	9FE3 9D A5			GETCCH	GET CURRENT INPUT CHAR
4364	9FE5 27 11			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	FPØEXP	GET EXPONENT OF FPAØ
4368	9FEF 8B Ø6		ADDA	#\$Ø6	ADD 6 TO EXPONENT - MULTIPLY EXPONENT BY 64
4369	9FF1 97 4F			FPØEXP	RESAVE EXPONENT
4370	9FF3 BD B7 ØE			LB7ØE	CONVERT FPAØ TO INTEGER IN ACCB
4371	9FF6 C4 3F			#\$3F	MAX VALUE OF 63
4372	9FF8 1F 98	L9FF8		В,А	SAVE VALUE IN ACCA ALSO
4373	9FFA C4 Ø7			#\$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	ØØB5	GRPRAM	ØØBC	L81EB	81EB	L8316	8316	L857D	857D
ANGLE	ØØE8	HEXDOL	8BDD	L81FØ	81FØ	L831A	831A	L8581	8581
ARYDIS	0008	HORBEG	ØØBD	L81F2	81F2	L834D	834D	L858A	858A
ARYEND	ØØ1F	HORBYT	ØØB9	L81F4	81F4	L835E	835E	L858C	858C
ARYTAB	ØØ1D	HORDEF	ØØC7	L81F6	81F6	L8367	8367	L8592	8592
ATN	83BØ	HOREND	ØØC3	L81F8	81F8	L836C	836C	L859D	859D
BAKCOL	ØØB3	INSTR	877E	L81FA	81FA	L837E	837E	L85AB	85AB
BASIC	AØØØ	INT	BCEE	L81FC	81FC	L83A3	83A3	L85AF	85AF
BAWMST	AØE8	IRQVEC	Ø1ØC	L81FE	81FE	L83A6	83A6	L85B3	85B3
BEGGRP	ØØBA	L8ØØ2	8002	L82ØØ	8200	L83AB	83AB	L85B4	85B4
BINVAL	ØØ2B	L8Ø31	8031	L82Ø2	8202	L83B8	83B8	L85B6	85B6
BLKLEN	ØØ7D	L8ØDØ	8ØDØ	L82Ø4	8204	L83C5	83C5	L85C2	85C2
BLKTYP	ØØ7C	L8ØDD	8ØDD	L82Ø6	8206	L83D7	83D7	L85C3	85C3
CASBUF	Ø1DA	L8ØDE	8ØDE	L82Ø8	8208	L83DC	83DC	L85C7	85C7
CBUFAD	Ø07E	L8ØDF	8ØDF	L82ØA	82ØA	L83DF	83DF	L85D1	85D1
CFNBUF	Ø1D1	L8ØE1	8ØE1	L82ØC	82ØC	L83EØ	83EØ	L85D5	85D5
CHARAC	0001	L8ØE3	8ØE3	L82ØE	82ØE	L83E1	83E1	L85DE	85DE
CHARAD	ØØA6	L8ØE4	8ØE4	L821Ø	8210	L83E6	83E6	L85F3	85F3
CHGFLG	ØØDB	L8ØE6	8ØE6	L8212	8212	L83EB	83EB	L85F5	85F5
CINBFL	0070	L8ØE8	8ØE8	L8214	8214	L83FØ	83FØ	L86ØF	86ØF
CINCTR	ØØ79	L8ØFF	8ØFF	L8216	8216	L83F5	83F5	L8613	8613
CINPTR	ØØ7A	L8100	8100	L8218	8218	L83FA	83FA	L861E	861E
CIRCLE	9E9D	L81Ø1	8101	L821A	821A	L83FF	83FF	L8625	8625
COLOR	9542	L8112	8112	L821C	821C	L84Ø4	84Ø4	L8626	8626
COMVEC	Ø12Ø	L8113	8113	L821E	821E	L84Ø9	8409	L862E	862E
COS	8378	L811C	811C	L8221	8221	L84ØE	84ØE	L863Ø	8630
CSSVAL	ØØC1	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	897Ø	L8148	8148	L822D	822D	L841E	841E	L865Ø	865Ø
DEVNUM	ØØ6F	L8154	8154	L823Ø	8230	L8423	8423	L8659	8659
DEVPOS	ØØ6C	L8165	8165	L8233	8233	L8428	8428	L865C	865C
DLBAUD	ØØE6	L8168	8168	L8236	8236	L842D	842D	L8665	8665
DLOAD	8C18	L817Ø	8170	L823A	823A	L8432	8432	L866B	866B
DOSBAS	CØØØ	L817D	817D	L824Ø	8240	L8437	8437	L8679	8679
DOTVAL	ØØE5	L8183	8183	L8245	8245	L843C	843C	L867C	867C
DRAW	9CB6	L8186	8186	L824A	824A	L8441	8441	L8685	8685
	8533	L818A				L8489	8489	L8687	8687
EDIT			818A	L825Ø	8250				
ENDFLG	0000	L818E	818E	L8257	8257	L8491	8491	L8694	8694
ENDGRP	ØØB7	L8193	8193	L8259	8259	L84AC	84AC	L86A6	86A6
ESC	ØØ1B	L8196	8196	L825B	825B	L84C4	84C4	L86D6	86D6
EVALEXPB		L8199	8199	L825D	825D	L84C9	84C9	L86EB	86EB
EXBAS	8000	L819D	819D	L825F	825F	L84CA	84CA	L86FD	86FD
EXECJP	ØØ9D	L81A1	81A1	L8261	8261	L84CF	84CF	L87ØE	87ØE
EXP	84F2	L81A5	81A5	L8263	8263	L84D4	84D4	L8724	8724
EXPJMP	Ø11D	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	ØØB2	L81BC	81BC	L826B	826B	L84E8	84E8	L8746	8746
FPØEXP	004F	L81C2	81C2	L826D	826D	L84ED	84ED	L8748	8748
FPØSGN	0054	L81C7	8107	L826F	826F	L85Ø1	8501	L8768	8768
FP1SGN	0061	L81CA	81CA	L8271	8271	L85Ø4	8504	L876B	876B
FPAØ	ØØ5Ø	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	ØØA5	L81DF	81DF	L82F1	82F1	L855C	855C	L87CD	87CD
GETNCH	ØØ9F	L81E4	81E4	L831Ø	8310	L855D	855D	L87D6	87D6
GIVABF	B4F4	L81E9	81E9	L8311	8311	L857Ø	857Ø	L87D8	87D8
A (D)		_01_7	J	_0011	3011	_00,5	20,2	_0,50	0,00

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	8800	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	9011	L927B	927B
L882E	882E	L8AE1	8AE1	L8D72	8D72	L9Ø15	9015	L9281	9281
L8834	8834	L8AE5	8AE5	L8D7C	8D7C	L9Ø23	9023	L928E	928E
L8845	8845	L8AE9	8AE9	L8D89	8D89	L9Ø2C	9Ø2C	L928F	928F
L8862	8862	L8AEB	8AEB	L8D8B	8D8B	L9Ø5Ø	9050	L9298	9298
L8866	8866	L8AED	8AED	L8DA7	8DA7	L9Ø54	9Ø54	L929C	929C
L886E	886E	L8AEF	8AEF	L8DB8	8DB8	L9Ø6Ø	9060	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	9096	L92A4	92A4
L88D9	88D9	L8B24	8B24	L8DD4	8DD4	L9Ø9E	9Ø9E	L92A6	92A6
L88EF	88EF	L8B41	8B41	L8DDE	8DDE	L9ØA9	9ØA9	L92C2	9202
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	L8EØ4	8EØ4	L9ØBF	9ØBF	L92F4	92F4
L8943	8943	L8B8C	8B8C	L8EØC	8EØC	L9ØCB	9ØCB	L92FC	92FC
L8944	8944	L8BAE	8BAE	L8E1D	8E1D	L9ØE2	9ØE2	L93Ø3	9303
L8952	8952	L8BBE	8BBE	L8E25	8E25	L9ØEA	9ØEA	L93Ø9	9309
L8955	8955	L8BC9	8BC9	L8E37	8E37	L9ØEE	9ØEE	L9317	9317
L896Ø L898C	896Ø 898C	L8BD9 L8BE5	8BD9	L8E3B L8E5F	8E3B 8E5F	L9ØFF L91Ø8	9ØFF 91Ø8	L931A L931D	931A 931D
L898C L899Ø	899Ø	L8BEA	8BE5 8BEA	L8E69	8E69	L9108 L910D	9108 910D	L9310 L932Ø	9310
L8992	8992	L8BFF	8BFF	L8E71	8E71	L9116	9116	L9320 L932C	9320 9320
L8993	8993	L8CØ7	8CØ7	L8E82	8E82	L911B	911B	L9326	9338
L899F	899F	L8CØB	8CØB	L8E88	8E88	L9129	9129	L9349	9349
L89AE	89AE	L8C1B	8C1B	L8E95	8E95	L913Ø	9130	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	8096	L8ED2	8ED2	L9185	9185	L93B2	93B2
L8AØ2	8AØ2	L8C9B	8C9B	L8ED8	8ED8	L919E	919E		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	9420
L8A3D	8A3D	L8CC6	8006	L8F1A	8F1A	L91CC	91CC		9429
L8A67	8A67	L8CCD	8CCD	L8F2Ø	8F2Ø	L91CD	91CD		943Ø
L8A68	8A68	L8CDØ	8CDØ	L8F24		L91DØ	91DØ		9432
L8A71	8A71	L8CDD	8CDD	L8F26	8F26	L91E4	91E4		9434
L8A77	8A77	L8CE2	8CE2	L8F41	8F41	L91E9	91E9		9443
L8A83	8A83	L8CE4	8CE4	L8F4F	8F4F	L91F1	91F1		9444
L8A86	8A86	L8DØ1	8DØ1	L8F5A	8F5A	L9200	9200		9451
L8A9Ø	8A9Ø	L8DØ2	8DØ2	L8F74	8F74	L92Ø2	9202		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	9490	L971Ø	9710	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	9502	L979F	979F	L99C6	9906	L9CØA	9CØA	L9E81	9E81
L95Ø6	9506	L97AE	97AE	L99CB	99CB	L9C1B	9C1B	L9E85	9E85
L95ØD	95ØD	L97B7	97B7	L99DF	99DF	L9C27	9027	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	9092	L9EDF	9EDF
L9552	9552	L98ØC	98ØC	L9A32	9A32	L9CC6	9006	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	9600	L9864	9864	L9AF2	9AF2	L9D82	9D82	L9FDF	9FDF
L96Ø7	9607	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	L989Ø	989Ø	L9B1F	9B1F	L9D92	9D92	LA176	A176
L965Ø	965Ø	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	L99ØA	99ØA	L9B88	9B88	L9DE8	9DE8	LA5Ø5	A5Ø5
L97Ø6	97Ø6	L9931	9931	L9B8C	9B8C	L9DF2	9DF2	LA578	A578

LA59A	A59A	LB4F3	B4F3	PAINT	98EC	V4D	ØØ4D
LA5A5	A5A5	LB5ØF	B5ØF	PCLEAR	968B	VARPTR	86BE
LA5AE	A5AE	LB518	B518	PCLS	9532	VALTMP	0006
LASC7	A5C7	LB51A	B51A	PCOPY	9723	VARDES	ØØ3B
LASE4	A5E4	LB56D	B56D	PIAØ	FFØØ	VARPTR	ØØ39
LA616	A616	LB643	B643	PIA1	FF2Ø	VARTAB	ØØ1B
LA619	A619	LB654	B654	PLAY	9A22	VCB	ØØCB
LA635	A635	LB657	B657	PLYTMR	ØØE3	VCD	ØØCD
LA644	A644	LB659	B659	PMOD	9621	VCF	ØØCF
LA648	A648	LB69B	B69B	PMODE	ØØB6	VD1	ØØD1
LA65F	A65F	LB6A4	B6A4	POS	86AC	VD3	ØØD3
LA7D8	A7D8	LB6AD	B6AD	PPOINT	9339	VD4	ØØD4
LA7E9	A7E9	LB7ØE	B7ØE	PRESET	9365	VD5	ØØD5
LA7F4	A7F4	LB734	B734	PSET	9361	VD6	ØØD6
LA714	A974	LB738	B734	PUT	9758	VD7	ØØD7
LA976	A976	LB73D	B73D	RELFLG	ØØØA	VD8	ØØD8
LA9A2	A9A2	LB74Ø	B74Ø	RENUM	8AØ9	VD9	ØØD9
LA9BB	A9BB	LB7C2	B7C2	RESET	FFFE	VDA	ØØDA
LAC1E	AC1E	LB958	B958	RESSGN	ØØ62	VERBEG	ØØBF
LAC33	AC33	LB95C	B95C	RSTVEC	0072	VERDEF	ØØC9
LAC46	AC46	LB99F	B99F	RVEC15	Ø18B	VEREND	ØØC5
LAC6Ø	AC6Ø	LB9AC	B9AC	RVEC17	Ø191	VOLHI	ØØDF
LAC73	AC73	LB9AF	B9AF	RVEC18	Ø194	VOLLOW	ØØEØ
LAC7C	AC7C	LB9B4	B9B4	RVEC19	Ø197	WCOLOR	ØØB4
LACA8	ACA8	LB9B9	B9B9	RVEC2Ø	Ø19A	XBWMST	8ØCØ
LACEF	ACEF	LB9C2	B9C2	RVEC22	Ø1AØ	XIRQSV	894C
	ACF1		BA1C		Ø1A3		8846
LACF1		LBA1C		RVEC23		XVEC15	
LADØ1	ADØ1	LBA3A	BA3A	RVEC3	Ø167	XVEC17	88FØ
LAD19	AD19	LBA92	BA92	RVEC4	Ø16A	XVEC18	829C
LAD21	AD21	LBACA	BACA	RVEC8	Ø176	XVEC19	87E5
LAD26	AD26	LBB48	BB48	RVEC9	Ø179	XVEC2Ø	82B9
LAD33	AD33	LBB5C	BB5C	SAM	FFCØ	XVEC23	83Ø4
LAD9E	AD9E	LBB6A	BB6A	SCALE	ØØE9	XVEC3	8273
LADC6	ADC6	LBB82	BB82	SCREEN	967Ø	XVEC4	8CF1
LADD4	ADD4	LBB8F	BB8F	SETFLG	ØØC2	XVEC8	8286
LADEB	ADEB	LBC14	BC14	SQR	8480	XVEC9	8E9Ø
LAE15	AE15	LBC2F	BC2F	STRINOUT	B99C	ZERO	ØØ8A
LAED2	AED2	LBC35	BC35	SYNCOMMA	B26D		22011
LAEEØ	AEEØ	LBC4C	BC4C	TAN	8381		
LAF67	AF67	LBC5F	BC5F	TEMPO	ØØE2		
	AFA4	LBC6D			ØØØF		
LAFA4			BC6D	TEMPTR			
LBØ35	BØ35	LBCAØ	BCAØ	TIMER	8968		
LB141	B141	LBCC8	BCC8	TIMOUT	ØØE7		
LB143	B143	LBD99	BD99	TIMVAL	Ø112		
LB146	B146	LBDC5	BDC5	TINPTR	ØØ2F		
LB156	B156	LBDCC	BDCC	TMPSTK	ØØDC		
LB158	B158	LBDD9	BDD9	TRCFLG	ØØAF		
LB244	B244	LBEE9	BEE9	TROFF	86A8		
LB262	B262	LBEFØ	BEFØ	TRON	86A7		
LB267	B267	LBEFF	BEFF	TXTTAB	0019		
LB26A	B26A	LBF78	BF78	USRØ	Ø13E		
LB26F	B26F	LBFA6	BFA6	USRADR	ØØBØ		
LB207	B277	LCØØ2	CØØ2	V4Ø	ØØ4Ø		
LB284	B284	LINBUF	Ø2DC	V41	0041		
LB2CE	B2CE	LINE	93BB	V43	0043		
LB357	B357	LOG	8446	V45	0045		
LB35C	B35C	MEMSIZ	0027	V47	0047		
LB3A2	B3A2	NOTELN	ØØE1	V4A	ØØ4A		
LB44A	B44A	OCTAVE	ØØDE	V4B	ØØ4B		

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

EXTENDED BASIC UNRAVELLED II		APPENDIX	D	ORIGIN:SPECTRAL ASSOC
	EXTENDED BASI	ROUTINES	AND ENTRY	POINTREVISED:12/26/1999 WALTER K ZYDHEK

		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	9408	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	9506	INCREMENT HORIZONTAL POSITION - Gets the current horizontal coordinate (HORBEG) and moves it one to the right.

EXTENDED BASIC	UNRAVELLED II	APPENDIX D ORIGIN: SPECTRAL AS EXTENDED BASIC ROUTINES AND ENTRY POINTREVISED: 12/26/1999 WALTER K ZYK	
X	95ØD	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.	
А,В,Х	9536	CLEAR GRAPHIC SCREEN - Clear the current graphics screen to the color in ACCB. If ACCB = \emptyset then clear to the current background color.	
А,В	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.	
А,В	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
8000	8001	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	8440	FLOATING POINT VALUE FOR5
8441	8445	FLOATING POINT VALUE FOR THE NATURAL LOG OF 2
84C4	84C8	FLOATING POINT VALUE FOR CORRECTION FACTOR OF EXPONENTIAL FUNCTION
8409	84F1	TAYLOR SERIES FOR E^X
89ØB	89ØE	ERROR MESSAGES
8BD9	8BDC	UL' (UNKNOWN LINE NUMBER) MESSAGE
9290	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
9/06	9/01	TABLE OF HOW MANY BYTES PER HURIZUNTAL GRAPHI

EXTENDED BASIC UNRAVELLED	II	APPENDIX E ORIGIN:SPECTRAL ASSOC EXTENDED BASIC'S DATA/ASCII TABLES REVISED:12/26/1999 WALTER K ZYDHEK
		ROW AND HOW MUCH RAM IS USED FOR ONE HIRES SCREEN
9839 9	851	LOOKUP TABLE FOR PSET, PRESET, AND, OR, AND NOT MODIFIERS FOR THE PUT COMMAND
9C5B 9	C61	NUMERICAL NOTE VALUES FOR LETTER NOTES
9C62 9	C79	TIMING DELAYS FOR OCTAVE 1
9C7A 9	C91	TIMING DELAYS FOR OCTAVE 2
9C92 9	CB5	TIMING DELAYS FOR OCTAVES 3, 4 AND 5
9E79 9	E9C	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 NF	NBR ØØ	LABEL LB1Ø8			DESCRIPTION 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
٥V	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	10	LBCØ6	BCØ6		DIVISION BY ZERO
ΙD	11	INPUT	AFF5	COND	ILLEGAL DIRECT STATEMENT
TM	12	LB151	B151		TYPE MISMATCH
0S	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
Α0	18	LA61C	A61C		FILE ALREADY OPEN
DN	19	LA61F	A61F		DEVICE NUMBER ERROR
ΙΟ	20	LA619	A619		INPUT/OUTPUT ERROR
FM	21	LA616	A616		BAD FILE MODE
NO	22	LA3FB	A3FB		FILE NOT OPEN
ΙE	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
ΝE	26	L8CDD	8CDD		FILE NOT FOUND

The addresses given for the entry points are valid for COLOR BASIC Versions 1.0, 1.1, 1.2 and Extended BASIC Versions 1.0, 1.1, and 2.0. 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 1.0 and 1.1 are not earthshaking. 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 1.0 AND 1.1

ADDRESS

80D0-80DD Patch to move the BASIC input pointer during PCLEAR (see Figure G1). The originial code in the 1.0 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.

		* THI	S CODE	IS NOT U	JSED BY ANY OF THE BASICS
8ØDØ	В6	FF22	LDA	PIA1+2	READ PIA PORT B
8ØD3	85	Ø2	BITA	#2	CHECK MEM SIZE JUMPER
8ØD5	26	Ø3	BNE	L8ØDA	BRANCH IF HIGH
8ØD7	В7	FFDD	STA	SAM+29	SET SAM CNTL REG MEM SIZE TO 64K
8ØDA	6E	84 L8ØDA	JMP	, X	JUMP TO ADDRESS IN X REG
8ØDC	ØØ	ØØ	FCB	\$00,\$00	DEAD SPACE

Figure G1 - Version 1.0 Code

8ØFF Change version number 1.0/1.1

8112 Change Copyright Year (units digit)

8C1B-8C22 Patch to fix the DLOAD bug (see Figure G2). The code in 1.0 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	Ø4	BNE	L8C25	NO
8C21	Α7	E4	STA	, S	SAVE THE M ON THE STACK

Figure G2 - Version 1.0 code

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

9179 Change op code address field from #12(1.0) to #10(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(1.0)	to	#'9'+1
	(1.1)								

- 962C-962D Change instruction from LDA #6(1.0) 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	1025	1DA3	LBL0	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	#200	ADD 200 - LEAVE SOME ROOM FOR STACK
96BØ	93	21	SUBD	FRETOP	SUBTRACT OUT START OF CLEARED SPACE
96B2	24	В9	BCC	L966D	NO ROOM LEFT - 'ILLEGAL FUNCTION CALL'

Figure G3 - Version 1.0 code

DISPLAY CHARACTER SET

HEX VALUE			HEX VALUE			HEX VALUE				
	Non-		CHARACTER	Non-		CHARACTER	Non-		CHARACTER	
	Inverted	Inverted		Inverted	Inverted		Inverted	Inverted		
	ØØ	40	@	18	58	Х	3Ø	40	Ø	
	Ø1	41	Α	19	59	Υ	31	41	1	
	Ø2	42	В	1A	5 A	Z	32	42	2 3	
	Ø3	43	С	1B	5B	[33	43		
	Ø4	44	D	1C	5 C	\	34	44	4	
	Ø5	45	Е	1 D	5 D]	35	45	5	
	Ø6	46	F	1 E	5 E	^	36	46	6	
	Ø7	47	G	1 F	5 F	←	37	47	7	
	Ø8	48	Н	20	60		38	48	8	
	Ø9	49	I	21	61	!	39	49	9	
	ØA	4A	J	22	62	"	3A	4 A	:	
	ØB	4B	K	23	63	#	3B	4B	;	
	ØC	4 C	L	24	64	\$	3C	4 C	<	
	ØD	4 D	М	25	65	%	3 D	4 D	=	
	ØE	4 E	N	26	66	&	3E	4 E	>	
	ØF	4 F	0	27	67	•	3F	4 F	?	
	10	50	Р	28	68	(
	11	51	Q	29	69)				
	12	52	R	2 A	6 A	*				
	13	53	S	2B	6B	+				
	14	54	Т	2C	6C	,				
	15	55	U	2 D	6 D	-				
	16	56	V	2 E	6 E	•				
	17	57	W	2 F	6 F	/				