EXTENDED BASIC UNRAVELLED II

FOR THE TANDY TRS-80 COLOR COMPUTER
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th></th>
<th>FOREWORD</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>INTRODUCTION</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>HOW TO USE THIS BOOK</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>DESCRIPTION OF EXTENDED BASIC</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>GRAPHICS ROUTINES</td>
<td>7</td>
</tr>
</tbody>
</table>

## APPENDICES

<table>
<thead>
<tr>
<th></th>
<th>MEMORY MAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>DISASSEMBLY OF EXTENDED BASIC</td>
</tr>
<tr>
<td>C</td>
<td>EXTENDED BASIC SYMBOL TABLE</td>
</tr>
<tr>
<td>D</td>
<td>EXTENDED BASIC ROUTINES AND ENTRY POINTS</td>
</tr>
<tr>
<td>E</td>
<td>EXTENDED BASIC S DATA/ASCII TABLES</td>
</tr>
<tr>
<td>F</td>
<td>EXTENDED BASIC ERROR ENTRY POINTS</td>
</tr>
<tr>
<td>G</td>
<td>EXTENDED BASIC 1.0 DIFFERENCES</td>
</tr>
<tr>
<td>H</td>
<td>ASCII CHART</td>
</tr>
</tbody>
</table>
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 O 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 1st Tandy Color Computer 2 for Christmas. It had 32K of ram, Cassette, and one Cartridge. I quickly learned to program in Basic and then moved into Assembly.

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

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

I had a couple of close friends that helped me explore the world of CoCo and by this time, I knew that my CoCo would be my friend forever. I give special thanks to Steve Cohn, who helped me get started with ADOS. Two other people whose names I can't remember were very beneficial to my mastering of the CoCo.

Shortly after getting my CoCo 3, I started BBS ing. Wow, a whole new world. My knowledge just kept growing.
A few years later, I moved to Oregon, then to Phoenix, Arizona to attend school. I studied Electronics Technology at Phoenix Institute of Technology. In the second year, we studied Micro-processor Theory. For our labs, we just happen to use the Tandy Color Computer 3 (for studying 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.
Extended BASIC Unravelled is a commented, disassembled listing of the Color Computer Extended BASIC ROM. The author has never seen any kind of source listing for the Color Computer ROMs, so the comments and disassembly are 100% unique. Some of the variable label literals, which were used, have come from published memory maps of systems, which use a BASIC similar to that used in the Color Computer.

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

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

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

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

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

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

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

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

\[ a + bx + cx^2 + dx^3 + ex^4 + \ldots \]

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 Tchebycheff 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 0-255 and a vertical coordinate from 0-191. 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 (CALcuate 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**

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NORMALIZING</td>
<td>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.</td>
</tr>
<tr>
<td>CALPOS</td>
<td>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.</td>
</tr>
<tr>
<td>PIXEL</td>
<td>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.</td>
</tr>
<tr>
<td>PIXEL MASK</td>
<td>A data mask which, if ANDed with a graphic byte from the video screen will leave only the information for one pixel.</td>
</tr>
</tbody>
</table>
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.

IF THIS AREA CONTAINS ANYTHING, IT WILL BE STORED WITH THE PUT
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.

ANYTHING IN THIS AREA WILL BE IGNORED
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 select 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 ENDCRP (which were set by PMODE). If no argument is specified, the current background color is used as a default value otherwise the ASCII 0 TO 3 which was parsed from the BASIC program line gets converted to a binary 0 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 ENDCRP since these memory locations contain the absolute address of the start and
end of the current graphics page. For example, a partial PCLS of the screen may be
accomplished from BASIC by saving the original values of BEGGRP and ENDGRP,
altering them, doing a PCLS and restoring the original values. Extreme caution must
be exercised when doing something like this, POKEing the wrong values could cause
your BASIC program to be erased, and if you forget to restore the original values,
the graphics commands will not work properly.

COLOR

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

PPOINT

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

LINE

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

PSET/PRESET

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

DRAW is a routine that has the ability to draw lines of a specified length in any one of 8 angles, 0, 45, 90, 135, 180, 225, 270, 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.
BLOCK 6.0

**EXPANDED MEMORY DEFINITIONS**

- **EI0**
- **EI1**
- **TCRN**
- **MC3**
- **FEN**
- **MMUEN**
- **COCO**
- **HB.SIZE**
- **HB.ADDR**
- **HBUFF HGET/HPUT BUFFER HEADER EQUATES**
  - **CURCHAR**
  - **EBHISTOK**
  - **EBHISTOK**
  - **HRESSCRN**
  - **RAMLINK**
- **SUPER EXTENDED BASIC EQUATES**
  - **SKP1LD**
  - **SKP2**
  - **OUTFIL**
  - **INPFIL**
  - **FCBLEN**
  - **GRANMX**
  - **TRKLEN**
  - **SECMAX**
  - **LBUFMX**
  - **FORMF**
  - **ESC**
  - **CR**
  - **ROMPAK**
- **HI-RES GRAPHICS SCREEN**
  - **1 = standard SCS**
  - **1 = RAM at XFEXX is constant**
- **1 = MMU enabled**
- **FIRST ENHANCED BASIC FUNCTION TOKEN NUMBER BUG - SHOULD BE $28**
- **FIRST ENHANCED BASIC TOKEN NUMBER**
- **ADDRESS OF THE HI-RES SCREEN IN THE CPU'S MEMORY SPACE**
- **MAXIMUM NUMBER OF ROWS IN HI-RES PRINT MODE**
- **MAXIMUM NUMBER OF SECTORS PER TRACK**
- **MAX NUMBER OF CHARS IN A BASIC LINE**
- **DEBOUNCE DELAY**
- **STACK BUFFER ROOM**
- **AND LOAD THE VALUE OF THAT BYTE INTO ACCA THIS**
- **IS USUALLY USED TO LOAD ACCA WITH A NON ZERO VALUE**
- *** SUPER EXTENDED BASIC EQUATES**
- **ROWMAX**
- **RAMLINK**
- **HRESSCRN**
- **HRESSCRN**
- **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**
- **CURCHAR**
- **SPACE**
- **HI-RES CURSOR CHARACTER**
- *** HBUFF HGET/HPUT BUFFER HEADER EQUATES**
- **HB.ADDR**
- **HB.NUM**
- **HB.SIZE**
- **HB.LEN**
- *** VIDEO REGISTER EQUATES**
- *** INIT0 BIT EQUATES**
- **COCO**
- **MMUEN**
- **IEN**
- **FEN**
- **MC3**
- **MC2**
- **MC1**
- **MC0**
- **TMR**
- **HBORO**
- **VBORD**
- **EI2**
- **EI1**
- **EI0**
- *** INTERRUPT REQUEST ENABLED**
- *** EXPANDED MEMORY DEFINITIONS**
- **BLOCK 6.0**
- **BLOCK 6.1**
- **BLOCK 6.2**
- **BLOCK 6.3**
- **BLOCKS $30-$33 ARE THE HI-RES GRAPHICS SCREEN**
- **HI-RES GRAPHICS SCREEN**
- **HI-RES GRAPHICS SCREEN**
- **HI-RES GRAPHICS SCREEN**
0075  0034  BLOCK 6.4  EQU  $34  GET/PUT BUFFER
0075  0035  BLOCK 6.5  EQU  $35  STACK AREA FOR HI-RES GRAPHICS COMMAND
0077  0036  BLOCK 6.6  EQU  $36  CHARACTER POINTERS
0078  0037  BLOCK 6.7  EQU  $37  UNUSED BY BASIC

0079

0080  * BLOCKS $48-$4F ARE USED FOR THE BASIC OPERATING SYSTEM
0081  0038  BLOCK7.0  EQU  $38
0082  0039  BLOCK7.1  EQU  $39
0083  003A  BLOCK7.2  EQU  $3A
0084  003B  BLOCK7.3  EQU  $3B
0085  003C  BLOCK7.4  EQU  $3C
0086  003D  BLOCK7.5  EQU  $3D
0087  003E  BLOCK7.6  EQU  $3E
0088  003F  BLOCK7.7  EQU  $3F

0089

0090

0091

0092  0000  ORG  0
0093  0000  SETDP  0

0094

0095  0000  ENDFLG  RMB  1  STOP/END FLAG:  POSITIVE=STOP, NEG=END
0096  0001  CHARAC  RMB  1  TERMINATOR FLAG 1
0097  0002  ENDCUR  RMB  1  TERMINATOR FLAG 2
0098  0003  TMPLOC  RMB  1  SCRATCH VARIABLE
0099  0004  IFCTR  RMB  1  IF COUNTER - HOW MANY IF STATEMENTS IN A LINE
0100  0005  DIFMFLG  RMB  1  *DV* ARRAY FLAG 0=EVALUATE, 1=DIMENSIONING
0101  0006  VALTYP  RMB  1  *DV* *PV TYPE FLAG:  0=NUMERIC, $FF=STRING
0102  0007  GARBEL  RMB  1  *TV STRING SPACE HOUSEKEEPING FLAG
0103  0008  ARYDIS  RMB  1  DISABLE ARRAY SEARCH: 0=ALLOW SEARCH
0104  0009  INFPLG  RMB  1  *TV INPUT FLAG: READ=0, INPUT<>0
0105  000A  RELFLG  RMB  1  *TV RELATIONAL OPERATOR FLAG
0106  000B  TEMPPT  RMB  2  *PV TEMPORARY STRING STACK POINTER
0107  000C  LASTPT  RMB  2  *PV ADDR OF LAST USED STRING STACK ADDRESS
0108  000D  TEMPR  RMB  2  TEMPORARY POINTER
0109  0010  TMPTR1  RMB  2  TEMPORARY DESCRIPTOR STORAGE (STACK SEARCH)

0110  ** FLOATING POINT ACCUMULATOR #2 (MANTISSA ONLY)
0111  0013  FPAT2  RMB  4  FLOATING POINT ACCUMULATOR #2 MANTISSA
0112  0017  BOSTK  RMB  2  BOTTOM OF STACK AT LAST CHECK
0113  0019  TXTTAB  RMB  2  *PV BEGINNING OF BASIC PROGRAM
0114  001B  VARTAB  RMB  2  *PV START OF VARIABLES
0115  001D  ARYTAB  RMB  2  *PV START OF ARRAYS
0116  001F  ARYEND  RMB  2  *PV END OF ARRAYS (+1)
0117  0021  FRETOP  RMB  2  *PV START OF STRING STORAGE (TOP OF FREE RAM)
0118  0023  STRTAB  RMB  2  *PV START OF STRING VARIABLES
0119  0025  FRESFC  RMB  2  UTILITY STRING POINTER
0120  0027  MEMSIZ  RMB  2  *PV TOP OF STRING SPACE
0121  0029  OLDTXT  RMB  2  SAVED LINE NUMBER DURING A "STOP"
0122  002B  BINVAL  RMB  2  BINARY VALUE OF A CONVERTED LINE NUMBER
0123  002D  OLDPTR  RMB  2  SAVED INPUT PTR DURING A "STOP"
0124  002F  TINPTR  RMB  2  TEMPORARY INPUT POINTER STORAGE
0125  0031  DATTXT  RMB  2  *PV 'DATA' STATEMENT LINE NUMBER POINTER
0126  0033  DATPTR  RMB  2  *PV 'DATA' STATEMENT ADDRESS POINTER
0127  0035  DATMP  RMB  2  DATA POINTER FOR 'INPUT' & 'READ'
0128  0037  VARNAM  RMB  2  *TV TEMP STORAGE FOR A VARIABLE NAME
0129  0039  VARPTR  RMB  2  *TV POINTER TO A VARIABLE DESCRIPTOR
0130  003B  VADES  RMB  2  TEMP POINTER TO A VARIABLE DESCRIPTOR
0131  003D  RELPTR  RMB  2  POINTER TO RELATIONAL OPERATOR PROCESSING ROUTINE
0132  003F  TRELFL  RMB  1  TEMPORARY RELATIONAL OPERATOR FLAG BYTE

0133

0134  * FLOATING POINT ACCUMULATORS #3,4 & 5 ARE MOSTLY
0135  * USED AS SCRATCH PAD VARIABLES.
0136  ** FLOATING POINT ACCUMULATOR #3 :PACKED: ($40-$44)
0137  0040  V40  RMB  1
0138  0041  V41  RMB  1
0139  0042  V42  RMB  1
0140  0043  V43  RMB  1
0141  0044  V44  RMB  1

0142  ** FLOATING POINT ACCUMULATOR #4 :PACKED: ($45-$49)
0143  0045  V45  RMB  1
0144  0046  V46  RMB  1
0145  0047  V47  RMB  1
0146  0048  V48  RMB  2

0147  ** FLOATING POINT ACCUMULATOR #5 :PACKED: ($4A $4E)
0148  004A  V4A  RMB  1
0149 0048  V4B  RMB 2
0150 0040  VAD  RMB 2
0151  ** FROUNDING POINT ACCUMULATOR #0
0152 004F  FPDEXP  RMB 1  *PV FROUNDING POINT ACCUMULATOR #0 EXPONENT
0153 0050  FPDA0  RMB 4  *PV FROUNDING POINT ACCUMULATOR #0 MANTISSA
0154 0054  FPDSGN  RMB 1  *PV FROUNDING POINT ACCUMULATOR #0 SIGN
0155 0055  COEFC  RMB 1  POLYNOMIAL COEFFICIENT COUNTER
0156 0056  STREDS  RMB 1  TEMPTARY STRING DESCRIPTOR
0157 0058  FPCRAY  RMB 1  FLOATING POINT CARRY BYTE
0158  ** FROUNDING POINT ACCUMULATOR #1
0159 005C  FP1EXP  RMB 1  *PV FROUNDING POINT ACCUMULATOR #1 EXPONENT
0160 005D  FP1A1  RMB 4  *PV FROUNDING POINT ACCUMULATOR #1 MANTISSA
0161 0061  FP1SGN  RMB 1  *PV FROUNDING POINT ACCUMULATOR #1 SIGN
0162 0062  RESSGN  RMB 1  SIGN OF RESULT OF FROUNDING POINT OPERATION
0164 0063  FPSBYT  RMB 1  FLOATING POINT SUB BYTE (FIFTH BYTE)
0165 0064  COEFFT  RMB 2  POLYNOMIAL COEFFICIENT POINTER
0166 0066  LSTXTT  RMB 2  CURRENT LINE POINTER DURING LIST
0167 0068  CURLIN  RMB 2  *PV CURRENT LINE # OF BASIC PROGRAM, $FFFF = DIRECT
0168 006A  DEVCFW  RMB 1  *TV TAB FIELD WIDTH
0169 006B  DEVLCF  RMB 1  *TV TAB ZONE
0170 006C  DEVPOS  RMB 1  *TV PRINT POSITION
0171 006D  DEWDW  RMB 1  *TV PRINT WIDTH
0172 006E  PRDEV  RMB 1  *TV PRINT DEVICE: =NOT CASSETTE, -1=CASSETTE
0173 006F  DEVNUM  RMB 1  *PV DEVICE NUMBER: -3=DLOAD, -2=PRINTER,
0174  *  -1=CASSETTE, 0=SCREEN, 1-15=DISK
0175 0070  CINBF0  RMB 1  *PV CONSOLE IN BUFFER FLAG: =NOT EMPTY, $FF=EMPTY
0176 0071  RSTFLG  RMB 1  *PV WARM START FLAG: 55=WARM, OTHER=COLD
0177 0072  RSTVEC  RMB 2  *PV WARM START VECTOR - JUMP ADDRESS FOR WARM START
0178 0074  TOPRAM  RMB 2  *PV TOP OF RAM
0179 0076  RM  RMB 2  SPARE: UNUSED VARIABLES
0180 0078  FLOST  RMB 1  *PV FILE STATUS FLAG: =CLOSED, 1=INPUT, 2=OUTPUT
0181 0079  CINCTR  RMB 1  *PV CONSOLE IN BUFFER CHAR COUNTER
0182 007A  CINPTR  RMB 2  *PV CONSOLE IN BUFFER POINTER
0183 007C  BLKYP  RMB 1  *TV CASS BLOCK TYPE: =HEADER, =DATA, $FF=EOF
0184 007D  BLKLEN  RMB 1  *TV CASSETTE BYTE COUNT
0185 007E  CBUFFD  RMB 2  *TV CASSETTE LOAD BUFFER POINTER
0186 0080  CKSUM  RMB 1  *TV CASSETTE CHECKSUM BYTE
0187 0081  CSRERR  RMB 1  *TV ERROR FLAG/CHARACTER COUNTER
0188 0082  CPUW  RMB 1  *TV PULSE WIDTH COUNT
0189 0083  CPERTM  RMB 1  *TV BIT counter
0190 0084  CBTHRA  RMB 1  *TV BIT PHASE FLAG
0191 0085  CLSTSN  RMB 1  *TV LAST SINE TABLE ENTRY
0192 0086  GRBLOC  RMB 1  *TV GRAPHIC BLOCK VALUE FOR SET, RESET AND POINT
0193 0087  IKEYIM  RMB 1  *TV INKEYS RAM IMAGE
0194 0088  CURPOS  RMB 2  *PV CURSOR LOCATION
0195 008A  ZERO  RMB 2  *PV DUMMY - THESE TWO BYTES ARE ALWAYS ZERO
0196 008C  SNTON  RMB 1  *TV TONE VALUE FOR SOUND COMMAND
0197 008D  SNODUR  RMB 2  *TV DURATION VALUE FOR SOUND COMMAND
0198 008E  ** THESE BYTES ARE MOVED DOWN FROM ROM
0200 0090  *** INIT DESCRIPTION
0201  ** VALUE
0202 0098  CROPMID  RMB 1  18  *PV 1200/2400 HERTZ PARTITION
0203 0099  CMP0  RMB 1  24  *PV UPPER LIMIT OF 1200 HERTZ PERIOD
0204 009A  CMP1  RMB 1  10  *PV UPPER LIMIT OF 2400 HERTZ PERIOD
0205 009B  SYNCLN  RMB 2  128  *PV NUMBER OF 55'S TO CASSETTE LEADER
0206 009C  BLKONT  RMB 1  11  *PV CURSOR BLINK DELAY
0207 009D  GBPLOC  RMB 2  08  *PV BAUD RATE CONSTANT (600)
0208 009E  LPTIND  RMB 2  1  *PV PRINTER CARRIAGE RETURN DELAY
0209 009F  LPTCFW  RMB 1  16  *PV TAB FIELD WIDTH
0210 00A0  LPTLFCF  RMB 1  112  *PV LAST TAB ZONE
0211 00A1  LPTWID  RMB 1  132  *PV PRINTER WIDTH
0212 00A2  LPTPOS  RMB 1  0  *PV LINE PRINTER POSITION
0213 00A3  EXCPYD  RMB 2  LBAAAA  *PV JUMP ADDRESS FOR EXEC COMMAND
0214  ** THIS ROUTINE PICKS UP THE NEXT INPUT CHARACTER FROM
0215 00A5  ** BASIC. THE ADDRESS OF THE NEXT BASIC BYTE TO BE
0216 00A6  ** INTERPRETED IS STORED AT CHARAD.
0217 00A7 009F 0C  A7  GETNCH  INC <CHARAD+1  *PV INCREMENT LS BYTE OF INPUT POINTER
0218 00A8 00A1  0D  02  BNE GETNCH  *PV BRANCH IF NOT ZERO (NO CARRY)
0219 00A9 00A3  0C  A6  INC <CHARAD  *PV INCREMENT MS BYTE OF INPUT POINTER
0220 00AA 00A5  B6  GETNCH  FCB $B6  *PV OPER CODE OF LDA EXTENDED
0223 @0A6 CHARAD 2 *PV THESE 2 BYTES CONTAIN ADDRESS OF THE CURRENT
0224 * CHARACTER WHICH THE BASIC INTERPRETER IS
0225 * PROCESSING
0226 @0AB 7E AA 1A JMP BROMHK JUMP BACK INTO THE BASIC RUN
0227
0228 @0A8 VAB RMB 1 = LOW ORDER FOUR BYTES OF THE PRODUCT
0229 @0AC VAC RMB 1 = OF A FLOATING POINT MULTIPLICATION
0230 @0AD VAD RMB 1 = THESE BYTES ARE USED AS RANDOM DATA
0231 @0AE VAE RMB 1 = BY THE RND STATEMENT
0232
0233 * EXTENDED BASIC VARIABLES
0234 @0AF TRCFLG RMB 1 *PV TRACE FLAG 0=OFF ELSE=ON
0235 @0B0 USRADR RMB 2 *PV ADDRESS OF THE START OF USR VECTORS
0236 @0B2 FORCL RMB 1 *PV FOREGROUND COLOR
0237 @0B3 BAKCOL RMB 1 *PV BACKGROUND COLOR
0238 @0B4 WCOLOR RMB 1 *TV WORKING COLOR BEING USED BY EX BASIC
0239 @0B5 ALCOL RMB 1 *TV ALL PIXELS IN THIS BYTE SET TO COLOR OF VB3
0240 @0B6 PMODE RMB 1 *PV PMODE'S MODE ARGUMENT
0241 @0B7 ENDGRP RMB 2 *PV END OF CURRENT GRAPHIC PAGE
0242 @0B9 HORBYT RMB 1 *PV NUMBER OF BYTES/HORIZONTAL GRAPHIC LINE
0243 @0BA BEGGRP RMB 2 *PV START OF CURRENT GRAPHIC PAGE
0244 @0BC GRPRAM RMB 1 *PV START OF GRAPHIC RAM (HS BYTE)
0245 @0BD HORBEG RMB 2 *DV* *PV HORIZ COORD - START POINT
0246 @0BF VERBEG RMB 2 *DV* *PV VERT COORD - START POINT
0247 @0C1 CSSYAL RMB 1 *PV SCREEN'S COLOR SET ARGUMENT
0248 @0C2 SETFLG RMB 1 *PV PRESET/PSET FLAG: 0=PRESET, 1=PSET
0249 @0C3 HOREND RMB 2 *DV* *PV HORIZ COORD - ENDING POINT
0250 @0C5 HEND RMB 2 *DV* *PV VERT COORD - ENDING POINT
0251 @0C7 HORDFM RMB 2 *PV HORIZ COORD - DEFAULT COORD
0252 @0C9 VERDEF RMB 2 *PV VERT COORD - DEFAULT COORD
0253
0254 * EXTENDED BASIC SCRATCH PAD VARIABLES
0255 @0C8 VCB RMB 2
0256 @0CD VCD RMB 2
0257 @0CF VCF RMB 2
0258 @0D1 VD1 RMB 2
0259 @0D3 VD3 RMB 1
0260 @0D4 VD4 RMB 1
0261 @0D5 VD5 RMB 1
0262 @0D6 VD6 RMB 1
0263 @0D7 VD7 RMB 1
0264 @0D8 VDB RMB 1
0265 @0D9 VDB RMB 1
0266 @0DA VDA RMB 1
0267
0268 @0D9 CHGFLG RMB 1 *TV FLAG TO INDICATE IF GRAPHIC DATA HAS BEEN CHANGED
0269 @0DC TMPSTK RMB 2 *TV STACK POINTER STORAGE DURING PAINT
0270 @0DE OCTAVE RMB 1 *PV OCTAVE VALUE (PLAY)
0271 @0DF VOLOH RMB 1 *DV* *PV VOLUME HIGH VALUE (PLAY)
0272 @0E0 VOLLW RMB 1 *DV* *PV VOLUME LOW VALUE (PLAY)
0273 @0E1 NOTELN RMB 1 *PV NOTE LENGTH (PLAY)
0274 @0E2 TEMPO RMB 1 *PV TEMPO VALUE (PLAY)
0275 @0E3 PLYTMR RMB 2 *TV TIMER FOR THE PLAY COMMAND
0276 @0E5 DOTYAL RMB 1 *TV DOTTED NOTE TIMER SCALE FACTOR
0277 @0E6 HMODE EQU * SUPER EXTENDED BASIC HI-RES MODE
0278 @0E6 DLBAUD RMB 1 *DV* *PV DLOAD BAUD RATE CONSTANT $B0=300, $2C=1200
0279 @0E7 HMRUN EQU * SUPER EXTENDED BASIC HI-RES TEXT MODE
0280 @0E7 TIMOUT RMB 1 *DV* *PV DLOAD TIMEOUT CONSTANT
0281 @0E9 ANGLE RMB 1 *DV* *PV ANGLE VALUE (DRAW)
0282 @0E9 SCALE RMB 1 *DV* *PV SCALE VALUE (DRAW)
0283
0284 * DSKCON VARIABLES
0285 @0EA DCPC RMB 1 *PV DSKCON OPERATION CODE 0-3
0286 @0EB DCDR RMB 1 *PV DSKCON DRIVE NUMBER 0 3
0287 @0EC DCTR RMB 1 *PV DSKCON TRACK NUMBER 0-34
0288 @0ED DSEC RMB 1 *PV DSKCON SECTOR NUMBER 1-18
0289 @0EE DCBPT RMB 2 *PV DSKCON DATA POINTER
0290 @0F8 DCSA RMB 1 *PV DSKCON STATUS BYTE
0291
0292 @0F1 FCBTMP RMB 2 TEMPORARY FCB POINTER
0293
0294 @0F3 RMB 13 SPARE: UNUSED VARIABLES
0295
0296
0297  * BASIC EXBASI\DOSBASIC
0298
0299  @100  SW3VEC RMB 3 $XXX $XXX $3B3B SWI3 VECTOR
0300  @103  SW2VEC RMB 3 $XXX $XXX $3B3B SWI2 VECTOR
0301  @106  SW1VEC RMB 3 $XXX $XXX $3B3B SWI1 VECTOR
0302  @109  NMIVEC RMB 3 $AAA5 $84C $D7BC IRQ VECTOR
0303  @10F  PROVEC RMB 3 $A8F6 $A8F6 $A8F6 FIQ VECTOR
0304
0305  @112  TIMVAL
0306
0307  @112  USRJUMP RMB 3 JUMP ADDRESS FOR BASIC’S USR FUNCTION
0308  *  RMB 2 TIMER VALUE FOR EXBAS
0309  *  RMB 1 UNUSED BY EXBAS OR DISK BASIC
0310  @115  RVSEED RMB 1 * FLOATING POINT RANDOM NUMBER SEED EXPONENT
0311  @116  RMB 4 * MANITISSA: INITIALLY SET TO $848FC75259
0312  @11A  CASFLG RMB 1 UPPER CASE/LOWER CASE FLAG: $FF=UPPER, 0=LOWER
0313  @11B  DEBVAL RMB 2 KEYBOARD DEBOUNCE DELAY (SET TO $45E)
0314  @11D  EXPJMP RMB 3 JUMP ADDRESS FOR EXPONENTIATION
0315  ** INITIALLY SET TO ERROR FOR BASIC, $8489 FOR EX BASIC
0316
0317  *** COMMAND INTERPRETATION VECTOR TABLE
0318
0319  ** FOUR SETS OF 10 BYTE TABLES:
0320
0321
0322  ** THE LAST USED TABLE MUST BE FOLLOWED BY A ZERO BYTE
0323  * THE JUMP TABLE VECTORS (3,4 AND 8,9) POINT TO THE JUMP TABLE FOR
0324  * THE FIRST TABLE. FOR ALL OTHER TABLES, THESE VECTORS POINT TO A
0325  * ROUTINE WHICH WILL VECTOR YOU TO THE CORRECT JUMP TABLE.
0326  * SUPER ENHANCED BASIC HAS MODIFIED THIS SCHEME SO THAT THE USER
0327  * TABLE MAY NOT BE ACCESSED. ANY ADDITIONAL TABLES WILL HAVE TO BE
0328  * ACCESSED FROM A NEW COMMAND HANDLER.
0329
0330  * BYTE DESCRIPTION
0331  * 0 NUMBER OF RESERVED WORDS
0332  * 1,2 LOOKUP TABLE OF RESERVED WORDS
0333  * 3,4 JUMP TABLE FOR COMMANDS (FIRST TABLE)
0334  * 5 VECTOR TO EXPANSION COMMAND HANDLERS (ALL BUT FIRST TABLE)
0335  * 6,7 LOOKUP TABLE OF SECONDARY FUNCTIONS (FIRST TABLE)
0336  * 8,9 JUMP TABLE FOR SECONDARY FUNCTIONS
0337  * 10 0 BYTE - END OF TABLE FLAG (LAST TABLE ONLY)
0338
0339  @120  COMVEC RMB 10 BASIC’S TABLE
0340  @12A  RMB 10 EX BASIC’S TABLE
0341  @134  RMB 10 DISK BASIC’S TABLE (UNUSED BY EX BASIC)
0342
0343  **** USR FUNCTION VECTOR ADDRESSES (EX BASIC ONLY)
0344  @13E  RMB 2 USR 0 VECTOR
0345  @140  RMB 2 USR 1
0346  @142  RMB 2 USR 2
0347  @144  RMB 2 USR 3
0348  @146  RMB 2 USR 4
0349  @148  RMB 2 USR 5
0350  @14A  RMB 2 USR 6
0351  @14C  RMB 2 USR 7
0352  @14E  RMB 2 USR 8
0353  @150  RMB 2 USR 9
0354
0355  **** THE ABOVE 20 BYTE USR ADDR VECTOR TABLE IS MOVED TO
0356  *** $95F-$972 BY DISK BASIC. THE 20 BYTES FROM $13E-$151
0357  *** ARE REDEFINED AS FOLLOWS:
0358
0359  * RMB 10 USER (SPARE) COMMAND INTERPRETATION TABLE SPACE
0360  * FCB 0 END OF COMM INTERP TABLE FLAG
0361  * RMB 9 UNUSED BY DISK BASIC
0362
0363  * COMMAND INTERPRETATION TABLE VALUES
0364  * BYTE BASIC EX BASIDISK BASIC
0365  * 0 53 BASIC TABLE
0366  * 1,2 $AA66
0367  * 3,4 $AB67
0371  *  5  20
0372  *  6,7  $ABA
0373  *  8,9  $AA
0374
0375  *  0  25  EX BASIC TABLE
0376  *  1,2  $B8
0377  *  3,4  $B1C $CE2E ($CFBA 2.1)
0378  *  5  14
0379  *  6,7  $82E
0380  *  8,9  $B168 $CE56 ($CF32 2.1)
0381
0382  *  0  19 (20 2.1) DISK BASIC TABLE
0383  *  1,2  $C17F
0384  *  3,4  $C2C0
0385  *  5  6
0386  *  6,7  $C2B1
0387  *  8,9  $C256
0388
0389
0390  Ø152  KEYBUF  RMB  8  KEYBOARD MEMORY BUFFER
0391  Ø15A  POTVAL  RMB  1  LEFT VERTICAL JOYSTICK DATA
0392  Ø15B  RMB  1  LEFT HORIZONTAL JOYSTICK DATA
0393  Ø15C  RMB  1  RIGHT VERTICAL JOYSTICK DATA
0394  Ø15D  RMB  1  RIGHT HORIZONTAL JOYSTICK DATA
0395
0396  * BASIC'S RAM VECTORS - INITIALIZED TO RTS BY COLOR BASIC
0397  * 25 SETS OF 3 BYTE INSTRUCTIONS WHICH ARE CALLED BY COLOR BASIC
0398  * EXTENDED DISK BASIC. THEIR PURPOSE IS TO ALLOW ENHANCEMENTS (SUCH
0399  * AS EX BASIC AND DOS BASIC) AS MORE ROMS ARE ADDED TO THE
0400  * SYSTEM BY EFFECTIVELY ALLOWING MORE CODE TO BE ADDED TO THE
0401  * ROUTINES IN EARLIER ROMS. THIS NEW CODE IS LOCATED IN THE NEW ROMS
0402  * AND THE ADDRESS TO GET TO THE NEW CODE IS IN BYTES 1 & 2 OF THE
0403  * RAM VECTOR. BYTE 0 WILL CONTAIN A $7E WHICH IS THE FIRST BYTE OF
0404  * THE JMP INSTRUCTION.
0405  * THE FIRST ADDRESS IN THIS TABLE IS THE ADDRESS IN BASIC WHICH
0406  * CALLS THE RAM VECTOR, THE SECOND ADDRESS IS THE VALUE WHICH
0407  * EX BASIC PUTS IN THE RAM VECTOR (IF ANY) AND THE THIRD ADDRESS
0408  * IS THE VALUE WHICH DISK BASIC PUTS THERE (IF ANY)
0409
0410
0411  *  2.0  2.1  1.0  1.1
0412  Ø15E  RVEC0  RMB  3  $A5F6  $C426  $C4A  OPEN COMMAND
0413  Ø161  RVEC1  RMB  3  $A5B9  $C838  $C88  DEVICE NUMBER VALIDITY CHECK
0414  Ø164  RVEC2  RMB  3  $A35F  $C843  $C93  SET PRINT PARAMETERS
0415  Ø167  RVEC3  RMB  3  $A282  $B273  $B4A  CONSOLE OUT
0416  Ø16A  RVEC4  RMB  3  $A176  $8C1F  $C58F  CONSOLE IN
0417  Ø16D  RVEC5  RMB  3  $A3E0  $C818  $C84B INPUT DEVICE NUMBER CHECK
0418  Ø170  RVEC6  RMB  3  $A406  $C818  $C84B PRINT DEVICE NUMBER CHECK
0419  Ø173  RVEC7  RMB  3  $A426  $C83B  $CAE9 CLOSE ALL FILES
0420  Ø176  RVEC8  RMB  3  $A420  $82B6  $CA4B  $CAF9 CLOSE ONE FILE
0421  Ø179  RVEC9  RMB  3  $8B1B  $8E90  $8E90  $8E90 PRINT
0422  Ø17C  RVEC10 RMB  3  $8B61  $C5B  $C35 INPUT
0423  Ø17F  RVEC11 RMB  3  $A549  $C859  $C89A BREAK CHECK
0424  Ø182  RVEC12 RMB  3  $A39B  $C887  $C6E4 INPUTTING A BASIC LINE
0425  Ø185  RVEC13 RMB  3  $A48F  $CAE9 TERMINATING BASIC LINE INPUT
0426  Ø188  RVEC14 RMB  3  $A5CE  $CA60  $C9C3 EOF COMMAND
0427  Ø18B  RVEC15 RMB  3  $B223  $8846  $C9F6  $CED2 EVALUATE AN EXPRESSION
0428  Ø18E  RVEC16 RMB  3  $A4C6  $C687  $C6E4 RESERVED FOR ERROR GOTO COMMAND
0429  Ø191  RVEC17 RMB  3  $A4C9  $88F0  $C240  $C265 ERROR DRIVER
0430  Ø194  RVEC18 RMB  3  $A7E5  $829C  $CAE  $CA36 RUN
0431  Ø197  RVEC19 RMB  3  $8022  $87EF ASCII TO FLOATING POINT CONVERSION
0432  Ø19A  RVEC20 RMB  3  $A09E  $82B9 $C88 BASIC'S COMMAND INTERPRETATION LOOP
0433  Ø19D  RVEC21 RMB  3  $A8C4 RESET/SET/POINT COMMANDS
0434  Ø1A0  RVEC22 RMB  3  $A918  CLS
0435  Ø1A5  $8160 EXBAS' SECONDARY TOKEN HANDLER
0436  *  $8AFA EXBAS' RENUM TOKEN CHECK
0437  *  $975C $C29A $C282 EXBAS' GET/PUT
0438  Ø1A3  RVEC23 RMB  3  $8B21  $83B4 CRUNCH BASIC LINE
0439  Ø1A6  RVEC24 RMB  3  $87C2 UNCRUNCH BASIC LINE
0440
0441  Ø1A9  STRSTK RMB  8*5 STRING DESCRIPTOR STACK
0442  Ø1D1  CFNBUF RMB  9 CASSETTE FILE NAME BUFFER
0443  Ø1DA  CASBUF RMB  256 CASSETTE FILE DATA BUFFER
0444  Ø2DA  LINHDR RMB  2 LINE INPUT BUFFER HEADER

A6
EXTENDED BASIC UNRAVELLED II
APPENDIX A
MEMORY MAP
REVISED: 12/26/1999 WALTER K ZYDEK

0445 020C LINBUF RMB LBUF+1 BASIC LINE INPUT BUFFER
0446 020D STRBUF RMB 41 STRING BUFFER
0447
0448 0400 VIDRAM RMB 200 VIDEO DISPLAY AREA
0449
0450 *START OF ADDITIONAL RAM VARIABLE STORAGE (DISK BASIC ONLY)
0451 0600 DBUF0 RMB SECLEN I/O BUFFER #0
0452 0700 DBUF1 RMB SECLEN I/O BUFFER #1
0453 0800 FATBLO RMB FATLEN FILE ALLOCATION TABLE - DRIVE 0
0454 0944 FATBL1 RMB FATLEN FILE ALLOCATION TABLE - DRIVE 1
0455 0948 FATBL2 RMB FATLEN FILE ALLOCATION TABLE - DRIVE 2
0456 094C FATBL3 RMB FATLEN FILE ALLOCATION TABLE - DRIVE 3
0457 0950 FCV0 RMB 16*2 FILE BUFFER VECTORS (15 USER, 1 SYSTEM)
0458 0970 RNBFD RMB 2 START OF FREE RANDOM FILE BUFFER AREA
0459 0974 FCBADR RMB 2 START OF FILE CONTROL BLOCKS
0460 0976 DNbfad RMB 2 START OF FREE RANDOM FILE BUFFER AREA
0461 0978 FCBADR RMB 2 START OF FILE CONTROL BLOCKS
0462 097A RNbfad RMB 2 START OF FREE RANDOM FILE BUFFER AREA
0463
0464 0958 DASCFL RMB 1 *DV* ASCII FLAG: 0=CRUNCHED OR BINARY, $FF=ASCII
0465 0959 DRunfl RMB 1 RUN FLAG: (IF BIT 1=1 THEN RUN, IF BIT 0=1, THEN CLOSE
0466 * ALL FILES BEFORE RUNNING)
0467 095A DEFDRV RMB 1 DEFAULT DRIVE NUMBER
0468 095B FCVBACT RMB 1 NUMBER OF FCBS ACTIVE
0469 095C DRESFL RMB 1 RESET FLAG: =0 WILL CAUSE A 'NEW' & SHUT DOWN ALL FCBS
0470 095D DLDADFL RMB 1 LOAD FLAG: CAUSE A 'NEW' FOLLOWING A LOAD ERROR
0471 095E DMRGFL RMB 1 MERGE FLAG: =0=N0 MERGE, $FF=MERGE
0472 095F DUSRVC RMB 20 DISK BASIC USR COMMAND VECTORS
0473 *** DISK FILE WORK AREA FOR DIRECTORY SEARCH
0474 * EXISTING FILE
0475 0973 V973 RMB 1 SECTOR NUMBER
0476 0977 V977 RMB 1 FIRST GRANULE NUMBER
0477 0978 V978 RMB 2 RAM DIRECTORY IMAGE ADDRESS
0478
0479 097A W.Fatvl RMB 2 WRITE FAT VALUE: NUMBER OF FREE GRANULES WHICH MUST BE TAKEN
0479 FROM THE FAT TO TRIGGER A WRITE FAT TO DISK SEQUENCE
0480 097C Dfflen RMB 2 DIRECT ACCESS FILE RECORD LENGTH
0481 097D DRTRK RMB 4 CURRENT TRACK NUMBER, DRIVES 0,1,2,3
0482 0982 NMIFLG RMB 1 NMI FLAG: =0=DON'T VECTOR <>VECTOR OUT
0483 0983 Dnmivc RMB 2 NMI VECTOR: WHERE TO JUMP FOLLOWING AN NMI
0484 * INTERRUPT IF THE NMI FLAG IS SET
0485 0985 RDymr RMB 1 MOTOR TURN OFF TIMER
0486 0986 Dgram RMB 1 RAM IMAGE OF DSKREG ($FF40)
0487 0987 Dverfl RMB 1 VERIFY FLAG: =0=OFF, $FF=ON
0487 * DISK WILL ATTEMPT TO RETRIEVE OR WRITE DATA
0488 * BEFORE IT GIVES UP AND ISSUES AN ERROR.
0489
0490 0989 DFLOBF RMB SECLEN INITIALIZED TO SECLEN BY DISKBAS
0491
0492 *RANDOM FILE RESERVED AREA
0493
0500 *FILE CONTROL BLOCKS AND BUFFERS
0501
0502 *GRAPHIC PAGE RESERVED AREA
0503
0504 *BASIC PROGRAM
0505
0506 *VARIABLE STORAGE AREA
0507
0508 *ARRAY STORAGE AREA
0509
0510 * FREE MEMORY
0511
0512
0513
0514 *STACK
0515
0516 *STRING SPACE
0517
0518 *USER PROGRAM RESERVED AREA
0519  *END OF RAM
0520
0521 0800  ORG $8000
0523
0524 0800  RMB $2000 EXTENDED BASIC ROM
0525 A800  RMB $2200 COLOR BASIC ROM
0526 C800  ROMPAK EQU *
0527 C800  DOSBAS RMB $2000 DISK BASIC ROM/ENHANCED BASIC INIT CODE
0528 E000  RMB $1F00 ENHANCED BASIC
0529
0530  * START OF ADDITIONAL VARIABLES USED BY SUPER EXTENDED BASIC
0531 FE00  H.CRSLOC RMB 2 CURRENT LOCATION OF CURSOR
0532 FE02  H.CURSX RMB 1 X POSITION OF CURSOR
0533 FE03  H.CURSY RMB 1 Y POSITION OF CURSOR
0534 FE04  H.COLUMN RMB 1 COLUMNS ON HI-RES ALPHA SCREEN
0535 FE05  H.ROW RMB 1 ROWS ON HI-RES ALPHA SCREEN
0536 FE06  H.DISPEN RMB 2 END OF HI-RES DISPLAY SCREEN
0537 FE08  H.CRSATT RMB 1 CURRENT CURSOR'S ATTRIBUTES
0538 FE09  RMB 1 UNUSED
0539 FE0A  H.FCOLOR RMB 1 FOREGROUND COLOR
0540 FE0B  H.BCOLOR RMB 1 BACKGROUND COLOR
0541 FE0C  H.COLUMN RMB 2 ON BRK GOTO LINE NUMBER
0542 FE0D  H.ONBRK RMB 2 ON ERR GOTO LINE NUMBER
0543 FE10  H.ERROR RMB 1 ERROR NUMBER ENCOUNTERED OR $FF (NO ERROR)
0544 FE11  H.ONERRS RMB 2 ON ERR SOURCE LINE NUMBER
0545 FE12  H.ERLINE RMB 2 LINE NUMBER WHERE ERROR OCCURRED
0546 FE13  H.ONBRKS RMB 2 ON BRK SOURCE LINE NUMBER
0547 FE14  H.ERRBRK RMB 1 STILL UNKNOWN, HAS TO DO WITH ERR, BRK
0548 FE15  H.PCOUNT RMB 1 PRINT COUNT, CHARACTERS TO BE HPRINTED
0549 FE16  H.PBUF RMB 80 PRINT BUFFER, HPRINT CHARs. STORED HERE
0550 FE69  RMB 132 UNUSED
0551 FE6A  INT.FLAG RMB 1 INTERRUPT VALID FLAG. 0=NOT VALID, $55=VALID
0552  * TABLE OF JUMP VECTORS TO INTERRUPT SERVICING ROUTINES
0553 FE6B  INT.JUMP
0554 FE6C  INT.SWI3 RMB 3
0555 FE6D  INT.SWI2 RMB 3
0556 FE6E  INT.FIRQ RMB 3
0557 FE6F  INT.IRQ RMB 3
0558 FE70  INT.NMI RMB 3
0559 FF00  PIA0 EQU *
0560  * I/O AREA
0561
0562 0563 FF00  PIA0 EQU * PERIPHERAL INTERFACE ADAPTER ONE
0564
0565 FF00  BIT0 KEYBOARD ROW 1 AND RIGHT JOYSTICK SWITCH 1
0566 FF01  BIT1 KEYBOARD ROW 2 AND LEFT JOYSTICK SWITCH 1
0567 FF02  BIT2 KEYBOARD ROW 3 AND RIGHT JOYSTICK SWITCH 2
0568 FF03  BIT3 KEYBOARD ROW 4 AND LEFT JOYSTICK SWITCH 2
0569 FF04  BIT4 KEYBOARD ROW 5
0570 FF05  BIT5 KEYBOARD ROW 6
0571 FF06  BIT6 KEYBOARD ROW 7
0572 FF07  BIT7 JOYSTICK COMPARISON INPUT
0573
0574 FF00  BIT0 CONTROL OF HSYNC (63.5ps) 0 = IRQ* TO CPU DISABLED
0575 FF01  INTERRUPT 1 = IRQ* TO CPU ENABLED
0576 FF02  BIT1 CONTROL OF INTERRUPT 0 = FLAG SET ON FALLING EDGE OF HS
0577 FF03  POLARITY 1 = FLAG SET ON RISING EDGE OF HS
0578 FF04  BIT2 NORMALLY 1 0 = CHANGES FF00 TO DATA DIRECTION
0579 FF05  BIT3 SEL 1 LSB OF TWO ANALOG MUX SELECT LINES
0580 FF06  BIT4 ALWAYS 1
0581 FF07  BIT5 ALWAYS 1
0582 FF08  BIT6 NOT USED
0583 FF09  BIT7 HORIZONTAL SYNC INTERRUPT FLAG
0584
0585 FF00  BIT0 KEYBOARD COLUMN 1
0586 FF01  BIT1 KEYBOARD COLUMN 2
0587 FF02  BIT2 KEYBOARD COLUMN 3
0588 FF03  BIT3 KEYBOARD COLUMN 4
0589 FF04  BIT4 KEYBOARD COLUMN 5
0590 FF05  BIT5 KEYBOARD COLUMN 6
0591 FF06  BIT6 KEYBOARD COLUMN 7 / RAM SIZE OUTPUT
0592 FF07  BIT7 KEYBOARD COLUMN 8
<table>
<thead>
<tr>
<th>Address</th>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0594 FF03</td>
<td>B0</td>
<td>Control of VSync (16.667ms) 0 = IRQ* to CPU disabled</td>
<td>1 = IRQ* to CPU enabled</td>
</tr>
<tr>
<td>0596 FF03</td>
<td>B1</td>
<td>Control of interrupt 0 = flag set on falling edge of FS</td>
<td>1 = flag set on rising edge of FS</td>
</tr>
<tr>
<td>0597 FF03</td>
<td>B2</td>
<td>Polarity 0 = changes FF02 to data direction</td>
<td>1 = flag set on rising edge of FS</td>
</tr>
<tr>
<td>0598 FF03</td>
<td>B3</td>
<td>Not used</td>
<td>1 = changes FF02 to data direction</td>
</tr>
<tr>
<td>0600 FF03</td>
<td>B4</td>
<td>Always 1</td>
<td>1 = changes FF02 to data direction</td>
</tr>
<tr>
<td>0601 FF03</td>
<td>B5</td>
<td>Always 1</td>
<td>1 = changes FF02 to data direction</td>
</tr>
<tr>
<td>0602 FF03</td>
<td>B6</td>
<td>Not used</td>
<td>1 = changes FF02 to data direction</td>
</tr>
<tr>
<td>0603 FF03</td>
<td>B7</td>
<td>Field sync interrupt flag</td>
<td>1 = changes FF02 to data direction</td>
</tr>
<tr>
<td>0606 FF04</td>
<td>RMB 2B</td>
<td>PIA0 images</td>
<td></td>
</tr>
<tr>
<td>0607 FF20</td>
<td>DA</td>
<td>PIA1 EQU *</td>
<td>Peripheral interface adapter two</td>
</tr>
<tr>
<td>0609 FF20</td>
<td>B0</td>
<td>Cassette data input</td>
<td></td>
</tr>
<tr>
<td>0610 FF20</td>
<td>B1</td>
<td>6 bit D/A LSB</td>
<td></td>
</tr>
<tr>
<td>0611 FF20</td>
<td>B2</td>
<td>6 bit D/A</td>
<td></td>
</tr>
<tr>
<td>0612 FF20</td>
<td>B3</td>
<td>6 bit D/A</td>
<td></td>
</tr>
<tr>
<td>0613 FF20</td>
<td>B4</td>
<td>6 bit D/A</td>
<td></td>
</tr>
<tr>
<td>0614 FF20</td>
<td>B5</td>
<td>6 bit D/A</td>
<td></td>
</tr>
<tr>
<td>0615 FF20</td>
<td>B6</td>
<td>6 bit D/A</td>
<td></td>
</tr>
<tr>
<td>0616 FF20</td>
<td>B7</td>
<td>6 bit D/A MSB</td>
<td></td>
</tr>
<tr>
<td>0617 FF21</td>
<td>B0</td>
<td>Control of CD 0 = IRQ* to CPU disabled</td>
<td>1 = IRQ* to CPU enabled</td>
</tr>
<tr>
<td>0619 FF21</td>
<td>B1</td>
<td>(RS-232C status) 1 = IRQ* to CPU enabled</td>
<td></td>
</tr>
<tr>
<td>0620 FF21</td>
<td>B2</td>
<td>Control of interrupt 0 = flag set on falling edge of CD</td>
<td>1 = flag set on rising edge of CD</td>
</tr>
<tr>
<td>0621 FF21</td>
<td>B3</td>
<td>Polarity 0 = changes FF20 to data direction</td>
<td>1 = changes FF20 to data direction</td>
</tr>
<tr>
<td>0622 FF21</td>
<td>B4</td>
<td>Cassette motor control 0 = off 1 = on</td>
<td></td>
</tr>
<tr>
<td>0623 FF21</td>
<td>B5</td>
<td>Always 1</td>
<td></td>
</tr>
<tr>
<td>0624 FF21</td>
<td>B6</td>
<td>Always 1</td>
<td></td>
</tr>
<tr>
<td>0625 FF21</td>
<td>B7</td>
<td>Not used</td>
<td></td>
</tr>
<tr>
<td>0627 FF21</td>
<td>B0</td>
<td>CD interrupt flag</td>
<td></td>
</tr>
<tr>
<td>0628 FF22</td>
<td>B0</td>
<td>RS-232C data input</td>
<td></td>
</tr>
<tr>
<td>0630 FF22</td>
<td>B1</td>
<td>Single bit sound output</td>
<td></td>
</tr>
<tr>
<td>0631 FF22</td>
<td>B2</td>
<td>Ram size input</td>
<td></td>
</tr>
<tr>
<td>0632 FF22</td>
<td>B3</td>
<td>RGB monitor sensing input CSS</td>
<td></td>
</tr>
<tr>
<td>0633 FF22</td>
<td>B4</td>
<td>VDG control output GM0 &amp; upper/lower case</td>
<td></td>
</tr>
<tr>
<td>0634 FF22</td>
<td>B5</td>
<td>VDG control output GM1 &amp; invert</td>
<td></td>
</tr>
<tr>
<td>0635 FF22</td>
<td>B6</td>
<td>VDG control output GM2</td>
<td></td>
</tr>
<tr>
<td>0636 FF22</td>
<td>B7</td>
<td>VDG control output A*/G</td>
<td></td>
</tr>
<tr>
<td>0638 FF23</td>
<td>B0</td>
<td>Control of cartridge 0 = IRQ* to CPU disabled</td>
<td>1 = IRQ* to CPU enabled</td>
</tr>
<tr>
<td>0639 FF23</td>
<td>B1</td>
<td>Control of interrupt 0 = flag set on falling edge of cart*</td>
<td>1 = flag set on rising edge of cart*</td>
</tr>
<tr>
<td>0640 FF23</td>
<td>B2</td>
<td>Polarity 0 = changes FF22 to data direction</td>
<td>1 = changes FF22 to data direction</td>
</tr>
<tr>
<td>0641 FF23</td>
<td>B3</td>
<td>Sound enable 0 = single 1 = double</td>
<td></td>
</tr>
<tr>
<td>0642 FF23</td>
<td>B4</td>
<td>Always 1</td>
<td></td>
</tr>
<tr>
<td>0643 FF23</td>
<td>B5</td>
<td>Always 1</td>
<td></td>
</tr>
<tr>
<td>0644 FF23</td>
<td>B6</td>
<td>Not used</td>
<td></td>
</tr>
<tr>
<td>0645 FF23</td>
<td>B7</td>
<td>Cartridge interrupt flag</td>
<td></td>
</tr>
<tr>
<td>0648 FF24</td>
<td>RMB 2B</td>
<td>PIA1 images</td>
<td></td>
</tr>
<tr>
<td>0650 FF40</td>
<td>PIA2</td>
<td>Disk control register</td>
<td></td>
</tr>
<tr>
<td>0651 FF40</td>
<td>DSKREG</td>
<td>RMB 1</td>
<td></td>
</tr>
<tr>
<td>0652 FF40</td>
<td>B0</td>
<td>Drive select 0</td>
<td></td>
</tr>
<tr>
<td>0653 FF40</td>
<td>B1</td>
<td>Drive select 1</td>
<td></td>
</tr>
<tr>
<td>0654 FF40</td>
<td>B2</td>
<td>Drive select 2</td>
<td></td>
</tr>
<tr>
<td>0655 FF40</td>
<td>B3</td>
<td>Drive motor enable 0 = motors off 1 = motors on</td>
<td></td>
</tr>
<tr>
<td>0656 FF40</td>
<td>B4</td>
<td>Write precompensation 0 = no precomp 1 = precomp</td>
<td></td>
</tr>
<tr>
<td>0657 FF40</td>
<td>B5</td>
<td>Density flag 0 = single 1 = double</td>
<td></td>
</tr>
<tr>
<td>0658 FF40</td>
<td>B6</td>
<td>Drive select 3</td>
<td></td>
</tr>
<tr>
<td>0659 FF40</td>
<td>B7</td>
<td>Halt flag 0 = disabled 1 = enabled</td>
<td></td>
</tr>
<tr>
<td>0660 FF40</td>
<td>RMB 7</td>
<td>DSKREG images</td>
<td></td>
</tr>
<tr>
<td>0661 FF40</td>
<td>B0</td>
<td>Floppy disk controller internal registers</td>
<td></td>
</tr>
<tr>
<td>0665 FF40</td>
<td>FDCREG</td>
<td>RMB 1</td>
<td>Status/command register</td>
</tr>
<tr>
<td>Command</td>
<td>Type</td>
<td>Command</td>
<td>Code</td>
</tr>
<tr>
<td>---------</td>
<td>------</td>
<td>---------</td>
<td>------</td>
</tr>
<tr>
<td>I RESTORE</td>
<td>$03</td>
<td>I SEEK</td>
<td>$17</td>
</tr>
<tr>
<td>I STEP</td>
<td>$23</td>
<td>I STEP IN</td>
<td>$43</td>
</tr>
<tr>
<td>I STEP OUT</td>
<td>$53</td>
<td>II READ SECTOR</td>
<td>$80</td>
</tr>
<tr>
<td>II WRITE SECTOR</td>
<td>$A0</td>
<td>III READ ADDRESS</td>
<td>$C0</td>
</tr>
<tr>
<td>III READ TRACK</td>
<td>$E4</td>
<td>III WRITE TRACK</td>
<td>$F4</td>
</tr>
<tr>
<td>IV FORCE INTERRUPT</td>
<td>$D0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Status Bit</th>
<th>Type I</th>
<th>Read Address/Sector/Track</th>
<th>Write Sector/Track</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0</td>
<td>BUSY</td>
<td>BUSY</td>
<td>BUSY</td>
</tr>
<tr>
<td>S1</td>
<td>INDEX</td>
<td>DRQ</td>
<td>DRQ</td>
</tr>
<tr>
<td>S2</td>
<td>TRACK 0</td>
<td>LOST DATA</td>
<td>LOST DATA</td>
</tr>
<tr>
<td>S3</td>
<td>CRC ERROR</td>
<td>CRC ERROR (EXCEPT TRACK)</td>
<td>CRC ERROR (EXCEPT TRACK)</td>
</tr>
<tr>
<td>S4</td>
<td>SEEK ERROR</td>
<td>RNF (EXCEPT TRACK)</td>
<td>RNF (EXCEPT TRACK)</td>
</tr>
<tr>
<td>S5</td>
<td>HEAD LOADED</td>
<td>RECORD TYPE (SECTOR ONLY)</td>
<td>WRITE FAULT</td>
</tr>
<tr>
<td>S6</td>
<td>WRITE PROTECT</td>
<td>WRITE PROTECT</td>
<td></td>
</tr>
<tr>
<td>S7</td>
<td>NOT READY</td>
<td>NOT READY</td>
<td>NOT READY</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Status Registers</th>
<th>Type</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>FF49</td>
<td>RMB 1</td>
<td>TRACK REGISTER</td>
</tr>
<tr>
<td>FF4A</td>
<td>RMB 1</td>
<td>SECTOR REGISTER</td>
</tr>
<tr>
<td>FF4B</td>
<td>RMB 1</td>
<td>DATA REGISTER</td>
</tr>
<tr>
<td>FF4C</td>
<td>RMB 4</td>
<td>FDREG IMAGES</td>
</tr>
<tr>
<td>FF50</td>
<td>RMB 16</td>
<td>UNUSED SPACE</td>
</tr>
<tr>
<td>FF60</td>
<td>RMB 1</td>
<td>X COORDINATE FOR X-PAD</td>
</tr>
<tr>
<td>FF61</td>
<td>RMB 1</td>
<td>Y COORDINATE FOR X-PAD</td>
</tr>
<tr>
<td>FF62</td>
<td>RMB 1</td>
<td>STATUS REGISTER FOR X-PAD</td>
</tr>
<tr>
<td>FF63</td>
<td>RMB 5</td>
<td>UNUSED</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Port</th>
<th>Type</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>FF90</td>
<td>INIT0</td>
<td>RMB 1</td>
</tr>
<tr>
<td>FF91</td>
<td>INIT1</td>
<td>RMB 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Port</th>
<th>Type</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC0</td>
<td>ROM MAPPING</td>
<td></td>
</tr>
<tr>
<td>MC1</td>
<td>ROM MAPPING</td>
<td></td>
</tr>
<tr>
<td>MC2</td>
<td>STANDARD SCS</td>
<td></td>
</tr>
<tr>
<td>MC3</td>
<td>DRAM AT 0xFEXX IS CONSTANT</td>
<td></td>
</tr>
<tr>
<td>MC4</td>
<td>CHIP IRQ OUTPUT ENABLED</td>
<td></td>
</tr>
<tr>
<td>MC5</td>
<td>CHIP IRQ OUTPUT ENABLED</td>
<td></td>
</tr>
<tr>
<td>MC6</td>
<td>MMU ENABLED</td>
<td></td>
</tr>
<tr>
<td>MC7</td>
<td>MMU ENABLED</td>
<td></td>
</tr>
<tr>
<td>MC8</td>
<td>MMU ENABLED</td>
<td></td>
</tr>
<tr>
<td>MC9</td>
<td>MMU ENABLED</td>
<td></td>
</tr>
<tr>
<td>MC10</td>
<td>MMU ENABLED</td>
<td></td>
</tr>
<tr>
<td>MC11</td>
<td>MMU ENABLED</td>
<td></td>
</tr>
<tr>
<td>MC12</td>
<td>MMU ENABLED</td>
<td></td>
</tr>
<tr>
<td>MC13</td>
<td>MMU ENABLED</td>
<td></td>
</tr>
<tr>
<td>MC14</td>
<td>MMU ENABLED</td>
<td></td>
</tr>
<tr>
<td>MC15</td>
<td>MMU ENABLED</td>
<td></td>
</tr>
</tbody>
</table>

A10
<table>
<thead>
<tr>
<th>Address</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0741</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0742 FF92</td>
<td>IRQENR RMB 1</td>
<td>IRQ INTERRUPT ENABLE REGISTER</td>
</tr>
<tr>
<td>0743</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0744 FF92</td>
<td>BIT0 E10</td>
<td>CARTRIDGE IRQ ENABLED</td>
</tr>
<tr>
<td>0745</td>
<td>BIT1 E11</td>
<td>KEYBOARD IRQ ENABLED</td>
</tr>
<tr>
<td>0746</td>
<td>BIT2 E12</td>
<td>SERIAL DATA IRQ ENABLED</td>
</tr>
<tr>
<td>0747</td>
<td>BIT3 V Bord</td>
<td>VERTICAL BORDER IRQ ENABLED</td>
</tr>
<tr>
<td>0748</td>
<td>BIT4 H Bord</td>
<td>HORIZONTAL BORDER IRQ ENABLED</td>
</tr>
<tr>
<td>0749</td>
<td>BIT5 TMR</td>
<td>INTERRUPT FROM TIMER ENABLED</td>
</tr>
<tr>
<td>0750</td>
<td>BIT6</td>
<td></td>
</tr>
<tr>
<td>0751</td>
<td>BIT7</td>
<td></td>
</tr>
<tr>
<td>0752</td>
<td>FF93</td>
<td>FIRQ INTERRUPT ENABLE REGISTER</td>
</tr>
<tr>
<td>0753</td>
<td>BIT0 EI0</td>
<td>CARTRIDGE FIRQ ENABLED</td>
</tr>
<tr>
<td>0754</td>
<td>BIT1 EI1</td>
<td>KEYBOARD FIRQ ENABLED</td>
</tr>
<tr>
<td>0755</td>
<td>BIT2 EI2</td>
<td>SERIAL DATA FIRQ ENABLED</td>
</tr>
<tr>
<td>0756</td>
<td>BIT3 V Bord</td>
<td>VERTICAL BORDER FIRQ ENABLED</td>
</tr>
<tr>
<td>0757</td>
<td>BIT4 H Bord</td>
<td>HORIZONTAL BORDER FIRQ ENABLED</td>
</tr>
<tr>
<td>0758</td>
<td>BIT5 TMR</td>
<td>INTERRUPT FROM TIMER ENABLED</td>
</tr>
<tr>
<td>0759</td>
<td>BIT6</td>
<td></td>
</tr>
<tr>
<td>0760</td>
<td>BIT7</td>
<td></td>
</tr>
<tr>
<td>0761</td>
<td>FF94</td>
<td>TIMER REGISTER</td>
</tr>
<tr>
<td>0762</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0763</td>
<td>FF96</td>
<td>RESERVES FOR FUTURE EXPANSION</td>
</tr>
<tr>
<td>0764</td>
<td>FF98</td>
<td>VIDEO MODE REGISTER</td>
</tr>
<tr>
<td>0765</td>
<td>LPR2 LPR1 LPR0</td>
<td>LINES PER CHARACTER ROW</td>
</tr>
<tr>
<td>0766</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0767</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0768</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0769</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0770</td>
<td>BIT0 LPR0</td>
<td>LINES PER ROW (SEE TABLE BELOW)</td>
</tr>
<tr>
<td>0771</td>
<td>BIT1 LPR1</td>
<td>LINES PER ROW (SEE TABLE BELOW)</td>
</tr>
<tr>
<td>0772</td>
<td>BIT2 LPR2</td>
<td>LINES PER ROW (SEE TABLE BELOW)</td>
</tr>
<tr>
<td>0773</td>
<td>BIT3 HS0</td>
<td>1 = 50 Hz VERTICAL REFRESH</td>
</tr>
<tr>
<td>0774</td>
<td>BIT4 MOCH</td>
<td>1 = MONOCHROME (ON COMPOSITE)</td>
</tr>
<tr>
<td>0775</td>
<td>BIT5 BPI</td>
<td>1 = BURST PHASE INVERTED</td>
</tr>
<tr>
<td>0776</td>
<td>BIT6</td>
<td></td>
</tr>
<tr>
<td>0777</td>
<td>BIT7 BP</td>
<td>0 = ALPHA, 1 = BIT PLANE</td>
</tr>
<tr>
<td>0778</td>
<td>LPR2 LPR1 LPR0</td>
<td>LINES PER FIELD (SEE TABLE BELOW)</td>
</tr>
<tr>
<td>0779</td>
<td>0 0 0 1</td>
<td>(GRAPHICS MODES)</td>
</tr>
<tr>
<td>0780</td>
<td>0 0 1 2</td>
<td>(COCO 1 &amp; 2 ONLY)</td>
</tr>
<tr>
<td>0781</td>
<td>0 1 0 3</td>
<td>(COCO 1 &amp; 2 ONLY)</td>
</tr>
<tr>
<td>0782</td>
<td>0 1 1 8</td>
<td></td>
</tr>
<tr>
<td>0783</td>
<td>1 0 0 9</td>
<td></td>
</tr>
<tr>
<td>0784</td>
<td>1 0 1 (RESERVED)</td>
<td></td>
</tr>
<tr>
<td>0785</td>
<td>1 1 0 12</td>
<td></td>
</tr>
<tr>
<td>0786</td>
<td>1 1 1 (RESERVED)</td>
<td></td>
</tr>
<tr>
<td>0787</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0788</td>
<td>FF99</td>
<td>VIDEO MODE REGISTER</td>
</tr>
<tr>
<td>0789</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0790</td>
<td>FF99</td>
<td>VIDEO MODE REGISTER</td>
</tr>
<tr>
<td>0791</td>
<td>BIT0 CRE50</td>
<td>COLOR RESOLUTION</td>
</tr>
<tr>
<td>0792</td>
<td>BIT1 CRE51</td>
<td>COLOR RESOLUTION</td>
</tr>
<tr>
<td>0793</td>
<td>BIT2 HRES0</td>
<td>HORIZONTAL RESOLUTION</td>
</tr>
<tr>
<td>0794</td>
<td>BIT3 HRES1</td>
<td>HORIZONTAL RESOLUTION</td>
</tr>
<tr>
<td>0795</td>
<td>BIT4 HRES2</td>
<td>HORIZONTAL RESOLUTION</td>
</tr>
<tr>
<td>0796</td>
<td>BIT5 LPF0</td>
<td>LINES PER FIELD (SEE TABLE BELOW)</td>
</tr>
<tr>
<td>0797</td>
<td>BIT6 LPF1</td>
<td>LINES PER FIELD (SEE TABLE BELOW)</td>
</tr>
<tr>
<td>0798</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0799</td>
<td>LPF1 LPF0</td>
<td>LINES PER FIELD</td>
</tr>
<tr>
<td>0800</td>
<td>0 0 192</td>
<td></td>
</tr>
<tr>
<td>0801</td>
<td>0 1 200</td>
<td></td>
</tr>
<tr>
<td>0802</td>
<td>1 0 RESERVED</td>
<td></td>
</tr>
<tr>
<td>0803</td>
<td>1 1 225</td>
<td></td>
</tr>
<tr>
<td>0804</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0805</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0806</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0807</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0808</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0809</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0810</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0811</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0812</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0813</td>
<td>640x4 1 1 0 1</td>
<td></td>
</tr>
<tr>
<td>0814</td>
<td>640x2 1 0 1 0 0</td>
<td></td>
</tr>
</tbody>
</table>

*ALPHA: BP = 0, COCO = 0

**MODE**

<table>
<thead>
<tr>
<th>CHARACTER</th>
<th>HRES2</th>
<th>HRES1</th>
<th>HRES0</th>
<th>CRE50</th>
<th>CRE51</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**GRAPHICS: BP = 1, COCO = 0

<table>
<thead>
<tr>
<th>PIXELS COLORS</th>
<th>HRES2</th>
<th>HRES1</th>
<th>HRES0</th>
<th>CRE50</th>
<th>CRE51</th>
</tr>
</thead>
<tbody>
<tr>
<td>640x4</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>640x2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Address</td>
<td>Memory Size</td>
<td>Origin</td>
<td>Bit 0</td>
<td>Bit 1</td>
<td>Bit 2</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td>--------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>0815</td>
<td>512x4</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0816</td>
<td>512x2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0817</td>
<td>320x16</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0818</td>
<td>320x4</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0819</td>
<td>256x16</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0820</td>
<td>256x4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0821</td>
<td>256x2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0822</td>
<td>160x16</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**MEMORY MAP**

**ORIGIN:SPECTRAL ASSOC**

**REVISED:12/26/1999 WALTER K ZYDHEK**

---

**MEMORY MAP**

**ORIGIN:SPECTRAL ASSOC**

**REVISED:12/26/1999 WALTER K ZYDHEK**

<table>
<thead>
<tr>
<th>Address</th>
<th>Pixel Size</th>
<th>Display Mode</th>
<th>Reg. FF22</th>
</tr>
</thead>
<tbody>
<tr>
<td>0815</td>
<td>512x4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0816</td>
<td>512x2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0817</td>
<td>320x16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0818</td>
<td>320x4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0819</td>
<td>256x16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0820</td>
<td>256x4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0821</td>
<td>256x2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0822</td>
<td>160x16</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**DISPLAY MODE**

**REG. FF22**

---

**ALPHANUMERIC MODES**

**TEXT SCREEN MEMORY**

<table>
<thead>
<tr>
<th>Address</th>
<th>EVEN BYTE</th>
<th>CHARACTERS</th>
<th>REG. FF22</th>
</tr>
</thead>
<tbody>
<tr>
<td>0841</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0842</td>
<td>BIT0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0843</td>
<td>BIT1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0844</td>
<td>BIT2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0845</td>
<td>BIT3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0846</td>
<td>BIT4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0847</td>
<td>BIT5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0848</td>
<td>BIT6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0849</td>
<td>BIT7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**ODD BYTE**

<table>
<thead>
<tr>
<th>Address</th>
<th>BACKGROUND COLOR</th>
<th>FOREGROUND COLOR</th>
<th>REG. FF22</th>
</tr>
</thead>
<tbody>
<tr>
<td>0851</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0852</td>
<td>BIT0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0853</td>
<td>BIT1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0854</td>
<td>BIT2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0855</td>
<td>BIT3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0856</td>
<td>BIT4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0857</td>
<td>BIT5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0858</td>
<td>BIT6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**ATTRIBUTES NOT AVAILABLE WHEN COCO = 1**

---

**GRAPHICS MODES**

16 COLOR MODES: (CRES1=1, CRES0 = 0)

<table>
<thead>
<tr>
<th>Address</th>
<th>BYTE FROM DRAm</th>
<th>PA0, SECOND PIXEL</th>
<th>PA1, SECOND PIXEL</th>
<th>PA2, SECOND PIXEL</th>
<th>PA3, SECOND PIXEL</th>
<th>PA4, FIRST PIXEL</th>
<th>PA5, FIRST PIXEL</th>
<th>PA6, FIRST PIXEL</th>
<th>PA7, PA3, FIRST PIXEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0863</td>
<td>BIT0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0864</td>
<td>BIT1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0865</td>
<td>BIT2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0866</td>
<td>BIT3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0867</td>
<td>BIT4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0868</td>
<td>BIT5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0869</td>
<td>BIT6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0870</td>
<td>BIT7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

4 COLOR MODES: (CRES1=0, CRES0 = 1)

<table>
<thead>
<tr>
<th>Address</th>
<th>BYTE FROM DRAm</th>
<th>PA0, FOURTH PIXEL</th>
<th>PA1, FOURTH PIXEL</th>
<th>PA2, THIRD PIXEL</th>
<th>PA3, THIRD PIXEL</th>
<th>PA4, SECOND PIXEL</th>
<th>PA5, SECOND PIXEL</th>
<th>PA6, FIRST PIXEL</th>
<th>PA7, PA1, FIRST PIXEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0873</td>
<td>BIT0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0874</td>
<td>BIT1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0875</td>
<td>BIT2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0876</td>
<td>BIT3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0877</td>
<td>BIT4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0878</td>
<td>BIT5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0879</td>
<td>BIT6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0880</td>
<td>BIT7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

2 COLOR MODES: (CRES1=0, CRES0 = 0)

<table>
<thead>
<tr>
<th>Address</th>
<th>BYTE FROM DRAm</th>
<th>PA0, EIGHTH PIXEL</th>
<th>PA1, SEVENTH PIXEL</th>
<th>PA2, SIXTH PIXEL</th>
<th>PA3, FIFTH PIXEL</th>
<th>PA4, FOURTH PIXEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0883</td>
<td>BIT0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0884</td>
<td>BIT1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0885</td>
<td>BIT2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0886</td>
<td>BIT3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0887</td>
<td>BIT4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0888</td>
<td>BIT5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Memory Map

**Origin:** Spectral Assoc  
**Revised:** 12/26/1999 Walter K Zydhek

<table>
<thead>
<tr>
<th>Address</th>
<th>Bit 5</th>
<th>PA0, Third Pixel</th>
</tr>
</thead>
<tbody>
<tr>
<td>0889</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>Bit 6</th>
<th>PA0, Second Pixel</th>
</tr>
</thead>
<tbody>
<tr>
<td>0890</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>Bit 7</th>
<th>PA0, First Pixel</th>
</tr>
</thead>
<tbody>
<tr>
<td>0891</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Palette Addresses

<table>
<thead>
<tr>
<th>Address</th>
<th>PA3</th>
<th>PA2</th>
<th>PA1</th>
<th>PA0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0892</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0893</td>
<td>FF00</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0894</td>
<td>FF01</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0895</td>
<td>FF02</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0896</td>
<td>FF03</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0897</td>
<td>FF04</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0898</td>
<td>FF05</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0899</td>
<td>FF06</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0900</td>
<td>FF07</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0901</td>
<td>FF08</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>V.BORDER</th>
<th>RMB</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0910</td>
<td>FF9A</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>BIT0</th>
<th>BLU0</th>
<th>BLUE LSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>0911</td>
<td>BIT1</td>
<td>GRN0</td>
<td>GREEN LSB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>BIT2</th>
<th>RED0</th>
<th>RED LSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>0912</td>
<td>BIT3</td>
<td>BLU1</td>
<td>BLUE MSB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>BIT4</th>
<th>GRN1</th>
<th>GREEN MSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>0913</td>
<td>BIT5</td>
<td>RED1</td>
<td>RED MSB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>BIT6</th>
</tr>
</thead>
<tbody>
<tr>
<td>0914</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>BIT7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0915</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>RMB</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0916</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>V.SCROLL</th>
<th>RMB</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0917</td>
<td>FF9C</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>BIT0</th>
<th>VSC0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0918</td>
<td>BIT1</td>
<td>VSC1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>BIT2</th>
<th>VSC2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0919</td>
<td>BIT3</td>
<td>VSC3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>BIT4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0920</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>BIT5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0921</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>BIT6</th>
</tr>
</thead>
<tbody>
<tr>
<td>0922</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>Reserved</th>
</tr>
</thead>
<tbody>
<tr>
<td>0923</td>
<td>FF9B</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>VERTICAL SCROLL REGISTER</th>
<th>RMB</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0924</td>
<td>FF9C</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>BIT0</th>
<th>Y1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0925</td>
<td>BIT1</td>
<td>Y2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>BIT2</th>
<th>Y3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0926</td>
<td>BIT3</td>
<td>Y4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>BIT4</th>
<th>Y5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0927</td>
<td>BIT5</td>
<td>Y6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>BIT6</th>
<th>Y7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0928</td>
<td>BIT7</td>
<td>Y8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>BIT0</th>
<th>Y9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0929</td>
<td>BIT1</td>
<td>Y10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>Y9-Y15</th>
<th>NOT EFFECTIVE, AND ARE CONTROLLED BY SAM BITS F8-F6. ALSO IN COCO MODE, Y16-Y18 SHOULD BE 1, ALL OTHERS 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0930</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>Y16-Y18</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0931</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>VERTICAL OFFSET 1 REGISTER</th>
<th>RMB</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0932</td>
<td>FF9D</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>BIT0</th>
<th>Y11</th>
</tr>
</thead>
<tbody>
<tr>
<td>0933</td>
<td>BIT1</td>
<td>Y12</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>BIT2</th>
<th>Y13</th>
</tr>
</thead>
<tbody>
<tr>
<td>0934</td>
<td>BIT3</td>
<td>Y14</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>BIT4</th>
<th>Y15</th>
</tr>
</thead>
<tbody>
<tr>
<td>0935</td>
<td>BIT5</td>
<td>Y16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>BIT6</th>
<th>Y17</th>
</tr>
</thead>
<tbody>
<tr>
<td>0936</td>
<td>BIT7</td>
<td>Y18</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>Y18</th>
</tr>
</thead>
<tbody>
<tr>
<td>0937</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>VERTICAL OFFSET 0 REGISTER</th>
<th>RMB</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0938</td>
<td>FF9E</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>BIT0</th>
<th>Y3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0939</td>
<td>BIT1</td>
<td>Y4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>BIT2</th>
<th>Y5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0940</td>
<td>BIT3</td>
<td>Y6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>BIT4</th>
<th>Y7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0941</td>
<td>BIT5</td>
<td>Y8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>BIT6</th>
<th>Y9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0942</td>
<td>BIT7</td>
<td>Y10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>Y10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0943</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>HORIZONTAL OFFSET 0 REGISTER</th>
<th>RMB</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0944</td>
<td>FF9F</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>BIT0</th>
<th>X0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0945</td>
<td>BIT1</td>
<td>X1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>X1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0946</td>
<td></td>
</tr>
</tbody>
</table>
### Memory Map

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

<table>
<thead>
<tr>
<th>Memory Address</th>
<th>Corresponding Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0963</td>
<td>BIT2 X2</td>
<td>Horizontal Offset Address</td>
</tr>
<tr>
<td>0964</td>
<td>BIT3 X3</td>
<td>Horizontal Offset Address</td>
</tr>
<tr>
<td>0965</td>
<td>BIT4 X4</td>
<td>Horizontal Offset Address</td>
</tr>
<tr>
<td>0966</td>
<td>BIT5 X5</td>
<td>Horizontal Offset Address</td>
</tr>
<tr>
<td>0967</td>
<td>BIT6 X6</td>
<td>Horizontal Offset Address</td>
</tr>
<tr>
<td>0968</td>
<td>BIT7 XVEN</td>
<td>Horizontal Virtual Enable</td>
</tr>
</tbody>
</table>

* WHEN ENABLED, A HORIZONTAL SCREEN WIDTH OF 128 BYTES REGARDLESS OF THE HRES BITS AND CRES BITS SELECTED. THIS WILL ALLOW A 'VIRTUAL' SCREEN SOMewhat LARGER THAN THE DISPLAYED SCREEN. THE USER CAN MOVE THIS 'WINDOW' (THE DISPLAYED SCREEN) BY MEANS OF THE HORIZONTAL OFFSET BITS. IN CHARACTER MODE, THE SCREEN WIDTH IS 128 CHARACTERS REGARDLESS OF ATTRIBUTE (OR 64, IF DOUBLE-WIDE IS SELECTED).

#### FF00 - FF0F

**MMUREG**

**RMB 16**

**MEMORY MANAGEMENT UNIT REGISTERS (6 BITS)**

**CORRESPONDING MEMORY ADDRESS**

<table>
<thead>
<tr>
<th>Address Range</th>
<th>TR A15 A14 A13</th>
<th>MMU Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>X0000 - X1FFF</td>
<td>0 0 0 0 0 0 FFA0</td>
<td></td>
</tr>
<tr>
<td>X2000 - X3FFF</td>
<td>0 0 0 1 1 FFA1</td>
<td></td>
</tr>
<tr>
<td>X4000 - X5FFF</td>
<td>0 0 1 0 0 FFA2</td>
<td></td>
</tr>
<tr>
<td>X6000 - X7FFF</td>
<td>0 1 0 1 1 FFA3</td>
<td></td>
</tr>
<tr>
<td>X8000 - X9FFF</td>
<td>0 1 1 0 0 FFA4</td>
<td></td>
</tr>
<tr>
<td>XA000 - XBFFF</td>
<td>0 1 1 1 1 FFA5</td>
<td></td>
</tr>
<tr>
<td>XC000 - XDFFF</td>
<td>1 0 1 1 0 FFA6</td>
<td></td>
</tr>
<tr>
<td>XE000 - XFFFF</td>
<td>1 1 1 1 1 FFA7</td>
<td></td>
</tr>
</tbody>
</table>

#### FFH0 - FFH7

**PALETREG**

**RMB 16**

**COLOR PALETTE REGISTERS (6 BITS)**

<table>
<thead>
<tr>
<th>Address Range</th>
<th>RMB 16</th>
<th>DATA BIT</th>
<th>TR A15 A14 A13</th>
<th>MMU Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFB0 - FFB7</td>
<td>0000</td>
<td>D5 D4 D3 D2 D1 D0</td>
<td>FFB0 - FFB7</td>
<td>FFB0 - FFB7</td>
</tr>
<tr>
<td>FFB8 - FFBF</td>
<td>0001</td>
<td>R0 R1 B1 G1 I1</td>
<td>FFB8 - FFBF</td>
<td>FFB8 - FFBF</td>
</tr>
<tr>
<td>FF00 - FF0F</td>
<td>0010</td>
<td>P0 P1 P2 P3</td>
<td>FF00 - FF0F</td>
<td>FF00 - FF0F</td>
</tr>
<tr>
<td>FF10 - FF1F</td>
<td>0011</td>
<td>FF10 FF11 FF12</td>
<td>FF10 - FF1F</td>
<td>FF10 - FF1F</td>
</tr>
<tr>
<td>FF20 - FF2F</td>
<td>0012</td>
<td>FF20 FF21 FF22</td>
<td>FF20 - FF2F</td>
<td>FF20 - FF2F</td>
</tr>
<tr>
<td>FF30 - FF3F</td>
<td>0013</td>
<td>FF30 FF31 FF32</td>
<td>FF30 - FF3F</td>
<td>FF30 - FF3F</td>
</tr>
<tr>
<td>FF40 - FF4F</td>
<td>0014</td>
<td>FF40 FF41 FF42</td>
<td>FF40 - FF4F</td>
<td>FF40 - FF4F</td>
</tr>
<tr>
<td>FF50 - FF5F</td>
<td>0015</td>
<td>FF50 FF51 FF52</td>
<td>FF50 - FF5F</td>
<td>FF50 - FF5F</td>
</tr>
<tr>
<td>FF60 - FF6F</td>
<td>0016</td>
<td>FF60 FF61 FF62</td>
<td>FF60 - FF6F</td>
<td>FF60 - FF6F</td>
</tr>
<tr>
<td>FF70 - FF7F</td>
<td>0017</td>
<td>FF70 FF71 FF72</td>
<td>FF70 - FF7F</td>
<td>FF70 - FF7F</td>
</tr>
<tr>
<td>FF80 - FF8F</td>
<td>0018</td>
<td>FF80 FF81 FF82</td>
<td>FF80 - FF8F</td>
<td>FF80 - FF8F</td>
</tr>
<tr>
<td>FF90 - FF9F</td>
<td>0019</td>
<td>FF90 FF91 FF92</td>
<td>FF90 - FF9F</td>
<td>FF90 - FF9F</td>
</tr>
<tr>
<td>FFA0 - FFAF</td>
<td>0020</td>
<td>FF9A FF9B FF9C</td>
<td>FF90 - FF9F</td>
<td>FF90 - FF9F</td>
</tr>
<tr>
<td>FFB0 - FFB7</td>
<td>0021</td>
<td>FFB0 FFB1 FFB2</td>
<td>FFB0 - FFB7</td>
<td>FFB0 - FFB7</td>
</tr>
<tr>
<td>FFB8 - FFBF</td>
<td>0022</td>
<td>FFB0 FFB1 FFB2</td>
<td>FFB0 - FFB7</td>
<td>FFB0 - FFB7</td>
</tr>
<tr>
<td>FF00 - FF0F</td>
<td>0023</td>
<td>FFB0 FFB1 FFB2</td>
<td>FFB0 - FFB7</td>
<td>FFB0 - FFB7</td>
</tr>
<tr>
<td>FF10 - FF1F</td>
<td>0024</td>
<td>FFB0 FFB1 FFB2</td>
<td>FFB0 - FFB7</td>
<td>FFB0 - FFB7</td>
</tr>
<tr>
<td>FF20 - FF2F</td>
<td>0025</td>
<td>FFB0 FFB1 FFB2</td>
<td>FFB0 - FFB7</td>
<td>FFB0 - FFB7</td>
</tr>
<tr>
<td>FF30 - FF3F</td>
<td>0026</td>
<td>FFB0 FFB1 FFB2</td>
<td>FFB0 - FFB7</td>
<td>FFB0 - FFB7</td>
</tr>
<tr>
<td>FF40 - FF4F</td>
<td>0027</td>
<td>FFB0 FFB1 FFB2</td>
<td>FFB0 - FFB7</td>
<td>FFB0 - FFB7</td>
</tr>
<tr>
<td>FF50 - FF5F</td>
<td>0028</td>
<td>FFB0 FFB1 FFB2</td>
<td>FFB0 - FFB7</td>
<td>FFB0 - FFB7</td>
</tr>
<tr>
<td>FF60 - FF6F</td>
<td>0029</td>
<td>FFB0 FFB1 FFB2</td>
<td>FFB0 - FFB7</td>
<td>FFB0 - FFB7</td>
</tr>
<tr>
<td>FF70 - FF7F</td>
<td>0030</td>
<td>FFB0 FFB1 FFB2</td>
<td>FFB0 - FFB7</td>
<td>FFB0 - FFB7</td>
</tr>
<tr>
<td>FF80 - FF8F</td>
<td>0031</td>
<td>FFB0 FFB1 FFB2</td>
<td>FFB0 - FFB7</td>
<td>FFB0 - FFB7</td>
</tr>
<tr>
<td>FF90 - FF9F</td>
<td>0032</td>
<td>FFB0 FFB1 FFB2</td>
<td>FFB0 - FFB7</td>
<td>FFB0 - FFB7</td>
</tr>
<tr>
<td>FFA0 - FFAF</td>
<td>0033</td>
<td>FFB0 FFB1 FFB2</td>
<td>FFB0 - FFB7</td>
<td>FFB0 - FFB7</td>
</tr>
<tr>
<td>FFB0 - FFB7</td>
<td>0034</td>
<td>FFB0 FFB1 FFB2</td>
<td>FFB0 - FFB7</td>
<td>FFB0 - FFB7</td>
</tr>
<tr>
<td>FFB8 - FFBF</td>
<td>0035</td>
<td>FFB0 FFB1 FFB2</td>
<td>FFB0 - FFB7</td>
<td>FFB0 - FFB7</td>
</tr>
<tr>
<td>FF00 - FF0F</td>
<td>0036</td>
<td>FFB0 FFB1 FFB2</td>
<td>FFB0 - FFB7</td>
<td>FFB0 - FFB7</td>
</tr>
</tbody>
</table>

* FOR COCO COMPATIBILITY, THE FOLLOWING SHOULD BE LOADED ON INITIALIZATION (RGB VALUES) FOR PAL VERSION, IGNORE TABLE FOR COMPOSITE
<table>
<thead>
<tr>
<th>Address</th>
<th>Hexadecimal</th>
<th>Instruction</th>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1037 FF6</td>
<td>F8CLR</td>
<td>RMB</td>
<td>1</td>
<td>CLEAR COCO GRAPHICS OFFSET F8</td>
</tr>
<tr>
<td>1038 FF7</td>
<td>F8SET</td>
<td>RMB</td>
<td>1</td>
<td>SET COCO GRAPHICS OFFSET F8</td>
</tr>
<tr>
<td>1039 FF8</td>
<td>F1CLR</td>
<td>RMB</td>
<td>1</td>
<td>CLEAR COCO GRAPHICS OFFSET F1</td>
</tr>
<tr>
<td>1040 FF9</td>
<td>F1SET</td>
<td>RMB</td>
<td>1</td>
<td>SET COCO GRAPHICS OFFSET F1</td>
</tr>
<tr>
<td>1041 FFCA</td>
<td>F2CLR</td>
<td>RMB</td>
<td>1</td>
<td>CLEAR COCO GRAPHICS OFFSET F2</td>
</tr>
<tr>
<td>1042 FF8</td>
<td>F2SET</td>
<td>RMB</td>
<td>1</td>
<td>SET COCO GRAPHICS OFFSET F2</td>
</tr>
<tr>
<td>1043 FFCC</td>
<td>F3CLR</td>
<td>RMB</td>
<td>1</td>
<td>CLEAR COCO GRAPHICS OFFSET F3</td>
</tr>
<tr>
<td>1044 FFD0</td>
<td>F3SET</td>
<td>RMB</td>
<td>1</td>
<td>SET COCO GRAPHICS OFFSET F3</td>
</tr>
<tr>
<td>1045 FFC</td>
<td>F4CLR</td>
<td>RMB</td>
<td>1</td>
<td>CLEAR COCO GRAPHICS OFFSET F4</td>
</tr>
<tr>
<td>1046 FFCE</td>
<td>F4SET</td>
<td>RMB</td>
<td>1</td>
<td>SET COCO GRAPHICS OFFSET F4</td>
</tr>
<tr>
<td>1047 FFD0</td>
<td>F5CLR</td>
<td>RMB</td>
<td>1</td>
<td>CLEAR COCO GRAPHICS OFFSET F5</td>
</tr>
<tr>
<td>1048 FFD1</td>
<td>F5SET</td>
<td>RMB</td>
<td>1</td>
<td>SET COCO GRAPHICS OFFSET F5</td>
</tr>
<tr>
<td>1049 FFD2</td>
<td>F6CLR</td>
<td>RMB</td>
<td>1</td>
<td>CLEAR COCO GRAPHICS OFFSET F6</td>
</tr>
<tr>
<td>1050 FFD3</td>
<td>F6SET</td>
<td>RMB</td>
<td>1</td>
<td>SET COCO GRAPHICS OFFSET F6</td>
</tr>
<tr>
<td>1051 FFD4</td>
<td></td>
<td>RMB</td>
<td>4</td>
<td>RESERVED</td>
</tr>
<tr>
<td>1052 FFD8</td>
<td>R1CLR</td>
<td>RMB</td>
<td>1</td>
<td>CLEAR CPU RATE, (0.89 MHz)</td>
</tr>
<tr>
<td>1053 FFD9</td>
<td>R1SET</td>
<td>RMB</td>
<td>1</td>
<td>SET CPU RATE, (1.78 MHz)</td>
</tr>
<tr>
<td>1054 FFD0</td>
<td></td>
<td>RMB</td>
<td>4</td>
<td>RESERVED</td>
</tr>
<tr>
<td>1055 FFD8</td>
<td>ROMCLR</td>
<td>RMB</td>
<td>1</td>
<td>ROM DISABLED</td>
</tr>
<tr>
<td>1056 FFDF</td>
<td>ROMSET</td>
<td>RMB</td>
<td>1</td>
<td>ROM ENABLED</td>
</tr>
<tr>
<td>1057</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1058 FFE0</td>
<td>*</td>
<td>RMB</td>
<td>18</td>
<td>RESERVED FOR FUTURE MPU ENHANCEMENTS</td>
</tr>
<tr>
<td>1059</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1060 FFF2</td>
<td>SWI3</td>
<td>RMB</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>1061 FFF4</td>
<td>SWI2</td>
<td>RMB</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>1062 FFF6</td>
<td>FIRQ</td>
<td>RMB</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>1063 FFF8</td>
<td>IRQ</td>
<td>RMB</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>1064 FFFA</td>
<td>SWI</td>
<td>RMB</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>1065 FFFC</td>
<td>NMI</td>
<td>RMB</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>1066 FFDE</td>
<td>RESETV</td>
<td>RMB</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
0193 B286 96 8B    L8296 FDB PCLEAR    PCLEAR C8
0194 B288 95 46    L8288 FDB COLOR    COLOR C1
0195 B28A 9E 90    L828A FDB CIRCLE    CIRCLE C2
0196 B28C 9C 8E    L828C FDB PAINT    PAINT C3
0197 B28E 97 55    L828E FDB GET    GET C4
0198 B28F 98 7A    L828F FDB PUT    PUT C5
0199 B2A2 2C 86    L82A2 FDB DRAW    DRAW C6
0200 B2A4 97 23    L82A4 FDB PCOPY    PCOPY C7
0201 B2A6 96 21    L82A6 FDB PMOD    PMOD C7
0202 B2A8 9A 22    L82A8 FDB PLAY    PLAY C9
0203 B2A9 8C 18    L82A9 FDB DLOAD    DLOAD C9
0204 B2AC 8A 89    L82AC FDB RENUM    RENUM C8

0205 * SECONDARY FUNCTION FOR EXTENDED BASIC
0206 * TOKENS ARE PRECEDED WITH AN 8F BYTE
0207 *
0208 * TOKEN #
0209 B216 41 94 CE    L8216 FCC 'AT',880+'N'    94
0210 B21A 43 4F D3    L821A FCC 'CD',880+'S'    95
0211 B21A 34 41 CE    L821A FCC 'TA',880+'N'    96
0212 B217 45 5B DB    L8217 FCC 'EX',880+'P'    97
0213 B21A 46 49 0B    L821A FCC 'FI',880+'X'    98
0214 B21D 40 4F C7    L821D FCC 'L0',880+'G'    99
0215 B218 56 40 D3    L8218 FCC 'PD',880+'S'    9A
0216 B232 53 51 D2    L8232 FCC 'SO',880+'R'    9B
0217 B236 4B 45 58 A8    L8236 FCC 'HEx',880+'1'    9C
0218 B23A 56 41 52 50 54 02 L823A FCC 'VARP',880+'R'    9D
0219 B248 49 4E 53 54 02 L8248 FCC 'TINST',880+'R'    9E
0220 B254 5A 49 40 49 D2 L8254 FCC 'TIME',880+'R'    9F
0221 B24A 58 58 4F 49 4E 04 L824A FCC 'POOP',880+'T'    A0
0222 B258 53 55 52 49 4E 47 L8258 FCC 'STRING',880+'S'    A1
0223 B256 8A 84

0224 * JUMP TABLE FOR EXTENDED BASIC SECONDARY FUNCTIONS
0225 *
0226 B257 B3 8B    L8257 FDB ATN    ATN 94
0227 B259 B3 7B    L8259 FDB COS    COS 95
0228 B258 B3 61    L8258 FDB TAN    TAN 96
0229 B25D B4 F2    L825D FDB EXP    EXP 97
0230 B25F B5 24    L825F FDB FIX    FIX 98
0231 B261 B4 46    L8261 FDB LOG    LOG 99
0232 B263 B6 AC    L8263 FDB POS    POS 9A
0233 B265 B4 8B    L8265 FDB SQR    SQR 9B
0234 B267 B8 BB    L8267 FDB HEXDOL    HEXDOL 9C
0235 B269 B6 8E    L8269 FDB VARPTR    VARPTR 9D
0236 B26B B7 81   L826B FDB INSTR    INSTR 9E
0237 B26D B9 6B    L826D FDB TIMER    TIMER 9F
0238 B26F B9 33    L826F FDB PPOINT    PPOINT A0
0239 B271 B7 4E    L8271 FDB STRING    STRING A1

0240 * CONSOLE OUT RAM HOOK
0241 B273 8D 6F    L8273 TST DEVMEM    CHECK DEVICE NUMBER
0242 B275 18 27 33    L8275 LBEQ L95AC    BRANCH IF SCREEN
0243 B279 34 84    L8279 PSHS B    SAVE CHARACTER
0244 B27B 56 6F    L827B DEVMEM    *GET DEVICE NUMBER AND
0245 B27C 01 F0    L827C CMPB #.3    *CHECK FOR DLOAD
0246 B27F 54 84    L827F PULS B    GET CHARACTER BACK
0247 B281 26 32    L8281 BNE LB285    RETURN IF NOT DLOAD
0248 B283 32 62    L8283 LEAS #8.2    **TAKE RETURN OFF STACK & GO BACK TO ROUTINE
0249 B285 3A    L8285 *    **THAT CALLED CONSOLE OUT
0250 B285 3B    L8285 RTS

0252 * CLOSE FILES RAM HOOK - THIS CODE CORRECTS A
0253 * BUG IN 1.8 BASIC WHICH WAS FIXED BY 1.1 BASIC
0254 B286 96 6F    L8286 LDA DEVMEM    GET DEVICE NUMBER
0255 B288 4C    L8288 INCA    CHECK FOR CASSETTE
0256 B289 26 FA    L8289 BNE LB285    RETURN IF NOT CASSETTE
0257 B288 9E 7B    L8288 FLDSTA    GET FILE STATA
0258 B28B 81 82    L828B CMPA #682    OPEN FOR OUTPUT
0259 B28F 26 FA    L828F BNE LB285    RETURN IF NOT OPEN FOR OUTPUT
0260 B291 96 79    L8291 CINCTR    GET CHARACTER BUFFER COUNTER
0261 B293 26 FB    L8293 BNE LB285    RETURN IF NOT EMPTY
0262 B295 8F 6F    L8295 CLR DEVMEM    SET DEVICE NUMBER TO SCREEN
0263 B297 32 62    L8297 LEAS #8.2    GET RETURN ADDRESS OFF OF STACK
0264 B299 7E 44    L8299 JMP LA444    WRITE END OF FILE TAPE BLOCK

0266 * RUN RAM HOOK
0267 B29C 5A 82    L829C LDO $80A2    MID HIGH VALUE + MID LOW VALUE
0268 B29D 6D 82    L829D STD VOLHI    INITIALIZE PLAY MODE
0269 B29F 8C 82    L829F BNE $882    NAME $82
0270 B2A2 97 E2    L82A2 STA TEMP    INITIALIZE TEMP TO 2
0271 B2A5 97 DE    L82A5 STA OCTAVE    INITIALIZE OCTAVE TO 3
0272 B2A7 4B    L82A7 ASLA    X2
0273 B2A8 97 E1    L82A8 STA NOTELN    INITIALIZE NOTE LENGTH TO 5
0274 B2AB 8F 6S    L82AB CLR DOTVAL    CLEAR NOTE TIMER SCALE FACTOR
0275 B2AC DC BA    L82AC LDO ZERO    ZERO ACCO
0276 B2AE 6D 8B    L82AE STD ANGLE    INITIALIZE DRAW ANGLE AND SCALE TO 1
0277 B2B8 C6 BB    L82B8 LDB #12B    * INITIALIZE HORIZONTAL DEFAULT
0278 B2BA 2D CE    L82BA STD HORDIF    * COORDINATE TO MID POSITION
0279 B2BC 66 6B    L82BC LDB #96    = INITIALIZE VERTICAL DEFAULT
0280 B2BD DD C9    L82BD STD VERDEF    = COORDINATE TO MID POSITION

0281 B288 39    L8288 RTS

0282 * COMMAND INTERPRETATION LOOP RAM HOOK
0283 B28B 32 62    L828B LEAS #8.2    PURGE RETURN ADDRESS FROM STACK
0284 B28B 1A AF    L828B ANDCC #4AF    ENABLE INTERRUPTS
0285 B28B 8D 8D 8E    L828B JSR LADER    CHECK FOR KEYBOARD BREAK
0286 B28C 9E 86    L828C LDX CHARAD    * GET CURRENT BASIC LINE
0287 B28C 9F 2F    L828C SET TINPTR    * POINTER AND SAVE IT
0288 B28C A6 88    L828C LDA $88    * GET CURRENT INPUT CHARACTER AND ADVANCE POINTER
0289  B2C7 27 #7  BEQ  L2CF  BRANCH IF END OF LINE
0290  B2CB  B1 3A  CMPA #*1’  CHECK FOR COLUMN
0291  B2CA  27 25  BEQ  L2F1  CONTINUE INTERPRETING IF COLUMN
0292  B2CC  27 77  JMP  L2F7  RETURN ERROR - COLUMN ONLY LEGAL LINE SEPARATOR
0293  B2CF  A6 81  L2CF  LDA ,+++  * GET 1ST BYTE OF ADDRESS OF NEXT
0294  B2D1  97 #A  STA  ENDFLG  * BASIC LINE AND SAVE IT
0295  B2D3  26 #3  BNE  L2DB  BRANCH IF NOT END OF PROGRAM
0296  B2D5  7E AE 15  JMP  LAE15  RETURN TO DIRECT MODE - PRINT OK
0297  B2DB  BE 0B  L2DB  LDD ,A  GET LINE NUMBER OF NEXT LINE
0298  B2DA  D6 68  STD  CURLIN  SAVE LINE NUMBER
0299  B2DC  9F A6  STX  CHARAD  SAVE ADDRESS NEXT BYTE TO INTERPRET
0300  B2DE  96 4F  LDA  TRCFLG  TEST THE TRACE FLAG
0301  B2E7  2F 8F  BEQ  L2F1  BRANCH IF TRAC OFF
0302  B2E2  86 5B  LDA #55B  <LEFT HAND MARKER FOR TRIM LINE NUMBER
0303  B2E4  8A 82  JSR  L2AE  OUTPUT A CHARACTER
0304  B2E7  96 68  LDA  CURLIN  GET MS BYTE OF LINE NUMBER
0305  B2E9  BD 80  JSR  L2BC  CONVERT ACC TO DECIMAL AND PRINT ON SCREEN
0306  B2EC  86 5D  LDA #5D  > RIGHT HAND MARKER FOR TRIM LINE NUMBER
0307  B2EE  BD 82  JSR  L2B2  OUTPUT A CHARACTER
0308  B2F1  90 9F  LBF1  JSR  GETRCM  GET A CHARACTER FROM BASIC
0309  B2F3  1F A9  TFR  CC,B  SAVE STATUS REG BACK
0310  B2F5  B1 98  CMPA #98  CSAVE TOKEN
0311  B2F7  27 10  BEQ  L236  DO A CSAVE
0312  B2F9  B1 97  CMPA #97  CLOAD TOKEN
0313  B2F7  27 14  BEQ  L231  PROCESS CLOAD
0314  B2F2  8F 9A  TFR  CC,B  GET STATUS REG BACK
0315  B2FF  BD 9D  JSR  L2AD  LINK BACK TO BASIC’S INTERPRETATION LOOP
0316  B302  2E 77  JSR  L2BB  GO TO MAIN INTERPRETATION LOOP
0317  * CRUNCH RAM HOOK
0318  B304  AE 62  XVECC3  LDX #R & S  *CHECK TO SEE IF THE ROUTINE CALLING CRUNCH
0319  B306  BC AC 90  CMPA #LACDD  *IS COMING FROM THE MAIN LOOP IN BASIC
0320  B309  26 #5  BNE  L21B  *AND BRANCH IF NOT
0321  B30B  BE BF 1F  LDX #LUFF1  =IF IT IS, DO NOT RETURN TO THE BASIC MAIN LOOP
0322  B30B  AE 62  STX #R & S  =BUT TO THE EXTRAS PATCH INSTEAD
0323  B310  89 8E  L83B  RTS  RETURN
0324  B311  82 BC 62  L831  JSR  L262  CHECK EXTRAS CLOAD HANDLER
0325  B314  28 A5  BRA  L288  GO TO MAIN INTERPRETATION LOOP
0326  B316  82 BD 62  L836  BSR  L21A  DO A CSAVE
0327  B318  28 A1  BRA  L288  GO TO MAIN INTERPRETATION LOOP
0328  B31A  90 9F  L83A  JSR  GETRCM  GET A CHARACTER FROM BASIC
0329  B31C  B1 4D  CMPA #’ W’  CHECK FOR CSAVE
0330  B31E  1E 26 21 2A  BNE  L264  BRANCH IF IT’S NOT CSAVE
0331  * CSAVE
0332  B322  90 9F  JSR  GETRCM  GET A CHARACTER FROM BASIC
0333  B324  BD 65 78  JSR  L2A7  GET NAME OF FILE FROM BASIC
0334  B327  BD 43  BSR  L23C  GO TO THE START ADDRESS
0335  B329  8F #F 7E  STX  CASBUF+13  PUT IT IN HEADER BUFFER
0336  B32C  BD 3E  BSR  L23C  GO TO END ADDRESS
0337  B32E  AC 62  CMPX #R & S  COMPARE TO START ADDRESS
0338  B330  1E 25 31 16  LBCS L24A  RCC ERROR IF START > END
0339  B334  8D 3E  BSR  L23C  GO TO HEADER ADDRESS
0340  B336  BF E5  STX  CASBUF+11  PUT IT IN HEADER BUFFER
0341  B339  8D A5  JSR  GETCM  GET NEW CHARACTER
0342  B33B  26 03  BNE  L21B  RETURN IF NOT END OF LINE.
0343  B33D  82 82  LDA #R & S  FILE TYPE (MACHINE LANGUAGE)
0344  B33F  9E 8A  LDX  ZERO  X = 0000 FILE MODE AND ASCII FLAG
0345  B341  BD 65  JSR  L265  WRITE HEADER BLOCK
0346  B344  FE 7B  CLRFILTA  CLOSE CASS CASSETTE FILES
0347  B346  4C 7C  INC  BLKTP  BLOCK TYPE = 1
0348  B34A  8D 08  JSR  L2AB  GO WRITE LEADER
0349  B34B  AE 64  LDX #R & S  START ADDRESS
0350  B34D  9F 7E  L83D  STX  CBUDR  FOLLOW BUFFER START ADD
0351  B34F  86 FF  LDA #255  BLOCK SIZE = 255
0352  B351  97 70  STA  BLKLEN  STORE IN BLOCK SIZE
0353  B353  EC 62  LDD #R & S  GET END ADDRESS
0354  B355  93 7E  SUBD CBUDR  SUBTRACT START ADDRESS
0355  B357  24 #5  BCC  L235  BRANCH IF MORE TO BE WRITTEN
0356  B359  32 66  LEAS #8, S  CLEAN UP STACK
0357  B35B  4E AE 91  JMP  L291  WRITE FINAL BLOCK
0358  B35E  1B B3 00 FF  L83E  CMPD #M8BFF  AT LEAST 1 FULL BLK LEFT?
0359  B362  2E 3A  BCC  L236  YES
0360  B364  5C  INCX  NO - PUT WHAT’S LEFT IN BLKLEN
0361  B365  D7 7D  STB  BLKLEN  BUFFER SIZE
0362  B367  BD 8F 4F  L837  JST  LADF  WRITE A BLOCK
0363  B36A  28 E1  BRA  L83D  GO DO SOME MORE
0364  B36C  BD 60  L83C  JSR SYMCOM  SYNTAX CHECK FOR COMMA
0365  B36F  BD 3D  JST  L273D  EVAL EXP - RETURN VALUE IN X
0366  B372  EE 4E  LDU ,S  SAVE RETURN ADDRESS IN U
0367  B374  AF 44  STX ,S  PUT THE EXPRESSION ON THE STACK
0368  B376  1F 35  TFR  U,PC  RETURN TO CALLING ADDRESS
0369  B378  * COS
0370  B379  * THE VALUE OF COS(X) IS DETERMINED BY THE TRIG IDENTITY COS(X)=5IN(PI/2+X)
0372  B37B  BE 73 4B  COS  LDX #L83AB  POINT X TO FP CONSTANT (PI/2)
0373  B37B  BD 89 C2  JSR  L2BC  ADD FPAB TO (X)
0374  B37E  7E BF 78  L837  JMP  L2BF8  JUMP TO SIN ROUTINE
0375  * TAN
0376  * THE VALUE OF TAN(X) IS DETERMINED BY THE TRIG IDENTITY TAN(X)=5IN(PI/2+X)
0377  B381  BD 1C 2F  TAN  JSR  L2CF  PACK FPAB AND MOVE IT TO FPAB
0378  B384  8A  BA  CLR  RELFFG  RESET QUADRANT FLAG
0379  B386  BD 66  JSR  L237  CALCULATE SIN OF ARGUMENT
0380  B388  BD 4A  JSR  L24A  POINT X TO FPAB
0381  B38B  BD 35  JSR  L2C5  PACK FPAB AND MOVE IT TO FPAB
0382  B38C  86 #A  JSR  L24B  POINT X TO FPAB
0383  B391  BD 1C 4A  JSR  L2B4  MOVE FPAB TO FPAB
LOG THE TERMS OF THE LATTER EXPRESSION ARE CONSTANTS EXCEPT FOR THE
NOTATION). THEREFORE, THE LOG ROUTINE DETERMINES THE VALUE OF
THE NATURAL OR NAPERIAN LOGARITHM IS CALCULATED USING
TCHEBYSHEV MODIFIED TAYLOR SERIES COEFFICIENTS FOR ARCTANGENT
IF X**2>=1 THEN ATN=X-(X**3)/3+(X**5)/5-(X**7)/7...
IF X**2<1 THEN ATN=PI/2-(X/1-X/3-X/5+X/7-…)

ATN = $\log(x)$

ATN MODIFIED TAYLOR SERIES COEFFICIENTS FOR ARCTANGENT

REVISED:12/26/1999 WALTER K ZYDHEK
ORIGIN:SPECTRAL ASSOC
* ARGUMENT. THE POLYNOMIAL COEFFICIENTS ARE MODIFIED TO REFLECT
* HERE. THE FORMULA USED TO EVALUATE EXPONENTIATION
* IS A**X = (X IN A) = (A)**(FPAB*LN(FPA)) +1, E=2.718281

$B4B9 27 67$ LB4B9 BEX EXP DO A NATURAL EXPONENTIAL IF EXPONENT = 0

$B4B8 40$ TSTA *CHECK VALUE BEING EXPONENTIALIZED

$B4B8 26 03$ LB4B1 $B4E1 * AND BRANCH IF IT IS = 0

$B4B8 6E 3A$ JMP LB4A3 FPAB, IF RAISING ZERO TO A POWER

$B4B9 0B 4A$ LB4A4 $B4A0 * PACK FPAB AND SAVE

$B4B4 35 35$ JSR LB4C5 * IT IN FPAB (ARGUMENT S EXPONENT)

$B4B9 7F 5F$ CLRB ACHB=DFEBEUILRTS SIGN FLAG, IF POSITIVE

$B4B8 96 61$ LDA FPISGN *CHECK THE SIGN OF ARGUMENT

$B4BA 2A 18$ BPL LB4AC *BRANCH IF POSITIVE

$B4B9 0C 0C$ JSR INT CONVERT EXPONENT INTO AN INTEGER

$B4B8 96 61$ LDA FPISGN *POINT X TO FPAB (ORIGINAL EXPONENT)

$B4B4 9E 61$ LDA FPISGN *GET MANTISSA SIGN OF FPAB (ARGUMENT)

$B4B4 0D 4B$ JSR LB4CA *COMPARE FPAB TO X AND

$B4B4 27 03$ BNE LB4AC *BRANCH IF NOT EQUAL

$B4B8 43 43$ TSBX #POSITIVE TOGGLE FPAB MANTISSA SIGN - FORCE POSITIVE

$B4B4 0D 61$ LDB CHARAC GET MS BYTE OF INTEGER VALUE OF EXPONENT (RESULT SIGN FLAG)

$B4B4 2C 4C$ LB4AC LB4AC COPY FPAB TO FPAC, ACCA = MANTISSA SIGN

$B4B8 34 4B$ PHSB S PUT RESULT SIGN FLAG ON THE STACK

$B4B8 8B 46$ JSR LOG GET NATURAL LOGARITHM OF FPAB

$B4B8 0B 4A$ LDA #AA 4A *POINT X TO FPAB

$B4B8 4B 8A$ JSR LB4CA MULTIPLY FPAB BY FPAB

$B4B8 36 36$ BSR EXP CALCULATE **(FPAB)

$B4B8 35 32$ PULS A *GET RESULT SIGN FLAG FROM THE STACK

$B4B8 4F$ RORA * AND BRANCH IF NEGATIVE

$B4B8 1B 25 3A 2B$ LBC FBECE9 CHANGE SIGN OF FPAB MANTISSA

$B4B4 39$ RTS

$B4B4 81$ * CORRECTION FACTOR FOR EXPONENTIAL FUNCTION

$B4B4 B1 A3 3B 29$ LB4C4 FCB $B481, $B48A, $B48B, $B48D, $B48F

$B4B4 2B$ 1.44267954 ( CF )

$B4B4 80$ *

$B4B4 38$ * THECHBESHEV MODIFIED TAYLOR SERIES COEFFICIENTS FOR E**X

$B4B4 32$ B8C9 #7$ LB8C9 FCB $87 EIGHT COEFFICIENTS

$B4B4 33$ B8CA 71 34 58 3E 56$ LB8CA FCB $B71, $B34, $B58, $B3E, $B56 EIGHT COEFFICIENTS

$B4B4 34$ B8CF 1A 1E 7E 1B 38$ LB8CF FCB $B1A, $B1E, $B7E, $B1B, $B38 EIGHT COEFFICIENTS

$B4B4 35$ B8D4 77 2F EE E3 86$ LB8D4 FCB $B77, $B2F, $EE, $E3, $B86 EIGHT COEFFICIENTS

$B4B4 36$ B8D9 7A 1A 84 1A 2A$ LB8D9 FCB $B7A, $B1A, $B84, $B1A, $B2A EIGHT COEFFICIENTS

$B4B4 37$ B8DF 6C 63 89 55 80$ LB8DF FCB $B6C, $B63, $B89, $B55, $B80 EIGHT COEFFICIENTS

$B4B4 38$ B8EB 7C 79 75 8D$ LB8EB FCB $B7C, $B79, $B75, $B8D EIGHT COEFFICIENTS

$B4B4 39$ B8EC 31 72 1B 1B$ LB8EC FCB $B31, $B72, $B1B, $B1B EIGHT COEFFICIENTS

$B4B4 3A$ B8ED FCB $B81, $B80, $B80, $B80, $B80 1.

$B4B4 41$ *

$B4B4 42$ * EXP ( E**X )

$B4B4 43$ * THE EXPONENTIAL FUNCTION IS EVALUATED BY FIRST MULTIPLYING THE

$B4B4 44$ * ARGUMENT BY A CORRECTION FACTOR (CF). AFTER THIS IS DONE, AN

$B4B4 45$ * ARGUMENT >> 127 WILL YIELD A ZERO RESULT (NO UNDERFLOW) FOR A

$B4B4 46$ * NEGATIVE ARGUMENT OR AN "0V" (OVERFLOW) ERROR FOR A POSITIVE

$B4B4 47$ * ARGUMENT, THE POLYNOMIAL COEFFICIENTS ARE MODIFIED TO REFLECT

$B4B4 48$ * THE CF MULTIPLICATION AT THE START OF THE EVALUATION PROCESS.

$B4B4 49$ *

$B4B4 58$ BAF2 BE 04$ EXP LDX HLBA4C POINT X TO THE CORRECTION FACTOR

$B4B4 59$ BAF7 BD BA$ JSR LB4CA MULTIPLY FPAB BY (X)

$B4B4 5A$ BAF2 BD BC$ JSR LB4CF PACK FPAB AND STORE IT IN FPAB

$B4B4 5B$ BAF9 BD 4F$ LDA FP6EFP *GET EXPONENT OF FPAB AND

$B4B4 5C$ BAFD 81 8B$ CMPA #80B *COMPARE TO THE MAXIMUM VALUE

$B4B4 5D$ BAF5 25 23$ BLD LB5B4 BRANCH IF FPAB = 128

$B4B4 5E$ B501 7E 85 SC$ LB501 JMP LB50C SET FPAB = 0 OR 0V ERROR

$B4B4 5F$ B504 BD BE$ IN LB504 CONVERT FPAB TO INTEGER

$B4B4 58$ B50B 96 61$ LDA CHARAC GET LS BYTE OF INTEGER

$B4B4 59$ B50B 0B 61$ ADDA #80B * WAS THE ARGUMENT -127, IF 0510

$B4B4 5B$ B50B 27 F4$ BEQ LB5B1 * THEN 0V ERROR; THIS WILL ALSO ADD THE $B81 BASE

$B4B4 5C$ +

$B4B4 5D$ * REQUIRED WHEN THE NEW EXPONENT IS CALCULATED BELOW

$B4B4 5E$ * DECREMENT ONE FROM THE EXPONENT, BECAUSE $B81, NOT $B82 WAS USED ABOVE

$B4B4 5F$ * POINT TO X FOR FPAB

$B4B4 60$ B513 BD 89$ SUBTRACT FPAB FROM (X) - GET FRACTIONAL PART OF ARGUMENT

$B4B4 61$ B516 BE 84$ LDX HLBA4C POINT X TO COEFFICIENTS

$B4B4 62$ B516 BD FF$ JSR LB4EF EVALUATE POLYNOMIAL FOR FRACTIONAL PART

$B4B4 63$ B51C 0F 62$ CLR RESO FORCE THE MANTISSA TO BE POSITIVE

$B4B4 64$ B51E 35 22$ PULS A GET INTEGER EXPONENT FROM STACK

$B4B4 65$ B520 BD 8B 8B$ JSR LB4AB *CALCULATE EXPONENT OF NEW FPAB BY ADDING THE EXPONENTS OF THE

$B4B4 66$ * Integer and Fractional Parts

$B4B4 67$ B523 39$ RTS

$B4B4 74$ * FIX

$B4B4 75$ B524 BD BC 6D$ FIX JSR LB65C0 CHECK STATUS OF FPAB

$B4B4 76$ B527 2B 03$ BML LB52C BRANCH IF FPAB = NEGATIVE
Extended Basic Unraveled II Appendix B

Extended Basic Unraveled II Appendix B

0577 8529 TE BC EE L8529 JMP INT CONVERT FPAR TO INTEGER
0578 852C #3 54 L852C COM FPARGW TOGGLE SIGN OF FPAR MANITISA
0579 852E BD F9 BSR L8529 CONVERT FPAR TO INTEGER
0580 8538 7E BE E9 JMP L85EE TOGGLE SIGN OF FPAR
0581
0582 * EDIT
0583 8533 BD 89 AE EDIT JSR L85AD GET LINE NUMBER FROM BASIC
0584 8536 32 62 L8536 LEAS $R2,$S PURGE RETURN ADDRESS OFF OF THE STACK
0585 8538 86 81 L8538 LDA #M&81 LIST FLAG
0586 853A 97 08 STA V0B SET FLAG TO LIST LINE
0587 853C BD AD 81 JSR L85B1 GO FIND THE LINE NUMBER IN PROGRAM
0588 853F 18 25 9F L853F LSR #25 AF ERROR #7 UNDEFINED LINE #7
0589 8543 BD B7 C2 JSR L85C7 GO UNGRACE LINE INTO BUFFER AT LINBUF+1
0590 8546 1F 28 TFR Y,D PUT ABSOLUTE ADDRESS OF END OF LINE TO ACCD
0591 8548 83 82 0E SUBTACT OUT THE START OF LINE
0592 854D 07 07 STB V07 SAVE LENGTH OF LINE
0593 854D 0C 2B L854D LDO BINVAL GET THE HEX VALUE OF LINE NUMBER
0594 854F BD CC JSR L85CC LIST THE LINE NUMBER ON THE SCREEN
0595 8552 BD 89 AC JSR L85AC PRINT A SPACE
0596 8556 8E 82 DD LDX LINBUF+1 PRINT X TO BUFFER
0597 8558 0D 08 LDB V0B * CHECK TO SEE IF LINE IS TO BE
0598 855A 26 25 BNE L85BD * LISTED TO SCREEN - BRANCH IF IT IS
0599 855C 8F CLBE L85CE RESET DIGIT ACCUMULATOR - DEFAULT VALUE
05A0 855D BD 86 87 L855D JSR L8567 GET KEY STROKE
05A1 8568 BD 98 AA JSR L99A SET CARRY IF NOT NUMERIC
05A2 856D 25 80 BLO L8578 BRANCH IF NOT NUMERIC
05A3 856F 3F 8F SUBA #M'8F MASK OFF ASCII
05A4 8574 24 82 PHSX SAVE IT ON STACK
05A5 8579 86 8A LDA #M10 NUMBER BEING CONVERTED IS BASE 10
05A6 857B 3D MUL ADD DIGIT TO ACCUMULATED
05A7 857C 0E 08 ADSE ,+$* ADD DIGIT TO ACCUMULATED VALUE
05A8 857E 26 8D BRA L8550 CHECK FOR THE NEXT DIGIT
05A9 857F 0D 18 L857F SUBB #M'81 * REPEAT PARAMETER IN ACBR; IF IT
05AA 8582 09 81 ADCB #M'81 * IS #, THEN MAKE IT 1
05AB 8574 81 41 CMPA #M'41 ABORT?
05AC 8576 26 85 BNE L8570 NO
05AD 857B 89 3B J85L95B PRINT CARRIAGE RETURN TO SCREEN
05AE 857D 0D 88 BRA L8531 RESTART EDIT PROCESS - CANCEL ALL CHANGES
05AF 857D 0C 81 CLBE L8571 LIST # M1
05B0 857E 26 8B BNE L855C NO
05B1 857F 0D 81 L857F BSR L8584 LIST THE LINE
05B2 8583 0F 08 CLR VD RESET THE LIST FLAG TO NO LIST
05B3 8585 BD 89 8B J8595B PRINT CARRIAGE RETURN
05B4 8588 28 03 L8588 LSR #28 C3 GO INTERPRET ANOTHER EDIT COMMAND
05B5 858A 32 62 L858A LEAS #2$2 PURGE RETURN ADDRESS OFF OF THE STACK
05B6 858B 81 0D L858B CMPA #CR ENTER KEY?
05B7 858C BD 86 8D BNE L85BD NO
05B8 858F BD 0B 22 BSR L8584 ECHO THE LINE TO THE SCREEN
05B9 8592 BD 89 8B L8592 J8595B PRINT CARRIAGE RETURN
05BA 8595 0D 30 L8595 LSR LINBUF+1 * RESET BASIC 5 INPUT POINTERS
05BB 8597 9F 46 STX CHARAD * TO THE LINE INPUT BUFFER
05BC 859A 7E AC A8 JMP LACAB GO PUT LINE BACK IN PROGRAM
05BD 859B 01 46 L859B CMPA #E' EXIT
05BE 859F 27 F1 BEX L8592 YES - SAME AS ENTER EXCEPT NO ECHO
05C0 85A1 81 51 CMPA #Q' QUIT
05C1 85A3 26 86 BNE L85A6 NO
05C2 85A5 BD 89 8B J8595B PRINT CARRIAGE RETURN TO SCREEN
05C3 85A6 28 7C L85A6 JSR #7AEC GO TO COMMAND LEVEL - MAKE NO CHANGES
05C4 85A8 BD 89 82 L85A8 BSR L856F INTERPRET THE REMAINING COMMANDS AS SUBROUTINES
05C5 85BB 28 00 L85BB BSR L855C GO INTERPRET ANOTHER EDIT COMMAND
05C6 85B7 81 2B L85B7 CMPA #SPACE SPACE BAR?
05C7 85B8 26 08 BNE L85BC NO
05C8 85B8 8C L85B8 FCB SKP2 SKIP TWO BYTES
05C9 85C0 1F * DISPLAY THE NEXT ACBR BYTES OF THE LINE IN THE BUFFER TO THE SCREEN
05CA 85C1 8F 08 *
05CB 85C2 06 69 L85C2 LDB #BLBUF+1 250 BYTES MAX IN BUFFER
05CD 85C6 A6 B4 L85C6 LDA ,X GET A CHARACTER FROM BUFFER
05CE 85C7 28 0A BEQ L85C2 EXIT IF IT'S A @
05CF 85C8 BD A2 82 JSR L82Z8 SEND CHAR TO CONSOLE OUT
05D0 85C9 3F 81 LEAX $R1,X MOVE POINTER UP ONE
05D1 85C2 4A 8A DBCB DECDEC DECREMENT CHARACTER COUNTER
05D2 85C8 26 64 BNE L8556 LOOP IF NOT DONE
05D3 85C9 23 L85C9 RTS
05D4 85C3 81 44 L85C3 CMPA #D' DELETE?
05D5 85C5 26 48 BNE L854F NO
05D6 85C7 6B 84 L85C7 TST ,X * CHECK FOR END OF LINE
05D7 85C9 27 F7 BEQ L85C2 * AND BRANCH IF SO
05D8 85CA 0D 04 BSR L8501 REMOVE A CHARACTER
05D9 85CC 5A DBCB DECREMENT REPEAT PARAMETER
05DA 85CE 26 6F BNE L85C7 BRANCH IF NOT DONE
05DB 85D3 39 RTS
05DC 85D8 81 49 L85DE CMPA #I' INSERT?
05DD 85E7 23 13 BEQ L85F5 YES
05DE 85E2 81 58 BSR L853F EXTEND!
05DF 85E4 27 8D BNE L8593 YES
05E0 85E6 81 48 CMPA #H' HACK?
05E1 85E2 26 5C DBCB DECDEC DECREMENT CURRENT BUFFER POINTER INTO END OF LINE FLAG
05E2 85EA 6F 84 CLR ,X
0673 85EC IF 1B TFR X,D  PUT CURRENT BUFFER POINTER IN ACCD
0674 85EE B3 @2 DE SUB @L#BUF+2  SUBTRACT INITIAL POINTER POSITION
0675 85F1 D7 D7 STB VD7  SAVE NEW BUFFER LENGTH
0676 85F3 B0 8F L85F3 BS R8L8B4  DISPLAY THE LINE ON THE SCREEN
0677 85F5 BD B6 L85F5 JSR R8L8B7  GET A KEYSTROKE
0678 85FB B1 @0 CMPA #CR  ENTER KEY?
0679 85FA 27 BE BEO R8L8B8  YES - INTERPRET ANOTHER COMMAND - PRINT LINE
0680 85FC B1 1B CMPA #ESC  ESCAPE?
0681 85F5 27 E6 R8L825  YES - RETURN TO COMMAND LEVEL - DON'T PRINT LINE
0682 8600 B1 #B CMPA #HS  BACK SPACE?
0683 8602 26 22 BNE L8626  NO
0684 8604 B0 2C @2 DO CMPX #L#BUF+1  COMPARE POINTER TO START OF BUFFER
0685 8607 EC BE JSR L85F5  DO NOT ALLOW BS IF AT START
0686 8609 BD 45 BSR L8658  MOVE POINTER BACK ONE, BS TO SCREEN
0687 860B BD C4 BSR L85D1  REMOVE ONE CHARACTER FROM BUFFER
0688 860D 2B @6 ERA L85F5  GET INSERT SUB COMMAND
0689 860F B1 43 L860F CMPA #C'  CHANGE?
0690 8611 26 @B BNE L85E6  NO
0691 8613 D0 B4 L8613 TST,X  CHECK CURRENT BUFFER CHARACTER
0692 8615 27 8C BEQ L8525  BRANCH IF END OF LINE
0693 8617 BD B6 B7 JSR L8587  GET A KEYSTROKE
0694 861A 25 @2 BLO L861E  BRANCH IF LEGITIMATE KEY
0695 861C F2 F5 BRA L8613  TRY AGAIN IF ILLEGAL KEY
0696 861E A7 @8 STA,X,++ INSERT NEW CHARACTER INTO BUFFER
0697 8620 BD 37 BSR L8659  SEND NEW CHARACTER TO SCREEN
0698 8622 5A DECB  DECIMATE REPEAT PARAMETER
0699 8623 2E EE BNE L8613  BRANCH IF NOT DONE
0700 8625 39 RT8 L8625  RTS
0701 8626 D6 @7 L8626 LDV VD7  GET LENGTH OF LINE
0702 8628 C1 F9 CMPF #LBUF+1  COMPARE TO MAXIMUM LENGTH
0703 862A 26 @2 BNE L862E  BRANCH IF NOT AT MAXIMUM
0704 862C 2B @C BRA L85F5  IGNORE INPUT IF LINE AT MAXIMUM LENGTH
0705 8632 34 1B L862E PSHS X  SAVE CURRENT BUFFER POINTER
0706 8634 60 @H L8638  * SCAN THE LINE UNTIL END OF
0707 8632 2F FC BNE L8638  * LINE (Q) IS FOUND
0708 8642 E6 82 L8634  DEC TEMP LINE POINTER AND GET A CHARACTER
0709 8636 E7 @1 STB #R1,X  PUT CHARACTER BACK DOWN ONE SPOT
0710 8638 AC FE CMPS,X  HAVE WE REACHED STARTING POINT?
0711 863A 25 @F BLO L8634  NO - KEEP GOING
0712 863C 32 62 LEAS #R2,S  PURGE BUFFER POINTER FROM STACK
0713 863E A7 @B STA,X++,  INSERT NEW CHARACTER INTO THE LINE
0714 8642 BD 17 BSR L8659  SEND A CHARACTE TO CONSOLE OUT
0715 8642 D0 @7 INC VD7  ADD ONE TO BUFFER LENGTH
0716 8644 2B @A BRA L85F5  GET INSERT SUB COMMAND
0717 8646 B1 #B L8646 CMPA #HS  BACK SPACE?
0718 8648 26 12 BNE L865C  NO
0719 864A BD 84 L8644 BSR L85D8  SEND POINTER BACK 1, SEND BS TO SCREEN
0720 864C 5A DECB  DECIMATE REPEAT PARAMETER
0721 864D 26 FB BNE L864A  LOOP UNTIL DONE
0722 864F 39 RT8
0723 8658 BC @2 DO L8658 CMPS #L#BUF+1  COMPARE POINTER TO START OF BUFFER
0724 8653 2B 08 BEQ L8525  DO NOT ALLOW BS IF AT START
0725 8655 38 IF L865C  MOVE POINTER BACK ONE
0726 8657 B0 #B LDA #HS  BACK SPACE
0727 8659 7E A2 @2 L8659 JMP L82B2  SEND TO CONSOLE OUT
0728 865C B1 4B L865C CMPA #K'  KILL?
0729 865E 27 #5 BEO L8665  YES
0730 865E 8A #3 SUBA #S'  SEARCH!
0731 8662 27 #1 BEO L8665  YES
0732 8664 39 RT8
0733 8665 34 @2 L8665 PSHS A  SAVE KILL/SEARCH FLAG ON STACK
0734 8667 BD 1E BSR L8687  * GET A KEYSTROKE (TARGET CHARACTER)
0735 8669 34 @2 PSHS A  * AND SAVE IT ON STACK
0736 866B A6 84 L8668 LDA,X = GET CURRENT BUFFER CHARACTER
0737 866D 27 6E BEQ L8685  = AND RETURN IF END OF LINE
0738 866F 60 61 TST #R1,S  CHECK KILL/SEARCH FLAG
0739 8671 26 @6 BNE L8679  BRANCH IF KILL
0740 8673 BD 0E BSR L8659  SEND A CHARACTER TO CONSOLE OUT
0741 8675 38 #1 LEX #R1,X  INCREMENT BUFFER POINTER
0742 8677 2B #3 BRA L867C  CHECK NEXT INPUT CHARACTER
0743 8679 BD 85 @0 L8679 JSR L85D1  REMOVE ONE CHARACTER FROM BUFFER
0744 867C A6 @B L867C LDA,X = GET CURRENT INPUT CHARACTER
0745 867E A1 E4 CMPA,#S  COMPARE TO TARGET CHARACTER
0746 8680 26 E9 BNE L8668  BRANCH IF NO MATCH
0747 8682 5A DECB  DECIMATE REPEAT PARAMETER
0748 8683 26 @6 BNE L8668  BRANCH IF NOT DONE
0749 8685 35 @A L8685 PSHS Y,PC  THE Y PULL WILL CLEAN UP THE STACK FOR THE 2 PSHS A
0750 8688 #  *
0751 8688 # GET A KEYSTROKE
0752 8687 BD A1 71 L8687 JSR LA171  CALL CONSOLE IN : DEV MBHI=SCREEN
0753 868A B1 7F CMPA #57F  GRAPHIC CHARACTER?
0754 868C 24 F9 BEC L86B7  YES - GET ANOTHER CHAR
0755 868E B1 5F CMPA #55F  SHIFT UP ARROW (QUIT INSERT)
0756 868F 26 @2 BSR L8694  NO
0757 8691 B2 18 LDA #ESC  REPLACE W/ESC CODE
0758 8694 B1 #D L8694 CMPA #CR  ENTER KEY
0759 8696 27 BE BEO L86A6  YES
0760 8698 B1 1B CMPA #ESC  ESCAPE?
0761 869A 27 BA BEO L86A6  YES
0762 869C B1 #B CMPA #HS  BACK SPACE?
0763 869E 27 @6 BEO L86A6  YES
0764 86A0 B1 28 CMPA #SPACE  SPACE
0765 86A2 25 E3 BLO L86E7  GET ANOTHER CHAR IF CONTROL CHAR
0766 86A4 1A #1 ORCA #91  SET CARRY
0767 86A6 39 RT8 L8668
0768
Disassembly of Extended Basic 1.1

0769 * TRON
0770 B6A7 B6 TRON FB CSDK10 SKIP ONE BYTE AND LDA #4F
0771
0772 * TRON
0773 B6A6 4F TROFF CLRA TROFF FLAG
0774 B6A9 97 AF STA TRCFLG TRON/TROFF FLAG = +TRON , -TROFF
0775 B6B5 39 RTS
0776
0777 * POS
0778 B6A6 96 6F POS LDA DEVNUM GET DEVICE NUMBER
0779 B6A3 24 B2 PSHS A SAVE IT ON STACK
0780 B6B8 B0 A5 AE JSR LASAE GET DEVICE NUMBER
0781 B6B3 BD A6 06 JSR LA486 FILE STATUS CHECK
0782 B6B6 BD A3 5F JSR LASF SET UP TAB FIELD WIDTH
0783 B6B9 D6 6C GET PRINT POSITION
0784 B6BB 7E A5 E4 JMP LASE4 CONVERT PRINT POSITION TO FLOATING POINT
0785
0786 * VARPTR
0787 B6BE BD B2 6A VARPTR JSR LB26A SYNTAX CHECK FOR ( 
0788 B6C1 DC 1F LDD ANYEND GET ADDR OF END OF ARRAYS
0789 B6C3 34 06 PSHS B,A SAVE IT ON STACK
0790 B6C5 BD B3 57 JSR LB357 GET VARIABLE DESCRIPTOR
0791 B6C8 BD B2 67 JSR LB267 SYNTAX CHECK FOR )
0792 B6CB 35 06 PULS A,B GET END OF ARRAYS ADDR BACK
0793 B6CD 1E 1B EXG A,D SWAP END OF ARRAYS AND VARIABLE DESCRIPTOR
0794 B6CF 9C 1F CMPX ANYEND COMPARE TO NEW END OF ARRAYS
0795 B6D1 26 51 BNE LB724 FC ERROR IF VARIABLE WAS NOT DEFINED PRIOR TO CALLING VARPTR
0796 B6D3 7E B4 F4 JMP GIVARP CONVERT VARIABLE DESCRIPTOR INTO A #F NUMBER
0797
0798 * MIDS(OLDSTRING,POSITION,LENGTH,=REPLACEMENT=""
0799 B6D6 9D 9F LBS0 DJSR GETCHR GET INPUT CHAR FROM BASIC
0800 B6D8 BD B2 6A JSR LB26A SYNTAX CHECK FOR ( 
0801 B6D9 BD B3 57 JSR LB357 * GET VARIABLE DESCRIPTOR ADDRESS AND
0802 B6D4 34 18 PKS X SAVE IT ON THE STACK
0803 B6DB EC 02 LDD $82,X POINT ACCA TO START OF OLDSTRING
0804 B6E2 18 93 21 CMPX FREETMP COMPARE TO START OF CLEARED SPACE
0805 B6E5 23 B4 BLS LBS68 BRANCH IF <$
0806 B6E7 93 27 SUBS MEMS12 SUBTRACT TOP OF CLEARED SPACE
0807 B6E9 23 12 LDS LBS6F BRANCH IF STRING IN STRING SPACE
0808 B6EB E6 04 LBS6B LDB,X GET LENGTH OF OLDSTRING
0809 B6E9 BD B5 6D JSR LB56D Reserve ACCB bytes in STRING SPACE
0810 B6E8 34 18 PKS X SAVE RESERVED SPACE STRING ADDRESS ON STACK
0811 B6F2 AE 62 LDX $82,U POINT X TO OLDSTRING DESCRIPTOR
0812 B6F4 BD B6 43 JSR LB643 MOVE OLDSTRING INTO STRING SPACE
0813 B6F7 35 50 PULS X,U * GET OLDSTRING DESCRIPTOR ADDRESS AND RESERVED STRING
0814 B6FA AF 42 STX $82,U * ADDRESS AND SAVE RESERVED ADDRESS AS OLDADDRESS STRING
0815 B6F7 34 48 PKS U SAVE OLDSTRING DESCRIPTOR ADDRESS
0816 B6FD BD B7 3B LB6FD JSR LB738 SYNTAX CHECK FOR COMMA AND EVALUATE LENGTH EXPRESSION
0817 B700 34 04 LBS0 PSHS B SAVE POSITION PARAMETER ON STACK
0818 B702 5D TSB * CHECK POSITION PARAMETER AND BRANCH
0819 B703 27 1F BEQ LB724 * IF START OF STRING
0820 B706 5C FF LDB #4FF DEFAULT REPLACEMENT LENGTH = $FF
0821 B707 BD BD 29 CKP X,* * CHECK FOR END OF MIDS STATEMENT AND
0822 B709 27 03 BEQ LB7BE * BRANCH IF AT END OF STATEMENT
0823 B70B BD B7 3B JSR LB738 SYNTAX CHECK FOR COMMA AND EVALUATE LENGTH EXPRESSION
0824 B70E 34 04 LBS0 PSHS B SAVE LENGTH PARAMETER ON STACK
0825 B710 BD B7 67 JSR LB67 SYNTAX CHECK FOR )
0826 B713 CE 03 LDB #$E3 TOKEN FOR =
0827 B715 BD B2 6F JSR LB26F SYNTAX CHECK FOR =
0828 B716 BD E2 1E BNE LB748 EVALUATE REPLACEMENT STRING
0829 B71A 1F 13 TFR X,U SAVE REPLACEMENT STRING ADDRESS IN U
0830 B71C AE 62 LDX $82,U POINT X TO OLDSTRING DESCRIPTOR ADDRESS
0831 B71E 46 04 LDA,X GET LENGTH OF OLDSTRING
0832 B720 BD B6 1F SUBA $1F,S SUBTRACT POSITION PARAMETER
0833 B722 24 03 BCC LB727 INSERT REPLACEMENT STRING INTO OLDSTRING
0834 B724 7E B4 4A LB724 LDB44A FC ERROR IF POSITION > LENGTH OF OLDSTRING
0835 B727 4C LBS72 INCA * NOW ACCA = NUMBER OF CHARACTERS TO THE RIGHT
0836 * * * (INCLUSIVE) OF THE POSITION PARAMETER
0837 B728 BD B1 44 CMPA ,S COMPARE TO LENGTH PARAMETER
0838 B731 BD B2 42 BCC LB72E BRANCH IF NEW STRING WILL FIT IN OLDSTRING
0839 B732 2C 04 STA ,S IF NOT, USE AS MUCH OF LENGTH PARAMETER AS WILL FIT
0840 B740 A2 61 LB72E LDA $1F,S GET POSITION PARAMETER
0841 B743 8E 09 EXG A,B ACCA=LENGTH OF REPL STRING, ACCA=POSITION PARAMETER
0842 B745 AE 62 LDX $82,X POINT X TO OLDSTRING ADDRESS
0843 B747 5A DECB * BASIC S POSITION PARAMETER STARTS AT 1; THIS ROUTINE
0844 * * WANTS IT TO START AT ZERO
0845 B745 3A ABX * POINT X TO POSITION IN OLDSTRING WHERE THE REPLACEMENT WILL GO
0846 B746 30 40 TSTA * IF THE LENGTH OF THE REPLACEMENT STRING IS ZERO
0847 B747 27 B0 BEQ LB746 * THEN RETURN
0848 B749 BD A1 44 CMPA ,S = IF THE LENGTH OF REPLACEMENT STRING IS < = THE
0849 B74C 23 02 BLS LB73F ADJUSTED LENGTH PARAMETER, THEN BRANCH
0850 B750 26 46 LDA ,S OTHERWISE USE AS MUCH ROOM AS IS AVAILABLE
0851 B753 1F 09 LB73F TFR A,B SAVE NUMBER OF BYTES TO MOVE IN ACCB
0852 B755 1E 31 EXG U,X SNAP SOURCE AND DESTINATION POINTERS
0853 B757 BD B5 9A LDBA9A MOVE (B) BYTES FROM (X) TO (U)
0854 B75E 35 96 LB746 PULS A,B,X,PC CLEAN UP THE STACK AND RETURN
0855 B765 BD B1 56 LB746 JSR LB156 EVALUATE EXPRESSION
0856 B768 7E B6 54 JMP LB654 * TM ERROR IF NUMERIC; RETURN WITH X POINTING
0857 * * **TO STRING, ACCB = LENGTH
0858
0859 * STRING
0860 B74E BD B2 6A STRING JSR LB26A SYNTAX CHECK FOR ( 
0861 B750 BD BD 08 JSR EVALENB EVALUATE EXPRESSION; ERROR IF > 255
0862 B754 34 04 PSHS B SAVE LENGTH OF STRING
0863 B756 BD B2 6D JSR SYNCOMMA SYNTAX CHECK FOR COMMA
0864 B759 BD B1 56 JSR LB156 EVALUATE EXPRESSION

89
* INSTR

#B7E BD B6 6A * INSTR JSR #B26A SYNTAX CHECK FOR ( )

#8B7 BD B1 B1 56 JSR #B156 EVALUATE EXPRESSION

#8B7 B6 61 LDA #$B1 DEFAULT POSITION = 1 (SEARCH START)

#8B7 34 34 PHSH $ B SAVE START

#8B8 96 96 LDA VALTYP GET VARIABLE TYPE

#8B8 26 18 BNE #B79C BRANCH IF STRING

#8B8 BD B7 9E JSR #B7S0 CONVERT FPA2 INTO INTEGER IN ACCB

#8B8 EF 64 STB, , , , SAVE START SEARCH VALUE

#8B9 21 91 BRA #B72A BRANCH IF START SEARCH AT ZERO

#8B9 BD B2 BD JSR #B654 SYNTAX CHECK FOR STRING

#8B9 BD B1 56 JSR #B156 EVALUATE EXPRESSION - SEARCH STRING

#8B9 BD B6 61 JSR #B654 SYNTAX CHECK FOR COMMA

#8B9 BD B1 56 JSR #B156 EVALUATE EXPRESSION - SEARCH STRING

#8B9 BD B1 46 JSR #B146 TM ERROR IF NUMERIC

#8B9 9E 52 L87B0 LOCAL FPAM+2 SEARCH STRING DESCRIPTOR ADDRESS

#8B9 34 16 PHSH X SAVE ON THE STACK

#8B9 BD B2 BD JSR #B654 SYNTAX CHECK FOR COMMA

#8B9 BD B7 48 JSR #B748 EVALUATE TARGET STRING EXPRESSION

#8B9 BD 34 14 SAVE ADDRESS AND LENGTH ON STACK

#8B9 BD B2 BD JSR #B654 SYNTAX CHECK FOR ( ' )

#8B9 BD AE 63 * LOAD X WITH SEARCH STRING DESCRIPTOR ADDRESS

#8B9 BD B6 59 JSR #B659 * AND GET THE LENGTH AND ADDRESS OF SEARCH STRING

#8B9 BD 34 84 PHSH $ B SAVE LENGTH ON STACK

* * *

* AT THIS POINT THE STACK HAS THE FOLLOWING INFORMATION

* ON IT: 8,5-SEARCH LENGTH; 1,5-TARGET LENGTH; 2,5-SEARCH POSITION

* ADDRESS: 4,5-SEARCH DESCRIPITOR ADDRESS; H,5-SEARCH POSITION

#8B7 BD 61 CMPB #$05,5 COMPARNE LENGTH OF STRING TO START

#8B7 BD 7C BD RTS BD +3 POSITION; RETURN @ IF LENGTH = START

#8B7 BD 66 LDA #$05,5 GET LENGTH OF TARGET STRING

#8B7 BD 2C BEQ #B706 BRANCH IF TARGET STRING = NULL

#8B7 BD 66 LDA #$05,5 GET START POSITION

#8B7 BD 5A DECB MOVE BACK ONE

#8B7 BD 3A ABX POINT X TO Position IN SEARCH STRING WHERE SEARCHING WILL START

#8B7 BD B4 L87BE LDA # X POINT Y TO SEARCH POSITION

#8B7 BD EE 62 LDU #$05,5 POINT U TO START OF TARGET

#8B7 BD EC 61 LDA #$05,5 LOAD ACCB WITH LENGTH OF TARGET

#8B7 BD AE 64 LDA #$05,5 LOAD ACCB WITH LENGTH OF SEARCH

#8B7 BD B0 66 SUBA #$05,5 SUBTRACT SEARCH POSITION FROM SEARCH LENGTH

#8B7 BD 4C INC A ADD ONE

#8B7 BD 91 CMPA #$05,5 COMPARE TO TARGET LENGTH

#8B7 BD 5C BLX #B70D RETURN IF TARGET LENGTH > WHAT 5 LEFT OF SEARCH STRING

#8B7 BD A6 68 L87CD LDA # X* GET A CHARACTER FROM SEARCH STRING

#8B7 BD 4E 61 CMPA ,U+ COMPARE IT TO TARGET STRING

#8B7 BD 24 61 BEQ #B70F BRANCH IF NO MATCH

#8B7 BD 5A DECB DECREMENT TARGET LENGTH

#8B7 BD 04 67 BNE #B7CD CHECK ANOTHER CHARACTER

#8B7 BD B6 66 L87CD LDB #$05,5 GET MATCH POSITION

#8B7 BD 21 L87BD FCB SKP SKIP NEXT BYTE

#8B7 BD 0F 5F L87BD CLR Branch ADDRESS = 0

#8B7 BD 32 67 LEAS #$05,5 CLEAN UP THE STACK

#8B7 BD 7E B4 F3 JMP #B4F3 CONVERT ACCB TO FP NUMBER

#8B7 BD 6F 66 L87F0 INC #$05,5 INCREMENT SEARCH POSITION

#8B7 BD 21 LEAS #$01,Y MOVE X TO NEXT SEARCH POSITION

#8B7 BD D9 BRA #B7BE KEEP LOOKING FOR A MATCH

* * *

* ASCII TO FLOATING POINT CONVERSION RAM HOOK

#8B7 BD 26 XVEC19 CMPA #04,7 * RETURN IF NOT HEX OR OCTAL VARIABLE

#8B7 BD 32 62 LEAS #$05,5 PURGE RETURN ADDRESS FROM STACK

* PROCESS A VARIABLE PRECEDED BY A & (AN, AD)

#8B7 BD 52 L87B8 CLR FPAM+2 * CLEAR BOTTOM TWO

#8B7 BD F3 CLR FPAM+1 * BYTES OF FPAM

#8B7 BD 5E BD 52 LDX #$05+2 BYTES 1,3 OF FPAM = (TEMPORARY ACCUMULATOR)

#8B7 BD 9F 9F JSR GETNS3 GET A CHARACTER FROM BASIC

#8B7 BD 4F CMPA #04,7 OCTAL VALUE

#8B7 BD 76 27 BEQ #B79B YES

#8B7 BD B1 48 JSR #B148 HEX VALUE?

#8B7 BD F3 27 BEQ #B79B YES

#8B7 BD 9D 90 JSR GETCN3 GET CURRENT IPUT CHARACTER

#8B7 BD 2B 9C BRA #B7BC DEFAULT TO OCTAL (AD)

#8B7 BD 3B 3B L88B0 CMPA #04,7 *

#8B7 BD 2B 7A 2B 71 L88B7 LDA #B277 * SYNTAX ERROR IF

#8B7 BD 6E 33 CMPA #$03 BASE B MULTIPLIER

#8B7 BD 2A BSR #B634 ADD DIGIT TO TEMPORARY ACCUMULATOR

* EVALUATE AN AD VARIABLE

#8B7 BD 9F 9F L88BA JSR GETNS3 GET A CHARACTER FROM BASIC

#8B7 BD 2F 2F B8 #B8F2 BRANCH IF NUMERIC

#8B7 BD 1F 5F 2B .88BE CLR FPAM+1 * CLEAR 2 HIGH ORDER

#8B7 BD 5F 51 CLR FPAM+1 * BYTES OF FPAM

#8B7 BD 8F 86 CLR VALTYP SET VARIABLE TYPE TO NUMERIC
* EXPRESSION EVALUATION RAM HOOK

XVECIS PULS U

PULL RETURN ADDRESS AND SAVE IN U REGISTER

CLR VALTP
SET VARIABLE TYPE TO NUMERIC

LDX CHARAD
CURRENT INPUT POINTER TO X

LDAC SD BY 23
GET CHARACTER FROM BASIC

CMPA #X'
HEX AND OCTAL VARIABLES ARE PRECEEDED BY &

LDY #29
BEQ LB7EB
PROCESS A & VARIABLE

LDX #CC
TOKEN FOR &CC

REX LB84
PROCESS FN CALL

LDY #FF
CHECK FOR SECONDARY TOKEN

BNE LB662
NOT SECONDARY

JSR GETHCN
GET CHARACTER FROM BASIC

LDB #83
TOKEN FOR USR

LDX #80,X
PROCESS USR CALL

LSR 2, X
STAD CHARAD
RESTORE BASIC'S INPUT POINTER

JSR LXRA
RETURN TO CALLING ROUTINE

LDX CURLIN
GET CURRENT LINE NUMBER

LDX #80,X
IN DIRECT MODEL

STD $01,X
RETURN NOT IN DIRECT MODE

LDX #2"11
ILLEGAL DIRECT STATEMENT ERROR

LDX #6 AE 46
LDX $01,X
JMP LAC46
PROCESS ERROR

* DEF

LDX [CHARAD]
GET TWO INPUT CHARs

CMPX #$FFB3
TOKEN FOR USR

LDX #80,X
BRANCH IF DEF USR

LDC BD BY 23
GET DESCRIPTOR ADDRESS FOR FN VARIABLE NAME

BRA SD BY 6
DON T ALLOW DEF FN IF IN DIRECT MODE

LDX #200
SYNTAX CHECK FOR =

LDX #80,X
* SAVE IT IN FIRST 2 BYTES OF THE DESCRIPTOR

LDX #50
* AND SAVE IT IN ARRAY VARIABLE SEARCH DISABLE FLAG

LDX #7
JSR LB567
GET VARIABLE DESCRIPTOR

LDX #B8B1
SYNTAX CHECK FOR )

LDX #80,X
TOKEN FOR FN

LDX #FB
DO A SYNTAX CHECK FOR =

LDX #80,X
GET THE ADDRESS OF THE FN NAME DESCRIPTOR

LDX #80,X
* GET THE CURRENT INPUT POINTER ADDRESS AND

LDX #80,X
* SAVE IT IN FIRST 2 BYTES OF THE DESCRIPER

LDX #80,X
= GET THE DESCRIPTOR ADDRESS OF THE ARGUMENT

LDA VARPTR
= VARIABLE AND SAVE IT IN THE DESCRIPTOR OF THE FN NAME

LDA #82,X
MOVE INPUT POINTER TO END OF LINE OR SUBLINE

LDA #30
MOVE TO ARRAY VARIABLE FLAG

LDA #80,X
SET bit 7 OF CURRENT INPUT CHARACTER TO INDICATE AN FN VARIABLE

LDX #83, SC
GET THE DESCRIPTION ADDRESS OF THIS

LDA #83, SC
* SAVE IT IN VARIABLE AND SAVE IT IN V48

LDX #83, SC
STV X48
* RESTORE REGISTERS
1857 88DE ED 84 STD ,X  * GET THE FF
1858 88EF E0 2#  STU #R2,X  * VALUE OF THE ARGUMENT
1859 88E2 35 2#  PULS A  * VARIABLE OFF OF THE
1860 88E4 25 84 STA #RA,X  * STACK AND RE-SAVE IT
1861 88E6 9D A5 JSR GETCH  GET FINAL CHARACTER OF THE FN FORMULA
1862 88EB 16 26 9B LBEQ L8277  SYNTAX ERROR IF NOT END OF LINE
1863 88EC 1# 8F A6 STY CHARAD  RESTORE INPUT POINTER
1864 88EF 39  LBEQ RTS
1865 88F6 4#  * ERROR DRIVER RAM HOOK
1867 88F8 C1 32  XVEC17 CMPB #2+25  CHECK FOR EXBAS ERROR NUMBER
1868 88F2 25 5B  BLD LBEQ  BRANCH IF < EXBAS ERROR
1869 88F4 B0 A7 E9 JSR LAFE9  TURN CASSETTE MOTOR OFF
1870 88F7 BD A9 74 JSR L974A  DISABLE ANALOG MULTIPLEXER
1871 88FA B0 AD 0#  JSR LAD3A  DO PART OF A NEW
1872 88FB 8F 6F CLR DEVMNUM  SET DEVICE NUMBER TO SCREEN
1873 88FD BD 9C JSR L895C  MOVE CURSOR TO START OF NEXT LINE
1874 88FE BD 89 AF JSR L89AF  SEND A ? TO CONSOLE OUT
1875 88F5 8E 8B 09 LDX #L8900-25+2  POINT X TO EXBAS ERRORS
1876 88F6 7E AC 60 JMP LAC6  PROCESS ERROR
1877 88F7 4#  * ADDITIONAL ERROR MESSAGES ADDED BY EXTENDED BASIC
1879 88F8 80 55 46 L890B FCC 'IF'  25 UNDEFINED FUNCTION (FN) CALL
1880 88FA 8D 4E 45 L890D FCC 'ME'  26 FILE NOT FOUND
1882 88F8 4#  * DEF USR
1884 88FA 8D 9F  L890E JSR GETCH  SKIP PAST SECOND BYTE OF DEF USR TOKEN
1885 88F9 A0 BD RSR L891C  GET FN NUMBER
1886 88F9 14 1#  PSHS X  SAVE FN EXEC ADDRESS STORAGE LOC
1887 88FA B0 2D BSR L8944  CALCULATE EXEC ADDRESS
1888 88F9 35 4#  PULS U  GET FN EXEC ADDRESS STORAGE LOC
1889 88F9 AF 0C STX ,U  SAVE EXEC ADDRESS
1890 88F9 39 RTS
1891 88F9 11 5F L891C CLR  DEFAULT TO USRB IF NO ARGUMENT
1892 88F9 9D 9F JSR GETCH  GET A CHARACTER FROM BASIC
1893 88FA 24 06 BCC L8927  BRANCH IF NOT NUMERIC
1894 88FB 8B 3#  SUBA ,A  MASK OFF ASCII
1895 88FC 8B 89 LDX A, A, B  SAVE USR NUMBER IN ACC
1896 88FD 9D 9F JSR GETCH  GET A CHARACTER FROM BASIC
1897 88FE 9D 88 L8927 LDX USRADDR  GET ADDRESS OF STORAGE LOC; FOR USR ADDRESS
1898 88FF 58  A3L  ADD OFFSET TO START ADDRESS OF STORAGE LOC
1899 88F9 3A 3A  ABX
1900 88F9 39 RTS
1901 88F9 02 8D L892C RSR L891C  GET STORAGE LOC OF EXEC ADDRESS FOR USR N
1902 88F9 02 8E LDX ,X  * GET EXEC ADDRESS AND
1903 88F9 34 1#  PSHS X  * PUSH IT ONTO STACK
1905 88F9 33 8D JSR L8926  SYNTAX CHECK FOR ( & EVALUATE EXPR
1906 88F9 3E 9E LDX #FTMEMP  POINT X TO FPB
1907 88F9 9E 86 LDA VALTP  GET VARIABLE TYPE
1908 88F9 A2 7E BEQ L8943  BRANCH IF NUMERIC, STRING IF =/= #
1909 88F9 3C 86 57 JSR L8657  GET LENGTH & ADDRESS OF STRING VARIABLE
1910 88F9 3F 9E LDX FPAP+2  GET POINTER TO STRING DESCRIPTOR
1911 88F9 41 9E LDA VALTP  GET VARIABLE TYPE
1912 88F9 43 39 L8943 RTS  JUMP TO USRB ROUTINE (PSHS X ABOVE)
1913 88F9 44 63  L8944 LDB #8E3  TOKEN FOR =
1914 88F9 46 8D JSR L826F  DO A SYNTAX CHECK FOR =
1915 88F9 49 7E B7 3D JMP L8730  EVALUATE EXPRESSION, RETURN VALUE IN X
1916 8117 8#  * EXTENDED BASIC S I Q ROUTINE
1918 88F9 49 B6 FF #3 XIROSXY LDA PIAW+3  GET PIAW, PORT B CONTROL REGISTER
1919 88F9 4F 2B #1 BML L8952  BRANCH IF #B HZ INTERRUPT
1920 88F9 51 3B RTI
1921 88F9 52 86 FF #2 L8952 LDA PIAW+2  RESET PIA INTERRUPT FLAG
1922 88F9 55 BE #1 12 L8955 LDX TIMVAL  GET REAL TIME CLOCK
1923 88F9 58 #1 #7 LEA #1L,X  INCREMENT IT
1924 88F9 5A BF #1 12 STX TIMVAL  SAVE IT
1925 88F9 5D 7E 9C 3E JMP L9C3E  GO CHECK SOME MORE STUFF
1926 8117 8#  * TIMER
1928 88F9 60 9D 9F L896B JSR GETCH  GET A CHARACTER FROM BASIC
1929 88F9 62 BD BSR L8944  GET NEW TIMER VALUE
1930 88F9 66 8F #1 12 STX TIMVAL  SET TIMER COUNTER
1931 88F9 67 39 RTS
1932 8117 8#  * DEL
1934 88F9 6E 8E #1 12 LDX TIMVAL  GET TIMER VALUE
1935 88F9 6F 9F 52 STA FPAP+2  SAVE TIMER VALUE IN BOTTOM OF FPB
1936 88F9 70 2E JMP L8BBE  CONVERT BALANCE OF FPB TO POSITIVE INTEGER
1937 8117 8#  * DEL
1939 88F9 70 1# 27 2A 06 DEL L8EQ L844A  FC ERROR IF NO ARGUMENT
1940 88F9 74 BD 67 JSR L9AF7  CONVERT A DECIMAL NUMERIC TO BINARY
1941 88F9 77 BD 4D JSR LAD3L  FIND RAM ADDRESS OF START OF A BASIC LINE
1942 88F9 7A 9F 03 STX V0  SAVE RAM ADDRESS OF STARTING LINE NUMBER
1943 88F9 7C 9D A5 JSR GETCH  GET CURRENT INPUT CHARACTER
1944 88F9 7E 27 1# BEQ L899A  BRANCH IF END OF LINE
1945 88F9 80 B1 AC CMPA #8AC  TOKEN FOR ,
1946 88F9 82 7E 2B BNE L89BF  TERMINATE COMMAND IF LINE NUMBER NOT FOLLOWED BY -
1947 88F9 84 9D 9F JSR GETCH  GET A CHARACTER FROM BASIC
1948 88F9 87 27 0# BEQ L89BC  IF END OF LINE, USE DEFAULT ENDING LINE NUMBER
1949 88F9 88 BD 24 JSR L83AE  * CONVERT ENDING LINE NUMBER TO BINARY
1950 88F9 8A 2B #8 BRA L899A  * AND SAVE IT IN BINVAL
1951 88F9 8C 86 FF L898C LDA #8FF  USE #8FF AS DEFAULT ENDING
1952 88F9 8E 7B STA BINVAL  = LINE NUMBER - SAVE IT IN BINVAL

812
1163  B998 DE 03  L899B LDU V03  POINT U TO STARTING LINE NUMBER ADDRESS
1154  B992 BC  L8992 FCB SKP2  SKIP TWO BYTES
1155  B993 EE C4  L8993 LDU ,U  POINT U TO START OF NEXT LINE
1156  B995 EC C4  LDU ,U  CHECK FOR END OF PROGRAM
1157  B997 27 68  BEQ L899F  BRANCH IF END OF PROGRAM
1158  B999 EC 42  LDU $E02U  LOAD ACC WITH THIS LINE S NUMBER
1159  B99B 93 2B  SUBSCTR ENDING LINE NUMBER ADDRESS
1160  B89D 23 F4  BLS L8993  BRANCH IF = END LINE NUMBER
1161  B99F 9E 02  L899F LDV3  GET STARTING LINE NUMBER
1162  B8A1 BD 15  BSR L898B  MOVE (U) TO (X) UNTIL END OF PROGRAM
1163  B8A3 BD AD 21  JSR LA02  RESET BASIC S INPUT POINTER AND ERASE VARIABLES
1164  B8A4 9E 03  LDU V03  GET STARTING LINE NUMBER ADDRESS
1165  B8AB BD AC F1  JSR LACF1  RECONFIG START OF NEXT LINE ADDRESSES
1166  B8AB 7E AC 73  JMP LAC73  JUMP TO BASIC S MAIN COMMAND LOOP
1167  B8AE BD AF 67  L89AE JSR LAF67  GO GET LINE NUMBER CONVERTED TO BINARY
1168  B8B1 7E A5 C7  JMP LAC57  MAKE SURE THERE S NO MORE ON THIS LINE
1169  B8BA 66 C8  L89B6 LDA ,U  GET A BYTE FROM (U)
1170  B8B6 9F 8B  STA ,+  MOVE THE BYTE TO (X)
1171  B8B8 11 9F 3B  L898B CMPU VARTMB  COMARE TO END OF BASIC
1172  B8BB 26 F7  BNE L898C  BRANCH IF NOT AT END
1173  B8BD 9F 1B  STX VARTMB  SAVE (X) AS NEW END OF BASIC
1174  B8BF 39  L89BF RTS

1175  * LINE INPUT
1176  B8C8 BD 88 66  L89CB JSR L886  BS ERROR IF IN DIRECT MODE
1177  B8C9 9D 9F  JSR GETCNCH  GET A CHAR FROM BASIC
1178  B8C5 B1 23  CMFA ?"  * CHECK FOR DEVICE NUMBER FLAG AND
1179  B8CF 26 89  BNE L8D02  * BRANCH IF NOT THERE
1180  B8C9 BD A5 A5  JSR LA5AS  CHECK FOR VALID DEVICE NUMBER
1181  B8CC BD A3 ED  JSR LAJED  CHECK FOR OPEN FILE
1182  B8CF BD B2 60  JSR SYCNOM  SYNTAX CHECK FOR COMMA
1183  B8D0 81 22  L8D02 CMFA "'"  * CHECK FOR END STRING
1184  B8D0 26 8B  BNE L8D01  BRANCH IF NOT PROMPT STRING
1185  B8D0 BD A2 44  JSR LBD44  STREP OFF PROMPT STRING & PUT IT ON STRING STACK
1186  B8D9 06 3B  LDB "'"  *
1187  B8D8 BD 82 6F  JSR LBF6F  * DO A SYNTAX CHECK FOR;
1188  B8D9 BD 8F 9F  JSR LBF9F  REMOVE PROMPT STRING FROM STRING STACK & SEND TO CONSOUT
1189  B8E1 32 7E  L89E1 LEAS ,-$2,5  RESERVE TWO STORAGE SAPTS ON STACK
1190  B8E3 BD BB 35  JSR LB35S  INPUT A LINE FROM CURRENT INPUT DEVICE
1191  B8E6 32 62  LEAS ,$2,5  COMPLET THE STACK
1192  B8EB 6F 6F  JSR DEVNUM  SET DEVICE NUMBER TO SCREEN
1193  B8EA BD 83 57  JSR LB57S  SEARCH FOR A VARIABLE
1194  B8ED 9F 3B  STX VARDRS  SAVE POINTER TO VARIABLE DESCRIPTOR
1195  B8EF 8D E6  JSP LB146  TM ERROR IF VARIABLE TYPE = NUMERIC
1196  B8F7 BD 82 0C  LDX HL1NB0  POINT X TO THE STRING BUFFER WHERE THE INPUT STRING WAS STORED
1197  B8F5 4F  CLRA  TERMINATOR CHARACTER # (END OF LINE)
1198  B8FE BD 85 1A  JSR LB51A  PARSE THE INPUT STRING AND STORE IT IN THE STRING SPACE
1199  B8F9 7E AE 44  JMP LAF64  REMOVE DESCRIPTOR FROM STRING STACK
1200  B8FC BD AF 67  L89FC JSR LAF67  STREP A DECIMAL NUMBER FROM BASIC INPUT LINE
1201  B8FF BD 8B 42  JSP BIWAV  GET BINARY VALUE
1202  BA00 39  RTS
1203  BA02 9E 01  L848B LDX V01  GET CURRENT OLD NUMBER BEING REMEMBERED
1204  BA0A 9F 2B  L848B STX BIWAL  SAVE THE LINE NUMBER BEING SEARCHED FOR
1205  BA0B 7E AD 01  JMP LA021  GO FIND THE LINE NUMBER IN BASIC PROGRAM

1207  * REMEN
1208  BA0B BD AD 26  REUM JSR LA026  ERASE VARIABLES
1209  BA0C BD 6E 0A  L848D LDV0 A  DEFAULT LINE NUMBER INTERVAL
1210  BA0F 0D 65  STD V05  SAVE DEFAULT NUMBERER START LINE NUMBER
1211  BA11 0D CF  STD VCF  SAVE DEFAULT INTERVAL
1212  BA13 5F  CLRA  NOW ACCO = 0
1213  BA14 0D D1  STD V01  DEFAULT LINE NUMBER OF WHERE TO START RENUMBERING
1215  BA16 9D A5  JSP GETCNCH  GET CURRENT INPUT NUMBER
1216  BA18 24 86  BCC LBA2A  BRANCH IF NOT NUMERIC
1217  BA1A 9D 0E  BSR L89BF  CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY
1218  BA1C 9F 05  STD V05  SAVE LINE NUMBER WHERE REMEMBERING STARTS
1219  BA1E 9D A5  JSR GETCNCH  GET CURRENT INPUT NUMBER
1220  BA20 27 1B  L842B BEQ L8ADB  BRANCH IF END OF LINE.
1221  BA22 BD BD 6D  JSR SYCNOM  SYNTAX CHECK FOR COMMA
1222  BA25 24 86  BCC LBA2D  BRANCH IF NEXT CHARACTER NOT NUMERIC
1223  BA27 BD 0D  BSR L89FC  CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY
1224  BA29 9F 01  STX V01  SAVE NEW NUMBER LINE
1225  BA2B 9D A5  JSR GETCNCH  GET CURRENT INPUT CHARACTER
1226  BA2D 27 8E  L8ADB BEQ L8ADB  BRANCH IF END OF LINE
1227  BA2F BD 6D  JSR SYCNOM  SYNTAX CHECK FOR COMMA
1228  BA32 24 86  BCC LBA3A  BRANCH IF NEXT CHARACTER NOT NUMERIC
1229  BA34 BD BD 6D  BCC L89BF  CONVERT DECIMAL NUMBER IN BASIC PROGRAM TO BINARY
1230  BA36 9F CF  STX VCF  SAVE NEW INTERVAL
1231  BA3E 27 49  BCC LBA39  FC ERROR
1232  BA3A BD A5 C7  L843A JSR LAC57  CHECK FOR MORE CHARACTERS ON LINE - SYNTAX ERROR IF ANY
1233  BA3D BD 03  L843D BSR LBAB2  GO GET ADDRESS OF OLD NUMBER BEING REMEMBERED
1234  BA3F 9F 03  STD V03  SAVE ADDRESS
1235  BA41 9E D5  LDX V05  GET NEXT REMEMBERED LINE NUMBER TO USE
1236  BA43 BD 08  BSR LBAB4  FIND THE LINE NUMBER IN THE BASIC PROGRAM
1237  BA45 9C 03  CMPU V03  COMPARE TO ADDRESS OF OLD LINE NUMBER
1238  BA47 25 3A  BLD LBAB3  FC ERROR IF NEW ADDRESS < OLD ADDRESS
1239  BA49 BD 47  BSR LB467  MAKE SURE REMEMBERED LINE NUMBERS WILL BE IN RANGE
1240  BA4B BD AD 0D  JSR LBADD  CONVRET ASCII LINE NUMBERS TO EXPANDED BINARY
1241  BA4E BD AF CE  JSR LACF  RECALCULATE NEXT LINE RAM ADDRESSES
1242  BA51 BD 8F 08  JSR LBAB2  GET RAM ADDRESS OF FIRST LINE TO BE REMEMBERED
1243  BA53 9F 03  STX V03  SAVE IT
1244  BA55 BD 0A  BSR LB491  MAKE SURE LINE NUMBERS EXIST
1245  BA57 BD 8F 08  BSR LB468  INSERT NEW LINE NUMBERS IN LINE HEADERS
1246  BA59 BD 3E  BSR LB491  INSERT NEW LINE NUMBERS IN PROGRAM STATEMENTS
1247  BA5A BD 7B  JSR LB708  CONVERT PACKED BINARY LINE NUMBERS TO ASCII
1248  BA5E BD AD 26  JSR LA026  ERASE VARIABLES
DISASSEMBLY OF EXTENDED BASIC 1.1

1249 BAE6 BD AC EF JSR LACEF RECALCULATE NEXT LINE RAM ADDRESS
1250 BAE6 7E AC 73 JMP LAC73 GO BACK TO BASIC S MAIN LOOP
1251 BAE7 B6 LBA67 FCB SKP100 SKIP ONE BYTE - LDA #54F
1252 BAE8 4F LBA68 NEW LINE NUMBER FLAG - 0; INSERT NEW LINE NUMBERS
1253 BAE9 97 D8 STA V09 SAVE NEW LINE NUMBER FLAG; 0 = INSERT NEW NUMBERS
1254 BAE8 9E D3 LDX V03 GET ADDRESS OF OLD LINE NUMBER BEING REMOVED
1255 BAE8 DC 65 GET THE CURRENT REMOVED LINE NUMBER
1256 BAE6 BD 15 BSR LBA68 RETURN IF END OF PROGRAM
1257 BAE7 B1 0D LBA71 V09 CHECK NEW LINE NUMBER FLAG
1258 BAE7 26 B2 BNE LBA77 BRANCH IF NOT INSERTING NEW LINE NUMBERS
1259 BAE7 ED 02 STD $82,X STORE THE NEW LINE NUMBER IN THE BASIC PROGRAM
125A BAE7 AC 84 LBA77 LDX POINT X TO THE NEXT LINE IN BASIC
125B BAE9 0D B0 BSR LBA68 RETURN IF END OF PROGRAM
125C BAE7 D0 0D STA V09 CHECK NEW LINE NUMBER FLAG
125D BAE7 3B 0E BNE LBA77 BRANCH IF NOT INSERTING NEW LINE NUMBERS
125E BAE7 ED 02 STD $82,X STORE THE NEW LINE NUMBER IN THE BASIC PROGRAM
125F BAE7 AC 84 LBA77 LDX POINT X TO THE NEXT LINE IN BASIC
1260 BAE7 1B 0D BSR LBA68 RETURN IF END OF PROGRAM
1261 BAE7 BD 0D BSR LBA68 RETURN IF END OF PROGRAM
1262 BAE7 D0 0D STA V09 CHECK NEW LINE NUMBER FLAG
1263 BAE7 D0 0D STA V09 CHECK NEW LINE NUMBER FLAG
1264 BAE7 2F 34 LBA85 CMPA @MAXLIN LARGEST LINE NUMBER = $F9FF
1265 BAE8 21 0E BLD LBA71 BRANCH IF LEGAL LINE NUMBER
1266 BAE8 7E 0A 4A LBA83 JMP LBA4A IF ERROR IF LINE NUMBER MS BYTE > $F9
1267 BAE7 * TEST THE TWO BYTES POINTED TO BY X.
1268 */ NORMAL RETURN IF <> 0. IF = 0 (END OF
1269 */ PROGRAM) RETURN IS PULLED OFF STACK AND
1270 */ YOU RETURN TO PREVIOUS SUBROUTINE CALL.
1271 BAE8 34 06 LBA86 PSK B,A RESTORE B
1272 BAE8 BC 4C LDO X TEST THE 2 BYTES POINTED TO BY X
1273 BAE8 35 06 PULS B,A SAVE ACC
1274 BAE8 26 02 BNE LBA99 BRANCH IF NOT END OF PROGRAM
1275 BAE8 32 62 LEAS $82,S PURGE RETURN ADDRESS FROM STACK
1276 BAE8 39 LBA95 RTS GET START ADDRESS OF PROGRAM
1277 BAE8 91 19 LBA91 LDX TXTTAM GET START OF BASIC PROGRAM
1278 BAE8 33 1F LEA X,$01,X MOVE POINTER BACK ONE
1279 BAE8 38 01 LBA95 LEA X,$81,X MOVE POINTER UP ONE
127A BAE8 BD 0D BSR LBA68 RETURN IF END OF PROGRAM
127B BAE8 33 03 LBA99 LEA X,$81,X SKIP OVER NEXT LINE ADDRESS AND LINE NUMBER
127C BAE8 33 01 LBA98 LEA X,$81,X MOVE POINTER TO NEXT CHARACTER
127D BAE8 AD 04 BLD X,CX CHECK CURRENT CHARACTER
127E BAE8 BF 27 FA REQS LBA95 BRANCH IF END OF LINE
127F BAE8 A1 08 STX TEMPTR SAVE CURRENT POINTER
1280 BAE8 3A 04 DEC LBA82 BRANCH IF START OF PACKED NUMERIC LINE
1281 BAE8 A4 07 BNE LBA92 BRANCH IF LINE NUMBER EXISTS
1282 BAE8 4A 04 DEC LBA93 BRANCH IF LINE NUMBER EXISTS
1283 BAE8 A5 08 DSAC LBA95 BRANCH IF LINE NUMBER EXISTS
1284 BAE8 2B 04 LBA91 LD $82,X GET LS BYTE OF LINE NUMBER
1285 BAE8 36 04 LD $84,X DEC ZERO CHECK BYTE
1286 BAE8 0D 50 BNE LBA91 BRANCH IF MS BYTE <> 0
1287 BAE8 9F 47 STA $84,X CLEAR MS BYTE
1288 BAE8 2E LEA $81,X CLEAR LS BYTE
1289 BAE8 00 08 LBA98 LEA X,$81,X MOVE NEXT CHARACTER IF > 3
128A BAE8 AD 0C REQS LBA95 BRANCH IF LINE NUMBER EXISTS
128B BAE8 32 03 LBA9C REQS X,$81,X SAVE BINARY LINE NUMBER
128C BAE8 DC 02 STD BIVAL SAVE TRIAL LINE NUMBER
128D BAE8 4B 0A LBA99 JSR LAM01 FIND RAM ADDRESS OF A BASIC LINE NUMBER
128E BAE8 BD 0E LBA9C JSR LAM01 GET BACK POINTER TO START OF PACKED LINE NUMBER
128F BAE8 A4 0B BLD LBA9C BRANCH IF NO LINE NUMBER MATCH FOUND
1290 BAE8 8C 07 LD $V47 GET START ADDRESS OF LINE NUMBER
1291 BAE8 4C 0B INC X,$0+ * SET 1ST BYTE = 0, TO INDICATE LINE NUMBER EXISTS IF CHECKING FOR
1292 BAE8 * EXISTENCE OF LINE NUMBER, SET 1 = 1 IF INSERTING LINE NUMBERS
1293 BAE9 0D 19 LBA9D LDX TXTTAM GET BEGINNING OF BASIC PROGRAM
1294 BAE9 BD 04 BNE LBA95 BRANCH IF MS BYTE <> 0
1295 BAE9 E1 06 LBA91 LDX HARAD *GET CURRENT INPUT POINTER
1296 BAE9 38 01 LEA X,$81,X *AND BUMP IT ONE
1297 BAE9 BD 0F LBA95 BSR LBA68 RETURN IF END OF PROGRAM
1298 BAE9 32 02 LEA X,$82,X SKIP PAST NEXT LINE ADDRESS
1299 BAE9 38 01 LBA9E LEA X,$81,X ADVANCE POINTER BY ONE
129A BAE9 9F 06 LBA9B STX HARAD SAVE NEW BASIC INPUT POINTER
129B BAE9 0D 0F LBA9D GET NEXT CHARACTER FROM BASIC
129C BAEF 4D 19 LBAFE TSTA CHECK THE CHARACTER
129D BAEF 27 EF REQS LBA91 BRANCH IF END OF LINE
129E BAEF 22 FA BPL LBA0D BRANCH IF NOT A TOKEN
129F BAEF 9E AD LDA CHARAD GET CURRENT INPUT POINTER
12A0 BAEF 01 FF CMPA #$FF IS THIS A SECONDARY TOKEN?
12A1 BAEF 27 EF REQS LBA91 YES - IGNORE IT
12A2 BAEF AD 01 A8 JSR RV£C22 HOOK INTO RAM AND CHECK FOR USER ADDED TOKENS
12A3 BAEF B1 07 CMPA #$47 TOKEN FOR THEM
12A4 BAEF 27 12 BNE LBA93 YES
12A5 BAEF B1 04 CMPA #$8A TOKEN FOR ELSE
12A6 BAEF 27 0E REQS LBA91 YES
12A7 BAEF 80 01 AL JMP LBA93 YES
12A8 BAEF 9F 09 LEA X,$81,X GET A CHARACTER FROM BASIC
12A9 BAEF 01 A5 CMPA #$4A TOKEN FOR TO
12AA BAEF 27 84 BNE LBA91 YES
12AB BAEF 9F 0A LEA X,$81,X GET A CHARACTER FROM BASIC
12AC BAEF 01 A5 CMPA #$4A TOKEN FOR SUB
1345 B811 26 DB RNE LBAEB NO
1346 B813 90 9F L8B13 JSR GETNCH GET A CHARACTER FROM BASIC
1347 B815 25 84 BLO L8B18 BRANCH IF NUMERIC
1348 B817 90 46 L8B17 JSR GETNCH GET CURRENT BASIC INPUT CHARACTER
1349 B819 20 04 BRA LBAEF KEEP CHECKING THE LINE
1350 B818 9E A6 L8B18 LDX CHARAD GET CURRENT INPUT ADDRESS
1351 B810 90 4 0 PHS X SAVE IT ON THE STACK
1352 B81F BD AF 67 JSR LAF67 CONVERT DECIMAL BASIC NUMBER TO BINARY
1353 B820 9E A6 L8B18 LDX CHARAD GET CURRENT INPUT POINTER
1354 B824 A6 82 L8B24 LDX ,X GET PREVIOUS INPUT CHARACTER
1355 B826 BD 9A AA JSR L9A0A CLEAR CARRY IF NUMERIC INPUT VALUE
1356 B829 25 F9 BLO L8B24 BRANCH IF NON-NUMERIC
1357 B82B 90 4 0 LEAX ,X MOVE POINTER UP ONE
1358 B82D 1F 10 TFR X,D NOW ACCO POINTS TO ONE END OF LINE NUMBER
1359 B82F 80 61 SUBX ,S SUBTRACT PRE-NUMERIC POINTER LS BYTE
1360 B831 C0 #5 SUBB #585 MAKE SURE THERE ARE AT LEAST 5 CHARACTERS IN THE NUMERIC LINE
1361 * 1362 B833 27 00 BEQ L8B55 BRANCH IF EXACTLY 5
1363 B835 25 8A BLO L8B41 BRANCH IF < 5
1364 B837 33 64 LDA ,X TRANSFER X TO U
1365 B839 58 NEGB NEGATE B
1366 B83A 38 85 LEAX B,X MOVE X BACK BY 5
1367 B83C BD 89 B8 JSR L8BB8 **MOVE BYTES FROM (U) TO (X) UNTIL
1368 * 1369 B83F 2B 14 BRA L8B55
1370 * FORCE FIVE BYTES OF SPACE FOR THE LINE NUMBER
1371 B841 90 47 L8B41 STX V47 SAVE END OF NUMERIC VALUE
1372 B843 9C 1B LDX VARTAB GET END OF BASIC PROGRAM
1373 B845 9F 43 STX V43 SAVE IT
1374 B847 58 NEGB NEGATE B
1375 B848 3B 85 LEAX B,X ADD IT TO END OF NUMERIC POINTER
1376 B84A 90 41 STX V41 SAVE POINTER
1377 B84B 9F 1B LDX VARTAB STORE END OF BASIC PROGRAM
1378 B84E BD AC 1E JSR LACIE ACCO = TOP OF ARRAYS - CHECK FOR ENOUGH ROOM
1379 B851 9E 45 LDX V45 * GET AND SAVE THE
1380 B853 9E A6 LDX CHARAD * NEW CURRENT INPUT POINTER
1381 B855 35 18 L8B55 PULS X RESTORE POINTER TO START OF NUMERIC VALUE
1382 B857 86 61 LDA #581 NEW LINE NUMBER FLAG
1383 B859 A7 84 STA ,X * SAVE NEW LINE FLAG
1384 B85B A7 02 STA #2,X *
1385 B85D A7 04 STA #4,X *
1386 B85F 56 2B LDX BINVAL SAVE MS BYTE OF BASIC LINE NUMBER
1387 B861 26 64 RNE L8B67 BRANCH IF IT IS NOT ZERO
1388 B863 C6 41 LDA #581 SAVE A 1 IF BYTE IS 0; OTHERWISE, BASIC WILL
1389 THINK IT IS THE END OF A LINE
1390 B865 6C 02 INC #2,X IF 2,X = 2, THEN PREVIOUS BYTE WAS A ZERO
1391 B867 E7 01 L8B67 LEAX ,X SAVE MS BYTE OF BASIC LINE NUMBER
1392 B869 62 2C LEAX B,X GET 5 BYTE OF BASIC LINE NUMBER
1393 B86B 26 64 RNE L8B67 BRANCH IF NOT A ZERO BYTE
1394 B86C 06 41 LDX #581 SAVE A 1 IF BYTE IS 0
1395 B86F 6C 04 INC #4,X IF 4,X = 2, THEN PREVIOUS BYTE WAS A 0
1396 B871 E7 03 L8B71 STX #3,X SAVE LS BYTE OF BASIC LINE NUMBER
1397 B873 90 45 JSR GETNCH GET CURRENT INPUT CHARACTER
1398 B875 81 2C CMPA #$00 IS IT A COMMA?
1399 B877 27 9A BEQ L8BB3 YES - PROCESS ANOTHER NUMERIC VALUE
1400 B879 20 9C BRA L8B17 NO - GO GET AND PROCESS AN INPUT CHARACTER
1401 B87B 9E 19 L8B78 LDX TXTAB POINT X TO START OF BASIC PROGRAM
1402 B87D 38 1F LEAX S,X,I MOVE POINTER BACK ONE
1403 B87F 3B 01 L8B7F LEAX #1,X MOVE POINTER UP ONE
1404 B881 EC 02 LDO #0,X GET ADDRESS OF NEXT LINE
1405 B883 08 68 STD CURLIN SAVE IT IN CURLIN
1406 B885 BD 8A 86 JSR L8BA8 RETURN IF END OF PROGRAM
1407 B887 3B 83 L8B83 SKIPOVERADDRESSOFNEXTLINE AND1BYTEOFLINETHE
1408 B88A 3B 01 L8B8A LEAX #1,X MOVE POINTER UP ONE
1409 B88C A6 84 LD#L8BB8
1410 B88C 04 8A LDX,X GET CURRENT CHARACTER
1411 B88E 27 EF BEQ L8B7F BRANCH IF END OF LINE
1412 B88F 4A BNE DECA INPUT CHARACTER = 17 - VALID LINE NUMBER
1413 B891 27 1B BEQ L8BAA YES
1414 B893 88 92 SUBA #582 INPUT CHARACTER 37 - UL LINE NUMBER
1415 B895 26 F3 BNE L8BBB NO
1416 B897 34 18 PHSX X SAVE CURRENT POSITION OF INPUT POINTER
1417 B899 BE 8B DB LDX #L8BO9-1 POINT X TO UL MESSAGE
1418 B89C BD 90 9C JSR STRINGOUT PRINT STRING TO THE SCREEN
1419 B89E BD 90 04 JSR AE4A GET INPUT POINTER
1420 B8A1 EC 01 LDO #01,X GET THE UNDEFINED LINE NUMBER
1421 B8A3 BD 8C 00 JSR L8BDC CONVERT NUMBER IN ACCO TO DECIMAL AND DISPLAY IT
1422 B8A5 BD 80 66 JSR L8BDC PRINT IN XXX XXXX XXXX = CURRENT LINE NUMBER
1423 B8A9 BD 89 5A JSR L8B9B SEND A CR TO CONSOLE OUT
1424 B8AC 35 18 LD#L8B8A
1425 B8AE 3C 04 L8B8A PULS X SAVE POSITION CURRENT INPUT POINTER
1426 B8B0 EC 01 LDO #01,X LOAD ACCO WITH BINARY VALUE OF LINE NUMBER
1427 B8B2 52 00 L8B08 ST#PPAF-1 SAVE 5 IN BOTTOM 2 BYTES PPAD OF PPAF
1428 B8B4 BD 88 0E JSR L8BBE ADJUST REST OF PPAD AS AN INTEGER
1429 B8B7 BD 00 69 JSR L8BB9 CONVERT PPAF TO ASCII, STORE IN LINE NUMBER
1430 B8BA 35 48 U LOAD X WITH PREVIOUS ADDRESS OF INPUT POINTER
1431 B8BC C6 05 SUBX ,S SUBTRACT PRE-NUMERIC POINTER LS BYTE
1432 B8BE 3B 81 L8B8E LEAX #1,X EACH EXPANDED LINE NUMBER USES 5 BYTES
1433 B8BE 3B 01 L8B8E MOVE POINTER FORWARD ONE
1434 B8CB A6 84 LDA ,X GET AN ASCII BYTE
1435 B8CC 27 95 BEQ L8BBC BRANCH IF END OF NUMBER
1436 B8CC 4A 5A DECB DECREMENT BYTE COUNTER
1437 B8CC A7 C8 STA +U/ STORE ASCII NUMBER IN BASIC LINE
1438 B8CC 2F F5 BRA L8BBE CHECK FOR ANOTHER DIGIT
1439 B8CB 3C 04 L8BBC Transfer NEW LINE POINTER TO (X)
1440 B8CB 5D TSTB DOES THE NEW LINE NUMBER REQUIRE 5 BYTES?
1441 B8CC 27 BE BEQ L8BBC YES - GO GET ANOTHER INPUT CHARACTER
1442 B8CE 31 C4 LEAX ,U SAVE NEW LINE POINTER IN X

185
1441  RB08 33 C5 LEA B,U POINT U TO END OF 5 BYTE PACKED LINE NUMBER BLOCK.
1442  RB02 BD 88 JSR L89BB MOVE BYTES FROM (U) TO (X) UNTIL END OF PROGRAM.
1443  RB05 3A 44 LEAX _Y LOAD (X) WITH NEW LINE POINTER
1444  RB07 2B B3 BRA L89BC GO GET ANOTHER INPUT CHARACTER
1445
1446  RB09 55 4C 28 L88D0 FFC 'UL ' UNKNOWN LINE MESSAGE
1447  RBDC B0 FB FCB #
1448
1449  * CONVERT AN INTEGER INTO AN ASCII STRING AND PRINT IT ON THE SCREEN
1450  RBDD BD B7 48 HEXDOL JSR L748 CONVERT FPAM INTO A POSITIVE 2 BYTE INTEGER
1451  RBEE BE 03 09 LDX #STRBUF+2 POINT TO TEMPORARY BUFFER
1452  RBEE C6 84 LDB #A4 CONVERT 4 NIBBLES
1453  RB8E 34 04 L88E5 PSHS B SAVE NIBBLE COUNTER
1454  RB87 5F CLRB CLEAR CARRY FLAG
1455  RB8E 94 04 LDA #A4 4 SHIFTS
1456  RB8A #B 53 L8B8A ASL FPAM+1 * SHIFT BOTTOM TWO BYTES OF
1457  RB8C 49 52 ROL FPAM+2 * FPAM LEFT ONE BIT (X2)
1458  RB8E 59 ROLC IF OVERFLOW, ACCB => 0
1459  RB8F 4A DECA * DECREMENT SHIFT COUNTER AND
1460  RB8F 26 0B BNE L8B8A * BRANCH IF NOT DONE
1461  RB8F 50 TSBT CHECK FOR OVERFLOW
1462  RBFD 26 0A BNE L8BFF BRANCH IF OVERFLOW
1463  RB8F 65 0A LDA . ,S * GET NIBBLE COUNTER,
1464  RB8F 7A 4A DECA . * DECREMENT IT AND
1465  RB8F 27 05 BEQ L8BFF * BRANCH IF DONE
1466  RB8A BC #D 09 CMPS R, #A4 = DO NOT DO A CONVERSION UNTIL A NON-ZERO
1467  RBFD 27 0C BEQ L8CBR = BYTE IS FOUND - LEADING ZERO SUPPRESSION
1468  RB8F C8 08 L8BFF ADD R, #0 ADD IN ASCII ZERO
1469  RB81 C1 39 CMPS R, #9 COMPARSE TO ASCII 9
1470  RB83 23 02 BLS L8CB7 BRANCH IF < 9
1471  RB81 C8 07 ROLC R, R ADD ASCII OFFSET IF HEX LETTER
1472  RB87 E7 0B L8C7B STB , +,X+ STORE HEX VALUE AND ADVANCE POINTER
1473  RB89 6F 04 CLR , + CLEAR NEXT BYTE - END OF STRING FLAG
1474  RB8C 38 04 L8C88 PULSB B GET NIBBLE COUNTER,
1475  RB8C 5A DECB * DECREMENT IT AND
1476  RB8E 26 05 BNE L88E5 * BRANCH IF NOT DONE
1477  RB8C 32 62 LEAS @R, +5 PURGE RETURN ADDRESS OFF OF STACK
1478  RB82 BE 03 08 LDX #STRBUF+1 RESET POINTER
1479  RB15 7E 95 1B JMP L851B SAVE STRING ON STRING STACK
1480
1481  * DLOAD
1482  RB18 BC AD 24 DLLOAD JSR L429 CLOSE FILES
1483  RB18 9D 05 L8C1B JSR GETCH GET THE CURRENT INPUT CHARACTER
1484  RB18 RC 4D RCIC R, #M CHECK FOR DLLOADM
1485  RB1F 34 02 PSHS A = SAVE DLLOADM (+1), DLLOAD (+M) FLAG
1486  RB21 26 02 BNE L8CB2 BRANCH IF LOAD
1487  RB23 9D 0F JSR GETCH GET AN INPUT CHAR FROM BASIC
1488  RB28 BD 05 7B L82C5 JSR L57B GET THE NAME OF FILE FROM BASIC
1489  RB28 9D 05 JSR GETCH GET CURRENT INPUT CHAR FROM BASIC
1490  RB2A 27 1B BNE L8C44 BRANCH IF END OF LINE
1491  RB2C 0B 0D JSR SYMCOMMA SYNTAX CHECK FOR COMMA
1492  RB37 81 2C CMPS R, #1 = CHECK FOR TWO CONSECUTIVE COMAS
1493  * BRANCH IF , IF THIS CASE IS SELECTED
1494
1495  * * THE BAUD DELAY MUST HAVE BEEN PREVIOUSLY STORED IN DIBAUD
1496  RB33 BD 07 0B JSR EVALNPB EVAL Expr, RETURN VALUE IN ACCB
1497  RB36 86 08 LDA #444* DELAY VALUE FOR 380 BAUD
1498  RB38 50 TSBT WAS ARBITRARY = @M
1499  RB39 27 07 BEQ L8C42 YES = 380 BAUD
1500  RB38 86 0C LDA #44 DELAY VALUE FOR 1200 BAUD
1501  RB3A 0A DECB = CHECK FOR ARGUMENT OF 1
1502  RB3E 1B 2B L8C47 STA L84A FC IF NOT ZERO OR ONE OR COMPA
1503  RB42 97 66 L8424 STA DLBAUD SAVE DELAY VALUE
1504  RB44 9D 08 L8444 B,C, D = TRANSMIT FILE NAME AND READ IN FILE STATUS
1505  RB47 34 02 PSHS A = SAVE ACCA
1506  RB49 8D 06 LDA #3 DLOAD DEVICE NUMBER TO -3
1507  RB4B 97 0F STA DEVENM SET DEVICE NUMBER TO DLOAD
1508  RB4D 35 02 PULS A RESTORE ACCA
1509  RB4F 0D 0B TST , +5 DLOAD OR DLOADM?
1510  RB51 27 32 BEQ L8CB5 DLOADM
1511
1512  * READ IN A DLOAD FILE
1513  RB53 2D AD 0C L853 JSR LAC7 CHECK FOR END OF LINE - SYNTAX ERROR IF NOT
1514  RB56 50 TSBT CHECK ASCII FLAG
1515  RB57 27 06 BEQ L855 FM ERROR IF NOT ASCII
1516  RB59 BD AD 19 JSR LAD19 GO DO A NEW
1517  RB5C 7E AC 7C JMP LAC7C * JUMP BACK TO BASIC S MAIN INPUT LOOP;
1518  * * DLOAD FILES MUST BE ASCII FILES
1519  RB5F 7E AD 16 L85F JMP L616 BAD FILE MODE ERROR
1520
1521  * EX5B DLOAD PROCESSOR
1522  RB62 9D 09 L8CB2 JSR GETCH GET A CHAR FROM BASIC
1523  RB64 81 0D CMPS R, #M CHECK FOR CLOADM
1524  RB66 1B 2B 2E L8BNE LA498 GO DO A DLOAD
1525  RB6A 87 07 CLR L85L STA CLOSE FILES
1526  RB6C 9D 0F JSR GETCH GET A CHAR FROM BASIC
1527  RB6E BD A5 7B JSR L57B STRIP A FILENAME OFF OF THE BASIC LINE
1528  RB71 0D AD 08 JSR L57B SEARCH FOR FILE
1529  RB74 7D 04 L8C47 CMDL CASBUP+10 CHECK FILE MODE
1530  RB77 1B 27 1A L8C9E LA585 BRANCH TO CLOAD IF NOT BLOCK LOAD
1531  RB7E FE 01 22 L8C8B+8 SAVE FILE TYPE AND ASCII FLAG IN U
1532  RB7E 9D 06 JSR L57B SET DEVICE NUMBER TO -1 (CASSETTE)
1533  RB8B BD AD 35 JSR L635 GO READ IN A DATA BLOCK
1534  RB83 1F 0D TST , U,D PUT FILE TYPE & ASCII FLAG BACK IN ACO
1535
1536  * STRIP A LOAD OFFSET FROM THE BASIC LINE, THEN LOAD IN BLOCKS OF
1537  * DATA (CLOADM,DLOADM) WHICH ARE PRECEEDED BY A 5 BYTE PRE OR POST.
* AMBLE. THE PREMABLE CONTAINS A BLOCK LENGTH AND A LOAD ADDRESS SO
* THAT ANY NUMBER OF NON-CONTIGUOUS BLOCKS MAY BE LOADED. THE POST-
* AMBLE WILL TERMINATE THE LOADING PROCESS AND PROVIDE A TRANSFER ADDRESS

BC85 B3 #0 #0 L8CB5 SUB UDBR   * CHECK FILE STATUS;
BC88 26 D5 BNE L8CF5   * FN ERR IF MODE <> 2 OR TYPE <> 0
BC8A 9E BA LDX ZERO ZERO THE X REG - DEFAULT OFFSET
BC8C 90 A5 JSR GETCCH GET CURRENT INPUT CHARACTER
BC8E 27 #6 BEQ L8CB6 BRANCH IF END OF LINE
BC8F B0 B2 DJNZ SYNTAX Check FOR COMMA
BC93 BD #7 #D JSR LB73D EVAL INTEGER EXP - RETURN VALUE IN X
BC96 9F D3 L8CB6 STX VD3 SAVE OFFSET
BC98 BD #5 L8CB7 JSR L8CB7 SYNTAX ERROR IF MORE CHAR ON LINE
BC9B B0 29 L8CB9 BSR L8CC6 GO GET EOF FLAG FROM CONSOLE IN
BC9D 34 #2 PSHS A SAVE IT ON THE STACK
BC9F B0 1E BSR L8CBF * READ IN BLOCK LENGTH FROM CONSOLE IN
BCA1 1F #2 TFR D,Y * AND SAVE IT IN Y
BCA3 BD IA BSR L8CBF GET LOAD ADDRESS FROM CONSOLE IN
BCA5 03 D3 ADD OFFSET TO LOAD ADDRESS
BCA7 DD 9D STD EXECJP SAVE IN EXEC ADDRESS
BCA9 1F #1 TFR D,X SAVE LOAD ADDRESS IN X
BCAB A6 #6 LDA ,+ GET EOF FLAG FROM STACK
BCAD #6 17 7C LBNE LA24D CLOSE FILES IF POSTMABLE BLOCK
BCAE B0 13 L8CB1 BSR L8CC6 GET A CHARACTER FROM CONSOLE IN
BCB3 A7 B4 STA ,X SAVE IT IN RAM
BCB5 A1 #8 CMPA ,A COMPARE SAVED BYTE TO ACTUAL BYTE
BCB7 26 14 BNE L8CC6 'IO ERROR IF NOT + (SAVED IN ROM OR BAD RAM)
BCB9 31 3F LEAY $-#1,Y DECEDMENT BYTE COUNT
BCBA 26 1B BNE L8CB1 READ MORE CHARACTERS
BCBD #0 DC BRA L8CB9 LOOK FOR ANOTHER BLOCK OF DATA

* GET 2 CHARACTERS - RETURN THEM IN ACCB
L8CBF BSR L8CC1 GET A CHARACTER IN ACCB
L8CC1 BSR L8CC6 GET A CHARACTER IN ACCA
L8C3 1E #9 EXX A,B SAVE IT IN ACCB
L8CC5 39 BSR RTS

L8CC6 BD A1 7E L8CB6 JSR LA176 GET A CHARACTER FROM CONSOLE IN
L8CC9 BD #7 TST CINBLF IS FILE EMPTY?
L8CCB 27 #B BEQ L8CC5 RETURN IF NOT EMPTY
L8CCD #E 16 9F L8CCD JMP LA919 ID ERROR IF EMPTY
L8CDB BD #2 L8CCD BSR L8DB14 TRANSMIT FILE NAME, RETURN FILE STATUS
L8CCF BD #6 PSHS B,A SAVE STATUS ON STACK
L8CD 04 4C ITCX CHECK FILE TYPE
L8CE 27 #6 BEQ L8CCD RE ERROR IF FILE NOT FOUND
L8CD 0D #A LDX ZERO RE U REG-FIRST BLOCK NUMBER
L8CD BD #9 BSR L8CC4 READ IN 12B CHARACTERS
L8CD 35 #6 PULS A,B,PC GET FILE STATUS BACK AND RETURN
L8CCD 6C #4 L8CCD BSR L8DB14 RE ERROR.
L8CDF 7E AC 46 JMP L8CA64 GO TO ERROR SERVICING ROUTINE

* REFILL CONSOLE IN CHARACTER BUFFER FROM DLLOAD
L8CE2 LDV C8BFAD GET BLOCK NUMBER
L8CE4 3B 41 L8CE4 LEAK #81,U * INCREMENT BLOCK NUMBER
L8CE8 9F #7 STX C8BFAD * AND SAVE IT
L8CEB BE #1 DA LDX #CASBUF USE CASBUF AS DLLOAD INPUT BUFFER
L8CEB BD #7 C3 JSR LB07C READ 12B CHARACTERS (ONE BLOCK) INTO BUFFER
L8CEE 7E A6 #4 JMP L8CA64 RESET CONSOLE IN BUFFER

* CONSOLE IN RAM HOOK
L8CF1 96 6F #XV64 LDA DEVRM GET DEVICE NUMBER
L8CF3 BD #0 CMPA #3 DLOAD DEVICE NUMBER
L8CF5 26 8A BNE L8DB15 BRANCH IF NOT DLOAD
L8CF7 32 #2 LEAS #82,S 5URST RETURN ADDR FROM STACK.
L8CF9 #F 7B CLR CINBLF RESET EMPTY/FULL FLAG
L8CFC BD #0 79 TST CINCR ANY CHARACTERS LEFT IN BUFFER?
L8CFD 26 #D BNE L8DB15 YES, GO ONCE
L8CFF #3 7B LDA CINBLF SET EMPTY/FULL FLAG TO EDF
L8D01 39 LDB01 RTS
L8D03 BD02 34 #4 LDB02 PSHS U,Y,X,B SAVE REGISTERS
L8D04 B0 7A LDX CINFR GET CONSOLE IN CHARACTER BUFFER
L8D06 A6 #B LDA ,X+ GET A CHARACTER
L8D08 BD34 #2 PSHS A SAVE IT ON THE STACK
L8D0B 9A 7F STX CINFR SAVE NEW CHARACTER BUFFER
L8D0C 8A 79 DEC CINCR DECREMENT CHARACTER COUNTER
L8D0E 26 #2 LDA L8DB15 RETURN IF BUFFER NOT EMPTY
L8D10 BD #8 BSR L8CE2 GO REFILL THE CHARACTER BUFFER
L8D12 35 F6 L8DB12 PULS A,B,X,Y,U,PC RESTORE REGISTERS AND RETURN

* TRANSMIT FILE NAME - READ FILE STATUS FROM SENDER
L8D14 4F L8DB14 PSHS A,B SAVE FILE NAME ON STACK For TEMP VARIABLES
L8D15 34 #6 L8DB15 LEAY ,S SAVE SPACE ON STACK FOR TEMP VARIABLES
L8D19 10 #2 LDA L8DB10 STACK TO Y (TFR S,Y) - SAVE VARIABLE POINTER
L8D1B BD #2 L8DB10 INCREMENT ATTEMPT COUNTER
L8D1D 06 #A L8DB10 * GET FILE REQUEST CONTROL CODE
L8D1F BD #7 LDA L8DB10 * AND TRANSMIT IT
L8D21 26 #F BNE L8DB10 BRANCH IF NO ECHO OR ERROR
L8D23 #E 1D 02 LDX #HCFBNBF+1 POINT TO CASS FILE NAME BUFFER
L8D24 BD #B L8DB14 JSR LE84 OUTPUT IT TO RS 232 PORT
L8D28 BC #1 DA CXPSX C8BFAD SAVE TO END OF BUFFER
L8D2E 26 #F BNE L8DB26 LOOP UNTIL DONE
L8D30 BD #3 BNE L8DB26 OUTPUT CHECK BYTE AND LOOK FOR ACKNOWLEDGE
L8D32 26 #7 JSR L8DB18 TRANSMIT NAME AGAIN IF NO ACKNOWLEDGE
L8D34 BD #0 C3 BSR L8DB72 GET FILE TYPE EFF <> NOT FOUND
L8D36 26 E3 BNE L8DB18 BRANCH IF ERROR
L8D38 A7 #2 STA ,#2,Y SAVE FILE TYPE
L8D3A BD #6 BSR L8DB72 READ ASCII FLAG
L8D3C BD #D BNE L8DB18 BRANCH IF ERROR

817
**DISASSEMBLY OF EXTENDED BASIC 1.1**

1633 BD3E A7 23 STA $B3,Y SAVE ASCII FLAG
1634 BD48 BD 29 BSR LB06B READ CHECK BYTE FROM SENDER
1635 BD42 26 07 BNE LB01B BRANCH IF NO CHECKBYTE MATCH
1636 BD44 32 62 LDA $25,P PURGE ATTEMPT COUNTER & CHECK BYTE FROM STACK
1637 BD46 35 86 PULS A,PC RETURN FILE STATUS IN ACCD
1638 * INCREMENT ATTEMPT COUNTER AFTER 5 TRIES, GIVE UP (0 ERROR)
1639 BD48 6C A4 LB04B INC,Y INCREMENT ATTEMPT COUNTER
1640 BD4A A4 A6 LDA,Y GET ATTEMPT COUNTER
1641 BD4C B1 85 OMPA $36 IS THIS THE FIFTH TRY?
1642 BD4E 25 1A BLD LB06A NO
1643 BD58 B6 8C LDA $3BC YES; TIME TO QUIT-GET ABORT CODE
1644 BD5D B2 9C JMP L8EBRC OUTPUT ABORT CODE OVER THE RS 232 PORT
1645 BD55 7E A6 19 JMP L6A91 10 ERROR
1646 * ECHO CHECK - OUTPUT A CHARACTER, READ A CHARACTER AND
1647 * COMPARE IT TO THE OUTPUT CHARACTER. Z=0 IF NO MATCH OR ERROR
1648 BD58 34 B2 LB058 PSHS A SAVE COMPARISON ON STACK
1649 BD5A B0 5C BSR LB0BB SEND A CHARACTER OUT
1650 BD5C 26 82 BNE LB04A BRANCH IF READ ERROR
1651 BD5E A1 E4 OMPA .S COMPAIR RECEIVED CHARACTER TO TRANSMITTED CHARACTER
1652 BD58 35 B2 L8DB8 PULS A,PC RESTORE COMPARISON CHARACTER AND RETURN
1653 * TRANSMIT XOR CHECKBYTE AND READ ACKNOWLEGE ($CB)
1654 * RETURN ZERO FLAG SET IF NO ERROR AND ACKNOWLEGE
1655 BD59 6D A1 LB062 LDA $11,Y GET XOR CHECKBYTE
1656 BD64 B0 52 BSR LB0BB OUTPUT XOR CHECKBYTE AND READ ONE BYTE
1657 BD66 26 92 BNE LB06A BRANCH IF READ ERROR
1658 BD68 B1 C8 OMPA $3C COMPAIR INPUT BYTE TO ACKNOWLEGE CODE
1659 BD6A 39 LB06A RTS
1660 * READ XOR CHECKBYTE THEN LOAD ACCUMULATED XOR CHECKBYTE.
1661 * SET ZERO FLAG IF ACCUMULATED CHECK BYTE = 0
1662 BD6B 8D 55 L8DB8 BSR LB072 INPUT A CHARACTER FROM RS 232
1663 BD6D 26 82 BNE LB06A BRANCH IF TIMEOUT
1664 BD6F 6D 21 LDA $11,Y GET CHECK BYTE
1665 BD71 39 RTS
1666 BD72 B0 48 L8DB7 BSR LB0BC INPUT A CHARACTER FROM RS 232
1667 BD74 34 3D L8DB8 PSHS A,CC SAVE CHARACTER AND ZERO FLAG ON STACK
1668 BD76 A8 21 * EXCLUSIVE OR INPUT
1669 BD78 A7 81 L8DB8 STA $11,Y * CHARACTER WITH CHECK BYTE
1670 BD7A 35 83 PULS CC,A,PC RESTORE CHARACTER AND ZERO FLAG
1671 * REQUEST A BLOCK FROM RS 232 INPUT -
1672 * LOAD THE RECEIVED DATA INTO THE BUFFER POINTED TO BY X
1673 * IF REGIFIER CONTAINS THE BLOCK NUMBER; RETURN Z=1 IF NO
1674 * ERRORS, CHARACTER COUNT IN ACCA; ACCA $0 IF FILE EMPTY
1675 BD7C 4F L8DB7 CLRA RESET ATTEMPT COUNTER
1676 BD7D 34 7E PSHS U,Y,X,A,A SAVE SPACE FOR STACK BUFFER
1677 BD7F 68 67 ASL $87,S * PUT 2 Bytes (U) CONTAIN THE 16 BIT BLOCK NUMBER -
1678 BD81 69 66 ROL $96,S * PUT THE TOP 7 Bits IN 7,5 AND THE
1679 BD83 64 67 LSR $97,S * TOP SEVEN BITS IN 6,5
1680 BD85 31 4E LEAY .S STACK POINTER TO Y (TFR S,Y)
1681 BD87 28 82 BRA LB0BB
1682 BD89 B0 8D BSR LB04B INCREMENT ATTEMPT COUNTER
1683 BD8B 86 97 L8DBB LDA #$97 * TRANSMIT A BLOCK REQUEST CODE, ECHO
1684 BD8B B0 99 BSR LB058 * CHECK AND RESET CHECK BYTE
1685 BD8F 26 2F BNE LB089 BRANCH IF NO MATCH OR ERROR
1686 BD91 A6 26 LDA $98,Y * SEND OUT HIGH ORDER SEVEN BITS
1687 BD93 B0 4F BSR LB04A * OF BLOCK NUMBER
1688 BD95 A6 27 LDA $107,Y = SEND OUT LOW ORDER SEVEN BITS
1689 BD97 B0 6B BSR LB04B = OF BLOCK NUMBER
1690 BD99 B0 7C BSR LB062 TRANSMIT CHECK BYTE AND GET ACKNOWLEDGE
1691 BD9B 26 EC BNE LB089 BRANCH IF ERROR OR NO ACKNOWLEDGE
1692 BD9D B0 83 BSR LB072 READ CHARACTER COUNT
1693 BD9F 68 2B BNE LB04A BRANCH IF READ ERROR
1694 BD9A A7 24 STA $78,Y SAVE CHARACTER COUNT IN STACK VARIABLES
1695 BD9A AE 22 L8DB8 PULS $78,Y GET VARIABLES POINTER FROM STACK BUFFER
1696 * READ IN A BLOCK OF 128 CHARACTERS - THE HOST WILL TRANSMIT 128
1697 * CHARACTERS REGARDLESS OF HOW MANY ARE VALID, OF HOW MANY ARE VALID.
1698 BDAC C6 8E L8DB9 BSR LB089 128 CHARACTERS/BUFFER
1699 BDAD BD 9C L8D97 BSR LB072 READ A CHARACTER
1700 BDAD 26 0E BNE LB089 RESTART PROCESS IF READ ERROR
1701 BDAD A7 B8 STA .X+ SAVE THE CHARACTER IN BUFFER
1702 BDAD 5A DECB DECREMENT CHARACTER COUNTER
1703 BDAD 46 F7 BNE LB04A BRANCH IF NOT DONE
1704 BDAD B0 8D BSR LB06B INPUT XOR CHECKBYTE
1705 BDAD 26 05 BNE LB089 RESTART PROCESS IF READ ERROR OR BAD CHECKBYTE
1706 BDAD 32 64 L8DB8 PSHS $48,S PURGE ATTEMPT COUNTER, CHECK BYTE AND LOAD ADDRESS FROM STACK
1707 BDAD 35 96 PULS A,R,X,PC RETURN CHARACTER COUNT IN ACCA
1708 BDAD 6F 21 L8DB8 CLRA CLEAR CHECK BYTE
1709 BDAD B0 58 BSR LB04A OUTPUT A CHARACTER OVER RS 232 PORT
1710 $ READ A CHARACTER FROM THE RS 232 INPUT PORT.
1711 * RETURN CHARACTER TO ACCA. EXIT WITH Z=0
1712 * FOR TIMEOUT ERROR, Z = 1 FOR VALID BYTE INPUT.
1713 BDCE 4F L8DB8 CLRA CLEAR ATTEMPT COUNTER
1714 BDCE 34 15 PSHS X,R,CC SAVE REGISTERS AND INTERRUPT STATUS
1715 BDCE 1A 58 ORCC #$58 DISABLE INTERRUPTS
1716 BDCE 96 E7 LDA TIMOUT GET TIMEOUT VARIABLE DELAY
1717 BDCE 93 9A LDX ZERO X=0; TIMEOUT CONSTANT DELAY
1718 BDCC 8D 0F L8DC5 BSR LB06E GO GET RS 232 STATUS
1719 BDCC 24 FC L8DC7 OUT IF SPACEING
1720 BDCC 8D 0B L8DC9 BSR LB06E GET RS 232 STATUS
1721 BDCC 25 FC BLD LB099 LOOP IF MARKING
1722 BDCC 8D 0A BSR LB04A DELAY 1/2 BIT TIME
1723 BDCC 6C 81 L8DB8 LDA $81 * GET BIT SHIFT COUNTER AND BIT
1724 BDCC 34 84 PSHS B * MASK AND SAVE IT ON STACK
1725 BDCC 3F 6C CLR INPUT DATA BYTE
1726 BDDE B0 21 L8DD4 BSR LB04F GO DELAY ONE BIT TIME
1727 BDDE F6 22 L8DB8 LDA $1A+2 * RS 232 INPUT TO
1728 BDDE 56 8D DBB * CARRY FLAG

818
DISASSEMBLY OF EXTENDED BASIC 1.1

1787 B0DA 24 #2
    BCC L8DE
    BRANCH IF RS 232 INPUT = 0 (SPACE/RG)

1788 B0DC AA E4
    ORA .S
    IF MARKING, OR A 1 BIT INTO DATA BYTE

1789 B0DE 68 E4
    LBDE ASL .S
    SHIFT BIT COUNTER ONE BIT TO LEFT

1790 B0DE 24 F2
    BRK LBDE
    CARRY WILL BE SET AFTER 8 SHIFTS

1791 B0DE 32 61
    LEAS $81,S
    PULL BIT COUNTER OFF THE STACK

1792 B0DE 35 95
    PULS CC,B,A,PC
    RESTORE INTERRUPT STATUS & RETURN

1793 B0DF 00

* PUT RS 232 STATUS INTO THE CARRY FLAG AND CHECK FOR TIMEOUT

1794 B0DF 06 FF 22
    LBDEE LBD P81+2
    * RS 232 INPUT TO CALL DELAY ROUTINE

1795 B0DF 56
    ROBAS
    * CARRY FLAG

1796 B0DE 3B #1
    LEAX $81,X
    = INCREMENT CONSTANT TIMEOUT

1797 B0DE 26 #0
    BNE LBDFG
    = DELAY, RETURN IF <> 0

1798 B0DE 44
    DECA
    = DECREMENT VARIABLE TIMEOUT

1799 B0DE 26 #5
    BNE LBDF6
    = DELAY: RETURN IF <> 0

1800 B0DF 32 62
    LEAS $82,S
    PURGE RETURN ADDRESS OFF STACK

1801 B0DF 35 15
    PULS CC,B,A
    CLEAN UP STACK/RESTORE INTERRUPTS

1802 B0DF 4C
    INC
    SET ACCA = 1; ZERO FLAG = 0

1803 B0DF 39
    LBDF6
    RTS

* DELAY LOOP -- COUNTER DOWN DLBAUD

1804 B0DF 74 00
    LBDF7 BS $8F
    CALL DELAY ROUTINE

1805 B0DF 34 #2
    LBDF9 PSHS A
    SAVE ACCA

1806 B0DF 32 06
    LDA $DAA
    GET DLBAUD - 1/2 BIT TIME DELAY

1807 B0DF 21 1E
    LBDFD BNN LBDF0
    DUMMY INST - JUST ADD TO DELAY

1808 B0DF 44
    DECA
    DLC DELAY TIMER

1809 B0DF 26 6F
    BNE LBDF0
    NOT DONE

1810 B0EE 35 82
    PULS A,PC
    RESTORE ACCA AND RETURN

1811 B0EE 04 34
    LBE04 PSHS A
    SAVE CHARACTER ON STACK

1812 B0EE 0A 21
    ED8A $81,Y
    * E8 CHARACTER WITH 1,Y AND

1813 B0EE 0B A7 21
    STA $81,Y
    * SAVE RESULT IN 1,Y

1814 B0EE 0A 35
    PULS A
    * GET CHARACTER BACK

1815 B0EE 77

* SEND CHAR IN ACCA OUT OVER RS232 OUTPUT

1816 B0EC 34 #7
    LBEC PSHS B,A,CC
    SAVE ACCO AND INTERRUPT STATUS

1817 B0EC 0A 58
    ORCC $58
    DISABLE INTERRUPTS

1818 B0EC 08 05
    BSR LBDF7
    DELAY AWHAILE

1819 B0EC 0D 33
    BSR LBDF7
    DELAY MORE

1820 B0EC 14 7F FF 20
    CLR P81A
    SET RS232 OUTPUT TO SPACING

1821 B0EC 17 0D 0E
    BSR LBDF7
    DELAY MORE - START BIT

1822 B0EC 19 C6 00
    LBH $81
    BIT CTR - SEND 8 BITS

1823 B0EB 34 #4
    PSHS B
    SAVE BIT CTR ON STACK

1824 B0EB 0D 62
    LBED1 LDA $82,S
    GET OUTPUT BYTE

1825 B0EB 0D 63
    REIF A4 E4
    AND IT W/ THE BIT CTR

1826 B0EB 21 07
    LBE2 LB8E5
    THIS BIT IN OUTPUT BYTE = 0

1827 B0EB 23 B6 #2
    LDA $82
    OUTPUT BIT = 1; SET RS232 TO MARKING

1828 B0EB 25 B7 FF 20
    STA P81A
    SET RS232 TO VALUE IN ACCA

1829 B0EB 28 B0 DO
    BSR LBDF7
    DELAY FOR AWHAILE

1830 B0EB 2D 64
    ASL .S
    SHIFT BIT CTR

1831 B0EB 2C 24 EF
    BCC LB1E0
    WHEN CARRY SET, 8 BITS DONE

1832 B0EB 2B 32 0F
    STA $82
    WHEN DONE, SET RS232 TO MARKING

1833 B0EB 87 FF 20
    STA P81A
    SET RS232 OUTPUT

1834 B0EB 33 32 61
    LDA $83
    PULL BIT CTR OFF THE STACK

1835 B0EB 35 87
    PULS CC,B,A,PC
    RESTORE ACCO, INTERRUPTS & RETURN

1836 B0EB 78

* PROCESS EXCLAMATION POINT

1837 B0EF 87 66 #1
    LBEE LDA $81
    * SET SPACES

1838 B0EF 97 09
    STA VD9
    * COUNTER = 1

1839 B0EF 7D

* PROCESS STRING ITEM - LIST

1840 B0EF 8A 5A
    LBEE DEC
    DECREMENT FORMAT STRING LENGTH COUNTER

1841 B0EF 3C BD 08
    JSR LB8F
    SEND A "=" TO CONSOLE OUT IF VDA =0

1842 B0EF 39 A5
    JSR GETCH
    GET CURRENT INPUT CHARACTER

1843 B0EF 41 10 00 33
    LBEE LBEE
    EXIT PRINT USING IF END OF LINE

1844 B0EF 47 D7 03
    STC VD3
    SAVE REMEMBER FORMAT STRING LENGTH

1845 B0EF 47 B1 56
    JSR LB166
    EVALUATE EXPRESSION

1846 B0EF 0A 0D 16
    JRE A4 LB166
    TM ERROR IF NUMERIC VARIABLE

1847 B0EF 40 E5 52
    LDX FPAM2
    * GET ITEM - LIST DESCRIPTOR ADDRESS

1848 B0EF 4F 4D
    STC V40
    * AND SAVE IT IN V40

1849 B0EF 51 66 09
    LDB VD9
    GET SPACES COUNTER

1850 B0EF 53 BD 6D 4D
    JSR LB6AD
    PUT ACBC BYTES INTO STRING SPACE & PUT DESCRIPTOR ON STRING STACK

1851 B0EF 56 BD 89 9F
    JSR LB99F
    PRINT THE FORMATTED STRING TO CONSOLE OUT

1852 B0EF 80

* PAD FORMAT STRING WITH SPACES IF ITEM - LIST STRING = FORMAT STRING LENGTH

1853 B0EF 59 9E 52
    LDX FPAM2
    POINT X TO FORMATTED STRING DESCRIPTOR ADDRESS

1854 B0EF 58 06 09
    LDX VD9
    GET SPACES COUNTER

1855 B0EF 50 08 84
    SUBB .X
    SUBTRACT LENGTH OF FORMATTED STRING

1856 B0EF 5F 5A
    LBEEF DEC
    DECREMENT DIFFERENCE

1857 B0EF 68 1B 2B 01 4F
    LDBE LB8F3
    GO INTERPRET ANOTHER ITEM - LIST

1858 B0EF 64 BD 89 1C
    JSR LB9AC
    PAD FORMAT STRING WITH A SPACE

1859 B0EF 67 2F 6F
    LRA LBEEF
    KEEP PADDINGS

1860 B0EF 77

* PERCENT SIGN - PROCESS A %SPACES COMMAND

1861 B0EF 69 07 03
    LBE69 STB VD3
    * SAVE THE CURRENT FORMAT STRING

1862 B0EF 6B 9F 0F
    STX TEMPTR
    * COUNTER AND POINTER

1863 B0EF 6D 86 02
    LDA $82
    INITIAL SPACES COUNTER = 2

1864 B0EF 6F 97 09
    STA VD9
    SAVE IN SPACES COUNTER

1865 B0EF 71 A6 84
    LBE71 LDA .X
    GET A CHARACTER FROM FORMAT STRING

1866 B0EF 73 B1 25
    CMPA $4X
    COMPARE TO TERMINATOR CHARACTER

1867 B0EF 75 C7 6A
    BRCS LBEEB
    BRANCH IF END OF SPACES COMMAND

1868 B0EF 77 B1 0C
    CMPA $4H
    BLANK

1869 B0EF 79 26 07
    BNE LBE82
    BRANCH IF ILLEGAL CHARACTER

1870 B0EF 7B 8C 09
    INC VD9
    ADD ONE TO SPACES COUNTER

1871 B0EF 7B 30 0F
    LEAS $81,X
    MOVE FORMAT POINTER UP ONE

1872 B0EF 7B 5A
    DEC
    DECREMENT LENGTH COUNTER

1873 B0EF 88 26 EF
    BRK LBE71
    BRANCH IF NOT END OF FORMAT STRING

1874 B0EF 82 9E 0F
    LBE82 LDX TEMPTR
    * RESTORE CURRENT FORMAT STRING COUNTER

1875 B0EF 84 BD 03
    LD VD3
    * AND POINTER TO POSITION BEFORE SPACES COMMAND

1876 B0EF 86 26 25
    LDA .X
    SEND A % TO CONSOLE OUT AS A DEBUGGING AID

ORIGIN:SPECTRAL ASSOC

DISASSEMBLY OF EXTENDED BASIC 1.1

REVISI0N:10/08/1989 WALTER K ZYDNER
1825 * ERROR PROCESSOR - ILLEGAL CHARACTER OR BAD SYNTAX IN FORMAT STRING
1826 BE8B BD 0F 08 L8EBB L8F0D SEND A * TO CONSOLE OUT IF VDA <> 0
1827 BE8B BD A2 02 JSR L8A02 SEND CHARACTER TO CONSOLE OUT
1828 BE8B Z8 29 BRA L8EBB GET NEXT CHARACTER IN FORMAT STRING
1829 1830 * PRINT RAW HOOK
1831 BE9B 01 CD XECC CMPA #00 USING TOKEN
1832 BE92 27 01 BEQ L8EB9 BRANCH IF PRINT USING
1833 BEF4 39 RTS
1834 1835 * PRINT USING
1836 * VDA IS USED AS A STATUS BYTE; BIT 6: COMM FORCE
1837 * BIT 5: LEADING ASTERISK FORCE; BIT 4: FLOATING $ FORCE
1838 * BIT 3 = PRE SIGN FORCE; BIT 2 = POST SIGN FORCE; BIT 0 = EXPONENTIAL FORCE
1839 L8F95 LEA #02, V PUNGE RETURN ADDRESS OFF THE STACK
1840 BE97 BD 01 58 JSR L8158 EVALUATE FORMAT STRING
1841 BE9A BD 01 46 JSR L8146 TM ERROR IF VARIABLE TYPE = NUMERIC
1842 BE9D 06 3B LDB ' ' CHECK FOR ITEM LIST SEPARATOR
1843 BE9F BD 02 6F JSR L826F SYNTAX CHECK FOR ;
1844 BEA2 9E 52 LDS #AP-2 * GET FORMAT STRING DESCRIPTOR ADDRESS
1845 BEA4 9F 05 STX V05 AND SAVE IT IN V05
1846 BEA6 29 06 BRA L8EAE GO PROCESS FORMAT STRING
1847 BEA8 9F 07 L8EA8 LDA V07 * CHECK NEXT PRINT ITEM FLAG AND
1848 BEAA 27 08 BEQ L8EBA4 * FC ERROR IF NO FURTHER PRINT ITEMS
1849 BEAC 9E 05 LDX V05 RESET FORMAT STRING POINTER TO START OF STRING
1850 BEAE 0F 07 L8EAE CLR V07 RESET NEXT PRINT ITEM FLAG
1851 BEB8 84 84 LDB ',' GET LENGTH OF FORMAT STRING
1852 BEB2 26 23 BNE L8EBB INTERPRET FORMAT STRING IF LENGTH > 0
1853 BEB4 7E 04 4A L8EB4 JMP L84AA FC ERROR IF FORMAT STRING = NULL
1854 BEB7 8E 2D LDX $82, X POINT X TO START OF FORMAT STRING
1855 * INTERPRET THE FORMAT STRING
1856 BEB9 8F 0A L8EB9 CLR VDA CLEAR THE STATUS BYTE
1857 BEBB 8F 09 L8EBB V09 CLEAR LEFT DIGIT COUNTER
1858 BEBD 09 98 LDB X',' GET A CHARACTER FROM FORMAT STRING
1859 BEBF 81 21 CMPA '=' Exclamation Point?
1860 BEC1 27 FF 72 L8EB L8E737 YES - STRING TYPE FORMAT
1861 BEC5 81 23 CMPA #R' Number Sign? (Digit Locator)
1862 BEC7 27 5B BEQ L8F24 YES - NUMERIC TYPE FORMAT
1863 BEC9 5A DECZ DECREMENT FORMAT STRING LENGTH
1864 BECA 26 16 BNE L8EE2 BRANCH IF NOT DONE
1865 BECC BD 0F 08 JSR L8F0D SEND A * TO CONSOLE OUT IF VDA <> 0
1866 BEED 9B 02 BRA L8A02 SEND CHARACTER TO CONSOLE OUT
1867 BEED 9D 05 L8EDD JSR GETCH GET CURRENT CHARACTER FROM BASIC
1868 BEED 9E 0D BNE L8EA8 BRANCH IF NOT END OF LINE
1869 BEED 9F 07 LDA V07 GET NEXT PRINT ITEM FLAG
1870 BEED 9F 03 BNE L8EDD BRANCH IF MORE PRINT ITEMS
1871 BEED 9F 05 BNE L8EDD SEND A CARRIAGE RETURN TO CONSOLE OUT
1872 BEED 9E 05 L8EDD LDX V05 POINT X TO FORMAT STRING DESCRIPTOR
1873 BEF7 8E 59 JMP L8E59 RETURN ADDRESS AND LENGTH OF FORMAT STRING - EXIT PRINT USING
1874 BEF7 8E 5B BEQ L8EE2 CHECK FOR ++ (PRE-SIGN FORCE)
1875 BEE4 26 09 BNE L8EEF NO PLUS
1876 BEE6 BD 0F 08 JSR L8F0D SEND A * TO CONSOLE OUT IF VDA <> 0
1877 BEE7 8E 08 BRA L868 $88 * TO INDICATE THAT THE OUTPUT WILL
1878 BEEB 97 DA STA VDA * PRE-SIGN FORCE FLAG
1879 BEED 28 CC BRA L8EBB INTERPRET REST OF THE FORMAT STRING
1880 BEEF 81 2E L8EEF CMPA #', ' DECIMAL POINT?
1881 BEF1 27 4E BEQ L8F41 YES
1882 BEF3 81 25 CMPA #* PERCENT SIGN?
1883 BEF5 18 27 FF 70 L8EB L8E69 YES
1884 BEF9 81 84 CMPA, X COMPARE THE PRESENT FORMAT STRING INPUT
1885 * CHARACTER TO THE NEXT ONE IN THE STRING
1886 BEFB 26 8B L8EBF BNE L8EBB NO MATCH - ILLEGAL CHARACTER
1887 * TWO CONSECUTIVE EQUAL CHARACTERS IN FORMAT STRING
1888 BEFD 21 24 CMPA #* '$ DOLLAR SIGN?
1889 BEFF 27 19 BEQ L8F1A YES - MAKE THE DOLLAR SIGN FLOAT
1890 BF01 81 2A CMPA #* ' ASTERISK #* '
1891 BF03 26 0F BNE L8EF8 NO - ILLEGAL CHARACTER
1892 BF05 9C 0D LDA VDA * GRAB THE STATUS BYTE AND TEST BIT 5
1893 BF07 BA 2B ORA #20 * TO INDICATE THAT THE OUTPUT WILL
1894 BF09 97 DA STA VDA * BE LEFT PADDED WITH ASTERISKS
1895 BF0B 01 2B CMPA #* ' CHECK TO SEE IF THE #* ARE THE LAST TWO
1896 BF0D 25 11 BLO L8F20 = CHARACTERS IN THE FORMAT STRING AND BRANCH IF SO
1897 BF0F 9A 01 LDA $81,X GET THE NEXT CHARACTER AFTER **
1898 BF11 81 24 BRA L8EBB CHECK FOR #*'
1899 BF13 26 08 BNE L8FB0 CHECK FOR MORE CHARACTERS
1900 BF15 5A DECB DECREMENT STRING LENGTH COUNTER
1901 BF16 38 01 LDA $81,X MOVE FORMAT STRING POINTER UP ONE
1902 BF18 8C 09 INC V09 ADD ONE TO LEFT DIGIT COUNTER - FOR ASTERISK PAD AND
1903 * FLOATING DOLLAR SIGN COMBINATION
1904 BF1A 96 0A L8FA L8F1A LDA VDA * GET THE STATUS BYTE AND SET
1905 BF1C 1A 10 ORA #10 * BIT 4 TO INDICATE A
1906 BF1E 97 DA STA VDA * FLOATING DOLLAR SIGN
1907 BF20 3B 01 L8F20 LEA $81,X MOVE FORMAT STRING POINTER UP ONE
1908 BF22 8C 09 INC V09 ADD ONE TO LEFT DIGIT (FLOATING $ OR ASTERISK PAD)
1909 * PROCESS CHARACTERS TO THE LEFT OF THE DECIMAL POINT IN THE FORMAT STRING
1910 BF24 8F 08 L8FB4 CLR V09 CLEAR THE RIGHT DIGIT COUNTER
1911 BF26 8C 09 L8FB6 LDA V09 ADD ONE TO LEFT DIGIT COUNTER
1912 BF28 5A DECB DECREMENT FORMAT STRING LENGTH COUNTER
1913 BF29 27 49 BEQ L8F47 BRANCH IF END OF FORMAT STRING
1914 BF2B 46 08 LDA X ',' GET THE NEXT FORMAT CHARACTER
1915 BF30 81 2E CMPA ', ' DECIMAL POINT?
1916 BF32 27 1E BEQ L8F47 YES
1917 BF34 21 23 CMPA #* ' NUMBER SIGN?
1918 BF36 27 81 BEQ L8FB2 YES
1919 BF38 21 2C CMPA #* ' COMPAT
1920 BF3A 26 21 BNE L85A NO
DISASSEMBLY OF EXTENDED BASIC 1.1

1975 B7F6 9D A5 LBF96 JSR GETCHC GET CURRENT CHARACTER
1976 B7F8 1B 27 FF 3C LBEQ LBEB0 BRANCH IF END OF LINE
1977 B7FC 07 D3 STB VD3 SAVE FORMAT STRING LENGTH WHEN FORMAT EVALUATION ENDED
1978 B8EE BD B1 41 JSR LB141 EVALUATE EXPRESSION
1979 B8F1 96 D9 LDA VD9 GET THE LEFT DIGIT COUNTER
1980 B8FA 9B DB ASHA VDB ADD IT TO THE RIGHT DIGIT COUNTER
1981 B8FA 81 11 CMPA #17
1982 B8FA 81 1B CMPA #17
1983 B9A2 1B 22 24 9F LBHI LB44A * FC ERROR IF MORE THAN 16 DIGITS AND DECIMAL POINT
1984 > B9B8 B0 8F ES JSR LBFES CONVERT ITEM-LIST TO FORMATTED ASCII STRING
1985 B9AE 3B 1F LBF96 LEX #-.X,1 MOVE BUFFER POINTER BACK ONE
1986 B9BF BD 07 CLR VD3 RESET NEXT PRINT ITEM Flag
1987 B8B5 9D A5 JSR GETCHC GET CURRENT INPUT CHARACTER
1988 B8B7 27 0D BEQ LBFC6 BRANCH IF END OF LINE
1989 B8B9 97 D7 STA VD7 SAVE CURRENT CHARACTER (<>0) IN NEXT PRINT ITEM Flag
1990 B8BB B1 3B CMPA #17 * CHECK FOR - : ITEM-LIST SEPARATOR AND
1991 B8BD 27 05 BEQ LBFC4 * BRANCH IF SEMICOLON
1992 B8BF BD B2 6D JSR SYNTAX CHECK FOR COMMA
1993 B8E2 0B 22 0C JSR GETCHC PROCESS NEXT PRINT ITEM
1994 > B9C4 90 9F LBFCS JSR GETCHC GET NEXT INPUT CHARACTER
1995 B9CF 9E 05 LBFC6 LDX VD5 GET FORMAT STRING DESCRIBER ADDRESS
1996 B9CF 64 B4 LDA ,A GET LENGTH OF FORMAT STRING
1997 B9E7 D3 08 SUBB VB3 SUBTRACT AMOUNT OF FORMAT STRING LEFT AFTER LAST PRINT ITEM
1998 B9EE AE 02 LDX #02,X * GET FORMAT STRING START ADDRESS AND ADVANCE
1999 B9EE AE 08 JSR ADDA * POINTER TO START OF UNUSED FORMAT STRING
2000 B9FE D0 03 LD VD3 =GETAMOUNTOFUNUSEDFORMATSTRING
2001 B9FD 1B 26 FE 64 LBNE LBEB9 =REINTERPRET FORMAT STRING FROM THAT POINT
2002 B9FD 7E BE 02 JMP LBEB2 REINTERPRET FORMAT STRING FROM THE START IF ENTIRELY
2003 * USED ON LAST PRINT ITEM
2004
2005 * PRINT A + TO CONSOLE OUT IF THE STATUS BYTE <> 0
2006 B8FD 34 02 LBFDB PSHS A RESTORE ACCA AND RETURN
2007 B8FD 34 2B LDF A GET ASCII PLUS SIGN
2008 B8FD 0A 00 TST VDA * CHECK THE STATUS BYTE AND
2009 B8FD 27 03 BEQ LBFE3 RETURN IF = 0
2010 B9BE BD 02 A2 JSR LADB2 SEND A CHARACTER TO CONSOLE OUT
2011 B9BE 35 02 LBFE3 PULS A,PC RETURN ACCA AND RETURN
2012
2013 * CONVERT ITEM-LIST TO DECIMAL ASCII STRING
2014 B8FE CE 03 DB LBFES LDU #STRBUF+4 POINT 0 TO STRING BUFFER
2015 B8FE CE 08 DB LDF A GET ASCII PLUS SIGN
2016 B8FE 9A 00 LDA VD * GET THE STATUS FLAG AND
2037 B8FC 05 8B BITA #888 * CHECK FOR A PRE-SIGN FORCE
2038 BFE6 27 B2 BEQ L8BF2 * BRANCH IF NO PRE-SIGN FORCE
2039 BFF8 C6 28 LDB #*+ PLUS SIGN
203A BF20 FF 8D L8FF2 CHECK THE SIGN OF FPAB
203B BF44 2A B4 BPL L8FFA BRANCH IF POSITIVE
203C BF66 8F 54 CLR FM8GN FORCE FPAB SIGN TO BE POSITIVE
203D BFF8 C6 2D LDR VX MINUS SIGN
203E BFFA E7 C8 L8FFA STB ,UX SAVE THE SIGN IN BUFFER
203F BFFC 6C 3B LDB #*- PUT A ZERO INTO THE BUFFER
2040 BFEE E7 C8 STB ,UX *
2041 0000 B4 81 ANDA #881 = CHECK THE EXPONENTIAL FORCE FLAG IN
2042 000B 18 46 87 LTX L8L0B = THE STATUS BYTE - BRANCH IF ACTIVE
2043 0026 B6 BD 08 LDX #LBD0 POINT X TO FLOATING POINT IE + #9
2044 0030 BD BC 0A JSR L8CA8 COMPARE FPAB TO X
2045 003B BC 2B 15 JSR L8M92 EXECUTE BRANCH IF FPA < #1000
2046 0040 BD BD 09 JSR L8D09 CONVERT FP NUMBER TO ASCII STRING
2047 0051 A6 B8 L9B11 LDA ,UX * ADVANCE POINTER TO END OF
2048 005E 26 FC BNE L8M11 * ASCII STRING (ZERO BYTE)
2049 005F A6 B2 L9B15 LDA ,X = MOVE THE
2050 0067 A7 81 STA #81,X = ENTIRE STRING
2051 0079 BC 03 DA CMPX #STRBUF+3 = UP ONE
2052 0081 C6 27 F7 BNE L9N15 = BYTE
2053 0082 86 25 CXR #SPACE * INSERT A % SIGN AT START OF
2054 008B A7 B4 STA ,X * STRING - OVERFLOW ERROR
2055 0091 22 39 RTS
2056 0092
2057 0093 B3 96 4F L9023 LDA FPMPXP = GET EXPONENT OF FPAB
2058 0099 8E 47 STA VX = AND SAVE IT IN VX
2059 009F 27 23 BEQ L9N02 BRANCH IF FPAB = #0
2060 00A2 BD 91 CD JSR L91CD CONVERT FPAB TO NUMBER WITH 9 SIGNIFICANT
2061 00A3 96 47 * PLACES TO LEFT OF DECIMAL POINT
2062 00A8 96 47 L902C LDA VX GET BASE #10 EXPONENT OFFSET
2063 00AA 18 2B 00 B1 LBM X #100,000,000
2064 00A8 83 48 MEG A = CALCULATE THE NUMBER OF LEADING ZEROS TO INSERT -
2065 00B1 98 09 ASDA VX9 = SUBTRACT BASE #10 EXPONENT OFFSET AND # (FPAB HAS
2066 00B7 89 53 SBCA VX9 = 9 PLACES TO LEFT OF EXPONENT FROM LEFT DIGIT COUNTER
2067 00BC 89 EA JSR L9W4E PUT ACCA ZEROS IN STRING BUFFER
2068 00BD 9A 83 JSR L9R63 INITIALIZE DECIMAL POINT AND COMMA COUNTERS
2069 00C2 9D 92 L9B1D LDA VX2 CONVERT FPAB TO DECIMAL ASCII IN THE STRING BUFFER
206A 00D6 9E 47 LDA VX = GET BASE #10 EXPONENT AND PUT THAT MANY
206B 00DB 92 81 JSR L9Z81 = ZEROS IN STRING BUFFER - STOP AT DECIMAL POINT
206C 00E5 96 47 L9047 LDA VX = WASTED INSTRUCTION - Serves NO PURPOSE
206D 00E7 92 49 JSR L9S29 CHECK FOR DECIMAL POINT
206E 00F4 96 48 L9A68 LDA VX = GET THE RIGHT DIGIT COUNTER
206F 0101 26 82 BNE L9N05 BRANCH IF RIGHT DIGIT COUNTER < = 0
2070 010B 33 5F LEAU $-1,U = MOVE BUFFER POINTER BACK ONE - DELETE
2071 010C 83 04 * = DECIMAL POINT IF NO RIGHT DIGITS SPECIFIED
2072 010D 90 4A L9058 DECX SUBTRACT ONE (DECIMAL POINT)
2073 010E 90 8A JSR L9A9E PUT ACCA ZEROS INTO BUFFER (TRAILING ZEROS)
2074 0111 9D 85 L9B54 JSR L9B19 INSERT ASTERISK PADDING, FLOATING 8, AND POST-SIGN
2075 0117 9F 40 TSTA WAS THERE A POST-SIGN?
2076 0118 9F 86 BEQ L9M6B NO
2077 0119 9B4A CL2A CMPX #** IS THE FIRST CHARACTER AN #?
2078 011E 2C 82 BEQ L9R68 YES
2079 0120 95 E7 CXN STB ,UX = STORE THE POST-SIGN
207A 0122 06 FC L9060 CLR ,U CLEAR THE LAST CHARACTER IN THE BUFFER
207B 012C 4F * REMOVE ANY EXTRA BLANKS OR ASTERISKS FROM THE
207D 012D 4E 3A * STRING BUFFER TO THE LEFT OF THE DECIMAL POINT
207E 0130 96 3B DA L9065 LDX #STRBUF+3 POINT X TO THE START OF THE BUFFER
207F 0137 98 3B L9065 LEAX #81,X MOVE BUFFER POINTER UP ONE
2080 0138 96 7F STX TEMPR SAVE BUFFER POINTER IN TEMPR
2081 0139 96 96 L9964 VARTPR Addresses of decimal point in buffer, subtract
2082 013A 98 18 SUBA TEMPR+1 CURRENT POSITION AND SUBTRACT LEFT DIGIT COUNTER -
2083 013B 9D 09 SUBA VARD9 * THE RESULT WILL BE ZERO WHEN TEMPR+1 IS POINTING
2084 013C 82 * TO THE FIRST DIGIT OF THE RIGHT STRING
2085 013D 8F 3B BEQ L9A99 RETURN IF NO DIGITS TO LEFT OF THE DECIMAL POINT
2086 013E 96 84 LDA VX GET THE CURRENT BUFFER CHARACTER
2087 013F 8B 7B CMPX #SPACE SPACE?
2088 0140 97 22 BNE L9S65 YES - ADVANCE POINTER
2089 0141 87 1A CMPX #** ASTERISK?
208A 0142 97 8A BNE L9M65 YES - ADVANCE POINTER
208B 0143 97 4F CLRA A ZERO ON THE STACK IS END OF DATA POINTER
208C 0144 98 7C PSHS A PUSH A CHARACTER ONTO THE STACK
208D 0145 97 E6 B8 LDA VX+,X GET NEXT CHARACTER FROM BUFFER
208E 0146 99 0B 2D CMPA #+ MINUS SIGN?
208F 0147 99 02 27 BNE L9N7C YES
2090 0148 99 BC 28 CMPA #+ PLUS SIGN?
2091 0149 99 86 2F BEQ L9N7C YES
2092 014A 99 BC 38 CMPA #+ ZERO?
2093 014B 99 BB 26 BNE L9N9E NO - ERROR
2094 014C 99 02 2D LDA #81,X GET CHARACTER FOLLOWING ZERO
2095 014D 99 BD 16 JSR L9R8A CLEAR CARRY IF NUMERIC
2096 014E 99 04 25 BLD L9L9E BRANCH IF NOT A NUMERIC CHARACTER - ERROR
2097 014F 99 06 32 L9H96 PLSA A = = 32 PULL A CHARACTER OFF OF THE STACK
2098 0150 99 A7 2B STA ,X *= AND PUT IT BACK IN THE STRING BUFFER
2099 0151 99 26 FA BNE L9N9E * KEEP GOING UNTIL ZERO FLAG
209A 0152 99 0C 2B JST L9L6S KEEP CLEANING UP THE INPUT BUFFER
209B 0153 99 35 2A L9L9E PLSA A = = 32 REMOVE THE CHARACTERS ON
209C 0154 99 4A 0D TSTA = THE STACK AND EXIT WHEN
209D 0155 99 21 26 BNE L9N9E = ZERO FLAG FOUND
209E 0156 99 31 8F LDX TEMPR GET THE STRING BUFFER START POINTER
209F 0157 99 06 25 STA #X,XY = PUT A % SIGN BEFORE THE ERROR POSITION TO
20A0 0158 99 A7 2B STA ,X *= INDICATE AN ERROR
* CLEAR CARRY IF NUMERIC
2110 L9BA4 COMA #." RETURN ACCA = ASCII 0
2111 * SUBA #"+1" SUBA #"(9'+1"
2112 SUBA #"(0'+1"
2113 * CARRY CLEAR IF NUMERIC
2114 L9BB2 RTS

* PROCESS AN ITEM-LIST WHICH IS < 100,000,000
2120 L9BB3 LDA DB
2121 L9BB4 BR LC
2122 * BRANCH IF NO FORMATTED DIGITS TO THE RIGHT OF DECIMAL PT
2123 L9BB5 DECA SUBTRACT ONE FOR DECIMAL POINT
2124 L9BB6 ADDA V47 * ADD THE BASE 10 EXPONENT OFFSET - ACCA CONTAINS THE
2125 * NUMBER OF DIGITS REQUIRED TO ADJUST FPAR TO THE SPECIFIED
2126 * NUMBER OF DIGITS TO THE RIGHT OF THE DECIMAL POINT
2127 L9BB8 2B #1 BMX L9BB0 IF ACCA = 0 THEN NO SHIFTS ARE REQUIRED
2128 * SAVE INITIAL SHIFT COUNTER ON THE STACK
2129 L9BB9 2B #2 L9BB0 PHSB A * GET SHIFT COUNTER FROM THE STACK
2130 L9BBC PHSB A * EXIT ROUTINE IF POSITIVE
2131 L9BC1 2B #2 PHSB A * SAVE SHIFT COUNTER ON STACK
2132 L9BC3 BB 82 JSR L9BB2 DIVIDE FPAR BY 10 - SHIFT ONE DIGIT TO RIGHT
2133 L9BC4 3E #2 PULS A * GET SHIT COUNTER FROM THE STACK
2134 L9BC6 4C INCA * BUMP SHIFT COUNTER UP BY ONE
2135 L9BC9 2B #4 BRA L9BBF CHECK FOR FURTHER DIVISION
2136 L9BCB LDA V47 * GET BASE 10 EXPONENT OFFSET, ADD INITIAL SHIFT COUNTER
2137 L9CCD #+ #8 SUBA .5+ * AND SAVE NEW BASE 10 EXPONENT OFFSET - BECAUSE
2138 L9CF7 STA V47 * FPAR WAS SHIFTED ABOVE
2139 L9DBB #8 #9 ADDA $89 * ADD NINE (SIGNIFICANT PLACES) AND BRANCH IF THERE ARE NO
2140 L9DB3 2B #9 BMX L9BB0 = ZEROS TO THE LEFT OF THE DECIMAL POINT IN THIS PRINT ITEM
2141 L9DB5 #8 #9 SUBA $89 * DETERMINE HOW MANY ZEROS TO THE LEFT OF THE DECIMAL
2142 L9DB7 #8 #9 ADDA $89 * POINT. GET THE NUMBER OF FORMAT PLACES TO LEFT OF DECIMAL
2143 L9DB9 #8 #9 SUBA $89 * POINT. SUBTRACT THE BASE 10 EXPONENT OFFSET AND THE CONSTANT 9
2144 L9DBB #8 #0 RSR $89 * (UNNORMALIZATION), THEN OUTPUT THAT MANY ZEROS TO THE BUFFER
2145 L9DBD BB 82 JSR L9263 INITIALIZE DECIMAL POINT AND COMMA COUNTERS
2146 L9DBF 2B #10 BRA L9BBF PROCESS THE REMAINDER OF THE PRINT ITEM

SAVE ZERO COUNTER
2150 L9DE4 3E #3 L9BB2 PHSB A * INSERT A ZERO INTO
2151 L9DE6 AB #0 STA .0+ * THE BUFFER
2152 L9DEB 3E #2 PULS A * RESTORE ZERO COUNTER

SAVE ACCA ASCII ZEROS INTO THE BUFFER
2155 L9DEA 4A L9BB0 DECA DECIMAL ZERO COUNTER
2156 L9DEB 2A F5 BPL L9BB2 BRANCH IF NOT DONE
2157 L9DE9 3D RTS

* GET THE LEFT DIGIT COUNTER AND PUT
2160 L9EE6 96 #9 L9BB0 LDA V09 * THAT MANY ZEROS IN THE STRING BUFFER
2161 L9EEF 80 #8 JSR L924D PUT THE DECIMAL POINT IN THE STRING BUFFER
2162 L9EF5 86 #7 LDA #9 * DETERMINE HOW MANY FILLER ZEROS BETWEEN THE DECIMAL POINT
2163 L9EF7 88 #8 SUBA V47 * AND SIGNIFICANT DATA. SUBTRACT BASE 10 EXPONENT FROM -9
2164 L9E9F BB 82 JSR L9BB0 * (UNNORMALIZATION) AND OUTPUT THAT MANY ZEROS TO BUFFER
2165 L9EBB #8 #9 CLR V45 CLEAR THE DECIMAL POINT COUNTER - SUPPRESS THE DECIMAL POINT
2166 L9EBD #8 #7 CLR V07 CLEAR THE COMMA COUNTER - SUPPRESS COMMA
2167 L9EBF #8 #3 L9BBF JSR L92B2 DECODE FPAR INTO A DECIMAL ASCII STRING
2168 L9EC0 96 #0 L9DE6 GET THE RIGHT DIGIT COUNTER
2169 L9EC4 26 #2 BNE L91B9 BRANCH IF RIGHT DIGIT COUNTER > 0
2170 L9EC6 3E #9 LDU V90P7R * RESET BUFFER PTR TO THE DECIMAL POINT IF NO DIGITS TO RIGHT
2171 L9ECB 98 #4 L90BB ADDA V47 * ADD BASE 10 EXPONENT - A POSITIVE ACCA WILL CAUSE THAT MANY
2172 L9ED4 #8 0, S+ * FILLER ZEROS TO BE OUTPUT TO THE RIGHT OF LAST SIGNIFICANT DATA
2173 L9EED 47 #6 L90BB PHSB A * SIGNIFICANT DATA
2174 L9F0A 16 FF 43 LBB90 DECA INSERT LEADING ASTERISKS, FLOATING DOLLAR SIGN, ETC
2175
* FORCE THE NUMERIC OUTPUT FORMAT TO BE EXPONENTIAL FORMAT
2177 L9F0D 96 #F L91B0 LDA FPMEXP * GET EXPONENT OF FPAR AND
2178 L9F10 3E #2 PHSB A * SAVE IT ON THE STACK
2179 L9F11 27 #3 BEQ L9316 BRANCH IF FPAR = 0
2180 L9F13 BB 91 CD JSR L91C0 * CONVERT FPAR INTO A NUMBER WITH 9 SIGNIFICANT
2181 L9F15 86 #3 * DIGITS TO THE LEFT OF THE DECIMAL POINT
2182 L9F18 #D #B L9116 LDA VDB * GET THE RIGHT DIGIT COUNTER
2183 L9F1B 3E #1 L9117 DECA SUBTRACT ONE FOR DECIMAL POINT
2184 L9F1F 86 #9 L911B ADDA V09 ADD TO THE LEFT DIGIT COUNTER
2185 L9F20 7F #3 OA CLR STDBUF+3 CLEAR BUFFER BYTE AS TEMPORARY STORAGE LOCATION
2186 L9F25 90 #0 LDB VDA * GET THE STATUS BYTE FOR A
2187 L9F29 #4 #0 ANDS #54 * POST-BYTE FORCE; BRANCH IF
2188 L9F2E 26 #3 BNE L9129 * A POST-BYTE FORCE
2189 L9F2E #F #0 COM STDBUF+3 TOGGLE BUFFER BYTE TO -1 IF NO POST-BYTE FORCE
2190 L9F2E 85 #3 #9 SUBA $89 * SUBTRACT 9 (DUE TO THE CONVERSION TO 9
2191 L9F2E #8 #9 *SIGNIFICANT DIGITS TO LEFT OF DECIMAL POINT
2192 L9F2E #4 #0 PHSB A * SAVE SHIFT COUNTER ON THE STACK - ACCA CONTAINS THE NUMBER
2193 L9F2F #0 #0 JSR L9BB2 DIVIDE FPAR BY 10 - SHIFT RIGHT ONE
2194 L9F2F 80 #2 PULS A * RESTORE THE SHIFT COUNTER
2195 L9F33 4C INCA * ADD 1 TO SHIFT COUNTER
2196 L9F34 2B F4 BRA L913B CHECK FOR FURTHER SHIFTING (DIVISION)
2197 L9F34 34 #2 L913C A6 #4 L913C LDA L7,5 * GET THE INITIAL VALUE OF THE SHIFT COUNTER
2198 L9F37 2B #1 BMX L9141 * AND BRANCH IF SHIFTING HAS TAKEN PLACE
2199 L9F37 4C INCA * ADD 1 TO SHIFT COUNTER
2200 L9F37 4B DECA * DECREMENT NUMBER OF SHIFTS
2201 L9F37 4F BPL L9141 * NO MORE SHIFTS WHEN ACCA = 0
2202 L9F37 4F BRA L913B * SAVE SHIFT COUNTER
2203 L9F37 4F JSR L9BB2 * BRANCH IF SHIFTING HAS TAKEN PLACE
2204 L9F37 4F INCA * ADD 1 TO SHIFT COUNTER
2205 L9F37 4F BRA L913B * CHECK FOR FURTHER SHIFTING (DIVISION)
2206 L9F37 4F JSR L9BB2 * BRANCH IF SHIFTING HAS TAKEN PLACE
2207 L9F37 4F DECA * DECREMENT NUMBER OF SHIFTS
2208 L9F37 4F BPL L9141 * NO MORE SHIFTS WHEN ACCA = 0
2209 L9F37 4F BRA L913B * SAVE SHIFT COUNTER
2210 L9F37 4F JSR L9BB2 * BRANCH IF SHIFTING HAS TAKEN PLACE
2211 L9F37 4F INCA * ADD 1 TO SHIFT COUNTER
2212 L9F37 4F BRA L913B * CHECK FOR FURTHER SHIFTING (DIVISION)
2213 L9F37 4F JSR L9BB2 * BRANCH IF SHIFTING HAS TAKEN PLACE
2214 L9F37 4F DECA * DECREMENT NUMBER OF SHIFTS
2215 L9F37 4F BPL L9141 * NO MORE SHIFTS WHEN ACCA = 0
2216 L9F37 4F BRA L913B * SAVE SHIFT COUNTER
2217 L9F37 4F JSR L9BB2 * BRANCH IF SHIFTING HAS TAKEN PLACE
2218 L9F37 4F INCA * ADD 1 TO SHIFT COUNTER
2219 L9F37 4F BRA L913B * CHECK FOR FURTHER SHIFTING (DIVISION)
2220 L9F37 4F JSR L9BB2 * BRANCH IF SHIFTING HAS TAKEN PLACE
2221 L9F37 4F DECA * DECREMENT NUMBER OF SHIFTS
2222 L9F37 4F BPL L9141 * NO MORE SHIFTS WHEN ACCA = 0
2223 L9F37 4F BRA L913B * SAVE SHIFT COUNTER

* INSERT ASTERISK PADDING, FLOATING $ AND PRE-SIGN

DISASSEMBLY OF EXTENDED BASIC 1.1

1985 5E 6D 3B 2F 6A 0B LDI #STRBUF+4,= POINT X TO START OF PRINT ITEM BUFFER
1986 6A 64 B2 LDO ,X = GET SIGN BYTE OF ITEM-LIST BUFFER
1987 6A 34 B4 PHS B = AND SAVE IT ON THE STACK
1988 6B EC 62 DB CMPB #SPACE = DEFAULT PAD WITH BLANKS
1989 5B EC 62 DA CMPB #00 = GET STATUS BYTE AND CHECK FOR
1990 5B EB 62 08 RITB #00 = ASTERISK LEFT PADDING
1991 5B EC 34 5B PULS B = GET SIGN BYTE AGAIN
1992 5B 27 6B BEQ L919E = BRANCH IF NO PADDING
1993 5B 96 62 3A CMPB #00 = PAD WITH ASTERISK
1994 5B 9C 2B CMPB #SPACE = WAS THE FIRST BYTE A BLANK (POSITIVE)?
1995 5B 9A 26 5B CMPB #00 = NO
1996 5C 5B 69 DB STB ,A,B = TRANSFER PAD CHARACTER TO ACCB
1997 5B 34 5B L918E PHS B = SAVE FIRST CHARACTER ON STACK
1998 5B 2A 6B 1A LDI #8,ST,= POINT X TO START OF PRINT ITEM BUFFER
1999 5B 2A 6B 64 LDO ,X = GET NEXT CHARACTER IN BUFFER
2000 5B 2A 6A 1B BEQ L9186 = INSERT A ZERO IF END OF BUFFER
2001 5B 2A 6C 45 CMPB #E,= * CHECK FOR AN E AND
2002 5B 2A 6B 2C BEQ L9186 = * PUT A ZERO BEFORE IT
2003 5B 2A 6C 3C CMPB #00 = REPLACE LEADING ZEROS WITH
2004 5B 2A 6C 27 FC CMPB #', = PAD CHARACTERS
2005 5B 2A 6C 1C CMPB #00 = REPLACE LEADING COMMAS
2006 5B 2A 6A 2E BEQ L919B = * WITH PAD CHARACTERS
2007 5B 2A 6C 1E CMPB #00 = FOR DECIMAL POINT
2008 5B 2A 6B 24 BEQ L918A = AND DON T PUT A ZERO BEFORE IT
2009 5B 2A 6B 1B 1B LDA #00 = REPLACE PREVIOUS CHARACTER
2010 5B 2A 6B 82 STA ,X = * WITH A ZERO
2011 5B 2A 6A 6A LDA VDA = GET STATUS BYTE, CHECK
2012 5B 2A 6C 5B BITA #10 = FOR FLOATING $ $ = ORIGIN:SPECTRAL ASSOC
2013 5B 2A 6B 27 BEQ L914C = BRANCH IF NO FLOATING $ $ = DISASSEMBLY OF EXTENDED BASIC 1.1
2014 5B 2A 6C 64 CMPB #00 = STORE A 0 IN
2015 5B 2A 6C 7E STM ,X = BUFFER
2016 5B 2A 6C 84 L91C4 CMPB #00 = CHECK PRE-SIGN FLAG
2017 5B 2A 6C 8B PULS B = GET SIGN CHARACTER
2018 5B 2A 6C 5B LDA !91CC = RETURN IF POST-SIGN REQUIRED
2019 5B 2A 6C 7C STA ,X = STORE FIRST CHARACTER
2020 5B 2A 6C 3D L91CC RTS
* IS HOW THIS ROUTINE KNOWS THAT A 'SUBTRACTION' IS OCCURRING.

* THE SAME AS SUBTRACTING A POSITIVE ONE AND BIT 7 OF ACCB

* IS NEGATIVE. WHEN YOU ADD A NEGATIVE MANTISSA, IT IS

* THE SAME AS SUBTRACTING A POSITIVE ONE AND BIT 7 OF ACCB

* IS HOW THIS ROUTINE KNOWS THAT A SUBTRACTION IS OCCURRING.

CHECK FOR COMMA INSERTION

ADD ONE TO DIGIT COUNTER

ROTATE CARRY INTO BIT 7

* SET OVERFLOW FLAG - BRANCH IF CARRY SET AND

* ADDING MANTISSA OR CARRY CLEAR AND SUBTRACTING MANTISSA

* BRANCH IF SUBTRACTING MANTISSA

* TAKE THE 9'S COMPLEMENT

* IF ADDING MANTISSA

ADD IN ASCII OFFSET

MOVE TO NEXT POWER OF 10 MANTISSA

SAVE DIGIT IN ACCA

* IF ADDING MANTISSA

BRANCH IF SUBTRACTING MANTISSA

CONVERT REMAINDER INTO A NUMBER FROM 1-3

= CLEAR COMMA COUNTER - NOW IT WILL TAKE 255

= DECREMENTS BEFORE ANOTHER COMM WILL BE INSERTED

DECIMAL POINT COUNTER AND CHECK FOR COMMA INSERTION

DECREMENT DECIMAL POINT COUNTER

NOT TIME FOR DECIMAL POINT

SAVE BUFFER POINTER-POSITION OF THE DECIMAL POINT

STORE DIGIT IN BUFFER

POINT IN THE OUTPUT BUFFER

= CLEAR COMMA COUNTER - NOW IT WILL TAKE 255

DECREMENTS BEFORE ANOTHER COMMA WILL BE INSERTED

DECIMAL POINT COUNTER AND COMMA COUNTERS

GET THE BASE 10 EXPONENT OFFSET

* ADD 10 (FPAB 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 REMINDER IN ACCA

= REMAINING INTO A NUMBER FROM 1-3

= CLEAR COMMA COUNTER

= CLEAR COMMA COUNTER - 55 DIGITS OUTPUT BEFORE A COMMA

BRANCH IF COMMA FLAG ACTIVE

BRANCH IF COMMA COUNTER TO 3; THREE

= PUT A COMMA INTO

= THE BUFFER

* INITIALIZE DECIMAL POINT AND COMMA COUNTERS

* INSERT ACCA ZEROS INTO THE BUFFER

* INSERT ACCA ZEROS INTO THE BUFFER

SAVE ZEROS COUNTER

CHECK FOR DECIMAL POINT

RESTORE ZEROS COUNTER

DECREMENT ZEROS COUNTER AND

* RETURN IF < 0

= SAVE ZEROS COUNTER

= PUT A ZERO INTO

= THE BUFFER

= RESTORE THE ZEROS COUNTER

= THE BUFFER

= RESTORE BUFFER POINTER AND RETURN
2401 ****** GRAPHICS PACKAGE ******
2402
2403 * GET THE ADDRESS OF THE ROUTINE WHICH
2404 * WILL CONVERT HOR & VER COORDINATES INTO
2405 * AN ABSOLUTE RAM ADDRESS AND PIXEL MASK
2406 * DEPENDING UPON THE CURRENT PMODE AND
2407 * RETURN THE ADDRESS IN U.
2408 *
2409 2410 92BF CE 92 9C L92BF LDU #L929C JUMP TABLE ADDRESS TO U
2411 2412 9292 96 B6 LDA PMODE GET PMODE VALUE
2412 2413 9294 48 ASLA MUL ACCA X2 - 2 BYTES PER ADDRESS
2413 2414 9295 EE C6 LDU A,U GET JUMP ADDRESS
2414 2415 9297 39 RTS
2416 *
2417 * CONVERT VER COORD (VERBEG) & NOR COORD (HORBEG) INTO
2418 * ABSOLUTE SCREEN ADDR IN X AND PIXEL MASK IN ACCA.
2419 2418 929B BD F5 L929B ESR L92BF GO GET JUMP ADDRESS
2419 2420 929A 6E C4 JMP ,U GO TO IT
2420 *
2421 * JUMP TABLE -- HOR, VER COORD CONVERSION
2422 2421 929C 92 A6 L929C FDB L92A6 PMODE Ø
2422 2422 929E 92 C2 L929E FDB L92C2 PMODE 1
2422 2423 929A 92 A6 L92A9 FDB L92A6 PMODE 2
2423 2424 92A2 92 C2 L92A2 FDB L92C2 PMODE 3
2424 2425 92A4 92 A6 L92A4 FDB L92A6 PMODE 4
2425 *
2426 * HOR, VER COORD CONVERSION ROUTINE FOR 2
2427 *
2428 * COLOR HIRES GRAPHICS MODES
2429 2428 92A6 34 44 L92A6 PSHS U,B SAVE REGISTERS
2430 2429 92A8 66 B9 LDB HOBGY GET NUMBER BYTES/HOR GRAPHIC ROW
2430 2430 92AA 96 C8 LDA VERBEG+1 GET VERTICAL COORDINATE
2431 2431 92AC D3 D0 MUL CALCULATE VERTICAL BYTE OFFSET
2432 2432 92AD 92A0 ADD BEGGRP ADD IN START OF GRAPHIC PAGE
2433 2433 92AF 1F 91 TFR D,X SAVE TEMP VALUE IN X REG
2434 2434 92BF 9E 46 LDB HOBEG+1 GET HORIZONTAL COORDINATE
2435 2435 92BE 54 LSRR * THREE LSRRS EQUALS DIVIDE BY B -
2436 2436 92B4 54 LSRR * IN THE TWO COLOR MODE THERE ARE
2437 2437 92B5 54 LSRR * 8 PIXELS/BYTE
2438 2438 92B6 3A ABX ADD HOR BYTE OFFSET
2439 2439 92B7 9E BE LDA HOBEG+1 GET HORIZONTAL COORDINATE
2440 2440 92BA 92 A7 ANDA #87 *KEEP ONLY BITS 0,1 WHICH ContAIN THE NUMBER
2441 2441 92BB CE 92 DO LDU #L92D0 POINT U TO MASK LOOKUP TABLE
2442 2442 92BE A6 C6 LDA A,U *GET PIXEL MASK - THE MASK WILL HAVE ONE BIT SET WHICH
2443 2443 92BC 35 C4 PULS B,U,PC RESTORE REGISTERS
2444 *
2445 * HOR, VER COORD CONVERSION ROUTINE FOR 4
2446 *
2447 * FOR 4 COLOR HI RES GRAPHICS MODES
2448 2448 92C2 34 44 L92C2 PSHS U,B SAVE REGISTERS
2449 2449 92C4 D6 B9 LDB HOBGY GET NUMBER BYTES/HOR GRAPHIC ROW
2450 2450 92C2 96 C8 LDA VERBEG+1 GET VERTICAL COORDINATE
2451 2451 92C4 D3 D0 MUL CALCULATE VERTICAL BYTE OFFSET
2452 2452 92C8 92A0 ADD BEGGRP ADD IN START OF GRAPHIC PAGE
2453 2453 92CB 1F 91 TFR D,X SAVE IN X REGISTER
2454 2454 92CD 9E 46 LDB HOBEG+1 GET HORIZONTAL COORDINATE
2455 2455 92CF 54 LSRR TWO LSRRS = DIVIDE BY 4, IN THE 4
2456 2456 92D0 54 LSRR COLOR MODE THERE ARE 4 PIXELS/BYTE
2457 2457 92D1 3A ABX ADD HORIZONTAL BYTE OFFSET
2458 2458 92D2 9E BE LDA HOBEG+1 GET HORIZONTAL COORDINATE
2459 2459 92D4 84 #87 ANDA #87 *KEEP ONLY BITS 0,1 WHICH ContAIN THE NUMBER OF THE PIXEL
2460 2460 92D6 CE 92 ES LDU #L92ES POINT U TO MASK LOOKUP TABLE
2461 2461 92D9 A6 C6 LDA A,U GET THE MASK FOR THE PROPER PIXEL
2462 2462 92DB 35 C4 PULS B,U,PC RESTORE REGISTERS AND RETURN
2463 *
2464 * 2 COLOR MODE PIXEL MASKS
2465 2465 92D0 #8 4B 20 00 00 04 92D0 FCB $88,440,428,830,508,558
2466 2466 92E3 #2 #1 FCB $82,411
2467 *
2468 * 4 COLOR MODE PIXEL MASKS
2469 2469 92E5 C8 3B 3C 93 L92ES FCB $88,440,428,830,508,558
2470 *
2471 *
2472 * 4 COLOR MODE PIXEL MASKS
2473 2473 92E7 3B 0C #3 L92ES FCB $88,440,428,830,508,558
2474 *
2475 * MOVE X REG DOWN ONE GRAPHIC ROW
2476 2476 92E9 D6 B9 L92E9 LDB HOBGY GET NUMBER BYTES/HOR ROW
2477 2477 92EB 3A ABX ADD TO ABSOLUTE SCREEN POSITION
2478 2478 92EC 39 RTS
2479 *
2480 * ENTER W/ABSOLUTE SCREEN POSITION IN X, THE PIXEL
2481 * MASK IN ACCA - ADJUST X AND ACCA TO THE NEXT
2482 * PIXEL TO THE RIGHT IN THE TWO COLOR MODE.
2483 2483 92ED 44 L92ED LSRA SHIFT ONE BIT TO RIGHT
2484 2484 92EE 24 #3 BCC L92F3 BRANCH IF IN SAME BYTE
2485 2485 92EF 46 L92E5 IF YOU HAVE MOVED TO NEXT BYTE, SET BIT 7 IN ACCA
2486 2486 92F1 3B #1 LEAX $81,X AND ADD ONE TO X.
2487 2487 92F3 39 L92F3 RTS
2488 *
2489 *
2490 *
2491 * MOVE ABSOLUTE SCREEN ADDRESS OF CURRENT
2492 * HOR, VER COORD ONE TO RIGHT AND ADJUST
2493 * THE PIXEL MASK FOR THE 4 COLOR MODE
2494 2494 92F4 44 L92F4 LSRA SHIFT MASK ONE BIT TO RIGHT
2495 2495 92F5 24 #6 BCC L92ED SHIFT RIGHT AGAIN IF SAME BYTE
2496 2496 92F7 6B #C LDA #CC SET PIXEL #2 IF NEW BYTE
2497 2497 92F9 3B #1 LEAX $81,X ADD ONE TO ABS SCREEN POSITION
* EVALUATE TWO EXPRESSIONS - PUT THE FIRST
* VALUE (HOR COORD) IN HORBEG AND THE
* SECOND (VER COORD) IN VERBEG.

*RETURN WITH THE 1ST VALUE IN BINVAL AND THE 2ND IN ACCB

*BYTE TO PSET/PRESET - RETURN ADDRESS IN X - THE MASK
*WILL BE ONE BYTE WITH ALL PIXELS SET TO THAT COLOR

=SHIFT RIGHT
BRANCH IF PMODE 0,2,4 (2 COLOR)

NORMALIZE EXPRESSIONS FOR PROPER PMODE

SUMMATION FOR PROPER PMODE

GET PHODE

GO GET HOR & VER COORDINATES

FORCE VER COORD TO 191
NO

* POINT

* TURN ON THE PIXEL (POINTED TO BY X, PIXEL MASK IN ACCA) TO THE COLOR

*BYTE TO PSET/PRESET - RETURN ADDRESS IN X - THE MASK
*WILL BE ONE BYTE WITH ALL PIXELS SET TO THAT COLOR

*SCREEN - ABS POSIT IN X, MASK IN ACCA, COLOR IN ALCOL

* ORIGIN:SPECTRAL ASSOC
* EVALUATE TWO SETS OF COORDINATES SEPARATED BY A MINUS
* PUT 1ST SET OF COORDS AT (HORBEG,VERBEG), SECOND
* SET AT (HOREND,VEREND), IF NOTHING BEFORE MINUS SIGN
* (HORDEF,VERDEF) AT (HORBEG, VERBEG)

LDX HORDEF
GET LAST HORIZ END POINT

STX HORBEG
PUT IN START POINT STORAGE LOC

LDX VERDEF
GET LAST VERT END POINT

STX VERBEG
PUT IN START POINT VERT STORAGE LOC

CMPA #$AC
TOKEN FOR MINUS SIGN

BEQ L939E
BRANCH IF NO STARTING COORDINATES GIVEN

JMP L938D
GO GET STARTING COORDINATES

LDX #F0

JMP L939C
EVALUATE TWO EXPRESSIONS

LDX #00

JSR L93A3
SYNTAX CHECK FOR A (TEMP STORAGE LOCATION FOR END COORDINATES OF LINE COMMAND

JSR L9303
GET END POINT COORDS

L938D
BRA L93BB
CHECK SYNTAX FOR )

L939C
JMP L9267
SYNTAX CHECK FOR ) AND RETURN

L93E9
CMPA #$BE
PRESET TOKEN?

L93BB
CMPA #$BD
PSET TOKEN?

L93E9
CMPA #$BF
PSET TOKEN?

L93BB
CMPA #$B7
PSET TOKEN?

L9303
LDX HOREND
TEMP STORAGE LOCATION FOR END COORDINATES OF LINE COMMAND

L93A3
JSR L9303
GET END POINT COORDS

L93BB
BRA L93BB
CHECK SYNTAX FOR )

L939C
JMP L9267
SYNTAX CHECK FOR ) AND RETURN

L93E9
CMPA #$BE
PRESET TOKEN?

L93BB
CMPA #$BD
PSET TOKEN?

L93E9
CMPA #$BF
PSET TOKEN?

L93BB
CMPA #$B7
PSET TOKEN?

L9303
LDX HOREND
TEMP STORAGE LOCATION FOR END COORDINATES OF LINE COMMAND

L93A3
JSR L9303
GET END POINT COORDS

L93BB
BRA L93BB
CHECK SYNTAX FOR )

L939C
JMP L9267
SYNTAX CHECK FOR ) AND RETURN

L93E9
CMPA #$BE
PRESET TOKEN?

L93BB
CMPA #$BD
PSET TOKEN?

L93E9
CMPA #$BF
PSET TOKEN?

L93BB
CMPA #$B7
PSET TOKEN?
2690 9451 IF #2 L9451 TFR D,Y SAVE DIFFERENCE IN Y
2690 9453 31 21 LEA $01,Y ADD ONE TO DIFFERENCE - TURN ON STARTING & ENDING COORDS
2691 9455 BD 92 98 JSR L9298 GET ABS SCREEN POS TO X AND PIXEL MASK TO ACCA
2692 9458 36 48 PULS U GET START COORDS
2693 945A BD 0D STU HORRNB RESTORE THEM
2694 945C BD 36 BSR L949A POINT U TO ROUTINE TO MOVE PIXEL POINTERS TO RIGHT
2695 945E BD 97 07 L945E STA VD SAVE PIXEL MASK
2696 9468 BD 93 77 JSR L9377 TURN ON PIXEL
2697 946C 96 07 LDA VD GET OLD PIXEL MASK
2698 946F BD 04 JSR U MOVE TO NEXT ONE TO RIGHT
2699 946F 31 3F LEA $-1,Y DEC COUNTER
2700 946F 26 F3 BNE L945E LOOP IF NOT DONE
2701 9468 39 L946B RTS
2702 946C 35 #6 L946C PULS A,B CLEAN UP STACK

2703
2704 * DRAW A VERTICAL LINE FROM VEREND TO VERBEG AT HOORD HORRNB
2705 946E 0C BF L946E LDO VERBEG GET END COORDS
2706 9478 34 #6 PSHS B,A SAVE THEM
2707 9472 BD 97 10 JSR L971B CALCULATE ABSOLUTE VALUE OF VEREND-VERBEG
2708 9475 28 84 BCC L9478 BRANCH IF END COORD = START COORD
2709 9477 9E 05 LDX VEREND *
2710 9479 9F BF STX VERBEG *SWITCH VER COORDS IF END COORD IS TO RIGHT OF START
2711 947A BF #2 L947A TFR D,Y LENGTH OF LINE TO Y
2712 9470 31 21 LEA $01,Y SET BOTH START AND END COORDS
2713 947F BD 92 9B JSR L929B GET ABSOLUTE SCREEN POS TO X, MASK TO ACCA
2714 9482 35 48 PULS U GET END COORD
2715 9484 BF 0D STU VERBEG RESTORE THEM
2716 9486 BD 15 BSR L949D POINT U TO ROUTINE TO MOVE DOWN ONE ROW
2717 9488 2D #0 BRA L945E DRAW LINE VERT

2718 *
2719 * JUMP TABLE OF ADDRESSES OF ROUTINES WHICH WILL MOVE THE
2720 * ABSOLUTE SCREEN ADDRESS POINTER ONE PIXEL TO THE RIGHT.
2721 948A 92 ED L948A FDB L92ED PMODE 0
2722 948C 92 F4 L948C FDB L92F4 PMODE 1
2723 948E 92 ED L948E FDB L92ED PMODE 2
2724 9492 92 F4 L9492 FDB L92F4 PMODE 3
2725 9492 92 ED L9492 FDB L92ED PMODE 4

2726 *
2727 * POINT U TO ROUTINE WHICH WILL MOVE PIXEL ONE TO RIGHT
2728 9494 CE 94 8A L9494 LDU L#948A POINT TO JUMP TABLE
2729 9497 0E #6 LDP PMODE GET PMODE VALUE
2730 9499 58 X2 ASLB
2731 949A EE #5 LDU B,U GET JUMP ADDRESS
2732 949C 39 RTS
2733 949D CE 92 E9 L949D LDU L#92E9 POINT U TO ROUTINE TO MOVE ABS POS DOWN ONE ROW
2734 9494 39 RTS

2735
2736 * DRAW LINE FROM (HORRNB,VERBEG) TO (HOREND,VEREND)
2737 94A1 1B #0 95 BD L94A1 LDY L#95BD POINT Y TO INCR VERBEG
2738 94A5 BD 97 10 JSR L971B CALCULATE VEREND-VERBEG (VERTICAL DIFFERENCE)
2739 > 94AB 1B 27 FF 98 LBD L9444 DRAW A HORIZONTAL LINE IF DELTA Y = 0
2740 94AC 24 #4 BCC L942B BRANCH IF END VEREND > VER START COORD
2741 94A8 1B #0 95 BD L9482 LDY #95BD POINT Y TO DECR VER COORD (VERBEG)
2742 94B2 34 #6 L94B2 PSHS B,A SAVE DELTA V
2743 94B4 CE 95 #6 LDU L#95BD POINT U TO INCR HORD COORD
2744 94B7 BD 97 1D JSR L971D CALCULATE HOREND-HORRNB (HOR DIFFERENCE)
2745 94BA 27 BB LBD L946C DRAW A VERTICAL LINE IF DELTA H = 0
2746 94BC 24 #3 BCC L941C BRANCH IF HORD END COORD > HOR START COORD
2747 94BE CE 95 14 LDU L#9514 POINT U TO DECR HORD COORD
2748 94C1 1B #3 E4 L94C1 CMPD S COMPARE DELTA H TO DELTA V
2749 94C4 35 #B PULS V PUT DELTA V IN X
2750 94C6 24 #4 BCC L944C BRANCH IF DELTA H = DELTA V
2751 94CB 1E #2 EXG U,Y SNAP CHANGE HORIZ & CHANGE VER ADDRESS
2752 94CA 1E #1 EXG D,X EXCHANGE DELTA H & DELTA V
2753 94CC 46 #4 L94CC PSHS U,B,A *SAVE THE LARGE OF DELTA V, DELTA H
2754 * AND THE INCREMENT/DECREMENT ADDRESS
2755 94CE 34 #6 PSHS B,A SAVE WHICHEVER IS LARGER OF DELTA V, DELTA H
2756 94DD 44 LSRA *
2757 94D1 56 RORR *
2758 94D2 25 #9 BLD L94DD BRANCH IF GOD NUMBER
2759 94D4 11 #3 95 BD CMU L#95BD+1 SEE IF INCR OR DECR
2760 94D8 25 #3 BLD L94DD BRANCH IF INCR
2761 94D3 33 #1 SUBT W |
2762 94D0 34 #6 L94DD PSHS X,R,A *SAVE SMALLEST DELTA (X) AND INITIAL MINOR COORDINATE
2763 * INCREMENT COUNTER WHICH IS 1/2 OF LARGEST DELTA
2764 94DF BD 8F #0 JSR L92BF POINT U TO PROPER COORDINATE TO SCREEN CONVERSION ROUTINE

2765
2766 * DRAW THE LINE HERE - AT THIS POINT THE STACK HAS THE DRAW DATA ON IT
2767
2768 * 1,0,MINOR COORDINATE INCREMENT COUNTER
2769 * 2,3,ASSOLUTE VALUE OF THE SMALLEST DELTA COORDINATE
2770 * 4,5,ASSOLUTE VALUE OF THE LARGEST DELTA COORDINATE
2771 * 6,7,LARGEST COORDINATE COUNTER (HOW MANY TIMES THROUGH THE DRAW
2772 ** LOOP), INITIALLY SET TO ABSOLUTE VALUE OF LARGEST DELTA COORD
2773 * 8,9,SHUTDOWN OF THE ROUTINE WHICH WILL INCREMENT OR DECREMENT
2774 * THE LARGEST DELTA COORDINATE

2775
2776 94E2 AD #4 L94E2 JSR U CONVERT (X,Y) COORDINATES TO ABSOLUTE SCREEN ADDRESS
2777 94E4 BD 93 77 JSR L9377 TURN ON A PIXEL
2778 94E7 4E 66 LDX $86,S GET DISTANCE COUNTER
2779 94E9 27 17 BEQ L9582 BRANCH IF LINE IS COMPLETELY DRAWN
2780 94EB 3B #1 LEA $-1,Y DEC X ONE
2781 94ED AF 66 STX $86,S SAVE IT
2782 94EF AD #8 JSR [$80,S] INCX/DECX COORDINATE WHICH HAS THE LARGEST DELTA
2783 94F3 EC #4 LDU S GET THE MINOR COORDINATE INCREMENT COUNTER
2784 94FA E3 62 ADD $2,Y ADD THE SMALLEST DIFFERENCE

829
DISASSEMBLY OF EXTENDED BASIC 1.1

2789 94FE ED E4 STD .S SAVE NEW MINOR COORDINATE INCREMENT COUNTER
2796 94FB A3 64 SUB $04,$5 *SUBTR OUT THE LARGEST DIFFERENCE AND
2797 94FA 25 E6 BLD L94E2 *BRANCH IF RESULT NOT > LARGEST DIFFERENCE
2798 94FC ED E4 STY 1+, THEN STORE NEW MINOR COORDINATE INCREMENT
2799 94FE A0 A4 JSR ,Y INCREMENT/DECIMENT COORDINATE WHICH HAS THE SMALLEST DELTA
2800 9500 2# 8B BRA L94E2 KEEP GOING
2801 9502 35 #0 L9502$ PULS X * PULS A, R, X, Y, U, PC
2802 9504 35 F6 *CLEAR UP THE STACK AND RETURN
2803 *
2804 * THESE ROUTINES ARE USED TO INCREMENT OR DECREMENT THE
2805 * HORIZONTAL & VERTICAL COORDINATES. THEY NEED TO BE KEPT
2806 * IN THIS ORDER (INCR, INCR, DECR, DECR).
2807 *
2808 * INCR HORBEG (HOR COORD)
2809 9506 9E BD L9506 LDX HORBEG GET COORDINATE
2810 9508 3# 01 LEAX $01,X ADD ONE
2811 950A 9F BD STX HORBEG SAVE COORDINATE
2812 950C 39 RTS
2813 *
2814 * INCR VERBEG (VER COORD)
2815 950D 9E BF L950D LDX VERBEG GET COORDINATE
2816 950F 3# 01 LEAX $01,X ADD ONE
2817 9511 9F BF STX VERBEG SAVE COORDINATE
2818 9513 39 RTS
2819 *
2820 * DECR HORSES (HOR COORD)
2821 9514 9E BD L9514 LDX HORBEG GET COORDINATE
2822 9516 3# 1F LEAX $-01,X SUBTRACT ONE
2823 9518 9F BD STX HORBEG SAVE COORDINATE
2824 951A 39 RTS
2825 *
2826 * DECR VERBEG (VER COORD)
2827 951B 9E BF L951B LDX VERBEG GET COORDINATE
2828 951D 3# IF LEAX $-01,X SUBTRACT ONE
2829 951F 9F BF STX VERBEG SAVE COORDINATE
2830 9521 39 RTS
2831 *
2832 * GET MAXIMUM VALUE OF HOR/VER COORDINATES
2833 *
2834 * NORMALIZED FOR PROPER PMODE. RETURN VALUES
2835 *
2836 * HOR = V23 VER = V05
2837 9522 CE #0 D3 L9522 LDU #V03 POINT U TO TEMP STORAGE AREA (V03)
2838 9525 2E #0 FF LDS #$55 MAXIMUM VALUE HORIZONTAL COORD (255)
2839 9528 AF C4 STX ,U SAVE IT
2840 952A 8E #0 BF LDX #$11 MAXIMUM VALUE VERTICAL COORD (191)
2841 952D 42 AF STX $02,U SAVE IT
2842 952F 7E 93 20 JMP L9320 GO CONVERT THEM TO PROPER PMODE
2843 *
2844 * PCLS
2845 9532 27 0E PCLS BEQ L9542 CLEAR TO BACKGROUND COLOR IF NO ARGUMENT
2846 9534 BD 24 L9534 LDA #$5A EVALUATE EXPRESSION, CONVERT TO PROPER COLOR CODE
2847 9536 86 55 L9536 LDA #$55 CONSIDER EACH BYTE AS 4 GROUPS OF 2 BIT SUB-NIBBLES
2848 9538 3D 0D MUL MUL BY COLOR
2849 9539 9E BA L9539 BEGEP GET STARTING ADDR
2850 953B 87 0A L953B STB ,X+ SET BYTE TO PROPER COLOR
2851 953C 90 C7 CMPX ENDRP AT END OF GRAPHIC PAGET
2852 953F 26 FA BRA L953B NO
2853 9541 39 RTS
2854 *
2855 * L9542 LDB BACOL GET BACKGROUND COLOR
2856 9544 20 88 BRA L9536
2857 *
2858 * COLOR
2859 9546 81 2C COLOR CMPL ' . ,.' *CHECK FOR COMMA AND
2860 9548 27 88 BEQ L9552 *BRANCH IF FOREGROUND COLOR ARGUMENT MISSING
2861 954A BD 8E BSR L955A EVALUATE FIRST ARGUMENT
2862 954C D7 82 STB FORCOL STORE IN FOREGROUND LOCATION
2863 954E 90 A5 JSR GETCOL GET NEXT INPUT CHARACTER
2864 9550 27 07 BEQ L9559 RETURN IF NONE
2865 9552 BD 60 L9552 JSR SYMCONMA SYNTAX CHECK FOR COMMA
2866 9555 BD 03 BSR L955A EVALUATE LAST ARGUMENT
2867 9557 D7 83 STB BACOL STORE IN BACKGROUND COLOR
2868 9559 39 L9559 RTS
2869 *
2870 * EVALUATE AN EXPRESSION AND CONVERT IT TO A PROPER COLOR CODE
2871 *
2872 * DEPENDING ON THE PMODE AND CSS; ILLEGAL FUNCTION CALL IF B > 8 -
2873 * RETURN COLOR VALUE IN ACCB; CSS VALUE IN ACCA
2874 955A BD 70 8B L955A JSR EVALEXP EVALUATE EXPRESSION
2875 955C BD 40 L955C CMPL #8B ONLY ALLOW =-B
2876 955E 1F 24 1E 07 LBCB LB44A ILLEGAL FUNCTION CALL IF BAD COLOR
2877 9560 3F 0F CLRA VDG CSS VALUE FOR FIRST COLOR SET
2878 9564 80 05 CMPL #85 FIRST OR SECOND COLOR SET?
2879 9566 25 40 BLD L956C BRANCH IF FIRST SET
2880 9568 86 8B LDS #8B VDG CSS VALUE FOR SECOND COLOR SET
2881 956A 84 04 SUBB #84 MAKE 5-8 BECOME 1-4
2882 956C 34 02 L956C PSHS A SAVE VDG CSS VALUE ON THE STACK
2883 956E 96 86 LDS PMODE GET PMODE
2884 9570 46 RORA 4 COLOR OR 2 COLOR
2885 9571 24 8B BEC L9578 2 COLOR
2886 9573 20 TSB WAS COLOR = 0
2887 9574 26 02 BNE L9578 NO
2888 9576 84 L9576 LDS #84 IF SO, MAKE IT 4
2889 9578 5A L9578 DECB CONVERT -1 & 0 TO -3
2890 9579 35 2B L9579 PULS A, PC PUT VDG CSS VALUE IN ACBA AND RETURN
2891 957B 56 L957B JSR CHECK ONLY THE LSB OF COLOR IF IN 2 COLOR MODE
2892 957C 25 FB BLD L9576 BRANCH IF 00 - FORCE ACCB TO 3
2893 957E 57 CLRB FORCE ACCB = 0 IF EVEN
2894 957F 2F 8B BRA L9579 RETURN
2895 *
2896 * SET THE CURRENT ACTIVE COLOR AND ALL PIXEL BYTE
2897 *
2898 * TO FOREGROUND/BACKGROUND COLOR DEPENDING ON

REVISED:12/26/1999 WALTER K ZYDHEK
ORIGIN: SPECTRAL ASSOC
* PSET, PRESET IF NO EXPRESSION , ) OR

- OTHERWISE EVALUATE THE EXPRESSION

> 9581 BD 95 9A L95B1 JSR L959A GET THE COLOR OF A BYTE

> 9584 BD 90 A5 JSR GETCHC CHECK CURRENT INPUT CHARACTER

> 9585 95B6 27 1# BEQ L959B BRANCH IF NONE

> 9586 95BA 29 COMA # 7 # CHECK FOR ) AND BRANCH IF

> 9587 95BA 27 8C BEQ L959B * NO MORE ARGUMENTS

> 958C BD B2 60 JSR SYNCCOMA SYNTAX CHECK FOR COMMA

> 958F BD 21 C0 COMA # 7 # WAS NEXT CHARACTER A COMMA?

> 9591 27 25 BEQ L959B YES

> 9593 BD 95 5A JSR L955A EVALUATE EXPRESSION, RETURN COLOR IN ACB

> 9596 BD 9A 8A L95A2 TEMP STOR COLOR AND ALL PIXEL BYTE

> 9598 9E A5 L959B JMP GETCH CHECK INPUT CHARACTER AND RETURN

* SET THE ACTIV COLOR BYTE AND THE ALL ACTIV COLOR BYTE

> 9599 BD 82 L959A LDB FDRCOL GET FOREGROUND COLOR

> 959C 9D C2 1ST SETPFLG CHECK PSET/PRESET FLAG

> 959E 9E 26 2 # BNE L95A2 BRANCH IF PSET

> 959F 9E 6B 83 LDB BACCOL GET BACKGROUND COLOR

L95A2 STR WCOLOR TEMP STOR COLOR

> 95A4 9E 80 LDA #55 CONSIDER A BYTE AS 4 PIXELS

L95AD 96 3D MUL SET COLOR ON ALL 4 PIXELS

> 95A7 07 85 STB ALLCOL SAVE BYTE WITH ALL PIXELS TURNED ON

> 95A9 39 RTS

L95AA BNE L95CF BRANCH IF GRAPHIC MODE, OTHERWISE SETUP ALPHA GRAPHIC MODE

* THIS CODE WILL RESET THE DISPLAY PAGE REGISTER IN THE

* SAM CHIP TO Z ($400) AND RESET THE SAM'S VDG CONTROL

* REGISTER TO 0 (ALPHA-NUMERIC). IN ADDITION, IT WILL

* RESET THE VDG CONTROL PINS TO ALPHA-GRAPHICS MODE.

** SET UP THE SAM AND VDG TO GRAPHICS MODE

L95AC PHLS X, A SAVE REGISTERS

> 95AD BE FF C8 LDX #53+8 POINT X TO THE MIDDLE OF THE SAM CNLT REG

> 95B1 A7 8A STA IX,X **

> 95B3 A7 8B STA #68,X ***

> 95B5 A7 87 STA #66,X ****

> 95B7 A7 84 STA #84,X ***** RESET SAM DISPLAY PAGE TO $400

> 95BB A7 #1 STA #81,X ***

> 95B9 A7 1E STA $-82,X **

> 95BF A7 1C STA $-81,X ***

> 95C1 A7 1A STA #86,X ***** RESET SAM VDG TO ALPHA-NUMERIC MODE

L95C3 A7 1B STA $-80,X ***

> 95C5 B6 FF 22 LDA PIA1+2 GET DATA FROM PIA1, PORT B

> 95CB 84 87 ANDA #$87 FORCE ALL BITS TO ZERO, KEEP ONLY CSS DATA

> 95CA 87 FF 22 STA PIA1+2 PUT THE VDG INTO ALPHA-GRAPHICS MODE

> 95CD 35 96 PULS A,B,X,PC RETURN

L95CF PHLS X,R,A

> 95D1 96 86 LDA PMODE GET CURRENT PMODE VALUE

> 95D3 88 83 ADDA #$83 ADD 3 - NOW 3-7 ONLY 5 OF 8 POSSIBLE MODES USED

> 95D5 9B C0 LDB #10 OFFSET BETWEEN PMODES

> 95D7 3D 03 MUL GET PMODE VALUES FOR VDG GM0, GM1, GM2

> 95DB CA 8B ORB #59B FORCE BIT 7 HIGH (VDG A/G CONTROL)

> 95DA DA C1 ORB CSSVAL OR IN THE VDG CSS DATA

> 95DC B6 FF 22 LDA PIA1+2 GET PIA1, PORT B

> 95DF 84 87 ANDA #$87 MASK OFF THE VDG CONTROL DATA

> 95E1 34 82 PHLS A SAVE IT

> 95E3 EA C8 ORB .5+ OR IT WITH THE VDG VALUES CALCULATED ABOVE

> 95E5 FF 22 STB PIA1+2 STORE IT INTO THE PIA

> 95EA 96 8A LDA BEGERR GET MSB OF START OF GRAPHIC PAGE

> 95EC 44 LSRA **DIVIDE BY 2 ACCA CONTAINS HOW MANY 812 BYTE

> 95F0 26 81 BNE L95F7 *BLOCKS IN STARTING ADDR

> 95F3 BD 96 8F JSR L95BF GO SET SAM CONTROL REGISTER

> 95F6 9E 86 LDA PMODE GET PMODE VALUE

> 95F9 BB 83 ADDA #$83 ADD IN BIAS TO ADJUST TO PMODE THE SAM REGISTER WANTS

> 95FA 81 C7 CMIA #$87 WAS PMODE 47

> 95FB 26 81 BNE L95F7 NO

> 95FD 4A DECA DECREMENT ACCA IF PMODE 4 (SAME VDG AS PMODE3)

> 95FF BD 82 L95F7 BSR L95BF SET THE SAM VDG REGISTER

> 95F9 35 96 PULS A,B,X,PC RESTORE REGISTERS AND RETURN

> 95FA C6 3 # L95F8 LDB #53 3 BITS IN SAM VDG CONTROL REGISTER

> 95FB C6 3 A SET ENTER WITH DATA TO GO IN VDG REGISTER IN BOTTOM 3 BITS OF ACCA

> 95FC 8E FF CB LDX #53+8 POINT X TO SAM CONTROL REGISTER

> 95FE 80 46 L9600 HORA PUT A BIT INTO CARRY FLAG

> 95F0 24 84 BCC L9607 BRANCH IF BIT WAS A ZERO

> 95F9 80 81 STA #81,X SET SAM REGISTER BIT

> 95FA 80 82 BRX L9609 DO NEXT BIT

> 95FB 80 87 STA ,X CLEAR SAM REGISTER

> 95F2 80 82 LEAX #82,X NEXT BIT IN REGISTER

> 95F3 80 5A DECB DONE ALL BITS?

> 95F4 8C 82 BNE L9600 NO

> 95F5 E6 39 RTS

L9606 LDN #53+8 7 BITS IN SAM DISPLAY PAGE REGISTER

L9607 BE FF C6 LDX #53+9 POINT X TO SAM DISPLAY PAGE REGISTER

> 95F8 8A EA BRA L9600 GO SET THE REGISTER

> 8619 86 FF 22 L9616 LDA PIA1+2 GET PIA1, PORT B

> 961A 8A 87 ANDA #$77 MASK OFF VDG CSS CONTROL BIT

> 961B 9A C1 ORA CSSVAL OR IN CSS COLOR DATA

> 961D 87 FF 22 STA PIA1+2 RESTORE IT IN PIA1

> 9628 39 RTS

* PMOD

> 9621 B1 2C PMOD CMIA #. ' CHECK FOR COMMA - FIRST ARGUMENT IS MIGHT BE MISSING
APPENDIX B

DISASSEMBLY OF EXTENDED BASIC 1.1

ORIGIN/SPECTRAL ASSOC

REVISED:12/28/1999 WALTER K ZFEDER

EXTENDED BASIC UNRAVELLED II

INFOGRAPHIC

3277 9632 27 B REG L966B IT IS A COMM
3278 9625 BD 87 BD JSR EVALEXPB EVALUATE EXPRESSION
3279 9628 C1 85 CMBR #$85 > 47
3280 962A 2A 41 BCC L966D YES, ILLEGAL FUNCTION CALL
3281 962C 96 BC LDA GRPRAM GET THE START OF GRAPHIC RAM
3282 962E 97 BA L962E STA BEGGRP SET START GRAPHIC PAGE
3283 9638 58 ADD #2, #2 MULT MODE BY 2 - TABLE HAS 2 BYTES PER ENTRY
3284 9631 CE 97 87 LUO #9768+1 LOOKUP TABLE
3285 9634 2A 45 ADDB #5 ADD THE AMOUNT OF MEMORY REQUIRED FOR ONE GRAPHIC PAGE
3286 9636 91 19 CMPC TXTTAB COMPARE TO HS# OF START OF BASIC PROGRAM
3287 963B 22 33 BHI L966D FC ERROR IF END OF GRAPHIC PAGE > START OF BASIC PROGRAM
3288 963A 97 87 STA ENDBRP STORE THE END OF GRAPHIC PAGE
3289 963C 33 5F LEAU $4, U POINT U TO PREVIOUS BYTE IN TABLE
328A 963E A6 C5 LDA B,U *GET THE NUMBER OF BYTES/HORIZONTAL LINE
328B 9648 97 B9 STA HOBRT *AND SAVE IT IN HOBRT
328C 9642 54 LSBB RESTORE PMODE VALUE
328D 9643 D7 BD STB PMODE SAVE IT
328E 9646 4F CLR A CLEAR BACKGROUND COLOR
328F 9646 97 B3 STA BAKCOL SET BACKGROUND COLOR TO ZERO
3290 964B 86 93 LDA #$83 FOREGROUND COLOR
3291 964A 97 B2 STA FORCOL SET FOREGROUND COLOR
3292 964C 90 A5 JSR GETCCH IS THERE A STARTING PAGE NUMBER?
3293 964C 7C 1C LEB #$86 NO
3294 9650 BD 97 38 L9650 JSR LB738 EVALUATE EXPRESSION
3295 9653 50 TSTB SET FLAGS
3296 9654 27 17 BEQ L966D ILLEGAL FUNCTION CALL - CAN'T T START ON PAGE ZERO
3297 9656 5A DECB BUMP ONE, BASIC STARTS ON PAGE 1, THIS ROUTINE AT #
3298 9657 96 #6 L9657 #586 EACH GRAPHIC PAGE = 6 X 256 (1.5K)
3299 9659 3D MUL MULT BY PAGE NUMBER
3300 965A DB 8C ADDB GRPRAM ADD TO START OF GRAPHIC RAM
3301 965C 24 94 POS B SAVE TEMP START ADDR
3302 965D 8B 97 BEQ ENDBRP ADD CURRENT END ADDR
3303 9660 DB BA SUBB BEGGRP SUB OUT CURRENT START ADDR - ADDS THE SIZE OF ONE GRAPHIC PAGE
3304 9662 01 CMPC TXTTAB IS IT > CURRENT START OF BASIC PROGRAM
3305 9664 22 87 BHI L966D YES! ILLEGAL FUNCTION CALL
3306 9666 97 B3 STB ENDBRP SAVE AS END OF GRAPHIC PAGE
3307 9668 35 9F BUMP B GET TEMP START ADDR
3308 966A D7 BA STB BEGGRP SAVE AS START OF GRAPHIC PAGE
3309 966C 2A 49 L966C RTS
3310 966D 7E 84 4A L966D JMP LB44A ILLEGAL FUNCTION CALL'
3311 9672 B7 4A
3312 9678 81 2C SCREEN CMPC #0, 'CHECK FOR A COMMA
3313 9679 28 2A BRK L676F BRANCH IF COMMA - FIRST ARGUMENT MISSING
3314 967A BD 97 BEQ L966D EVALUATE EXPRESSION
3315 967B 50 TSTB ZERO FLAG SET IF ALPHA, NOT SET IF GRAPHIC SCREEN
3316 967C 8D 95 A JSR L954A SET UP THE SAM & VDU FOR PROPER GRAPHIC MODE
3317 967D 90 A5 JSR GETCCH GET NEXT CHARACTER
3318 967E 7D EO BEQ L966C RETURN IF NOTHING ELSE ON LINE
3319 967F 9D 83 L967F JSR LB738 CHECK FOR COMMA AND EVALUATE EXPRESSION
3320 9682 50 TSTB SET FLAGS
3321 9683 27 BD BEQ L9687 BRANCH IF COLOR SET ZERO
3322 9686 CB 90 LEB #$86 VALUE FOR COLOR SET ONE
3323 9687 D7 C1 L9687 STB CSVAL SAVE IN V04 CSS RAM IMAGE
3324 9689 2B 8B BRA L9616 GO SET IT INTO PIA
3325 968B D7 87 
3326 968B DB 87 BD JSR EVALEXPB EVALUATE EXPRESSION, RETURN VALUE IN ACCB
3327 968E 50 TSTB SET FLAGS
3328 968F 27 DC REG L966D BRANCH IF PCELABB - FC ERROR
3329 9691 C1 89 CMBR #$89 TRYING TO CLEAR MORE THAN 8 PAGES?
332A 9693 24 08 BCC L966D YES ILLEGAL FUNCTION CALL
332B 9695 86 96 LDA #$86 6 X 256 (1.5K) PER GRAPHIC PAGE
332C 9697 3D MUL MULT BY NUMBER OF PAGES
332D 9698 DB 8C ADDB GRPRAM ADD TO START OF GRAPHIC RAM
332E 9699 1F 9B TRB B,A MOVE B TO MSR OF REG ACCB
332F 969C 65 C1 LDB #$81 REG D NOW CONTAINS TOP OF PCELAB SPACE +1
3330 969F 1F 82 TFR D,Y SAVE IN Y
3331 969F 9B 93 87 CMPC ENDBP COMPARE TOP OF PCELAB SPACE TO END OF CURRENT GRAPHIC PAGE
3332 96A5 23 C8 ALE J966D FC ERROR IF TRYING TO CLEAR < END OF GRAPHIC RAM
3333 96A8 93 19 SUBD TXTTAB SUBTRACT START OF BASIC PROGRAM
3334 96A9 03 1B ADD VARTAB ADD END OF BASIC PROGRAM
3335 96A9 1F 81 TFR D,Y TOP OF PCELAB SPACE + LENGTH OF BASIC PROGRAM
3336 96B1 4C INCN ADD 256 - LEAVE SOME ROOM FOR THE STACK
3337 96B3 9C 21 SUBT PRETOP SUBTRACT TOP OF CLEARED SPACE
3338 96B4 2A 4D BCC L966D FC ERROR - NO ROM LEFT
3339 96B8 BD 80 CMPC L980# ADD BASIC'S 5 INPUT POINTER
3340 96B9 11 12 NOP SPACE FILLER FOR EXAB 1.1
3341 96B4 04 1D LDA VARTAB GET END OF BASIC PROGRAM
3342 96B6 8F 1B STX VARTAB STORE NEW END OF BASIC PROGRAM
3343 96B8 11 13 8B CMPU VARTAB COMPARE OLD END TO NEW END
3344 96BB 24 17 BCC L9604 BRANCH IF OLD END = NEW END
3345 96B8 A6 C2 L968B LDA -$9 GET BYTE FROM OLD PROGRAM
3346 96BA AF 82 STA X MOVE TO NEW PROGRAM LOCATION
3347 96C1 11 13 99 CMPU TXTTAB AT THE BEGINNING OF OLD PROGRAM?
3348 96C4 26 F7 BNE L968D NO
3349 96C6 18 9F 99 STY TXTTAB SAVE NEW STARTING ADDRESS
3350 96C9 6F 3F CLR $-81,Y CLEAR BYTE JUST BEFORE PROGRAM
3351 96CA 8D 0C EF L966B JSEP LACEF PUT CORRECT ADDRESSES IN FIRST 2 BYTES OF EACH LINE
3352 96CE BD AD 26 JSR LAD02 DO PART OF A NEW
3353 96D0 7E 9A JMP LAD09 GO BACK TO BASIC 5 MAIN LOOP
3354 96D4 0E 19 L966D STA TXTTAB GET START OF BASIC PROGRAM
3355 96D6 18 9F 99 STY TXTTAB STORE NEW STARTING ADDR
3356 96D9 6F 3F $-81,Y CLEAR THE BYTE JUST BEFORE PROGRAM
3357 96DB A6 C9 L968B LDA -$9 GET BYTE FROM OLD PROGRAM

832
APPENDIX B

DISASSEMBLY OF EXTENDED BASIC 1.1

REVISED: 10/28/1999 WALTER K. ZYDHEK

*INITIALIZATION ROUTINE FOR EWBAS GRAPHICS VARIABLES

* TABLE OF HOW MANY BYTES/GRAPHIC ROW AND HOW MUCH RAM

* FOR ONE HI RES SCREEN FOR THE PMODES. ROWS FIRST,

* TABLE OF HOW MANY BYTES/GRAPHIC ROW AND HOW MUCH RAM

* FOR ONE HI RES SCREEN FOR THE PMODES. ROWS FIRST,

* TABLE OF HOW MANY BYTES/GRAPHIC ROW AND HOW MUCH RAM

* FOR ONE HI RES SCREEN FOR THE PMODES. ROWS FIRST,

* TABLE OF HOW MANY BYTES/GRAPHIC ROW AND HOW MUCH RAM

* FOR ONE HI RES SCREEN FOR THE PMODES. ROWS FIRST,

* TABLE OF HOW MANY BYTES/GRAPHIC ROW AND HOW MUCH RAM

* FOR ONE HI RES SCREEN FOR THE PMODES. ROWS FIRST,

* TABLE OF HOW MANY BYTES/GRAPHIC ROW AND HOW MUCH RAM

* FOR ONE HI RES SCREEN FOR THE PMODES. ROWS FIRST,
1319 977A 33 CB LEAU D,U POINT U TO END OF ARRAY
1317 977E DF D1 STU VD0 SAVE END OF DATA (END OF ARRAY)
1317 977B 3F #2 LEAK $82,X POINT X TO NUMBER OF DIMENSIONS AND
1317 977A E6 84 LBD ,X GET NUMBER DIMENSIONS IN ACCB
1313 977C 58 ASLB TIMES 2 - 2 BYTES/DIMENSION
1317 977D 30 APX POINT X TO START OF ARRAY DATA
1317 977E 9F CF STH VCF SAVE START OF DATA (START OF ARRAY DATA)
1317 9788 96 #6 LDA VALTP CHECK VARIABLE TYPE
1317 9782 26 CE BNE L9752 FC ERROR IF STRING VARIABLE
1317 978A #F 04 CLR VD4 GET/PUT GRAPHIC/ACTION FLAG
1317 9786 90 A5 JSR GETCH1 GET CURRENT INPUT CHAR
1317 9788 27 20 BSE L9877 BRANCH IF END OF LINE
1317 979A #D 04 COM VD4 TOGGLE GET/PUT GRAPHIC/ACTION FLAG
1317 9786 0D B2 JSR SYNCCOMA SYNTAX CHECK FOR COMM
1317 9786 0D D0 JSR SETVFD CHECK/GET FLAG
1317 9791 26 #7 BNE L979A BRANCH IF PUT
1317 9793 46 77 LDB &H' DO A SYNTAX CHECK FOR & 6
1317 9790 #B 6F JSR L9B6F DO A SYNTAX CHECK FOR & 6
1317 979B 2B #8 BRA L97CA SKIP AROUND THE NO G OPTION CODE
1317 979A C6 #S L979A LD #S5 FIVE LEGAL TOKENS AT END OF PUT
1317 979C 9E 9B 39 LDX #H9B39 POINT X TO LOOK UP TABLE
1317 979F EE B1 L979F LDU ,X,** GET CLEAR BIT ACTION ROUTINE ADDRESS
1317 97A1 1B AE B1 LDX ,X,** SET BIT ACTION ROUTINE ADDRESS
1317 97AA A1 #8 CMPA ,X,* CHECK FOR ONE OF THE FIVE LEGAL TOKENS
1317 97AA 27 #6 BEQ L9UA1 FOUND ONE
1319 97A8 5A DECB CHECKED ALL FIVE?
1319 97A9 26 F4 BNE L979F NO - KEEP GOING
1319 97A6 7C B2 77 JMP L8B77 BRANCH IF END OF LINE
1317 97A6 B7 9F 05 L97AE STY VD5 ARRAY SET BIT ACTION ROUTINE ADDRESS
1317 97B1 0F D9 STU VD9 ARRAY CLEAR BIT ACTION ROUTINE ADDRESS
1317 979B 9D 9F JSR GETCH BRANCH IF END OF LINE
1317 97B0 2B 13 BRA L97CA SKIP AROUND THE NO G OPTION CODE
1320 97B1 #0 GC OPTION OR ACTION SPECIFIED BY BASIC INPUT LINE
1320 97B6 27 18 L97DB LDB #F8 BOTTOM 3 BITS MASK (8 PIXELS/BYTE)
1320 97B9 96 B6 LDX PMODE GET CURRENT PMODE
1320 97BA 46 RORA BIT @ CARRY
1320 97BC 24 #2 LECC L97C8 BRANCH IF PMODE $,4,2 (COLOR)
1320 97BE C6 4C LBD FC BOTTOM 2 MASKS (4 COLOR MODE - 4 PIXELS/BYTE)
1320 97CB 1F 9B L97CB TFR B,A SAVE ACCB IN ACCA
1320 97CD 4E 8E ANDB HORBEG+1 *
1320 97CD 07 #E LTX VERBEG *
1320 97C0 9C 44 ANDA HOREND+1 *
1320 97C4 9D 44 THE HORIZONTAL DIFFERENCE
1320 97CA BD 97 1D L97CA JSR L971D CALC HORIZ DIFFERENCE AHS(HOREND-HORBEG)
1320 97CD 24 #4 BCC L97D3 BRANCH IF END = START
1320 97CF 9E C3 LDH HOREND * MAKE START = END IF
1320 97D1 0F B0 STX VERBEG *
1320 97D2 0D #3 JSR L97B1 CALC VERT DIFFERENCE AHS (VEREND-VERBEG)
1320 97DB 24 #4 BCC L97DE BRANCH IF END > START
1320 97DB 9E C5 LDV VEREND *
1320 97D4 0B BF STV VERBEC *
1320 97D5 2C #5 L97D0 STD VEREND SAVE VERT DIFFERENCE
1320 97D7 96 BD LDA PMODE *
1320 97D8 46 RORA *
1320 97D9 06 DC STX VEREND *
1320 97DB 06 #5 L97DB JSR L9AE2 NORMALIZE DIFFERENCES
1320 97D7 06 DC L97DE LDX HOREND GET HORIZ DIFFERENCE
1320 97DA 9E C5 LDV VEREND *
1320 97DC 94 #1 LEAX $1,X *
1320 97DD 9F C5 STX VEREND *
1320 97DE 00 #0 TST VD4 *
1320 97DF 2F #4 TST VD4 = CHECK FOR & OPTION OR ACTION GET ACTION
1320 97EF 26 5B BNE L98B2 = AND BRANCH IF GIVEN
1320 97FA 94 LSRA *
1320 97FB 56 HORB *
1320 97FC 44 LSRA *
1320 97FD 56 HORB *
1320 97FE 44 LSRA *
1320 97FF 56 HORB *
1320 97FF 60 ADDO #2 ADDO #1 * DIVIDE HORIZONTAL DIFFERENCE BY 8
1320 9803 DD 00 STD HOREND SAVE NEW HORIZ DIFFERENCE
1320 9805 9D 9B JSR L9296 =CONVERT (HOREND,VEREND) INTO ABSOLUTE SCREEN
1320 9805 *POS (X) AND PIXEL MASK (4 COLOR MODE - 4 PIXELS/BYTE)
1320 9806 0D C4 LECH $2 ,X POINT X TO MERGE POINTS IN SCREEN
1320 980A 24 18 L98B4 LDX #B,18 SAVE SCREEN POSITION
1320 980C 0D 00 L98BC TST VDB CHECK THE GET/PUT FLAG
1320 980E 27 16 BEO L9B31 BRANCH IF GET
1320 9818 0B 11 BSR L9B83 INCREMENT ARRAY DATA POINTER
1320 981B A6 C4 LDA ,U GET DATA FROM ARRAY
1320 981B A7 #B STA ,X,* PUT IT ON THE SCREEN
1320 981B 5A DECB DECREMENT HORIZ DIFFERENCE
1320 981E 26 F3 BNE L98BC BRANCH IF NOT AT END OF HORIZ LINE
1320 981F 35 18 GET SCREEN POSITION BACK
1320 981B 0D 92 09 JSR L92DE MOVE POSITION DOWN ONE ROW
1320 981B 0E 9A DEC VEREND+1 DECREMENT VERTICAL DIFFERENCE
1320 9825 0D 00 BNE L9888 BRANCH IF NOT DONE
1320 982A 39 L982Z RTS
3265 983C 26 F4 BNE L9822 RETURN IF NOT AT END
3266 982E 7E 4A 4A JMP L844A ILLEGAL FUNCTION CALL
3267
3268 9831 A6 08 L9831 LDA ,X+ GET DATA FROM SCREEN
3269 9833 BD EE BSR L9823 INCREMENT ARRAY DATA POINTER
3270 9835 AE C4 STA ,U STORE IN ARRAY
3271 9837 2E 00 BRA L9316 KEEP LOOPING TILL DONE
3272 *
3273 9839 9B 94 9B L9839 FDB L9894,L989B TOKEN FOR PSET
3274 9830 D0 L9830 FCB $B0 TOKEN FOR PSET
3275 983E 9B 9B 9B L983E FDB L9899,L989B
3276 9842 8E FCB $BE TOKEN FOR PRESET
3277 9843 9B 91 9B L9843 FDB L98B1,L989B
3278 9847 81 L9847 FCB $B1 TOKEN FOR OR
3279 984B 9B 9B 9B L984B FDB L9899,L98B1
3280 984C 80 FCB $B0 TOKEN FOR AND
3281 984D 9B 9A 9B L984D FDB L98A1,L98A1
3282 9851 A8 L9851 FCB $A0 TOKEN FOR NOT
3283
3284 * GET/PUT WITH ‘G’ OPTION SPECIFIED
3285 9852 C3 ## #1 L9852 ADDO #1 ADD ONE TO HORZ DIFFERENCE
3286 9855 DD C3 STD HOREND AND SAVE IT
3287 9857 96 08 LDA V00 *CHECK GET/PUT FLAG AND
3288 9859 26 09 BNE L9864 *BRANCH IF PUT
3289 985B DE 01 LDV V01 GET END OF ARRAYS
3290 985D A7 C2 L985D STA ,U *THIS CODE WILL
3291 985F 11 93 CF CMPP VCF *ZERO OUT THE ENTIRE
3292 9862 DF 9F L9862 SSHS VCF "GET" ARRAY
3293 9864 9B 92 9B L9864 JSR L929B =CONVERT (HOREND,VERBEG) INTO ABSOLUTE SCREEN POSITION
3294 *
3295 9866 DB 06 LPD PMODE GET CURRENT PMODE
3296 9869 56 ROBB BIT Ø TO CARRY
3297 986A 24 02 BCC L9866 BRANCH IF PMODE 0,2,4 (2 COLOR)
3298 986B 86 0A AND #AA USE BA AS THE PIXEL MASK IN 4 COLOR MODE
3299 986C 61 01 L9866 LSDH #SH1 INITIALIZE SHIFT CTR
3300 986E 18 NC 01 LV VCF POINT Y TO ARRAY DATA
3301 9873 34 12 L9873 PSHS X,A SAVE PIXEL MASK (ACCA) AND ABS SCRN POS (X) ON STACK
3302 9875 DE 03 LDH HOREND GET THE HORIZONTAL DIFFERENCE
3303 9877 34 42 L9877 PSHS U,A SAVE PIXEL MASK AND HORZ DIFF
3304 9879 54 L5BB SHIFT BIT CTR RIGHT
3305 987A 24 00 BCC L9866 BRANCH IF ALL #0 SHIFTS NOT DONE
3306 987C 56 ROBB SHIFT CARRY BACK INTO ACCA
3307 987D 31 21 LEAY $B1,Y INCREMENT ARRAY DATA POINTER
3308 987F 18 NC 01 CMPP V01 COMPARE TO END OF ARRAY
3309 9882 27 AA BEQ L982E FC ERROR IF AT END
3310 9884 0D 08 L9844 TSTD V08 CHECK THE GET/PUT FLAG AND
3311 9886 27 1F BEQ L98A7 BRANCH IT GET
3312 9888 0E 44 BITB,Y TEST A BIT IN ARRAY DATA
3313 988A 27 04 BEQ L9899 BRANCH IF ZERO
3314 988C 6E 9F 0D 05 JMP [V05] JUMP TO ACTION ROUTINE FOR ARRAY BIT SET
3315 9890 6E 9F 0D 09 L989B JMP [V09] JUMP TO ACTION ROUTINE FOR ARRAY BIT CLEAR
3316 9894 43 L9894 CMPP *MASK SOURCE DATA
3317 9895 A4 B4 ANDA,X *OFF OF SCREEN DATA
3318 9897 A7 B4 STA ,X SAVE TO SCREEN
3319 9898 26 16 BRA L98B1 THE ARRAY DATA
3320 989B AA 04 BRA L989B ORA ,X OR SOURCE DATA WITH SCREEN
3321 989D A7 B4 STA ,X SAVE TO SCREEN
3322 989F 26 10 BRA L98B1 THE ARRAY DATA
3323 99A1 0A B4 L98A1 EDRA,X INVERT THE PIXEL
3324 99A3 07 B4 STA ,X SAVE TO SCREEN
3325 99A5 2B 0A BRA L98B1 THE ARRAY DATA
3326 99A7 05 B4 L98A7 BITA,X TEST THE PIXEL
3327 99A9 2F 06 BEQ L98B1 BRANCH IF IT IS OFF
3328 99AB 1F 98 TFR B,A PUT SHIFT CTR IN ACCA
3329 99AD AA 0A ORA,Y TURN ON PROPER BIT IN
3330 99AF 07 B4 STA,Y THE ARRAY DATA
3331 99B1 35 42 L98B1 PULLS A,U RESTORE PIXEL MASK AND HLR DIFF
3332 99B3 02 9D JSR L92ED MOVE SCRNS POS & PIXEL MASK ONE TO RIGHT (TWO COLOR MODE)
3333 99B6 33 5F LEA $-1,0 *
3334 99BB 11 93 8A CMPP ZERD *DEC HORIZ DIFFERENCE AND
3335 99BC 26 B2 BNE L9877 *BRANCH IF NOT ZERO
3336 99BD AE 61 LDX $01,5 GET ABS SCRNS POS FROM STACK
3337 99BF 86 09 LDA HDRBT GET NUMBER BYTES/GRAPHIC ROW
3338 99C1 3B 86 MOVE SCRNS POS DOWN ONE RUN
3339 99C3 35 02 PULLS A,PULL PIXEL MASK OFF THE STACK
3340 99C5 32 62 LEAS $2,5 GET A OFF THE STACK
3341 99C7 8A C6 DEC VEREND+1 DECK VERT ROW CTR
3342 99C9 26 A8 BNE L9873 BRANCH IF NOT DONE
3343 99CB 39 RTS RETURN FROM GET/PUT COMMAND
3344
3345 99CC BD B3 57 L98CC JSR L9357 EVAL ALPHA EXPR, RETURN DESCRIPTOR PTR IN X
3346 99CF 67 62 LEA -$4,0,0 *STRIP OFF THE VARIABLE
3347 99D1 0A 82 LDA ,X *NAME (2 ALPHA-NUMERIC CHARACTERS) AND
3348 99D3 1F 03 TFR D,U *STORE THEM IN U
3349 99D5 90 1D LDV ENTRY GET START OF ARRAYS
3350 99D7 9C 1F L98D7 CMPP ENTRYEND COMPARE TO END OF ARRAYS
3351 99D9 1B 27 0D LEA L844A FC ERROR IF UNDEFINED ARRAY
3352 99DD 11 A3 84 CMPP,X COMPARE TARGET NAME TO ARRAY NAME
3353 99EB 27 06 BEQ L98EB RETURN IF CORRECT ARRAY FOUND
3354 99E2 EC 82 LDB $02,X *GET OFFSET TO NEXT ARRAY AND
3355 99E4 3B 08 LEAX D,X *ADD TO POINTER
3356 99E6 20 EF BRA L98D7 KEEP SEARCHING FOR MATCH
3357 99E8 08 22 L98EB LEA $2,4 MOVE POINTER TO OFFSET TO NEXT ARRAY
3358 99EA 39 RTS WASTED BYTE
3359 99EB 39 L98EB RTS
3360
DISASSEMBLY OF EXTENDED BASIC 1.1

PAINT FORWARD TO MAX HOR COORD OR BORDER
GET AND SAVE HOR COORD

*OPPOSITE TO THE 'UP/DN' FLAG
MOVE THE HORIZONTAL COORDINATE TWO PIXELS TO THE LEFT

*CHECK FOR MORE DATA TO 'PAINT' ON THE LINE TO THE
THREE PIXELS WERE PAINTED THEN THERE IS NO NEED TO

SET VER COORD TO ZERO TO FORCE WRAP AROUND
BRANCH IF NOT GREATER - PROCESS LINE

COMPARE TO MAXIMUM VER COORD
SAVE 'UP/DN FLAG'
TEMP STORE STACK POINTER
GET PAINT ACTIVE COLORS BACK
GET BORDER COLOR ALL PIXEL BYTE
GET CURRENT CHARACTER FROM INPUT LINE
SAVE THEM ON STACK
NORMALIZE THE HOR, VER COORDINATES

*SYNTAX CHECK FOR '(', TWO EXPRESSION, AND ')'.
CHECK FOR @ SIGN

*UP/DN FLAG UP=1; DOWN=$FF
'PAINT' TOWARD MAX HOR COORD
BRANCH IF NO PAINTING DONE - HIT BORDER INSTANTLY

GET NORMALIZED MAX HOR/VER VALUES - RETURN RESULT IN VD3, VD5
GET PAINT ACTIVE COLORS BACK
PAINT COLOR TO BACKGROUND

EVALUATE THE BORDER COLOR
GET COLOR BORDER ALL PIXEL BYTE

* STORE A BLOCK OF PAINT DATA ON THE STACK WHICH
*THE CRIA WILL CAUSE THE UP/ON FLAG TO BE ZERO WHICH IS
*USED AS A FLAG TO EXIT THE PAINT ROUTINE

GET NORMALIZED MAX HOR/VER VALUES - RETURN RESULT IN VD3, VD5
POINT U TO THE ROUTINE WHICH WILL SELECT A PIXEL

"PAINT" THE FIRST HORIZONTAL LINE FROM THE START COORDINATES

SAVE IT
PAINT FROM CURRENT HOR COORD TO ZERO
BRANCH IF NO PAINTING DONE - HIT BORDER INSTANTLY
PAINT TOWARD MAX HOR COORD

SAVE ‘UP/DN FLAG’
TEMP SAVE STACK
SEE IF 'PAINTED' COLOR IS DIFFERENT THAN ORIGINAL COLOR
TEMP STORE STACK POINTER
GET PAINT ACTIVE COLORS BACK

ADD ONE TO START HOR COORD - 1
PUT IT AT CURRENT HOR COORD ADDR
LENGTH OF PARENT LINE
SAVE UP/ON FLAG
BRANCH IF UP/ON FLAG = OFF
BRANCH IF UP/ON FLAG = ON

INCREMENT VER COORD
COMPARE TO MAXIMUM VER COORD
BRANCH IF NOT GREATER - PROCESS LINE
SET VER COORD TO ZERO TO FORCE WRAP AROUND
TEST VER COORD

*PROCESS ANOTHER BLOCK OF PAINT DATA IF
*WRAP AROUND - DISCARD ANY LINE BELOW
*VER COORD = 0 OR ABOVE MAX VER COORD
DEC VER COORD - WE ARE TESTING UP/ON FLAG = UP IF THERE

PROCESS A HORIZ LINE THAT WAS STORED ON STACK - LIMIT CHECKS HAVE BEEN DONE
PAINT FROM HOR COORD TO ZERO OR BORDER
IF NUMBER OF PAINTED PIXELS = 0, COMPLEMENT LENGTH
*SEE IF = 3 PIXELS WERE PAINTED - IF FEWER THAN
*THREE PIXELS WERE PAINTED THEN THERE IS NO NEED TO
*CHECK FOR MORE DATA TO PAINT ON THE LINE TO THE
*LEFT OF THE CURRENT POSITION IN THE OPPOSITE
*DIRECTION THAT THE UP/ON FLAG IS CURRENTLY SET TO
BRANCH IF NO NEED TO PAINT FOR PAINTABLE DATA
MOVE THE HORIZONTAL COORDINATE TWO PIXELS TO THE LEFT
BRANCH IF UP/ON FLAG = OFF

SAVE A BLOCK OF PAINT DATA IN THE DIRECTION
*OPPOSITE TO THE UP/ON FLAG
CONTINUE PAINTING LINE TO THE RIGHT
SAVE A BLOCK OF PAINT DATA IN THE SAME
*DIRECTION AS THE UP/ON FLAG

* THIS CODE WILL ENSURE THAT THE CURRENT LINE IS
*EXAMINED TO THE RIGHT FOR 'PAINTABLE' PIXELS FOR
*A LENGTH EQUAL TO THE LENGTH OF THE 'PARENT' LINE

COMPLEMENT LENGTH OF LINE JUST PAINTED
PAINT FROM HOR COORD TO ZERO OR BORDER
ADD TO LENGTH OF PARENT LINE
SAVE DIFFERENCE OF LINE JUST PAINTED AND PARENT LINE
BRANCH IF PARENT LINE IS SHORTER
INCHor COORD
CHECK FOR BORDER COLOR

NOT BORDER COLOR -
* LINE AND KEEP LOOKING FOR HORIZ BORDER COLOR
* CHECK FOR MORE DATA TO PAINT ON THE LINE TO THE
*LEFT OF THE CURRENT POSITION IN THE OPPOSITE
*DIRECTION THAT THE UP/ON FLAG IS CURRENTLY SET TO
BRANCH IF NO NEED TO PAINT FOR PAINTABLE DATA
MOVE THE HORIZONTAL COORDINATE TWO PIXELS TO THE LEFT
BRANCH IF UP/ON FLAG = OFF

SAVE A BLOCK OF PAINT DATA IN THE DIRECTION
*OPPOSITE TO THE UP/ON FLAG
CONTINUE PAINTING LINE TO THE RIGHT
SAVE A BLOCK OF PAINT DATA IN THE SAME
*DIRECTION AS THE UP/ON FLAG

* THIS CODE WILL ENSURE THAT THE CURRENT LINE IS
*EXAMINED TO THE RIGHT FOR 'PAINTABLE' PIXELS FOR
*A LENGTH EQUAL TO THE LENGTH OF THE 'PARENT' LINE

COMPLEMENT LENGTH OF LINE JUST PAINTED
PAINT FROM HOR COORD TO ZERO OR BORDER
ADD TO LENGTH OF PARENT LINE
SAVE DIFFERENCE OF LINE JUST PAINTED AND PARENT LINE
BRANCH IF PARENT LINE IS SHORTER
INCHor COORD
CHECK FOR BORDER COLOR

NOT BORDER COLOR -
* LINE AND KEEP LOOKING FOR HORIZ BORDER COLOR
* CHECK FOR MORE DATA TO PAINT ON THE LINE TO THE
*LEFT OF THE CURRENT POSITION IN THE OPPOSITE
*DIRECTION THAT THE UP/ON FLAG IS CURRENTLY SET TO
BRANCH IF NO NEED TO PAINT FOR PAINTABLE DATA
MOVE THE HORIZONTAL COORDINATE TWO PIXELS TO THE LEFT
BRANCH IF UP/ON FLAG = OFF

SAVE A BLOCK OF PAINT DATA IN THE DIRECTION
*OPPOSITE TO THE UP/ON FLAG
CONTINUE PAINTING LINE TO THE RIGHT
SAVE A BLOCK OF PAINT DATA IN THE SAME
*DIRECTION AS THE UP/ON FLAG

* THIS CODE WILL ENSURE THAT THE CURRENT LINE IS
*EXAMINED TO THE RIGHT FOR 'PAINTABLE' PIXELS FOR
*A LENGTH EQUAL TO THE LENGTH OF THE 'PARENT' LINE

COMPLEMENT LENGTH OF LINE JUST PAINTED
PAINT FROM HOR COORD TO ZERO OR BORDER
ADD TO LENGTH OF PARENT LINE
SAVE DIFFERENCE OF LINE JUST PAINTED AND PARENT LINE
BRANCH IF PARENT LINE IS SHORTER
INCHor COORD
CHECK FOR BORDER COLOR

NOT BORDER COLOR -
* LINE AND KEEP LOOKING FOR HORIZ BORDER COLOR
* CHECK FOR MORE DATA TO PAINT ON THE LINE TO THE
*LEFT OF THE CURRENT POSITION IN THE OPPOSITE
*DIRECTION THAT THE UP/ON FLAG IS CURRENTLY SET TO
BRANCH IF NO NEED TO PAINT FOR PAINTABLE DATA
MOVE THE HORIZONTAL COORDINATE TWO PIXELS TO THE LEFT
BRANCH IF UP/ON FLAG = OFF

SAVE A BLOCK OF PAINT DATA IN THE DIRECTION
*OPPOSITE TO THE UP/ON FLAG
CONTINUE PAINTING LINE TO THE RIGHT
SAVE A BLOCK OF PAINT DATA IN THE SAME
*DIRECTION AS THE UP/ON FLAG

* THIS CODE WILL ENSURE THAT THE CURRENT LINE IS
*EXAMINED TO THE RIGHT FOR 'PAINTABLE' PIXELS FOR
*A LENGTH EQUAL TO THE LENGTH OF THE 'PARENT' LINE

COMPLEMENT LENGTH OF LINE JUST PAINTED
PAINT FROM HOR COORD TO ZERO OR BORDER
ADD TO LENGTH OF PARENT LINE
SAVE DIFFERENCE OF LINE JUST PAINTED AND PARENT LINE
BRANCH IF PARENT LINE IS SHORTER
INCHor COORD
CHECK FOR BORDER COLOR

NOT BORDER COLOR -
* LINE AND KEEP LOOKING FOR HORIZ BORDER COLOR
* CHECK FOR MORE DATA TO PAINT ON THE LINE TO THE
*LEFT OF THE CURRENT POSITION IN THE OPPOSITE
*DIRECTION THAT THE UP/ON FLAG IS CURRENTLY SET TO
BRANCH IF NO NEED TO PAINT FOR PAINTABLE DATA
MOVE THE HORIZONTAL COORDINATE TWO PIXELS TO THE LEFT
BRANCH IF UP/ON FLAG = OFF

SAVE A BLOCK OF PAINT DATA IN THE DIRECTION
*OPPOSITE TO THE UP/ON FLAG
CONTINUE PAINTING LINE TO THE RIGHT
SAVE A BLOCK OF PAINT DATA IN THE SAME
*DIRECTION AS THE UP/ON FLAG

* THIS CODE WILL ENSURE THAT THE CURRENT LINE IS
*EXAMINED TO THE RIGHT FOR 'PAINTABLE' PIXELS FOR
*A LENGTH EQUAL TO THE LENGTH OF THE 'PARENT' LINE

COMPLEMENT LENGTH OF LINE JUST PAINTED
PAINT FROM HOR COORD TO ZERO OR BORDER
ADD TO LENGTH OF PARENT LINE
SAVE DIFFERENCE OF LINE JUST PAINTED AND PARENT LINE
BRANCH IF PARENT LINE IS SHORTER
INCHor COORD
CHECK FOR BORDER COLOR

NOT BORDER COLOR -
* LINE AND KEEP LOOKING FOR HORIZ BORDER COLOR
* CHECK FOR MORE DATA TO PAINT ON THE LINE TO THE
*LEFT OF THE CURRENT POSITION IN THE OPPOSITE
*DIRECTION THAT THE UP/ON FLAG IS CURRENTLY SET TO
BRANCH IF NO NEED TO PAINT FOR PAINTABLE DATA
MOVE THE HORIZONTAL COORDINATE TWO PIXELS TO THE LEFT
BRANCH IF UP/ON FLAG = OFF

SAVE A BLOCK OF PAINT DATA IN THE DIRECTION
*OPPOSITE TO THE UP/ON FLAG
CONTINUE PAINTING LINE TO THE RIGHT
SAVE A BLOCK OF PAINT DATA IN THE SAME
*DIRECTION AS THE UP/ON FLAG

* THIS CODE WILL ENSURE THAT THE CURRENT LINE IS
*EXAMINED TO THE RIGHT FOR 'PAINTABLE' PIXELS FOR
*A LENGTH EQUAL TO THE LENGTH OF THE 'PARENT' LINE

COMPLEMENT LENGTH OF LINE JUST PAINTED
PAINT FROM HOR COORD TO ZERO OR BORDER
ADD TO LENGTH OF PARENT LINE
SAVE DIFFERENCE OF LINE JUST PAINTED AND PARENT LINE
BRANCH IF PARENT LINE IS SHORTER
INCHor COORD
CHECK FOR BORDER COLOR

NOT BORDER COLOR -
* LINE AND KEEP LOOKING FOR HORIZ BORDER COLOR
* CHECK FOR MORE DATA TO PAINT ON THE LINE TO THE
*LEFT OF THE CURRENT POSITION IN THE OPPOSITE
*DIRECTION THAT THE UP/ON FLAG IS CURRENTLY SET TO
BRANCH IF NO NEED TO PAINT FOR PAINTABLE DATA
MOVE THE HORIZONTAL COORDINATE TWO PIXELS TO THE LEFT
BRANCH IF UP/ON FLAG = OFF

SAVE A BLOCK OF PAINT DATA IN THE DIRECTION
*OPPOSITE TO THE UP/ON FLAG
CONTINUE PAINTING LINE TO THE RIGHT
SAVE A BLOCK OF PAINT DATA IN THE SAME
*DIRECTION AS THE UP/ON FLAG

* THIS CODE WILL ENSURE THAT THE CURRENT LINE IS
*EXAMINED TO THE RIGHT FOR 'PAINTABLE' PIXELS FOR
*A LENGTH EQUAL TO THE LENGTH OF THE 'PARENT' LINE

COMPLEMENT LENGTH OF LINE JUST PAINTED
PAINT FROM HOR COORD TO ZERO OR BORDER
ADD TO LENGTH OF PARENT LINE
SAVE DIFFERENCE OF LINE JUST PAINTED AND PARENT LINE
BRANCH IF PARENT LINE IS SHORTER
INCHor COORD
CHECK FOR BORDER COLOR

NOT BORDER COLOR -
* LINE AND KEEP LOOKING FOR HORIZ BORDER COLOR
* CHECK FOR MORE DATA TO PAINT ON THE LINE TO THE
*LEFT OF THE CURRENT POSITION IN THE OPPOSITE
*DIRECTION THAT THE UP/ON FLAG IS CURRENTLY SET TO
BRANCH IF NO NEED TO PAINT FOR PAINTABLE DATA
MOVE THE HORIZONTAL COORDINATE TWO PIXELS TO THE LEFT
BRANCH IF UP/ON FLAG = OFF

SAVE A BLOCK OF PAINT DATA IN THE DIRECTION
*OPPOSITE TO THE UP/ON FLAG
CONTINUE PAINTING LINE TO THE RIGHT
SAVE A BLOCK OF PAINT DATA IN THE SAME
*DIRECTION AS THE UP/ON FLAG
* THE RIGHT THAN THE PARENT LINE AND PUT A BLOCK OF
3458  * PAINT DATA ON THE STACK IF IT IS MORE THAN 2 PIXELS
3459  * PAST THE END OF THE PARENT LINE
3460  9908 B0 95 86 L9908 JSR L9904 INC CURRENT HORIZONTAL COORD
3461  99083 B 88 LEAX D,X * POINT T TO THE RIGHT END OF THE PARENT LINE
3462  99091 90 BD STX HORBEGIN * AND SAVE IT AS THE CURRENT HORIZ COORD
3463  99093 43 COMB = ACCO CONTAINS A NEGATIVE NUMBER CORRESPONDING TO
3464  99094 53 COMB = THE NUMBER OF PIXELS THE CURRENT LINE EXTENDS
3465  99095 83 0 81 SMAN = PAST THE RIGHT END OF THE PARENT LINE. CONVERT
3466  99099 2F 84 BLE L9994 = TO A POSITIVE NUMBER AND BRANCH IF THE LINE DOESN'T EXTEND
3467  9909A 1F 81 TFR D,X *SAVE THE PORTION OF THE LINE TO THE RIGHT OF THE PARENT LINE
3468  9909A 85 85 *AS THE LENGTH
3469  9909C BD 83 BSR L99A1 =SAVE A BLOCK OF PAINT DATA IN THE DIRECTION OPPOSITE THE
3470  * =CURRENT UP/DOWN FLAG
3471  > 999E 7F 99 34 L999E JMP L9934 PROCESS MORE PAINT DATA BLOCKS
3472  *
3473  * BLOCKS OF PAINT DATA ARE STORED ON THE STACK SO THAT PAINT
3474  * CAN REMEMBER WHERE IT SHOULD GO BACK TO PAINT UP OR DOWN
3475  * FROM THE CURRENT LINE IT IS PAINTING. THESE BLOCKS OF DATA
3476  * REPRESENT HORIZONTAL LINES ABOVE OR BELOW THE CURRENT LINE
3477  * BEING PAINTED AND REQUIRE SIX BYTES OF STORAGE ON THE STACK.
3478  * THE DATA ARE STORED AS FOLLOWS: 0,SUP/DOWN FLAG; 1,INDEX COORD
3479  * OF LINE; 2,3,4,HORIZONTALS COORD OF LINE; 5,6,LENGTH OF LINE
3480  *
3481  * SAVE A BLOCK OF PAINT DATA FOR A LINE IN THE
3482  * OPPOSITE DIRECTION OF THE CURRENT UP/DOWN FLAG
3483  99A1 DD CB L99A1 STD VCB SAVE NUMBER PIXELS PAINTED
3484  99A4 3 8 CB F3 D8 XOR X,Y PUT RETURN ADDR IN Y
3485  99A5 DC 83 LD SHORBEGIN GET HORIZONTAL START COORDINATE
3486  99A7 34 16 PSHS X,8,A PUT ON STACK
3487  99A9 96 87 LDS V=6 GET UP/DOWN FLAG
3488  99AA 4B NEG REVERSE THE UP/DOWN FLAG
3489  99AC 6C 80 L99AC LDB VERBEGIN+1 GET VERTICAL START COORDINATE
3490  99AE 3 8 86 PSHS B,A SAVE VERTICAL START COORDINATE AND U/D FLAG
3491  99A9 BB 80 PUT RETURN ADDR BACK ON STACK
3492  *
3493  * CODE BELOW CHECKS FOR ABILITY TO STORE FOUR BYTES IN
3494  * FREE RAM, HOWEVER THE PAINT ROUTINE WILL STORE SIX
3495  * BYTES IN FREE RAM - FIRST INSTRUCTION SHOULD BE LDR #3
3496  99B2 C6 #2 LDB #92 * CHECK TO SEE IF THERE'S ENOUGH FREE
3497  99B4 BD 33 JSR LAC33 * RAM FOR 4 BYTES TEMPERATURE STORAGE
3498  99B7 DC CB LDB VCB GET LENGTH OF RIGHT PAINTED LINE
3499  99BA 39 RTS
3500  *
3501  * SAVE A BLOCK OF PAINT DATA FOR A LINE IN
3502  * THE SAME DIRECTION AS THE CURRENT UP/DOWN FLAG
3503  99BA DD CB L99BA STD VCB SAVE LENGTH OF HORIZONTAL PAINTED LINE
3504  99BC 35 82 PULS Y SAVE RETURN ADDRESS IN Y
3505  99BD BC 83 LDS HOREND HORIZONTAL START COORDINATE
3506  99BE 34 16 PSHS X,8,A SAVE HORIZONTAL START COORDINATE AND LENGTH
3507  99C2 96 87 LDS V=6 GET UP/DOWN FLAG (1 OR -1)
3508  99C4 2B 66 BRS L99AC SAVE THE PAINT DATA ON THE STACK
3509  99C6 9E 8D L99C6 LDS HOREND GET CURRENT HORIZONTAL COORD
3510  99C8 9F 83 STX HOREND SAVE IT
3511  99CA 39 RTS
3512  *
3513  * GO HERE TO FINISH PAINTING TO RIGHT AFTER YOU HAVE PAINTED TO THE LEFT
3514  99CB DD CB L99CB STD VCB SAVE COUNT OF THE NUMBER OF PAINTED PIXELS.
3515  99CD 1F 8E LDS Y=0 GET LAST HORIZONTAL COORD
3516  99DE BD 44 BSR L996C *SAVE CURRENT HORIZONTAL COORD - NOW HOREND CONTAINS COORDINATE
3517  99D2 1F 8D STX HORBEGIN START PAINTING TO RIGHT FROM THE LEFT PAINT START COORD
3518  99D5 BD 11 BSR L99BE PAINT TOWARDS THE RIGHT
3519  99D7 9E 8D LDS VCB GET THE NUMBER OF PAINTED PIXELS WHEN GOING TOWARDS LEFT PIXELS
3520  99D9 3 8 BB BRS L99AC ADD TO NUMBER PAINTED GOING TOWARDS RIGHT
3521  99DB C3 #1 ADD #1 ADD 1 TO PAINT COUNT TOWARDS RIGHT - ACC = LENGTH OF PAINTED LINE
3522  *
3523  99DE 39 RTS
3524  *
3525  * PAINT FROM HORIZ COORD TO ZERO OR HORIZ BOUNDARY
3526  99DF BD 99 C6 L99DF JSR L99C6 PUT STARTING HORIZONTAL COORD IN HOREND
3527  99F2 8B 95 14 L99F2 LMBL9514 (DECK HORIZ COORD ADDRESS) TO Y
3528  99FA 36 86 BRA L99EE GO PAINT THE LINE
3529  *
3530  * PAINT FROM HORIZ COORD TO MAX HORIZ COORD OR HIT BOUNDARY
3531  99FB BD 99 66 L99FB JSR L99C6 PUT INCHR HORIZ COORD ADDR IN Y
3532  99FC AD 4A JSR L992E INC HORIZ COORD - THE LEFT PAINT ROUTINE PAINTED THE FIRST COORD
3533  99FD 9E 8A LDS Y=0 ZERO U = REG - INITIAL PIXEL PAINT COUNTER
3534  99FE 9E 8D LDS HOREND GET HORIZ COORD
3535  99FF 2B 17 L99FF BRANCH IF HORIZ COORD IS > 8 OR < 0
3536  99F4 9C 83 CMPX V=3 COMPARE CURRENT HORIZ COORD TO MAX VALUE
3537  99F6 22 13 BHI L9AB BRANCH IF > MAX
3538  99F8 34 68 PSHS U,Y SAVE PAINT COUNTER, INC/DEC POINTER
3539  99FA BD 16 BRS L9A12 CHECK FOR BORDER PIXEL
3540  99FC 27 #B BQD L9AB HIT BORDER
3541  99FD BD 97 77 LDS Y=0 SET PIXEL TO PAINT COLOR - PAINTING IS DONE HERE
3542  9A01 35 68 PULS Y,U RESTORE PAINT COUNTER AND INC/DEC POINTER
3543  9A03 33 41 L9A1 U=0 ADDED ONE TO PAINT COUNTER
3544  9A04 AD 4A JSR Y INC OR DEC HORIZ COORD DEPENDING ON CONTENTS OF Y
3545  9A07 2B 99 BRA L99F2 KEEP PAINTING THE LINE
3546  9A09 35 68 L9A9 PULS Y,U RESTORE PAINT COUNTER AND INC/DEC POINTER
3547  9A0B 1F 81 L9AA TFR U,D SAVE PAINT COUNTER IN ACC
3548  9A0D 1F 81 TFR X,D SAVE PAINT COUNTER IN X
3549  9A0F 93 8A SUB ZERO SUB FLAGS ACCORDING TO CONDITION OF PAINT COUNTER
3550  9A11 39 RTS
3551  * CHECK FOR BORDERS COLOR - ENTER W/VD9 CONTAINING
3552  * ADDRESS OF ROUTINE TO GET ABS SCREEN ADDRESS
* AND PIXEL MASK - EXIT WITH Z = 1 IF HIDE BORDER COLOR PIXEL

3554  9A12  AD 9F  #0 D9   L9A12  JSR  (V09) GET THE ADDR AND PIXEL MASK

3555  9A16  1F 89   TFR  A, B COPY PIXEL MASK TO ACCB

3556  9A18  DA 08   AND V08 AND PIXEL MASK & BORDER COLOR; ACCB = ONE PIXEL OF BORDER COLOR

3557  9A1A  34 86   PSHS B, A PUSH MASK AND BORDER PIXEL

3558  9A1C  A4 84   ANDA ,X * PUT CURRENT PIXEL DATA INTO ACCB AND

3559  9A1E  A1 61   CMPA R1, 5 * COMPARE IT TO BORDER COLOR; Z FLAG = 1 IF MATCH

3560  9A28  35 86   PULS A,B,FC RESTORE MASK AND BORDER PIXEL - THEN RETURN

3561

3562

3563  9A22  9E 8A   PLAY LDX ZERO *DEFAULT VALUES FOR LENGTH OF PLAY AND ADDRESS

3564  9A2A  C6 81   LDY #1 *OF START OF PLAY STRING IF USED FOR PLAY (NULL STRING)

3565  9A26  34 14   PSHS X,B SAVE DEFAULT VALUES

3566  9A2B  80 56   JSR L9B16 EVALUATE EXPRESSION

3567  9A2B  5F   CLRB

3568  9A2C  B0 42   JSR L9A92 * SET UP DA TO PASS THROUGH ANA MIX

3569  9A2F  B0 76   JSR L9A76 ENABLE ANA MIX

3570  9A32  BD 64   L9A32 JSR L9B54 *POINT X TO START OF PLAY STRING AND PUT LENGTH

3571

3572

3573  9A35  2B 82   BRA L9A39 INEFFICIENT - SHOULD BE FCB SKIP

3574  9A37  35 14   L9A37 PULS B,X GET PLAY STRING START AND LENGTH

3575  9A39  D7 0B   STB V0B LENGTH OF PLAY COMMAND

3576  9A3B  27 FA   BEQ L9A37 GET NEW STRING DATA IF LENGTH = 0

3577  9A3D  9F D9   STX V09 START OF PLAY STRING

3578  9A3F  1B 31   L9A43 LBEQ L9A74 DISABLE ANA MIX AND RETURN IF X = 0

3579  9A45  27 F0   BEQ L9A37 GET NEW DATA IF 50

3580  9A47  BD 9B   JSR L9B95 GET A COMMAND CHARACTER IF NOT

3581  9A4A  B1 3B   CMPA #';' SUB COMMAND TERMINATED

3582  9A4C  27 F5   BEQ L9A43 IGNORE SEMICOLONS

3583  9A4E  B1 27   CMPA #';' CHECK FOR APOSTROPHE

3584  9A58  27 F1   BEQ L9A43 IGNORE THEM TOO

3585  9A52  B1 58   CMPA #"X" CHECK FOR AN EXECUTABLE STRING

3586  9A54  1B 81   L9A5A GET PROCESS SUB COMMAND

3587  9A5B  8D 02   BSR L9A5C CHECK FOR OTHER COMMANDS

3588  9A5A  2E 87   BRA L9A43 KEEP GOING THROUGH INTERPRETATION LOOP

3589

3590  9A5C  27 8F   L9A5C CMPA #"0" ADJUST OCTAVE?

3591  9A5E  26 0D   BNE L9A60 NO

3592  9A68  06 0E   LOD OCTAVE GET CURRENT OCTAVE

3593  9A62  5C   INCB LEGAL VALUES ARE 1-5 BUT INTERNALLY THE COMPUTER USES 0-4

3594  9A63  6B 5B   BSR L9A4C MODIFIER CHECK

3595  9A65  5A   DECB COMPENSATE FOR INCB ABOVE

3596  9A66  C1 84   CMPB #184 MAXIMUM VALUE OF 4

3597  9A68  22 63   BHI L9A4C FC ERROR

3598  9A6A  D7 0E   STB OCTAVE SAVE NEW VALUE OF OCTAVE

3599  9A6C  39   RTS

3600

3601  9A6D  B1 56   L9A6D CMPA #"V" ADJUST VOLUME?

3602  9A6F  26 1A   BNE L9A6B NO

3603  9A71  06 0F   LOD VOLH GET CURRENT HIGH VOLUME LIMIT

3604  9A73  54   LSRR *SHIFT 2 BITS TO RIGHT; DA IS ONLY 6 BITS (BIT 2 - BIT 7) -

3605  9A76  54   LDD SB5 *TO MANIPULATE THE DATA IT MUST BE IN BITS 5-7

3606  9A76  C1 8F   SUBB #31 SUBTRACT OUT MID VALUE OFFSET

3607  9A77  BD 47   BSR L9A4C MODIFIER CHECK

3608  9A79  C1 1F   CMPB #43 MAXIMUM ALLOWED RANGE IS 31

3609  9A7B  22 5F   BHI L9A4C FC ERROR

3610  9A7D  58   ASLB

3611  9A7E  5B   ASLR

3612  9A7F  24 84   PSHX B *MOVE NEW VALUE BACK TO BITS 2-7

3613  9A81  CC 7E   LDD #5767E PUT MID VALUE IN HIGH AND LOW LIMIT

3614  9A84  AB 5D   ADSR ,5 ADD NEW VOLUME TO HIGH LIMIT

3615  9A86  EA 8B   SUBT ,5 SUBTRACT NEW VOLUME FROM LOW LIMIT

3616  9A88  DD 0F   STD VOLH SAVE NEW VOLUME LIMITS

3617  9A8A  39   RTS

3618

3619  9A8B  B1 4C   L9A8B CMPA #"L" SET NOTE LENGTH?

3620  9A8D  26 23   BNE L9A82 NO

3621  9A8F  D6 01   LOD NOTEIN GET CURRENT LENGTH

3622  9A91  BD 2D   BSR L9A4C MODIFIER CHECK

3623  9A93  5D   TSTB *

3624  9A94  27 37   BEQ L8ACD * FC ERROR IF LENGTH = 0

3625  9A96  D7 01   STB NOTEIN SAVE NEW NOTE LENGTH

3626  9A98  0F 05   CLR Dotted Scale Factor

3627  9A9A  BD 03   L9A9A BSR L9A9F CHECK FOR A DOTTED NOTE

3628  9A9C  24 FC   BCC L9A9A BRANCH IF DOTTED NOTE

3629  9A9E  39   RTS

3630

3631  9A9F  0D 0B   L9A9F TST V08 CHECK COMMAND LENGTH

3632  9AA1  27 8A   BEQ L9AAD IT S EMPTY

3633  9AA3  BD 9B   JSR L9B98 GET COMMAND CHARACTER

3634  9AA6  B1 2E   CMPA '#.--' CHECK FOR DOTTED NOTE

3635  9AAB  27 95   BEQ L9A9F BRANCH ON DOTTED NOTE AND CLEAR CARRY FLAG

3636  9AA7  BD 82   JSR L9B82 *MOVE COMMAND STRING POINTER BACK ONE AND ADD ONE TO

3637

3638  9AA9  43   L9AAD COMA SET CARRY FLAG

3639  9AAE  39   RTS

3640  9AFA  BC 05   L9AFA INC DOTVAL ADD ONE TO NOTE TIMER SCALE FACTOR

3641  9AB1  39   RTS

3642

3643  9AB2  B1 54   L9A82 CMPA #"T" MODIFY TEMPO?

3644  9AB4  26 0D   BNE L9A93 NO

3645  9AB6  06 62   LOD TEMPO GET CURRENT TEMPO

3646  9ABB  BD 86   BSR L9A4C EVALUATE MODIFIER

3647  9ABA  5D   TSTB SET FLGS

3648  9ABB  27 08   BEQ L9A4C FC ERROR IF IT IS 0
3649 9ABD D7 E2  STR  TEMPO  SAVE NEW TEMPO
3650 9ABF 39  RTS
3651 9AC8 7E 9B AC  L9ACB  JMP  L9ABC  EVALUATE THE >,<,*,= OPERATORS
3652 9AC9 51  * PAUSE
3653 9AC9 B1 58  L9AC3  CMPA #P'  PAUSE COMMAND?
3654 9AC9 26 24  BNE  L9AEB  NO
3655 9AC7 B0 9C CB  JSR  L9AC8  EVALUATE A DECIMAL COMMAND STRING VALUE
3656 9ACA 5A  TSB  * CHECK FOR LEGAL EXPRESSION AND
3657 9ACB 26 93  BNE  L9AQB  * BRANCH IF PAUSE VALUE <= 0
3658 9ACD 7E 4A 44 L9ACD  JMP  L84AA  FC ERROR IF PAUSE <= 0
3659 9ADB 96 E5  L9ADB  LDA  DOTVAL  *SAVE CURRENT VALUE OF VOLUME AND NOTE
3660 9ACD 9E DF  LDX  VOLHI  *TIMER SCALE
3661 9ADA 34 12  PSHX X,A  *
3662 9AEB 86 7E  LDA #87E  MID VALUE OF DA CONVERTER
3663 9AED 97 DF  STA  VOLHI  *SET VOLHI = TEMP 0
3664 9AFA 97 E8  STA  VOLLow  *
3665 9AFA 8F 65  CLR  DOTVAL  RESET NOTE TIMER SCALE FACTOR
3666 9AFA 8E B7  BSR  L9A77  GO PLAY A NOTE OF # VOLUME
3667 9AFA 35 12  PULS A,X  *
3668 9AB2 9E 55  STA  DOTVAL  *RESTORE VALUE OF VOLUME
3669 9AB4 9F 05  STX  VOLHI  *AND NOTE TIMER SCALE.
3670 9AB6 39  RTS
3671 9AB7 6F E2  L9AEB  LDA  .5  PUSH NOTE NUMBER 0 ONTO STACK
3672 9AB9 2B 4B  BSR  L9B2B  GO PLAY IT
3673 * NOTE
3674 9AEB B1 4E L9AEB  CMPA #N'  LETTER N BEFORE THE NUMBER OF A NOTE!
3675 9AED 26 93  BNE  L9AF2  NO - IT'S OPTIONAL
3676 9AFA 8F 98 9B 98  L9AR9  GET NEXT COMMAND CHARACTER
3677 9AFA 81 41  L9AF2  CMPA #A'  CHECK FOR NOTE A
3678 9AFA 25 54  BLD  L9AF9  BELOW
3679 9AFA 86 B1 47  CMPA #G'  CHECK FOR NOTE B
3680 9ABF 23 55  BLS  L9AFF  FOUND NOTE A-G
3681 9AFA 8D 8E 9E L9AFJ  JSR  L9BBE  EVALUATE DECIMAL NUMERIC EXPRESSION IN COMMAND STRING
3682 9AFA 8F 2B 23  BSR  L9B2B  PROCESS NOTE VALUE
3683 * PROCESS A NOTE HERE
3684 9AFF 8B 41  L9AFF  SUBA #A'  MASK OFF ASCII
3685 9AB0 B1 3B  LDL #X95C  LOAD X WITH NOTE JUMP TABLE
3686 9AB0 B1 56  LDB A,X  GET NOTE
3687 9AB6 80 DF  TSB VDB  ANY COMMAND CHARACTERS LEFT?
3688 9AB8 27 1B  BEQ L9B22  NO
3689 9AB9 80 98 9B 98  JSR L9AR9  GET NEXT COMMAND CHARACTER
3690 9ABD B1 23  CMPA #M'  SHARP NOTE'
3691 9ABF 27 4E  BEQ L9B15  YES
3692 9ABF B1 2B  CMPA #M'  90B1 SHARP NOTE
3693 9AB3 26 93  BNE L9B18  NO
3694 9AB5 5C  L9B15  INCB  ADD 1 TO NOTE NUMBER (SHARP)
3695 9AB6 2B 0A  BRA L9B22  PROCESS NOTE
3696 9AB8 21 2D  L9B18  CMPA #.'  FLAT NOTE?
3697 9AB9 2A 5E  BRA L9B1F  NO
3698 9ABD B1 23  CMPA #M'  COMPUTER USES 0-11
3699 9ABD B1 2B  CMPA #M'  ADJUST NOTE NUMBER, BASIC USES NOTE NUMBERS 1-12, INTERNALLY
3700 * TO COMMAND LENGTH CRT
3701 9AB2 5A  DECB  SUBTR 1 FROM NOTE NUMBER (FLAT)
3702 9ABD B1 23  CMPA #M'  COMPUTER USES 8-11
3703 * =COMPUTE USES 8-11
3704 9ABD C1 0B  CMPM #12-1  MAXIMUM NOTE VALUE
3705 9AB5 22 16  BHI L9AD0  FC ERROR IF F = 11
3706 9AB6 2B 54  PSRS B  SAVE NOTE VALUE
3707 9AB9 06 E1  LDB NOTELN  GET NOTE LENGTH
3708 9ABD B6 96 22  L9B22  LDA  TEMPO  GET TEMPO
3709 9ABD B0 3D  MUL  CALCULATE NOTE DURATION
3710 9ABE D5  STD  VOS  SAVE NOTE DURATION
3711 * THE IRQ INTERRUPT IS USED TO PROVIDE A MASTER TIMING REFERENCE FOR
3712 * THE PLAY COMMAND.  WHEN A NOTE IS DONE, THE IRQ SERVICING
3713 * ROUTINE WILL RETURN CONTROL TO THE MAIN PLAY COMMAND INTERPRETATION LOOP
3714 9ABE 33 61  LEAX #1,5  *LOAD V Internal VALUE OF (STACK POINTER+) 5 THAT THE STACK
3715 * *POINTER WILL BE PROPERLY RESET WHEN IRQ VECTORS
3716 * *YOU OUT OF THE PLAY TIMING ROUTINES BELOW
3717 9ABE 32 2C  BHI L9B64  * BRANCH IF OCTAVE > 1
3718 * OCTAVES 1 AND 2 USE A TWO BYTE DELAY TO SET THE PROPER FREQUENCY
3719 9ABE 3B 62  LDL #M962  POINT TO DELAY TABLE
3720 9ABE 3C 18  LDB #2412  24 BYTES DATA/OCTAVE
3721 9ABD 30 3D  MUL  CALC OCTAVE TABLE OFFSET
3722 9AEE 3A  AEX  POINT TO CORRECT OCTAVE TABLE
3723 9ABF 35 54  PULS B  GET NOTE VALUE BACK
3724 9ABD 5B  ASLB X 2 - 2 BYTES/NOTE
3725 9AEC 3A  AEX  POINT TO CORRECT NOTE
3726 9ABD 31 34  LEAX ,X  GET POINTER TO Y REG (TFR X,Y)
3727 9ABE 45 8D  BSR L9B8C  CALCULATE NOTE TEMPO VALUE
3728 9ABD 87 43  STD PLYTM  SAVE IT
3729 9ABE 4D 03  PLYTM  *MAIN SOUND GENERATION LOOP - ONLY THE IRQ SERVICE WILL GET YOU OUT
3730 * OF THIS LOOP (OCTAVES 1 AND 2)
3731 9ABE 8D 0C  L9B49  BSR L9B57  MID VALUE TO DA AND WAIT
3732 9ABE 96 DF  LDA VOLHI  GET HIGH VALUE
3733 9ABE 8D 8B  BSR L9B5A  STORE TO DA AND WAIT
3734 9ABF 8D 86  BSR L9B57  MID VALUE TO DA AND WAIT
3735 9ABF 96 98  LDA VOLLOW  GET LOW VALUE
3736 9ABF 98 9B  BSR L9B5A  STORE
3737 9ABF 55 4F 2F  BRA L9B49  KEEP LOOPING
3738 9ABE 87 B6  L9B57  LDA #57E  DA MID VALUE AND #5 232 MARKING
3739 9ABE 95 12  DELAY SOME - FINE TUNE PLAY FREQUENCY
3740 9ABE 8A 7F 20  L9B5A  STA PIA1  STORE TO DA CONVERTER
3741 9ABE 9D 0C  LIX ,7  GET DELAY FROM OCTAVE TABLE
3742 9ABE 9F 3F  L9B5F  LEAX #-.1,X  *
* OCTAVES 3, 4 AND 5 USE A ONE BYTE DELAY TO SET THE PROPER FREQUENCY

* CHECK FOR VARIABLE EQUATE - BRANCH IF SO; ACCB WILL BE

* DIVIDE ACCOD BY TWO - EACH INCREMENT OF DOTVAL

* GET NEXT COMMAND - RETURN VALUE IN ACCA

* THE LARGER ACC IS, THE LONGER THE NOTE WILL PLAY

* CLEAR CARRY IF NUMERIC

* GET A COMMAND CHARACTER

* CHECK FOR VARIABLE EQUATE - BRANCH IF SO; ACCB WILL BE

* STRIP A DECIMAL ASCII VALUE OFF OF THE COMMAND STRING

* AND RETURN BINARY VALUE IN ACCB

* MASK OFF ASCII

* SAVE VALUE TEMPORARILY

* SAVE BASE 10

* MULTIPLY BY DIGIT

* ADD INPUT TO COMMANDRY VALUE

* ADD INPUT TO COMMANDRY VALUE

* ADD INPUT TO COMMANDRY VALUE

* MOVE COMMAND STRING BACK ONE
DISASSEMBLY OF EXTENDED BASIC 1.1

3841 9BFR 27 F1 BEQ L98EB * FC ERROR IF DIVIDING BY ZERO
3842 9BFA 54 LSRR DIVIDE BY TWO
3843 9BFB 39 RTS
3844 9BFC 50 L98FC TSTB *
3845 9BFD 2B EC BMI L98EB * FC ERROR IF RESULT WOULD BE > 255
3846 9BF7 58 ASL8 MULIPLY BY TWO
3847 9BFB 39 RTS
3848 9C01 34 68 L9C01 PSHS U,Y SAVE U,Y REGISTERS
3849 9C03 8D 16 BNE L9C10 INTERPRET COMMAND STRING AS IF IT WERE A BASIC VARIABLE
3850 9C05 8D 87 8E JSR L87BE CONVERT FPA8 TO AN INTEGER VALUE IN ACCB
3851 9C08 35 88 PULS Y,PC RESTORE U,Y REGISTERS AND RETURN
3852 > 9C0A 8D 18 L9C0A JSR L9C18 EVALUATE AN EXPRESSION IN THE COMMAND STRING
3853 9C0D C6 02 LDB #2 =
3854 9C0F 8D AC 33 JSR LAC33 =ROOM FOR 4 BYTES ON STACK?
3855 9C12 D6 08 LDB VD # GET THE CURRENT COMMAND LENGTH AND REGISTER AND
3856 9C14 9E D9 LDX VD * SAVE THEM ON THE STACK
3857 9C16 34 14 PSHS X,R *
3858 9C18 7E 9A 32 JMP L9A32 GO INTERPRET AND PROCESS THE NEW PLAY SUB COMMAND
3859 9C18 7E 9A 32

3860 * INTERPRET THE PRESENT COMMAND STRING AS IF IT WERE A BASIC VARIABLE
3861 9C18 9E D9 L9C18 LDX VD9 GET COMMAND POINTER
3862 9C1D 34 10 PSHS X SAVE IT
3863 9C1F 8D 98 9B JSR L98BA GET A COMMAND CHARACTER
3864 9C22 8D 83 A2 JSR LB3A2 SET CARRY IF NOT ALPHA
3865 9C25 25 C4 BLD L98BE FC ERROR IF NOT ALPHA - ILLEGAL VARIABLE NAME
3866 9C27 8D 98 9B L9C27 JSR L98BA GET A COMMAND CHARACTER
3867 9C2A B1 3B CMPA #':;' CHECK FOR DECIMAL - COMMAND SEPARATOR
3868 9C2C 26 F9 BNE L9C27 BRANCH UNTIL FOUND
3869 9C2E 35 10 PULS X GET THE CURRENT COMMAND POINTER
3870 9C30 8E A6 LDU CHARAD GET BASIC S INPUT POINTER
3871 9C32 34 48 PSHS U SAVE IT
3872 9C34 9F A6 STX CHARAD PUT PLAY COMMAND POINTER IN PLACE OF BASIC S INPUT POINTER
3873 9C36 8D B2 84 JSR LB3B4 EVALUATE AN ALPHABET EXPRESSION 4 NEW STRING DESCRIPTOR
3874 9C39 35 10 PULS X RESTORE BASIC S INPUT POINTER
3875 9C3B 9F A6 STX CHARAD *
3876 9C3D 39 RTS

3877 * MORE OF EXTENDED BASIC S IRQ ROUTINE
3878 *

3880 9C3E 4F L9C3E CLR A CLEAR ACCA
3881 9C3F 8F 0B TFR A,DP SET THE DIRECT PAGE TO ZERO
3882 9C41 8C 83 LDB 10 BYTMR GET THE PLAY TIMER
3883 9C43 10 27 70 74 LBEQ L98BA BRANCH TO COLOR BASIC S IRQ ROUTINE IF ZERO
3884 9C47 8D 05 SUBD VDS SUBTRACT OUT PLAY INTERVAL
3885 9C49 D0 83 STDYMR SAVE THE NEW TIMER VALUE
3886 9C4B 22 02 BHI L9C5A BRANCH IF PLAY COMMAND NOT DONE
3887 9C4C 8F 03 CLR PLYTMN RESET MSB OF PLAY TIMER IF DONE
3888 9C4F 8F 04 CLR PLYTMN1 RESET LSB OF PLAY TIMER
3889 9C51 35 02 PULS A GET THE CONDITION CODE REG
3890 9C53 18 EE 67 LDS #57,S *LOAD THE STACK POINTER WITH THE CONTENTS OF THE U REGISTER
3891 * *WHICH WAS STACKED WHEN THE INTERRUPT WAS_HONORED.
3892 9C56 B4 7F ANDA #$7F CLEAR E FLAG - MAKE COMPUTER THINK THIS WAS AN IRQ
3893 9C58 8D 39 RTS SAVE CONDITION CODE
3894 9C5A 3B L9C5A RTI RETURN

3895 * TABLE OF NUMERICAL VALUES FOR LETTERS 06
3896 9C5B 8A 0C 03 05 06 L9C5B FCB 18,12,1,3,5,6,8 NOTES A,B,C,D,E,F,G
3897 9C61 8B*

3898 * TABLE OF DELAYS FOR OCTAVE 1
3899 9C62 #8 #A 91 90 01 7A L9C62 FDB #$1B,#10,#00,#07A DELAYS FOR OCTAVE 1
3899 9C66 #1 #A 51 50 01 3D FDB #$15A,#050,#03D
3899 9C66 #1 #B 1A 01 9A FDB #$12B,#011,#08A
3899 9C7D ## #F #D #D #D #D #D FDB #$08F,#00ED,#00DF
3899 9C8 E

3900 * TABLE OF DELAYS FOR OCTAVE 2
3901 9C7A #D #D #C 07 08 8B L9C7A FDB #$0D3,#00C7,#008B DELAYS FOR OCTAVE 2
3902 9C88 #B #1 0A 06 0D FDB #$088,#006A,#009D
3903 9C8C #B #4 08 00 83 FDB #$094,#0083,#0003
3904 9C8C #C #C 07 70 60 61 FDB #$07C,#0075,#006E
3905 9C9B * TABLE OF DELAYS FOR OCTAVES 3,4,5
3906 9C98 #9 #C 03 08 83 79 L9C98 FCB #$94,#93,#92,#91,#90 DELAYS FOR OCTAVES 3,4,5
3907 9C98 74 67 61 58 56 FCB #74,#70,#67,#61,#58,#56
3908 9C9E 51 47 43 3F 3B FCB #51,#44,#47,#43,#3F,#3B
3909 9C9E 2B 34 31 0E 2B FCB #37,#34,#31,#0B,#2B
3909 9C9E 2B 23 21 1F 18 FCB #27,#23,#21,#12,#18
3910 9C9E 19 18 16 14 12 FCB #19,#18,#16,#14,#12
3911 9C9E 2B

3912 9CB 89 8A DRAW LDY ZERO * X=0, ACB=1; END OF DRAW COMMAND LINE VALUES -
3913 9CB 81 #A8 #9B * WHEN THESE VALUES ARE PULLED OFF THE
3914 9CB 8A 14 PSHS X,R * STACK, THE DRAW COMMAND WILL END
3914 9CB C7 22 STB SETFLG SET PSET/PRESET FLAG TO PSET
3915 9CBE 8F 05 STX VDS CLEAR UPDATE FLAG AND DRAW FLAG
3916 9CB8 BD 95 9A JSR L959A SET ACTIVE COLOR BYTE
3917 9CBE BD 81 56 JLR LB156 EVALUATE EXPRESSION
3918 9CC6 8D 96 64 L9CC6 JSR L9664 GET THE LENGTH AND ADDRESS OF THE COMMAND STRING
3919 9CC9 28 #8 BRA L9C33 INTERPRET THE COMMAND STRING
3920 9CCB 8B 98 L9CCB JSR L98BA GET NEXT CHARACTER FROM COMMAND LINE
3921 9CCE 7E 9B 9E JMP L9BEB EVALUATE A DECIMAL VALUE IN COMMAND LINE

841
ADJUST COLOR CODE FOR PROPER PMODE
SAVE NEW FOREGROUND COLOR
SET COLOR BYTES (WCOLD,ALCOL)
GO PROCESS ANOTHER COMMAND
* CHANGE ANGLE
ONLY 0-3 ARE LEGAL
FC ERROR IF ANGLE NUMBER > 3
SAVE DRAW ANGLE
GO PROCESS ANOTHER COMMAND
* CHANGE SCALE
ONLY 0-63 ARE LEGAL
FC ERROR IF SCALE > 63
SAVE DRAW SCALE
GO PROCESS ANOTHER COMMAND
* 315 DEGREES
*NEGATE ACCO - MAKE HORIZONTAL
SKIP ONE BYTE - KEEP HORIZONTAL DIFFERENCE NEGATIVE
CLEAR MS BYTE OF 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
EXCHANGE HORIZONTAL AND VERTICAL DIFFERENCES
GO MOVE THE DRAW POSITION
CLEAR MS BYTE OF HORIZONTAL DIFFERENCE
COPY HORIZONTAL DIFFERENCE TO VERTICAL DIFFERENCE
NEGATE ACCO - MAKE HORIZONTAL DIFFERENCE NEGATIVE
GO MOVE THE DRAW POSITION
* GO RIGHT
* THE CURRENT POSITION; DRAW A LINE AFTER THE MOVE

* MULTIPLY HOR OR VER DIFFERENCE BY SCALE FACTOR.

* DIVIDE PRODUCT BY 4 AND RETURN VALUE IN ACCD

* IN ACCD (HORIZONTAL) AND X (VERTICAL) TO

* THE CURRENT POSITION; DRAW A LINE AFTER THE MOVE

* MOVE THE DRAW POSITION; ADD THE ODD AND EVEN DIFFERENCES

* IN REGISTER AND ABSOLUTE VERTICAL POSITION IN X REGISTER.

* MOVE THE DRAW POSITION; ENTER ABSOLUTE HORIZONTAL POSITION

L9D9B CLRA CLEAR MS BYTE OF HORIZONTAL DIFFERENCE
L9D9B LDX ZERO X = OD; VERT DIFF = 0
L9D9B BRA L9DB8 BRANCH IF NO UPDATE

4087 4F CLRA *NEGATE ACCO - MAKE THE HORIZONTAL
B6 3A 03 04 SBR L9DC4 *DIFFERENCE NEGATIVE
4088 4F 07 FF 03 LDHV L9DB8 MAKE VERTICAL DIFFERENCE ZERO AND MOVE THE DRAW POSITION
4089 4F 07 FF 00 BRA L9DCB BRANCH IF NO UPDATE
408A 4F 07 FF 00 SBR L9DB8 *GO DOWN
408B 4F 07 FF 00 LDHV L9DB8 CLEAR MS BYTE OF HORIZONTAL DIFFERENCE
408C 4F 07 FF 00 BRA L9D92 *MAKE VERTICAL DIFFERENCE = 0, EXCHANGE HORIZONTAL AND
408D 4F 07 FF 00 *VERTICAL DIFFERENCES AND MOVE THE DRAW POSITION
408E 4F 07 FF 00 *GO UP
408F 4F 07 FF 00 L9D9F CLRA *NEGATE ACCO - MAKE THE HORIZONTAL
4090 4F 07 FF 00 SBR L9DC4 *DIFFERENCE NEGATIVE
4091 4F 07 FF 00 LDHV L9DB8 X = 0; HORIZ DIFF = 0
4092 4F 07 FF 00 EXG X, D EXCHANGE THE HORIZONTAL AND VERTICAL DIFFERENCES
4093 4F 07 FF 03 BRA L9DCB GO MOVE THE DRAW POSITION
4094 4F 07 FF 00 * EXECUTE A COMMAND SUB STRING
4095 4F 07 FF 00 L9DB8 JSR L9C1B INTERPRET CURRENT COMMAND AS IF IT WERE A BASIC VARIABLE
4096 4F 07 FF 00 L9DB8 #S70 =
4097 4F 07 FF 00 L9DB0 AC 33 JSR LAC33 ==FOUR BYTES OF FREE RAM LEFT?
4098 4F 07 FF 00 LDHV L9DB8 * GET CURRENT COMMAND LENGTH AND POINTER
4099 4F 07 FF 00 PSHS X, B * AND SAVE THEM ON THE STACK
409A 4F 07 FF 00 L9D96 7E 9C EVALUATE NUMERICAL VALUE IN COMMAND LINE
409B 4F 07 FF 00 * MULTIPLY HOR OR VER DIFFERENCE BY SCALE FACTOR.
409C 4F 07 FF 00 9970 0F 09 #HORZ =
409D 4F 07 FF 00 L9D9A LDB SCALE DRAW SCALE AND BRANCH IF ZERO - THIS WILL CAUSE A
409E 4F 07 FF 00 L9D9B #017 =
409F 4F 07 FF 00 L9D9D 2B 1F ZERO DEFAULT TO FULL SCALE
40A0 4F 07 FF 00 L9D9E 04 29 CLEAR MS BYE
40A1 4F 07 FF 00 L9D9F 04 2F EXG D, X EXCHANGE DIFFERENCE AND SCALE FACTOR
40A2 4F 07 FF 00 L9DB0 04 2F STA , D-5, S SAVE MS BYTE OF DIFFERENCE ON STACK (SIGN INFORMATION)
40A3 4F 07 FF 00 L9DB1 04 2F SBR L9DB9 BRANCH IF POSITIVE DIFFERENCE
40A4 4F 07 FF 00 L9DB2 04 2F L9DB4 NEGATE ACCO
40A5 4F 07 FF 00 L9DB6 04 2F L9DB8 MULT DIFFERENCE BY SCALE FACTOR
40A6 4F 07 FF 00 L9DB9 04 2F TFR U, D SAVE 2 MS BYTES IN ACCD
40A7 4F 07 FF 00 L9DBA 04 2F LSBA
40A8 4F 07 FF 00 L9DBB 04 2F L9DBD LSRA
40A9 4F 07 FF 00 L9DBE 04 2F L9DBF LSRB
40AA 4F 07 FF 00 L9DBF 04 2F *DIVIDE ACCO BY 4 - EACH SCALE INCREMENT IS 1/4 FULL SCALE
40AB 4F 07 FF 00 L9DBF 0E 00 TST , 0, D = padd sign of original difference and
40AC 4F 07 FF 00 L9DBF 0E 04 SBR L9C07 =return IF POSITIVE
40AD 4F 07 FF 00 L9DBF 0E 06 NEGB
40AE 4F 07 FF 00 L9DBF 0E 07 #NEGATE ACCUMULATOR D
40AF 4F 07 FF 00 L9DBF 0E 08 L9DC3 NEG8
40B0 4F 07 FF 00 L9DBF 0E 09 L9DC4 NEGB
40B1 4F 07 FF 00 L9DBF 0E 0A SBCA #000
40B2 4F 07 FF 00 L9DBF 0E 0B #SBCA IF ACCA=#
40B3 4F 07 FF 00 L9DC7 0F 01 RTS
40B4 4F 07 FF 00 L9DCB 0F 01 TFR X, D TRANSFER UNCHANGED DIFFERENCE TO ACCD
40B5 4F 07 FF 00 L9DAC 0E 01 BD
40B6 4F 07 FF 00 L9DCA 0E 01 RTS
40B7 4F 07 FF 00 L9DB8 0D 01 L9DCB 0E 03 TFR X, D CLEAR MS BYTE OF HORIZONTAL DIFFERENCE
40B8 4F 07 FF 00 L9DB9 0D 03 PULL X * GET THE DRAW ANGLE AND SCALE AND SAVE THEM ON
40B9 4F 07 FF 00 L9DBF 0E 0C #192
40BA 4F 07 FF 00 L9DBF 0E 0D #255
40BB 4F 07 FF 00 L9D9B 0E 0E LDB ZERO U = 0; DEFAULT HORIZ END POSITION = 0
40BC 4F 07 FF 00 L9DBD 0E 0F ADDHORDEF ADD DIFFERENCE TO HORIZ START
40BD 4F 07 FF 00 L9DF2 0E 0A #B = HORIZ Coord = # IF Result is Neg
40BE 4F 07 FF 00 L9D9F 0E 0B TFR D, U SAVE HORIZ END POSITION IN U
40BF 4F 07 FF 00 L9DF2 0E 0C PUT VER DIFFERENCE IN ACCD
40C0 4F 07 FF 00 L9DBE 0E 0D LDB ZERO X = 0; DEFAULT VER END POSITION = 0
40C1 4F 07 FF 00 L9DBF 0E 0E ADDDEFERENCE TO VER START
40C2 4F 07 FF 00 L9DF2 0E 0F #C
40C3 4F 07 FF 00 L9DF2 0E 10 #D = ADD DIFFERENCE TO VER START
40C4 4F 07 FF 00 L9DF3 0E 11 #F
40C5 4F 07 FF 00 L9DF2 0E 12 #D CLEAR VER END POSITION IN X
40C6 4F 07 FF 00 L9DBE 0E 13 #F
40C7 4F 07 FF 00 L9DF2 0E 14 #D save vert end position in X
4129 9E24 A6 D6 TST VD6 GET DRAW FLAG
4130 9E26 A6 D3 BNE L9E28 BRANCH IF NO DRAW
4131 9E28 BD 94 A1 JSR L94A1 DRAW A LINE FROM (HORBEG,VERBEGIN) TO (HOREND,VEREND)
4132 9E28 B6 D5 L9E28 CLR VD5 RESET UPDATE FLAG
4133 9E2D B6 D6 CLR VD6 RESET DRAW FLAG
4134 9E2F 7E 9C OD JMP L9CDD GO GET ANOTHER COMMAND
4135
4136 * SET THE DRAW POSITION
4137 9E32 BD 9B 98 L9E32 JSR L9E39 GET A CHAR FROM COMMAND LINE
4138 9E36 34 A6 PHS B A SAVE CHARACTER
4139 > 9E37 BD 9E 9E JSR L9E56 EVALUATE HORIZ DIFFERENCE
4140 9E3A 34 96 PHS B A SAVE IT ON STACK
4141 9E3C BD 9B 9B JSR L9E98 GET A CHAR FROM COMMAND LINE
4142 9E3F B1 2C CMPA #0:0:0 CHECK FOR COMM
4143 9E41 1B 26 FF #7 L9E4D LINE L9E4D IF NO COMM
4144 > 9E45 BD 9E 98 JSR L9E58 EVALUATE VERT DIFFERENCE
4145 9E4B 1F #1 TFR D,X SAVE VERT DIFFERENCE IN U
4146 9E4A 35 A8 PULLS U GET HORIZ DIFFERENCE IN U
4147 9E4C 35 A2 PULLS A GET FIRST COMMAND CHARACTER
4148 9E4E 2B CMPA #0:0:0 IF FIRST COMMAND CHAR WAS EITHER + OR -, TREAT
4149 9E5B 27 A4 BEQ L9E56 *THE VALUES IN U & X AS DIFFERENCES AND MOVE
4150 9E5B 21 0D CMPA #0:0:0 *POINTER, OTHERWISE TREAT U & X AS AN ABSOLUTE
4151 9E54 26 A6 RE 3B BNE L9D0F *POSITION AND MOVE THE CURRENT POSITION THERE.
4152 9E56 1F 3B L9E56 TFR U,D PUT HORIZ DIFFERENCE IN ACCD
4153 9E58 7E 9D CB JMP L9DCC GO MOVE THE DRAW POSITION
4154
4155 9E5B BD 9B 98 L9E5B JSR L9E98 GET A CHAR FROM COMMAND LINE
4156 9E5E 2A 98 L9E5E CMPA #0:0:0 *CHECK FOR A LEADING PLUS SIGN (RELATIVE MOTION)
4157 9E6B 27 A7 BEQ L9E69 *AND BRANCH IF RELATIVE
4158 9E6B B1 0D CMPA #0:0:0 *CHECK FOR A LEADING MINUS SIGN (RELATIVE MOTION)
4159 9E6E 27 A4 BEQ L9E6A ==AND BRANCH IF RELATIVE
4160 9E6E BD 9B E2 JSR L9E82 MOVE COMMAND STRING BACK ONE IF NOT RELATIVE MOTION
4161 9E69 4F L9E69 CLRA ACCA = U := ; ACCA => # IS :
4162 9E6A 34 A2 L9E6A PHS B A SAVE ADD/SUB FLAG
4163 9E6C BD 9C 0B JSR L9C9B EVALUATE DECIMAL NUMBER IN COMMAND STRING - RETURN VALUE IN ACCB
4164 *
4165 9E6F 35 A2 PULLS A GET ADD/SUB FLAG
4166 9E71 4D TSTA CHECK IT, #; ADD, => #; SUB
4167 9E72 27 A4 BEQ L9E78 RETURN IF ADD
4168 9E74 4F CLRA
4169 9E75 58 NEQB
4170 9E76 B2 #0 SBEA #00 *NEGATE ACCB INTO A TWO BYTE SIGNED VALUE IN ACCD
4171 9E78 39 L9E78 RTS
4172
4173 *
4174 *
4175 9E79 #0 #0 #0 #1 L9E79 FBD $0000,$0001 SUBARC 0
4176 9E7D FE C5 19 19 L9E7D FBD $0E65,$1919 SUBARC 1
4177 9E81 FB 16 31 F2 L9EB1 FBD $0FB6,$31F2 SUBARC 2
4178 9E85 45 85 4A 51 L9EB5 FBD $0458,$54A5 SUBARC 3
4179 9E89 EC B6 61 9F L9E9B FBD $0C84,$619F SUBARC 4
4180 9E8D E1 C7 7B AE L9EBD FBD $0E1C7,$78AE SUBARC 5
4181 9E8F 61 D2 3E 04 L9EBF FBD $061D2,$3E04 SUBARC 6
4182 9E8S C6 62 A2 69 L9E95 FBD $0C65,$A269 SUBARC 7
4183 9E89 BS 85 BS 85 L9E99 FBD $0B56,$B586 SUBARC 8
4184 *
4185 *
4186 * CIRCLE
4187 * THE CIRCLE IS ACTUALLY DRAWN AS A 64 SIDED
4188 *
4189 * POLYGON. IT IS COMPOSED OF 64 LINE COMMANDS
4190 9E90 B1 4B CIRCLE CMPA #0:0:0 CHECK FOR # SIGN
4191 9E9F 26 A3 BNE L9E9A SKIP IF NOT
4192 9E9A 90 9F JSR GETCHAR GET ANOTHER CHARACTER FROM BASIC
4193 9E9E 8A BD 95 22 L9EAFE GET MAX HOR & VER COORD VALUES AND PUT THEM IN VD3 AND VD5
4194 9E9B BD 9B 82 JSR L9E82 GET HOR & VER CENTER COORDS AND PUT IN HORBEGIN,VERBEGIN
4195 9E9D BD 9D 10 JSR L931D NORMALIZE START COORDS FOR PROPER PMODE
4196 9E9F AE 4C LDX #1 GET HOR COORD
4197 9EA0 9F CB STX VCB SAVE IT
4198 9EA6 AE 42 LDX $82,O GET VERT COORD
4199 9EB2 9F CD STX VCD SAVE IT
419A 9EB4 BD 02 6D JSR SYNCOMMA SYNTAX CHECK FOR COMMA
419B 9EB7 BD 07 0D JSR LB73D EVALUATE EXPRESSION RETURN VALUE IN X
419C 9EBA CE #0 CF LDU #VCF POINT U TO TEMP DATA STORAGE
419D 9EBD AF C4 STX ,U SAVE RADIUS
419E 9EBF BD 93 20 JSR L932E NORMALIZE RADIUS
419F 9EC2 BD 81 LDA #81 SET TO PSET
4200 9EC4 97 C2 STA SETFLG SAVE PSET/PRESET FLAG
4201 9EC6 BD 95 B1 JSR L9E61 GO EVALUATE COLOR EXPRESSION AND SAVE IN WCOLOR
4202 9EC9 BE #1 #0 LDX #100! HEIGHT/WIDTH RATIO DEFAULT VALUE
4203 9EEC 90 A5 JSR GETCHAR GET AN INPUT CHARACTER FROM BASIC
4204 9EE7 B6 8F BEQ L9EEF BRANCH IF NONE
4205 9EEB BD 02 6D JSR SYNCOMMA SYNTAX CHECK FOR COMMA
4206 9EEC BD 81 41 JSR LB141 EVALUATE EXPRESSION
4207 9EEF 96 4F LDA FPEXP1 *GET FPAW EXPONENT, ADD B TO IT AND RESAVE IT - THIS
4208 9EF2 8B #8 ADDA #88 *WILL EFFECTIVELY MULTIPLY FPAW BY 256.
4209 9EF3 97 4F STA FPEXP2 *
420A 9EF4 BD 47 80 JSR LB74D EVALUATE EXPRESSION, RETURN VALUE IN X
420B 9EF6 96 86 L9E6F LDA PMODE GET CURRENT PMODE VALUE
420C 9EE1 85 82 B7A #82 TEST FOR PMODE #1,4
420D 9EE3 27 A4 BEQ L9EE9 BRANCH IF 0
420E 9EE6 81 4F TFR X,0 *MULT X BY 2 - FOR PMODES 2,3 THE HOR PIXELS ARE 2X AS LONG AS
420F 9EE7 3B 8B LEAX D,X *PMODES #0,4; MULT HN RATIO BY 2 TO COMPENSATE
4210 9EE9 9F D1 L9E99 STX VD1 SAVE HM RATIO
4211 9EEB C6 #1 LDA #81
4212 9EEF D7 C2 STB SETFLG *
4213 9EEF D7 D8 STA VD8 FIRST TIME FLAG - SET TO # AFTER ARC DRAW
4214 9EF1 BD 9F 82 JSR L9FE2 EVALUATE CIRCLE START POINT (OCTANT, SUBARC)
4245 9EF4 34 #6
  PSHS B,A
  SAVE START POINT
4246 9EF6 BD 9F E2
  JSR L9FE2
  EVALUATE CIRCLE END POINT (OCTANT, SUBARC)
4247 9EF9 DD 09
  STD VD9
  SAVE END POINT
4248 9EFB 35 #6
  PULS A,B
  GET START POINT
4249 9EFD 34 #6
  L9FED
  STORE CURRENT CIRCLE POSITION
4250 9EFF 9E C3
  LDX HOREND
  * MOVE HOR, VER COORDS FROM HOREND, VEREND TO
4251 9F01 9F BD
  STX HOREND
  * HOR, VERBEG R MOVE OLD END COORDINATES
4252 9F03 9E C5
  LDX VEREND
  * NEW START COORDINATES
4253 9F05 9F BF
  STX VERBEG
4254 9F07 CE 9E 7B
  LDV ML9F79+2
  POINT TO TABLE OF SINES & COSINES
4255 9F0A B4 #1
  BEQ L9F11
  *BRANCH IF EVEN
4256 9F0E 58
  NEGB
4257 9F0F CB #8
  ADDB #98
  *CONVERT 0-7 TO 8-1 FOR OLD OCTANT NUMBERS
4258 9F11 58
  L9F11
4259 9F12 58
  ASLB
  *=FOUR BYTES/TABLE ENTRY
4260 9F13 3C 55
  LEA B,U
  POINT U TO CORRECT TABLE ENTRY
4261 9F15 34 48
  PHSB U
  SAVE SIN/COS TABLE ENTRY
4262 9F17 BD 9F A7
  JSR L9FA7
  CALCULATE HORIZ OFFSET
4263 9F19 35 48
  PULS U
  GET SIN/COS TABLE PTR
4264 9F1C 33 5E
  LEA $-#4,U
  MOVE TO COSINE (VERT)
4265 9F1E 34 1B
  PSHS X
  SAVE HORIZ OFFSET
4266 9F1F BD 9F A7
  JSR L9FA7
  CALCULATE VERT OFFSET
4267 9F23 35 2B
  PULS Y
  PUT HORIZ OFFSET IN Y
4268 9F25 A6 E4
  LDA ,Y
  *
4269 9F27 B4 #3
  ANDA #93
  *
4270 9F29 27 #6
  BEQ L9F31
  *BRANCH IF OCTANT #3,4,7
4271 9F2B 81 #3
  CMPA #93
  *
4272 9F2D 27 #2
  BEQ L9F31
  *BRANCH IF OCTANT #3,4,7
4273 9F2F 1E 12
  EXG X,Y
  SWAP HORIZ AND VERT OFFSETS
4274 9F31 9F C3
  LDV ML9F31
  STX HOREND
4275 9F33 1F 21
  VFR Y,X
  LOAD X WITH THE CALCULATED VERT OFFSET
4276 9F35 3C D1
  LDD VD1
  GET HW RATIO
4277 9F37 BD 9F 85
  JSR L9FB5
  MULT VERT OFFSET BY HW RATIO
4278 9F39 1F 2B
  TFR Y,D
  TRANSFER THE PRODUCT TO ACC
4279 9F3C 4D
  TSTA
  CHECK OVERFLOW FLAG AND GET MSB RESULT
4280 9F3D 1E 26 15 #9
  LBNL L9F4A
  FC ERROR IF RESULT > 255
4281 9F41 D7 C5
  STB VEREND
  SAVE DELTA VER MBS
4282 9F43 1F 3B
  TFR U,D
  LSB RESULT TO ACCA
4283 9F45 9F 97
  STA VEREND+1
  SAVE DELTA VER LSB
4284 9F47 A6 E4
  LDA ,Y
  *
4285 9F49 B1 #2
  CMPA #92
  * BRANCH IF OCTANT = #1,6,7 (SUBARC HOR END
4286 9F4B 25 1E
  RLCA
  POINT := (HOR CENTER)
4287 9F4D 81 #6
  CMPA #96
  * BRANCH IF OCTANT = #1,6,7 (SUBARC HOR END
4288 9F4F 24 4A
  BCC L9F5B
  = POINT := (HOR CENTER)
4289 9F51 DC CB
  LDD VCB
  GET HOR COORD OF CENTER
4290 9F53 93 C3
  SUBD HOREND
  SUBTRACT HORIZONTAL DIFFERENCE
4291 9F55 24 11
  BCC L9F68
  BRANCH IF NOT UNDERFLOW
4292 9F57 6F
  CLRA
4293 9F58 5F
  CLRB
  * IF NEW HOR < #, FORGE IT TO BE #
4294 9F59 2B #B
  BRA L9F68
  SAVE NEW COORD
4295 9F5B DC CB
  L9F5B
  LDD VCB
  GET HOR COORD OF CENTER
4296 9F5D 03 C3
  ADDD HOREND
  ADD HORIZONTAL DIFFERENCE
4297 9F5F 25 #5
  BLD L9F66
  BRANCH IF OVERFLOW
4298 9F61 1E 93 D3
  CMPD VD3
  COMPARE TO MAX HORIZ COORD
4299 9F64 25 #2
  BLD L9F68
  BRANCH IF > MAX HORIZ
429A 9F66 DC D3
  L9F66
  LDD VD3
  GET MAX HOR COORD
429B 9F68 DD 03
  L9F68
  STD HOREND
  SAVE NEW HORIZ SUBARC END COORD
429C 9F6A A6 E4
  LDA ,Y
  *
429D 9F6C B1 #4
  CMPA #94
  * BRANCH IF OCTANT = 0,1,2,3 (SUBARC VERT END
429E 9F6E 25 4A
  BLD L9F7A
  * POINT := (VERT CENTER)
429F 9F70 DC CB
  LDD VCD
  GET VER COORD OF CENTER
42A0 9F72 93 C2
  SUBD VEREND
  SUBTRACT VERTICAL DIFFERENCE
42A1 9F74 24 11
  BCC L9F87
  BRANCH IF NOT UNDERFLOW
42A2 9F76 6F
  CLRA
42A3 9F77 5F
  CLRB
  *IF NEW VERT < #, FORCE IT TO BE #
42A4 9F7B 2B #B
  BRA L9F87
  SAVE NEW COORD
42A5 9F7E 3C C0
  L9F7A
  LDD VCD
  GET VER COORD OF CENTER
42A6 9F7C D3 C5
  ADDD VEREND
  ADD VERTICAL DIFFERENCE
42A7 9F7F 25 #5
  BLD L9F95
  BRANCH IF OVERFLOW
42A8 9F80 1E 93 D5
  CMPD VD5
  COMPARE TO MAX VERT COORD
42A9 9F83 25 #2
  BLD L9F8B
  BRANCH IF > MAX VERT
42AA 9F85 DC D5
  L9F85
  LDD VD5
  GET MAX VERT COORD
42AB 9F87 DD 05
  L9F87
  STD VEREND
  SAVE NEW VERT SUBARC END COORD
42AC 9F8B 00 88
  TST VD8
  CHECK FIRST TIME FLAG
42AD 9F8B 26 #2
  BNE L9F8B
  *DO NOT DRAW A LINE FIRST TIME THRU -
42AE * *BECAUSE THE FIRST TIME YOU WOULD DRAW A LINE
42AF * *FROM THE CENTER TO THE FIRST POINT ON THE CIRCLE
42B0 9F8D BD 58
  BSR L9FD
  DRAW A LINE
42B1 9F8F 35 #6
  L9FBF
  PULS A,B
  GET END COORDS
42B2 9F91 #4 0B
  LSR VD8
  SHIFT FIRST TIME FLAG
42B3 9F93 25 #5
  BLD L9F9A
  DO NOT CHECK FOR END POINT AFTER DRAWING FIRST ARC
42B4 9F95 1E 93 D9
  CMPD VD9
  COMPARE CURRENT POSITION TO END POINT
42B5 9F97 27 8C
  BNE L9FA6
  CIRCLE DRAWING FINISHED
42B6 * INCREMENT SUBARC CTR, IF > 7 THEN INCRT OCTANT CTR
42B7 9FA1 5C
  L9FAE
  INCRT SUBARC CTR
42B8 9FA3 CB #98
  > ?7
42B9 9FA5 BD 24
  BNE L9FA3
  NO
42BA 9FAF 4C
  INCR OCTANT CTR
42BB 9FAF 5F
  CLRB
  RESET SUBARC CTR
42BC 9FA1 B4 #7
  ANDA #97
  *KEEP IN RANGE OF 0-7; ONCE ACCA = #7, THIS WILL MAKE ACCA = #,
42BD * *SO THE END POINT WILL BE (0,0) AND THE CIRCLE ROUTINE WILL END.
42BE 9FA3 7E 9E FD
  L9FA3
  JMP L9FE0
  KEEP DRAWING CIRCLE
42BF 9FA6 39
  L9F9A
  RTS
  EXIT CIRCLE ROUTINE
42C0 * *MULTIPLY RADIUS BY SIN/COS VALUE AND RETURN OFFSET IN X

845
DISASSEMBLY OF EXTENDED BASIC 1.1

**DISASSEMBLY OF EXTENDED BASIC 1.1**

- **3D32** 9F97 9E CF L9FA7 LDX VCF GET RADIUS
- **3D32** 9FA9 EC C4 LDD ,U GET SIN/COS TABLE MODIFIER
- **3D33** 9FAB 27 87 BSR L9FB4 BRANCH IF $ = 0 - OFFSET = RADIUS
- **3D34** 9FAD 83 8D #1 SUBR #1
- **3D35** 9FBD BD #3 BSR L9FB5 MUL RADIUS BY SIN/COS
- **3D36** 9FB2 1F 21 TFR Y,X RETURN RESULT IN X REG
- **3D37** L9FB4 RTS
- **3D38** * MULTIPLY (UNSIGNED) TWO 36 BIT NUMBERS TOGETHER *
- **3D39** * ENTER WITH ONE NUMBER IN ACCD, THE OTHER IN A *
- **3D3A** * REG. THE 4 BYTE PRODUCT WILL BE STORED IN 4,S-7,S *
- **3D3B** * (Y, U REG ON THE STACK). I.E. (AA AB) X (XY XL) = *
- **3D3C** * 256*AA*XY+16*(AA*YL+AB*XL). THE 2 BYTE *
- **3D3D** * MULTIPLIER AND MULTIPLICAND ARE TREATED AS A 1 *
- **3D3E** * BYTE INTEGER PART (MSB) WITH A 1 BYTE FRACTIONAL PART (LSB) *
- **3D3F** L9FB5 PSHS U,Y,X,A,A SAVE REGISTERS AND RESERVE STORAGE SPACE ON THE STACK
- **3D40** L9FB7 CLR $04,5 RESET OVERFLOW FLAG
- **3D41** L9FB9 A6 63 LDA $03,5 =
- **3D42** L9FBB 3D MUL =
- **3D43** L9FBC ED 66 STD $06,5 = CALCULATE ACCB*XL. STORE RESULT IN 6,5
- **3D44** L9FBE EC 61 LDD $01,5 *
- **3D45** L9FC8 3D MUL * CALCULATE ACCB*XH
- **3D46** L9FCF 3D ADD $06,5 =
- **3D47** L9FC3 89 #0 ADDA $0000 =
- **3D48** L9FC5 ED 65 STD $05,5 = ADD THE CARRY FROM THE 1ST MUL TO THE RESULT OF THE 2ND MUL
- **3D49** L9FC7 E6 44 LDB ,5 *
- **3D4A** L9FC9 A6 63 LDA $03,5 *
- **3D4B** L9FCB 3D MUL * CALCULATE ACCA*XH
- **3D4C** L9FCC ED 65 STD $05,5 = ADD RESULT TO TOTAL OF 2 PREVIOUS MULTS
- **3D4D** L9FDE 2A 02 BCC L9FD4 BRANCH IF NO OVERFLOW
- **3D4E** L9FDF 5C 6A INC $04,5 SET OVERFLOW FLAG (ACCD > $FFFF)
- **3D4F** L9FD4 L9FA7 LDA $05,5 *
- **3D50** L9FD6 E6 62 LDB $02,5 *
- **3D51** L9FDB 3D MUL = CALCULATE ACCA*XH
- **3D52** L9FD9 E3 64 ADD $04,5 =
- **3D53** L9FDB ED 64 STD $04,5 = ADD TO PREVIOUS RESULT
- **3D54** L9FDD 3F 5F PULS A,B,X,Y,U,PC RETURN RESULT IN U,Y
- **3D55** L9FDF 4E 2A 01 L9FD4 JMP L9FA1 GO DRAW A LINE FROM (HORBEG,VERBEG) TO (HOREND,VEREND)
- **3D56**
- **3D57** L9FEB CLRB L9FEB CLRB DEFAULT VALUE OF ZERO
- **3D58** L9FED 9F15 JSR GETCHAR GET CURRENT INPUT CHAR
- **3D59** L9FEE 27 11 BSR L9FEB BRANCH IF NONE
- **3D5A** L9FE7 BD 2D 60 JSR SYNCOMMA SYNTAX CHECK FOR COMMA
- **3D5B** L9FEA BD 81 41 JSR LB141 EVALUATE NUMERIC EXPRESSION
- **3D5C** L9FED 94 4F LDD FPBEEP GET EXPONENT OF FPAB
- **3D5D** L9FEF 88 06 ADDA #$86 ADD TO EXponent - MULTIPLY EXponent BY 64
- **3D5E** L9FF1 97 4F STA FPBEFP RESAVE EXPONENT
- **3D5F** L9FFF BD 2E 01 JSR LB7BE CONVERT FPAB TO INTEGER IN ACCB
- **3D60** L9FF6 C4 3F ANDB #$3F MAX VALUE OF 63
- **3D61** L9FF8 1F 9B L9FFB JSR B,A SAVE VALUE IN ACCA ALSO
- **3D62** L9FFA C4 87 ANDB #$87 LOW ACCB CONTAINS SUBARC NUMBER
- **3D63** L9FFC 44 LSRA *
- **3D64** L9FFD 44 LSRA *
- **3D65** L9FFE 44 LSRA *
- **3D66** L9FFF 39 RTS
| LA59A | A59A | LB4F3 | B4F3 | PAINT | 98EC | V4D | 004D |
| LA5A5 | A5A5 | LB50F | B50F | PCLEAR | 968B | VARPTR | 86BE |
| LA5AE | A5AE | LB51B | B51B | PCLS | 9532 | VALTMP | 006E |
| LA5C7 | A5C7 | LB51A | B51A | PCOPY | 9723 | VARDES | 003B |
| LA5E4 | A5E4 | LB56D | B56D | PIA0 | FF00 | VARPTR | 0039 |
| LA616 | A616 | LB643 | B643 | PIA1 | FF20 | VARTAB | 001B |
| LA619 | A619 | LB634 | B654 | PLAY | 9A22 | VCB | 00CB |
| LA635 | A635 | LB657 | B657 | PLYTMR | 00E3 | VCD | 00CD |
| LA644 | A644 | LB659 | B659 | PMOD | 9621 | VCF | 00CF |
| LA648 | A648 | LB69B | B69B | PMODE | 0086 | VD1 | 00D1 |
| LA65F | A65F | LB64A | B64A | POS | 86AC | VD3 | 00D3 |
| LA708 | A708 | LB6AD | B6AD | PPOINT | 9339 | VD4 | 00D4 |
| LA7E9 | A7E9 | LB70E | B70E | PRESET | 9365 | VD5 | 00D5 |
| LA7F4 | A7F4 | LB734 | B734 | PSET | 9361 | VD6 | 00D6 |
| LA974 | A974 | LB73B | B73B | PUT | 9758 | VD7 | 00D7 |
| LA976 | A976 | LB73D | B73D | RELFLG | 00A | VDB | 00DB |
| LA9A2 | A9A2 | LB740 | B740 | RENUM | 8A09 | VD9 | 00D9 |
| LA9BB | A9BB | LB7C2 | B7C2 | RESET | FFFE | VDA | 00DA |
| LA1E1 | AC1E | LB95B | B95B | RESSGN | 0062 | VERBEG | 00BF |
| LA33C | AC33 | LB95C | B95C | RSTVEC | 0072 | VERDEF | 00C9 |
| LA46C | AC46 | LB99F | B99F | RVEC15 | 018B | VEREND | 00C5 |
| LA690 | AC60 | LB9AC | B9AC | RVEC17 | 0191 | VOLHI | 00DF |
| LA73C | AC73 | LB9AF | B9AF | RVEC18 | 0194 | VOLLW | 00E0 |
| LA7CC | AC7C | LB984 | B984 | RVEC19 | 0197 | WCOLOR | 00B4 |
| LAC8 | AC8 | LB984 | B984 | RVEC20 | 019A | XBMST | 00C0 |
| LAC1F | AC1F | LB9C2 | B9C2 | RVEC22 | 01A0 | XIQSV | 89C4 |
| LAD51 | AD51 | LBAC | BACA | RVEC23 | 01A3 | XVEC15 | 8846 |
| LADD0 | AD01 | LB3A | BA3A | RVEC3 | 0167 | XVEC17 | 88F0 |
| LAD9A | AD9A | LB2 | B2 | RVEC4 | 016A | XVEC18 | 829C |
| LAD2 | AD2 | LC5 | BC5 | RVEC5 | 0176 | XVEC19 | 87E5 |
| LAD3 | AD3 | LC6 | BC6 | RVEC6 | 0179 | XVEC20 | 82B9 |
| LAD9 | AD9 | LC7 | BC7 | SAM | FFC0 | XVEC23 | 8304 |
| LAD9E | AD9E | LC8A | BC8A | SCALE | 00E9 | XVEC3 | 8273 |
| LAD6 | AD6 | LC9B | BC9B | SCREEN | 9670 | XVEC4 | 8CF1 |
| LADD0 | AD04 | LCA0 | BC0 | SETFLG | 00C2 | XVEC8 | 826D |
| LADEB | ADEB | LCB14 | BC14 | SQR | 8400 | XVEC9 | 8E90 |
| LAC15 | AE15 | LBC2F | BC2F | STRINOUT | 899C | ZERO | 008A |
| LAD2 | AED2 | LBC35 | BC35 | SYNCOMMA | B26D | | |
| LAE0 | AEE0 | LBC4 | BC4 | TAN | 8381 | | |
| LDF6 | AF6 | LBC5 | BC5 | TEMPO | 00E2 | | |
| LAF4 | AF4 | LBC6 | BC6 | TEMPR | 000F | | |
| LB035 | B035 | LBCA0 | BCA0 | TIMER | 8968 | | |
| LB141 | B141 | LBCB8 | BC8 | TIMOUT | 00E7 | | |
| LB143 | B143 | LB99 | B99 | TIMVAL | 0112 | | |
| LB146 | B146 | LBD5 | BDC5 | TINPTR | 002F | | |
| LB156 | B156 | LBDCC | BDC | TPMSTK | 00DC | | |
| LB158 | B158 | LBD0 | BDD | TRCFGL | 00AF | | |
| LB244 | B244 | LBE9 | BEE | TROFF | 86A8 | | |
| LB262 | B262 | LBF | BEF | TRON | 86A7 | | |
| LB267 | B267 | LBEF | BEF | TXTTAB | 0019 | | |
| LB26A | B26A | LBF7 | BF7 | USR | 013E | | |
| LB26F | B26F | LBF | BFA | USRADR | 00B0 | | |
| LB277 | B277 | LCC02 | C02 | V40 | 0040 | | |
| LB284 | B284 | LINBUF | 02DC | V41 | 0041 | | |
| LB2CE | B2CE | LINE | 93B8 | V43 | 0043 | | |
| LB357 | B357 | LOG | 8446 | V45 | 0045 | | |
| LB35C | B35C | MEM | MEM | V47 | 0047 | | |
| LB3A2 | B3A2 | NOTELN | 00E1 | V4A | 004A | | |
| LB44A | B44A | OCTAVE | 00DE | V4B | 004B | | |
EXPLANATION OF TERMS:

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

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

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

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

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

A,X 92ED  MOVE A PIXEL TO THE RIGHT (2 COLOR) - Adjust the X register and ACCA one pixel position to the right in the 2 color mode. Enter with the absolute screen address in the X register and the pixel mask in ACCA.

A,X 92F4  MOVE A PIXEL TO THE RIGHT (4 COLOR) - Adjust the X register and ACCA one pixel position to the right in the 4 color mode. Enter with the absolute screen address in the X register and the pixel mask in ACCA.

A,B,U 931D  NORMALIZE COORDINATES - Adjust the horizontal and vertical coordinates for the current PMODE. Enter with X,Y coordinates in HORBEG and VERBEG, the normalized coordinates will be returned in the same.

A,B 9377  TURN ON A PIXEL - Turn on the pixel which is being pointed to by the X register (absolute screen address) and ACCA (pixel mask) to the color in ALLCOL. Set CHGFLG <> 0 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.
X  950D  INCREMENT VERTICAL POSITION - Gets the current vertical coordinate (VERBEG) and moves it one down.

X  9515  DECREMENT HORIZONTAL POSITION - Gets the current horizontal coordinate (HORBEG) and moves it one to the left.

X  951B  DECREMENT VERTICAL POSITION - Gets the current vertical coordinate (VERBEG) and moves it one up.

A,B,X,U  9522  GET MAXIMUM COORDINATES - Get the maximum values of the horizontal and vertical coordinates for the current PMODE. Return HOR in VD3 and VER in VD5.

A,B,X  9536  CLEAR GRAPHIC SCREEN - Clear the current graphics screen to the color in ACCB. If ACCB = 0 then clear to the current background color.

A,B  9710  CALCULATE ABS(VEREND-VERBEG) - Calculate the absolute value of the distance between VEREND and VERBEG. Carry flag will indicate which was the larger coordinate.

A,B  971D  CALCULATE ABS(HOREND-HORBEG) - Calculate the absolute value of the distance between HOREND and HORBEG. Carry flag will indicate which was the larger coordinate.

U,Y  9FB5  16 BIT MULTIPLY - Multiply (unsigned) two 16 bit numbers together. Enter with one number in ACCD and the other in the X register. The four byte product will be returned in the Y and U registers.
<table>
<thead>
<tr>
<th>START</th>
<th>END</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>8000</td>
<td>8001</td>
<td>EXTENDED BASIC ROM IDENTIFIER</td>
</tr>
<tr>
<td>80DE</td>
<td>80E7</td>
<td>COMMAND INTERPRETATION TABLE ROM IMAGE</td>
</tr>
<tr>
<td>80E8</td>
<td>813B</td>
<td>COPYRIGHT MESSAGES</td>
</tr>
<tr>
<td>8183</td>
<td>81E9</td>
<td>PRIMARY RESERVED WORD TABLE</td>
</tr>
<tr>
<td>81F0</td>
<td>821D</td>
<td>PRIMARY RESERVED WORD DISPATCH TABLE</td>
</tr>
<tr>
<td>821E</td>
<td>8256</td>
<td>SECONDARY RESERVED WORD TABLE</td>
</tr>
<tr>
<td>8257</td>
<td>8272</td>
<td>SECONDARY RESERVED WORD DISPATCH TABLE</td>
</tr>
<tr>
<td>83AB</td>
<td>83AF</td>
<td>FLOATING POINT VALUE FOR PI/2</td>
</tr>
<tr>
<td>83E0</td>
<td>841C</td>
<td>TAYLOR SERIES COEFFICIENTS FOR ARCTANGENT</td>
</tr>
<tr>
<td>841D</td>
<td>8431</td>
<td>TAYLOR SERIES COEFFICIENTS FOR NATURAL LOG(X)</td>
</tr>
<tr>
<td>8432</td>
<td>8436</td>
<td>FLOATING POINT VALUE FOR .5*SQR(2)</td>
</tr>
<tr>
<td>8437</td>
<td>843B</td>
<td>FLOATING POINT VALUE FOR THE SQUARE ROOT OF 2</td>
</tr>
<tr>
<td>843C</td>
<td>8440</td>
<td>FLOATING POINT VALUE FOR -.5</td>
</tr>
<tr>
<td>8441</td>
<td>8445</td>
<td>FLOATING POINT VALUE FOR THE NATURAL LOG OF 2</td>
</tr>
<tr>
<td>84C4</td>
<td>84C8</td>
<td>FLOATING POINT VALUE FOR CORRECTION FACTOR OF EXPONENTIAL FUNCTION</td>
</tr>
<tr>
<td>84C9</td>
<td>84F1</td>
<td>TAYLOR SERIES FOR E^X</td>
</tr>
<tr>
<td>890B</td>
<td>890E</td>
<td>ERROR MESSAGES</td>
</tr>
<tr>
<td>8BD9</td>
<td>8BDC</td>
<td>UL' (UNKNOWN LINE NUMBER) MESSAGE</td>
</tr>
<tr>
<td>929C</td>
<td>92A5</td>
<td>JUMP TABLE FOR CALPOS ROUTINES</td>
</tr>
<tr>
<td>92DD</td>
<td>92E4</td>
<td>2 COLOR MODE PIXEL MASKS</td>
</tr>
<tr>
<td>92E5</td>
<td>92E8</td>
<td>4 COLOR MODE PIXEL MASKS</td>
</tr>
<tr>
<td>948A</td>
<td>9493</td>
<td>JUMP TABLE OF ADDRESSES WHICH WILL MOVE POINTERS ONE PIXEL TO THE RIGHT</td>
</tr>
<tr>
<td>9706</td>
<td>970F</td>
<td>TABLE OF HOW MANY BYTES PER HORIZONTAL GRAPHIC</td>
</tr>
<tr>
<td>Row</td>
<td>Value1</td>
<td>Value2</td>
</tr>
<tr>
<td>-----</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>9839</td>
<td>9851</td>
<td>LOOKUP TABLE FOR PSET, PRESET, AND, OR, AND NOT MODIFIERS FOR THE PUT COMMAND</td>
</tr>
<tr>
<td>9C5B</td>
<td>9C61</td>
<td>NUMERICAL NOTE VALUES FOR LETTER NOTES</td>
</tr>
<tr>
<td>9C62</td>
<td>9C79</td>
<td>TIMING DELAYS FOR OCTAVE 1</td>
</tr>
<tr>
<td>9C7A</td>
<td>9C91</td>
<td>TIMING DELAYS FOR OCTAVE 2</td>
</tr>
<tr>
<td>9C92</td>
<td>9CB5</td>
<td>TIMING DELAYS FOR OCTAVES 3, 4 AND 5</td>
</tr>
<tr>
<td>9E79</td>
<td>9E9C</td>
<td>TABLE OF SINES AND COSINES FOR CIRCLE</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>NAME</th>
<th>NBR</th>
<th>LABEL</th>
<th>ADDR</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>NF</td>
<td>00</td>
<td>LB108</td>
<td>B108</td>
<td>NEXT WITHOUT FOR</td>
</tr>
<tr>
<td>SN</td>
<td>01</td>
<td>LB277</td>
<td>B277</td>
<td>SYNTAX ERROR</td>
</tr>
<tr>
<td>RG</td>
<td>02</td>
<td>LAECF</td>
<td>AECF</td>
<td>RETURN WITHOUT GOSUB</td>
</tr>
<tr>
<td>OD</td>
<td>03</td>
<td>LB0C3</td>
<td>B0C3</td>
<td>COND OUT OF DATA</td>
</tr>
<tr>
<td>FC</td>
<td>04</td>
<td>LB44A</td>
<td>B44A</td>
<td>ILLEGAL FUNCTION CALL</td>
</tr>
<tr>
<td>OV</td>
<td>05</td>
<td>LBA92</td>
<td>BA92</td>
<td>OVERFLOW</td>
</tr>
<tr>
<td>OM</td>
<td>06</td>
<td>LAC44</td>
<td>AC44</td>
<td>OUT OF MEMORY</td>
</tr>
<tr>
<td>UL</td>
<td>07</td>
<td>LAED2</td>
<td>AED2</td>
<td>UNDEFINED LINE NUMBER</td>
</tr>
<tr>
<td>BS</td>
<td>08</td>
<td>LB447</td>
<td>B447</td>
<td>BAD SUBSCRIPT</td>
</tr>
<tr>
<td>DD</td>
<td>09</td>
<td>LB43B</td>
<td>B43B</td>
<td>COND REDIMENSIONED ARRAY</td>
</tr>
<tr>
<td>/Ø</td>
<td>10</td>
<td>LB06</td>
<td>BC06</td>
<td>DIVISION BY ZERO</td>
</tr>
<tr>
<td>ID</td>
<td>11</td>
<td>AFF5</td>
<td>COND</td>
<td>ILLEGAL DIRECT STATEMENT</td>
</tr>
<tr>
<td>TM</td>
<td>12</td>
<td>LB151</td>
<td>B151</td>
<td>TYPE MISMATCH</td>
</tr>
<tr>
<td>OS</td>
<td>13</td>
<td>LB585</td>
<td>B585</td>
<td>COND OUT OF STRING SPACE</td>
</tr>
<tr>
<td>LS</td>
<td>14</td>
<td>LB625</td>
<td>B625</td>
<td>STRING TOO LONG</td>
</tr>
<tr>
<td>ST</td>
<td>15</td>
<td>LB553</td>
<td>B553</td>
<td>STRING FORMULA TOO COMPLEX</td>
</tr>
<tr>
<td>CN</td>
<td>16</td>
<td>AE32</td>
<td>AE32</td>
<td>COND CAN'T CONTINUE</td>
</tr>
<tr>
<td>FD</td>
<td>17</td>
<td>AF06</td>
<td>COND</td>
<td>BAD FILE DATA</td>
</tr>
<tr>
<td>AO</td>
<td>18</td>
<td>A61C</td>
<td>A61C</td>
<td>FILE ALREADY OPEN</td>
</tr>
<tr>
<td>DN</td>
<td>19</td>
<td>A61F</td>
<td>A61F</td>
<td>DEVICE NUMBER ERROR</td>
</tr>
<tr>
<td>IO</td>
<td>20</td>
<td>A619</td>
<td>A619</td>
<td>INPUT/OUTPUT ERROR</td>
</tr>
<tr>
<td>FM</td>
<td>21</td>
<td>A616</td>
<td>A616</td>
<td>BAD FILE MODE</td>
</tr>
<tr>
<td>NO</td>
<td>22</td>
<td>A3FB</td>
<td>A3FB</td>
<td>FILE NOT OPEN</td>
</tr>
<tr>
<td>IE</td>
<td>23</td>
<td>B03F</td>
<td>B03F</td>
<td>COND INPUT PAST END OF FILE</td>
</tr>
<tr>
<td>DS</td>
<td>24</td>
<td>AC94</td>
<td>AC94</td>
<td>COND DIRECT STATEMENT IN FILE</td>
</tr>
<tr>
<td>UF</td>
<td>25</td>
<td>B8BF</td>
<td>B8BF</td>
<td>UNDEFINED FUNCTION CALL</td>
</tr>
<tr>
<td>NE</td>
<td>26</td>
<td>BCDD</td>
<td>BCDD</td>
<td>FILE NOT FOUND</td>
</tr>
</tbody>
</table>

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

**DIFFERENCES BETWEEN EXTENDED BASIC 1.0 AND 1.1**

**ADDRESS**

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>B6FF22</td>
<td>LDA PIA1+2</td>
<td>Read PIA PORT B</td>
</tr>
<tr>
<td>02</td>
<td>BITA #2</td>
<td>Check MEM SIZE JUMPER</td>
</tr>
<tr>
<td>03</td>
<td>BNE L00DA</td>
<td>Branch if high</td>
</tr>
<tr>
<td>03</td>
<td>STA SAM+29</td>
<td>Set SAM CNTL REG MEM SIZE to 64K</td>
</tr>
<tr>
<td>84</td>
<td>L80DA JMP</td>
<td>Jump to address in X REG</td>
</tr>
<tr>
<td>00</td>
<td>FCB $00,$00</td>
<td>Dead space</td>
</tr>
</tbody>
</table>

*This code is not used by any of the Basics*

---

**Figure G1 - Version 1.0 Code**

**ADDRESS**

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6F</td>
<td>CLR ,-S</td>
<td>Save default token (non DLOADM) on stack</td>
</tr>
<tr>
<td>4D</td>
<td>CMPA #&quot;M&quot;</td>
<td>Is it DLOADM?</td>
</tr>
<tr>
<td>04</td>
<td>BNE LBC25</td>
<td>No</td>
</tr>
<tr>
<td>E4</td>
<td>STA ,S</td>
<td>Save the M on the stack</td>
</tr>
</tbody>
</table>

---

**Figure G2 - Version 1.0 Code**

**ADDRESS**

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>61</td>
<td>SAVE DEFAULT TOKEN (NON DLOADM) ON STACK</td>
<td></td>
</tr>
<tr>
<td>4D</td>
<td>IS IT DLOADM?</td>
<td></td>
</tr>
<tr>
<td>04</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>E4</td>
<td>SAVE THE M ON THE STACK</td>
<td></td>
</tr>
</tbody>
</table>

---

**Change instruction op code from BNE(1.0) to BEQ(1.1).**

This change was necessitated by the previous fix to DLOAD.

**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 absolute 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 L80D0) to the routine which will adjust the BASIC input pointer.

```
96A3 1025 1DA3  LBL0  LB44A  IF TRYING TO CLEAR LESS THAN END
96A7  93  19   SUBD  TXTTAB  OF CURRENT PAGE = 'ILLEGAL FUNCTION CALL'
96A9  D3  1B   ADDD  VARTAB  SUBTRACT START OF RAM
96AB  1F  01   TFR  D,X    ADD END OF BASIC PROGRAM
96AD  C3  00C8  ADDD  #200  X=TOP OF PCLEARED SPACE+SIZE OF BASIC
96B0  93  21   SUBD  FRETOP  PROGRAM
96B2  24  B9   BCC  L966D  ADD 200 - LEAVE SOME ROOM FOR STACK
96B2  24  B9   BCC  L966D  SUBTRACT OUT START OF CLEARED SPACE
96B2  24  B9   BCC  L966D  NO ROOM LEFT - 'ILLEGAL FUNCTION CALL'
```

Figure G3 - Version 1.0 code
# ASCII Chart

## Display Character Set

<table>
<thead>
<tr>
<th>HEX VALUE</th>
<th>CHARACTER</th>
<th>HEX VALUE</th>
<th>CHARACTER</th>
<th>HEX VALUE</th>
<th>CHARACTER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Non-</td>
<td>Inverted</td>
<td>Non-</td>
<td>Inverted</td>
<td>Non-</td>
</tr>
<tr>
<td>00</td>
<td>40</td>
<td>@</td>
<td>1B</td>
<td>5B</td>
<td>X</td>
</tr>
<tr>
<td>01</td>
<td>41</td>
<td>A</td>
<td>19</td>
<td>59</td>
<td>Y</td>
</tr>
<tr>
<td>02</td>
<td>42</td>
<td>B</td>
<td>1A</td>
<td>5A</td>
<td>Z</td>
</tr>
<tr>
<td>03</td>
<td>43</td>
<td>C</td>
<td>1B</td>
<td>5B</td>
<td>[</td>
</tr>
<tr>
<td>04</td>
<td>44</td>
<td>D</td>
<td>1C</td>
<td>5C</td>
<td>\</td>
</tr>
<tr>
<td>05</td>
<td>45</td>
<td>E</td>
<td>1D</td>
<td>5D</td>
<td>]</td>
</tr>
<tr>
<td>06</td>
<td>46</td>
<td>F</td>
<td>1E</td>
<td>5E</td>
<td>↑</td>
</tr>
<tr>
<td>07</td>
<td>47</td>
<td>G</td>
<td>1F</td>
<td>5F</td>
<td>←</td>
</tr>
<tr>
<td>08</td>
<td>48</td>
<td>H</td>
<td>20</td>
<td>50</td>
<td>38</td>
</tr>
<tr>
<td>09</td>
<td>49</td>
<td>I</td>
<td>21</td>
<td>61</td>
<td>39</td>
</tr>
<tr>
<td>0A</td>
<td>4A</td>
<td>J</td>
<td>22</td>
<td>62</td>
<td>&quot;</td>
</tr>
<tr>
<td>0B</td>
<td>4B</td>
<td>K</td>
<td>23</td>
<td>63</td>
<td>#</td>
</tr>
<tr>
<td>0C</td>
<td>4C</td>
<td>L</td>
<td>24</td>
<td>64</td>
<td>$</td>
</tr>
<tr>
<td>0D</td>
<td>4D</td>
<td>M</td>
<td>25</td>
<td>65</td>
<td>%</td>
</tr>
<tr>
<td>0E</td>
<td>4E</td>
<td>N</td>
<td>26</td>
<td>66</td>
<td>&amp;</td>
</tr>
<tr>
<td>0F</td>
<td>4F</td>
<td>O</td>
<td>27</td>
<td>67</td>
<td>'</td>
</tr>
</tbody>
</table>

### Notes
- The chart includes non-inverted and inverted characters for each hex value.
- Non-inverted characters are on the left, and inverted characters are on the right.
- The table includes the ASCII values and their corresponding characters.