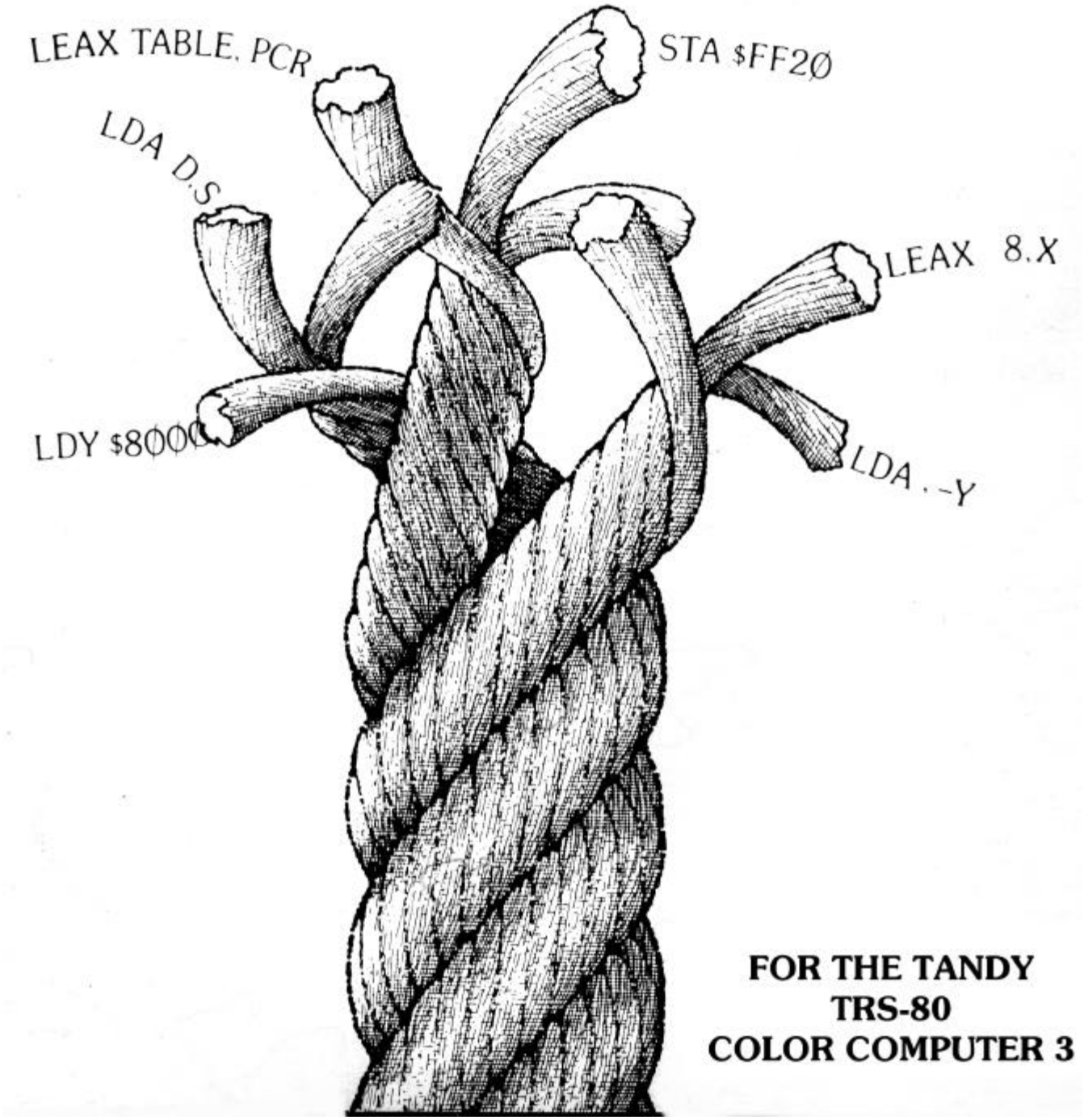
SUPER EXTENDED BASIC UNRAVELLED II



FOR THE TANDY
TRS-80
COLOR COMPUTER 3

TABLE OF CONTENTS

APPENDICES

FOREWORD

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

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

1. The comments have been cleaned up some. In cases where a comments continued onto the next line, a * is placed in the Labels column, as well as a * at the beginning of each line of the comment. In cases where the previous comment used this format, a = was used. This was done in the original, but not all comments stuck to this format.
2. I have renumbered all the line numbers. Each Appendix (with code) starts at Line 0001.
3. Some spell checking, and context checking was done to verify accuracy.
4. I used the Letter Gothic MT Bold Font. This allows for display of Slashed Zeros. I thought it important to be able to distinguish between 0 and O.
5. All the Hex code now shows the Opcodes.

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

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

About Me

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Walter K. Zydhek

INTRODUCTION

Super Extended Basic is the definitive source of information on the super high-resolution graphics commands and Basic enhancements available from the Color Computer 3. Super Extended Basic Unravelled will deal with the enhancements to Color Computer Basic that make Basic versions 2.Ø and higher. These Basic versions were introduced in the Color Computer 3. Super Extended Basic follows in the fine tradition of the Basic Unravelled series. We are proud to say that these books are the best documentation available concerning the internal structure of Color Computer Basic. We believe that Color and Extended Basic Unravelled were used as a guideline during the creation of Super Extended Basic 2.Ø!

Super Extended Basic Unravelled will provide the reader with a complete detailed and fully commented source listing of the super high-resolution 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 routines. The reader will need to have a basic knowledge of 68Ø9 assembly language programming to be able to take full advantage of the opportunities, which this book presents. It is also assumed that the reader is familiar with the contents of the Color Computer 3 Extended Basic 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 Super Extended Basic.

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 Super Extended Basic is not a stand-alone system. There are many direct calls into Basic and Extended Basic. These calls are not explained in this book and it will be necessary for the reader to refer to the other 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 and a symbol table is given in Appendix C.

HISTORY OF THE COLOR COMPUTERS

The original Color Computer was introduced in August of 198Ø with a standard 4K of memory. Enclosed in a battleship gray case, it sold for about $4ØØ. The Color Computer had a unique combination of random access memory (RAM) and read only memory (ROM). There were two levels of Basic available: Color Basic and Extended Color Basic. Disk Extended Color Basic was soon added to the group. Each of these three levels of Basic were stored in their own ROM. The Basic ROM started at $AØØØ, the Extended Basic ROM at $8ØØØ, and the Disk Basic ROM was plugged into the expansion port (ROM PAK slot) and started at $CØØØ. Adding Extended Basic to your system was as simple as inserting a ROM into the circuit board. The user added Disk Basic by installing the Disk Controller into the expansion slot. This system of adding ROMs to upgrade the system caused some problems during the design of Basic. Several routines in Color Basic had to be changed to work in Extended Basic and Disk Basic. However, since they are in ROM. they couldn t be changed. The problem was solved by the use of RAM hooks. Within Basic and Extended Basic, several routines jump to these vectors located in lower RAM (at $15E). From here, control

can be redirected to another routine. For example, Basic has a routine that checks for a valid device number (Ø=screen, -1=cassette, and -2=printer). With just Basic installed, any other value returns an error. With Disk Basic installed, however, the routine has to also allow numbers 1-15. The Basic routine executes a JSR to the vector in low RAM. With just Basic installed, control is returned immediately. With Disk Basic installed, control is re-routed into the Disk Basic ROM. to a routine that allows values 1-15.

The Color Computer 2 was introduced in 1983 sporting a small white case and a new keyboard. The changes were more cosmetic than anything else. At this point, several home computers were competing with the Color Computer, and prices were falling fast. By combining several chips into one and replacing a few components. the Color Computer 2 primarily allowed Tandy to produce the computer less expensively.

In early 1984, Tandy considered producing a new version of the Color Computer - entitled the Deluxe - but the project was later canceled due to costs and the planned super edition of the Color Computer: the Color Computer 3 was finally introduced in August, 1986, six years after the original Color Computer, offering 512K and advanced graphics.

Many legacies of the Color Computer 1 and 2 remain in the Basic ROM of the CoCo 3. Throughout Basic, Extended Basic, and Disk Basic you will find many sections of code that were written to deal with the three ROM system. Much of it is unnecessary, since Basic on the CoCo is now in RAM, but was retained to insure compatibility with previous Color Computer versions.

In October 1983, Spectral Associates introduced a 3-volume set of books: the Basic Unravelled series. Those who have the 3-book set will find that Super Extended Basic Unravelled will be a welcome addition to the 3-book set. Those who don t already have these books should consider purchasing EXTENDED COLOR BASIC UNRAVELLED and DISK BASIC UNRAVELLED.

HOW TO USE THIS BOOK

Most users will undoubtedly spend the majority of their time using Appendix B, which contains a source code listing of the top half of the Color Basic 2.Ø ROM. This source code was developed independently by the author who has never seen or had access in any way to any source code developed by Microsoft, Tandy or Microware.

Most labels used in Appendix B correspond to absolute addresses in ROM/RAM preceded by an 'L'. Literal labels have been assigned to RAM variables (memory locations that contain data which may change) and some routines and data tables. The symbol table in Appendix C will allow the user to locate the address of the literal label. The symbol table is composed of a long list of entries, arranged in alphabetical order. Each entry contains an address, a type code and the actual symbol (label) itself. The typecode maybe D, E or L. If it is a D, the symbol is a variable name and it will be found in Appendix A. If the code is an E, the symbol has been defined by an EQUATE pseudo-op. Almost all of the equates may be found at the start of the variable listing in Appendix A. If the code is an L, the symbol is

a label and will be found in Appendix B or in Extended Color Basic Unravelled s Appendix B.

Super Extended Basic Unravelled only covers the top half of the CoCo 3 ROM. Extended Color Basic Unravelled covers the bottom half of the ROM. There are several calls from the Super Extended portion of the ROM into the bottom half that you will not be able to follow unless you have the Extended Color Basic book. Many people have the Unravelled series, which was produced for the CoCo 2. The Extended Color Basic Unravelled book is essentially a merged version of the older Color Basic Unravelled (version 1.2) and Extended Basic Unravelled (version 1.1). If you have both of these books, Appendix I provides a listing of all of the changes made to convert Color Basic 1.2 and Extended Basic 1.1 into the bottom half of the Extended Color Basic 2.Ø ROM. The Disk Basic ROM (1.Ø or 1.1) has not been modified at all.

The CoCo 3 ROM (version 2.Ø) from addresses $CØØØ - $DFFF contains the code used to initialize the system and the cute digitized picture of the authors which you get when you hold down the ALT and CTRL keys on power up or reset. The code located in this area must be of a temporary nature because the Disk Basic code is loaded into this area when the contents of the ROMs are transferred to RAM. A substantial portion of this code is used to patch Color Extended (and Disk if there) Basic once it has been loaded into RAM. The patches make use of labels of the following types: PATCHxx, ALINKxx and BLINKxx. The PATCHxx addresses correspond to the actual address where the patch will be made. The ALINKxx addresses correspond to those addresses where the patches will transfer control. The BLINKxx addresses correspond to where the patch code will re-enter the mainstream code after the patch code has been executed. Not all patches will have a BLINKxx type address since control may be returned by an RTS.

The FCS pseudo-op code is used in this listing. For those readers who are unfamiliar with this pseudo-op, it means exactly the same as an FCC pseudo-op with the exception that the last character in the literal string has a bias of $8Ø added to it. If, for example, the last character of an FCS instruction was an E, it would be assembled to $Ø5 ($45+$8Ø).

COLOR COMPUTER 3 HARDWARE DIFFERENCES

This chapter deals with the major hardware and software differences between the original Color Computer and the Color Computer 3. The designers of the Color Computer 3 were guided by several design criteria, which occasionally forced some odd decisions. First, the CoCo 3 had to be as compatible as was possible the older CoCos so that as much of the old CoCo software as possible would function on the CoCo 3. Also the CoCo 3 had to be as inexpensive as possible so that it would have a market niche (other than just selling it to CoCo 2 owners). This constraint led to the GIME chip (or custom or tequila chip as it was also known).

Memory

The most apparent difference with the Color Computer 3 is the capability of having up to 512K of Random Access Memory (RAM). This RAM is made up of 256K Dynamic RAM chips. The 128K version of the computer has four (64K x 4 bit) chips, whereas the 512K version has sixteen (256K x 1 bit) chips. Upgrading from 128K to 512K primarily consists of removing the existing RAMs and inserting the 512K upgrade board into the provided sockets.

It should be noted that several problems were encountered early on with the RAMs supplied on early Color Computer 3s (the computer would crash during manipulation of the screen in some of the horizontal virtual enable modes). These models contained 150ns RAMs from Mitsubishi. These problems appeared to be solved by replacing the RAMs with 150ns RAMs from Nippon Electronics Corp (NEC) or by replacing them with 120ns parts.

The GIME Chip

The Color Computer 3 has many features not available on the original Color Computer, including memory management, advanced interrupt processing, and advanced graphics. All of these functions, in addition to the original Color Computer graphics modes, are handled by one large chip referred to as the GIME chip (pronounced gimmee", for Graphics/Interrupt/Memory Enhancement). We will touch on these subjects in the following paragraphs, and go into detail on them a little later.

The Central Processing Unit (CPU) in the Color Computer 3 is the 6809. This processor, by its very design, is limited to accessing 64K of memory at one time. Making this chip work in a 512K computer is, therefore, a neat trick. To do this, a system called memory management is employed. Memory within the computer is divided up into 8K blocks (producing 16 blocks in a 128K system, 64 blocks in 512K). From this pool of 8K blocks you may select any 8 to fill the CPU s memory space of 64K.

Two additional interrupts have been added to the Color Computer 3. The first is a timer interrupt which is a 12-bit interval timer, allowing you to set it to any value from 0-4095. This timer is counted down, and when it goes below 0, an interrupt may be triggered. The count is decremented every 70nsec or 63.5usec (selectable). The other interrupt is a keyboard interrupt, causing an interrupt to occur whenever a key or joystick button is pressed.

Super High Resolution Graphics

No fewer than 15 super high resolution graphics modes have been added to the Color Computer 3, four of which are accessible from Basic. These range from 128 pixels across with 2 colors to 640 pixels across with 4 colors in addition, each graphics mode can have any one of 4 depths (192, 200, 210, and 225 rows). This allows up to 60 different possibilities (actually there are more...which we ll discuss a little later). Basic is limited to 192 vertical rows.

In addition to the new graphics modes, the Color Computer 3 has 64 colors available, with a maximum of 16 on the screen at a time (actually, if you do some fancy stuff with interrupts, you can get all 64 at a time, but that s beyond the scope of this book). To allow up 64 different colors, palette registers were incorporated into the GIME chip. Palette registers are discussed in detail in Chapter Five.

The original Color Computer allowed you to start the screen display on any 512-byte boundary. This has been improved in the Color Computer 3 to allow the screen to be set on any eight-byte boundary. This allows a true smooth vertical scroll. In addition, there is a technique that allows smooth horizontal scrolling. Along with the new graphic capabilities, there are also new text modes available. Text can be displayed with 32, 40, 64, or 80 characters-per horizontal row.

In the Color Computer 3, you have control over the color of the border, which you did not on the original Color Computer.

Sound

In order to keep the cost of the Color Computer 3 down, no sound chips were installed into the computer. Sound is still generated using the CPU or the optional Sound/Speech Cartridge.

$FF22

In the old CoCos the graphics display was taken care of by the Video Display Generator (VDG). Controls were passed from the CPU to the VDG by way of Peripheral Interface Adapter 1 (PIA1). The graphics display of the CoCo 3 is handled entirely by the GIME chip, which has eliminated the need to pass controls through PIA1. However, in order to maintain compatibility with the older CoCos, a register has been built into the GIME chip which will retain any information written to the old VDG control bits of $FF22. This internal GIME chip register is not accessible by the user and any data returned by reading $FF22 will come from PIA1, not from the GIME chip. The PIA1 bits, which provided control to the VDG in the older CoCos, are not used in the CoCo 3. Bit 2 of $FF22 (RAMSZ) is also not used - there is no hardware flag in the CoCo 3 to tell the user if the system contains 128K or 512K.

The existence of the GIME chip s internal $FF22 register has allowed the addition of some extra features to the CoCo 3's CoCo compatible mode (32 column). Bits 4 (upper/lower case) and 5 (invert) can be used to invert the foreground and background colors of the text screen or to allow true lower case characters.

If bit 4 = 0, the ASCII codes from 0-31 will be the inverse video representations of the codes from 64-95. If bit 4 = 1, the ASCII codes from 0-31 will contain lower case characters. Appendix I contains a complete chart of theses codes. If bit 5 = 0, the text screen will be black characters on a green background. If bit 5 = 1, the text screen will be green characters on a black background.

Peculiarities and Compromises

During the design of the Color Computer 3, Tandy was particularly careful to insure, as much as possible, that all software written for the original Color Computers would work on the Color Computer 3. This involved some peculiarities and compromises.

Tandy s primary method of insuring this compatibility was to have a mode of operation similar to the original Color Computer. This mode is referred to as the CoCo compatible Mode, and is active when bit 7 of $FF9Ø is set. In this mode the primary difference is that the SAM registers (used to set graphics modes and screen addresses in the original Color Computer) are enabled. When this bit is cleared, you are in the CoCo 3 mode and the video display and vertical offset modes of the SAM registers are disabled.

The original Color Computer was limited to 64K, and Basic was designed to operate within that constraint. Since making Basic work with more than 64K would have required major changes in Basic (which would mean software would be incompatible), Basic still is limited to 64K (32K for operating system code, 32K for workspace). It should be noted that a few commands do access memory outside of this 64K range (LPEEK, LPOKE, HGET, HSCREEN, etc.), but the Basic program is limited to this 64K block. Several enhancements were made to Basic, though, including super high-resolution graphics (up to 64Ø x 192) and a 4Ø or 8Ø column text mode. Fortunately, both the super hi-res graphics screen and the 4Ø/8Ø-column text screen are located outside of the 32K workspace (unlike the original Color Computers, where memory for these were taken out of the workspace). Additionally, a super hi-res HGET/HPUT buffer is located outside of the workspace. This means that high-resolution graphics and text can be achieved without sacrificing workspace. Of course while Basic can t have more than 32K for its program, machine language programs have full use of the 128K or 512K that you have in your system.

Most of the original Color Computer graphics modes have been implemented in the Color Computer 3. However, the Semigraphics 4 mode (the standard 32 column text screen) is the only semigraphics mode available. Any software using the other semigraphics modes will not display properly on the Color Computer 3.

Other Differences

One of the important aspects of memory management is insuring that the code the CPU must execute is always in place. For example, you can t tell the memory management unit (MMU) to move in a new memory section when the CPU is getting its instructions from the section you re replacing. This is critically important with interrupts. When an interrupt occurs, which could happen anytime, control of the CPU must be transferred to a safe area of memory. The area from $FEØØ-$FEFF is especially good for this purpose since it is a special area of the logical address space. Bit 3 of INITØ may be used to exempt this area from the effects of the MMU registers, thus guaranteeing that the RAM in this area is constant regardless of the changing contents of the MMU registers. Programs written for the original Color Computer that try to use the top of this area of RAM will most likely not work on the CoCo 3 because the CoCo 3 routes its interrupt vectors through there.

There are four new keys on the CoCo 3 s keyboard. The OS-9 operating system uses the Control (CTRL) and Alternate (ALT) keys. Basic doesn t use or recognize any of the new keys (except on power-on, as described below).

One of the popular graphic modes on the original Color Computer is the artifacting mode. This mode, accessed from Basic by the command PMODE 4:SCREEN 1,1, allows 128 x 192 graphics with red, blue, black, and white colors. Depending upon how the computer fired up, the red and blue colors may be switched, so most programs ask you to press the reset button to change the colors. On the Color Computer 3, these colors will fire up in a uniform way, and pressing reset alone won t change anything. If, when you press reset or turn on the computer, you hold the F1 key down, the colors will be reversed. This method allows full compatibility with previous Color Computer software.

The original Color Computer had the capability of working at double the clock speed (referred to as double speed). This didn t work in all machines, and was never supported by Tandy. The Color Computer 3 is guaranteed to work in double speed which can be turned on by storing data at $FFD9, and turned off by storing data at $FFD8. Note that the Sound/Speech Cartridge (SSC) does not work in the double speed mode. At the time of this writing, there is no hardware fix to allow the SSC to work in double speed, but it is expected that several fixes will be available soon. The fix would undoubtedly not be supported by Tandy.

The Color Computer 3 also supports two button joysticks or mice. Super Extended Basic and OS-9 Level Two will allow you to read the second joystick button.

The last major addition to the Color Computer 3 is the inclusion of composite and RGB output. This was primarily done to allow reasonable display of the super high-resolution graphics. The computer may be connected to any standard composite monitor or any analog RGB monitor (this is different than TTL RGB or RGBI).

MEMORY MANAGEMENT

The 68Ø9 microprocessor can only address 64K of memory. In order to address more than 64K, a method must be found which will allow the user to switch different blocks of memory into the CPU s address space. The ability to perform this function is generally referred to as memory management'. There are as many different ways to implement a memory management scheme as there are different computers in the world, and each method will have its own strong points and limitations. In the Color Computer 3, the GIME chip performs the Memory Management Unit (MMU) function.

The GIME chip will allow 512K of RAM to be accessed by the CoCo 3. This 512K-address range is called the physical address space. The physical address space is broken down into 64 blocks of 8K each. The six high order bits of any address ($Ø-$7FFFF) are the block number. In a 128K machine that means that there will be 16 blocks which will actually have RAM in them, and the other 48 blocks will be treated as three sets of 16 blocks all three of which are mirrors of the high order 16 blocks. A 512K machine will, of course, have 64 blocks of RAM. The GIME chip determines this configuration and there is no known way at this time to trick, fool, or otherwise cajole the chip into allowing you to hang more RAM on the system without adding hardware to the computer. From this pool of 64 8K blocks you may select any eight to fill the CPU s memory space of 64K. The 64K range, which comprises the address range of the CPU, is referred to as the logical address space. In order to simplify the task of understanding how this is done, it is best for the reader to discard the concept of the fixed memory map of the computer s memory. From the point of view of the CPU, the Color Computer 3 s RAM is not one large contiguous block from Ø - $7FFFF. This will undoubtedly cause a certain amount of confusion because the video display section of the Color Computer 3 does consider the RAM as one large contiguous block.

Now, you may ask, if the memory is to be considered as 64 blocks of 8K, how does the CPU know where its memory is. That job is performed by the MMU registers which are located at $FFAØ. The eight blocks, which you select as the CPU s memory, are mapped into the CPU s address space by the MMU registers as shown in Figure 1.

MMU Register	CPU Address Space	Logical block number
$FFA7	$EØØØ-$FDFF	7
$FFA6	$CØØØ-$DFFF	6
$FFA5	$AØØØ-$BFFF	5
$FFA4	$8ØØØ-$9FFF	4
$FFA3	$6ØØØ-$7FFF	3
$FFA2	$4ØØØ-$5FFF	2
$FFA1	$2ØØØ-$3FFF	1
$FFAØ	$ØØØØ-$1FFF	Ø

Figure 1 - Memory Management Unit Registers

It is important to thoroughly understand the concept of memory blocks. The physical address space is composed of 64 physical blocks (they will be referred to simply as blocks). The logical address space is the range of $Ø-$FFFF which can be addressed by the CPU. The logical address space should be considered as composed of eight 8K blocks of RAM. The MMU registers determine which eight of the 64 blocks from the physical address space will compose the logical address space. As a natural extension, the logical address space may be thought of as being composed of eight logical blocks. The logical blocks are numbered from Ø-7 as described in

Figure 1 above. The logical blocks are not really actual memory (the physical blocks are actual memory), they are an 8K address space in the address range of the CPU and their position relative to one another may not change in the eyes of the CPU.

The MMU registers have no effect whatsoever on the manner in which the GIME chip displays graphic or text information. For the purpose of graphics, the 512K is considered as one large contiguous super chunk of RAM. In order to make this easier to understand since we are in a "block" frame of mind, just consider the video display memory as 64 contiguous 8K blocks. In other words, the video display memory is just the physical address space and there is no way to move the blocks relative to one another.

The process of setting up the CPUs memory space requires that you select eight blocks, which will comprise the logical address space. Then you must program the MMU registers with the block numbers selected. For example, if you wanted block 56 ($38) to occupy the CPU addresses Ø - $1FFF (logical block Ø), you must store the value $38 into address $FFAØ. If you wanted the high 64K of RAM of either a 128K or 512K machine to occupy the logical address space as one contiguous 64K segment, you just load the values $38 - $3F consecutively into the consecutive addresses $FFAØ - $FFA7. This is how Basic sets up the CPU s memory space.

It is important to realize that there is no prohibition against using the same block in more than one block of the logical address space. If you put the same block number in all of the MMU registers, then the same 8K block of RAM would be mapped into all eight of the logical blocks.

As an example of the power and flexibility which this system of memory management offers, we will consider the logical address space arrangement used by Basic to manipulate super hi-res graphic screens. It is not possible to read data from or write data into the Color Computer 3 s memory unless the memory is in the logical address space. For example, if you wanted to read address $4F859, you would not be able to unless block 39 had been mapped into a logical block by an MMU register. Or, put another way, the value 39 must be in one of the MMU registers ($FFAØ - $FFA7). Basic allocates 32K of memory for its super hi-res graphics screen. In order to manipulate the screen, the 32K screen must be in the logical address space. The bottom 32K of memory in a 128K system ($6ØØØØ - $67FFF) is used for the super hi-res screen by Basic. In order to access the screen, this memory is mapped into logical block 1 as shown in Figure 2. Block numbers 48-51 are the super hi-res graphics screen. Block 56 must remain in logical block Ø because it contains all of Basic s system variables and interrupt vectors, and block 63 must remain in logical block 7 because it contains the Basic program code, which manipulates the super hi-res graphics screen. Block 53 is moved into logical block 6 (overlaying Disk Basic) and is used as the HPUT/HGET buffer.

MMU Register	Block Number	Logical Block Number	Physical Address	
$FFA7	63	7	$7EØØØ-$7FFFF	Program
$FFA6	53	6	$6AØØØ-$6BFFF	HGET Buffer
$FFA5	61	5	$7AØØØ-$7BFFF	Program
$FFA4	51	4	$66ØØØ-$67FFF	Screen
$FFA3	5Ø	3	$64ØØØ-$65FFF	Screen
$FFA2	49	2	$62ØØØ-$63FFF	Screen
$FFA1	48	1	$6ØØØØ-$61FFF	Screen
$FFAØ	56	Ø	$7ØØØØ-$71FFF	System DP

Figure 2 - Super Hi-Res Graphics Memory Configuration

There is one final aspect of the Color Computer 3 s memory management system, which must be addressed. The Color Computer 3 has two sets of MMU registers. The first set of eight registers located at $FFAØ should be very familiar to you by now. The second set of eight registers is located at $FFA8 and their function is identical to that of the first set in every aspect. Bit Ø of initialization register 1 ($FF91) is used to determine which one of the sets of registers is determining the makeup of the logical address space. If bit Ø if $FF91 is set to zero, then the eight MMU registers at $FFAØ (task register Ø) control the makeup of the logical address space. If bit Ø of $FF91 is set, then the eight MMU registers at $FFA8 (task register 1) control the makeup of the logical address space (see Figure 3). The theory behind the two sets of registers is that each set of registers may be allowed to control a different task by allocating two independent segments of 64K to each task and then simply selecting the desired set of registers in order to enable the desired task. This will work fine but you must be careful to remember that switching between the task registers will do nothing to preserve the status of the CPU registers, nor will it protect you from disasters if you should be interrupted during the transition. Whenever new memory is switched into a logical address space, be sure it isn t where the program counter, stack, or interrupt service routine is located. Major problems may happen it is:

If the MMU registers have the data below in them		Then the following blocks compose the logical address space		
			$FF91 bit0=0	$FF91 bit0=1
$FFAØ 24	$FFA8 34	$ØØØØ	24	34
$FFA1 26	$FFA9 56	$2ØØØ	26	56
$FFA2 15	$FFAA 43	$4ØØØ	15	43
$FFA3 56	$FFAB 34	$6ØØØ	56	34
$FFA4 41	$FFAC 35	$8ØØØ	41	35
$FFAS 42	$FFAD Ø8	$AØØØ	42	Ø8
$FFA6 62	$FFAE 36	$CØØØ	62	36
$FFA7 61	$FFAF ØØ	$EØØØ	61	ØØ

Figure 3 - MMU task registers

Special notes:

1) All of the MMU registers may be read from as well as written to. However, only the lower 6 bits of data are accurate. The top two bits should be masked off after they are read. Also, in order to enable the MMU registers, bit 6 of $FF9Ø must be set.

2) The CoCo enable bit (bit 7, $FF9Ø) does not have any effect upon the operation of the MMU registers. The MMU enable bit (bit 6, $FF9Ø) must be set in order for the MMU registers to be operable.

3) The area from $FFØØ - $FFFF is used for system input/output and is never affected by the MMU registers. The area from $FEØØ - $FEFF is a special page (256 bytes) of RAM and may be affected by the MMU registers if MC3 (bit 3, $FF9Ø) is clear.

SUPER HIGH RESOLUTION GRAPHICS

The CoCo 3 will support several, new-high resolution graphics and alphanumeric text modes in addition to most of the older low-resolution graphics and alphanumeric modes of the CoCo 2. The only CoCo 2 alphanumeric mode supported by the CoCo 3 is the semi-graphics 4 mode.

The characteristics of the graphics modes are controlled by the graphics control registers ($FF98-$FF9F). These registers are write-only registers (attempting to read these registers will not return accurate data). The graphics control registers can have their function modified by the CoCo compatible bit (bit 7, $FF90) and the BP bit (bit 7, $FF98). It is important to realize that certain graphics control registers will be valid only if the COCO and BP bits are set up in a certain way. You may be able to produce interesting effects if you violate these restrictions, but you will have no guarantee that the effect will be supported by future versions of the Color Computer (if there are to be any future versions).

The GIME chip treats the system RAM as one contiguous 512K block for the purposes of video display. In a 128K system the true RAM is at the top of the physical address space and there are three 128K images below it. The graphics control registers are used to define the size of the screen and place it anywhere within the 512K that you wish. If you wish to modify the contents of a high resolution graphics or text screen, you must use the MMU registers to place that portion of the screen into the logical address space of the CPU in order to change the data - remember that the MMU registers will NOT affect the manner in which the screen is DISPLAYED but you must use them in order to change the data.

One last warning: be careful how you use the COCO and BP bits. You may get some interesting effects if you set both of these bits, but it may bite you in the end. We cannot say what the results will be if you use a mode which is not specifically defined. All of the video control registers are designed to be used when the COCO bit is cleared with the notable exception of the vertical offset registers. A condensed summary of the control registers is contained in Appendix D.

The registers from FF90 - FF97 are general-purpose control registers for the GIME chip

FF90 Initialization register 0 INIT0

Bit 7	COCO	1=CoCo compatible mode
Bit 6	MMUEN	1=MMU enabled
Bit 5	IEN	1 = GIME chip IRQ enabled
Bit 4	FEN	1 = GIME chip FIRQ enabled
Bit 3	MC3	1 = RAM at FEXX is constant
Bit 2	MC2	1 = standard SCS (Spare Chip Select)
Bit 1	MC1	ROM map control
Bit Ø	MCØ	ROM map control

COCO: This bit is used to toggle the CoCo compatible mode on and off. The term CoCo compatible mode is somewhat of a misnomer as there are some CoCo 2 graphics modes, which are not supported by the CoCo 3, and some of the video control registers are active even when the COCO bit is in the CoCo compatible mode. The programmer is best advised to use this bit for exactly what it was intended for - to be set when you are using CoCo 2 graphics modes and to be clear when you are using the new CoCo 3 graphics modes. The descriptions of the CoCo 3 registers given

below will explicitly state those instances in which the programmer should use the new registers with the COCO bit set.

MMUEN: When this bit is set the MMU registers are enabled. If this bit is clear, the MMU registers are inoperable and the 64K, which makes up the logical address space is, the contiguous segment from $70000 - $7FFFF.

IEN: When this bit is set, the GIME chip's IRQ Interrupt structure is enabled. If the bit is clear, the old CoCo 2 PIA IRQ interrupt structure is used.

FEN: When this bit is set, the GIME chip's FIRQ Interrupt structure is enabled. If the bit is clear, the old CoCo 2 PIA FIRQ interrupt structure is used.

MC3: When this bit is set, the RAM which occupies the CPU's address range of $FE00-$FEFF will always be taken from $7FE00-$.7FEFF. If this bit is clear and the MMUEN bit is set the RAM in the CPU's address range of $FE00-$FEFF will be taken from the block as specified by the MMU register controlling logical block 7.

MC2: Spare Chip Select (SCS) control; if 0, then the SCS line (to the expansion slot) will only be active in the $FF50-$FF5F range. If this bit is 1, then the SCS line will be active in the $FF40-$FF5F range.

MC1: ROM map control

MC0: ROM map control

MC1	MC0	ROM configuration
0	X	16K internal, 16K external
1	0	32K internal
1	1	32K external (except interrupt vectors)

FF91 Initialization register 1 INIT1

Bit 7	Unused	
Bit 6	Unused	
Bit 5	TINS	Timer input select; 1 = 70 nsec, 0 = 63.5 usec
Bit 4	Unused	
Bit 3	Unused	
Bit 2	Unused	
Bit 1	Unused	
Bit 0	TR	Task register select

TINS: This bit controls the clock input to the 12-bit interval timer. If the bit is set, the input source will be 14.31818 MHz which will produce a clock pulse approximately every 70 nanoseconds. If the bit is clear, the input source will be the horizontal blanking pulse which will produce a clock pulse approximately every 63.5 microseconds.

TR: If this bit is set, then $FFA8-$FFAF will be the active MMU registers, if the bit is clear, then $FFA0-$FFA7 will be the active MMU registers.

FF92 Interrupt request enable register IRQENR

Bit 7	Unused

Bit 6	Unused	
Bit 5	TMR	Timer interrupt
Bit 4	HBORD	Horizontal border interrupt
Bit 3	VBORD	Vertical border interrupt
Bit 2	EI2	Serial data interrupt
Bit 1	EI1	Keyboard interrupt
Bit Ø	EIØ	Cartridge interrupt

TMR: A timer interrupt is generated whenever the 12-bit interval timer ($FF94-$FF95) counts down to zero.

HBORD: The horizontal border interrupt is generated on the falling edge of the horizontal sync pulse.

VBORD: The vertical border interrupt is generated on the falling edge of the vertical sync pulse.

EI2: The serial data interrupt is generated on the falling edge of a signal on pin 4 of the serial I/O connector (JK 3).

EI1: The keyboard interrupt will be triggered whenever a zero appears on any one of the PA0-PA6 pins of PIA0. These pins are normally programmed as inputs and are used to read the keyboard. The programmer should be warned that it is not chiseled into tablets of granite that these pins remain inputs - some interesting effects may be had by programming one as an output and using it to generate an interrupt. In their normal condition as inputs, an interrupt will be generated if a key is pressed and the proper keyboard column is strobed by placing a zero in the correct column strobe register ($FF00) bit OR if a joystick fire button is pressed. It is Important to note that a keyboard interrupt cannot be generated if there is not at least one zero in the keyboard column strobe register (ignoring joystick fire buttons). Also note that there is no way to mask off the joystick fire buttons - they will always generate a keyboard interrupt.

EI0: A cartridge interrupt will be generated on the falling edge of a Signal found on pin 8 (CART) of the expansion connector.

FF93 Fast interrupt request enable register FIRQENR

Bit 7	Unused	
Bit 6	Unused	
Bit 5	TMR	Timer interrupt
Bit 4	HBORD	Horizontal border interrupt
Bit 3	VBORD	Vertical border interrupt
Bit 2	EI2	Serial border interrupt
Bit 1	EI1	Keyboard interrupt
Bit Ø	EIØ	Cartridge interrupt

The bits of FIRQENR are defined identically to those of IRQENR.

FF94 Timer register MSB

Bits 4-7	Unused
Bits Ø-3	High order four bits of the timer

See the description of the timer register low order bits ($FF95).

FF95 Timer register LSB

Bits Ø-7 Low order eight bits of the timer

The 12-bit interval timer located at $FF94-$FF95 may be set to any value from Ø to 4Ø95. When a value is loaded into the timer MS byte, the count will be automatically started. The timer will count down (it cannot count up) until it gets to zero at which time the initial count will be reloaded and the count down will restart. If the timer registers are loaded with Ø, the count down process will be inhibited. The clock input to the timer may be either 14.31818 MHz or 15.734 KHz as selected by bit 5 of INIT1.

FF96 Reserved

FF97 Reserved

The registers from $FF98 - $FF9F are the video control registers and are used to control the new video modes of the GIME chip.

FF98 Video Mode Register

Bit 7	BP	0 = Text modes, 1 = Graphics modes
Bit 6		Unused
Bit 5	BPI	Burst Phase Invert (Color Set)
Bit 4	MOCH	1 = Monochrome on Composite
Bit 3	H50	1 = 50 Hz power, 0 = 60 Hz power
Bits 0-2	LPR	Lines per row

BP: (Bit Plane): Determines whether the computer is to display graphics or text. If this bit is set to Ø, the screen is displayed as text. If it is 1, graphics are displayed.

BPI: Setting this bit will put you in the alternate color set. Technically, this bit tells the computer to invert the color burst phase going to the TV or composite monitor. Setting this bit will reverse the red and blue colors in the artifacting mode.

MOCH: When this bit is set to 1, the composite (including TV) output of the Color Computer 3 is changed to black and white (monochrome). This allows easier reading and better resolution in higher resolution text and graphics modes. This bit will not affect the RGB display.

H5Ø: If this bit is set, the power source is 5Ø Hertz, if the bit is clear; the power source is 6Ø Hz.

LPR: (Lines Per character Row): These bits determine the number of vertical lines used for each character in the text display. The one, two and three lines per row settings have little practical value, as the character itself is seven rows high. Changing the setting will not change the size of the character; it will only change the number of rows between characters. These settings only affect the way text is displayed on the screen; it has no effect on the amount of memory used to contain the screen data.

Bit pattern	Lines per character row
xxxxxØØØ	One line

xxxxx00 1	Two lines
xxxxx01 0	Three lines
xxxxx011	Eight lines
xxxxx10 0	Nine lines
xxxxx10 1	Ten lines
xxxxx11 0	Twelve lines
xxxxx111	Reserved

FF99 Video Resolution Register

The Video Resolution Register controls the resolution and colors displayed on the computer.

Bit 7		Undefined
Bits 5-6	LPF	Lines per Field (Number of Rows)
Bits 2-4	HRES	Horizontal Resolution
Bits 0-1	CRES	Color Resolution

LPF: These two bits determine the number of vertical rows on the high-resolution graphics display.

Bit Pattern	Rows Displayed
x00xxxxx	192
x01xxxxx	200
x10xxxxx	210
x11xxxxx	225

HRES: These three bits (HR0-HR2) determine the horizontal resolution. The HRES bits set the display to a specific number of bytes (not pixels) across the screen.

	BP=1	BP=0
Bit Pattern	Bytes/Row (Graphics)	Text Resolution
xxx111xx	160	80 Characters/Row
xxx110xx	128	64 Characters/Row
xxx101xx	80	80 Characters/Row
xxx100xx	64	64 Characters/Row
xxx011xx	40	40 Characters/Row
xxx010xx	32	32 Characters/Row
xxx001xx	20	40 Characters/Row
xxx000xx	16	32 Characters/Row

CRES: If BP=1, these two bits (CR0-CR1) determine the number of colors available and the number of pixels contained in each byte. Multiplying pixels/byte by the bytes hi each row will give you the number of pixels in each row.

If BP=0, then bit 1 has no effect and bit 0 is the attribute enable flag. If attributes are not enabled, the number of characters appearing on the hi-res text screen is determined by the number of characters per row set by the HRES bits, the number of rows displayed as set by the LPF bits and the number of lines per row as set by the LPR bits of the video mode register. If the attributes are enabled, the

number of bytes required to display a hi-res text screen is doubled. Each character byte is followed by an attribute byte as defined in Figure 4. Therefore, if attributes are enabled, all even bytes are character bytes, the make-up of which is determined by the GIME chip s internal character generator, and all odd bytes are attribute bytes. If the blink bit is set, the characters will blink at a rate which is determined by the interval timer ($FF94, 5). If the timer is set to zero the characters will not blink. The foreground colors are controlled by palette register numbers 8-15 and the background colors are controlled by palette register numbers Ø-7. Attributes are not available if COCO=1.

	BP=1		BP=Ø
Bit Pattern	Colors Available	Pixels/ Byte	Attributes
xxxxxx11	Undefined	Undefined	enabled
xxxxxxl0	16	2	disabled
xxxxxxØ1	4	4	enabled
xxxxxxØØ	2	8	disabled

Bit 7	BLINK	1=Character blinks
Bit 6	UNDLN	1=Character is underlined
Bit 5	FGND2	Foreground Color (MSB)
Bit 4	FGND1	Foreground Color
Bit 3	FGNDØ	Foreground Color (LSB)
Bit 2	BGND2	Background Color (MSB)
Bit 1	BGND1	Background Color
Bit Ø	BGNDØ	Background Color (LSB)

Figure 4 - Attribute byte

Summarized in Figure 5 are all of the allowed high-resolution graphics modes allowed on the CoCo 3. You will notice that not all possible combinations of the CRES and HRES bits are given below. Only those combinations listed below are guaranteed and any other combinations, although they may appear cute and useful ARE NOT GUARANTEED TO BE SUPPORTED IN FUTURE VERSIONS OF THE COCO.

HR2	HR1	HRØ	CR1	CRØ	Graphics mode
1	1	1	Ø	1	64Ø pixels, 4 colors
1	Ø	1	Ø	Ø	64Ø pixels, 2 colors
1	1	Ø	Ø	1	512 pixels, 4 colors
1	Ø	Ø	Ø	Ø	512 pixels, 2 colors
1	1	1	1	Ø	32Ø pixels, 16 colors
1	Ø	1	Ø	1	32Ø pixels, 4 colors
Ø	1	1	Ø	Ø	32Ø pixels, 2 colors
1	1	Ø	1	Ø	256 pixels, 16 colors
1	Ø	Ø	Ø	1	256 pixels, 4 colors
Ø	1	Ø	Ø	Ø	256 pixels, 2 colors
1	Ø	1	1	Ø	16Ø pixels, 16 colors
Ø	1	1	Ø	1	16Ø pixels, 4 colors
Ø	Ø	1	Ø	Ø	16Ø pixels, 2 colors
1	Ø	Ø	1	Ø	128 pixels, 16 colors
Ø	1	Ø	Ø	1	128 pixels, 4 colors
Ø	Ø	Ø	Ø	Ø	128 pixels, 2 colors

Figure 5 - High-resolution graphics modes

* The 320-pixel, 2-color mode is not guaranteed to work at all possible starting addresses of the high-resolution screen.

FF9A Border Register

Bits 6,7		Unused
Bits Ø-5	BRDR	Border color

This register controls the color of the border around the text or graphics screen. To set the border color, simply store the appropriate color code (composite or RGB) in the register. The colors available for use as a border color may be found in Appendix D.

FF9B Unused

FF9C Vertical Scroll Register

Bits 4-7		Reserved
Bits Ø-3	VSC	Vertical Scroll bits

The Vertical Scroll Register is used to allow smooth vertical scrolling while in the hi-res text modes, and is used in conjunction with the LPR bits of the video mode register. By storing consecutively larger numbers in the VSC bits, the screen will scroll up one graphics row at a time. This will continue until you reach the lines per character row value that was set by the LPR bits. Once you reach this value, to continue scrolling you should reset the vertical scroll register and then use the vertical offset registers to move the display down one entire character row.

FF9D, FF9E Vertical Offset Registers

The Vertical Offset Registers combine to determine the address (Y15-YØ) in memory where the video display starts when in the non CoCo compatible mode. The video display is treated as one large contiguous block, starting at $ØØØØØ and extending to $7FFFF (if the system has only 128K, the RAM is located from $6ØØØØ to $7FFFF and is mirrored into lower RAM in three 128K sections). The screen can be set to start on any 8-byte boundary. The video display address is set by taking the desired address, dividing it by 8, and storing that value in the vertical offset registers.

Y15	Y14	Y13	Y12	Y11	Y10	Y9	Y8

Vertical offset 1 ($FF9D)

Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0

Vertical offset 0 ($FF9E)

Figure 6 Vertical offset registers (non-CoCo compatible mode)

Setting the screen display address while in the CoCo compatible mode is different than the non-CoCo compatible mode. The address is set using a combination

of the vertical offset registers and the Synchronous Address Multiplexer s (SAM) display offset register, located from $FFC6 to $FFD3 (see Figure 8). The vertical offset registers are used to position the video display within the 512K-address space as shown in Figure 7. The three high order bits of $FF9D (YHØ-YH2) determine in which 64K segment of the physical address space the start of the video display will be found. By setting the SAM display offset register; you can specify a 512-byte offset that will be added to the segment boundary as defined by YHØ - YH2. The bottom five bits of $FF9E (YLØ-YL5) allow you to further refine the start of the video display to any eight byte boundary. The video starting address may be determined by the following formula: start = YH*64K+SAM*512+YL*8 where SAM represents the value of the SAM display offset register.

YH2	YH1	YH0	X	X	X	X	X

Vertical offset 1 ($FF9D)

X	X	VL5	VL4	VL3	VL2	VL1	VL0

Vertical offset 0 ($FF9E)

Figure 7 Vertical offset registers (CoCo compatible mode)

FF9F Horizontal Offset Register

Bit 7	HVEN	Horizontal Virtual Enable
Bits Ø-6	XØ-X6	Horizontal Offset Address

The Horizontal Offset Register allows you to add a horizontal offset to the video display. The value in the bottom 7 bits of this register is multiplied by two and added to the beginning screen address set in the vertical offset registers. For example, setting this register to 3 will make the screen appear to shift left 6 bytes.

One of the more interesting features incorporated into the Color Computer 3 is the horizontal virtual enable mode, which is turned on by setting bit 7 of the horizontal offset register to 1. When you are in this mode, the screen width is forced to be 256 bytes across. The value stored in the video resolution register determines how many bytes of this 256 byte wide screen will be displayed.

This may sound confusing, but let s try an example. First the graphics mode is set up by storing a $16 in the video resolution register (this sets the screen display to 8Ø bytes across). Next we store a $CØØØ (the screen address) into the vertical offset registers. Lastly, we store a $8Ø into the horizontal offset register, turning on the horizontal virtual enable feature. The screen now displays 8Ø bytes of a 256 byte wide screen. The display starts at $6ØØØØ. Now, simply by storing an $81 into the horizontal offset register, the screen scrolls left 2 bytes. We are now looking at a screen displaying 8Ø bytes of a 256-byte wide screen. The display starts at $6ØØØ2.

By using the horizontal virtual enable along with the vertical offset registers, you can effectively have a window displaying memory on a 256-byte wide screen, extending vertically as high as memory will allow. In addition, when the seam (where the ends of the rows meet the start of the rows) is displayed, the display is adjusted to make the two ends of the rows join together. This is truly one of the more exciting features in the Color Computer 3.

The horizontal offset addresses are intended to be used while the horizontal virtual enable mode is on, and peculiar things happen when horizontal offsets are used while not in the horizontal virtual enable mode. Let s say, for example, you set up a screen that is 16Ø bytes wide (e.g. HSCREEN 2). Since the horizontal virtual enable is off, the video display circuitry recognizes that each row

consists of 160 bytes, and each row starts 160 bytes after the start of the previous row (makes sense...). However, the horizontal offset circuitry does not recognize the graphics mode, and tries to force a 256 byte wide screen. This is not a problem when the horizontal offset register is set to 0. However, conflicts occur when other values are stored in the horizontal offset register and the horizontal virtual enable mode is off.

The results of this conflict are: 1) The horizontal offset circuitry displays the row as if it were 256 bytes wide. The first 160 bytes of the row are taken from the current row. The next 96 bytes (160 + 96 = 256) of the row are obtained from the beginning of the next row and 2) The video display circuitry starts each row 160 bytes past the start of the previous row. These two factors together appear to mirror the first 96 bytes of the screen into the 96 bytes following the screen as it is scrolled horizontally.

This effect is moderately interesting, but has few practical uses. Since it is probably the result of a compromise in the computer design, this effect will probably not be supported in future versions of the Color Computer. The horizontal offset is most efficiently used only if the horizontal virtual enable bit is set.

Synchronous Address Multiplexer (SAM)

The Synchronous Address Multiplexer is a special purpose chip used in the older CoCos to control the addressing of the various chips in the CoCo such as the RAMs, ROMs and the PIAs. This function has been incorporated into the GIME chip and the SAM control registers have been retained in the addressing arrangement of the GIME chip in order to provide compatibility with the older CoCos. The SAM registers are located from $FFC0-$FFDF and each pair of addresses in this range represents one bit of a SAM control register. The bits are cleared by writing any data to the even numbered bit, and set by writing any data to the odd numbered bit. Only those registers listed in Figure 8 are active in the CoCo 3.

In the old CoCos, the CPU speed was-controlled by two bits and true double speed could not be obtained without losing the video display. The CoCo 3 allows true, non-address dependent double speed so the CPU rate only requires one bit - either single or double speed.

The map type bit controls the ROM select lines of the GIME chip. If it is clear, the ROM select lines are allowed to be active and the MC1 and MC0 bits of INIT0 specify the configuration of the ROM. If it is set, none of the ROM select lines are allowed to be active and the system ROM is disabled.

While in the CoCo compatible mode the SAM chip selects the 512-byte boundary (within the 64K segment specified by the YH2-YH0 bits) where the screen display will start. This is done by setting or clearing the appropriate display offset register bits. For example, to set the SAM to an offset of $400 you would set bit F1 of the SAM display offset and clear the other SAM display offset bits. This is done by writing any data to addresses $FFC6, $FFC9, $FFCA, $FFCC, $FFCE, $FFD0, and $FFD2.

The display mode control and the display offset registers have no effect in the non-CoCo compatible mode.

Address		SAM Register Bit
$FFDE,F	TY	map type: 0=ROM,1=RAM
$FFD8,9	R1	CPU rate: 0=normal, 1=double speed
$FFD2,3	F6	display offset register (MSB)

$FFDØ,1	F5	display offset register
$FFCE,F	F4	display offset register
$FFCC,D	F3	display offset register
$FFCA,B	F2	display offset register
$FFC8,9	Fl	display offset register
$FFC6,7	FØ	display offset register (LSB)
$FFC4,5	V2	display mode control register (MSB)
$FFC2,3	V1	display mode control register
$FFCØ,1	VØ	display mode control register (LSB)

Figure 8 - CoCo 3 SAM registers

COLORS AND PALETTES

There are 64 color codes available on the Color Computer 3, numbered from Ø-63. By storing these values in the correct palette register (which well discuss in a bit), these colors are displayed on the screen. There are two color sets used on the Color Computer 3, one used for televisions and composite monitors, and the other used for RGB monitors. These color sets are derived in different ways, and we will discuss each of those separately.

Colors on an RGB Monitor

The term RGB is derived from the three-color signals sent to the monitor: one each for red, green, and blue. These colors correspond to the three primary colors that make up each pixel on the screen. The Color Computer produces a signal for each of these three colors, which may be any one of four strengths, numbered from Ø-3. When any one signal is Ø, the corresponding dot is off; when the signal is 3, it is on at full strength. A value of 1 or 2 would be one of the intermediate strengths. By combining these colors and intensities, a wide range of colors can be generated. The computer determines the strength of the red, green, and blue signals by the number of the color selected.

Each RGB color value uses two bits to determine the strength of each of the red, green, and blue signals. This means that a total of six bits are used to determine the value of a color (six bits, of course, allows 64 possibilities). Figure 9 shows how each color is derived. Note that the bottom three bits of the color value are used as the low order bits for each signal. The upper three bits are used as the high order bits for each signal.

Bits 6,7	Unused
Bit 5	(R1)High Order Red
Bit 4	(G1)High Order Green
Bit 3	(B1)High Order Blue
Bit 2	(RØ)Low Order Red
Bit 1	(GØ)Low Order Green
Bit Ø	(BØ)Low Order Blue

Figure 9 - RGB Color Makeup

For example, let s make the color purple which is made with the following color strengths: Blue = 3, Green = 1, Red = 2. Using the table above, this translates to the following bit pattern: xx1Ø1Ø11, or to a decimal value of 43. Refer to appendix D for a complete color chart.

Colors on a Composite Monitor or Television

Colors on a composite monitor are generated in the same way as colors on a TV. They are, however, derived in a completely different way than RGB colors. The colors are, again, specified using 6-bit. The bottom 4 bits determine the base color, and the top 2 bits determine the intensity of the base color. Figure 1Ø shows the base colors.

Bit Pattern	Base Color
ØØØØ	Black/White

0001	Blue
0010	Green
0011	Cyan
0100	Red
0101	Magenta
0110	Brown
0111	Blue-Green
1000	Sky-Blue
1001	Peacock
1010	Cyan-Green
1011	Red-Magenta
1100	Red-Orange
1101	Orange
1110	Yellow Green
1111	Blue-Purple

Figure 10 - Composite Base Colors

Composite intensity values range from 0-3, and occupy bits 4 and 5 of the color value. For example, $04 sets the color dark red, $14 is red, $24 is medium red, and $34 is bright red. See appendix D for a complete list of available colors.

Palettes

Colors in the original Color Computers were determined by storing a specific pattern of bits (pixel) within the screen memory. This pixel corresponded to a specific color. In the Color Computer 3, the pixel now corresponds to a palette , or color register (see Figure 12). When it is time to display the screen, the computer determines the palette number of a pixel, and then looks inside the palette register to get the color to display. The palette registers are located from $FFB0 - $FFBF and are read/write registers, but the top two bits must be masked off after a read operation since only six bits contain valid data.

This is a dramatic change and offers a flexibility that didn t exist before. First of all, the number of available colors as no longer limited to the resolution of the screen. However, even more exciting is what happens when you change palette registers. When a new value is stored in a palette register, say palette 1, all pixels that correspond to palette 1 change colors. This allows you to change the colors on large areas of the screen by simply changing one byte (or executing one PALETTE command). The possibilities with this method of changing colors are immense, including limited animation.

Even though there are 16 palette registers, not all of the palette registers may be active. For all 16 registers to be active, you must be in a 16-color hi-res graphics mode. It you are in a hi-res four-color mode, only the first four palette registers are active and if you are in a hi-res two-color mode, then only the first two palette registers are active. Figure 11 shows the configuration of the pixels in the byte.

Graphic byte	16-color mode	4-color mode	2-color mode
Bit 7	PA3, pixel 1	PA1, pixel 1	PA0, pixel 1
Bit 6	PA2, pixel 1	PA0, pixel 1	PA0, pixel 2
Bit 5	PA1, pixel 1	PA1, pixel 2	PA0, pixel 3
Bit 4	PA0, pixel 1	PA0, pixel 2	PA0, pixel 4
Bit 3	PA3, pixel 2	PA1, pixel 3	PA0, pixel 5

Bit 2	PA2, pixel 2	PAØ, pixel 3	PAØ, pixel 6
Bit 1	PA1, pixel 2	PA1, pixel 4	PAØ, pixel 7
Bit Ø	PAØ, pixel 2	PAØ, pixel 4	PAØ, pixel 8

Figure 11 Pixel/palette register configuration

Palette number	Pixel bit pattern	Palette register address
Ø	ØØØØ	$FFBØ
1	ØØØ1	$FFB1
2	ØO1Ø	$FFB2
3	ØØ11	$FFB3
4	Ø1ØØ	$FFB4
5	Ø1Ø1	$FFB5
6	Ø11Ø	$FFB6
7	Ø111	$FFB7
8	1ØØØ	$FFB8
9	1ØØ1	$FFB9
1Ø	1Ø1Ø	$FFBA
11	1Ø11	$FFBB
12	11ØØ	$FFBC
13	11Ø1	$FFBD
14	111Ø	$FFBE
15	1111	$FFBF

Figure 12 Pixel pattern/palette register relationship

The palette registers are not affected by the COCO bit (bit 7, $FF9Ø). Figure 13 shows the palette registers that are used in the different low and high-resolution graphics and text modes.

Graphics/Text Mode	Palette Registers Used	Palette Addresses
32 x 16 Lo-res text		
Background	13	$FFBD
Foreground	12	$FFBC
32/4Ø/64/8Ø Column Hi-res text		
Background	Ø-7	$FFBØ-$FFB7
Foreground	8-15	$FFB8-$FFBF
Lo-res graphics		
RG2, CSS=Ø	8,9	$FFB8-$FFB9
RG2, CSS=1	1Ø,11	$FFBA-$FFBB
CG3, CSS=Ø	Ø-3	$FFBØ-$FFB3
CG3, CSS=1	4-7	$FFB4-$FFB7
RG3, CSS=Ø	8,9	$FFB8-$FFB9
RG3, CSS=1	1Ø,11	$FFBA-$FFBB
CG6, CSS=Ø	Ø-3	$FFBØ-$FFB3
CG6, CSS=1	4-7	$FFB4-$FFB7
RG6, CSS=Ø	8,9	$FFB8-$FFB9

RG6, CSS=1	10,11	$FFBA-$FFBB
Hi-res graphics		
16 COLOR	0-15	$FFB0-$FFBF
4 COLOR	0-3	$FFB0-$FFB3
2 COLOR	0-1	$FFB0-$FFB1

Figure 13 - Palettes used in graphics modes

INTERRUPTS

A new system of interrupts has been added with the advent of the Color Computer 3. This section will discuss the two new interrupt sources (keyboard and timer), enabling the interrupts, and processing individual interrupts. There will be no discussion of the CoCo 2 PIA based interrupts.

The new interrupt features are enabled by setting bits 4 (FIRQ) and 5 (IRQ) of $FF90. If these bits are clear, interrupts are handled as they were in the original Color Computer. Setting these bits allows you to use the new interrupt system. The new system of interrupts is based entirely upon the GIME chip and makes no use whatsoever of the PIA interrupt structure which was the basis of the old (CoCo 2) system of interrupts.

The IRQ Enable/Status Register and FIRQ Enable/Status Register are located at $FF92 and $FF93 respectively. These registers are functionally identical, and are defined according to Figure 14.

Bit 7		Undefined
Bit 6		Undefined
Bit 5	TMR	Timer
Bit 4	HBORD	Horizontal Border
Bit 3	VBORD	Vertical Border
Bit 2	EI2	Serial Data
Bit 1	EI1	Keyboard
Bit 0	EI0	Cartridge

Figure 14 - Interrupt Enable/Status Register

To enable a specific interrupt, simply set the bit in the appropriate enable register. For example, in order to enable the timer to trigger an IRQ interrupt, simply store a $20 in $FF92. It is up to the interrupt servicing routine to determine what caused the interrupt, which is done by reading the appropriate status register. For example, if we have set up the interrupts to trigger an FIRQ interrupt when a key is pressed, the service routine should contain the following code to make sure the keyboard generated the interrupt:

```
LDA  $FF93  READ INTERRUPT STATUS REGISTER
BITA #2     CHECK FOR KEYBOARD INTERRUPT
BEQ         BRANCH IF NOT KEY
```

In addition to determining the source of the interrupt, reading the status register resets the interrupt flags (those same flags that told you where the interrupt originated). The programmer must preserve the contents of the status register if you wish to make use of their contents after the status register has been read.

The GIME chip interrupts are triggered on the high to low transition of the interrupt source when the enable line is high. The design of the interrupt input circuitry also causes an interrupt to occur if the interrupt source is high when the enable line is brought low. This will cause a spurious interrupt, which your interrupt handling routines must detect and reject. A current anomaly in the interrupt circuitry causes the interrupt status register to be cleared when a zero is written to the interrupt enable bit.

The Keyboard Interrupt

One of the exciting new interrupts included in the Color Computer 3 is the keyboard interrupt. When set up properly, the user program can continue execution without continually checking to see if a key is down. When a key is pressed, an interrupt is generated. At this point, the interrupt servicing routine can determine which key was pressed and process it.

To set up the keyboard interrupt, several things must be done. First of all, the interrupt enable/status registers must be turned on by setting the appropriate bits in $FF9Ø (as discussed above). Then, the keyboard interrupt itself must be enabled by setting bit 1 of the appropriate interrupt enable register. Lastly, the keyboard strobe lines must be reset by storing a Ø at $FFØ2. Once this has been done, pressing a key on the keyboard or pressing a joystick button will generate an interrupt.

The Timer Interrupt

The timer is a 12-bit interval timer located at $FF94-$FF95. When a value is loaded into the most significant byte ($FF94), the count is automatically started. The input clock is set to either 14 MHz or horizontal sync, as selected by setting or clearing bit 5 of $FF91. As the count falls through zero, an interrupt is generated (if enabled), and the count is automatically reloaded. Setting bit 5 of the appropriate interrupt enable register enables the timer interrupt.

The HBORD, VBORD, EI2, and EI0 Interrupts

The other interrupts are similar to their counterparts in the Color Computer 2. HBORD causes an interrupt at the falling edge of the horizontal sync (the Color Computer 2 actually generated this interrupt at the blanking pulse a subtle difference). The VBORD interrupt is generated at the falling edge of the vertical sync. The EI2 interrupt is connected to the status line of the RS-232C serial connector (printer port), and the EIØ interrupt is connected to the expansion (ROM PAK) port.

An Example

Lastly, as an example, let s set up the computer to generate an IRQ interrupt when a key is pressed on the keyboard. The following assembly code would produce this result:

```
First enable the IRO interrupt
      LDA #$2Ø    CODE TO ENABLE IRQ INTERRUPT
      STA $FF9Ø   TURN ON INTERRUPT
Now enable the Keyboard Interrupt at $FF92
      LDA #2      CODE TO ENABLE KEYBOARD INTERRUPT
      STA $FF92   ENABLE KEYBOARD IRQ
      CLR $FFØ2   CLEAR KEYBOARD STROBE LINES
```

The service routine, of course, would read $FF92 and check to make sure the keyboard interrupt was responsible for the interrupt.

Interrupt Vectors

When an interrupt occurs, the computer must know where to go to process the interrupt. To find this information, the computer looks into the $FFØ2 - $FFFF range, which is defined as follows:

Address	Interrupt	CoCo 2 Vector	CoCo 3 Vector
$FFF2	SWI3	$1ØØ	$FEEE
$FFF4	SWI2	$1Ø3	$FEF1
$FFF6	FIRQ	$1ØF	$FEF4
$FEF8	IRQ	$1ØC	$FEF7
$FFFA	SWI	$1Ø6	$FEFA
$FFFC	NMI	$1Ø9	$FEFD
$FFFE	RESET	$A027	$8C1B

Figure 15 - Interrupt Vectors

When an interrupt such as IRQ interrupt occurs, control is transferred to the interrupt vector table ($FFFØ-$FFFF) as shown in Figure 15. The GIME chip (and the SAM chip in the older CoCos) redirect the CPU s address request from the $FFFØ-$FFFF range to $BFFØ-$BFFF so that the interrupt vectors can be stored in the Basic ROM. In the original Color Computer, control would then be sent to $1ØC. At this address was (and still is) a jump table which redirects control to the desired IRQ routine. Since this jump table is in RAM, it may be modified by Extended Basic, Disk Basic, or any user program.

In the Color Computer 3, there is no guarantee that Basic or the Basic jump table is in memory (because of the MMU). For this reason, an intermediate jump table was made in RAM in the $FEEE range, which can be forced to be in the logical address space at all times. This jump table, when Basic is running, contains LBRAs to the appropriate address in Basic s interrupt jump table ($1ØØ). This also means that when a user program wants to replace the memory at $1ØØ - $111, it should deal with the interrupts at the intermediate jump table at $FEEE. For example, if a user program wishes to replace the IRQ vector, the following code could be used:

```
ORCC  #$5Ø        TURN OFF INTERRUPTS DURING CHANGE
LDA   #$7E        OP CODE FOR JMP INSTRUCTION
STA   $FEF7       REPLACE LBRA WITH JMP
LDX   #SERVIC     POINT X TO SERVICE ROUTINE
STX   $FEF8       PLACE ADDRESS AFTER JMP ADDRESS
ANDCC #$AF        TURN INTERRUPTS BACK ON
```

SUPER EXTENDED BASIC

Super Extended Basic has two major functions. First of all, it provides the necessary machine code to initialize the computer and make Basic work therein. Secondly, several new Commands have been added, primarily to make use of the advanced graphics and memory capabilities. In the following pages, we will discuss each of these functions

Initialization

In addition to the all RAM mode (where all memory in the computer is RAM), there are 3 different ROM configurations (where some of the memory in the computer is in ROM). ROM may be configured as one 32K block inside the computer ($8000-$FDFF), 16K inside the computer ($8000-$BFFF) and 16K from the cartridge port ($C000-$FDFF), or 32K (except for the interrupt vectors) accessed through the cartridge port.

When the Color Computer 3 is turned on, the system is set up for 32K of ROM inside the computer. After some preliminary initialization, a routine is copied from the 32K ROM to $4000 in RAM and executed. This routine copies Extended Color Basic, Super Extended Basic and Disk Basic (if available) into RAM. Once this is done, the routine patches several of the routines in Basic to work in the Color Computer 3. Unfortunately, the authors did not include patches that would fix any of the inherent bugs in the old Basic. The main benefit of this complex system, as far as the user is concerned, is that Basic is now located in RAM, and is easily changed by pokes.

The initialization routine for the Color Computer 3 begins at $8C1B. This code writes over the DLOAD routine that was in the original Color Computer (actually, typing DLOAD will simulate pressing the reset button). This initialization routine is used for both a warm start (simply getting control of the computer back from a runaway program) and a cold start (where the computer and Basic have to be reinitialized). In the following paragraphs, we will discuss the fundamental steps used to initialize the system.

The body of the initialization routine is located in the 32K internal ROM at $C000. Therefore, one of the first actions which the routine at $8C1B does is to enable the 32K internal ROM and jump to $C000. The routine then does the following steps. (in order).

1) Clear the Screen. The screen is cleared by storing $12s in all of the palette registers. Note that the memory where the screen is pointing is not necessarily clear, just all of the differing values display identical colors.

2) Set up MMU Registers. The routine initializes the MMU registers to values it needs.

3) Copy Initialization Routine. The routine that copies Basic into RAM and patches the code is moved to $4000 in RAM. This insures that it will be there with all configurations of ROM/RAM. Control is then transferred to this routine.

4) Text Screen Display Set. The Video Registers are set up to display the 32x16 text screen.

5) Initialize Registers. The Peripheral Interface Adapters (PIAs) and SAM registers are initialized.

6) F1 Key Check. The F1 key is polled, and a flag is set if the key is down (this is used to force the alternate color set).

7) ALT and CTRL Keys Checked. If the ALT and CTRL keys are both pressed, control is transferred to another routine that displays a digitized picture of Basic s authors.

8) Check the flag at $FFED (INT.FLAG). If this flag is not $55 (which would indicate that it was set up before), control is transferred and a cold start is forced.

9) Check Reset Flag. Next the Reset Flag (RSTFLG, $71) is checked. If it is not $55 (indicating that Basic has already been initialized), a cold start is forced. Otherwise, the warm start routine is executed.

The Warm Start Routine

The warm start routine is used when the initialization routine has determined that Basic is still intact. First, the address of the warm start routine is retrieved from the Reset Vector (RSTVEC, $72). Next, the first byte at this address is checked. If it is a NOP instruction, control is transferred to this warm start address. Otherwise, a cold start is forced.

Cold Start

First, Basic, Extended Basic, Disk Basic (if there), and Super Extended Basic are copied into RAM. Next, several patches are made in Basic, Extended Basic, and Disk Basic (these patches are detailed in Appendix B, $C256). The intermediate jump table for the interrupts is then moved to $FFEE (as well as the flag at $FFED discussed earlier). If the flag indicating the alternate color set was chosen (i.e. the F1 key was down), the color set is selected. Next the low-resolution text screen is cleared to spaces. Lastly the palette registers are set to their default values and control is transferred to the reset address in Basic (at $AØ27).

New Commands

Shortly after a prototype Color Computer 3 was created, Tandy contracted with Microware in Des Moines, Iowa (the makers of OS-9) to upgrade Basic to work with the new features of the computer. Microware decided that the best system to use would be to patch Basic during the initialization of the computer. The result of this is a somewhat complicated system of ROM and RAM switching.

The Color Computer 3 added several new commands to Basic, including ON ERROR and ON BREAK trapping, high-resolution text commands, and high-resolution graphics commands. You can even print characters on a hi-res graphics screen!

Most of the routines that make up the super high-resolution graphics commands (HPAINT, HDRAW, HLINE, etc) were derived from the related commands in Extended Basic. Though mimicking these routines is not necessarily a bad philosophy, the Extended Basic routines were never designed to handle 64Ø pixel wide screens. Unfortunately, very little was done to increase the resolution of the routines. The most obvious example of this is the HCIRCLE command, which has little more detail on the super high-resolution screens than on the low-resolution screens.

Inconsistencies

In upgrading the graphics commands to work on the Color Computer 3 some of the conventions used in Extended Basic were ignored. The most apparent example of this is the HSCREEN command. Extended Basic requires that you set up the graphics

mode using the PMODE command, then (if you wish) clear the screen using the PCLS command, and lastly display the screen with the SCREEN command. Super Extended Basic has replaced all of these commands with one command, HSCREEN, which sets the mode, clears the screen, and displays the screen. This does not allow you to view a screen loaded in from disk or cassette, create the screen before viewing it (which would be helpful with 32K screens), or switch between the text and graphics modes without redrawing the graphics screen.

The original programmers of Basic also went to great lengths to allow you to draw the same picture on a higher resolution PMODE by simply changing the PMODE command. All coordinates are Ø-255 across and Ø-191 vertically, no matter what graphics mode you are using. Unfortunately, the new authors did not adhere to this convention, and the coordinates for drawing on the super high-resolution screen must change depending upon the HSCREEN resolution you are using.

There are several key routines within Basic, which are described in the back of the Basic User s manual (ROM ROUTINES). Programmers have been encouraged to use the indirect calls to these routines, as they are the only calls supported by Tandy. One of these calls (CHROUT) prints a character to a device (Ø = screen, -1 = cassette, -2 = printer). The code for this device is located at $6F (DEVNUM). Basic 2.Ø now also checks the byte at $E7 (HRWIDTH). If this byte is Ø, text is printed to the standard 32 x 16 text screen. Otherwise, text is printed to the hi-res text (HRWIDTH) screen. This change is not documented in the Basic manual. Many CoCo 2 programs use the official CHROUT ROM call, but do not insure that HRWIDTH is zero. This will cause Basic to attempt to write its message on the hi-res screen with unpredictable results since using the hi-res screen is not supported by the CoCo 2.

Inefficiencies

Several aspects of Super Extended Basic are somewhat inefficient. It is, unfortunately, clear that the people who wrote Super Extended Basic did not use Color Basic regularly. The most glaring example is the omission of a routine that would save a super high-resolution screen to disk or tape.

When Basic version 1.2 was released, one of the changes was an alteration to the Read Key routine. The result was that Basic ran faster (instead of individually checking each key to see if it was down). Basic was changed to first check to see if any key was down). Basic 2.Ø changed this upgrade back to the original method. There were probably intentions of making Basic work with the keyboard interrupt, then the idea was scrapped and the patch accidentally left in.

Ram Hooks

Many of the Super Extended Basic command and functions have been provided with a pseudo RAM hook. Since Basic is run in RAM in the CoCo 3, it doesn t really make sense to call them a RAM hook but it does make it easier to draw a parallel to the RAM hooks used in the earlier versions of Basic. The RAM hooks come in the form of a LBRN Ø instruction. This is a convenient way to allow the user to patch or modify any of the routines, which have a RAM hook. Of course, it should be obvious that ANY Basic routine may be easily patched in the normal manner if the user desires to do so since Basic runs in RAM.

Bugs

There are several bugs within Super Extended Basic. Some are minor and without too much consequence. Others, however, are potentially disastrous. Here are a few of the more important ones.

Any Basic program containing Disk Basic commands must be listed out with Disk Basic installed. If you try to list the program without Disk Basic, the computer will hang. For example, let s look at the line 1Ø KILL TEMP/DAT . The program will load into a system that does not have Disk Basic installed. The program will even run and will return an ?SN ERROR IN LINE 1Ø. However, when the line is LISTED, the computer will hang (Basic gets confused when it can t find the word KILL for the Basic token in line 1Ø).

The Super Extended function tokens have been forced to start at $29. They should be forced to start at $28. As a result, function token number $28 will never be used. This is not a bug of earth shaking proportions, but one should be aware of it.

The ERLIN function will return a negative number if the line number in which the error occurred is greater than 32767. This is caused by the fact that the ERLIN function returns the line number as a two-byte integer instead of a floating point number, as it should.

Extended Basic graphics commands (LINE, CIRCLE, DRAW, etc.) don t work well with their Super Extended Basic counterparts (HLINE, HCIRCLE, HDRAW, etc). For example, the command HLINE -(192,639),PSET:LINE -(Ø,Ø),PSET will cause problems (often destroying the Basic program). This means you must be very careful to include the H before the Super Extended graphic commands. These problems are caused by the fact that Super Extended graphics routines such as HDRAW, HLINE, HCIRCLE etc. use the same direct page variables as their lower resolution Extended Basic counterparts. As a result, mixing up the two types of commands may cause problems.

HDRAW does not work properly with relative motion in the negative direction that is greater than 255. For example: HDRAW BM-32Ø . The distance is not calculated property due to an error in the negate routine.

HPUT will not work with the NOT action. The command is supposed to reverse the image in the HGET/HPUT buffer and place it on the screen. Because of the bug, the command reverses the specified section of the screen and does nothing with the image.

The RGB and CMP commands function by copying an image of the palette registers from RAM into the palette registers. As they now stand, these commands will only copy 15 instead of 16 palette registers when invoked. Palette register 15 is not copied which generally will not cause problems but the user should be aware of this flaw in the RGB and CMP commands.

Listed below are the Spectral approved fixes for the easily fixable bugs listed above.

To force HSCREEN to clear the hi-res screen:

```
POKE &HE6C6,&H21
```

To fix the RGB and CMP commands:

```
POKE &HE64C,16    -
```

To fix the HPUT NOT option:

```
POKE &HEF13,&HC4
```

```
To fix the HDRAW bug:
      POKE &HF58D,&HBD        JSR $F4CC
      POKE &HF58E,&HF4
      POKE &HF58F,&HCC
```

```
0001  C000  ROMPAK    EQU  $C000
0002
0003  0008  BS        EQU  8              BACKSPACE
0004  000D  CR        EQU  $D             ENTER KEY
0005  001B  ESC       EQU  $1B            ESCAPE CODE
0006  000A  LF        EQU  $A             LINE FEED
0007  000C  FORMF     EQU  $C             FORM FEED
0008  0020  SPACE     EQU  $20            SPACE (BLANK)
0009
0010  003A  STKBUF    EQU  58             STACK BUFFER ROOM
0011  045E  DEBDEL    EQU  $45E           DEBOUNCE DELAY
0012  00FA  LBUFMX    EQU  250            MAX NUMBER OF CHARS IN A BASIC LINE
0013  00FA  MAXLIN    EQU  $FA            MAXIMUM MS BYTE OF LINE NUMBER
0014
0015  2600  DOSBUF    EQU  $2600          RAM LOAD LOCATION FOR THE DOS COMMAND
0016  0020  DIRLEN    EQU  32             NUMBER OF BYTES IN DIRECTORY ENTRY
0017  0100  SECLEN    EQU  256            LENGTH OF SECTOR IN BYTES
0018  0012  SECMAX    EQU  18             MAXIMUM NUMBER OF SECTORS PER TRACK
0019  1200  TRKLEN    EQU  SECMAX*SECLEN  LENGTH OF TRACK IN BYTES
0020  0023  TRKMAX    EQU  35             MAX NUMBER OF TRACKS
0021  004A  FATLEN    EQU  6+(TRKMAX-1)*2 FILE ALLOCATION TABLE LENGTH
0022  0044  GRANMX    EQU  (TRKMAX-1)*2   MAXIMUM NUMBER OF GRANULES
0023  0119  FCBLEN    EQU  SECLEN+25      FILE CONTROL BLOCK LENGTH
0024  0010  INPFIL    EQU  $10            INPUT FILE TYPE
0025  0020  OUTFIL    EQU  $20            OUTPUT FILE TYPE
0026  0040  RANFIL    EQU  $40            RANDOM/DIRECT FILE TYPE
0027
0028              * PSEUDO PSEUDO OPS
0029  0021  SKP1      EQU  $21            OP CODE OF BRN   SKIP ONE BYTE
0030  008C  SKP2      EQU  $8C            OP CODE OF CMPX # - SKIP TWO BYTES
0031  0086  SKP1LD    EQU  $86            OP CODE OF LDA # - SKIP THE NEXT BYTE
0032              *                        AND LOAD THE VALUE OF THAT BYTE INTO ACCA   THIS
0033              *                        IS USUALLY USED TO LOAD ACCA WITH A NON ZERO VALUE
0034
0035              * SUPER EXTENDED BASIC EQUATES
0036  0018  ROWMAX    EQU  24             MAXIMUM NUMBER OF ROWS IN HI-RES PRINT MODE
0037  0000  RAMLINK   EQU  0              DUMMY RAM LINK VECTOR
0038  2000  HRESSCRN  EQU  $2000          ADDRESS OF THE HI-RES SCREEN IN THE CPU'S MEMORY SPACE
0039  C000  HRESBUFF  EQU  $C000          ADDRESS OF THE GET/PUT BUFFERS IN THE CPU'S MEMORY SPACE
0040  DFFF  TMPSTACK  EQU  $DFFF          ADDRESS OF THE HI-RES GRAPHICS STACK IN THE CPU'S MEMORY SPACE
0041  0062  EBHITOK   EQU  $62            FIRST ENHANCED BASIC TOKEN NUMBER
0042  0029  EBHISTOK  EQU  $29            FIRST ENHANCED BASIC FUNCTION TOKEN NUMBER BUG - SHOULD BE $28
0043  0020  CURCHAR   EQU  SPACE          HI-RES CURSOR CHARACTER
0044
0045              * HBUFF HGET/HPUT BUFFER HEADER EQUATES
0046  0000  HB.ADDR   EQU  0              ADDRESS OF THE NEXT BUFFER - 2 BYTES
0047  0002  HB.NUM    EQU  2              NUMBER OF THIS BUFFER - 1 BYTES
0048  0003  HB.SIZE   EQU  3              NUMBER OF BYTES IN THE BUFFER - 2 BYTES
0049  0005  HB.LEN    EQU  5              NUMBER OF BYTES IN THIS HEADER
0050
0051              * VIDEO REGISTER EQUATES
0052              * INIT0 BIT EQUATES
0053  0080  COCO      EQU  $80            1 = Color Computer compatible
0054  0040  MMUEN     EQU  $40            1 = MMU enabled
0055  0020  IEN       EQU  $20            1 = GIME chip IRQ output enabled
0056  0010  FEN       EQU  $10            1 = GIME chip FIRQ output enabled
0057  0008  MC3       EQU  8              1 = RAM at XFEXX is constant
0058  0004  MC2       EQU  4              1 = standard SCS
0059  0002  MC1       EQU  2              ROM map control
0060  0001  MC0       EQU  1              ROM map control
0061
0062              * INTERRUPT REQUEST ENABLED
0063  0020  TMR       EQU  $20            TIMER
0064  0010  HBORD     EQU  $10            HORIZONTAL BORDER
0065  0008  VBORD     EQU  8              VERTICAL BORDER
0066  0004  EI2       EQU  4              SERIAL DATA
0067  0002  EI1       EQU  2              KEYBOARD
0068  0001  EI0       EQU  1              CARTRIDGE
0069
0070              * EXPANDED MEMORY DEFINITIONS
0071  0030  BLOCK 6.0 EQU  $30            BLOCKS $30-$33 ARE THE HI-RES GRAPHICS SCREEN
0072  0031  BLOCK 6.1 EQU  $31            HI-RES GRAPHICS SCREEN
0073  0032  BLOCK 6.2 EQU  $32            HI-RES GRAPHICS SCREEN
0074  0033  BLOCK 6.3 EQU  $33            HI-RES GRAPHICS SCREEN
```

```
0075       0034      BLOCK 6.4 EQU   $34             GET/PUT BUFFER
0076       0035      BLOCK 6.5 EQU   $35             STACK AREA FOR HI-RES GRAPHICS COMMAND
0077       0036      BLOCK 6.6 EQU   $36             CHARACTER POINTERS
0078       0037      BLOCK 6.7 EQU   $37             UNUSED BY BASIC
0079
0080                 * BLOCKS $48-$4F ARE USED FOR THE BASIC OPERATING SYSTEM
0081       0038      BLOCK7.0  EQU   $38
0082       0039      BLOCK7.1  EQU   $39
0083       003A      BLOCK7.2  EQU   $3A
0084       003B      BLOCK7.3  EQU   $3B
0085       003C      BLOCK7.4  EQU   $3C
0086       003D      BLOCK7.5  EQU   $3D
0087       003E      BLOCK7.6  EQU   $3E
0088       003F      BLOCK7.7  EQU   $3F
0089
0090
0091
0092 0000                    ORG   0
0093       0000              SETDP 0
0094
0095 0000           ENDFLG    RMB   1               STOP/END FLAG: POSITIVE=STOP, NEG=END
0096 0001           CHARAC    RMB   1               TERMINATOR FLAG 1
0097 0002           ENDCUR    RMB   1               TERMINATOR FLAG 2
0098 0003           TMPLOC    RMB   1               SCRATCH VARIABLE
0099 0004           IFCTR     RMB   1               IF COUNTER - HOW MANY IF STATEMENTS IN A LINE
0100 0005           DIMFLG    RMB   1               *DV* ARRAY FLAG 0=EVALUATE, 1=DIMENSIONING
0101 0006           VALTYP    RMB   1               *DV* *PV TYPE FLAG: 0=NUMERIC, $FF=STRING
0102 0007           GARBFL    RMB   1               *TV STRING SPACE HOUSEKEEPING FLAG
0103 0008           ARYDIS    RMB   1               DISABLE ARRAY SEARCH: 00=ALLOW SEARCH
0104 0009           INPFLG    RMB   1               *TV INPUT FLAG: READ=0, INPUT<>0
0105 000A           RELFLG    RMB   1               *TV RELATIONAL OPERATOR FLAG
0106 000B           TEMPPT    RMB   2               *PV TEMPORARY STRING STACK POINTER
0107 000D           LASTPT    RMB   2               *PV ADDR OF LAST USED STRING STACK ADDRESS
0108 000F           TEMPTR    RMB   2               TEMPORARY POINTER
0109 0011           TMPTR1    RMB   2               TEMPORARY DESCRIPTOR STORAGE (STACK SEARCH)
0110                ** FLOATING POINT ACCUMULATOR #2 (MANTISSA ONLY)
0111 0013           FPA2      RMB   4               FLOATING POINT ACCUMULATOR #2 MANTISSA
0112 0017           BOTSTK    RMB   2               BOTTOM OF STACK AT LAST CHECK
0113 0019           TXTTAB    RMB   2               *PV BEGINNING OF BASIC PROGRAM
0114 001B           VARTAB    RMB   2               *PV START OF VARIABLES
0115 001D           ARYTAB    RMB   2               *PV START OF ARRAYS
0116 001F           ARYEND    RMB   2               *PV END OF ARRAYS (+1)
0117 0021           FRETOP    RMB   2               *PV START OF STRING STORAGE (TOP OF FREE RAM)
0118 0023           STRTAB    RMB   2               *PV START OF STRING VARIABLES
0119 0025           FRESPC    RMB   2               UTILITY STRING POINTER
0120 0027           MEMSIZ    RMB   2               *PV TOP OF STRING SPACE
0121 0029           OLDTXT    RMB   2               SAVED LINE NUMBER DURING A "STOP"
0122 002B           BINVAL    RMB   2               BINARY VALUE OF A CONVERTED LINE NUMBER
0123 002D           OLDPTR    RMB   2               SAVED INPUT PTR DURING A "STOP"
0124 002F           TINPTR    RMB   2               TEMPORARY INPUT POINTER STORAGE
0125 0031           DATTXT    RMB   2               *PV 'DATA' STATEMENT LINE NUMBER POINTER
0126 0033           DATPTR    RMB   2               *PV 'DATA' STATEMENT ADDRESS POINTER
0127 0035           DATTMP    RMB   2               DATA POINTER FOR 'INPUT' & 'READ'
0128 0037           VARNAM    RMB   2               *TV TEMP STORAGE FOR A VARIABLE NAME
0129 0039           VARPTR    RMB   2               *TV POINTER TO A VARIABLE DESCRIPTOR
0130 003B           VARDES    RMB   2               TEMP POINTER TO A VARIABLE DESCRIPTOR
0131 003D           RELPTR    RMB   2               POINTER TO RELATIONAL OPERATOR PROCESSING ROUTINE
0132 003F           TRELFL    RMB   1               TEMPORARY RELATIONAL OPERATOR FLAG BYTE
0133
0134                * FLOATING POINT ACCUMULATORS #3,4 & 5 ARE MOSTLY
0135                * USED AS SCRATCH PAD VARIABLES.
0136                ** FLOATING POINT ACCUMULATOR #3 :PACKED: ($40-$44)
0137 0040           V40       RMB   1
0138 0041           V41       RMB   1
0139 0042           V42       RMB   1
0140 0043           V43       RMB   1
0141 0044           V44       RMB   1
0142                ** FLOATING POINT ACCUMULATOR #4 :PACKED: ($45-$49)
0143 0045           V45       RMB   1
0144 0046           V46       RMB   1
0145 0047           V47       RMB   1
0146 0048           V48       RMB   2
0147                ** FLOATING POINT ACCUMULATOR #5 :PACKED: ($4A $4E)
0148 004A           V4A       RMB   1
```

```
0149 004B           V4B       RMB   2
0150 004D           V4D       RMB   2
0151                ** FLOATING POINT ACCUMULATOR #0
0152 004F           FP0EXP    RMB   1               *PV FLOATING POINT ACCUMULATOR #0 EXPONENT
0153 0050           FPA0      RMB   4               *PV FLOATING POINT ACCUMULATOR #0 MANTISSA
0154 0054           FP0SGN    RMB   1               *PV FLOATING POINT ACCUMULATOR #0 SIGN
0155 0055           COEFCT    RMB   1               POLYNOMIAL COEFFICIENT COUNTER
0156 0056           STRDES    RMB   5               TEMPORARY STRING DESCRIPTOR
0157 005B           FPCARY    RMB   1               FLOATING POINT CARRY BYTE
0158                ** FLOATING POINT ACCUMULATOR #1
0159 005C           FP1EXP    RMB   1               *PV FLOATING POINT ACCUMULATOR #1 EXPONENT
0160 005D           FPA1      RMB   4               *PV FLOATING POINT ACCUMULATOR #1 MANTISSA
0161 0061           FP1SGN    RMB   1               *PV FLOATING POINT ACCUMULATOR #1 SIGN
0162
0163 0062           RESSGN    RMB   1               SIGN OF RESULT OF FLOATING POINT OPERATION
0164 0063           FPSBYT    RMB   1               FLOATING POINT SUB BYTE (FIFTH BYTE)
0165 0064           COEFPT    RMB   2               POLYNOMIAL COEFFICIENT POINTER
0166 0066           LSTTXT    RMB   2               CURRENT LINE POINTER DURING LIST
0167 0068           CURLIN    RMB   2               *PV CURRENT LINE # OF BASIC PROGRAM, $FFFF = DIRECT
0168 006A           DEVCFW    RMB   1               *TV TAB FIELD WIDTH
0169 006B           DEVLCF    RMB   1               *TV TAB ZONE
0170 006C           DEVPOS    RMB   1               *TV PRINT POSITION
0171 006D           DEVWID    RMB   1               *TV PRINT WIDTH
0172 006E           PRTDEV    RMB   1               *TV PRINT DEVICE: 0=NOT CASSETTE, -1=CASSETTE
0173 006F           DEVNUM    RMB   1               *PV DEVICE NUMBER: -3=DLOAD, -2=PRINTER,
0174                *                                                  -1=CASSETTE, 0=SCREEN, 1-15=DISK
0175 0070           CINBFL    RMB   1               *PV CONSOLE IN BUFFER FLAG: 00=NOT EMPTY, $FF=EMPTY
0176 0071           RSTFLG    RMB   1               *PV WARM START FLAG: $55=WARM, OTHER=COLD
0177 0072           RSTVEC    RMB   2               *PV WARM START VECTOR - JUMP ADDRESS FOR WARM START
0178 0074           TOPRAM    RMB   2               *PV TOP OF RAM
0179 0076                     RMB   2               SPARE: UNUSED VARIABLES
0180 0078           FILSTA    RMB   1               *PV FILE STATUS FLAG: 0=CLOSED, 1=INPUT, 2=OUTPUT
0181 0079           CINCTR    RMB   1               *PV CONSOLE IN BUFFER CHAR COUNTER
0182 007A           CINPTR    RMB   2               *PV CONSOLE IN BUFFER POINTER
0183 007C           BLKTYP    RMB   1               *TV CASS BLOCK TYPE: 0=HEADER, 1=DATA, $FF=EOF
0184 007D           BLKLEN    RMB   1               *TV CASSETTE BYTE COUNT
0185 007E           CBUFAD    RMB   2               *TV CASSETTE LOAD BUFFER POINTER
0186 0080           CCKSUM    RMB   1               *TV CASSETTE CHECKSUM BYTE
0187 0081           CSRERR    RMB   1               *TV ERROR FLAG/CHARACTER COUNT
0188 0082           CPULWD    RMB   1               *TV PULSE WIDTH COUNT
0189 0083           CPERTM    RMB   1               *TV BIT COUNTER
0190 0084           CBTPHA    RMB   1               *TV BIT PHASE FLAG
0191 0085           CLSTSN    RMB   1               *TV LAST SINE TABLE ENTRY
0192 0086           GRBLOK    RMB   1               *TV GRAPHIC BLOCK VALUE FOR SET, RESET AND POINT
0193 0087           IKEYIM    RMB   1               *TV INKEY$ RAM IMAGE
0194 0088           CURPOS    RMB   2               *PV CURSOR LOCATION
0195 008A           ZERO      RMB   2               *PV DUMMY - THESE TWO BYTES ARE ALWAYS ZERO
0196 008C           SNDTON    RMB   1               *TV TONE VALUE FOR SOUND COMMAND
0197 008D           SNDDUR    RMB   2               *TV DURATION VALUE FOR SOUND COMMAND
0198
0199                ** THESE BYTES ARE MOVED DOWN FROM ROM
0200                ***                             INIT   DESCRIPTION
0201                *                               VALUE
0202 008F           CMPMID    RMB   1               18     *PV 1200/2400 HERTZ PARTITION
0203 0090           CMP0      RMB   1               24     *PV UPPER LIMIT OF 1200 HERTZ PERIOD
0204 0091           CMP1      RMB   1               10     *PV UPPER LIMIT OF 2400 HERTZ PERIOD
0205 0092           SYNCLN    RMB   2               128    *PV NUMBER OF $55'S TO CASSETTE LEADER
0206 0094           BLKCNT    RMB   1               11     *PV CURSOR BLINK DELAY
0207 0095           LPTBTD    RMB   2               88     *PV BAUD RATE CONSTANT (600)
0208 0097           LPTLND    RMB   2               1      *PV PRINTER CARRIAGE RETURN DELAY
0209 0099           LPTCFW    RMB   1               16     *PV TAB FIELD WIDTH
0210 009A           LPTLCF    RMB   1               112    *PV LAST TAB ZONE
0211 009B           LPTWID    RMB   1               132    *PV PRINTER WIDTH
0212 009C           LPTPOS    RMB   1               0      *PV LINE PRINTER POSITION
0213 009D           EXECJP    RMB   2               LB4AA  *PV JUMP ADDRESS FOR EXEC COMMAND
0214
0215                ** THIS ROUTINE PICKS UP THE NEXT INPUT CHARACTER FROM
0216                ** BASIC. THE ADDRESS OF THE NEXT BASIC BYTE TO BE
0217                ** INTERPRETED IS STORED AT CHARAD.
0218
0219 009F 0C A7     GETNCH    INC   <CHARAD+1       *PV INCREMENT LS BYTE OF INPUT POINTER
0220 00A1 26 02              BNE   GETCCH          *PV BRANCH IF NOT ZERO (NO CARRY)
0221 00A3 0C A6              INC   <CHARAD         *PV INCREMENT MS BYTE OF INPUT POINTER
0222 00A5 B6        GETCCH    FCB   $B6             *PV OP CODE OF LDA EXTENDED
```

```
0223 00A6           CHARAD          2               *PV THESE 2 BYTES CONTAIN ADDRESS OF THE CURRENT
0224                *                                   CHARACTER WHICH THE BASIC INTERPRETER IS
0225                *                                   PROCESSING
0226 00A8 7E AA 1A          JMP   BROMHK           JUMP BACK INTO THE BASIC RUM
0227
0228 00AB           VAB         RMB   1               = LOW ORDER FOUR BYTES OF THE PRODUCT
0229 00AC           VAC         RMB   1               = OF A FLOATING POINT MULTIPLICATION
0230 00AD           VAD         RMB   1               = THESE BYTES ARE USE AS RANDOM DATA
0231 00AE           VAE         RMB   1               = BY THE RND STATEMENT
0232
0233                * EXTENDED BASIC VARIABLES
0234 00AF           TRCFLG      RMB   1               *PV TRACE FLAG 0=OFF ELSE=ON
0235 00B0           USRADR      RMB   2               *PV ADDRESS OF THE START OF USR VECTORS
0236 00B2           FORCOL      RMB   1               *PV FOREGROUND COLOR
0237 00B3           BAKCOL      RMB   1               *PV BACKGROUND COLOR
0238 00B4           WCOLOR      RMB   1               *TV WORKING COLOR BEING USED BY EX BASIC
0239 00B5           ALLCOL      RMB   1               *TV ALL PIXELS IN THIS BYTE SET TO COLOR OF VB3
0240 00B6           PMODE       RMB   1               *PV PMODE'S MODE ARGUMENT
0241 00B7           ENDGRP      RMB   2               *PV END OF CURRENT GRAPHIC PAGE
0242 00B9           HORBYT      RMB   1               *PV NUMBER OF BYTES/HORIZONTAL GRAPHIC LINE
0243 00BA           BEGGRP      RMB   2               *PV START OF CURRENT GRAPHIC PAGE
0244 00BC           GRPRAM      RMB   1               *PV START OF GRAPHIC RAM (MS BYTE)
0245 00BD           HORBEG      RMB   2               *DV* *PV HORIZ COORD - START POINT
0246 00BF           VERBEG      RMB   2               *DV* *PV VERT COORD - START POINT
0247 00C1           CSSYAL      RMB   1               *PV SCREEN'S COLOR SET ARGUMENT
0248 00C2           SETFLG      RMB   1               *PV PRESET/PSET FLAG: 0=PRESET, 1=PSET
0249 00C3           HOREND      RMB   2               *DV* *PV HORIZ COORD - ENDING POINT
0250 00C5           VEREND      RMB   2               *DV* *PV VERT COORD - ENDING POINT
0251 00C7           HORDEF      RMB   2               *PV HORIZ COORD - DEFAULT COORD
0252 00C9           VERDEF      RMB   2               *PV VERT COORD - DEFAULT COORD
0253
0254                * EXTENDED BASIC SCRATCH PAD VARIABLES
0255 00CB           VCB         RMB   2
0256 00CD           VCD         RMB   2
0257 00CF           VCF         RMB   2
0258 00D1           VD1         RMB   2
0259 00D3           VD3         RMB   1
0260 00D4           VD4         RMB   1
0261 00D5           VD5         RMB   1
0262 00D6           VD6         RMB   1
0263 00D7           VD7         RMB   1
0264 00D8           VD8         RMB   1
0265 00D9           VD9         RMB   1
0266 00DA           VDA         RMB   1
0267
0268 00DB           CHGFLG      RMB   1               *TV FLAG TO INDICATE IF GRAPHIC DATA HAS BEEN CHANGED
0269 00DC           TMPSTK      RMB   2               *TV STACK POINTER STORAGE DURING PAINT
0270 00DE           OCTAVE      RMB   1               *PV OCTAVE VALUE (PLAY)
0271 00DF           VOLHI       RMB   1               *DV* *PV VOLUME HIGH VALUE (PLAY)
0272 00E0           VOLLOW      RMB   1               *DV* *PV VOLUME LOW VALUE (PLAY)
0273 00E1           NOTELN      RMB   1               *PV NOTE LENGTH (PLAY)
0274 00E2           TEMPO       RMB   1               *PV TEMPO VALUE (PLAY)
0275 00E3           PLYTMR      RMB   2               *TV TIMER FOR THE PLAY COMMAND
0276 00E5           DOTYAL      RMB   1               *TV DOTTED NOTE TIMER SCALE FACTOR
0277 00E6           HRMODE      EQU   *               SUPER EXTENDED BASIC HI-RES MODE
0278 00E6           DLBAUD      RMB   1               *DV* *PV DLOAD BAUD RATE CONSTANT $B0=300, $2C=1200
0279 00E7           HRWIDTH     EQU   *               SUPER EXTENDED BASIC HI-RES TEXT MODE
0280 00E7           TIMOUT      RMB   1               *DV* *PV DLOAD TIMEOUT CONSTANT
0281 00E8           ANGLE       RMB   1               *DV* *PV ANGLE VALUE (DRAW)
0282 00E9           SCALE       RMB   1               *DV* *PV SCALE VALUE (DRAW)
0283
0284                * DSKCON VARIABLES
0285 00EA           DCOPC       RMB   1               *PV DSKCON OPERATION CODE 0-3
0286 00EB           DCDRV       RMB   1               *PV DSKCON DRIVE NUMBER 0 3
0287 00EC           DCTRK       RMB   1               *PV DSKCON TRACK NUMBER 0 34
0288 00ED           DSEC        RMB   1               *PV DSKCON SECTOR NUMBER 1-18
0289 00EE           DCBPT       RMB   2               *PV DSKCON DATA POINTER
0290 00F0           DCSTA       RMB   1               *PV DSKCON STATUS BYTE
0291
0292 00F1           FCBTMP      RMB   2               TEMPORARY FCB POINTER
0293
0294 00F3                       RMB   13              SPARE: UNUSED VARIABLES
0295
0296
```

```
0297                *                               BASIC  EXBASI(DOSBASIC
0298
0299 0100           SW3VEC     RMB   3              $XXXX  $XXXX  $3B3B  SWI3 VECTOR
0300 0103           SW2VEC     RMB   3              $XXXX  $XXXX  $3B3B  SWI2 VECTOR
0301 0106           SWIVEC     RMB   3              $XXXX  $XXXX  $XXXX  SWI VECTOR
0302 0109           NMIVEC     RMB   3              $XXXX  $XXXX  $D7AE  NMI VECTOR
0303 010C           IRQVEC     RMB   3              $A9B3  $894C  $D7BC  IRQ VECTOR
0304 010F           FRQVEC     RMB   3              $A0F6  $A0F6  $A0F6  FIRQ VECTOR
0305
0306 0112           TIMVAL
0307 0112           USRJMP     RMB   3              JUMP ADDRESS FOR BASIC'S USR FUNCTION
0308                *          RMB   2              TIMER VALUE FOR EXBAS
0309                *          RMB   1              UNUSED BY EXBAS OR DISK BASIC
0310 0115           RVSEED     RMB   1              * FLOATING POINT RANDOM NUMBER SEED EXPONENT
0311 0116                      RMB   4              * MANTISSA: INITIALLY SET TO $804FC75259
0312 011A           CASFLG     RMB   1              UPPER CASE/LOWER CASE FLAG: $FF=UPPER, 0=LOWER
0313 011B           DEBVAL     RMB   2              KEYBOARD DEBOUNCE DELAY (SET TO $45E)
0314 011D           EXPJMP     RMB   3              JUMP ADDRESS FOR EXPONENTIATION
0315                **                              INITIALLY SET TO ERROR FOR BASIC, $8489 FOR EX BASIC
0316
0317                ***        COMMAND INTERPRETATION VECTOR TABLE
0318
0319                ** FOUR SETS OF 10 BYTE TABLES:
0320
0321
0322                ** THE LAST USED TABLE MUST BE FOLLOWED BY A ZERO BYTE
0323                * THE JUMP TABLE VECTORS (3,4 AND 8,9) POINT TO THE JUMP TABLE FOR
0324                * THE FIRST TABLE. FOR ALL OTHER TABLES, THESE VECTORS POINT TO A
0325                * ROUTINE WHICH WILL VECTOR YOU TO THE CORRECT JUMP TABLE.
0326                * SUPER ENHANCED BASIC HAS MODIFIED THIS SCHEME SO THAT THE USER
0327                * TABLE MAY NOT BE ACCESSED.  ANY ADDITIONAL TABLES WILL HAVE TO BE
0328                * ACCESSED FROM A NEW COMMAND HANDLER.
0329
0330                *          BYTE  DESCRIPTION
0331                *          0     NUMBER OF RESERVED WORDS
0332                *          1,2   LOOKUP TABLE OF RESERVED WORDS
0333                *          3,4   JUMP TABLE FOR COMMANDS (FIRST TABLE)
0334                *                VECTOR TO EXPANSION COMMAND HANDLERS (ALL BUT FIRST TABLE)
0335                *          5     NUMBER OF SECONDARY FUNCTIONS
0336                *          6,7   LOOKUP TABLE OF SECONDARY FUNCTIONS (FIRST TABLE)
0337                *                VECTOR TO EXPANSION SECONDARY COMMAND HANDLERS (ALL BUT
0338                *                FIRST TABLE)
0339                *          8,9   JUMP TABLE FOR SECONDARY FUNCTIONS
0340                *          10    0 BYTE - END OF TABLE FLAG (LAST TABLE ONLY)
0341
0342 0120           COMVEC     RMB   10             BASIC'S TABLE
0343 012A                      RMB   10             EX BASIC'S TABLE
0344 0134                      RMB   10             DISC BASIC'S TABLE (UNUSED BY EX BASIC)
0345
0346                **** USR FUNCTION VECTOR ADDRESSES (EX BASIC ONLY)
0347 013E                      RMB   2              USR 0 VECTOR
0348 0140                      RMB   2              USR 1
0349 0142                      RMB   2              USR 2
0350 0144                      RMB   2              USR 3
0351 0146                      RMB   2              USR 4
0352 0148                      RMB   2              USR 5
0353 014A                      RMB   2              USR 6
0354 014C                      RMB   2              USR 7
0355 014E                      RMB   2              USR 8
0356 0150                      RMB   2              USR 9
0357
0358                *** THE ABOVE 20 BYTE USR ADDR VECTOR TABLE IS MOVED TO
0359                *** $95F-$972 BY DISC BASIC. THE 20 BYTES FROM $13E-$151
0360                *** ARE REDEFINED AS FOLLOWS:
0361
0362                *          RMB   10             USER (SPARE) COMMAND INTERPRETATION TABLE SPACE
0363                *          FCB   0              END OF COMM INTERP TABLE FLAG
0364                *          RMB   9              UNUSED BY DISK BASIC
0365
0366                *          COMMAND INTERPRETATION TABLE VALUES
0367                *               BYTE            BASIC  EX BAS DISK BASIC
0368                *               0               53                  BASIC TABLE
0369                *               1,2             $AA66
0370                *               3,4             $AB67
```

```
0371                *               5               20
0372                *               6,7             $AB1A
0373                *               8,9             $AA29
0374
0375                *               0                       25            EX BASIC TABLE
0376                *               1,2                     $8183
0377                *               3,4                     $813C  $CE2E  ($CF0A 2.1)
0378                *               5                       14
0379                *               6,7                     $821E
0380                *               8,9                     $8168  $CE56  ($CF32 2.1)
0381
0382                *               0                       19 (20 2.1)   DISK BASIC TABLE
0383                *               1,2                     $C17F
0384                *               3,4                     $C2C0
0385                *               5                       6
0386                *               6,7                     $C201
0387                *               8,9                     $C236
0388
0389
0390 0152           KEYBUF    RMB   8               KEYBOARD MEMORY BUFFER
0391 015A           POTVAL    RMB   1               LEFT VERTICAL JOYSTICK DATA
0392 015B                     RMB   1               LEFT HORIZONTAL JOYSTICK DATA
0393 015C                     RMB   1               RIGHT VERTICAL JOYSTICK DATA
0394 015D                     RMB   1               RIGHT HORIZONTAL JOYSTICK DATA
0395
0396                * BASIC'S RAM VECTORS - INITIALIZED TO RTS BY COLOR BASIC
0397                * 25 SETS OF 3 BYTE INSTRUCTIONS WHICH ARE CALLED BY COLOR BASIC
0398                * EXTENDED AND DISK BASIC. THEIR PURPOSE IS TO ALLOW ENHANCEMENTS (SUCH
0399                * AS EX BASIC AND DOS BASIC) AS MORE ROMS ARE ADDED TO THE
0400                * SYSTEM BY EFFECTIVELY ALLOWING MORE CODE TO BE ADDED TO THE
0401                * ROUTINES IN EARLIER ROMS. THIS NEW CODE IS LOCATED IN THE NEW ROMS
0402                * AND THE ADDRESS TO GET TO THE NEW CODE IS IN BYTES 1 & 2 OF THE
0403                * RAM VECTOR. BYTE 0 WILL CONTAIN A $7E WHICH IS THE FIRST BYTE OF
0404                * THE JMP INSTRUCTION.
0405                * THE FIRST ADDRESS IN THIS TABLE IS THE ADDRESS IN BASIC WHICH
0406                * CALLS THE RAM VECTOR, THE SECOND ADDRESS IS THE VALUE WHICH
0407                * EX BASIC PUTS IN THE RAM VECTOR (IF ANY) AND THE THIRD ADDRESS
0408                * IS THE VALUE WHICH DISK BASIC PUTS THERE (IF ANY)
0409
0410
0411                *                               2.0    2.1    1.0    1.1
0412 015E           RVEC0     RMB   3               $A5F6          $C426  $C44B  OPEN COMMAND
0413 0161           RVEC1     RMB   3               $A5B9          $C838  $C888  DEVICE NUMBER VALIDITY CHECK
0414 0164           RVEC2     RMB   3               $A35F          $C843  $C893  SET PRINT PARAMETERS
0415 0167           RVEC3     RMB   3               $A282  $8273   $CB4A  $CC1C  CONSOLE OUT
0416 016A           RVEC4     RMB   3               $A176  $8CF1   $C58F  $C5BC  CONSOLE IN
0417 016D           RVEC5     RMB   3               $A3ED          $C818  $C848  INPUT DEVICE NUMBER CHECK
0418 0170           RVEC6     RMB   3               $A406          $C81B  $C84B  PRINT DEVICE NUMBER CHECK
0419 0173           RVEC7     RMB   3               $A426          $CA3B  $CAE9  CLOSE ALL FILES
0420 0176           RVEC8     RMB   3               $A42D  $8286   $CA4B  $CAF9  CLOSE ONE FILE
0421 0179           RVEC9     RMB   3               $B918  $8E90   $8E90  $8E90  PRINT
0422 017C           RVEC10    RMB   3               $B061          $CC5B  $CD35  INPUT
0423 017F           RVEC11    RMB   3               $A549          $C859  $C8A9  BREAK CHECK
0424 0182           RVEC12    RMB   3               $A390          $C6B7  $C6E4  INPUTTING A BASIC LINE
0425 0185           RVEC13    RMB   3               $A4BF          $CA36  $CAE4  TERMINATING BASIC LINE INPUT
0426 0188           RVEC14    RMB   3               $A5CE          $CA60  $C90C  EOF COMMAND
0427 018B           RVEC15    RMB   3               $B223  $8846   $CDF6  $CED2  EVALUATE AN EXPRESSION
0428 018E           RVEC16    RMB   3               $AC46          $C6B7  $C6E4  RESERVED FOR ON ERROR GOTO COMMAND
0429 0191           RVEC17    RMB   3               $AC49  $88F0   $C24D  $C265  ERROR DRIVER
0430 0194           RVEC18    RMB   3               $AE75  $829C   $C990  $CA3E  RUN
0431 0197           RVEC19    RMB   3               $BD22  $87EF                 ASCII TO FLOATING POINT CONVERSION
0432 019A           RVEC20    RMB   3               $AD9E  $82B9          $C8B0  BASIC'S COMMAND INTERPRETATION LOOP
0433 019D           RVEC21    RMB   3               $A8C4                        RESET/SET/POINT COMMANDS
0434 01A0           RVEC22    RMB   3               $A910                        CLS
0435                *                               $8162                        EXBAS' SECONDARY TOKEN HANDLER
0436                *                               $8AFA                        EXBAS' RENUM TOKEN CHECK
0437                *                               $975C          $C29A  $C2B2  EXBAS' GET/PUT
0438 01A3           RVEC23    RMB   3               $B821  $8304                 CRUNCH BASIC LINE
0439 01A6           RVEC24    RMB   3               $B7C2                        UNCRUNCH BASIC LINE
0440
0441 01A9           STRSTK    RMB   8*5             STRING DESCRIPTOR STACK
0442 01D1           CFNBUF    RMB   9               CASSETTE FILE NAME BUFFER
0443 01DA           CASBUF    RMB   256             CASSETTE FILE DATA BUFFER
0444 02DA           LINHDR    RMB   2               LINE INPUT BUFFER HEADER
```

```
0445 02DC           LINBUF     RMB   LBUFMX+1        BASIC LINE INPUT BUFFER
0446 03D7           STRBUF     RMB   41              STRING BUFFER
0447
0448 0400           VIDRAM     RMB   200             VIDEO DISPLAY AREA
0449
0450                *START OF ADDITIONAL RAM VARIABLE STORAGE (DISK BASIC ONLY)
0451 0600           DBUF0      RMB   SECLEN          I/O BUFFER #0
0452 0700           DBUF1      RMB   SECLEN          I/O BUFFER #1
0453 0800           FATBL0     RMB   FATLEN          FILE ALLOCATION TABLE - DRIVE 0
0454 084A           FATBL1     RMB   FATLEN          FILE ALLOCATION TABLE - DRIVE 1
0455 0894           FATBL2     RMB   FATLEN          FILE ALLOCATION TABLE - DRIVE 2
0456 08DE           FATBL3     RMB   FATLEN          FILE ALLOCATION TABLE - DRIVE 3
0457 0928           FCBV1      RMB   16*2            FILE BUFFER VECTORS (15 USER, 1 SYSTEM)
0458 0948           RNBFAD     RMB   2               START OF FREE RANDOM FILE BUFFER AREA
0459 094A           FCBADR     RMB   2               START OF FILE CONTROL BLOCKS
0460 094C           DNAMBF     RMB   8               DISK FILE NAME BUFFER
0461 0954           DEXTBF     RMB   3               DISK FILE EXTENSION NAME BUFFER
0462 0957           DFLTYP     RMB   1               *DV* DISK FILE TYPE: 0=BASIC, 1=DATA, 2=MACHINE
0463                *                                     LANGUAGE, 3=TEXT EDITOR SOURCE FILE
0464 0958           DASCFL     RMB   1               *DV* ASCII FLAG: 0=CRUNCHED OR BINARY, $FF=ASCII
0465 0959           DRUNFL     RMB   1               RUN FLAG: (IF BIT 1=1 THEN RUN, IF BIT 0=1, THEN CLOSE
0466                *                                ALL FILES BEFORE RUNNING)
0467 095A           DEFDRV     RMB   1               DEFAULT DRIVE NUMBER
0468 095B           FCBACT     RMB   1               NUMBER OF FCBS ACTIVE
0469 095C           DRESFL     RMB   1               RESET FLAG: <>0 WILL CAUSE A 'NEW' & SHUT DOWN ALL FCBS
0470 095D           DLOADFL    RMB   1               LOAD FLAG: CAUSE A 'NEW' FOLLOWING A LOAD ERROR
0471 095E           DMRGFL     RMB   1               MERGE FLAG: 0=N0 MERGE, $FF=MERGE
0472 095F           DUSRVC     RMB   20              DISK BASIC USR COMMAND VECTORS
0473                *** DISK FILE WORK AREA FOR DIRECTORY SEARCH
0474                *   EXISTING FILE
0475 0973           V973       RMB   1               SECTOR NUMBER
0476 0974           V974       RMB   2               RAM DIRECTORY IMAGE ADDRESS
0477 0976           V976       RMB   1               FIRST GRANULE NUMBER
0478                *   UNUSED FILE
0479 0977           V977       RMB   1               SECTOR NUMBER
0480 0978           V978       RMB   2               RAM DIRECTORY IMAGE ADDRESS
0481
0482 097A           WFATVL     RMB   2               WRITE FAT VALUE: NUMBER OF FREE GRANULES WHICH MUST BE TAKEN
0483                                                 FROM THE FAT TO TRIGGER A WRITE FAT TO DISK SEQUENCE
0484 097C           DFFLEN     RMB   2               DIRECT ACCESS FILE RECORD LENGTH
0485 097E           DR0TRK     RMB   4               CURRENT TRACK NUMBER, DRIVES 0,1,2,3
0486 0982           NMIFLG     RMB   1               NMI FLAG: 0=DON'T VECTOR <>0=YECTOR OUT
0487 0983           DNMIVC     RMB   2               NMI VECTOR: WHERE TO JUMP FOLLOWING AN NMI
0488                *                                INTERRUPT IF THE NMI FLAG IS SET
0489 0985           RDYTMR     RMB   1               MOTOR TURN OFF TIMER
0490 0986           DRGRAM     RMB   1               RAM IMAGE OF DSKREG ($FF40)
0491 0987           DVERFL     RMB   1               VERIFY FLAG: 0=OFF, $FF=ON
0492 0988           ATTCTR     RMB   1               READ/WRITE ATTEMPT COUNTER: NUMBER OF TIMES THE
0493                *                                DISK WILL ATTEMPT TO RETRIEVE OR WRITE DATA
0494                *                                BEFORE IT GIVES UP AND ISSUES AN ERROR.
0495
0496 0989           DFLBUF     RMB   SECLEN          INITIALIZED TO SECLEN BY DISKBAS
0497
0498                *RANDOM FILE RESERVED AREA
0499
0500                *FILE CONTROL BLOCKS AND BUFFERS
0501
0502                *GRAPHIC PAGE RESERVED AREA
0503
0504                *BASIC PROGRAM
0505
0506                *VARIABLE STORAGE AREA
0507
0508                *ARRAY STORAGE AREA
0509
0510
0511                * FREE MEMORY
0512
0513
0514                *STACK
0515
0516                *STRING SPACE
0517
0518                *USER PROGRAM RESERVED AREA
```

```
0519
0520                *END OF RAM
0521
0522 8000           ORG       $8000
0523
0524 8000                     RMB   $2000           EXTENDED BASIC ROM
0525 A000                     RMB   $2000           COLOR BASIC ROM
0526 C000           ROMPAK    EQU   *
0527 C000           DOSBAS    RMB   $2000           DISK BASIC ROM/ENHANCED BASIC INIT CODE
0528 E000                     RMB   $1F00           ENHANCED BASIC
0529
0530                * START OF ADDITIONAL VARIABLES USED BY SUPER EXTENDED BASIC
0531 FE00           H.CRSLOC  RMB   2              CURRENT LOCATION OF CURSOR
0532 FE02           H.CURSX   RMB   1              X POSITION OF CURSOR
0533 FE03           H.CURSY   RMB   1              Y POSITION OF CURSOR
0534 FE04           H.COLUMN  RMB   1              COLUMNS ON HI-RES ALPHA SCREEN
0535 FE05           H.ROW     RMB   1              ROWS ON HI-RES ALPHA SCREEN
0536 FE06           H.DISPEN  RMB   2              END OF HI-RES DISPLAY SCREEN
0537 FE08           H.CRSATT  RMB   1              CURRENT CURSOR'S ATTRIBUTES
0538 FE09                     RMB   1              UNUSED
0539 FE0A           H.FCOLOR  RMB   1              FOREGROUND COLOR
0540 FE0B           H.BCOLOR  RMB   1              BACKGROUND COLOR
0541 FE0C           H.ONBRK   RMB   2              ON BRK GOTO LINE NUMBER
0542 FE0E           H.ONERR   RMB   2              ON ERR GOTO LINE NUMBER
0543 FE10           H.ERROR   RMB   1              ERROR NUMBER ENCOUNTERED OR $FF (NO ERROR)
0544 FE11           H.ONERRS  RMB   2              ON ERR SOURCE LINE NUMBER
0545 FE13           H.ERLINE  RMB   2              LINE NUMBER WHERE ERROR OCCURRED
0546 FE15           H.ONBRKS  RMB   2              ON BRK SOURCE LINE NUMBER
0547 FE17           H.ERRBRK  RMB   1              STILL UNKNOWN, HAS TO DO WITH ERR, BRK
0548 FE18           H.PCOUNT  RMB   1              PRINT COUNT, CHARACTERS TO BE HPRINTED
0549 FE19           H.PBUF    RMB   80             PRINT BUFFER, HPRINT CHARS. STORED HERE
0550 FE69                     RMB   132            UNUSED
0551 FEED           INT.FLAG  RMB   1              INTERRUPT VALID FLAG. 0=NOT VALID, $55=VALID
0552                * TABLE OF JUMP VECTORS TO INTERRUPT SERVICING ROUTINES
0553 FEEE           INT.JUMP
0554 FEEE           INT.SWI3  RMB   3
0555 FEF1           INT.SWI2  RMB   3
0556 FEF4           INT.FIRQ  RMB   3
0557 FEF7           INT.IRQ   RMB   3
0558 FEFA           INT.SWI   RMB   3
0559 FEFD           INT.NMI   RMB   3
0560
0561                * I/O AREA
0562
0563 FF00           PIA0      EQU   *              PERIPHERAL INTERFACE ADAPTER ONE
0564
0565 FF00           BIT0      KEYBOARD ROW 1 AND RIGHT JOYSTICK SWITCH 1
0566                BIT1      KEYBOARD ROW 2 AND LEFT JOYSTICK SWITCH 1
0567                BIT2      KEYBOARD ROW 3 AND RIGHT JOYSTICK SWITCH 2
0568                BIT3      KEYBOARD ROW 4 AND LEFT JOYSTICK SWITCH 2
0569                BIT4      KEYBOARD ROW 5
0570                BIT5      KEYBOARD ROW 6
0571                BIT6      KEYBOARD ROW 7
0572                BIT7      JOTSTICK COMPARISON IINPUT
0573
0574 FF01           BIT0      CONTROL OF HSYNC (63.5ps)   0 = IRQ* TO CPU DISABLED
0575                          INTERRUPT                   1 = IRQ* TO CPU ENABLED
0576                BIT1      CONTROL OF INTERRUPT        0 = FLAG SET ON FALLING EDGE OF HS
0577                          POLARITY                    1 = FLAG SET ON RISING EDGE OF HS
0578                BIT2      NORMALLY 1                  0 = CHANGES FF00 TO DATA DIRECTION
0579                BIT3      SEL 1                       LSB OF TWO ANALOG MUX SELECT LINES
0580                BIT4      ALWAYS 1
0581                BIT5      ALWAYS 1
0582                BIT6      NOT USED
0583                BIT7      HORIZONTAL SYNC INTERRUPT FLAG
0584
0585 FF02           BIT0      KEYBOARD COLUMN 1
0586                BIT1      KEYBOARD COLUMN 2
0587                BIT2      KEYBOARD COLUMN 3
0588                BIT3      KEYBOARD COLUMN 4
0589                BIT4      KEYBOARD COLUMN 5
0590                BIT5      KEYBOARD COLUMN 6
0591                BIT6      KEYBOARD COLUMN 7 / RAM SIZE OUTPUT
0592                BIT7      KEYBOARD COLUMN 8
```

```
0593
0594 FF03          BIT0      CONTROL OF VSYNC (16.667ms)  0 = IRQ* TO CPU DISABLED
0595                         INTERRUPT                    1 = IRQ* TO CPU ENABLED
0596               BIT1      CONTROL OF INTERRUPT         0 = FLAG SET ON FALLING EDGE OF FS
0597                         POLARITY                     1 = FLAG SET ON RISING EDGE OF FS
0598               BIT2      NORMALLY 1                   0 = CHANGES FF02 TO DATA DIRECTION
0599               BIT3      SEL 2                        MSB OF TWO ANALOG MUX SELECT LINES
0600               BIT4      ALWAYS 1
0601               BIT5      ALWAYS 1
0602               BIT6      NOT USED
0603               BIT7      FIELD SYNC INTERRUPT FLAG
0604
0605 FF04                    RMB   28              PIA0 IMAGES
0606 FF20          DA
0607 FF20          PIA1      EQU   *               PERIPHERAL INTERFACE ADAPTER TWO
0608
0609 FF20          BIT0      CASSETTE DATA INPUT
0610               BIT1      RS-232C DATA OUTPUT
0611               BIT2      6 BIT D/A LSB
0612               BIT3      6 BIT D/A
0613               BIT4      6 BIT D/A
0614               BIT5      6 BIT D/A
0615               BIT6      6 BIT D/A
0616               BIT7      6 BIT D/A MSB
0617
0618 FF21          BIT0      CONTROL OF CD                0 = FIRQ* TO CPU DISABLED
0619                         (RS-232C STATUS)             1 = FIRQ* TO CPU ENABLED
0620               BIT1      CONTROL OF INTERRUPT         0 = FLAG SET ON FALLING EDGE OF CD
0621                         POLARITY                     1 = FLAG SET ON RISING EDGE OF CD
0622               BIT2      NORMALLY 1                   0 = CHANGES FF20 TO DATA DIRECTION
0623               BIT3      CASSETTE MOTOR CONTROL       0 = OFF       1 = ON
0624               BIT4      ALWAYS 1
0625               BIT5      ALWAYS 1
0626               BIT6      NOT USED
0627               BIT7      CD INTERRUPT FLAG
0628
0629 FF22          BIT0      RS-232C DATA INPUT
0630               BIT1      SINGLE BIT SOUND OUTPUT
0631               BIT2      RAM SIZE INPUT
0632               BIT3      RGB MONITOR SENSING INPUT    CSS
0633               BIT4      VDG CONTROL OUTPUT           GM0 & UPPER/LOWER CASE*
0634               BIT5      VDG CONTROL OUTPUT           GM1 & INVERT
0635               BIT6      VDG CONTROL OUTPUT           GM2
0636               BIT7      VDG CONTROL OUTPUT           A*/G
0637
0638 FF23          BIT0      CONTROL OF CARTRIDGE         0 = FIRQ* TO CPU DISABLED
0639                         INTERRUPT                    1 = FIRQ* TO CPU ENABLED
0640               BIT1      CONTROL OF INTERRUPT         0 = FLAG SET ON FALLING EDGE OF CART*
0641                         POLARITY                     1 = FLAG SET ON RISING EDGE OF CART*
0642               BIT2      NORMALLY 1                   0 = CHANGES FF22 TO DATA DIRECTION
0643               BIT3      SOUND ENABLE
0644               BIT4      ALWAYS 1
0645               BIT5      ALWAYS 1
0646               BIT6      NOT USED
0647               BIT7      CARTRIDGE INTERRUPT FLAG
0648
0649 FF24                    RMB   28              PIA1 IMAGES
0650 FF40          PIA2
0651 FF40          DSKREG    RMB   1               DISK CONTROL REGISTER
0652
0653 FF40          BIT0      DRIVE SELECT 0
0654               BIT1      DRIVE SELECT 1
0655               BIT2      DRIVE SELECT 2
0656               BIT3      DRIVE MOTOR ENABLE           0 = MOTORS OFF       1 = MOTORS ON
0657               BIT4      WRITE PRECOMPENSATION        0 = NO PRECOMP       1 = PRECOMP
0658               BIT5      DENSITY FLAG                 0 = SINGLE           1 = DOUBLE
0659               BIT6      DRIVE SELECT 3
0660               BIT7      HALT FLAG                    0 = DISABLED         1 = ENABLED
0661
0662 FF41                    RMB   7               DSKREG IMAGES
0663
0664               * FLOPPY DISK CONTROLLER INTERNAL REGISTERS
0665 FF48          FDCREG    RMB   1               STATUS/COMMAND REGISTER
0666
```

```
0667                COMMANDS   TYPE  COMMAND              CODE
0668                           I     RESTORE              $Ø3
0669                           I     SEEK                 $17
0670                           I     STEP                 $23
0671                           I     STEP IN              $43
0672                           I     STEP OUT             $53
0673                           II    READ SECTOR          $8Ø
0674                           II    WRITE SECTOR         $AØ
0675                           III   READ ADDRESS         $CØ
0676                           III   READ TRACK           $E4
0677                           III   WRITE TRACK          $F4
0678                           IV    FORCE INTERRUPT      $DØ
0679
0680                STATUS     BIT   TYPE I         READ ADDRESS/SECTOR/TRACK   WRITE SECTOR/TRACK
0681                           SØ    BUSY           BUSY                        BUSY
0682                           S1    INDEX          DRQ                         DRQ
0683                           S2    TRACK Ø        LOST DATA                   LOST DATA
0684                           S3    CRC ERROR      CRC ERROR (EXCEPT TRACK)    CRC ERROR (EXCEPT TRACK)
0685                           S4    SEEK ERROR     RNF (EXCEPT TRACK)          RNF (EXCEPT TRACK)
0686                           S5    HEAD LOADED    RECORD TYPE (SECTOR ONLY)   WRITE FAULT
0687                           S6    WRITE PROTECT                              WRITE PROTECT
0688                           S7    NOT READY      NOT READY                   NOT READY
0689
0690 FF49                      RMB   1              TRACK REGISTER
0691 FF4A                      RMB   1              SECTOR REGISTER
0692 FF4B                      RMB   1              DATA REGISTER
0693 FF4C                      RMB   4              FDCREG IMAGES
0694
0695 FF5Ø                      RMB   16             UNUSED SPACE
0696 FF6Ø                      RMB   1              X COORDINATE FOR X-PAD
0697 FF61                      RMB   1              Y COORDINATE FOR X-PAD
0698 FF62                      RMB   1              STATUS REGISTER FOR X-PAD
0699 FF63                      RMB   5              UNUSED
0700                * RS-232 PROGRAM PAK
0701 FF68                      RMB   1              READ/WRITE DATA REGISTER
0702 FF69                      RMB   1              STATUS REGISTER
0703 FF6A                      RMB   1              COMMAND REGISTER
0704 FF6B                      RMB   1              CONTROL REGISTER
0705 FF6C                      RMB   4
0706 FF7Ø                      RMB   13
0707 FF7D                      RMB   1              SOUND/SPEECH CARTRIDGE RESET
0708 FF7E                      RMB   1              SOUND/SPEECH CARTRIDGE READ/WRITE
0709 FF7F                      RMB   1              MULTI-PAK PROGRAMMING REGISTER
0710
0711 FF8Ø                      RMB   64             RESERVED FOR FUTURE EXPANSION
0712
0713                * VIDEO CONTROL REGISTERS
0714 FF9Ø          INITØ       RMB   1              INITIALIZATION REGISTER Ø
0715
0716 FF9Ø          BITØ        MCØ                        ROM MAP CONTROL (SEE TABLE BELOW)
0717               BIT1        MC1                        ROM MAP CONTROL (SEE TABLE BELOW)
0718               BIT2        MC2                        STANDARD SCS
0719               BIT3        MC3                        1 = DRAM AT ØxFEXX IS CONSTANT
0720               BIT4        FEN                        1 = CHIP FIRQ OUTPUT ENABLED
0721               BIT5        IEN                        1 = CHIP IRQ OUTPUT ENABLED
0722               BIT6        M/P                        1 = MMU ENABLED
0723               BIT7        COCO                       1 = COCO 1 & 2 COMPATIBLE
0724
0725                           MC1   MCØ            ROM MAPPING
0726                           Ø     x              16K INTERNAL, 16K EXTERNAL
0727                           1     Ø              32K INTERNAL
0728                           1     1              32L EXTERNAL (EXCEPT FOR VECTORS)
0729
0730 FF91          INIT1       RMB   1              INITIALIZATION REGISTER 1
0731
0732 FF91          BITØ        TR                         MMU TASK REGISTER SELECT
0733               BIT1
0734               BIT2
0735               BIT3
0736               BIT4
0737               BIT5        TINS                       TIMER INPUT SELECT: 1=7Øns, Ø=63ns
0738               BIT6
0739               BIT7
0740
```

```
0741
0742 FF92          IRQENR    RMB   1            IRQ INTERRUPT ENABLE REGISTER
0743
0744 FF92          BIT0      EI0                      CARTRIDGE IRQ ENABLED
0745               BIT1      EI1                      KEYBOARD IRQ ENABLED
0746               BIT2      EI2                      SERIAL DATA IRQ ENABLED
0747               BIT3      VBORD                    VERTICAL BORDER IRQ ENABLED
0748               BIT4      HBORD                    HORIZONTAL BORDER IRQ ENABLED
0749               BIT5      TMR                      INTERRUPT FROM TIMER ENABLED
0750               BIT6
0751               BIT7
0752
0753 FF93          FIRQENR   RMB   1            FIRQ INTERRUPT ENABLE REGISTER
0754
0755 FF93          BIT0      EI0                      CARTRIDGE FIRQ ENABLED
0756               BIT1      EI1                      KEYBOARD FIRQ ENABLED
0757               BIT2      EI2                      SERIAL DATA FIRQ ENABLED
0758               BIT3      VBORD                    VERTICAL BORDER FIRQ ENABLED
0759               BIT4      HBORD                    HORIZONTAL BORDER FIRQ ENABLED
0760               BIT5      TMR                      INTERRUPT FROM TIMER ENABLED
0761               BIT6
0762               BIT7
0763
0764 FF94          V.TIMER   RMB   2            TIMER REGISTER
0765 FF96                    RMB   2            RESERVED FOR FUTURE EXPANSION
0766
0767 FF98          VIDEOREG  RMB   1            VIDEO MODE REGISTER
0768
0769 FF98          BIT0      LPR0                     LINES PER ROW (SEE TABLE BELOW)
0770               BIT1      LPR1                     LINES PER ROW (SEE TABLE BELOW)
0771               BIT2      LPR2                     LINES PER ROW (SEE TABLE BELOW)
0772               BIT3      H50                      1 = 50 Hz VERTICAL REFRESH
0773               BIT4      MOCH                     1 = MONOCHROME (ON COMPOSITE)
0774               BIT5      BPI                      1 = BURST PHASE INVERTED
0775               BIT6
0776               BIT7      BP                       0 = ALPHA, 1 = BIT PLANE
0777
0778               LPR2      LPR1  LPR0         LINES PER CHARACTER ROW
0779               0         0     0            1            (GRAPHICS MODES)
0780               0         0     1            2            (COCO 1 & 2 ONLY)
0781               0         1     0            3            (COCO 1 & 2 ONLY)
0782               0         1     1            8
0783               1         0     0            9
0784               1         0     1            (RESERVED)
0785               1         1     0            12
0786               1         1     1            (RESERVED)
0787
0788 FF99          VIDEOREG  RMB   1            VIDEO MODE REGISTER
0789
0790 FF99          BIT0      CRES0                    COLOR RESOLUTION
0791               BIT1      CRES1                    COLOR RESOLUTION
0792               BIT2      HRES0                    HORIZONTAL RESOLUTION
0793               BIT3      HRES1                    HORIZONTAL RESOLUTION
0794               BIT4      HRES2                    HORIZONTAL RESOLUTION
0795               BIT5      LPF0                     LINES PER FIELD (SEE TABLE BELOW)
0796               BIT6      LPF1                     LINES PER FIELD (SEE TABLE BELOW)
0797               BIT7
0798
0799                         LPF1  LPF0         LINES PER FIELD
0800                         0     0            192
0801                         0     1            200
0802                         1     0            RESERVED
0803                         1     1            225
0804
0805               * VIDEO RESOLUTION
0806                         ALPHA: BP = 0, COCO = 0
0807                              MODE          HRES2  HRES1  HRES0  CRES1  CRES0
0808                              32 CHARACTER  0            0            1
0809                              40 CHARACTER  0            1            1
0810                              80 CHARACTER  1            1            1
0811                         GRAPHICS: BP = 1, COCO = 0
0812                              PIXELSxCOLORS HRES2  HRES1  HRES0  CRES1  CRES0
0813                              640x4         1      1      1      0      1
0814                              640x2         1      0      1      0      0
```

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

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

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

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

```
0889                                BIT5            PAØ, THIRD PIXEL
0890                                BIT6            PAØ, SECOND PIXEL
0891                                BIT7            PAØ, FIRST PIXEL
0892                * PALETTE ADDRESSES
0893                                ADDRESS         PA3    PA2    PA1    PAØ
0894                                FFBØ            Ø      Ø      Ø      Ø
0895                                FFB1            Ø      Ø      Ø      1
0896                                FFB2            Ø      Ø      1      Ø
0897                                FFB3            Ø      Ø      1      1
0898                                FFB4            Ø      1      Ø      Ø
0899                                FFB5            Ø      1      Ø      1
0900                                FFB6            Ø      1      1      Ø
0901                                FFB7            Ø      1      1      1
0902                                FFB8            1      Ø      Ø      Ø
0903                                FFB9            1      Ø      Ø      1
0904                                FFBA            1      Ø      1      Ø
0905                                FFBB            1      Ø      1      1
0906                                FFBC            1      1      Ø      Ø
0907                                FFBD            1      1      Ø      1
0908                                FFBE            1      1      1      Ø
0909                                FFBF            1      1      1      1
0910
0911 FF9A           V.BORDER   RMB   1              BORDER REGISTER
0912
0913 FF9A           BITØ       BLUØ                      BLUE LSB
0914                BIT1       GRNØ                      GREEN LSB
0915                BIT2       REDØ                      RED LSB
0916                BIT3       BLU1                      BLUE MSB
0917                BIT4       GRN1                      GREEN MSB
0918                BIT5       RED1                      RED MSB
0919                BIT6
0920                BIT7
0921
0922 FF9B                      RMB   1              RESERVED
0923 FF9C           V.SCROLL   RMB   1              VERTICAL SCROLL REGISTER
0924
0925 FF9C           BITØ       VSCØ
0926                BIT1       VSC1
0927                BIT2       VSC2
0928                BIT3       VSC3
0929                BIT4
0930                BIT5
0931                BIT6
0932                BIT7
0933                * IN COCO MODE, THE VSC'S MUST BE INITIALIZED TO $ØF
0934
0935 FF9D           V.OFSET1   RMB   1              VERTICAL OFFSET 1 REGISTER
0936
0937 FF9D           BITØ       Y11
0938                BIT1       Y12
0939                BIT2       Y13
0940                BIT3       Y14
0941                BIT4       Y15
0942                BIT5       Y16
0943                BIT6       Y17
0944                BIT7       Y18
0945
0946 FF9E           V.OFSETØ   RMB   1              VERTICAL OFFSET Ø REGISTER
0947
0948 FF9E           BITØ       Y3
0949                BIT1       Y4
0950                BIT2       Y5
0951                BIT3       Y6
0952                BIT4       Y7
0953                BIT5       Y8
0954                BIT6       Y9
0955                BIT7       Y1Ø
0956                * IN COCO MODE, Y9-Y15 ARE NOT EFFECTIVE, AND ARE CONTROLLED BY
0957                  SAM BITS FØ-F6. ALSO IN COCO MODE, Y16-Y18 SHOULD BE 1, ALL OTHERS Ø
0958
0959 FF9F           H.OFSETØ   RMB   1              HORIZONTAL OFFSET Ø REGISTER
0960
0961 FF9F           BITØ       XØ                  HORIZONTAL OFFSET ADDRESS
0962                BIT1       X1                  HORIZONTAL OFFSET ADDRESS
```

```
0963                BIT2      X2                  HORIZONTAL OFFSET ADDRESS
0964                BIT3      X3                  HORIZONTAL OFFSET ADDRESS
0965                BIT4      X4                  HORIZONTAL OFFSET ADDRESS
0966                BIT5      X5                  HORIZONTAL OFFSET ADDRESS
0967                BIT6      X6                  HORIZONTAL OFFSET ADDRESS
0968                BIT7      XVEN                HORIZONTAL VIRTUAL ENABLE
0969                * HVEN ENABLES A HORIZONTAL SCREEN WIDTH OF 128 BYTES REGARDLESS OF THE
0970                  HRES BITS AND CRES BITS SELECTED. THIS WILL ALLOW A 'VIRTUAL' SCREEN
0971                  SOMEWHAT LARGER THAN THE DISPLAYED SCREEN.  THE USER CAN MOVE THIS
0972                  'WINDOW' (THE DISPLAYED SCREEN) BY MEANS OF THE HORIZONTAL OFFSET
0973                  BITS.  IN CHARACTER MODE, THE SCREEN WIDTH IS 128 CHARACTERS REGARDLESS
0974                  OF ATTRIBUTE (OR 64, IF DOUBLE-WIDE IS SELECTED).
0975
0976 FFAØ          MMUREG    RMB   16              MEMORY MANAGEMENT UNIT REGISTERS (6 BITS)
0977
0978                * RELATIONSHIP BETWEEN DATA IN TASK REGISTER AND GENERATED ADDRESS
0979                                  BIT             D5     D4     D3     D2     D1     DØ
0980                                  CORRESPONDING
0981                                  MEMORY ADDRESS  A18    A17    A16    A15    A14    A13
0982
0983                * DATA FROM THE MMU IS THEN USED AS THE UPPER 6 ADDRESS LINES (A13-A18)
0984                  FOR MEMORY ACCESS
0985                                  ADDRESS RANGE   TR     A15    A14    A13    MMU LOCATION
0986                                  XØØØØ - X1FFF   Ø      Ø      Ø      Ø      FFAØ
0987                                  X2ØØØ - X3FFF   Ø      Ø      Ø      1      FFA1
0988                                  X4ØØØ - X5FFF   Ø      Ø      1      Ø      FFA2
0989                                  X6ØØØ - X7FFF   Ø      Ø      1      1      FFA3
0990                                  X8ØØØ - X9FFF   Ø      1      Ø      Ø      FFA4
0991                                  XAØØØ - XBFFF   Ø      1      Ø      1      FFA5
0992                                  XCØØØ - XDFFF   Ø      1      1      Ø      FFA6
0993                                  XEØØØ - XFFFF   Ø      1      1      1      FFA7
0994
0995                                  XØØØØ - X1FFF   1      Ø      Ø      Ø      FFA8
0996                                  X2ØØØ - X3FFF   1      Ø      Ø      1      FFA9
0997                                  X4ØØØ - X5FFF   1      Ø      1      Ø      FFAA
0998                                  X6ØØØ - X7FFF   1      Ø      1      1      FFAB
0999                                  X8ØØØ - X9FFF   1      1      Ø      Ø      FFAC
1000                                  XAØØØ - XBFFF   1      1      Ø      1      FFAD
1001                                  XCØØØ - XDFFF   1      1      1      Ø      FFAE
1002                                  XEØØØ - XFFFF   1      1      1      1      FFAF
1003
1004 FFBØ          PALETREG  RMB   16              COLOR PALETTE REGISTERS (6 BITS)
1005
1006                                  DATA BIT        D5     D4     D3     D2     D1     DØ
1007                                  RGB OUTPUT      R1     G1     B1     RØ     GØ     BØ
1008                                  COMP. OUTPUT    I1     IØ     P3     P2     P1     PØ
1009
1010                * FOR COCO COMPATIBILITY, THE FOLLOWING SHOULD BE LOADED ON INITIALIZATION
1011                  (RGB VALUES) FOR PAL VERSION, IGNORE TABLE FOR COMPOSITE
1012                          FFBØ  GREEN           $12
1013                          FFB1  YELLOW          $36
1014                          FFB2  BLUE            $Ø9
1015                          FFB3  RED             $24
1016                          FFB4  BUFF            $3F
1017                          FFB5  CYAN            $1Ø
1018                          FFB6  MAGENTA         $2D
1019                          FFB7  ORANGE          $26
1020                          FFB8  BLACK           $ØØ
1021                          FFB9  GREEN           $12
1022                          FFBA  BLACK           $ØØ
1023                          FFBB  BUFF            $3F
1024                          FFBC  BLACK           $ØØ
1025                          FFBD  GREEN           $12
1026                          FFBE  BLACK           $ØØ
1027                          FFBF  ORANGE          $26
1028
1029 FFCØ          SAMREG    EQU   *               SAM CONTROL REGISTERS
1030
1031 FFCØ          VØCLR     RMB   1               CLEAR COCO GRAPHICS MODE VØ
1032 FFC1          VØSET     RMB   1               SET COCO GRAPHICS MODE VØ
1033 FFC2          V1CLR     RMB   1               CLEAR COCO GRAPHICS MODE V1
1034 FFC3          V1SET     RMB   1               SET COCO GRAPHICS MODE V1
1035 FFC4          V2CLR     RMB   1               CLEAR COCO GRAPHICS MODE V2
1036 FFC5          V2SET     RMB   1               SET COCO GRAPHICS MODE V2
```

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

```
0001   C000                           ORG   $C000
0002
0003   C000 1A 50             LC000   ORCC  #$50                          DISABLE IRQ, FIRQ INTERRUPTS
0004   C002 10 CE 5E FF               LDS   #$5EFF                        INITIALIZE STACK POINTER
0005   C006 86 12                     LDA   #$12                          PALETTE COLOR: COMPOSITE-GREEN, RGB-INDIGO
0006                          * INITIALIZE ALL PALETTE REGISTERS TO GREEN (COMPOSITE)
0007   C008 C6 10                     LDB   #16                           16 PALETTE REGISTERS
0008   C00A 8E FF B0                  LDX   #PALETREG                     POINT X TO THE PALETTE REGISTERS
0009   C00D A7 80             LC00D   STA   ,X+                           SAVE THE COLOR IN THE PALETTE REGISTER
0010   C00F 5A                        DECB                                BUMP COUNTER
0011   C010 26 FB                     BNE   LC00D                         LOOP UNTIL ALL PALETTE REGISTERS DONE
0012
0013   C012 8E FF A0                  LDX   #MMUREG                       POINT X TO THE MMU REGISTERS
0014   C015 31 8D 02 2D               LEAY  MMUIMAGE,PC                   POINT Y TO THE MMU REGISTER IMAGES
0015   C019 C6 10                     LDB   #16                           16 MMU REGISTERS
0016   C01B A6 A0             LC01B   LDA   ,Y+                           GET A BYTE FROM THE IMAGE
0017   C01D A7 80                     STA   ,X+                           SAVE IT IN THE MMU REGISTER
0018   C01F 5A                        DECB                                BUMP COUNTER
0019   C020 26 F9                     BNE   LC01B                         LOOP UNTIL DONE
0020   C022 86 CE                     LDA   #COCO+MMUEN+MC3+MC2+MC1       ENABLE COCO COMPATIBLE MODE; ENABLE MMU
0021   C024 B7 FF 90                  STA   INIT0                         AND TURN ON THE NORMAL SPARE CHIP SELECT
0022                          * MOVE THE INITIALIZATION CODE FROM ROM TO RAM($4000); THIS IS DONE IN
0023                          * PREPARATION FOR MOVING BASIC FROM ROM TO RAM.
0024   C027 30 8D 00 14               LEAX  BEGMOVE,PC                    POINT TO START OF ROM CODE
0025   C02B 10 8E 40 00               LDY   #$4000                        RAM LOAD ADDRESS
0026   C02F EC 81             LC02F   LDD   ,X++                          GRAB TWO BYTES
0027   C031 EE 81                     LDU   ,X++                          GRAB TWO MORE BYTES
0028   C033 ED A1                     STD   ,Y++                          MOVE FIRST SET OF BYTES
0029   C035 EF A1                     STU   ,Y++                          AND THEN THE SECOND
0030   C037 8C C3 6C                  CMPX  #ENDMOVE                      ARE ALL BYTES MOVED?
0031   C03A 25 F3                     BCS   LC02F                         KEEP GOING UNTIL DONE
0032   C03C 7E 40 00                  JMP   L4000                         JUMP INTO THE MOVED CODE
0033
0034                          * THE REST OF THE CODE IS MOVED INTO RAM TO BE EXECUTED
0035   C03F 32 7F             BEGMOVE LEAS  $-01,S                        MAKE A TEMPORARY STORAGE LOCATION ON THE STACK
0036   C041 12                        NOP
0037   C042 12                        NOP
0038   C043 12                        NOP
0039   C044 12                        NOP
0040   C045 12                        NOP                                 SPACE FILLER NOPS - THEY SERVE NO PURPOSE
0041   C046 86 FF                     LDA   #$FF
0042   C048 B7 FF 94                  STA   V.TIMER
0043   C04B B7 FF 95                  STA   V.TIMER+1                     SET THE TIMER TO $FFFF AND START IT COUNTING
0044
0045                          * SET UP THE VIDEO CONTROL REGISTERS
0046   C04E 30 8D 01 DC               LEAX  VIDIMAGE,PC                   POINT X TO THE VIDEO CONTROL REGISTER IMAGE
0047   C052 10 8E FF 98               LDY   #VIDEOREG                     POINT Y TO THE VIDEO CONTROL REGISTERS
0048   C056 A6 80             LC056   LDA   ,X+                           GET A BYTE FROM THE IMAGE
0049   C058 A7 A0                     STA   ,Y+                           SAVE IT IN THE VIDEO REGISTER
0050   C05A 10 8C FF A0               CMPY  #MMUREG                       CHECK FOR THE END OF THE VIDEO MODE REGISTERS
0051   C05E 26 F6                     BNE   LC056                         LOOP UNTIL DONE
0052                          * INITIALIZE PIA1
0053   C060 8E FF 20                  LDX   #PIA1                         POINT X TO PIA 1
0054   C063 CC FF 34                  LDD   #$FF34
0055   C066 6F 01                     CLR   $01,X                         CLEAR CONTROL REGISTER A
0056   C068 6F 03                     CLR   $03,X                         CLEAR CONTROL REGISTER B; ENABLE BOTH DATA DIRECTION REGISTERS
0057   C06A 4A                        DECA                                SET ACCA TO $FE
0058   C06B A7 84                     STA   ,X                            BIT 0 INPUT, ALL OTHERS OUTPUT ON PORT A
0059   C06D 86 F8                     LDA   #$F8
0060   C06F A7 02                     STA   $02,X                         BITS 0-2 INPUT, 3-7 OUTPUT ON PORT B
0061   C071 E7 01                     STB   $01,X                         SET PORT TO PERIPHERAL REGISTER, CA1 DISABLED, CA2 ENABLED AS INPUT
0062   C073 E7 03                     STB   $03,X                         SET PORT TO PERIPHERAL REGISTER, CB1 DISABLED, CB2 ENABLED AS INPUT
0063   C075 6F 02                     CLR   $02,X                         SET THE GRAPHICS MODE TO NORMAL LO-RES COCO ALPHA
0064   C077 86 02                     LDA   #$02
0065   C079 A7 84                     STA   ,X                            SET THE DA OUTPUT TO ZERO AND THE RS232 OUTPUT TO MARKING
0066   C07B 86 FF                     LDA   #$FF
0067                          * INITIALIZE PIA0
0068   C07D 8E FF 00                  LDX   #PIA0                         POINT X TO PIA 0
0069   C080 6F 01                     CLR   $01,X                         CLEAR CONTROL REGISTER A; ENABLE BOTH DATA DIRECTION REGISTERS
0070   C082 6F 03                     CLR   $03,X                         CLEAR CONTROL REGISTER B; ENABLE BOTH DATA DIRECTION REGISTERS
0071   C084 6F 84                     CLR   ,X                            SET PORT A TO ALL INPUTS
0072   C086 A7 02                     STA   $02,X                         SET PORT B TO ALL OUTPUTS
0073   C088 E7 01                     STB   $01,X                         SET PORT TO PERIPHERAL REGISTER, CA1 DISABLED, CA2 ENABLED AS INPUT
0074   C08A E7 03                     STB   $03,X                         SET PORT TO PERIPHERAL REGISTER, CB1 DISABLED, CB2 ENABLED AS INPUT
0075                          * INITIALIZE THE SAM MIRROR REGISTERS IN THE CUSTOM CHIP
0076   C08C C6 0C                     LDB   #12                           RESET 12 SAM IMAGE REGISTERS
0077   C08E CE FF C0                  LDU   #SAM                          POINT U TO THE SAM REGISTERS
0078   C091 A7 C1             LC091   STA   ,U++                          CLEAR THE BIT AND SKIP TO THE NEXT BIT
0079   C093 5A                        DECB                                BUMP COUNTER
0080   C094 26 FB                     BNE   LC091                         LOOP UNTIL ALL BITS CLEARED
0081   C096 B7 FF C9                  STA   SAM+9                         SET THE VIDEO DISPLAY PAGE TO $400
0082   C099 1F 9B                     TFR   B,DP                          SET THE DIRECT PAGE TO PAGE ZERO
0083   C09B 6F 02                     CLR   $02,X                         STROBE ALL KEYBOARD COLUMNS; USELESS INSTRUCTION
0084   C09D A7 5D                     STA   $-03,U                        SAMREG+21 (FFD5); SELECT RAM PAGE 1; USELESS IN THE COCO 3
0085   C09F 8E FF 00                  LDX   #PIA0                         POINT X TO PIA 0; WHY?? IT'S ALREADY POINTED THERE
0086   C0A2 C6 DF                     LDB   #$DF                          COLUMN TWO STROBE
0087   C0A4 E7 02                     STB   $02,X                         STROBE THE COLUMNS
0088   C0A6 A6 84                     LDA   ,X                            READ THE ROWS
0089   C0A8 43                        COMA
0090   C0A9 84 40                     ANDA  #$40                          LOOK FOR ROW 6 ONLY (F1 KEY)
0091   C0AB A7 E4                     STA   ,S                            SAVE IN TEMPORARY STORAGE
0092                          * CHECK FOR THE CONTROL AND ALT KEYS
0093   C0AD 10 8E 00 02               LDY   #2                            CHECK FOR TWO KEYS
0094   C0B1 57                LC0B1   ASRB                                SHIFT THE COLUMN STROBE -- WASTED, SHOULD BE ASR 2,X
0095   C0B2 E7 02                     STB   $02,X                         SAVE THE NEW COLUMN STROBE
0096   C0B4 A6 84                     LDA   ,X                            READ THE KEYBOARD ROWS
```

```
0097   C0B6 43                    COMA
0098   C0B7 84 40                 ANDA  #$40                        KEEP ONLY ROW 6
0099   C0B9 27 07                 BEQ   LC0C2                       BRANCH IF KEY NOT DOWN
0100   C0BB 31 3F                 LEAY  $-01,Y                      LET'S CHECK FOT EH ALT KEY NOW
0101   C0BD 26 F2                 BNE   LC0B1
0102   C0BF 16 01 2E              LBRA  LC1F0                       GO DISPLAY THE HI-RES PICTURE IF CONTROL AND ALT KEYS ARE DOWN
0103   C0C2 86 CA          LC0C2  LDA   #COCO+MMUEN+MC3+MC1         TURN OFF THE NORMAL SCS; THE EXTERNAL DISK CONTROLLER
0104   C0C4 B7 FF 90              STA   INIT0                       MAY NOT BE ACCESSED NOW
0105
0106                       * THE FOLLOWING CODE CHECKS TO DETERMINE IF A JUMP TO WARM START RESET CODE SHOULD BE DONE.
0107                       * THE JUMP TO A WARM START RESET WILL BE DONE IF 1) INT.FLAG CONTAINS A $55 AND,
0108                       * 2) RSTFLG CONTAINS A $55 AND, 3) THE ADDRESS IN RSTVEC POINTS TO A $12 (NOP INSTRUCTION.)
0109                       * IF THE ABOVE CONDITIONS ARE MET, BASIC WILL BE WARM-STARTED.  IF INT.FLAG DOES NOT CONTAIN
0110                       * A $55, BASIC WILL BE COLD-STARTED.  IF INT.FLAG DOES CONTAIN A $55, BUT 2) AND 3) ABOVE
0111                       * ARE NOT MET, BLOCK 6.0 (128K SYSTEM) OR BLOCK 0.0 (512K SYSTEM) WILL BE LOADED INTO CPU
0112                       * BLOCK 0. THIS WILL GIVE THE CPU A NEW DIRECT PAGE AND CHECKS 2) AND 3) ABOVE WILL BE
0113                       * PERFORMED ON THIS NEW DIRECT PAGE TO SEE IF BASIC SHOULD BE WARM-STARTED.
0114
0115   C0C7 B6 FE ED              LDA   INT.FLAG                    GET THE INTERRUPT JUMP TABLE VALIDITY FLAG.
0116   C0CA 81 55                 CMPA  #$55                        CHECK FOR VALID INTERRUPT JUMP TABLE FLAG
0117   C0CC 26 28                 BNE   LC0F6                       INTERRUPT JUMP TABLE IS NOT VALID' COPY ROM TO RAM
0118   C0CE 96 71                 LDA   RSTFLG                      GET THE SYSTEM RESET FLAG
0119   C0D0 81 55                 CMPA  #$55                        CHECK FOR THE WARM START FLAG
0120   C0D2 26 0A                 BNE   NOWARM                      BRANCH IF NO WARM START
0121   C0D4 9E 72                 LDX   RSTVEC                      GET THE SYSTEM RESET VECTOR
0122   C0D6 A6 84                 LDA   ,X                          GET THE FIRST BYTE POINTED TO BY THE RESET VECTOR
0123   C0D8 81 12                 CMPA  #$12                        IS IT A NOP?
0124   C0DA 10 27 00 AE           LBEQ  LC18C                       DON'T COPY ROM TO RAM, ETC.
0125   C0DE 7F FF A0       NOWARM CLR   MMUREG                      PUT BLOCK 6.0 (128K RAM) OR BLOCK 0.0 (512K RAM) INTO CPU BLOCK 0
0126   C0E1 96 71                 LDA   RSTFLG
0127   C0E3 81 55                 CMPA  #$55                        CHECK FOT THE WARM START FLAG
0128   C0E5 26 0A                 BNE   LC0F1                       BRANCH IF NO WARM START
0129   C0E7 9E 72                 LDX   RSTVEC                      POINT X TO THE WARM START CODE
0130   C0E9 A6 84                 LDA   ,X                          GET THE FIRST BYTE OF THE WARM START CODE
0131   C0EB 81 12                 CMPA  #$12                        IS IT A NOP?
0132   C0ED 10 27 00 9B           LBEQ  LC18C                       DON'T COPY ROM TO RAM IF IT IS.
0133   C0F1 86 38          LC0F1  LDA   #BLOCK7.0                   GET BACK BLOCK 7.0
0134   C0F3 B7 FF A0              STA   MMUREG                      PUT IT BACK INTO CPU BLOCK 0
0135   C0F6 8E C0 00       LC0F6  LDX   #DOSBAS                     POINT TO THE END OF THE COLOR BASIC ROM
0136   C0F9 10 8E 80 00           LDY   #EXBAS                      POINT TO START OF EXTENDED BASIC
0137   C0FD 17 00 AA              LBSR  LC1AA                       MOVE COLOR AND EXTENDED BASIC ROM TO RAM
0138
0139                       * PATCH COLOR AND EXTENDED BASIC
0140   C100 31 8D 01 52           LEAY  PATCHTAB,PC                 POINT Y TO THE PATCH TABLE
0141   C104 A6 A0                 LDA   ,Y+                         GET THE NUMBER OF PATCHES TO BE MADE
0142   C106 34 02          LC106  PSHS  A                           SAVE THE PATCH COUNTER
0143   C108 AE A1                 LDX   ,Y++                        GET THE ADDRESS WHERE THE PATCH IS TO BE PLACED
0144   C10A E6 A0                 LDB   ,Y+                         GET THE NUMBER OB BYTES IN THE PATCH
0145   C10C A6 A0          LC10C  LDA   ,Y+                         GET A BYTE
0146   C10E A7 80                 STA   ,X+                         PATCH THE CODE IN RAM
0147   C110 5A                    DECB                              BUMP THE COUNTER
0148   C111 26 F9                 BNE   LC10C                       LOOP UNTIL DONE
0149   C113 35 02                 PULS  A                           RESTORE THE PATCH COUNTER
0150   C115 4A                    DECA
0151   C116 26 EE                 BNE   LC106                       LOOP UNTIL ALL PATCHES DONE
0152   C118 7F FF DE              CLR   SAM+30                      ENABLE THE ROM MODE
0153   C11B 86 C8                 LDA   #COCO+MMUEN+MC3             ENABLE 16K INTERNAL, 16K EXTERNAL ROM
0154   C11D B7 FF 90              STA   INIT0
0155   C120 FC C0 00              LDD   DOSBAS                      GET THE FIRST TWO BYTES OF AN EXTERNAL ROM, IF ANY
0156                       * CHECK FOR A 'DK' AT $C000 (DISK BASIC) - THIS SHOULD BE CMPD
0157   C123 81 44                 CMPA  #'D'
0158   C125 26 10                 BNE   LC137
0159   C127 C1 4B                 CMPB  #'K'
0160   C129 26 0C                 BNE   LC137
0161                       * COPY THE DISK BASIC ROM INTO RAM
0162   C12B 8E E0 00              LDX   #SUPERVAR                   POINT TO THE END OF THE DISK BASIC ROM
0163   C12E 10 8E C0 00           LDY   #DOSBAS                     POINT TO THE START OF THE DISK BASIC ROM
0164   C132 8D 76                 BSR   LC1AA                       COPY ROM INTO RAM
0165   C134 17 01 EB              LBSR  LC322                       PATCH DISK BASIC
0166   C137 7F FF DE       LC137  CLR   SAM+30                      ENABLE ROM MODE
0167   C13A 86 CA                 LDA   #COCO+MMUEN+MC3+MC1
0168   C13C B7 FF 90              STA   INIT0                       ENABLE 32K INTERNAL ROM
0169
0170                       * COPY SUPER EXTENDED BASIC FROM ROM TO RAM
0171   C13F 8E FE 00              LDX   #H.CRSLOC                   POINT TO THE END OF ENHANCED BASIC ROM
0172   C142 10 8E E0 00           LDY   #SUPERVAR                   POINT TO THE START OF ENHANCED BASIC ROM
0173   C146 8D 62                 BSR   LC1AA                       COPY ROM TO RAM
0174   C148 17 00 93              LBSR  LC1DE                       PATCH THE ENHANCEMENTS (MOVE THE AUTHORS' DECODED NAMES)
0175   C14B 31 8D 02 0A           LEAY  INTIMAGE,PC                 POINT X TO THE INTERRUPT JUMP VECTOR IMAGES
0176   C14F 8E FE ED              LDX   #INT.FLAG                   DESTINATION FOR INTERRUPT VECTORS
0177   C152 C6 13                 LDB   #19                         6 INTERRUPT JUMP ADDRESSES * 3 BYTES/JUMP ADDRESS + VALIDITY FLAG
0178   C154 17 00 7F              LBSR  MOVE.XY                     COPY THE INTERRUPT JUMP VECTORS
0179   C157 7F FF DF              CLR   SAM+31                      ENABLE THE RAM MODE
0180   C15A 6D E4                 TST   ,S                          WAS THE F1 KEY DEPRESSED?
0181   C15C 27 22                 BEQ   LC180                       NO
0182   C15E 8E E0 32              LDX   #IM.TEXT                    TEXT MODE VIDEO CONTROL REGISTER IMAGES IN SUPER EXTENDED BASIC
0183   C161 C6 03                 LDB   #$03                        THREE SETS OF IMAGES
0184   C163 30 01                 LEAX  $01,X                       SKIP PAST THE $FF90 TEXT MODE IMAGE
0185   C165 A6 84          LC165  LDA   ,X                          GET THE INIT0 IMAGE
0186   C167 8A 20                 ORA   #$20                        FORCE THE ALTERNATE COLOR SET
0187   C169 A7 84                 STA   ,X                          RE-SAVE THE INIT0 IMAGE
0188   C16B 30 09                 LEAX  $09,X                       SKIP TO NEXT SET OF IMAGES
0189   C16D 5A                    DECB                              BUMP COUNTER
0190   C16E 26 F5                 BNE   LC165                       LOOP UNTIL DONE
0191   C170 C6 02                 LDB   #$02                        TWO SETS OF GRAPHICS MODE IMAGES
0192   C172 8E E0 70              LDX   #IM.GRAPH                   GRAPHICS MODE VIDEO CONTROL REGISTER IMAGES IN SUPER EXTENDED BASIC
```

```
0193   C175 A6 84            LC175    LDA   ,X                          GET THE INITØ IMAGE
0194   C177 8A 2Ø                     ORA   #$2Ø                        FORCE THE ALTERNATE COLOR SET
0195   C179 A7 84                     STA   ,X                          RE-SAVE THE INITØ IMAGE
0196   C17B 3Ø Ø9                     LEAX  $Ø9,X                       SKIP TO NEXT SET OF IMAGES
0197   C17D 5A                        DECB                              BUMP COUNTER
0198   C17E 26 F5                     BNE   LC175                       LOOP UNTIL DONE
0199                          * CLEAR THE LO-RES VIDEO SCREEN
0200   C18Ø 8E Ø4 ØØ         LC18Ø    LDX   #VIDRAM                     POINT X TO THE START OF THE VIDEO DISPLAY
0201   C183 86 6Ø                     LDA   #$6Ø                        GREEN SPACE
0202   C185 A7 8Ø            LC185    STA   ,X+                         PUT A GREEN SPACE ON THE LO-RES SCREEN
0203   C187 8C Ø6 ØØ                 CMPX  #VIDRAM+512                 AT THE END OF THE DISPLAY?
0204   C18A 25 F9                     BCS   LC185                       NO
0205   C18C 86 CE            LC18C    LDA   #COCO+MMUEN+MC3+MC2+MC1
0206   C18E B7 FF 9Ø                  STA   INITØ                       ENABLE THE NORMAL SPARE CHIP SELECT (EXTERNAL $FF4Ø)
0207   C191 6D E4                     TST   ,S                          WAS THE F1 KEY DEPRESSED?
0208   C193 27 Ø5                     BEQ   LC19A                       NO
0209   C195 86 2Ø                     LDA   #$2Ø                        ALTERNATE COLOR SET FLAG
0210   C197 B7 FF 98                  STA   VIDEOREG                    FORCE THE ALTERNATE COLOR SET
0211   C19A 8E FF BØ          LC19A    LDX   #PALETREG                   POINT X TO THE PALETTE REGISTERS
0212   C19D 31 8D ØØ 95               LEAY  PALIMAGE,PC                 POINT Y TO THE PALETTE REGISTER IMAGES
0213   C1A1 C6 1Ø                     LDB   #16                         16 PALETTE REGISTERS
0214   C1A3 8D 31                     BSR   MOVE.XY                     FILL THE PALETTE REGISTERS FROM THEIR IMAGE
0215   C1A5 32 61                     LEAS  $Ø1,S                       REMOVE THE TEMPORARY STORAGE BYTE
0216   C1A7 7E A Ø 27                 JMP   RESVEC                      JUMP TO THE COCO 2 RESET ENTRY POINT
0217
0218                          * COPY DATA POINTED TO BY (Y) FROM ROM TO RAM UNTIL THE ADDRESS IN
0219                          * (X) IS REACHED; PSHING AND PULING FROM U OR S WOULD BE MUCH MORE EFFICIENT
0220   C1AA BF 5F Ø2          LC1AA    STX   L5FØ2                      TEMPORARILY SAVE THE END OF COPY ADDRESS
0221   C1AD 1Ø FF 5F ØØ               STS   L5FØØ                      AND THE STACK POINTER
0222   C1B1 7F FF DE          LC1B1    CLR   SAM+3Ø                      ENABLE THE ROM
0223   C1B4 EC A4                     LDD   ,Y
0224   C1B6 AE 22                     LDX   $Ø2,Y
0225   C1B8 EE 24                     LDU   $Ø4,Y
0226   C1BA 1Ø EE 26                  LDS   $Ø6,Y
0227   C1BD 7F FF DF                  CLR   SAM+31                      DISABLE THE ROM
0228   C1CØ ED A4                     STD   ,Y                          NOW SAVE THE DATA FROM THE CPU REGISTERS INTO ROM
0229   C1C2 AF 22                     STX   $Ø2,Y
0230   C1C4 EF 24                     STU   $Ø4,Y
0231   C1C6 1Ø EF 26                  STS   $Ø6,Y
0232   C1C9 31 28                     LEAY  $Ø8,Y                       MOVE THE COPY POINTER UP 8 BYTES
0233   C1CB 1Ø BC 5F Ø2               CMPY  L5FØ2                       CHECK FOR END OF THE COPY RANGE
0234   C1CF 25 EØ                     BCS   LC1B1
0235   C1D1 1Ø FE 5F ØØ               LDS   L5FØØ                      RESTORE THE STACK
0236   C1D5 39                        RTS
0237
0238                          * MOVE ACCB BYTES FROM (Y) TO (X)
0239   C1D6 A6 AØ             MOVE.XY  LDA   ,Y+
0240   C1D8 A7 8Ø                     STA   ,X+
0241   C1DA 5A                        DECB
0242   C1DB 26 F9                     BNE   MOVE.XY
0243   C1DD 39                        RTS
0244
0245                          * DECODE AND COPY THE AUTHOR'S NAMES INTO RAM
0246   C1DE 8E F7 1B          LC1DE    LDX   #AUTHORMS                   POINT X TO THE DESTINATION FOR THE AUTHORS' NAMES
0247   C1E1 31 8D Ø1 28               LEAY  LC3ØD,PC                    POINT Y TO THE CODED NAMES OF THE AUTHORS
0248   C1E5 C6 15                     LDB   #21                         21 BYTES IN THE AUTHORS' NAMES
0249   C1E7 A6 AØ             LC1E7    LDA   ,Y+                         GET A CODED BYTE OF THE AUTHORS' NAMES
0250   C1E9 43                        COMA                              DECODE THE BYTE
0251   C1EA A7 8Ø                     STA   ,X+                         SAVE THE UNCODED BYTE
0252   C1EC 5A                        DECB                              BUMP COUNTER DOWN ONE.
0253   C1ED 26 F8                     BNE   LC1E7                       LOOP UNTIL ALL BYTES DECODED
0254   C1EF 39                        RTS
0255
0256                          * THIS IS THE CODE WHICH DISPLAYS THE HIGH RESOLUTION PICTURE OF THE
0257                          * AUTHORS OF SUPER EXTENDED BASIC
0258   C1FØ 4F                LC1FØ    CLRA
0259   C1F1 B7 FE ED                  STA   INT.FLAG                    SET THE INTERRUPT FLAG TO NOT VALID (NOT INITIALIZED)
0260   C1F4 97 71                     STA   RSTFLG                      FORCE THE ROMS TO BE COPIED INTO RAM
0261   C1F6 B7 FF DE                  STA   SAM+3Ø                      ENABLE THE ROMS
0262   C1F9 C6 Ø9                     LDB   #$Ø9
0263   C1FB F7 FF BA                  STB   PALETREG+1Ø
0264   C1FE C6 3F                     LDB   #63                         WHITE (COMPOSITE AND RGB)
0265   C2ØØ F7 FF BB                  STB   PALETREG+11
0266   C2Ø3 8E C4 Ø5                  LDX   #AUTHPIC                    POINT X TO THE AUTHORS' PICTURE DATA
0267   C2Ø6 1Ø 8E ØE ØØ               LDY   #$ØE ØØ                     DESTINATION OF THE AUTHORS' PICTURE DATA
0268   C2ØA EC 81             LC2ØA    LDD   ,X++
0269   C2ØC EE 81                     LDU   ,X++                        GET FOUR BYTES OF PICTURE DATA
0270   C2ØE ED A1                     STD   ,Y++
0271   C21Ø EF A1                     STU   ,Y++                        PUT THE DATA ON THE HI-RES SCREEN
0272   C212 8C DC Ø5                  CMPX  #LDC05                      AT THE END OF THE PICTURE DATA?
0273   C215 25 F3                     BCS   LC2ØA                       NO
0274   C217 86 F9                     LDA   #$F9                        256x192 GREEN/BUFF COCO 2 HI-RES GRAPHICS MODE
0275   C219 B7 FF 22                  STA   PIA1+2                      PROGRAM THE GRAPHICS MODE INTO THE PIA AND THE GIME CHIP
0276   C21C 4F                        CLRA
0277   C21D 8E FF CØ                  LDX   #SAM                        POINT X TO THE SAM REGISTERS
0278   C22Ø A7 84                     STA   ,X
0279   C222 A7 Ø3                     STA   $Ø3,X
0280   C224 A7 Ø5                     STA   $Ø5,X                       PROGRAM THE SAM REGISTERS FOR HI-RES MODE
0281   C226 A7 Ø7                     STA   $Ø7,X
0282   C228 A7 Ø9                     STA   $Ø9,X
0283   C22A A7 ØB             LC22A    STA   $ØB,X                       SET THE VIDEO DISPLAY PAGE TO $EØØ
0284   C22C 2Ø FE             WAITLOOP BRA   WAITLOOP                    ENDLESS WAIT LOOP
0285
0286                          * IMAGES OF THE VIDEO CONTROL REGISTERS (FF98-FF9F)
0287   C22E ØØ ØØ ØØ ØØ ØF EØ VIDIMAGE FCB   $ØØ,$ØØ,$ØØ,$ØØ,$ØF,$EØ
0288   C234 ØØ ØØ                     FCB   $ØØ,$ØØ
```

```
0289
0290                            * IMAGES OF THE PALETTE REGISTERS (FFB0-FFBF)
0291   C236 12 24 0B 07 3F 1F PALIMAGE FCB   18,36,11,7,63,31
0292   C23C 09 26 00 12 00 3F          FCB   9,38,0,18,0,63
0293   C242 00 12 00 26                FCB   0,18,0,38
0294
0295                            * IMAGES OF THE MMU REGISTERS (FFA0-FFAF)
0296
0297                            * TASK REGISTER 0
0298   C246 38 39 34            MMUIMAGE FCB   BLOCK7.0,BLOCK7.1,BLOCK6.4
0299   C249 3B 3C 3D                    BLOCK7.3,BLOCK7.4,BLOCK7.5
0300   C24C 3E 3F                       BLOCK7.6,BLOCK7.7
0301
0302                            * TASK REGISTER 1
0303   C24E 38 30 31            LC24E    FCB   BLOCK 7.0,BLOCK6.0,BLOCK6.1
0304   C251 32 33 3D                    BLOCK6.2,BLOCK6.3,BLOCK7.5
0305   C254 35 3F                       BLOCK6.5,BLOCK7.7
0306
0307                            * TABLE OF PATCHES TO BE MADE TO COLOR AND EXTENDED BASIC. THE FIRST BYTE
0308                            * IS THE TOTAL NUMBER OF PATCHES TO BE MADE FOLLOWED BY THE CODE FOR ALL OF
0309                            * THE PATCHES.  THE INDIVIDUAL PATCHES HAVE A THREE BYTE HEADER CONSISTING OF THE
0310                            * ADDRESS WHERE THE PATCH IS TO GO AND THE NUMBER OF BYTES IN THE PATCH.
0311
0312   C256 1B                  PATCHTAB FCB   27                               NUMBER OF PATCHES
0313
0314                            * PATCH 1 - ENABLE EXTENDED BASIC WARM START CODE
0315   C257 80 C0               LC257    FDB   PATCH1                          $80C0
0316   C259 01                  LC259    FCB   $01
0317   C25A 12                  LC25A    NOP
0318
0319                            * PATCH 2 - CRUNCH A TOKEN
0320   C25B B8 D4               LC25B    FDB   PATCH2                          $B8D4
0321   C25D 03                  LC25D    FCB   $03
0322   C25E 7E E1 38            LC25E    JMP   ALINK2                          $E138
0323
0324                            * PATCH 3 - UNCRUNCH A TOKEN
0325   C261 B7 F3               LC261    FDB   PATCH3                          $B7F3
0326   C263 03                  LC263    FCB   $03
0327   C264 7E E1 72            LC264    JMP   ALINK3                          $E172
0328
0329                            * PATCH 4 - EXTENDED BASIC'S COMMAND INTERPRETATION LOOP
0330   C267 81 50               LC267    FDB   PATCH4                          $8150
0331   C269 04                  LC269    FCB   $04
0332   C26A 7E E1 92            LC26A    JMP   ALINK4                          $E192
0333   C26D 12                           NOP
0334
0335                            * PATCH 5 - EXTENDED BASIC'S SECONDARY COMMAND HANDLER
0336   C26E 81 6C               LC26E    FDB   PATCH5                          $816C
0337   C270 04                  LC270    FCB   $04
0338   C271 7E E1 A6            LC271    JMP   ALINK5                          $E1A6
0339   C274 12                           NOP
0340
0341                            ** PATCHES 6 - 11 MODIFY THE WAY A '&H' VARIABLE IS PROCESSED
0342                            * PATCH 6
0343   C275 88 34               LC275    FDB   PATCH6                          $8834
0344   C277 12                  LC277    FCB   $12
0345   C278 7E E3 F8            LC278    JMP   ALINK6A                         $E3F8
0346   C27B 0F 51                        CLR   FPA0+1
0347   C27D 0F 52                        CLR   FPA0+2
0348   C27F 0F 53                        CLR   FPA0+3
0349   C281 20 B0                        BRA   ((PATCH7+4)-(PATCH6+9))
0350   C283 0F 50                        CLR   FPA0
0351   C285 20 CF                        BRA   (PATCH6A-(PATCH6+13))
0352   C287 7E E4 0C                     JMP   ALINK6B
0353
0354                            * PATCH 7
0355   C28A 87 EB               LC28A    FDB   PATCH7                          $87EB
0356   C28C 07                  LC28C    FCB   $07
0357   C28D 20 4A               LC28D    BRA   ((PATCH6+3)-PATCH7)
0358   C28F 12                           NOP
0359   C290 39                           RTS
0360   C291 8E 00 51                     LDX   #FPA0+1
0361
0362                            * PATCH 8
0363   C294 88 0C               LC294    FDB   PATCH8                          $880C
0364   C296 02                  LC296    FCB   $02
0365   C297 20 35               LC297    BRA   ((PATCH6+15)-PATCH8)
0366
0367                            * PATCH 9
0368   C299 88 26               LC299    FDB   PATCH9                          $8826
0369   C29B 02                  LC29B    FCB   $02
0370   C29C 25 17               LC29C    BCS   ((PATCH6+11)-PATCH9)
0371
0372                            * PATCH 10
0373   C29E 87 E7               LC29E    FDB   PATCH10                         $87E7
0374   C2A0 02                  LC2A0    FCB   $02
0375   C2A1 26 05               LC2A1    BNE   ((PATCH7+3)-PATCH10)
0376
0377                            * PATCH 11 - NEEDED BECAUSE PATCH 5 REMOVED AN RTS WHICH THIS ROUTINED USED
0378   C2A3 88 6A               LC2A3    FDB   PATCH11                         $886A
0379   C2A5 02                  LC2A5    FCB   $02
0380   C2A6 26 82               LC2A6    BNE   ((PATCH7+3)-PATCH11)
0381
0382                            * PATCH 12 - EX BASIC'S COPYRIGHT MESSAGE
0383   C2A8 80 B2               LC2A8    FDB   PATCH12                         $80B2
0384   C2AA 03                  LC2AA    FCB   $03
```

```
0385   C2AB 7E E2 88          LC2AB    JMP   ALINK12                          $E288
0386
0387                          * PATCH 13 - REMOVE ONE CR FROM ONE OF EX BAS COPYRIGHT MESSAGE
0388   C2AE 81 3A             LC2AE    FDB   PATCH13                          $813A
0389   C2B0 01                LC2B0    FCB   $01
0390   C2B1 00                LC2B1    FCB   $00
0391
0392                          * PATCH 14 - ADD ONTO END OF EX BAS GRAPHICS INITIALIZATION ROUTINE
0393   C2B2 97 03             LC2B2    FDB   PATCH14                          $9703
0394   C2B4 03                LC2B4    FCB   $03
0395   C2B5 7E E3 89          LC2B5    JMP   ALINK14                          $E389
0396
0397                          * PATCH 15 - BREAK CHECK
0398   C2B8 AD F0             LC2B8    FDB   PATCH15                          $ADF0
0399   C2BA 04                LC2BA    FCB   $04
0400   C2BB 7E E4 29          LC2BB    JMP   ALINK15                          $E429
0401   C2BE 12                         NOP
0402
0403                          * PATCH 16 - CHECK FOR BREAK KEY ON BASIC'S LINE INPUT
0404   C2BF A3 C2             LC2BF    FDB   PATCH16                          $A3C2
0405   C2C1 04                LC2C1    FCB   $04
0406   C2C2 7E E4 13          LC2C2    JMP   ALINK16                          $E413
0407   C2C5 12                         NOP
0408
0409                          * PATCH 17 - PATCH INPUT TO RESPOND TO ON BRK
0410   C2C6 B0 3D             LC2C6    FDB   PATCH17+1                        $B03C+1
0411   C2C8 02                LC2C8    FCB   $02
0412   C2C9 E5 32             LC2C9    FDB   ALINK17                          $E532
0413
0414                          * PATCH 18 - 'ON' COMMAND
0415   C2CB AF 42             LC2CB    FDB   PATCH18                          $AF42
0416   C2CD 03                LC2CD    FCB   $03
0417   C2CE 7E E3 B4          LC2CE    JMP   ALINK18                          $E3B4
0418
0419                          * PATCH 19 - END OF 'NEW' COMMAND
0420   C2D1 AD 3F             LC2D1    FDB   PATCH19                          $AD3F
0421   C2D3 04                LC2D3    FCB   $04
0422   C2D4 7E E4 D0          LC2D4    JMP   ALINK19                          $E4D0
0423   C2D7 12                         NOP
0424
0425                          * PATCH 20 - ERROR SERVICING ROUTINE
0426   C2D8 AC 46             LC2D8    FDB   PATCH20                          $AC46
0427   C2DA 03                LC2DA    FCB   $03
0428   C2DB 7E E4 70          LC2DB    JMP   ALINK20                          $E470
0429
0430                          * PATCH 21 - BASIC'S MAIN LOOP IN THE DIRECT MODE
0431   C2DE AC 73             LC2DE    FDB   PATCH21                          $AC73
0432   C2E0 03                LC2E0    FCB   $03
0433   C2E1 7E E5 02          LC2E1    JMP   ALINK21                          $E502
0434
0435                          * PATCH 22
0436   C2E4 A3 0A             LC2E4    FDB   PATCH22                          $A30A
0437   C2E6 03                LC2E6    FCB   $03
0438   C2E7 7E 8C 37          LC2E7    JMP   PATCH22A                         $8C37
0439
0440                          * PATCH 23 - 'CLS' ROUTINE
0441   C2EA A9 10             LC2EA    FDB   PATCH23                          $A910
0442   C2EC 03                LC2EC    FCB   $03
0443   C2ED 7E 8C 46          LC2ED    JMP   PATCH23A                         $8C46
0444
0445                          * PATCH 24 - CURSOR BLINK
0446   C2F0 A1 B1             LC2F0    FDB   PATCH24                          $A1B1
0447   C2F2 08                LC2F2    FCB   $08
0448   C2F3 7E A0 CE          LC2F3    JMP   LA0CE                            $A0CE
0449   C2F6 12                         NOP
0450   C2F7 12                         NOP
0451   C2F8 12                         NOP
0452   C2F9 12                         NOP
0453   C2FA 12                         NOP
0454
0455                          * PATCH 25 - PRINT @ COMMAND
0456   C2FB B9 02             LC2FB    FDB   PATCH25                          $B902
0457   C2FD 03                LC2FD    FCB   $03
0458   C2FE 7E F8 C3          LC2FE    JMP   ALINK25                          $F8C3
0459
0460                          * PATCH 26
0461   C301 B9 5C             LC301    FDB   PATCH26                          $B95C
0462   C303 03                LC303    FCB   $03
0463   C304 7E F8 A3          LC304    JMP   ALINK26                          $F8A3
0464
0465                          * PATCH 27 - GET A BASIC INPUT LINE
0466   C307 A3 8D             LC307    FDB   PATCH27                          $A38D
0467   C309 03                LC309    FCB   $03
0468   C30A 7E F7 57          LC30A    JMP   ALINK27                          $F757
0469
0470                          * THESE DATA ARE THE NAMES OF THE AUTHORS IN COMPLEMENTED ASCII (T.Harris & T.Earles,CR,0)
0471   C30D AB D1 B7 9E 8D 8D LC30D    FCB   $AB,$D1,$B7,$9E,$8D,$8D
0472   C313 96 8C DF D9 DF AB         FCB   $96,$8C,$DF,$D9,$DF,$AB
0473   C319 D1 BA 9E 8D 93 9A         FCB   $D1,$BA,$9E,$8D,$93,$9A
0474   C31F 8C F2 FF                  FCB   $8C,$F2,$FF
0475
0476   C322 B6 C0 04          LC322    LDA   DCNVEC                           LOOK FOR THE MS BYTE OF THE ADDRESS OF DSKCON
0477   C325 81 D6                      CMPA  #$D6                             IF IT IS D6, THEN WE HAVE DISK BASIC 1.0
0478   C327 26 0B                      BNE   LC334                            BRANCH IF DISK BASIC 1.1
0479   C329 8E C0 C6                   LDX   #PATCH28                         POINT X TO DISK BASIC 1.0 PATCH ADDRESS ($C0C6)
0480   C32C 31 8D 00 25                LEAY  LC355,PC                         POINT Y TO THE PATCH DATA
```

```
0481   C330 E6 A0                    LDB   ,Y+                          GET THE NUMBER OF BYTES TO PATCH
0482   C332 20 15                    BRA   LC349
0483   C334 8E C8 B4          LC334  LDX   #PATCH30                      POINT X TO DISK BASIC 1.1 KEYBOARD PATCH ($C8B4)
0484   C337 86 12                    LDA   #$12                          OP CODE OF A NOP INSTRUCTION
0485   C339 C6 0B                    LDB   #11                           PATCH 11 BYTES
0486   C33B A7 80             LC33B  STA   ,X+                           STORE A NOP
0487   C33D 5A                       DECB                               DECREMENT COUNTER
0488   C33E 26 FB                    BNE   LC33B                        LOOP UNTIL DONE
0489   C340 8E C0 D9                 LDX   #PATCH29                     POINT X TO DISK BASIC 1.1 PATCH ADDRESS ($C0D9)
0490   C343 31 8D 00 0A              LEAY  LC351,PC                     POINT Y TO THE PATCH DATA
0491   C347 E6 A0                    LDB   ,Y+                          GET THE NUMBER OF BYTES TO PATCH
0492   C349 A6 A0             LC349  LDA   ,Y+                          GET A PATCH BYTE
0493   C34B A7 80                    STA   ,X+                          STORE THE PATCH BYTE
0494   C34D 5A                       DECB                               DECREMENT THE PATCH COUNTER
0495   C34E 26 F9                    BNE   LC349                        LOOP UNTIL DONE
0496   C350 39                       RTS
0497                          * DISK BASIC ROM PATCHES (COPYRIGHT MESSAGE)
0498   C351 03                LC351  FCB   $03
0499   C352 7E E2 9D          LC352  JMP   ALINK29                       $E29D
0500   C355 03                LC355  FCB   $03
0501   C356 7E E2 97          LC356  JMP   ALINK28                       $E297
0502
0503                          * INTERRUPT VECTOR IMAGES
0504                          * THESE LBRAs WILL LINK TO BASIC'S RAM INTERRUPT VECTORS AT $100
0505   C359 55                INTIMAGE FCB   $55                         VALIDITY FLAG (INTERRUPT VECTORS VALID/INVALID)
0506   C35A 16 02 0F          LC35A  LBRA  (INTIMAGE+1)-(INT.JUMP)+SW3VEC
0507   C35D 16 02 0F                 LBRA  (INTIMAGE+1)-(INT.JUMP)+SW2VEC
0508   C360 16 02 18                 LBRA  (INTIMAGE+1)-(INT.JUMP)+FRQVEC
0509   C363 16 02 12                 LBRA  (INTIMAGE+1)-(INT.JUMP)+IRQVEC
0510   C366 16 02 09                 LBRA  (INTIMAGE+1)-(INT.JUMP)+SWIVEC
0511   C369 16 02 09                 LBRA  (INTIMAGE+1)-(INT.JUMP)+NMIVEC
0512
0513                          * END OF THE DATA COPIED INTO RAM
0514   C36C                   ENDMOVE RMB   153                          UNUSED
0515
0516   C405                   AUTHPIC RMB   $1800                        COCO 2 COMPATIBLE DIGITIZED PICTURE OF THE AUTHORS
0517
0518   DC05                   LDC05   RMB   1019                         UNUSED
0519
0520                          * THE NEW SUPER EXTENDED BASIC CODE STARTS HERE
0521
0522                          * THE CODE FROM THIS POINT TO $FDFF IS THE ENHANCEMENTS ADDED TO THE 'OLD' COCO BASIC
0523                          * TO SUPPORT THE NEW FEATURES AVAILABLE IN THE COCO 3.
0524
0525                          * THESE ARE THE ONLY 'SANCTIONED BY TANDY' LEGAL ENTRY POINTS INTO THE SUPER
0526                          * EXTENDED (ENHANCED) PORTION OF THE BASIC ROM
0527
0528   E000 00 E6             SUPERVAR FDB   HRMODE                      ADDRESS OF DIRECT PAGE VARIABLES UNIQUE TO ENHANCED BASIC
0529   E002 E0 19             PRGTEXT  FDB   SETTEXT                     SET THE VIDEO CONTROL REGISTERS TO DISPLAY HI-RES TEXT
0530   E004 E0 4D             PRGGRAPH FDB   SETGRAPH                    SET THE VIDEO CONTROL REGISTERS TO DISPLAY HI-RES GRAPHICS
0531   E006 E0 97             PRGMMU   FDB   SETMMU                      PROGRAM THE MMU REGISTERS FROM THEIR IMAGES
0532   E008 E0 B5             GETTEXT  FDB   SELTEXT                     PLACE THE HI-RES TEXT SCREEN INTO LOGICAL BLOCK 1
0533   E00A E0 A1             GETBLOK0 FDB   SELBLOK0                   PLACE THE BLOCK NUMBER IN ACCB INTO LOGICAL BLOCK 0
0534   E00C E0 FF             GETTASK0 FDB   SELTASK0                   RE-SELECT TASK REGISTER 0
0535   E00E E1 19             GETTASK1 FDB   SELTASK1                   SELECT TASK REGISTER 1
0536   E010 7E A0 5E          LE010    JMP   EXECCART                    EXECUTE A ROM CARTRIDGE ($A05E)
0537   E013 00 00             SPARE0   FDB   $0000                      UNDEFINED
0538   E015 00 00             SPARE1   FDB   $0000                      UNDEFINED
0539   E017 00 00             SPARE2   FDB   $0000                      UNDEFINED
0540
0541                          * SET UP THE VIDEO CONTROL REGISTERS ACCORDING TO THE SELECTED WIDTH
0542   E019 34 32             SETTEXT  PSHS  Y,X,A
0543   E01B 10 21 1F E1               LBRN  RAMLINK                      RAM HOOK
0544   E01F 8E E0 32                  LDX   #IM.TEXT                     POINT TO THE 32 COLUMN VIDEO MODE REGISTER TABLE
0545   E022 96 E7                     LDA   HRWIDTH                      CHECK THE HI-RES TEXT MODE
0546   E024 27 5C                     BEQ   SETVIDEO                     BRANCH IF 32 COLUMN MODE
0547   E026 8E E0 3B                  LDX   #LE03B                       POINT TO THE 40 COLUMN VIDEO MODE REGISTER TABLE
0548   E029 81 01                     CMPA  #$01                         VIDEO MODE WIDTH SET TO 40 COLUMN?
0549   E02B 27 55                     BEQ   SETVIDEO                     YES
0550   E02D 8E E0 44                  LDX   #LE044                       POINT TO THE 80 COLUMN VIDEO MODE REGISTER TABLE
0551   E030 20 50                     BRA   SETVIDEO
0552
0553                          * VIDEO MODE REGISTER IMAGES FOR THE HI-RES TEXT MODES
0554                          * INITIAL VIDEO CONTROL REGISTER DATA FOR 32 COLUMN COCO COMPATIBLE MODE
0555   E032 CC                IM.TEXT  FCB   COCO+MMUEN+MC3+MC2           FF90
0556   E033 00 00 00 00 0F E0 LE033    FCB   $00,$00,$00,$00,$0F,$E0      FF98
0557   E039 00 00                     FCB   $00,$00
0558                          * INITIAL VIDEO CONTROL REGISTER DATA FOR 40 COLUMN HI-RES MODE
0559   E03B 4C                LE03B    FCB   MMUEN+MC3+MC2                FF90
0560   E03C 03 05 12 00 00 D8 LE03C    FCB   $03,$05,$12,$00,$00,$D8      FF98
0561   E042 00 00                     FCB   $00,$00
0562                          * INITIAL VIDEO CONTROL REGISTER DATA FOR 80 COLUMN HI-RES MODE
0563   E044 4C                LE044    FCB   MMUEN+MC3+MC2                FF90
0564   E045 03 15 12 00 00 D8 LE045    FCB   $03,$15,$12,$00,$00,$D8      FF98
0565   E04B 00 00                     FCB   $00,$00
0566
0567   E04D 34 32             SETGRAPH PSHS  Y,X,A
0568   E04F 10 21 1F AD               LBRN  RAMLINK                      RAM HOOK
0569   E053 8E E0 70                  LDX   #IM.GRAPH                    POINT TO THE VIDEO MODE RAM IMAGE FOR HSCREEN MODES 1,2
0570   E056 10 8E E0 6C               LDY   #RESTABLE                    POINT TO THE VIDEO RESOLUTION TABLE
0571   E05A 96 E6                     LDA   HRMODE                       GET THE HI-RES GRAPHICS MODE
0572   E05C 81 02                     CMPA  #$02                         1 OR 2 ARE 40 COLUMN MODES
0573   E05E 23 03                     BLS   LE063                        BRANCH IF 40 COLUMN TEXT MODE
0574   E060 8E E0 79                  LDX   #LE079                       POINT TO THE VIDEO RAM IMAGE FOR 80 COLUMN MODE
0575   E063 80 01             LE063    SUBA  #$01                         ADJUST MODE NUMBERS TO START AT ZERO
0576   E065 A6 A6                     LDA   A,Y                          GRAB THE PROPER VIDEO RESOLUTION MODE
```

```
0577   E067 A7 02                    STA   $02,X                          SAVE IT IN THE PROPER IMAGE
0578   E069 7E E0 82                 JMP   SETVIDEO                       GO SET UP THE VIDEO REGISTERS
0579
0580                         * VIDEO RESOLUTION MODE REGISTER (FF99) DATA FOR HSCREEN MODES
0581   E06C 15               RESTABLE FCB   $15                           320 PIXELS, 4 COLORS
0582   E06D 1E                LE06D   FCB   $1E                           320 PIXELS, 16 COLORS
0583   E06E 14                LE06E   FCB   $14                           640 PIXELS, 2 COLORS
0584   E06F 1D                LE06F   FCB   $1D                           640 PIXELS, 4 COLORS
0585
0586                         * VIDEO MODE REGISTER IMAGES FOR THE HI-RES GRAPHICS MODES
0587                         * VIDEO MODE REGISTER IMAGE FOR THE 320x192 GRAPHICS MODE
0588   E070 4C               IM.GRAPH FCB   MMUEN+MC3+MC2                 FF90
0589   E071 80 00 00 00 00 C0 LE071   FCB   $80,$00,$00,$00,$00,$C0        FF98
0590   E077 00 00                    FCB   $00,$00
0591                         * VIDEO MODE REGISTER IMAGE FOR THE 640x192 GRAPHICS MODE
0592   E079 4C                LE079   FCB   MMUEN+MC3+MC2                 FF90
0593   E07A 80 00 00 00 00 C0 LE07A   FCB   $80,$00,$00,$00,$00,$C0        FF98
0594   E080 00 00                    FCB   $00,$00
0595
0596                         * PROGRAM INIT0 AND THE 8 VIDEO MODE REGISTERS
0597                         * ENTER WITH X POINTING TO THE DATA TO PUT INTO THE REGISTERS
0598   E082 A6 80            SETVIDEO LDA   ,X+                           GET THE FIRST BYTE
0599   E084 B7 FF 90                 STA   INIT0                          AND PUT IT INTO INIT0
0600   E087 10 8E FF 98              LDY   #VIDEOREG                      POINT TO THE VIDEO MODE REGISTERS
0601   E08B A6 80             LE08B  LDA   ,X+                            GET A BYTE
0602   E08D A7 A0                    STA   ,Y+                            AND STICK IT INTO THE VIDEO MODE REGISTER
0603   E08F 10 8C FF A0              CMPY  #MMUREG                        END OF THE VIDEO MODE REGISTERS?
0604   E093 25 F6                    BCS   LE08B                          NO - KEEP STUFFING REGISTERS
0605   E095 35 B2                    PULS  A,X,Y,PC
0606
0607                         * PROGRAM THE MMU REGISTERS; ENTER WITH X POINTING TO THE DATA TO PLACE INTO THE MMU REGISTERS
0608   E097 34 36            SETMMU   PSHS  Y,X,B,A
0609   E099 30 8D 00 44              LEAX  IM.MMU,PC                      POINT TO THE RAM IMAGE OF THE MMU REGISTERS
0610   E09D 8D 52                    BSR   LE0F1                          MOVE 16 BYTES INTO THE MMU REGISTERS
0611   E09F 35 B6                    PULS  A,B,X,Y,PC
0612
0613                         * PLACE A BLOCK INTO LOGICAL ADDRESS SPACE BLOCK 0.
0614                         * ENTER WITH ACCB CONTAINING THE BLOCK NUMBER TO BE PLACED INTO THE LOGICAL ADDRESS SPACE
0615                         * EXIT WITH BLOCK 7.0 REPLACED IN BLOCK 0 OF THE LOGICAL ADDRESS SPACE RAM IMAGE
0616   E0A1 34 36            SELBLOK0 PSHS  Y,X,B,A
0617   E0A3 30 8D 00 3A              LEAX  IM.MMU,PC                      POINT TO THE RAM IMAGE OF THE MMU REGISTERS
0618   E0A7 34 10                    PSHS  X                              TEMP SAVE
0619   E0A9 E7 84                    STB   ,X                             SAVE THE NEW BLOCK NUMBER IN LOGICAL ADDRESS SPACE BLOCK 0 (TR0)
0620   E0AB 8D 44                    BSR   LE0F1                          COPY THE RAM IMAGE OF THE MMU REGISTERS INTO THE MMU REGISTERS
0621   E0AD C6 38                    LDB   #BLOCK7.0                      GET BLOCK 7.0
0622   E0AF 35 10                    PULS  X                              RESTORE THE MMU IMAGE POINTER
0623   E0B1 E7 84                    STB   ,X                             RESTORE BLOCK 7.0 TO BLOCK 0 OF MMU RAM IMAGE
0624   E0B3 35 B6                    PULS  A,B,X,Y,PC
0625
0626                         * PLACE THE HI-RES TEXT SCREEN INTO LOGICAL ADDRESS SPACE BLOCK 1
0627                         * EXIT WITH BLOCK 7.1 REPLACED INTO BLOCK 1 OF THE LOGICAL ADDRESS SPACE RAM IMAGE
0628   E0B5 34 36            SELTEXT  PSHS  Y,X,B,A
0629   E0B7 30 8D 00 26              LEAX  IM.MMU,PC                      POINT TO THE RAM IMAGE OF THE MMU REGISTERS
0630   E0BB 34 10                    PSHS  X                              TEMP SAVE
0631   E0BD C6 36                    LDB   #BLOCK6.6                      GET THE BLOCK WHICH CONTAINS THE HI-RES TEXT SCREEN
0632   E0BF E7 01                    STB   $01,X                          AND SAVE IT IN THE MMU IMAGE OF TASK REGISTER 0
0633   E0C1 8D 2E                    BSR   LE0F1                          COPY THE RAM IMAGE OF THE MMU REGISTERS INTO THE MMU REGISTERS
0634   E0C3 35 10                    PULS  X                              RESTORE THE MMU IMAGE PONTER
0635   E0C5 C6 39                    LDB   #BLOCK7.1                      GET BLOCK 7.1 (BASIC'S NORMAL LOGICAL BLOCK 1)
0636   E0C7 E7 01                    STB   $01,X                          AND SAVE IT IN THE MMU IMAGE
0637   E0C9 35 B6                    PULS  A,B,X,Y,PC
0638
0639   E0CB 34 36             LE0CB  PSHS  Y,X,B,A
0640   E0CD 30 8D 00 10              LEAX  IM.MMU,PC                      POINT TO THE MMU RAM IMAGE
0641   E0D1 34 10                    PSHS  X                              TEMP SAVE
0642   E0D3 C6 34                    LDB   #BLOCK6.4                      GET BLOCK 6.4
0643   E0D5 E7 0E                    STB   14,X                           AND SAVE IT IN LOGICAL BLOCK 6 OF TASK REGISTER 1
0644   E0D7 8D 18                    BSR   LE0F1                          COPY THE RAM IMAGE OF THE MMU REGISTERS INTO THE MMU REGISTERS
0645   E0D9 35 10                    PULS  X                              RESTORE MMU IMAGE POINTER
0646   E0DB C6 35                    LDB   #BLOCK6.5                      GET THE 'NORMAL' BLOCK FOR TASK REGISTER 1, LOGICAL BLOCK 6
0647   E0DD E7 0E                    STB   14,X                           PUT IT BACK INTO TASK REGISTER 1 IMAGE
0648   E0DF 35 B6                    PULS  A,B,X,Y,PC
0649
0650                         * MASTER IMAGES USED TO PROGRAM THE CUSTOM CHIP'S MMU REGISTERS
0651                         * TASK REGISTER 0
0652   E0E1 38 39 3A 3B 3C 3D IM.MMU  FCB   BLOCK7.0,BLOCK7.1,BLOCK7.2      DEFAULT VALUES
0653   E0E7 3E 3F                    BLOCK7.3,BLOCK7.4,BLOCK7.5
0654                                 BLOCK7.6,BLOCK7.7
0655                         * TASK REGISTER 1
0656   E0E9 38 30 31 32 33 3D LE0E9   FCB   BLOCK7.0,BLOCK6.0,BLOCK6.1      DEFAULT VALUES
0657   E0EF 35 3F                    BLOCK6.2,BLOCK6.3,BLOCK7.5
0658                                 BLOCK6.5,BLOCK7.7
0659
0660                         * COPY 16 BYTES INTO THE MMU REGISTERS
0661                         * ENTER WITH X POINTING TO THE 16 BYTES
0662   E0F1 10 8E FF A0       LE0F1   LDY   #MMUREG                        POINT TO THE MMU REGISTERS
0663   E0F5 C6 10                    LDB   #16                            16 MMU REGISTERS
0664   E0F7 A6 80             LE0F7  LDA   ,X+                            GET A BYTE
0665   E0F9 A7 A0                    STA   ,Y+                            AND PUT IT INTO THE MMU REGISTER
0666   E0FB 5A                       DECB                                 DECREMENT THE BYTE COUNT
0667   E0FC 26 F9                    BNE   LE0F7                          KEEP GOING UNTIL ALL REGISTERS MOVED
0668   E0FE 39                       RTS
0669
0670                         * SELECT TASK REGISTER 0 AS THE ACTIVE TASK REGISTER
0671                         * ENTER WITH THE STACK POINTING TO A TEMPORARY LOCATION; THE PERMANENT
0672                         * STACK POINTER WAS SAVED ON THIS TEMPORARY STACK WHEN TASK REGISTER 1
```

```
0673                            * WAS SELECTED AS THE ACTIVE TASK REGISTER
0674
0675   E0FF DD 40               SELTASK0 STD   V40                          TEMPORARILY SAVE ACCD
0676   E101 EC E4                       LDD   ,S                            GET THE RETURN ADDRESS OFF THE STACK
0677   E103 DD 42                       STD   V42                           AND TEMPORARILY SAVE IT IN V42
0678   E105 EC 62                       LDD   $02,S                         GET THE PERMANENT STACK POINTER FROM THE STACK
0679   E107 DD 44                       STD   V44                           AND TEMPORARILY SAVE IT IN V44
0680   E109 5F                          CLRB                                TASK REGISTER 0 AND TIMER INPUT OF 63.5 MICROSECONDS
0681   E10A F7 FF 91                    STB   INIT1                         PROGRAM INITIALIZATION REGISTER 1
0682   E10D 10 DE 44                    LDS   V44                           RESET THE STACK POINTER
0683   E110 DC 42                       LDD   V42                           GET BACK THE RETURN ADDRESS
0684   E112 34 06                       PSHS  B,A                           AND PUT IT ONTO THE STACK
0685   E114 DC 40                       LDD   V40                           RESTORE ACCD
0686   E116 1C AF                       ANDCC #$AF                          TURN ON IRQ, FIRQ
0687   E118 39                          RTS
0688
0689                            * SELECT TASK REGISTER 1 AS THE ACTIVE TASK REGISTER
0690                            * EXIT WITH THE STACK POINTER SET TO A TEMPORARY LOCATION
0691   E119 1A 50               SELTASK1 ORCC  #$50                          DISABLE INTERRUPTS
0692   E11B DD 40                       STD   V40                           TEMPORARILY SAVE ACCD IN V40
0693   E11D 35 06                       PULS  A,B                           GET THE RETURN ADDRESS
0694   E11F DD 42                       STD   V42                           AND TEMPORARILY SAVE IT IN V42
0695   E121 10 DF 44                    STS   V44                           TEMPORARILY SAVE THE STACK POINTER IN V44
0696   E124 C6 01                       LDB   #$01                          TASK REGISTER 1 AND TIMER INPUT AT 63.5 MICROSECONDS
0697   E126 F7 FF 91                    STB   INIT1                         SETUP INITIALIZATION REGISTER 1
0698   E129 10 CE DF FF                 LDS   #TMPSTACK                     PUT THE STACK JUST BELOW THE START OF ENHANCED BASIC
0699   E12D DC 44                       LDD   V44                           GET THE OLD STACK POINTER BACK
0700   E12F 34 06                       PSHS  B,A                           AND STUFF IT ONTO THE STACK
0701   E131 DC 42                       LDD   V42                           GET THE RETURN ADDRESS BACK
0702   E133 34 06                       PSHS  B,A                           AND STUFF IT ONTO THE STACK TOO
0703   E135 DC 40                       LDD   V40                           GET BACK ACCD
0704   E137 39                          RTS
0705
0706                            * CRUNCH A TOKEN PATCH ENTERED FROM $B8D4
0707   E138 0D 41               ALINK2   TST   V41                           CHECK THE TOKEN FLAG
0708   E13A 26 16                       BNE   LE152                         BRANCH IF IT IS A FUNCTION TOKEN
0709   E13C 96 42                       LDA   V42                           GET THE TOKEN COUNTER
0710   E13E 81 62                       CMPA  #$62                          COMPARE TO THE FIRST ENHANCED BASIC TOKEN
0711   E140 23 06                       BLS   LE148                         BRANCH IF BEFORE FIRST TOKEN
0712   E142 CE 01 1B                    LDU   #COMVEC-5                     POINT U TO EXTENDED COLOR BASIC'S INTERPRETATION TABLE
0713   E145 7E B8 D7                    JMP   LB8D7                         RE-ENTER THE MAIN STREAM CODE
0714   E148 86 62               LE148    LDA   #$62                          FORCE THE TOKEN COUNTER TO THE FIRST ENHANCED BASIC TOKEN NUMBER
0715   E14A CE E1 58                    LDU   LE158                         POINT TO ENHANCED BASIC'S COMMAND INTERPRETATION TABLE
0716   E14D 97 42               LE14D    STA   V42                           SAVE THE NEW TOKEN COUNTER
0717   E14F 7E B8 9D                    JMP   LB89D                         RE-ENTER THE MAIN STREAM CODE
0718   E152 96 42               LE152    LDA   V42                           GET THE TOKEN COUNTER
0719   E154 81 29                       CMPA  #$29                          COMPARE TO THE FIRST ENHANCED FUNCTION TOKEN NUMBER
0720   E156 23 03                       BLS   LE15B                         BRANCH IF LESS THAN ENHANCED TOKEN NUMBER
0721   E158 7E B8 D7            LE158    JMP   LB9D7                         RE-ENTER THE MAIN STREAM CODE
0722   E15B 86 29               LE15B    LDA   #$29                          FORCE COUNTER TO FIRST ENHANCED FUNCTION
0723   E15D CE E1 5D            LE15D    LDU   #LE15D                        POINT TO THE ENHANCED FUNCTION INTERPRETATION TABLE
0724   E160 20 EB                       BRA   LE14D
0725
0726                            * BASIC 2.0 COMMAND INTERPRETATION VECTOR TABLE
0727   E162 17                  EBCOMTAB FCB   23                            23 BASIC 2.0 COMMANDS
0728   E163 E1 C5               LE163    FDB   COMDIC20                     BASIC 2.0'S COMMAND DICTIONARY
0729   E165 E1 92               LE165    FDB   ALINK4                      COMMAND PROCESSING ROUTINE ENTRY POINT
0730   E167 05                  LE167    FCB   5                             5 BASIC 2.0 FUNCTIONS
0731   E168 E2 64               LE168    FDB   FUNDIC20                    FUNCTION DICTIONARY TABLE
0732   E16A E1 A6               LE16A    FDB   ALINK5                      FUNCTION PROCESSING ROUTINE ENTRY POINT
0733   E16C 00 00 00 00 00 00   LE16C    FCB   $00,$00,$00,$00,$00,$00      DUMMY SPACE USED TO SIMULATE AN EMPTY COMMAND INTERP. VECTOR TABLE
0734
0735                            * UNCRUNCH A TOKEN PATCH ENTERED FROM $B7F3
0736   E172 33 4A               ALINK3   LEAU  10,U                         SKIP TO THE NEXT COMMAND INTERPRETATION TABLE
0737   E174 6D C4                       TST   ,U                            IS THIS A VALID TABLE?
0738   E176 10 26 D6 7F                 LBNE  LD67F                         YES - RE-ENTER THE MAIN STREAM CODE
0739   E17A 30 1F                       LEAX  $-01,X                        UNNECESSARY INSTRUCTION; NEXT ONE SHOULD JUST BE LDA -1,X
0740   E17C A6 80                       LDA   ,X+                           GET THE TOKEN FROM BASIC'S INPUT LINE
0741   E17E 84 7F                       ANDA  #$7F                          STRIP OFF THE $80 COMMAND TOKEN BIAS
0742   E180 81 62                       CMPA  #$62                          FIRST LEGAL BASIC 2.0 COMMAND TOKEN NUMBER
0743   E182 25 07                       BCS   LE18B                         BRANCH IF LEGAL TOKEN
0744   E184 80 62                       SUBA  #$62                          ADJUST BASIC 2.0 TOKENS TO START AT 0
0745   E186 CE E1 58                    LDU   #LE158                        POINT TO ENHANCED BASIC'S COMMAND INTERPRETATION TABLE
0746   E189 20 E7                       BRA   ALINK3
0747   E18B 80 29               LE18B    SUBA  #$29                          SUBTRACT OUT THE FIRST ENHANCED FUNCTION TABLE
0748   E18D CE E1 5D                    LDU   #LE15D                        POINT U TO BE ABLE TO SEARCH FOR AN ENHANCED FUNCTION TOKEN
0749   E190 20 E0                       BRA   ALINK3
0750
0751                            * BASIC 2.0 COMMAND PROCESSING ROUTINE ENTRY POINT PATCH ENTERED FROM $8150
0752   E192 81 E2               ALINK4   CMPA  #$E2                          TOKEN NUMBER OF FIRST ENHANCED BASIC COMMAND
0753   E194 25 04                       BCS   LE19A                         BRANCH IF LESS THAN ENHANCED TOKEN
0754   E196 81 F8                       CMPA  #$F8                          COMPARE TO THE HIGHEST ENHANCED BASIC TOKEN
0755   E198 23 04                       BLS   LE19E                         BRANCH IF ENHANCED BASIC TOKEN
0756   E19A 6E 9F 01 37         LE19A    JMP   [COMVEC+23]                   GO TO DISK BASIC'S COMMAND HANDLER
0757   E19E 80 E2               LE19E    SUBA  #$E2                          SUBTRACT OUT THE NON-ENHANCED BASIC TOKENS
0758   E1A0 8E E2 36                    LDX   #COMDIS20                     POINT X TO ENHANCED BASIC'S COMMAND DISPATCH TABLE
0759   E1A3 7E AD D4                    JMP   LADD4                         RE-ENTER THE MAIN STREAM CODE
0760
0761                            * BASIC 2.0 FUNCTION PROCESSING ROUTINE PATCH ENTERED FROM $816C
0762   E1A6 C1 52               ALINK5   CMPB  #$52                          COMPARE TO THE FIRST ENHANCED BASIC FUNCTION TOKEN
0763   E1A8 25 04                       BCS   LE1AE                         BRANCH IF LESS THAN ENHANCED TOKEN
0764   E1AA C1 5A                       CMPB  #$5A                          COMPARE TO THE HIGHEST FUNCTION TOKEN
0765   E1AC 23 04                       BLS   LE1B2                         BRANCH IF ENHANCED TOKEN
0766   E1AE 6E 9F 01 3C         LE1AE    JMP   [COMVEC+28]                   JUMP TO DISK BASIC'S FUNCTION HANDLER
0767   E1B2 C0 52               LE1B2    SUBB  #$52                          SUBTRACT OUT THE NON-ENHANCED BASIC TOKENS
0768   E1B4 C1 04                       CMPB  #2*2                          CHECK FOR LPEEK, BUTTON, HPOINT
```

```
0769   E1B6 24 07                    BCC   LE1BF                          BRANCH IF ERNO, ERLIN
0770   E1B8 34 04                    PSHS  B                              SAVE THE TOKEN COUNTER
0771   E1BA BD B2 62                 JSR   LB262                          EVALUATE AN EXPRESSION IN PARENTHESIS
0772   E1BD 35 04                    PULS  B                              RESTORE THE TOKEN COUNTER
0773   E1BF 8E E2 7E          LE1BF  LDX   #FUNDIS2Ø                      POINT TO ENHANCED BASIC'S FUNCTION DISPATCH TABLE
0774   E1C2 7E B2 CE                 JMP   LB2CE                          RE-ENTER THE MAIN STREAM CODE
0775                          *
0776                          * BASIC 2.Ø COMMAND DICTIONARY TABLE
0777                          *
0778                          *                                            TOKEN #
0779   E1C5 57 49 44 54 C8    COMDIC2Ø FCC   'WIDT',$8Ø+'H'               E2
0780   E1CA 5Ø 41 4C 45 54 54 LE1CA  FCC   'PALETT',$8Ø+'E'               E3
0781   E1DØ C5
0782   E1D1 48 53 43 52 45 45 LE1D1  FCC   'HSCREE',$8Ø+'N'               E4
0783   E1D7 CE
0784   E1D8 4C 5Ø 4F 4B C5    LE1D8  FCC   'LPOK',$8Ø+'E'                 E5
0785   E1DD 48 43 4C D3       LE1DD  FCC   'HCL',$8Ø+'S'                  E6
0786   E1E1 48 43 4F 4C 4F D2 LE1E1  FCC   'HCOLO',$8Ø+'R'               E7
0787   E1E7 48 5Ø 41 49 4E D4 LE1E7  FCC   'HPAIN',$8Ø+'T'               E8
0788   E1ED 48 43 49 52 43 4C LE1ED  FCC   'HCIRCL',$8Ø+'E'              E9
0789   E1F3 C5
0790   E1F4 48 4C 49 4E C5    LE1F4  FCC   'HLIN',$8Ø+'E'                 EA
0791   E1F9 48 47 45 D4       LE1F9  FCC   'HGE',$8Ø+'T'                  EB
0792   E1FD 48 5Ø 55 D4       LE1FD  FCC   'HPU',$8Ø+'T'                  EC
0793   E2Ø1 48 42 55 46 C6    LE2Ø1  FCC   'HBUF',$8Ø+'F'                 ED
0794   E2Ø6 48 5Ø 52 49 4E D4 LE2Ø6  FCC   'HPRIN',$8Ø+'T'               EE
0795   E2ØC 45 52 D2          LE2ØC  FCC   'ER',$8Ø+'R'                   EF
0796   E2ØF 42 52 CB          LE2ØF  FCC   'BR',$8Ø+'K'                   FØ
0797   E212 4C 4F 43 41 54 C5 LE212  FCC   'LOCAT',$8Ø+'E'               F1
0798   E218 48 53 54 41 D4    LE218  FCC   'HSTA',$8Ø+'T'                 F2
0799   E21D 48 53 45 D4       LE21D  FCC   'HSE',$8Ø+'T'                  F3
0800   E221 48 52 45 53 45 D4 LE221  FCC   'HRESE',$8Ø+'T'               F4
0801   E227 48 44 52 41 D7    LE227  FCC   'HDRA',$8Ø+'W'                 F5
0802   E22C 43 4D DØ          LE22C  FCC   'CM',$8Ø+'P'                   F6
0803   E22F 52 47 C2          LE22F  FCC   'RG',$8Ø+'B'                   F7
0804   E232 41 54 54 D2       LE232  FCC   'ATT',$8Ø+'R'                  F8
0805                          *
0806                          * BASIC 2.Ø COMMAND DISPATCH TABLE
0807                          *
0808                          *                                                    TOKEN #
0809   E236 F6 36             COMDIS2Ø FDB   WIDTH                         WIDTH   E2
0810   E238 E5 FØ             LE238    FDB   PALETTE                       PALETTE E3
0811   E23A E6 88             LE23A    FDB   HSCREEN                      HSCREEN E4
0812   E23C E5 45             LE23C    FDB   LPOKE                        LPOKE   E5
0813   E23E E6 CF             LE23E    FDB   HCLS                         HCLS    E6
0814   E24Ø E6 F4             LE24Ø    FDB   HCOLOR                       HCOLOR  E7
0815   E242 EB F5             LE242    FDB   HPAINT                       HPAINT  E8
0816   E244 EA 49             LE244    FDB   HCIRCLE                      HCIRCLE E9
0817   E246 E8 82             LE246    FDB   HLINE                        HLINE   EA
0818   E248 ED E5             LE248    FDB   HGET                         HGET    EB
0819   E24A ED ED             LE24A    FDB   HPUT                         HPUT    EC
0820   E24C ED 58             LE24C    FDB   HBUFF                        HBUFF   ED
0821   E24E EF 3F             LE24E    FDB   HPRINT                       HPRINT  EE
0822   E25Ø E3 D4             LE25Ø    FDB   ERR                          ERR     EF
0823   E252 E3 E6             LE252    FDB   BRK                          BRK     FØ
0824   E254 F8 D2             LE254    FDB   LOCATE                       LOCATE  F1
0825   E256 F9 25             LE256    FDB   HSTAT                        HSTAT   F2
0826   E258 E7 61             LE258    FDB   HSET                         HSET    F3
0827   E25A E7 65             LE25A    FDB   HRESET                       HRESET  F4
0828   E25C F3 9D             LE25C    FDB   HDRAW                        HDRAW   F5
0829   E25E E6 76             LE25E    FDB   CMP                          CMP     F6
0830   E26Ø E6 74             LE26Ø    FDB   RGB                          RGB     F7
0831   E262 F9 B9             LE262    FDB   ATTR                         ATTR    F8
0832                          *
0833                          * BASIC 2.Ø FUNCTION DICTIONARY TABLE
0834                          *
0835                          *                                            TOKEN #
0836   E264 4C 5Ø 45 45 CB    FUNDIC2Ø FCC   'LPEE',$8Ø+'K'               A8
0837   E269 42 55 54 54 4F CE LE269    FCC   'BUTTO',$8Ø+'N'               A9
0838   E26F 48 5Ø 4F 49 4E D4 LE26F    FCC   'HPOIN',$8Ø+'T'               AA
0839   E275 45 52 4E CF       LE275    FCC   'ERN',$8Ø+'O'                 AB
0840   E279 45 52 4C 49 CE    LE279    FCC   'ERLI',$8Ø+'N'                AC
0841                          *
0842                          * BASIC 2.Ø FUNCTION DISPATCH TABLE
0843                          *
0844                          *                                                    TOKEN #
0845   E27E E5 73             FUNDIS2Ø FDB   LPEEK                         LPEEK   A8
0846   E28Ø E5 B1             LE28Ø    FDB   BUTTON                        BUTTON  A9
0847   E282 E8 5C             LE282    FDB   HPOINT                       HPOINT  AA
0848   E284 E4 E9             LE284    FDB   ERNO                         ERNO    AB
0849   E286 E4 FD             LE286    FDB   ERLIN                        ERLIN   AC
0850
0851                          * PRINT THE COPYRIGHT MESSAGE PATCH ENTERED FROM $8ØB2
0852   E288 8E 8Ø E7          ALINK12  LDX   #L8ØE7                        POINT TO EXTENDED BASIC'S COPYRIGHT MESSAGE
0853   E28B BD B9 9C                   JSR   STRINOUT                       COPY A STRING FROM (X) TO CONSOLE OUT
0854   E28E 8E E2 F7                   LDX   #MWAREMS-1                     MICROWARE'S COPYRIGHT MESSAGE
0855   E291 BD B9 9C                   JSR   STRINOUT                       COPY A STRING FROM (X) TO CONSOLE OUT
0856   E294 7E 8Ø B8                   JMP   L8ØB8                          EXTENDED BASIC'S WARM START REENTRY
0857
0858                          * PRINT THE DISK BASIC 2.Ø COPYRIGHT MESSAGE PATCH ENTERED FROM $CØC6
0859   E297 8E E2 A2          ALINK28  LDX   #DISK2ØMS-1                   POINT TO DISK BASIC 2.Ø MESSAGE
0860   E29A 7E CØ C9                   JMP   LCØC9                          COPY MESSAGE TO SCREEN AND WARM START DISK BASIC 2.Ø
0861                          * PRINT THE DISK BASIC 2.1 COPYRIGHT MESSAGE PATCH ENTERED FROM $CØC6
0862   E29D 8E E3 15          ALINK29  LDX   #LE313+2                      POINT TO DISK BASIC 2.1 MESSAGE
0863   E2AØ 7E CØ DC                   JMP   LCØDC                          COPY MESSAGE TO SCREEN AND WARM START DISK BASIC 2.1
0864
```

```
0865   E2A3 44 49 53 4B 20 45 DISK20MS FCC   'DISK EXTENDED COLOR BASIC 2.0'
0866   E2A9 58 54 45 4E 44 45
0867   E2AF 44 20 43 4F 4C 4F
0868   E2B5 52 20 42 41 53 49
0869   E2BB 43 20 32 2E 30
0870   E2C0 0D                LE2C0    FCB   $0D
0871   E2C1 43 4F 50 52 2E 20 LE2C1    FCC   'COPR. 1981, 1986 BY TANDY'
0872   E2C7 31 39 38 31 2C 20
0873   E2CD 31 39 38 36 20 42
0874   E2D3 59 20 54 41 4E 44
0875   E2D9 59
0876   E2DA 0D                LE2DA    FCB   $0D
0877   E2DB 55 4E 44 45 52 20 LE2DB    FCC   'UNDER LICENSE FROM MICROSOFT'
0878   E2E1 4C 49 43 45 4E 53
0879   E2E7 45 20 46 52 4F 4D
0880   E2ED 20 4D 49 43 52 4F
0881   E2F3 53 4F 46 54
0882   E2F7 0D                LE2F7    FCB   $0D
0883   E2F8 41 4E 44 20 4D 49 MWAREMS  FCC   'AND MICROWARE SYSTEMS CORP.'
0884   E2FE 43 52 4F 57 41 52
0885   E304 45 20 53 59 53 54
0886   E30A 45 4D 53 20 43 4F
0887   E310 52 50 2E
0888   E313 0D 0D 00          LE313    FCB   $0D,$0D,$00
0889   E316 44 49 53 4B 20 45 DISK21MS FCC   'DISK EXTENDED COLOR BASIC 2.1'
0890   E31C 58 54 45 4E 44 45
0891   E322 44 20 43 4F 4C 4F
0892   E328 52 20 42 41 53 49
0893   E32E 43 20 32 2E 31
0894   E333 0D                LE333    FCB   $0D
0895   E334 43 4F 50 52 2E 20 LE334    FCC   'COPR. 1981, 1986 BY TANDY'
0896   E33A 31 39 38 32 2C 20
0897   E340 31 39 38 36 20 42
0898   E346 59 20 54 41 4E 44
0899   E34C 59
0900   E34D 0D                LE33D    FCB   $0D
0901   E34E 55 4E 44 45 52 20 LE33E    FCC   'UNDER LICENSE FROM MICROSOFT'
0902   E354 4C 49 43 45 4E 53
0903   E35A 45 20 46 52 4F 4D
0904   E360 20 4D 49 43 52 4F
0905   E366 53 4F 46 54
0906   E36A 0D                LE36A    FCB   $0D
0907   E36B 41 4E 44 20 4D 49 LE36B    FCC   'AND MICROWARE SYSTEMS CORP.'
0908   E371 43 52 4F 57 41 52
0909   E377 45 20 53 59 53 54
0910   E37D 45 4D 53 20 43 4F
0911   E383 52 50 2E
0912   E386 0D 0D 00          LE386    FCB   $0D,$0D,$00
0913
0914                          * GRAPHICS INITIALIZATION PATCH ENTERED FROM $9703
0915   E389 4F                ALINK14  CLRA
0916   E38A 5F                         CLRB
0917   E38B 10 21 1C 71                LBRN  RAMLINK                       RAM HOOK
0918   E38F F7 FE 08                   STB   H.CRSATT                      SET CURSOR ATTRIBUTES TO ZERO
0919   E392 DD E6                      STD   HRMODE                        SET HI-RES GRAPHICS AND TEXT MODES TO OFF
0920   E394 FD FE 0C                   STD   H.ONBRK                       RESET THE ON BRK ADDRESS TO ZERO; NON-INITIALIZED
0921   E397 FD FE 0E                   STD   H.ONERR                       RESET THE ON ERROR ADDRES TO ZERO; NON-INITIALIZED
0922   E39A B7 FE 0B                   STA   H.BCOLOR                      PALETTE REGISTER ZERO IS THE DEFAULT BACKGROUND COLOR
0923   E39D 86 01                      LDA   #$01                          DEFAULT PALETTE REGISTER FOR THE FOREGROUND COLOR
0924   E39F B7 FE 0A                   STA   H.FCOLOR                      USE PALETTE REGISTER1 AS THE FOREGROUND COLOR
0925   E3A2 86 34                      LDA   #BLOCK6.4                     GET THE HPUT/HGET BUFFER BLOCK
0926   E3A4 B7 FF A0                   STA   MMUREG                        PIT IT INTO LOGICAL BLOCK 0
0927   E3A7 CC FF FF                   LDD   #$FFFF                        HPUT/HGET BUFFER EMPTY FLAG
0928   E3AA DD 00                      STD   $0                            RESET THE HPUT/HGET BUFFER TO EMPTY
0929   E3AC 86 38                      LDA   #BLOCK7.0
0930   E3AE B7 FF A0                   STA   MMUREG                        RESTORE BLOCK 7.0 TO LOGICAL BLOCK 0 OF TASK REGISTER 0
0931   E3B1 7E AD 19                   JMP   LAD19                         GO DO A COMPLETE 'NEW'
0932
0933                          * ON COMMAND (FOR ON ERR AND ON BRK) PATCH ENTERED FROM $AF42
0934   E3B4 81 EF             ALINK18  CMPA  #$EF                          'ERR' TOKEN
0935   E3B6 27 1C                      BEQ   ERR
0936   E3B8 81 F0                      CMPA  #$F0                          'BRK' TOKEN
0937   E3BA 27 2A                      BEQ   BRK
0938   E3BC BD B7 0B                   JSR   EVALEXPB                      EVALUATE EXPRESSION, RETURN VALUE IN ACCB
0939   E3BF 7E AF 45                   JMP   LAF45                         JUMP TO THE ON COMMAND($AF45)
0940   E3C2 9D 9F             LE3C2    JSR   GETNCH                        GET THE NEXT CHARACTER FROM BASIC'S INPUT LINE
0941   E3C4 81 81                      CMPA  #$81                          'GO' TOKEN
0942   E3C6 26 07                      BNE   LE3CF                         SYNTAX ERROR IF NOT GO
0943   E3C8 9D 9F                      JSR   GETNCH                        GET THE NEXT CHARACTER FROM BASIC'S INPUT LINE
0944   E3CA 81 A5                      CMPA  #$A5                          'TO' TOKEN
0945   E3CC 26 01                      BNE   LE3CF                         SYNTAX ERROR IF NOT GOTO
0946   E3CE 39                         RTS
0947   E3CF 32 62             LE3CF    LEAS  $02,S                         REMOVE ONE RETURN ADDRESS FROM THE STACK
0948   E3D1 7E B2 77                   JMP   LB277                         'SYNTAX' ERROR
0949
0950                          * ERR
0951   E3D4 8D EC             ERR      BSR   LE3C2                        CHECK FOR THE 'GO' AND 'TO' TOKENS
0952   E3D6 9D 9F                      JSR   GETNCH                        GET THE NEXT CHARACTER FROM BASIC'S INPUT LINE
0953   E3D8 BD AF 67                   JSR   LAF67                         STRIP THE 'GOTO' LINE NUMBER FROM THE BASIC INPUT LINE
0954   E3DB DC 2B                      LDD   BINVAL                        GET THE 'GOTO' LINE NUMBER
0955   E3DD FD FE 0E                   STD   H.ONERR                       SAVE IT
0956   E3E0 DC 68                      LDD   CURLIN                        GET THE CURRENT LINE NUMBER
0957   E3E2 FD FE 11                   STD   H.ONERRS                      AND SAVE IT AS THE SOURCE LINE NUMBER
0958   E3E5 39                         RTS
0959
0960                          * BRK
```

```
0961   E3E6 8D DA            BRK      BSR   LE3C2                          CHECK FOR THE 'GO' AND THE 'TO' TOKENS
0962   E3E8 9D 9F                     JSR   GETNCH                         GET THE NEXT CHARACTER FROM BASIC'S INPUT LINE
0963   E3EA BD AF 67                  JSR   LAF67                         STRIP THE 'GOTO' LINE NUMBER FROM THE BASIC INPUT LINE
0964   E3ED DC 2B                     LDD   BINVAL                        GET THE 'GOTO' LINE NUMBER
0965   E3EF FD FE 0C                  STD   H.ONBRK                       SAVE IT
0966   E3F2 DC 68                     LDD   CURLIN                        GET THE CURRENT LINE NUMBER
0967   E3F4 FD FE 15                  STD   H.ONBRKS                      AND SAVE IT AS THE SOURCE LINE NUMBER
0968   E3F7 39                        RTS
0969
0970                          * &H TYPE VARIABLE EVALUATION PATCH ENTERED FROM $8834
0971   E3F8 68 02            ALINK6A  LSL   $02,X                         *
0972   E3FA 69 01                     ROL   $01,X                         * MULTIPLY THE TEMPORARY
0973   E3FC 69 84                     ROL   ,X                            * ACCUMULATOR BY TWO
0974   E3FE 10 25 D6 90               LBCS  LBA92                         'OV' OVERFLOW ERROR ($BA92)
0975   E402 5A                        DECB                                DECREMENT THE SHIFT COUNTER
0976   E403 26 F3                     BNE   ALINK6A                       LOOP UNTIL DONE
0977   E405 80 30                     SUBA  #'0'                          MASK OFF ASCII
0978   E407 AB 02                     ADDA  $02,X                         * ADD DIGIT TO TEMPORARY
0979   E409 A7 02                     STA   $02,X                         * ACCUMULATOR AND SAVE IT
0980   E40B 39                        RTS
0981
0982                          * &H TYPE VARIABLE EVALUATION PATCH ENTERED FROM $8843
0983   E40C 10 25 A3 F0       ALINK6B  LBCS  L8800                         ($8800)
0984   E410 7E 88 3F                  JMP   L883F                         ($883F)
0985
0986                          * BASIC'S LINE INPUT PATCH ENTERED FROM $A3C2
0987   E413 81 03             ALINK16  CMPA  #$03                         BREAK KEY DEPRESSED?
0988   E415 1A 01                     ORCC  #$01                          SET THE CARRY FLAG
0989   E417 26 0D                     BNE   LE426                         BRANCH IF NOT THE BREAK KEY
0990   E419 34 03                     PSHS  A,CC                          SAVE REGISTERS
0991   E41B 96 E6                     LDA   HRMODE                        CHECK THE HI-RES GRAPHICS MODE
0992   E41D 27 05                     BEQ   LE424                         BRANCH IF IN COCO COMPATIBLE MODE
0993   E41F 0F E6                     CLR   HRMODE                        FORCE TO COCO COMPATIBLE MODE
0994   E421 BD E0 19                  JSR   SETTEXT                       PROGRAM THE VIDEO MODE REGISTERS
0995   E424 35 03             LE424    PULS  CC,A                          RESTORE REGISTERS
0996   E426 7E A3 C6          LE426    JMP   LA3C6                         RE-ENTER THE MAIN STREAM OF CODE ($A3C6)
0997
0998                          * BREAK CHECK PATCH ENTERED FROM $ADF0
0999   E429 81 03             ALINK15  CMPA  #$03                         BREAK KEY DEPRESSED?
1000   E42B 27 03                     BEQ   LE430                         YES
1001   E42D 7E AD F4                  JMP   LADF4                         RE-ENTER THE MAIN STREAM OF CODE ($ADF4)
1002   E430 86 01             LE430    LDA   #$01                          'BREAK' FLAG
1003   E432 B7 FE 17                  STA   H.ERRBRK                      SAVE IN THE ERROR/BREAK FLAG
1004   E435 96 68                     LDA   CURLIN                        DIRECT MODE?
1005   E437 4C                        INCA                                $FF SIGNIFIES DIRECT MODE
1006   E438 27 05                     BEQ   LE43F                         BRANCH IF DIRECT MODE
1007   E43A FC FE 0C                  LDD   H.ONBRK                       HAS AN ON BRK TRAP BEEN SET UP?
1008   E43D 26 0A                     BNE   LE449                         YES
1009   E43F 96 E6             LE43F    LDA   HRMODE                        CHECK THE HI-RES GRAPHICS MODE
1010   E441 27 03                     BEQ   LE446                         BRANCH IF COCO COMPATIBLE
1011   E443 BD E0 19                  JSR   SETTEXT                       PROGRAM THE VIDEO DISPLAY REGISTERS
1012   E446 7E AE 09          LE446    JMP   LAE09                         JUMP TO THE STOP COMMAND ($AE09)
1013   E449 DD 2B             LE449    STD   BINVAL                        SAVE THE SEARCH LINE NUMBER
1014   E44B 7D FE 17                  TST   H.ERRBRK                      CHECK THE ERROR/BREAK FLAG
1015   E44E 26 08                     BNE   LE458                         BRANCH IF BREAK
1016   E450 10 DE 21                  LDS   FRETOP                        IF ERROR, RESET THE STACK POINTER
1017   E453 CC AD C4                  LDD   #LADC4                        * GET THE ADDRESS ($ADC4) OF THE MAIN COMMAND INTERPRETATION
1018   E456 34 06                     PSHS  B,A                           * LOOP AND SAVE IT AS THE NEW RETURN ADDRESS
1019   E458 BD AE EB          LE458    JSR   LAEEB                         MOVE THE INPUT POINTER TO THE END OF THE LINE
1020   E45B 30 01                     LEAX  $01,X                         SKIP TO THE START OF THE NEXT LINE
1021   E45D DC 2B                     LDD   BINVAL                        GET THE LINE NUMBER WE'RE LOOKING FOR
1022   E45F 10 93 68                  CMPD  CURLIN                        COMPARE TO THE CURRENT LINE NUMBER
1023   E462 22 02                     BHI   LE466                         BRANCH IF SEARCH LINE NUMBER GREATER THAN CURRENT LINE NUMBER
1024   E464 9E 19                     LDX   TXTTAB                        POINT X TO THE BEGINNING OF THE PROGRAM
1025   E466 BD AD 05          LE466    JSR   LAD05                         SEARCH FOR THE PROGRAM LINE NUMBER IN ACCD
1026   E469 10 25 00 B1               LBCS  LE51E                         BRANCH IF LINE NUMBER NOT FOUND
1027   E46D 7E AE BB                  JMP   LAEBB                         RESET BASIC'S INPUT POINTER AND RETURN ($AEBB)
1028
1029                          * ERROR SERVICING ROUTINE PATCH ENTERED FROM $AC46
1030   E470 7F FE 17          ALINK20  CLR   H.ERRBRK                      SET THE ERROR/BREAK FLAG TO ERROR (0)
1031   E473 96 68                     LDA   CURLIN                        GET THE CURRENT LINE NUMBER
1032   E475 4C                        INCA                                CHECK FOR DIRECT MODE
1033   E476 27 05                     BEQ   LE47D                         BRANCH IF DIRECT MODE
1034   E478 BE FE 0E                  LDX   H.ONERR                       HAS AN ON ERROR TRAP BEEN SET UP?
1035   E47B 26 36                     BNE   LE4B3                         BRANCH IF ONE HAS
1036   E47D 34 02             LE47D    PSHS  A                             SAVE ACCA
1037   E47F 96 E6                     LDA   HRMODE                        TEST THE HI-RES GRAPHICS MODE
1038   E481 35 02                     PULS  A                             RESTORE ACCA
1039   E483 27 03                     BEQ   LE488                         BRANCH IF HI-RES GRAPHICS NOT SET UP
1040   E485 BD E0 19                  JSR   SETTEXT                       PROGRAM THE VIDEO CONTROL REGISTERS FOR THE CURRENT MODE
1041   E488 C1 4C             LE488    CMPB  #38*2                         HI-RES GRAPHICS ERROR
1042   E48A 26 13                     BNE   LE49F                         BRANCH IF NOT
1043   E48C BD B9 5C                  JSR   LB95C                         SET UP PRINT PARAMETERS
1044   E48F BD B9 AF                  JSR   LB9AF                         SEND A '?' TO CONSOLE OUT
1045   E492 30 8D 00 36               LEAX  BAS20ERR,PC                   POINT TO ENHANCED BASIC'S ADDITIONAL ERROR CODES
1046   E496 BD AC A0          LE496    JSR   LACA0                         GET A CHARACTER FROM X AND SEND IT TO CONSOLE OUT
1047   E499 BD AC A0                  JSR   LACA0                         DO IT AGAIN
1048   E49C 7E AC 65                  JMP   LAC65                         RE-ENTER THE MAIN STREAM OF CODE ($AC65)
1049   E49F C1 4E             LE49F    CMPB  #39*2                         HI-RES TEXT MODE ERROR
1050   E4A1 26 0D                     BNE   LE4B0                         BRANCH IF NOT
1051   E4A3 BD B9 5C                  JSR   LB95C                         SET UP THE PRINT PARAMETERS
1052   E4A6 BD B9 AF                  JSR   LB9AF                         SEND A '?' TO CONSOLE OUT
1053   E4A9 30 8D 00 21               LEAX  LE4CE,PC                      POINT TO ENHANCED BASIC'S ADDITIONAL ERROR CODES
1054   E4AD 7E E4 96                  JMP   LE496                         GO PRINT THE ERROR CODE POINTED TO BY X
1055   E4B0 7E AC 49          LE4B0    JMP   LAC49                         JUMP TO THE ERROR SERVICING ROUTINE ($AC49)
1056   E4B3 F7 FE 10          LE4B3    STB   H.ERROR                       SAVE THE ERROR NUMBER
```

```
1057   E4B6 34 04              PSHS  B                         ALSO PUT IT ON THE STACK TEMPORARILY
1058   E4B8 DC 68              LDD   CURLIN                     GET THE CURRENT LINE NUMBER
1059   E4BA FD FE 13           STD   H.ERLINE                   SAVE THE LINE NUMBER WHERE THE ERROR OCCURRED
1060   E4BD 35 04              PULS  B                         GET BACK THE ERROR NUMBER
1061   E4BF C1 06              CMPB  #3*2                      WAS IT AN OUT OF DATA ERROR?
1062   E4C1 26 04              BNE   LE4C7                     BRANCH IF NOT
1063   E4C3 DC 2B              LDD   BINVAL                    THE INPUT POINTER IS SAVED IN BINVAL BY THE READ COMMAND
1064   E4C5 DD A6              STD   CHARAD                    SAVE NEW ADDRESS FOR BASIC'S INPUT POINTER
1065   E4C7 1F 10      LE4C7   TFR   X,D                       SAVE THE ON ERROR DESTINATION LINE NUMBER IN ACCD
1066   E4C9 16 FF 7D           LBRA  LE449                     GO TRANSFER CONTROL TO THAT LINE NUMBER
1067
1068                   * ENHANCED BASIC'S ERROR CODES
1069   E4CC 48 52      BAS20ERR FCC  'HR'                      38  HIRES GRAHICS ERROR
1070   E4CE 48 50      LE4CE   FCC   'HP'                      39  HIRES TEXT ERROR
1071
1072                   * LINE INTO 'NEW' FROM $AD3F
1073   E4D0 34 06      ALINK19 PSHS  B,A                       SAVE THE CONTENTS OF ACCD
1074   E4D2 4F                 CLRA
1075   E4D3 5F                 CLRB
1076   E4D4 DD 2D              STD   OLDPTR                    RESET 'CONT' ADDRESS SO THAT YOU CAN'T CONTINUE
1077   E4D6 FD FE 0C           STD   H.ONBRK                   RESET THE ON BRK ADDRESS TO ZERO: NON-INITIALIZED
1078   E4D9 FD FE 0E           STD   H.ONERR                   RESET THE ON ERROR ADDRESS TO ZERO: NON-INITIALIZED
1079   E4DC FD FE 13           STD   H.ERLINE                  RESET THE ERLIN LINE NUMBER TO ZERO: NO ERROR
1080   E4DF 86 FF              LDA   #$FF                      INDICATES NO ERROR
1081   E4E1 B7 FE 10           STA   H.ERROR                   RESET ERROR NUMBER TO NO ERROR
1082   E4E4 35 06              PULS  A,B                       RESTORE ACCD
1083   E4E6 7E AD 43           JMP   LAD43                     JUMP TO THE END OF THE NEW COMMAND ($AD43)
1084
1085                   * ERNO
1086   E4E9 4F         ERNO    CLRA                            CLEAR THE MS BYTE OF ACCD
1087   E4EA F6 FE 10           LDB   H.ERROR                   GET THE ERROR NUMBER
1088   E4ED C1 FF              CMPB  #$FF                      IS IT A REAL ERROR
1089   E4EF 26 03              BNE   LE4F4                     BRANCH IF YES
1090   E4F1 1D                 SEX                             NOW ACCD = $FFFF IF NOT A REAL ERROR
1091   E4F2 20 06              BRA   LE4FA                     CONVERT ACCD TO FLOATING POINT
1092   E4F4 C1 F1      LE4F4   CMPB  #$F1                      CHECK FOR ERROR NUMBER $F1
1093   E4F6 26 01              BNE   LE4F9                     BRANCH IF NOT ERROR $F1
1094   E4F8 53                 COMB                            CONVERT TO 7*2 (UNDEFINED LINE NUMBER)
1095   E4F9 57         LE4F9   ASRB                            DIVIDE ERROR NUMBER BY 2
1096   E4FA 7E B4 F4   LE4FA   JMP   GIVABF                    CONVERT ACCD INTO A FLOATING POINT NUMBER
1097
1098                   * ERLIN
1099   E4FD FC FE 13   ERLIN   LDD   H.ERLINE                  GET THE LINE NUMBER WHERE THE ERROR OCCURRED
1100   E500 20 F8              BRA   LE4FA                     CONVERT IT INTO A FLOATING POINT NUMBER
1101
1102                   * BASIC'S MAIN LOOP IN THE DIRECT MODE PATCH ENTERED FROM $AC73
1103   E502 BD E0 19   ALINK21 JSR   SETTEXT                   SET UP HI-RES TEXT MODE IF ENABLED
1104   E505 BD B9 5C           JSR   LB95C                     SET UP VARIOUS PRINT PARAMETERS
1105   E508 1A 50              ORCC  #$50                      DISABLE IRQ, FIRQ
1106   E50A 86 34              LDA   #BLOCK6.4                 GET/PUT BUFFER BLOCK
1107   E50C B7 FF A0           STA   MMUREG                    PUT IT INTO LOGICAL BLOCK 0
1108   E50F CC FF FF           LDD   #$FFFF                    NO HGET/HPUT BUFFERS USED FLAG
1109   E512 DD 00              STD   0                         SET THE HGET/HPUT BUFFER SPACE TO SHOW NO BUFFERS IN USE
1110   E514 86 38              LDA   #BLOCK7.4                 GET NORMAL LOGICAL BLOCK 0
1111   E516 B7 FF A0           STA   MMUREG                    PUT BACK INTO THE LOGICAL ADDRESS SPACE
1112   E519 1C AF              ANDCC #$AF                      ENABLE IRQ, FIRQ
1113   E51B 7E AC 76           JMP   LAC76                     RE-ENTER THE MAIN STREAM CODE ($AC76)
1114
1115   E51E 7D FE 17   LE51E   TST   H.ERRBRK                  CHECK THE ERROR/BREAK FLAG
1116   E521 27 05              BEQ   LE528                     BRANCH IF ERROR BROUGHT US HERE
1117   E523 FC FE 15           LDD   H.ONBRKS                  GET THE ON BRK SOURCE LINE NUMBER IF BREAK VECTORED US HERE
1118   E526 20 03              BRA   LE52B
1119   E528 FC FE 11   LE528   LDD   H.ONERRS                  GET THE ON ERROR SOURCE LINE NUMBER
1120   E52B DD 68      LE52B   STD   CURLIN                    SAVE THE SOURCE LINE NUMBER AS THE CURRENT LINE NUMBER
1121   E52D C6 0E              LDB   #7*2                      UNDEFINED LINE NUMBER ERROR
1122   E52F 7E AC 49           JMP   LAC49                     JUMP TO THE ERROR SERVICING ROUTINE ($AC49)
1123
1124                   * INPUT PATCH ENTERED FROM $B03D
1125   E532 FC FE 0C   ALINK17 LDD   H.ONBRK                   GET THE ON BRK SOURCE LINE NUMBER
1126   E535 10 27 C8 D8        LBEQ  LAE11                     BRANCH IF ON BRK NOT INITIALIZED ($AE11)
1127   E539 34 06              PSHS  B,A                       SAVE THE ON BRK SOURCE ADDRESS
1128   E53B 86 01              LDA   #$01                      BREAK FLAG
1129   E53D B7 FE 17           STA   H.ERRBRK                  SET THE ERROR/BREAK FLAG TO BREAK
1130   E540 35 06              PULS  A,B                       RESTORE SOURCE ADDRESS - INEFFICIENT, LDD H.ONBRK IS BETTER
1131   E542 16 FF 04           LBRA  LE449
1132
1133                   * LPOKE
1134   E545 BD B1 41   LPOKE   JSR   LB141                     EVALUATE A NUMERIC EXPRESSION
1135   E548 10 21 1A B4        LBRN  RAMLINK                   ROM HOOK
1136   E54C 8D 40              BSR   LE58E                     CONVERT FPA0 INTO AN EXTENDED ADDRESS
1137   E54E C1 3F              CMPB  #BLOCK7.7                 HIGHEST POSSIBLE BLOCK NUMBER
1138   E550 10 22 CE F6        LBHI  ILLFUNC                   ILLEGAL FUNCTION CALL ERROR IF BLOCK NUMBER TOO BIG
1139   E554 34 14              PSHS  X,B                       SAVE REGISTERS
1140   E556 BD B2 6D           JSR   SYNCOMMA                  DO A SYNTAX CHECK FOR A COMMA
1141   E559 BD B7 0B           JSR   EVALEXPB                  EVALUATE EXPRESSION, RETURN VALUE IN ACCB
1142   E55C 1F 98              TFR   B,A                       SAVE THE BLOCK NUMBER IN ACCA
1143   E55E 35 14              PULS  B,X                       RESTORE REGISTERS
1144   E560 C1 3F              CMPB  #BLOCK7.7                 COMPARE TO HIGHEST POSSIBLE BLOCK NUMBER
1145   E562 10 22 CE E4        LBHI  ILLFUNC                   ILLEGAL FUNCTION CALL ERROR
1146   E566 1A 50              ORCC  #$50                      DISABLE INTERRUPTS
1147   E568 17 FB 36           LBSR  SELBLOK0                  PUT THE INTERPRETED BLOCK INTO LOGICAL BLOCK 0
1148   E56B A7 84              STA   ,X                        STORE THE VALUE BEING POKEd
1149   E56D 17 FB 27           LBSR  SETMMU                    RESTORE THE MMU REGISTERS TO WHAT BASIC EXPECTS
1150   E570 1C AF              ANDCC #$AF                      ENABLE THE IRQ AND FIRQ INTERRUPTS
1151   E572 39                 RTS
1152
```

```
1153                            * LPEEK
1154   E573 8D 19               LPEEK   BSR   LE58E                          CONVERT FPA0 INTO AN EXTENDED ADDRESS
1155   E575 10 21 1A 87                 LBRN  RAMLINK                        RAM HOOK
1156   E579 C1 3F                       CMPB  #BLOCK7.7                      COMPARE TO HIGHEST LEGAL BLOCK NUMBER
1157   E57B 10 22 CE CB                 LBHI  ILLFUNC                        ILLEGAL FUNCTINO CALL ERROR IF BLOCK NUMBER TOO BIG
1158   E57F 1A 50                       ORCC  #$50                           DISABLE INTERRUPTS
1159   E581 17 FB 1D                    LBSR  SELBLOK0                       GET THE INTERPRETED BLOCK NUMBER INTO CPU BLOCK 0
1160   E584 E6 84                       LDB   ,X                             GET THE VALUE BEING LPEEKed
1161   E586 17 FB 0E                    LBSR  SETMMU                         RESTORE THE MMU REGISTERS TO WHAT BASIC EXPECTS
1162   E589 1C AF                       ANDCC #$AF                           ENABLE THE IRQ AND FIRQ INTERRUPTS
1163   E58B 7E B4 F3                    JMP   LB4F3                          CONVERT THE VALUE IN ACCB INTO A FLOATING POINT NUMBER
1164
1165                            * CONVERT FPA0 INTO A 'LONG' ADDRESS
1166                            * THE 'LONG' ADDRESS WIL BE RETURNED IN TWO PIECES: THE LOW ORDER 13 BITS
1167                            * WILL BE IN THE X REGISTER, AND THE HIGH ORDER 6 BITS, WHICH ARE THE
1168                            * BLOCK NUMBER, WILL BE IN ACCB
1169   E58E 34 02               LE58E   PSHS  A
1170   E590 96 4F                       LDA   FP0EXP                         GET THE EXPONENT OF FPA0
1171   E592 81 93                       CMPA  #$93                           EXPONENT OF 512K-1
1172   E594 23 04                       BLS   LE59A                          BRANCH IF <= 512K-1
1173   E596 C6 40                       LDB   #BLOCK7.7+1                    MAKE IT ONE BLOCK BIGGER THAN THE BIGGEST ALLOWABLE
1174   E598 20 15                       BRA   LE5AF                          EXIT ROUTINE
1175   E59A BD BC C8            LE59A   JSR   LBCC8                          DE-NORMALIZE FPA0
1176   E59D DC 52                       LDD   FPA0+2                         GET THE TWO LEAST SIGNIFICANT BITS OF FPA0
1177   E59F 84 1F                       ANDA  #$1F                           MASK OFF THE 3 HIGH ORDER BITS
1178   E5A1 1F 01                       TFR   D,X                            SAVE THE 13 LOW ORDER BITS IN X REGISTER
1179   E5A3 DC 51                       LDD   FPA0+1                         GET THE SECOND AND THIRD BYTES IF FPA0
1180   E5A5 47                          ASRA
1181   E5A6 56                          RORB
1182   E5A7 47                          ASRA
1183   E5A8 56                          RORB
1184   E5A9 47                          ASRA
1185   E5AA 56                          RORB
1186   E5AB 47                          ASRA                                 NOT NECESSARY WITH MAXIMUM OF 512K RAM
1187   E5AC 56                          RORB
1188   E5AD 47                          ASRA                                 NOT NECESSARY WITH MAXIMUM OF 512K RAM
1189   E5AE 56                          RORB                                 SHIFT ACCD RIGHT 5 TIMES - THE BLOCK NUMBER IS IN ACCB
1190   E5AF 35 82               LE5AF   PULS  A,PC
1191
1192                            * BUTTON
1193   E5B1 BD B3 ED            BUTTON  JSR   INTCNV                         CONVERT FPA0 INTO AN INTEGER IN ACCB
1194   E5B4 10 21 1A 48                 LBRN  RAMLINK                        RAM HOOK
1195   E5B8 C1 03                       CMPB  #$03                           ONLY BUTTON NUMBERS 0-3 ALLOWD
1196   E5BA 10 22 CE 8C                 LBHI  ILLFUNC                        ILLEGAL FUNCTION ERROR
1197   E5BE 1F 98                       TFR   B,A                            SAVE BUTTON NUMBER IN ACCA
1198   E5C0 5F                          CLRB
1199   E5C1 53                          COMB                                 NOW ACCB = $FF
1200   E5C2 8E FF 00                    LDX   #PIA0                          POINT TO THE KEYBOARD STROBE PIO
1201   E5C5 E7 02                       STB   $02,X                          SET THE COLUMN STROBE TO $FF - ALLOW ONLY BUTTONS TO BE CHECKED
1202   E5C7 E6 84                       LDB   ,X                             READ THE KEYBOARD ROWS
1203   E5C9 C1 0F                       CMPB  #$0F                           THE BUTTONS ARE ON THE BOTTOM FOUR ROWS
1204   E5CB 27 1D                       BEQ   LE5EA                          BRANCH IF NO BUTTONS DOWN
1205   E5CD 30 8D 00 04                 LEAX  LE5D5,PC                       POINT TO THE BUTTON MASKING ROUTINES
1206   E5D1 48                          ALSA
1207   E5D2 48                          ALSA                                 MULT ACCA BY FOUR - FOUR BYTES/EACH MASKING ROUTINE
1208   E5D3 6E 86                       JMP   A,X                            JUMP TO THE APPROPRIATE MASKING ROUTINE
1209
1210                            * MASK OFF ALL BUT BUTTON 1, RIGHT JOYSTICK
1211   E5D5 C4 01               LE5D5   ANDB  #$01
1212   E5D7 20 0A                       BRA   LE5E3
1213
1214                            * MASK OFF ALL BUT BUTTON 1, LEFT JOYSTICK
1215   E5D9 C4 04               LE5D9   ANDB  #$04
1216   E5DB 20 06                       BRA   LE5E3
1217
1218                            * MASK OFF ALL BUT BUTTON 2, RIGHT JOYSTICK
1219   E5DD C4 02                       ANDB  #$02
1220   E5DF 20 02                       BRA   LE5E3
1221
1222                            * MASK OFF ALL BUT BUTTON 2, LEFT JOYSTICK
1223   E5E1 C4 08                       ANDB  #$08
1224   E5E3 26 05               LE5E3   BNE   LE5EA                          BRANCH IF MASKED BUTTON NOT DOWN
1225   E5E5 CC 00 01                    LDD   #1                             IF BUTTON DOWN, RETURN A VALUE OF ONE
1226   E5E8 20 02                       BRA   LE5EC
1227   E5EA 4F                  LE5EA   CLRA
1228   E5EB 5F                          CLRB                                 RETURN A ZERO IF BUTTON IS NOT DOWN
1229   E5EC BD B4 F4            LE5EC   JSR   GIVABF                         CONVERT ACCD INTO A FLOATING POINT NUMBER IN FPA0
1230   E5EF 39                          RTS
1231
1232                            * PALETTE
1233   E5F0 81 F7               PALETTE CMPA  #$F7                           'RGB' TOKEN?
1234   E5F2 10 21 1A 0A                 LBRN  RAMLINK                        RAM HOOK
1235   E5F6 26 08                       BNE   LE600                          NOT THE 'RGB' TOKEN, CHECK FOR 'CMP'
1236   E5F8 9D 9F                       JSR   GETNCH                         GET THE NEXT CHARACTER FROM BASIC'S INPUT LINE
1237                            * RGB ENTRY POINT - SET THE PALETTE REGISTERS FOR DEFAULT RGB VALUES
1238   E5FA 30 8D 00 66         LE5FA   LEAX  IM.RGB,PC                      POINT TO THE DEFAULT RGB PALETTE COLORS
1239   E5FE 20 34                       BRA   LE634                          PUT THE DATA POINTED TO BY X INTO THE PALETTE REGISTERS
1240   E600 81 F6               LE600   CMPA  #$F6                           'CMP' TOKEN?
1241   E602 26 08                       BNE   LE60C                          NO, GET A REGISTER NUMBER AND COLOR
1242   E604 9D 9F                       JSR   GETNCH                         GET THE NEXT CHARACTER FROM BASIC'S INPUT LINE
1243                            * CMP ENTRY POINT - SET THE PALETTE REGISTERS FOR DEFAULT CMP VALUES
1244   E606 30 8D 00 4A         LE606   LEAX  IM.CMP,PC                      POINT TO THE DEFAULT CMP PALETTE COLORS
1245   E60A 20 28                       BRA   LE634                          PUT THE DATA POINTED TO BY X INTO THE PALETTE REGISTERS
1246   E60C BD E7 B2            LE60C   JSR   LE7B2                          EVALUATE TWO EXPRESSIONS, NORMALLY A HORIZONTAL & VERTICAL COORDINATE
1247   E60F 8E FF B0                    LDX   #PALETREG                      POINT TO THE GIME CHIP'S PALETTE REGISTERS
1248   E612 10 8E E6 78                 LDY   #IM.PALET                      POINT TO THE RAM IMAGE OF THE PALETTE REGISTERS
```

```
1249   E616 96 2C                    LDA   BINVAL+1                        GET THE NUMBER OF THE PALETTE REGISTER TO CHANGE
1250   E618 81 10                    CMPA  #16                             16 PALETTE REGISTERS MAXIMUM
1251   E61A 10 24 CE 2C              LBCC  ILLFUNC                         ILLEGAL FUNCTION CALLERROR IF PALETTE REGISTER > 15
1252   E61E 30 86                    LEAX  A,X                             POINT TO THE SELECTED PALETTE REGISTER
1253   E620 31 A6                    LEAY  A,Y                             POINT TO THE SELECTED PALETTE REGISTER RAM IMAGE
1254   E622 D6 C0                    LDB   VERBEG+1                        GET THE NEW COLOR FOR THE PALETTE REGISTER
1255   E624 C1 3F                    CMPB  #63                             MAXIMUM OF 64 COLORS (ZERO IS A LEGIT COLOR)
1256   E626 23 02                    BLS   LE62A                           BRANCH IF LEGITIMATE COLOR SELECTED
1257   E628 C6 3F                    LDB   #63                             USE COLOR 63 IF BAD COLOR NUMBER SELECTED
1258   E62A 1A 50             LE62A  ORCC  #$50                            DISABLE INTERRUPTS
1259   E62C 13                       SYNC                                  WAIT FOR AN INTERRUPT TO CHANGE PALETTE REGISTERS - THIS WILL
1260                                                                       PREVENT THE SCREEN FROM FLASHING WHEN THE CHANGE IS MADE.
1261   E62D E7 84                    STB   ,X                             SAVE THE NEW COLOR IN THE PALETTE REGISTER
1262   E62F E7 A4                    STB   ,Y                             SAVE THE NEW COLOR IN THE PALETTE REGISTER RAM IMAGE
1263   E631 1C AF                    ANDCC #$AF                            ENABLE IRQ, FIRQ INTERRUPTS
1264   E633 39                       RTS
1265
1266   E634 34 10             LE634  PSHS  X                               SAVE THE SOURCE REGISTER POINTER
1267   E636 10 8E E6 78              LDY   #IM.PALET                       POINT TO THE PALETTE REGISTER RAM IMAGE
1268   E63A 8D 0C                    BSR   LE648                           COPY THE SOURCE PALETTE REGISTER TO THE RAM IMAGE
1269   E63C 35 10                    PULS  X                               RESTORE THE SOURCE REGISTER POINTER
1270   E63E 10 8E FF B0              LDY   #PALETREG                       POINT TO THE PALETTE REGISTERS
1271   E642 1A 50                    ORCC  #$50                            DIABLE INTERRUPTS
1272   E644 13                       SYNC                                  COPY IMMEDIATELY AFTER AN INTERRUPT TO PREVENT SPARKING
1273   E645 8D 01                    BSR   LE648                           COPY THE SOURCE REGISTER DATE INTO THE PALETTE REGISTERS
1274   E647 39                       RTS
1275
1276   E648 C6 0F             LE648  LDB   #16-1                           NUMBER OF BYTES TO COPY - BUG - SHOULD BE 16
1277   E64A A6 80             LE64A  LDA   ,X+                             GET A BYTE
1278   E64C A7 A0                    STA   ,Y+                             MOVE IT
1279   E64E 5A                       DECB                                  BUMP COUNTER DOWN ONE
1280   E64F 26 F9                    BNE   LE64A                           LOOP UNTIL DONE
1281   E651 1C AF                    ANDCC #$AF                            ENABLE IRQ, FIRQ INTERRUPTS
1282   E653 39                       RTS
1283
1284                         * PALETTE COLORS FOR A COMPOSITE MONITOR
1285   E654 12 24 0B 07 3F 1F IM.CMP FCB   18,36,11,7,63,31
1286   E65A 09 26 00 12 00 3F        FCB   9,38,0,18,0,63
1287   E660 00 12 00 26              FCB   0,18,0,38
1288
1289                         * PALETTE COLORS FOR AN RGB MONITOR
1290   E664 12 36 09 24 3F 1B IM.RGB  FCB   18,54,9,36,63,27
1291   E66A 2D 26 00 12 00 3F        FCB   45,38,0,18,0,63
1292   E670 00 12 00 26              FCB   0,18,0,38
1293
1294   E674 20 84             RGB    BRA   LE5FA
1295
1296   E676 20 8E             CMP    BRA   LE606
1297
1298                         * MASTER IMAGES USED TO PROGRAM THE CUSTOM CHIP'S PALETTE REGISTERS
1299   E678 12 24 0B 07 3F 1F IM.PALET FCB   18,36,11,7,63,31
1300   E67E 09 26 00 12 00 3F        FCB   9,38,0,18,0,63
1301   E684 00 12 00 26              FCB   0,18,0,38
1302
1303                         * HSCREEN
1304   E688 81 00             HSCREEN CMPA  #$00                           CHECK FOR END OF LINE
1305   E68A 10 21 19 72              LBRN  RAMLINK                         RAM HOOK
1306   E68E 26 03                    BNE   LE693                           BRANCH IF NOT END OF LINE
1307   E690 5F                       CLRB                                  IF END OF LINE, SET ARGUMENT TO ZERO
1308   E691 20 09                    BRA   LE69C                           SET THE HSCREEN MODE
1309   E693 BD B7 0B          LE693  JSR   EVALEXPB                        EVALUATE EXPRESSION, RETURN VALUE IN ACCB
1310   E696 C1 04                    CMPB  #$04                            ONLY 4 HSCREEN MODES ALLOWED
1311   E698 10 22 CD AE              LBHI  ILLFUNC                         ILLEGAL FUNCTION CALL ERROR
1312   E69C D7 E6             LE69C  STB   HRMODE                          SAVE THE HI-RES GRAPHICS MODE
1313   E69E C1 00                    CMPB  #$00                            CHECK FOR MODE 0
1314   E6A0 26 03                    BNE   LE6A5                           BRANCH IF NOT HSCREEN 0
1315   E6A2 7E E0 19                 JMP   SETTEXT                         SETUP THE VIDEO MODE REGISTERS FOR COCO  COMPATIBLE MODE
1316   E6A5 D7 E6             LE6A5  STB   HRMODE                          SAVE THE HI-RES GRAPHICS MODE
1317   E6A7 8E E6 CB                 LDX   #LE6CB                          POINT TO THE TABLE OF NUMBER OF BYTES/HORIZONTAL ROW
1318   E6AA C0 01                    SUBB  #$01                            CONVERT THE HI-RES MODE FROM 1-4 TO 0-3
1319   E6AC A6 85                    LDA   B,X                             GET THE NUMBER OF BYTES/HORIZONTAL ROW
1320   E6AE 97 B9                    STA   HORBYT                          AND SAVE IT
1321   E6B0 C1 01                    CMPB  #$01                            ONE OF THE FIRST TWO MODES?
1322   E6B2 2E 05                    BGT   LE6B9                           BRANCH IF NOT
1323   E6B4 CC 00 A0                 LDD   #160                            HORIZONTAL CENTER OF 320 COORDINATE SCREEN
1324   E6B7 20 03                    BRA   LE6BC
1325   E6B9 CC 01 40          LE6B9  LDD   #320                            HORIZONTAL CENTER OF 640 COORDINATE SCREEN
1326   E6BC DD C7             LE6BC  STD   HORDEF                          SAVE AS HORIZONTAL DEFAULT COORD
1327   E6BE CC 00 60                 LDD   #96                             VERTICAL CENTER COORDINATE
1328   E6C1 DD C9                    STD   VERDEF                          SAVE AS VERTICAL DEFAULT
1329   E6C3 F6 FE 0B                 LDB   H.BCOLOR                        GET THE BACKGROUND COLOR
1330   E6C6 8D 10                    BSR   CLRHIRES                        CLEAR THE HI-RES GRAPHICS SCREEN TO THE BACKGROUND COLOR
1331   E6C8 7E E0 4D                 JMP   SETGRAPH                        GROGRAM THE VIDEO RESOLUTION MODE
1332
1333                         * TABLE OF THE NUMBER OF BYTES PER HORIZONTAL ROW FOR EACH HSCREEN MODE
1334   E6CB 50 A0 50 A0       LE6CB  FCB   80,160,80,160
1335
1336                         * HCLS
1337   E6CF 26 05             HCLS   BNE   LE6D6                           BRANCH IF NOT END OF LINE
1338   E6D1 F6 FE 0B                 LDB   H.BCOLOR                        GET THE BACKGROUND COLOR
1339   E6D4 20 02                    BRA   CLRHIRES                        CLEAR THE SCREEN TO THE BACKGROUND COLOR
1340   E6D6 8D 36             LE6D6  BSR   LE70E                           EVALUATE AN EXPRESSION, SYNTAX CHECK FOR NOT > 16
1341
1342                         * CLEAR THE HI-RES GRAPHICS SCREEN TO THE COLOR IN ACCB
1343   E6D8 0D E6             CLRHIRES TST  HRMODE                         CHECK THE HI-RES MODE
1344   E6DA 27 13                    BEQ   LE6EF                           HR' ERROR IF IN THE 32 COLUMN MODE
```

```
1345   E6DC 8D 64                      BSR   PIXELFIL                        FILL ACCB WITH THE SELECTED COLOR
1346   E6DE BD E1 19                   JSR   SELTASK1                        SELECT TASK REGISTER 1 AS THE ACTIVE TASK REGISTER
1347                          * FILL MEMORY FROM HRESSCRN TO $A000 WITH ACCB; THIS IS THE HI-RES GRAPHICS SCREEN
1348   E6E1 8E 20 00                   LDX   #HRESSCRN                       POINT TO START OF HI-RES GRAPHICS SCREEN
1349   E6E4 E7 80              LE6E4   STB   ,X+                             'CLEAR' A BYTE
1350   E6E6 8C A0 00                   CMPX  #BASIC                          CHECK FOR END OF THE HI-RES GRAPHICS SCREEN
1351   E6E9 26 F9                      BNE   LE6E4                           KEEP 'CLEARING' UNTIL DONE
1352   E6EB BD E0 FF                   JSR   SELTASK0                        SET TASK REGISTER 0 AS THE ACTIVE TASK REGISTER
1353   E6EE 39                         RTS
1354   E6EF C6 4C              LE6EF   LDB   #38*2                           'HR' ERROR
1355   E6F1 7E AC 46                   JMP   LAC46                           JUMP TO THE ERROR HANDLER
1356
1357                          * HCOLOR
1358   E6F4 81 2C              HCOLOR  CMPA  #','                            CHECK FOR COMMA, FIRST ARGUMENT NOT GIVEN
1359   E6F6 10 21 19 06                LBRN  RAMLINK                         RAM HOOK
1360   E6FA 27 09                      BEQ   LE705                           BRANCH IF FIRST ARGUMENT NOT GIVEN
1361   E6FC 8D 10                      BSR   LE70E                           EVALUATE EXPRESSION, SYNTAX CHECK FOR EXPRESSION > 16
1362   E6FE F7 FE 0A                   STB   H.FCOLOR                        SAVE THE NEW FORGROUND COLOR
1363   E701 9D A5                      JSR   GETCCH                          GET BASIC'S CURRENT INPUT CHARACTER
1364   E703 27 08                      BEQ   LE70D                           BRANCH IF END OF LINE, NO BACKGROUND COLOR GIVEN
1365   E705 BD B2 6D           LE705   JSR   SYNCOMMA                        DO A SYNTAX CHECK FOR A COMMA
1366   E708 8D 04                      BSR   LE70E                           EVALUATE EXPRESSION, SYNTAX CHECK FOR EXPRESSION > 16
1367   E70A F7 FE 0B                   STB   H.BCOLOR                        SAVE THE NEW BACKGROUND COLOR
1368   E70D 39                 LE70D   RTS
1369
1370   E70E BD B7 0B           LE70E   JSR   EVALEXPB                        EVALUATE EXPRESSION, RETURN VALUE IN ACCB
1371   E711 C1 10              LE711   CMPB  #16                             MAXIMUM OF 16 DIFFERENT COLORS
1372   E713 10 24 CD 33                LBCC  ILLFUNC                         ILLEGAL FUNCTION CALL ERROR
1373   E717 39                         RTS
1374
1375   E718 BD E7 31           LE718   JSR   LE731                          SET THE WORKING COLOR AND ALL PIXEL BYTES TO DEFAULT VALUES
1376   E71B 9D A5                      JSR   GETCCH                          GET BASIC'S CURRENT INPUT CHARACTER
1377   E71D 27 10                      BEQ   LE72F                           BRANCH IF END OF LINE
1378   E71F 81 29                      CMPA  #')'                            SYNTAX CHECK FOR ')'
1379   E721 27 0C                      BEQ   LE72F                           EXIT IF ')'
1380   E723 BD B2 6D                   JSR   SYNCOMMA                        DO A SYNTAX CHECK FOR A COMMA
1381   E726 81 2C                      CMPA  #','                            SYNTAX CHECK FOR A COMMA
1382   E728 27 05                      BEQ   LE72F                           USE DEFAULT COLORS IF TWO COMMAS
1383   E72A BD E7 0E                   JSR   LE70E                           EVALUATE COLOR ARGUMENT
1384   E72D 8D 0C                      BSR   LE73B                           SET THE WORKING AND ALL COLOR BYTES TO THE COLOR ARGUMENT
1385   E72F 0E A5              LE72F   JMP   GETCCH                          GET BASIC'S CURRENT INPUT CHARACTER AND RETURN
1386
1387   E731 F6 FE 0A           LE731   LDB   H.FCOLOR                        GET THE FOREGOUND COLOR
1388   E734 0D C2                      TST   SETFLG                          TEST THE HSET/HRESET FLAG
1389   E736 26 03                      BNE   LE73B                           BRANCH IF HSET
1390   E738 F6 FE 0B                   LDB   H.BCOLOR                        GET THE BACKGROUND COLOR IF HRESET
1391   E73B D7 B4              LE73B   STB   WCOLOR                          SAVE THE NEW WORKING COLOR
1392   E73D 8D 03                      BSR   PIXELFIL                        FILL ALL PIXELS IN A BYTE WITH THE WORKING COLOR
1393   E73F D7 B5                      STB   ALLCOL                          SAVE THE FILLED WITH WORKING COLOR BYTE
1394   E741 39                         RTS
1395
1396                          * FILL ACCB WITH PIXELS OF THE COLOR CONTAINED IN ACCB
1397   E742 34 10              PIXELFIL PSHS  X
1398   E744 96 E6                      LDA   HRMODE                          GET THE HI-RES GRAPHICS MODE
1399   E746 80 01                      SUBA  #$01                            CONVERT 1-4 TO 0-3
1400   E748 8E E7 59                   LDX   #LE759                          POINT TO THE TABLE OF PIXEL MASKS
1401   E74B E4 86                      ANDB  A,X                             KEEP ONLY ONE PIXEL'S WORTH OF COLOR INFORMATION
1402   E74D 96 E6                      LDA   HRMODE                          * BOTH OF THESE INSTRUCTIONS
1403   E74F 80 01                      SUBA  #$01                            * ARE SUPERFLUOUS
1404   E751 8E E7 5D                   LDX   #LE75D                          POINT TO THE TABLE OF MULTIPLIERS
1405   E754 A6 86                      LDA   A,X                             GET THE APPROPRIATE MULTIPLIER
1406   E756 3D                         MUL                                   NOW THE COLOR INFORMATION IS IN EVERY PIXEL IN THE BYTE
1407   E757 35 90                      PULS  X,PC
1408
1409                          * PIXEL MASKS FOR THE HI-RES GRAPHICS MODES
1410   E759 03 0F 01 03        LE759   FCB   $03,$0F,$01,$03
1411
1412                          * MULTIPLIERS TO SPREAD HI-RES PIXELS THROUGH AN ENTIRE BYTE
1413   E75D 55 11 FF 55        LE75D   FCB   $55,$11,$FF,$55
1414
1415                          * HSET
1416   E761 86 01              HSET    LDA   #$01                            HSET FLAG
1417   E763 20 05                      BRA   LE76A
1418
1419                          * HRESET
1420   E765 4F                 HRESET  CLRA                                  HRESET FLAG
1421   E766 10 21 18 96                LBRN  RAMLINK
1422   E76A 0D E6              LE76A   TST   HRMODE                          IS THE HI-RES GRAPHICS MODE ENABLED?
1423   E76C 27 81                      BEQ   LE6EF                           HR' ERROR IF HI-RES MODE NOT ENABLED
1424   E76E 97 C2                      STA   SETFLG                          SAVE THE HSET/HRESET FLAG
1425   E770 BD B2 6A                   JSR   LB26A                           SYNTAX CHECK FOR '('
1426   E773 BD E7 AA                   JSR   LE7AA                           EVALUATE TWO EXPRESSIONS
1427   E776 0D C2                      TST   SETFLG                          CHECK THE HSER/HRESET FLAG
1428   E778 26 05                      BNE   LE77F                           BRANCH IF HSET
1429   E77A BD E7 31                   JSR   LE731                           SET THE WORKING COLOR AND ALL PIXEL BYTE
1430   E77D 20 03                      BRA   LE782
1431   E77F BD E7 18           LE77F   JSR   LE718                          GET THE HSET COLOR
1432   E782 BD B2 67           LE782   JSR   LB267                          SYNTAX CHECK FOR ')'
1433   E785 BD E7 DA                   JSR   HCALPOS                         LOAD X WITH PIXEL BYTE ADDRESS; ACCA WITH PIXEL MASK
1434   E788 BD E1 19           LE788   JSR   SELTASK1                        MAKE TASK REGISTER 1 THE ACTIVE TASK REGISTER
1435   E78B BD E7 92                   JSR   LE792                           SET OR RESET A PIXEL
1436   E78E BD E0 FF                   JSR   SELTASK0                        RESET TASK REGISTER 0 TO BE THE ACTIVE TASK REGISTER
1437   E791 39                         RTS
1438
1439                          * HSET/HRESET A PIXEL; ENTER W/X POINTING TO THE BYTE CONTAINING THE PIXEL AND
1440                          * ACCA POINTING TO THE MASK FOR THE PROPER PIXEL
```

```
1441   E792 E6 84             LE792    LDB   ,X                              GET THE BYTE WHICH CONTAINS THE PIXEL
1442   E794 34 Ø4                      PSHS  B                               AND SAVE IT ON THE STACK
1443   E796 1F 89                      TFR   A,B                             COPY THE MASK TO ACCB
1444   E798 43                         COMA                                  INVERT THE MASK
1445   E799 A4 84                      ANDA  ,X                              ERASE OLD PIXEL DATA
1446   E79B D4 B5                      ANDB  ALLCOL                          FORCE THE PIXEL MASK TO BE THE CORRECT COLOR
1447   E79D 34 Ø4                      PSHS  B                               AND SAVE THE 'COLORED' DATA ON THE STACK
1448   E79F AA EØ                      ORA   ,S+                             REPLACE THE 'ERASED' PIXEL WITH THE NEW COLOR DATA
1449   E7A1 A7 84                      STA   ,X                              AND SAVE IT IN THE SCREEN MEMORY
1450   E7A3 A0 EØ                      SUBA  ,S+                             ACCA=Ø IF OLD AND NEW PIXELS WERE IDENTICAL
1451   E7A5 9A DB                      ORA   CHGFLG                          SET CHGFLG <> Ø IF THE PIXEL WAS CHANGED
1452   E7A7 97 DB                      STA   CHGFLG                          SAVE THE 'CHANGED' STATUS
1453   E7A9 39                         RTS
1454   E7AA BD E7 B2          LE7AA    JSR   LE7B2                           EVALUATE TWO EXPRESSIONS
1455   E7AD CE ØØ BD          LE7AD    LDU   #HORBEG                         POINT U TO EVALUATED COORDINATES' STORAGE LOCATIONS
1456                          * THE 'NORMALIZATION' ($932Ø) ROUTINE FROM EXTENDED BASIC WENT HERE - IT IS NOT NEEDED
1457                          * IN ENHANCED BASIC SO IT WAS REPLACED WITH AN RTS.
1458   E7BØ 39                LE7BØ    RTS
1459   E7B1 39                         RTS                                   WASTED BYTE
1460
1461                          * EVALUATE TWO EXPRESSIONS - NORMALLY A HORIZONTAL AND VERTICAL COORDINATE
1462                          * PERFORM COORDINATE SYNTAX RANGE CHECKS ON THE EXPRESSIONS
1463   E7B2 BD B7 34          LE7B2    JSR   LB734                           EVALUATE TWO EXPRESSIONS; RETURN 1ST VALUE IN BINVAL, SECOND IN ACCB
1464   E7B5 1Ø 8E ØØ BD               LDY   #HORBEG                         POINT TO THE COORDINATE STORAGE VARIABLES
1465   E7B9 C1 CØ             LE7B9    CMPB  #192                            CHECK FOR MAXIMUM VERTICAL COORDINATE
1466   E7BB 25 Ø2                      BCS   LE7BF                           BRANCH IF WITHIN RANGE
1467   E7BD C6 BF                      LDB   #192-1                          FORCE TO MAXIMUM VALUE IF OUT OF RANGE
1468   E7BF 4F                LE7BF    CLRA                                  CLEAR THE MOST SIGNIFICANT BYTE OF ACCD
1469   E7CØ ED 22                      STD   $Ø2,Y                           SAVE THE VERTICAL COORDINATE
1470   E7C2 96 E6                      LDA   HRMODE                          GET THE HI-RES GRAPHICS MODE
1471   E7C4 81 Ø2                      CMPA  #$Ø2                            IS MAXIMUM PIXEL WIDTH=32Ø?
1472   E7C6 2E Ø5                      BGT   LE7CD                           NO
1473   E7C8 CC Ø1 3F                   LDD   #32Ø-1                          LOAD ACCD WITH MAXIMUM HORIZONTAL COORDINATE FORE 32Ø PIXEL WIDE
1474   E7CB 2Ø Ø3                      BRA   LE7DØ                           DO THE HORIZONTAL RANGE CHECK
1475   E7CD CC Ø2 7F          LE7CD    LDD   #64Ø-1                          LOAD ACCD WITH MAXIMUM HORIZONTAL COORDINATE FORE 64Ø PIXEL WIDE
1476   E7DØ 1Ø 93 2B          LE7DØ    CMPD  BINVAL                          IS THE HORIZONTAL COORDINATE > MAXIMUM VALUE?
1477   E7D3 25 Ø2                      BCS   LE7D7                           YES, USE THE MAXIMUM HORIZONTAL COORDINATE
1478   E7D5 DC 2B                      LDD   BINVAL                          GET THE NEW HORIZONTAL COORDINATE
1479   E7D7 ED A4             LE7D7    STD   ,Y                              SAVE THE HORIZONTAL COORDINATE
1480   E7D9 39                         RTS
1481
1482                          * THIS ROUTINE WILL CONVERT THE X,Y COORDINATES OF A PIXEL INTO THE SCREEN ADDRESS (X REG) AND
1483                          * PIXEL OFFSET (ACCA) OF THE BYTE ON THE SCREEN CONTAINING THE PIXEL.
1484   E7DA 8D ØA             HCALPOS  BSR   LE7E6                           POINT U TO THE HCALPOS SUBROUTINE FOR THE CURRENT HRMODE
1485   E7DC 6E C4                      JMP   ,U                              EXECUTE THE HCALPOS SUBROUTINE
1486
1487                          * CALTABLE
1488   E7DE E8 2Ø E8 3F E7 FF CALTABLE FDB   G2BITBIX,G4BITPIX,G1BITPIX
1489   E7E4 E8 2Ø                      FDB   G2BITBIX
1490
1491                          * POINT U TO THE PROPER CALPOS SUBROUTINE
1492   E7E6 CE E7 DE          LE7E6    LDU   #CALTABLE                       POINT U TO THE CALPOS ADDRESS TABLE
1493   E7E9 96 E6                      LDA   HRMODE                          GET THE HI-RS GRAPHICS MODE
1494   E7EB 8Ø Ø1                      SUBA  #$Ø1                            (DECA WOULD DO) CONVERT FROM 1-4 TO Ø-3
1495   E7ED 48                         ALSA                                  X2 BYTES PER ADDRESS
1496   E7EE EE C6                      LDU   A,U                             GET THE APPROPRIATE CALPOS ADDRESS FROM THE TABLE
1497   E7FØ 39                         RTS
1498
1499                          * TABLE OF 1 BIT PIXEL MASKS
1500   E7F1 8Ø 4Ø 2Ø 1Ø Ø8 Ø4 PIX1MASK FCB   $8Ø,$4Ø,$2Ø,$1Ø,$Ø8,$Ø4
1501   E7F7 Ø2 Ø1                      FCB   $Ø2,$Ø1
1502
1503                          * TABLE OF 2 BIT PIXEL MASKS
1504   E7F9 CØ 3Ø ØC Ø3       PIX2MASK FCB   $CØ,$3Ø,$ØC,$Ø3
1505
1506                          * TABLE OF 4 BIT PIXEL MASKS
1507   E7FD FØ ØF             PIX4MASK FCB   $FØ,$ØF
1508
1509                          **************
1510
1511                          * CONVERT HORIZONTAL, VERTICAL COORDINATES INTO THE ADDRESS (X) FOR THE BYTE WHICH CONTAINS THE DESIRED
1512                          * PIXEL AND A MASK (ACCA) WHICH HAS ONLY THOSE BITS WHICH CORRESPOND TO THE DESIRED PIXEL
1513
1514   E7FF 34 44             G1BITPIX PSHS  U,B                             SAVE REGISTERS
1515   E8Ø1 D6 B9                      LDB   HORBYT                          GET THE NUMBER OF BYTES PER HORIZONTAL ROW
1516   E8Ø3 96 CØ                      LDA   VERBEG+1                        GET THE VERTICAL COORDINATE
1517   E8Ø5 3D                         MUL                                   NOW ACCD CONTAINS THE ROW OFFSET IN BYTES FROM THE TOP OF SCREEN
1518   E8Ø6 C3 2Ø ØØ                   ADDD  #HRESSCRN                       ADD THE ROW OFFSET TO THE START OF THE SCREEN
1519   E8Ø9 1F Ø1                      TFR   D,X                             X CONTAINS THE ADDRESS OF THE START OF THE ROW CONTAINING A PIXEL
1520   E8ØB DC BD                      LDD   HORBEG                          GET THE HORIZONTAL COORDINATE
1521   E8ØD 44                         LSRA
1522   E8ØE 56                         RORB
1523   E8ØF 44                         LSRA
1524   E81Ø 56                         RORB
1525   E811 44                         LSRA                                  * DIVIDE HORIZONTAL COORDINATE BY EIGHT - THERE ARE 8 PIXELS PER BYTE
1526   E812 56                         RORB                                  * ACCD CONTAINS THE COLUMN OFFSET TO THE PIXEL IN BYTES
1527   E813 3Ø 8B                      LEAX  D,X                             ADD THE COLUMN OFFSET - X POINTS TO THE BYTE CONTAINING THE PIXEL
1528   E815 96 BE                      LDA   HORBEG+1                        GET THE LEAST SIGNIFICANT BYTE OF THE HORIZONTAL COORDINATE
1529   E817 84 Ø7                      ANDA  #$Ø7                            KEEP BITS Ø-2 WHICH ARE THE PIXEL POSITION IN THE BYTE
1530   E819 CE E7 F1                   LDU   #PIX1MASK                       POINT TO THE TABLE OF TWO COLOR PIXEL MASKS
1531   E81C A6 C6                      LDA   A,U                             GET THE CORRECT PIXEL MASK
1532   E81E 35 C4                      PULS  B,U,PC                          RESTORE THE REGISTERS
1533
1534   E82Ø 34 44             G2BITBIX PSHS  U,B                             SAVE REGISTERS
1535   E822 D6 B9                      LDB   HORBYT                          GET THE NUMBER OF BYTES/ROW
1536   E824 96 CØ                      LDA   VERBEG+1                        GET THE VERTICAL COORDINATE
```

```
1537   E826 3D                     MUL                     NOW ACCD CONTAINS THE ROW OFFSET IN BYTES FROM THE TOP OF SCREEN
1538   E827 C3 20 00               ADDD  #HRESSCRN         ADD THE ROW OFFSET TO THE START OF THE SCREEN
1539   E82A 1F 01                  TFR   D,X               X CONTAINS THE ADDRESS OF THE START OF THE ROW CONTAINING A PIXEL
1540   E82C DC BD                  LDD   HORBEG            GET THE HORIZONTAL COORDINATE
1541   E82E 44                     LSRA
1542   E82F 56                     RORB
1543   E830 44                     LSRA                    * DIVIDE HORIZONTAL COORDINATE BY FOUR - THERE ARE 4 PIXELS PER BYTE
1544   E831 56                     RORB                    * ACCD CONTAINS THE COLUMN OFFSET TO THE PIXEL IN BYTES
1545   E832 30 8B                  LEAX  D,X               ADD THE COLUMN OFFSET - X POINTS TO THE BYTE CONTAINING THE PIXEL
1546   E834 96 BE                  LDA   HORBEG+1          GET THE LEAST SIGNIFICANT BYTE OF THE HORIZONTAL COORDINATE
1547   E836 84 03                  ANDA  #$03              KEEP BITS 0,1 WHICH ARE THE PIXEL POSITION IN THE BYTE
1548   E838 CE E7 F9               LDU   #PIX2MASK         POINT TO THE TABLE OF FOUR COLOR PIXEL MASKS
1549   E83B A6 C6                  LDA   A,U               GET THE CORRECT PIXEL MASK
1550   E83D 35 C4                  PULS  B,U,PC            RESTORE THE REGISTERS
1551
1552   E83F 34 44           G4BITPIX PSHS  U,B               SAVE REGISTERS
1553   E841 D6 B9                  LDB   HORBYT            GET THE NUMBER OF BYTES/ROW
1554   E843 96 C0                  LDA   VERBEG+1          GET THE VERTICAL COORDINATE
1555   E845 3D                     MUL                     NOW ACCD CONTAINS THE ROW OFFSET IN BYTES FROM THE TOP OF SCREEN
1556   E846 C3 20 00               ADDD  #HRESSCRN         ADD THE ROW OFFSET TO THE START OF THE SCREEN
1557   E849 1F 01                  TFR   D,X               X CONTAINS THE ADDRESS OF THE START OF THE ROW CONTAINING A PIXEL
1558   E84B DC BD                  LDD   HORBEG            GET THE HORIZONTAL COORDINATE
1559   E84D 44                     LSRA                    * DIVIDE HORIZONTAL COORDINATE BY TWO - THERE ARE 2 PIXELS PER BYTE
1560   E84E 56                     RORB                    * ACCD CONTAINS THE COLUMN OFFSET TO THE PIXEL IN BYTES
1561   E84F 30 8B                  LEAX  D,X               ADD THE COLUMN OFFSET - X POINTS TO THE BYTE CONTAINING THE PIXEL
1562   E851 96 BE                  LDA   HORBEG+1          GET THE LEAST SIGNIFICANT BYTE OF THE HORIZONTAL COORDINATE
1563   E853 84 01                  ANDA  #$01              KEEP BITS 0 WHICH IS THE PIXEL POSITION IN THE BYTE
1564   E855 CE E7 FD               LDU   #PIX4MASK         POINT TO THE TABLE OF 16 COLOR PIXEL MASKS
1565   E858 A6 C6                  LDA   A,U               GET THE CORRECT PIXEL MASK
1566   E85A 35 C4                  PULS  B,U,PC            RESTORE THE REGISTERS
1567
1568                       * HPOINT
1569   E85C 0D E6          HPOINT  TST   HRMODE             CHECK FOR HI-RES GRAPHICS MODE
1570   E85E 10 27 FE 8D            LBEQ  LE6EF             'HR' ERROR IF NOT GRAPHICS
1571   E862 BD B2 6A               JSR   LB26A             SYNTAX CHECK FOR '('
1572   E865 BD E7 AA               JSR   LE7AA             EVALUATE TWO EXPRESSIONS (X,Y COORDS)
1573   E868 BD B2 67               JSR   LB267             SYNTAX CHECK FOR ')'
1574   E86B BD E1 19               JSR   SELTASK1          SELECT TASK REGSTER 1
1575   E86E BD E7 DA               JSR   HCALPOS           POINT X TO PIXEL, ACCA CONTAINS MASK
1576   E871 1F 89                  TFR   A,B               PUT MASK IN ACCB
1577   E873 E4 84                  ANDB  ,X                MASK OFF ALL BUT DESIRED PIXEL
1578   E875 44             LE875   LSRA                    SHIFT MASK TO THE RIGHT
1579   E876 25 03                  BCS   LE87B             STOP SHIFTING IF DATA IS RIGHT JUSTIFIED
1580   E878 54                     LSRB                    SHIFT PIXEL TO THE RIGHT
1581   E879 20 FA                  BRA   LE875             KEEP SHIFTING UNTIL DATA IS RIGHT JUSTIFIED
1582   E87B BD B4 F3       LE87B   JSR   LB4F3             CONVERT ACCB INTO A FLOATING POINT NUMBER
1583   E87E BD E0 FF               JSR   SELTASK0          SELECT TASK REGISTER 0
1584   E881 39                     RTS
1585
1586                       * HLINE
1587   E882 0D E6          HLINE   TST   HRMODE             CHECK HI-RES GRAPHICS MODE
1588   E884 10 27 FE 67            LBEQ  LE6EF             'HR' ERROR IF NOT GRAPHICS
1589   E888 10 21 17 74            LBRN  RAMLINK           RAM HOOK
1590   E88C 81 28                  CMPA  #'('              CHECK FOR '('
1591   E88E 27 09                  BEQ   LE899             GO LOOK FOR START AND END POINTS
1592   E890 81 AC                  CMPA  #$AC              CHECK FOR MINUS SIGN TOKEN
1593   E892 27 05                  BEQ   LE899             BRANCH IF NO STARTING POINTS GIVEN
1594   E894 C6 40                  LDB   #'@'              CHECK FOR '@' SIGN
1595   E896 BD B2 6F               JSR   LB26F             GO DO A SYNTAX CHECK
1596   E899 BD E9 E1       LE899   JSR   LE9E1             GET STARTING AND ENDING COORDINATES
1597   E89C 9E C3                  LDX   HOREND            GET ENDING HORIZONTAL COORDINATE
1598   E89E 9F C7                  STX   HORDEF            PUT IN LAST USED HORIZONTAL END POINT
1599   E8A0 9E C5                  LDX   VEREND            GET ENDING VERTICAL COORDINATE
1600   E8A2 9F C9                  STX   VERDEF            PUT IN LAST USED VERTICAL END POINT
1601   E8A4 BD B2 6D               JSR   SYNCOMMA          DO A SYNTAX CHECK FOR A COMMA
1602   E8A7 81 BE                  CMPA  #$BE              PRESET TOKEN?
1603   E8A9 27 09                  BEQ   LE8B4             BRANCK IF YES
1604   E8AB 81 BD                  CMPA  #$BD              PSET TOKEN?
1605   E8AD 10 26 C9 C6            LBNE  LB277             'SYNTAX' ERROR IF NOT PSET OR PRESET
1606   E8B1 C6 01                  LDB   #$01              PSET FLAG
1607   E8B3 86             LE8B3   FCB   SKP1LD            OP CODE FOR LDA #; EFFECTIVELY SKIP NEXT INSTRUCTION
1608   E8B4 5F             LE8B4   CLRB                    PRESET FLAG
1609   E8B5 34 04                  PSHS  B                 SAVE PSET/PRESET FLAG
1610   E8B7 9D 9F                  JSR   GETNCH            GET NEXT CHARACTER FROM BASIC'S INPUT LINE
1611   E8B9 BD EA 0D               JSR   LEA0D             NORMALIZE START/END COORDS
1612   E8BC 35 04                  PULS  B                 GET PSET/PRESET FLAG
1613   E8BE D7 C2                  STB   SETFLG            SAVE IT
1614   E8C0 BD E7 31               JSR   LE731             SET ACTIVE COLOR BYTE
1615   E8C3 9D A5                  JSR   GETCCH            GET BASIC'S CURRENT INPUT CHARACTER
1616   E8C5 10 27 00 85            LBEQ  LE94E             BRANCH IF NO BOX TO BE DRAWN
1617   E8C9 BD B2 6D               JSR   SYNCOMMA          DO A SYNTAX CHECK FOR A COMMA
1618   E8CC C6 42                  LDB   #'B'              DRAW A BOX?
1619   E8CE BD B2 6F               JSR   LB26F             GO DO A SYNTAX CHECK FOR A 'B'
1620   E8D1 26 18                  BNE   LE8EB             FOUND A 'B' AND SOMETHING FOLLOWS
1621   E8D3 8D 31                  BSR   LE906             DRAW A HORIZONTAL LINE
1622   E8D5 8D 5A                  BSR   LE931             DRAW A VERTICAL LINE
1623   E8D7 9E BD                  LDX   HORBEG            GET HORIZONTAL START COORD
1624   E8D9 34 10                  PSHS  X                 SAVE IT ON THE STACK
1625   E8DB 9E C3                  LDX   HOREND            GET HORIZONTAL END COORDINATE
1626   E8DD 9F BD                  STX   HORBEG            PUT IN HORIZONTAL START COORDINATE
1627   E8DF 8D 50                  BSR   LE931             DRAW A VERTICAL LINE
1628   E8E1 35 10                  PULS  X                 GET THE PREVIOUS HORIZONTAL START COORDINATE
1629   E8E3 9F BD                  STX   HORBEG            RESTORE IT
1630   E8E5 9E C5                  LDX   VEREND            GET VERTICAL END COORDINATE
1631   E8E7 9F BF                  STX   VERBEG            PUT INTO START COORD
1632   E8E9 20 1B                  BRA   LE906             DRAW A HORIZONTAL LINE
```

```
1633   E8EB C6 46            LE8EB    LDB   #'F'                          CHECK FOR FILL OPTION
1634   E8ED BD B2 6F                  JSR   LB26F                         GO DO A SYNTAX CHECK FOR AN 'F'
1635   E8F0 20 04                     BRA   LE8F6                         GO 'FILL' THE BOX
1636   E8F2 30 1F            LE8F2    LEAX  $-01,X                        MOVE VERTICAL COORD UP ONE
1637   E8F4 9F BF            LE8F4    STX   VERBEG                        SAVE THE NEW VERTICAL START COORDINATE
1638                        * DRAW A SERIES OF HORIZONTAL LINES FROM VERTICAL START TO VERTICAL END
1639   E8F6 BD E9 06         LE8F6    JSR   LE906                         DRAW A HORIZONTAL LINE
1640   E8F9 9E BF                     LDX   VERBEG                        GET START VERTICAL COORD
1641   E8FB 9C C5                     CMPX  VEREND                        COMPARE TO END VERTICAL COORD
1642   E8FD 27 06                     BEQ   LE905                         RETURN IF EQUAL
1643   E8FF 24 F1                     BCC   LE8F2                         BRANCH IF START HORIZONTAL > END HORIZONTAL
1644   E901 30 01                     LEAX  $01,X                         MOVE HORIZONTAL COORD DOWN ONE
1645   E903 20 EF                     BRA   LE8F4                         KEEP DRAWING LINES
1646   E905 39               LE905    RTS
1647                        * DRAW A HORIZONTAL LINE FROM HOREND TO HORBEG AT VERTICAL COORD VERBEG; COLOR IN ALLCOL
1648   E906 9E BD            LE906    LDX   HORBEG                         GET STARTING COORDINATES
1649   E908 34 10                     PSHS  X                             SAVE 'EM
1650   E90A BD E9 DB                  JSR   LE9DB                         GET ABSOLUTE VALUE OF HOREND-HORBEG (HORIZONTAL COORD)
1651   E90D 24 04                     BCC   LE913                         BRANCH IF END > START
1652   E90F 9E C3                     LDX   HOREND                        GET END COORD
1653   E911 9F BD                     STX   HORBEG                        MAKE IT THE START COORD
1654   E913 1F 02            LE913    TFR   D,Y                           SAVE DIFFERENCE IN Y
1655   E915 31 21                     LEAY  $01,Y                         ADD ONE TO DIFFERENCE - TURN ON STARTING AND ENDING COORDS
1656   E917 BD E7 DA                  JSR   HCALPOS                       GET ABSOLUTE SCREEN ADDRESS IN X AND PIXEL MASK IN ACCA
1657   E91A 35 40                     PULS  U                             GET START COORDS
1658   E91C DF BD                     STU   HORBEG                        RESTORE THEM
1659   E91E 17 00 F5                  LBSR  LEA16                         POINT U TO ROUTINE TO MOVE PIXEL POINTERS TO RIGHT
1660   E921 97 D7            LE921    STA   VD7                           SAVEL PIXEL MASK
1661   E923 BD E7 88                  JSR   LE788                         TURN ON PIXEL
1662   E926 96 D7                     LDA   VD7                           GET OLD PIXEL MASK
1663   E928 AD C4                     JSR   ,U                            MOVE TO NEXT ONE TO RIGHT
1664   E92A 31 3F                     LEAY  $-01,Y                        DEC COUNTER
1665   E92C 26 F3                     BNE   LE921                         LOOP IF NOT DONE
1666   E92E 39                        RTS
1667   E92F 35 06            LE92F    PULS  A,B                           CLEAN UP STACK
1668                        * DRAW A VERTICAL LINE FROM VEREND TO VERBEG AT HORIZONTAL COORD HORBEG
1669   E931 DC BF            LE931    LDD   VERBEG                        GET END VERTICAL COORDS
1670   E933 34 06                     PSHS  B,A                           SAVE 'EM
1671   E935 BD E9 CD                  JSR   LE9CD                         CALCULATE ABSOLUTE VALUE OF VEREND-VERBEG
1672   E938 24 04                     BCC   LE93E                         BRANCH IF END COORD > START COORD
1673   E93A 9E C5                     LDX   VEREND                        GET VERTICAL END COORDINATE
1674   E93C 9F BF                     STX   VERBEG                        MAKE IT THE START COORD IF END COORD WAS RIGHT OF START
1675   E93E 1F 02            LE93E    TFR   D,Y                           LENGTH OF LINE TO Y
1676   E940 31 21                     LEAY  $01,Y                         SET BOTH START AND END COORDS
1677   E942 BD E7 DA                  JSR   HCALPOS                       GET ABSOLUTE SCREEN ADDRESS IN X AND PIXEL MASK IN ACCA
1678   E945 35 40                     PULS  U                             GET END COORDS
1679   E947 DF BF                     STU   VERBEG                        RESTORE THEM
1680   E949 17 00 D5                  LBSR  LEA21                         POINT U TO ROUTINE TO MOVE DOWN ONE ROW
1681   E94C 20 D3                     BRA   LE921                         DRAW A VERTICAL LINE
1682
1683                        * DRAW A LINE FROM (HORBEG, VERBEG) TO (HOREND, VEREND)
1684   E94E 10 8E E9 B8      LE94E    LDY   #LE9B8                        POINT Y TO INCREMENT VERBEG (VERTICAL START COORD)
1685   E952 BD E9 CD                  JSR   LE9CD                         CALCULATE VERTICAL DIFFERENCE (VEREND-VERBEG)
1686   E955 27 AF                     BEQ   LE906                         DRAW A HORIZONTAL LINE IF DELTA V=0
1687   E957 24 04                     BCC   LE95D                         BRANCH IF VERTICAL END COORD > VERTICAL START COORD
1688   E959 10 8E E9 C6               LDY   #LE9C6                        POINT Y TO DECR VERTICAL COORD
1689   E95D 34 06            LE95D    PSHS  B,A                           SAVE DELTA V
1690   E95F CE E9 B1                  LDU   #LE9B1                        POINT U TO INCR HORIZONTAL COORD
1691   E962 BD E9 DB                  JSR   LE9DB                         CALCULATE HORIZONTAL DIFFERENCE (HOREND-HORBEG)
1692   E965 27 C8                     BEQ   LE92F                         DRAW A VERTICAL LINE IF DELTA H=0
1693   E967 24 03                     BCC   LE96C                         BRANCH IF HORIZONTAL END COORD > HORIZONTAL START COORD
1694   E969 CE E9 BF                  LDU   #LE9BF                        POINT U TO DECR HORIZONTAL COORD
1695   E96C 10 A3 E4         LE96C    CMPD  ,S                            COMPARE DELTA H TO DELTA V
1696   E96F 35 10                     PULS  X                             PUT DELTA V IN X
1697   E971 24 04                     BCC   LE977                         BRANCH IF DELTA H > DELTA V
1698   E973 1E 32                     EXG   U,Y                           SWAP CHANGE HORIZONTAL AND CHANGE VERTICAL ADDRESS
1699   E975 1E 01                     EXG   D,X                           EXCHANGE DELTA HORIZONTAL AND DELTA VERTICAL
1700   E977 34 46            LE977    PSHS  U,B,A                         SAVE THE LARGER OF DELTA V, DELTA H AND INCR/DECR ADDRESS
1701   E979 34 06                     PSHS  B,A                           SAVE THE LARGER OF DELTA V, DELTA H
1702   E97B 44                        LSRA
1703   E97C 56                        RORB                                DIVIDE BY 2, SHIFT ACCD RIGHT ONE BIT
1704   E97D 25 09                     BCS   LE988                         BRANCH IF ODD NUMBER
1705   E97F 11 83 E9 B9               CMPU  #LE9B9                        SEE IF INCR OR DECR
1706   E983 25 03                     BCS   LE988                         BRANCH IF INCR
1707   E985 83 00 01                  SUBD  #1                            SUBTRACT ONE IF DECR
1708   E988 34 16            LE988    PSHS  X,B,A                         SAVE SMALLEST DELTA (X) AND INITIAL MINOR COORDINATE
1709                        * INCREMENT COUNTER WHICH IS 1/2 OF LARGEST DELTA
1710   E98A BD E7 E6                  JSR   LE7E6                         POINT U TO PROPER COORDINATE TO SCREEN CONVERSION ROUTINE
1711
1712                        ** DRAW THE LINE HERE - AT THIS POINT THE STACK HAS THE DRAW DATA ON IT
1713                        * 0 1,S=MINOR COORDINATE INCREMENT COUNTER
1714                        * 2 3,S=ABSOLUTE VALUE OF THE SMALLEST DELTA COORDINATE
1715                        * 4 5,S=ABSOLUTE VALUE OF THE LARGEST DELTA COORDINATE
1716                        * 6 7,S=LARGEST COORDINATE COUNTER (HOW MANY TIMES THROUGH THE DRAW LOOP)
1717                                INITIALLY SET TO ABSOLUTE VALUE OF LARGEST DELTA
1718                        * 8 9,S=ADDRESS OF THE ROUTINE WHICH WILL INCREMENT OR DECREMENT THE LARGEST DELTA COORDINATE
1719
1720   E98D AD C4            LE98D    JSR   ,U                            CONVERT (X,Y) COORDINATES TO ABSOLUTE SCREEN ADDRESS
1721   E98F BD E7 88                  JSR   LE788                         TURN ON A PIXEL
1722   E992 AE 66                     LDX   $06,S                         GET DISTANCE COUNTER
1723   E994 27 17                     BEQ   LE9AD                         BRANCH IF LINE COMPLETELY DRAWN
1724   E996 30 1F                     LEAX  $-01,X                        DECR ONE
1725   E998 AF 66                     STX   $06,S                         SAVE IT
1726   E99A AD F8 08                  JSR   [$08,S]                       INCR/DECR COORDINATE WHICH HAS THE SMALLEST DELTA
1727   E99D EC E4                     LDD   ,S                            GET THE MINOR COORDINATE INCREMENT COUNTER
1728   E99F E3 62                     ADDD  $02,S                         ADD THE SMALLEST DIFFERENCE
```

```
1729   E9A1 ED E4                STD   ,S                          SAVE NEW MINOR COORDINATE INCREMENT COUNTER
1730   E9A3 A3 64                SUBD  $04,S                       SUBTACT OUT THE LARGEST DIFFERENCE
1731   E9A5 25 E6                BCS   LE98D                       BRANCH IF RESULT NOT > LARGEST DIFFERENCE
1732   E9A7 ED E4                STD   ,S                          IF >=, THEN STORE NEW MINOR COORDINATE INCREMENT
1733   E9A9 AD A4                JSR   ,Y                          INCR/DECR COORDINATE WHICH HAS THE SMALLEST DELTA
1734   E9AB 20 E0                BRA   LE98D                       KEEP GOING
1735   E9AD 35 10          LE9AD PULS  X                           CLEAN UP STACK
1736   E9AF 35 F6                PULS  A,B,X,Y,U,PC                CLEAN UP STACK AND RETURN
1737
1738                       * THESE ROUTINES ARE USED TO INCREMENT OR DECREMENT THE HORIZONTAL AND VERTICAL
1739                       * COORDINATES. THEY NEED TO BE KEPT IN THIS ORDER (INCR, INCR, DECR, DECR)
1740   E9B1 9E BD          LE9B1 LDX   HORBEG                      GET HORIZONTAL COORD
1741   E9B3 30 01                LEAX  $01,X                       ADD ONE
1742   E9B5 9F BD                STX   HORBEG                      SAVE NEW HORIZONTAL COORD
1743   E9B7 39                   RTS
1744   E9B8 9E BF                LDX   VERBEG                      GET VERTICAL COORD
1745   E9BA 30 01                LEAX  $01,X                       ADD ONE
1746   E9BC 9F BF                STX   VERBEG                      SAVE NEW VERTICAL COORD
1747   E9BE 39                   RTS
1748   E9BF 9E BD          LE9BF LDX   HORBEG                      GET HORIZONTAL COORD
1749   E9C1 30 1F                LEAX  $-01,X                      SUBTRACT ONE
1750   E9C3 9F BD                STX   HORBEG                      SAVE NEW HORIZONTAL COORD
1751   E9C5 39                   RTS
1752   E9C6 9E BF                LDX   VERBEG                      GET VERTICAL COORD
1753   E9C8 30 1F                LEAX  $-01,X                      SUBTRACT ONE
1754   E9CA 9F BF                STX   VERBEG                      SAVE NEW VERTICAL COORD
1755   E9CC 39             LE9CC RTS
1756
1757   E9CD DC C5          LE9CD LDD   VEREND                      GET VERTICAL ENDING ADDRESS
1758   E9CF 93 BF                SUBD  VERBEG                      SUBTRACT OUT VERTICAL BEGINNING ADDRESS
1759   E9D1 24 F9          LE9D1 BCC   LE9CC                       RETURN IF END >= START
1760   E9D3 34 01                PSHS  CC                          SAVE STATUS (WHICH COORDINATE IS GREATER)
1761
1762                       * THE NEXT THREE INSTRUCTIONS WILL NEGATE ACCD
1763   E9D5 40                   NEGA
1764   E9D6 50                   NEGB
1765   E9D7 82 00                SBCA  #$00                        NEGATE ACCB
1766   E9D9 35 81                PULS  CC,PC                       RESTORE STATUS AND RETURN
1767
1768   E9DB DC C3          LE9DB LDD   HOREND                      GET HORIZONTAL END COORD
1769   E9DD 93 BD                SUBD  HORBEG                      SUBTRACT OUT HORIZONTAL START COORD
1770   E9DF 20 F0                BRA   LE9D1                       GET ABSOLUTE VALUE
1771
1772                       * EVALUATE TWO SETS OF COORDINATES SEPERATED BY A MINUS
1773                       * SIGN. PUT 1ST SET OF COORDINATES AT (HORBEG,VERBEG), SECOND
1774                       * SET AT (HOREND,VEREND). IF NOTHING BEFORE MINUS SIGN, PUT
1775                       * (HORDEF,VERDEF) DEFAULTS AT (HORBEG,VERBEG).
1776
1777   E9E1 9E C7          LE9E1 LDX   HORDEF                      GET THE LAST HORIZONTAL END POINT
1778   E9E3 9F BD                STX   HORBEG                      PUT AS START POINT
1779   E9E5 9E C9                LDX   VERDEF                      GET THE LAST VERTICAL END POINT
1780   E9E7 9F BF                STX   VERBEG                      PUT AS VERTICAL START POINT
1781   E9E9 81 AC                CMPA  #$AC                        CHECK FOR MINUS SIGN (-) TOKEN
1782   E9EB 27 03                BEQ   LE9F0                       BRANCH IF NO STARTING COORDINATES GIVEN
1783   E9ED BD EA 04             JSR   LEA04                       GO GET THE STARTING COORDINATES
1784   E9F0 C6 AC          LE9F0 LDB   #$AC                        TOKEN FOR THE MINUS SIGN (-)
1785   E9F2 BD B2 6F             JSR   LB26F                       DO A SYNTAX CHECK FOR A MINUS SIGN
1786   E9F5 BD B2 6A             JSR   LB26A                       SYNTAX CHECK FOR A '('
1787   E9F8 BD B7 34             JSR   LB734                       EVALUATE 2 EXPRESSIONS
1788   E9FB 10 8E 00 C3          LDY   #HOREND                     TEMP STORAGE LOCS FOR END COORDS OF LINE COMMAND
1789   E9FF BD E7 B9             JSR   LE7B9                       GET END POINT COORDINATES
1790   EA02 20 06                BRA   LEA0A                       SYNTAX CHECK FOR A ')'
1791   EA04 BD B2 6A       LEA04 JSR   LB26A                       SYNTAX CHECK FOR A '('
1792   EA07 BD E7 B2             JSR   LE7B2                       EVALUATE  HORIZONTAL & VERTICAL COORDINATES WITH RANGE CHECK
1793   EA0A 7E B2 67       LEA0A JMP   LB267                       SYNTAX CHECK FOR ')' AND RETURN
1794   EA0D BD E7 AD       LEA0D JSR   LE7AD                       POINT U TO HORBEG; USELESS GIVEN THE FOLLOWINF INSTRUCTION
1795   EA10 CE 00 C3             LDU   #HOREND                     POINT U TO HOREND
1796   EA13 7E E7 B0             JMP   LE7B0                       JUMP TO AN RTS; ONCE WAS A JUMP TO NORMALIZATION ROUTINE
1797
1798                       * POINT U TO ROUTINE WHICH WILL MOVE PIXEL ONE TO RIGHT
1799   EA16 CE EA 25       LEA16 LDU   #LEA25                      POINT TO JUMP TABLE
1800   EA19 D6 E6                LDB   HRMODE                      GET HI-RES GRAPHICS MODE VALUE
1801   EA1B C0 01                SUBB  #$01                        ADJUST OUT MODE 0 (WHY NOT DECB)
1802   EA1D 58                   ALSB                              TWO BYTES PER ENTRY
1803   EA1E EE C5                LDU   B,U                         GET JUMP ADDRESS
1804   EA20 39                   RTS
1805
1806   EA21 CE EA 45       LEA21 LDU   #LEA45                      POINT U TO ROUTINE TO MOVE ABSOLUTE POS DOWN ONE ROW
1807   EA24 39                   RTS
1808
1809                       * JUMP TABLE OF ADDRESSES OF ROUTINES WHICH WILL MOVE THE
1810                       * ABSOLUTE SCREEN ADDRESS POINTER ONE PIXEL TO THE RIGHT
1811   EA25 EA 34          LEA25 FDB   LEA34                       HSCREEN 1
1812   EA27 EA 3D          LEA27 FDB   LEA3D                       HSCREEN 2
1813   EA29 EA 2D          LEA29 FDB   LEA2D                       HSCREEN 3
1814   EA2B EA 34          LEA2B FDB   LEA34                       HSCREEN 4
1815
1816                       * ENTER WITH ABSOLUTE SCREEN POSITION IN X, PIXEL MASK
1817                       * IN ACCA - ADJUST X AND ACCA TO THE NEXT PIXEL TO THE RIGHT FOR HSCREEN 3
1818   EA2D 44             LEA2D LSRA                              SHIFT ONE BIT TO THE RIGHT
1819   EA2E 24 03                BCC   LEA33                       BRANCH IF SAME BYTE
1820   EA30 46                   RORA                              SET BIT 7 OF ACCA IF JUST MOVED TO NEXT BYTE
1821   EA31 30 01                LEAX  $01,X                       ADD ONE TO SCREEN POSITION
1822   EA33 39             LEA33 RTS
1823
1824                       * ENTER WITH ABSOLUTE SCREEN POSITION IN X, PIXEL MASK IN ACCA -
```

```
1825                          * ADJUST X AND ACCA TO THE NEXT PIXEL TO THE RIGHT FOR HSCREEN 1 & 4
1826   EA34 44                LEA34   LSRA                            SHIFT MASK ONE BIT TO THE RIGHT
1827   EA35 44                        LSRA                            DO IT AGAIN
1828   EA36 24 FB                     BCC   LEA33                     BRANCH IF SAME BYTE
1829   EA38 86 C0                     LDA   #$C0                      SET PIXEL #3 IF NEW BYTE
1830   EA3A 30 01                     LEAX  $01,X                     ADD ONE TO SCREEN ADDRESS
1831   EA3C 39                        RTS
1832
1833                          * ENTER WITH ABSOLUTE SCREEN POSITION IN X, PIXEL MASK IN ACCA -
1834                          * ADJUST X AND ACCA TO THE NEXT PIXEL TO THE RIGHT FOR HSREEN 2
1835   EA3D 43                LEA3D   COMA                            SET TO ALTERNATE PIXEL
1836   EA3E 81 F0                     CMPA  #$F0                      SEE IF TOP HALF OF BYTE
1837   EA40 26 02                     BNE   LEA44                     BRANCH IF SAME BYTE
1838   EA42 30 01                     LEAX  $01,X                     MOVE POINTER TO NEXT SCREEN ADDRESS
1839   EA44 39                LEA44   RTS
1840
1841                          * ROUTINE TO MOVE DOWN ONE ROW
1842                          * ENTER WITH ABSOLUTE SCREEN ADDRESS IN X
1843   EA45 D6 B9             LEA45   LDB   HORBYT                    GET NUMBER OF BYTES PER HORIZONTAL GRAPHICS ROW
1844   EA47 3A                        ABX                             ADD A ROW TO CURRENT ADDRESS (MOVE DOWN ONE ROW)
1845   EA48 39                        RTS
1846
1847                          * HCIRCLE
1848   EA49 0D E6             HCIRCLE TST   HRMODE                    CHECK HI-RES GRAPHICS MODE
1849   EA4B 10 27 FC A0               LBEQ  LE6EF                     BRANCH IF NOT HI-RES GRAPHICS
1850   EA4F 10 21 15 AD               LBRN  RAMLINK                   RAM HOOK
1851   EA53 81 40                     CMPA  #'@'                      CHECK FOR @ SIGN (HCIRCLE@ IS LEGAL SYNTAX)
1852   EA55 26 02                     BNE   LEA59                     BRANCH IF NOT
1853   EA57 9D 9F                     JSR   GETNCH                    GET THE NEXT CHARACTER FROM BASIC'S INPUT LINE
1854   EA59 BD EB 60          LEA59   JSR   LEB60                     GET MAX HORIZONTAL & VERTICAL COORD VALUES AND PUT THEM IN VD3 & VD5
1855   EA5C BD EA 04                  JSR   LEA04                     GET HORIZONTAL & VERTICAL CENTER COORDS AND PUT THEM IN VBD AND VBF
1856   EA5F BD E7 AD                  JSR   LE7AD                     NORMALIZE START COORDS FOR PROPER HI-RES GRAPHICS MODE
1857   EA62 AE C4                     LDX   ,U                        GET HORIZONTAL COORD
1858   EA64 9F CB                     STX   VCB                       SAVE IT
1859   EA66 AE 42                     LDX   $02,U                     GET VERTICAL COORD
1860   EA68 9F CD                     STX   VCD                       SAVE IT
1861   EA6A BD B2 6D                  JSR   SYNCOMMA                  DO A SYNTAX CHECK FOR A COMMA
1862   EA6D BD B7 3D                  JSR   LB73D                     EVALUATE EXPRESSION, RETURN VALUE IN X
1863   EA70 CE 00 CF                  LDU   #VCF                      POINT U TO TEMP DATA STORAGE
1864   EA73 AF C4                     STX   ,U                        SAVE RADIUS
1865   EA75 BD E7 B0                  JSR   LE7B0                     NOW A JSR TO AN RTS; WAS A CALL TO A NORMALIZATION ROUTINE
1866   EA78 86 01                     LDA   #$01                      PSET FLAG
1867   EA7A 97 C2                     STA   SETFLG                    SAVE PSET/PRESET FLAG
1868   EA7C BD E7 18                  JSR   LE718                     GO EVALUATE COLOR EXPRESSION AND SAVE VALUE
1869   EA7F 8E 01 00                  LDX   #$100                     DEFAULT HEIGHT/WIDTH RATIO
1870   EA82 9D A5                     JSR   GETCCH                    GET BASIC'S CURRENT INPUT CHARACTER
1871   EA84 27 0F                     BEQ   LEA95                     BRANCH IF NONE
1872   EA86 BD B2 6D                  JSR   SYNCOMMA                  DO A SYNTAX CHECK FOR A COMMA
1873   EA89 BD B1 41                  JSR   LB141                     EVALUATE A NUMERIC EXPRESSION
1874   EA8C 96 4F                     LDA   FP0EXP                    GET FPA0 EXPONENT
1875   EA8E 8B 08                     ADDA  #$08                      ADD 8 TO IT (EFFECTIVELY MULTIPLIES BY 256)
1876   EA90 97 4F                     STA   FP0EXP                    SAVE NEW VALUE
1877   EA92 BD B7 40                  JSR   LB740                     EVALUATE EXPRESSION, RETURN VALUE IN X
1878   EA95 96 E6             LEA95   LDA   HRMODE                    GET CURRENT HI-RES GRAPHICS MODE
1879   EA97 81 02                     CMPA  #$02                      SEE WHICH MODE IT IS
1880   EA99 22 04                     BHI   LEA9F                     BRANCH IF HSCREEN 4
1881   EA9B 1F 10                     TFR   X,D                       PREPARE TO DOUBLE THE HEIGHT/WIDTH RATIO FOR MODES 0-2
1882   EA9D 30 8B                     LEAX  D,X                       DOUBLE H/W RATIO TO COMPENSATE FOR HORIZONTAL PIXEL SIZE
1883   EA9F 9F D1             LEA9F   STX   VD1                       SAVE H/W RATIO
1884   EAA1 C6 01                     LDB   #$01                      CODE FOR PSET
1885   EAA3 D7 C2                     STB   SETFLG                    SET PSET/PRESET FLAG TO PSET
1886   EAA5 D7 D8                     STB   VD8                       FIRST TIME FLAG - SET TO 0 AFTER ARC DRAWN
1887   EAA7 BD EB 7B                  JSR   LEB7B                     EVALUATE CIRCLE START POINT (OCTANT, SUBARC)
1888   EAAA 34 06                     PSHS  B,A                       SAVE START POINT
1889   EAAC BD EB 7B                  JSR   LEB7B                     EVALUATE CIRCLE END POINT (OCTANT, SUBARC)
1890   EAAF DD D9                     STD   VD9                       SAVE END POINT
1891   EAB1 35 06                     PULS  A,B                       GET BACK START POINT
1892   EAB3 34 06             LEAB3   PSHS  B,A                       STORE CURRENT CIRCLE POSITION
1893   EAB5 9E C3                     LDX   HOREND                    GET END HORIZONTAL COORD
1894   EAB7 9F BD                     STX   HORBEG                    MAKE IT THE NEW START
1895   EAB9 9E C5                     LDX   VEREND                    GET END VERTICAL COORD
1896   EABB 9F BF                     STX   VERBEG                    MAKE IT THE NEW START
1897   EABD CE EB 9B                  LDU   #LEB9B                    POINT TO TABLE OF SINES AND COSINES
1898   EAC0 84 01                     ANDA  #$01                      TEST OCTANT NUMBER
1899   EAC2 27 03                     BEQ   LEAC7                     BRANCH IF EVEN
1900   EAC4 50                        NEGB
1901   EAC5 CB 08                     ADDB  #$08                      CONVERT 0-7 TO 8-1 FOR ODD OCTANT NUMBERS
1902   EAC7 58                LEAC7   ALSB                            MUL BY 2
1903   EAC8 58                        ALSB                            DO IT AGAIN (FOUR BYTES PER TABLE ENTRY)
1904   EAC9 33 C5                     LEAU  B,U                       POINT TO CORRECT TABLE ENTRY
1905   EACB 34 40                     PSHS  U                         SAVE SIN/COS TABLE ENTRY
1906   EACD BD EB BD                  JSR   LEBBD                     CALCULATE HORIZONTAL OFFSET
1907   EAD0 35 40                     PULS  U                         GET BACK SIN/COS TABLE POINTER
1908   EAD2 33 5E                     LEAU  $-02,U                    MOVE TO COSINE (VERTICAL)
1909   EAD4 34 10                     PSHS  X                         SAVE HORIZONTAL OFFSET
1910   EAD6 BD EB BD                  JSR   LEBBD                     CALCULATE VERTICAL OFFSET
1911   EAD9 35 20                     PULS  Y                         PUT HORIZONTAL OFFSET IN Y
1912   EADB A6 E4                     LDA   ,S                        GET OCTANT NUMBER
1913   EADD 84 03                     ANDA  #$03                      MASK OFF BOTTOM TWO BITS
1914   EADF 27 06                     BEQ   LEAE7                     BRANCH IF OCTANT 0 OR 4
1915   EAE1 81 03                     CMPA  #$03                      NOW SEE IF BOTH BITS WERE SET
1916   EAE3 27 02                     BEQ   LEAE7                     BRANCH IF OCTANT 3 OR 7
1917   EAE5 1E 12                     EXG   X,Y                       SWAP HORIZONTAL AND VERTICAL OFFSETS
1918   EAE7 9F C3             LEAE7   STX   HOREND                    SAVE HORIZONTAL OFFSET
1919                          * H/W RATIO WILL ONLY MODIFY THE VERTICAL COORD
1920   EAE9 1F 20                     TFR   Y,D                       PUT CALCULATED VERTICAL OFFSET INTO ACCD
```

```
1921  EAEB 44                     LSRA
1922  EAEC 56                     RORB                            DIVIDE OFFSET BY 2
1923  EAED 9E D1                  LDX   VD1                       GET H/W RATIO
1924  EAEF BD EB CB               JSR   LEBCB                     MULT VERTICAL OFFSET BY H/W RATIO
1925  EAF2 1F 20                  TFR   Y,D                       TRANSFER PRODUCT TO ACCD
1926  EAF4 4D                     TSTA                            CHECK OVERFLOW AND GET MS BYTE RESULT
1927  EAF5 10 26 C9 51            LBNE  ILLFUNC                   ILLEGAL FUNCTION CALL ERROR (RESULT > 255)
1928  EAF9 D7 C5                  STB   VEREND                    SAVE DELTA VERTICAL MS BYTE
1929  EAFB 1F 30                  TFR   U,D                       LS BYTE RESULT TO ACCA
1930  EAFD 97 C6                  STA   VEREND+1                  SAVE DELTA VERTICAL LS BYTE
1931  EAFF A6 E4                  LDA   ,S                        GET OCTANT NUMBER
1932  EB01 81 02                  CMPA  #$02                      CHECK FOR OCTANT 0,1,6,7
1933  EB03 25 0E                  BCS   LEB13                     BRANCH IF SUBARC HORIZONTAL END POINT >= HORIZONTAL CENTER
1934  EB05 81 06                  CMPA  #$06                      MORE CHECKS FOR OCTANT 0,1,6,7
1935  EB07 24 0A                  BCC   LEB13                     BRANCH IF SUBARC HORIZONTAL END POINT >= HORIZONTAL CENTER
1936  EB09 DC CB                  LDD   VCB                       GET HORIZONTAL COORD OF CENTER
1937  EB0B 93 C3                  SUBD  HOREND                    SUBTRACT HORIZONTAL DIFFERENCE
1938  EB0D 24 11                  BCC   LEB20                     BRANCH IF NO UNDERFLOW
1939  EB0F 4F                     CLRA
1940  EB10 5F                     CLRB                            FORCE COORD TO 0 IF RESULT WAS LESS THAN 0
1941  EB11 20 0D                  BRA   LEB20                     SAVE NEW COORD
1942  EB13 DC CB            LEB13 LDD   VCB                       GET HORIZONTAL COORD OF CENTER
1943  EB15 D3 C3                  ADDD  HOREND                    ADD HORIZONTAL DIFFERENCE
1944  EB17 25 05                  BCS   LEB1E                     BRANCH IF OVERFLOW
1945  EB19 10 93 D3               CMPD  VD3                       COMPARE TO MAX HORIZONTAL COORDINATE
1946  EB1C 25 02                  BCS   LEB20                     BRANCH IF < MAX HOR
1947  EB1E DC D3            LEB1E LDD   VD3                       GET MAX HORIZONTAL COORD
1948  EB20 DD C3            LEB20 STD   HOREND                    SAVE NEW HORIZONTAL SUBARC END COORD
1949  EB22 A6 E4                  LDA   ,S                        GET OCTANT NUMBER
1950  EB24 81 04                  CMPA  #$04                      CHECK FOR OCTANT 0,1,2 OR 3
1951  EB26 25 0A                  BCS   LEB32                     BRANCH IF SUBARC VERTICAL END POINT >= VERTICAL CENTER
1952  EB28 DC CD                  LDD   VCD                       GET VERTICAL COORD OF CENTER
1953  EB2A 93 C5                  SUBD  VEREND                    SUBTRACT VERTICAL DIFFERENCE
1954  EB2C 24 11                  BCC   LEB3F                     BRANCH IF NO UNDERFLOW
1955  EB2E 4F                     CLRA
1956  EB2F 5F                     CLRB                            FORCE NEW VERTICAL TO 0 IF MINUS
1957  EB30 20 0D                  BRA   LEB3F                     SAVE NEW COORD
1958  EB32 DC CD            LEB32 LDD   VCD                       GET VERTICAL COORD OF CENTER
1959  EB34 D3 C5                  ADDD  VEREND                    ADD VERTICAL DIFFERENCE
1960  EB36 25 05                  BCS   LEB3D                     BRANCH IF OVERFLOW
1961  EB38 10 93 D5               CMPD  VD5                       COMPARE TO MAX VERTICAL COORD
1962  EB3B 25 02                  BCS   LEB3F                     BRANCH IF < MAX VER
1963  EB3D DC D5            LEB3D LDD   VD5                       GET MAX VERTICAL COORD
1964  EB3F DD C5            LEB3F STD   VEREND                    SAVE NEW VERTICAL SUBARC END COORD
1965  EB41 0D D8                  TST   VD8                       CHECK FIRST TIME FLAG
1966  EB43 26 03                  BNE   LEB48                     DO NOT DRAWE A LINE FIRST TIME THROUGH -
1967                                                              BECAUSE THE FIRST TIME YOU WOULD DRAW A LINE
1968                                                              FROM THE CENTER TO THE FIRST POINT ON THE CIRCLE
1969  EB45 17 FE 06               LBSR  LE94E                     DRAW A LINE
1970  EB48 35 06            LEB48 PULS  A,B                       GET END COORDS
1971  EB4A 04 D8                  LSR   VD8                       SHIFT FIRST TIME FLAG
1972  EB4C 25 05                  BCS   LEB53                     DO NOT CHECK FOR END POINT AFTER DRAWING FIRST ARC
1973  EB4E 10 93 D9               CMPD  VD9                       COMPARE CURRENT POSITION TO END POINT
1974  EB51 27 0C                  BEQ   LEB5F                     BRANCH IF CIRCLE DRAWING IS FINISHED
1975                        * INCREMENT SUBARC CTR, IF . 7 THEN INC OCTANT CTR
1976  EB53 5C               LEB53 INCB                            INC SUBARC COUNTER
1977  EB54 C1 08                  CMPB  #$08                      > 7?
1978  EB56 26 04                  BNE   LEB5C                     BRANCH IF NOT
1979  EB58 4C                     INCA                            INCR OCTANT COUNTER
1980  EB59 5F                     CLRB                            RESET SUBARC COUNTER
1981  EB5A 84 07                  ANDA  #$07                      KEEP IN RANGE OF 0-7; ONCE ACCA=ACCB, THIS WILL MAKE ACCA=0
1982                                                              SO THE END POINT WILL BE (0,0) AND THE CIRCLE ROUTINE WILL END
1983  EB5C 7E EA B3         LEB5C JMP   LEAB3                     KEEP DRAWING THE CIRCLE
1984  EB5F 39               LEB5F RTS                             EXIT CIRCLE ROUTINE
1985
1986                        * GET MAXIMUM VALUE OF HORIZONTAL & VERTICAL COORDINATES NORMALIZED FOR
1987                        * PROPER GRAPHICS MODE. RETURN VALUES: HORIZONTAL IN VD3, VERTICAL IN VD5
1988  EB60 CE 00 D3         LEB60 LDU   #VD3                      POINT U TO STORAGE AREA
1989  EB63 8E 02 7F               LDX   #640-1                    GET MAXIMUM HORIZONTAL COORD
1990  EB66 AF C4                  STX   ,U                        SAVE IT
1991  EB68 96 E6                  LDA   HRMODE                    GET CURRENT GRAPHICS MODE
1992  EB6A 81 02                  CMPA  #$02                      SEE WHICH MODE
1993  EB6C 2E 05                  BGT   LEB73                     BRANCH IF MODES 3 OR 4
1994  EB6E 8E 01 3F               LDX   #320-1                    MAXIMUM VALUE FOR HORIZONTAL COORD IN MODES 1 AND 2
1995  EB71 AF C4                  STX   ,U                        SAVE IT
1996  EB73 8E 00 BF         LEB73 LDX   #192-1                    GET THE MAXIMUM VERTICAL COORD
1997  EB76 AF 42                  STX   $02,U                     SAVE IT
1998  EB78 7E E7 B0               JMP   LE7B0                     JUMP TO AN RTS; ONCE WAS A NORMALIZATION ROUTINE
1999
2000                        * EVALUATE CIRCLE START POINT (OCTANT, SUBARC)
2001                        * CALCULATE START OF END POINT WHICH IS A NUMBER FROM
2002                        * 0-63 SAVED AS AN OCTANT NUMBER (0-7) AND SUBARC NUMBER (0-7)
2003  EB7B 5F               LEB7B CLRB                            SET DEFAULT VALUE TO 0
2004  EB7C 9D A5                  JSR   GETCCH                    GET BASIC'S CURRENT INPUT CHARACTER
2005  EB7E 27 11                  BEQ   LEB91                     BRANCH IF NONE
2006  EB80 BD B2 6D               JSR   SYNCOMMA                  DO A SYNTAX CHECK FOR A COMMA
2007  EB83 BD B1 41               JSR   LB141                     EVALUATE A NUMERIC EXPRESSION
2008  EB86 96 4F                  LDA   FP0EXP                    GET EXPONENT OF FPA0
2009  EB88 8B 06                  ADDA  #$06                      ADD 6 TO EXPONENT - MULTIPLY EXPONENT BY 64
2010  EB8A 97 4F                  STA   FP0EXP                    RESAVE IT
2011  EB8C BD B7 0E               JSR   LB70E                     CONVERT FPA0 TO INTEGER IN ACCB
2012  EB8F C4 3F                  ANDB  #$3F                      FORCE MAX VALUE OF 63
2013  EB91 1F 98            LEB91 TFR   B,A                       SAVE VALUE IN ACCA ALSO
2014  EB93 C4 07                  ANDB  #$07                      NOW ACCB CONTAINS SUBARC NUMBER
2015  EB95 44                     LSRA
2016  EB96 44                     LSRA
```

```
2017  EB97 44                  LSRA                            DIVIDE ACCA BY 8 - OCTANT NUMBER
2018  EB98 39                  RTS
2019
2020  EB99 00 00 00 01   CIRCDATA FDB   $0000,$0001            SUBARC 0
2021  EB9D FE C5 19 19   LEB9D    FDB   $FEC5,$1919            SUBARC 1
2022  EBA1 FB 16 31 F2   LEBA1    FDB   $FB16,$31F2            SUBARC 2
2023  EBA5 F4 FB 4A 51   LEBA5    FDB   $F4FB,$4A51            SUBARC 3
2024  EBA9 EC 84 61 F9   LEBA9    FDB   $EC84,$61F9            SUBARC 4
2025  EBAD E1 C7 78 AE   LEBAB    FDB   $E1C7,$78AE            SUBARC 5
2026  EBB1 D4 DC 8E 3B   LEBAF    FDB   $D4DC,$8E3B            SUBARC 6
2027  EBB5 C5 E5 A2 69   LEBB5    FDB   $C5E5,$A269            SUBARC 7
2028  EBB9 B5 06 B5 06   LEBB9    FDB   $B506,$B506            SUBARC 8
2029
2030                     * MULTIPLY RADIUS BY SIN/COS VALUE AND RETURN OFFSET IN X
2031  EBBD 9E CF         LEBBD    LDX   VCF                     GET RADIUS
2032  EBBF EC C4                  LDD   ,U                      GET SIN/COS TABLE MODIFIER
2033  EBC1 27 07                  BEQ   LEBCA                   BRANCH IF 0 (OFFSET = RADIUS)
2034  EBC3 83 00 01               SUBD  #1                      SUBTRACT ONE
2035  EBC6 8D 03                  BSR   LEBCB                   MULTIPLY RADIUS BY SIN/COS
2036  EBC8 1F 21                  TFR   Y,X                     RETURN RESULT IN X
2037  EBCA 39            LEBCA    RTS
2038
2039                     * MULTIPLY (UNSIGNED) TWO 16 BIT NUMBERS TOGETHER -
2040                     * ENTER WITH ONE NUMBER IN ACCD, THE OTHER IN X REGISTER
2041                     * THE 4 BYTE PRODUCT WILL BE STORED IN 4,S - 7,S
2042                     * (Y, U REGISTERS ON THE STACK). I.E. (AA AB) x (XH,XL)=
2043                     * 256 * AA * XH + 16 * (AA * XL + AB * HX) + AB * XL. THE TWO BYTE
2044                     * MULTIPLIER AND THE MULTIPLICAND ARE TREATED AS A 1
2045                     * BYTE INTEGER PART (MSB) WITH A 1 BYTE FRACTIONAL PART (LSB)
2046
2047  EBCB 34 76         LEBCB    PSHS  U,Y,X,B,A               SAVE REGISTERS AND RESERVE STORAGE SPACE ON THE STACK
2048  EBCD 6F 64                  CLR   $04,S                   RESET OVERFLOW FLAG
2049  EBCF A6 63                  LDA   $03,S                   =
2050  EBD1 3D                     MUL                           =
2051  EBD2 ED 66                  STD   $06,S                   = CALCULATE ACCB*XL, STORE RESULT IN 6,S
2052  EBD4 EC 61                  LDD   $01,S                   *
2053  EBD6 3D                     MUL                           * CALCULATE ACCB*XH
2054  EBD7 EB 66                  ADDB  $06,S                   *
2055  EBD9 89 00                  ADCA  #$00                    =
2056  EBDB ED 65                  STD   $05,S                   = ADD THE CARRY FROM THE 1ST MUL  TO THE RESULT OF THE 2ND MUL
2057  EBDD E6 E4                  LDB   ,S                      *
2058  EBDF A6 63                  LDA   $03,S                   *
2059  EBE1 3D                     MUL                           * CALCULATE ACCA*XL
2060  EBE2 E3 65                  ADDD  $05,S                   =
2061  EBE4 ED 65                  STD   $05,S                   = ADD RESULT TO TOTAL OF 2 PREVIOUS MULTS
2062  EBE6 24 02                  BCC   LEBEA                   BRANCH IF NO OVERFLOW
2063  EBE8 6C 64                  INC   $04,S                   SET OVERFLOW FLAG (ACCD > $FFFF)
2064  EBEA A6 E4         LEBEA    LDA   ,S                      *
2065  EBEC E6 62                  LDB   $02,S                   *
2066  EBEE 3D                     MUL                           * CALCULATE ACCA*XH
2067  EBEF E3 64                  ADDD  $04,S                   =
2068  EBF1 ED 64                  STD   $04,S                   = ADD TO PREVIOUS RESULT
2069  EBF3 35 F6                  PULS  A,B,X,Y,U,PC            RETURN WITH RESULT IN U AND Y
2070
2071                     * HPAINT
2072  EBF5 0D E6         HPAINT   TST   HRMODE                   CHECK HI-RES GRAPHICS MODE
2073  EBF7 10 27 FA F4            LBEQ  LE6EF                   'HR' ERROR IF HI-RES GRAPHICS MODE NOT SET UP
2074  EBFB 10 21 14 01            LBRN  RAMLINK                 RAM HOOK
2075  EBFF 81 40                  CMPA  #'@'                    CHECK FOR @ SIGN
2076  EC01 26 02                  BNE   LEC05                   BRANCH IF NOT
2077  EC03 9D 9F                  JSR   GETNCH                  GET THE NEXT CHARACTER FROM BASIC'S INPUT LINE
2078  EC05 BD EA 04      LEC05    JSR   LEA04                   SYNTAX CHECK FOR '(', TWO EXPRESSIONS, AND ')'
2079  EC08 BD E7 AD               JSR   LE7AD                   NORMALIZE THE HORIZONTAL AND VERTICAL COORDS
2080  EC0B 86 01                  LDA   #$01                    CODE FOR PSET
2081  EC0D 97 C2                  STA   SETFLG                  SET PSET/PRESET FLAG TO PSET
2082  EC0F BD E7 18               JSR   LE718                   GET PAINT COLOR CODE & SET ACTIVE COLOR AND ALL PIXEL BYTES
2083  EC12 DC B4                  LDD   WCOLOR                  GET THEM
2084  EC14 34 06                  PSHS  B,A                     SAVE THEM ON THE STACK
2085  EC16 9D A5                  JSR   GETCCH                  GET BASIC'S CURRENT INPUT CHARACTER
2086  EC18 27 03                  BEQ   LEC1D                   BRANCH IF NONE LEFT - DEFAULT BORDER COLOR TO FOREGROUND,
2087                                                            PAINT COLOR TO BACKGROUND
2088  EC1A BD E7 18               JSR   LE718                   EVALUATE THE BORDER COLOR
2089  EC1D 96 B5         LEC1D    LDA   ALLCOL                  GET BORDER COLOR ALL PIXEL BYTE
2090  EC1F 97 D8                  STA   VD8                     TEMP SAVE IT
2091  EC21 35 06                  PULS  A,B                     GET PAINT ACTIVE COLORS BACK
2092  EC23 DD B4                  STD   WCOLOR                  RESAVE THEM
2093  EC25 BD E1 19               JSR   SELTASK1
2094  EC28 4F                     CLRA                          * STORE A BLOCK OF 'PAINT' DATA ON THE STACK WHICH
2095  EC29 34 56                  PSHS  U,X,B,A                 * WILL ACT AS AN END OF 'PAINT' DATA FLAG.
2096                     * THE CLRA WILL CAUSE THE UP/DN FLAG TO BE ZERO WHICH IS USED TO EXIT THE HPAINT ROUTINE
2097  EC2B BD EB 60               JSR   LEB60                   GET NORMALIZED MAX HOR/VERTICAL VALUES - RETURN RESULT IN VD3,VD5
2098  EC2E BD E7 E6               JSR   LE7E6                   POINT U TO THE ROUTINE WHICH WILL SELECT A PIXEL
2099                     *
2100                     * 'PAINT' THE FIRST HORIZONTAL LINE FROM THE START COORDINATES
2101  EC31 DF D9                  STU   VD9                     SAVE ADDRESS
2102  EC33 BD EC BE               JSR   LECBE                   'PAINT' FROM THE CURRENT HORIZONTAL COORD TO ZERO
2103  EC36 27 0F                  BEQ   LEC47                   BRANCH IF NO PAINTING DONE - HIT BORDER INSTANTLY
2104  EC38 BD ED 01               JSR   LED01                   PAINT TOWARD MAX HORIZONTAL COORD
2105  EC3B 86 01                  LDA   #$01                    SET UP/DN FLAG TO UP (1=UP, $FF=DOWN)
2106  EC3D 97 D7                  STA   VD7                     SAVE IT
2107  EC3F BD ED 2E               JSR   LED2E                   SAVE POSITIVE GOING LINE INFO ON STACK
2108  EC42 00 D7                  NEG   VD7                     SET UP/DN FLAG TO $FF (DOWN)
2109  EC44 BD ED 2E               JSR   LED2E                   SAVE NEGATIVE GOING LINE INFO ON STACK
2110  EC47 10 DF DC      LEC47    STS   TMPSTK                  TEMP STORE STACK POINTER
2111  EC4A 0D DB         LEC4A    TST   CHGFLG                  SEE IF PAINTED COLOR IS DIFFERENT THAN THE ORIGINAL COLOR
2112  EC4C 26 03                  BNE   LEC51                   BRANCH IF DATA HAS BEEN MODIFIED
```

```
2113   EC4E 1Ø DE DC            LDS   TMPSTK                        GET STACK POINTER BACK
2114   EC51 35 56       LEC51   PULS  A,B,X,U                      GET DATA FOR NEXT LINE SEGMENT TO CHECK FROM THE STACK
2115   EC53 ØF DB               CLR   CHGFLG                        CLEAR THE CHANGE FLAG
2116   EC55 1Ø DF DC            STS   TMPSTK                        TEMP SAVE THE STACK ADDRESS
2117   EC58 3Ø Ø1               LEAX  $Ø1,X                         ADD ONE TO 'START HORIZONTAL COORD -1'
2118   EC5A 9F BD               STX   HORBEG                        PIT IT AT 'CURRENT HORIZONTAL COORD ADDRESS'
2119   EC5C DF D1               STU   VD1                           SAVE LENGTH OF PARENT LINE
2120   EC5E 97 D7               STA   VD7                           SAVE UP/DN FLAG
2121   EC6Ø 27 58               BEQ   LECBA                         EXIT ROUTINE IF UP/DN FLAG = Ø
2122   EC62 2B Ø6               BMI   LEC6A                         BRANCH IF UP/DN FLAG = DOWN
2123                           * CHECK ONE LINE BELOW CURRENT DATA
2124   EC64 5C                  INCB                                INCREMENT VERTICAL COORD
2125   EC65 D1 D6               CMPB  VD6                           COMPARE TO MAXIMUM VERTICAL COORD
2126   EC67 23 Ø5               BLS   LEC6E                         BRANCH IF NOT GREATER - PROCESS LINE
2127   EC69 5F                  CLRB                                SET VERTICAL COORD TO ZERO TO FORCE WRAP AROUND
2128   EC6A 5D          LEC6A   TSTB                                CHECK VERTICAL COORD
2129   EC6B 27 DD               BEQ   LEC4A                         PROCESS ANOTHER BLOCK OF PAINT DATA IF WRAP AROUND -
2130                                                                DISCARD ANY LINE BELOW VERTICAL COORD = Ø OR ABOVE MAX VER COORD
2131   EC6D 5A                  DECB                                DEC VERTICAL COORD
2132                           * PROCESS A HORIZONTAL LINE THAT WAS STORED ON STACK - LIMIT CHECK HAVE BEEN DONE
2133   EC6E D7 CØ       LEC6E   STB   VERBEG+1                      SAVE CURRENT VERTICAL COORD
2134   EC7Ø BD EC BE            JSR   LECBE                         PAINT FROM HORIZONTAL COORD TO ZERO OR BORDER
2135   EC73 27 11               BEQ   LEC86                         BRANCH IF NO PIXELS WERE PAINTED
2136   EC75 1Ø 83 ØØ Ø3         CMPD  #3                            SEE IF FEWER THAN 3 PIXELS WERE PAINTED
2137   EC79 25 Ø5               BCS   LEC8Ø                         BRANCH IF NO NEED TO CHECK FOR PAINTABLE DATA
2138   EC7B 3Ø 1E               LEAX  $-Ø2,X                        MOVE HORIZONTAL COORD TWO PIXELS TO THE LEFT
2139   EC7D BD ED 15            JSR   LED15                         SAVE A BLOCK OF PAINT DATA IN THE DIRECTION OPPOSITE TO UP/DN FLAG
2140   EC8Ø BD ED Ø1    LEC8Ø   JSR   LEDØ1                        CONTINUE PAINTING LINE TO THE RIGHT
2141   EC83 BD ED 2E    LEC83   JSR   LED2E                        SAVE A BLOCK OF PAINT DATA IN THE SAME DIRECTION AS UP/DN FLAG
2142                           * THIS CODE WILL INSURE THAT THE CURRENT LINE IS
2143                           * EXAMINED TO THE RIGHT FOR PAINTABLE PIXELS FOR A
2144                           * LINE EQUAL TO THE LENGTH OF THE PARENT LINE
2145   EC86 43          LEC86   COMA                                *
2146   EC87 53                  COMB                                * COMPLEMENT LENGTH OF LINE JUST PAINTED
2147   EC88 D3 D1       LEC88   ADDD  VD1                           ADD TO LENGTH OF PARENT LINE
2148   EC8A DD D1               STD   VD1                           SAVE DIFFERENCE OF LINE JUST PAINTED AND PARENT LINE
2149   EC8C 2F 17               BLE   LECA5                         BRANCH IF PARENT LINE IS SHORTER
2150   EC8E BD E9 B1            JSR   LE9B1                         GO INCR HORIZONTAL COORD
2151   EC91 BD EC F1            JSR   LECF1                         CHECK FOR BORDER COLOR
2152   EC94 26 Ø5               BNE   LEC9B                         BRANCH IF NOT BORDER COLOR
2153   EC96 CC FF FF            LDD   #-1                           * GO DECREMENT ONE FROM LENGTH OF DIFFERENCE
2154   EC99 2Ø ED               BRA   LEC88                         * LINE AND KEEP LOOKING FOR NON BORDER COLOR
2155   EC9B BD E9 BF    LEC9B   JSR   LE9BF                         GET DECR HORIZONTAL COORD
2156   EC9E BD ED 3A            JSR   LED3A                         GET AND SAVE HORIZONTAL COORD
2157   ECA1 8D 24               BSR   LECC7                         PAINT FORWARD TO MAX HORIZONTAL COORD OR BORDER
2158   ECA3 2Ø DE               BRA   LEC83                         SAVE BLOCK OF PAINT DATA AND KEEP CHECKING
2159                           *
2160                           * CHECK TO SEE IF THE CURRENT LINE EXTENDS FURTHER TO
2161                           * THE RIGHT THAN THE PARENT LINE AND PUT A BLOCK OF
2162                           * PAINT DATA ON THE STACK IF IT IS MORE THAN 2 PIXELS
2163                           * PAST THE END OF THE PARENT LINE
2164   ECA5 BD E9 B1    LECA5   JSR   LE9B1                         INC CURRENT HORIZONTAL COORD
2165   ECA8 3Ø 8B               LEAX  D,X                           POINT X TO THE RIGHT END OF THE PARENT LINE
2166   ECAA 9F BD               STX   HORBEG                        SAVE AS THE CURRENT HORIZONTAL COORDINATE
2167   ECAC 43                  COMA                                = ACCA CONTAINS A NEGATIVE NUMBER CORRESPONDING TO THE NUMBER
2168   ECAD 53                  COMB                                = OF PIXELS THE CURRENT LINE EXTENDS PAST THE RIGHT END
2169   ECAE 83 ØØ Ø1            SUBD  #1                            = OF THE PARENT LINE. CONVERT TO POSITIVE NUMBER AND BRANCH
2170   ECB1 2F Ø4               BLE   LECB7                         = IF THE LINE DOESN'T EXTEND PAST THE END OF THE PARENT.
2171   ECB3 1F Ø1               TFR   D,X                           SAVE PORTION OF THE LINE TO THE RIGHT OF THE PARENT LINE
2172                                                                AS THE LENGTH
2173   ECB5 8D 5E               BSR   LED15                         SAVE BLOCK OF PAINT DATA IN THE DIRECTION OPPOSITE THE
2174                                                                CURRENT UP/DN FLAG
2175   ECB7 7E EC 4A    LECB7   JMP   LEC4A                         PROCESS MORE PAINT DATA BLOCKS
2176   ECBA BD E Ø FF   LECBA   JSR   SELTASKØ                      ENABLE TASK REGISTER Ø
2177   ECBD 39                  RTS
2178
2179                           * PAINT FROM HORIZONTAL COORD TO ZERO OR HIT BORDER; RETURN WITH Z=1 IF NO PAINTING DONE
2180   ECBE BD ED 3A    LECBE   JSR   LED3A                         PUT STARTING COORD IN HOREND
2181   ECC1 1Ø 8E E9 BF         LDY   #LE9BF                        ROUTINE TO DEC HORIZONTAL ADDRESS
2182   ECC5 2Ø Ø6               BRA   LECCD                         GO PAINT THE LINE
2183                           * PAINT FROM HORIZONTAL COORD TO MAX HORIZONTAL COORD OR HIT BORDER; RETURN Z=1 IF NO PAINTING DONE
2184   ECC7 1Ø 8E E9 B1 LECC7   LDY   #LE9B1                        ROUTINE TO INCR HORIZONTAL COORD
2185   ECCB AD A4               JSR   ,Y                            INCR HORIZONTAL COORD - LEFT PAINT ROUTINE PAINTED FIRST COORD
2186   ECCD DE 8A       LECCD   LDU   ZERO                          ZERO INITIAL PIXEL PAINT COUNTER
2187   ECCF 9E BD               LDX   HORBEG                        GET HORIZONTAL COORD
2188   ECD1 2B 17       LECD1   BMI   LECEA                         BRANCH IF HORIZONTAL COORD IS > $7F OR < Ø
2189   ECD3 9C D3               CMPX  VD3                           COMPARE CURRENT COORD TO MAX VALUE
2190   ECD5 22 13               BHI   LECEA                         BRANCH IF > MAX
2191   ECD7 34 6Ø               PSHS  U,Y                           SAVE PAINT COUNTER AND INC/DEC POINTER
2192   ECD9 8D 16               BSR   LECF1                         CHECK FOR BORDER PIXEL
2193   ECDB 27 ØB               BEQ   LECE8                         BRANCH IF HIT BORDER
2194   ECDD BD E7 92            JSR   LE792                         SET PIXEL TO PAINT COLOR - PAINTING IS DONE HERE
2195   ECEØ 35 6Ø               PULS  Y,U                           RESTORE PAINT COUNTER AND INC/DEC POINTER
2196   ECE2 33 41               LEAU  $Ø1,U                         ADD ONE TO PAINT COUNTER
2197   ECE4 AD A4               JSR   ,Y                            INCR OR DECR HORIZONTAL COORD DEPENDING ON CONTENTS OF Y
2198   ECE6 2Ø E9               BRA   LECD1                         KEEP PAINTING LINE
2199   ECE8 35 6Ø       LECE8   PULS  Y,U                           RESTORE PAINT COUNTER AND INC/DEC POINTER
2200   ECEA 1F 3Ø       LECEA   TFR   U,D                           SAVE PAINT COUNTER IN ACCD
2201   ECEC 1F Ø1               TFR   D,X                           ALSO SAVE IT IN X
2202   ECEE 93 8A               SUBD  ZERO                          SET COUNTERS ACCORDING TO CONDITION OF PAINT COUNTER
2203   ECFØ 39                  RTS
2204                           * CHECK FOR BORDER COLOR - ENTER WITH VD9 CONTAINING
2205                           * ADDRESS OF ROUTINE TO GET ABSOLUTE SCREEN ADDRESS
2206                           * AND PIXEL MASK - EXIT WITH Z=1 IF HIT BORDER COLOR PIXEL
2207   ECF1 AD 9F ØØ D9 LECF1   JSR   [VD9]                         GET SCREEN ADDRESS AND PIXEL MASK
2208   ECF5 1F 89               TFR   A,B                           COPY PIXEL MASK IN ACCB
```

```
2209  ECF7 D4 D8                ANDB  VD8                         AND PIXEL MASK WITH BORDER COLOR
2210  ECF9 34 06                PSHS  B,A                         SAVE MASK AND BORDER PIXEL
2211  ECFB A4 84                ANDA  ,X                          TEST THE PIXEL ON THE SCREEN
2212  ECFD A1 61                CMPA  $01,S                       COMPARE WITH ACCB ON THE STACK
2213  ECFF 35 86                PULS  A,B,PC                      EXIT WITH Z FLAG=1 IF MATCH
2214                    * GO HERE TO FINISH PAINTING TO RIGHT AFTER YOU HAVE PAINTED LEFT
2215  ED01 DD CD        LED01   STD   VCD                         SAVE NUMBER OF PIXELS PAINTED
2216  ED03 10 9E C3             LDY   HOREND                      GET LAST HORIZONTAL START COORD
2217  ED06 8D 32                BSR   LED3A                       SAVE CURRENT HORIZONTAL COORD - HOREND NOW CONTAINS COORDINATE
2218                                                              OF THE LEFT BORDER OF THIS HORIZONTAL LINE
2219  ED08 10 9F BD             STY   HORBEG                      START PAINTING TO RIGHT FROM THE LEFT PAINT START COORD
2220  ED0B 8D BA                BSR   LECC7                       PAINT TOWARDS THE RIGHT
2221  ED0D 9E CD                LDX   VCD                         GET THE NUMBER OF PIXELS PAINTED WHEN GOING TOWARDS LEFT PIXELS
2222  ED0F 30 8B                LEAX  D,X                         ADD NUMBER OF PAINTED GOING TOWARD THE RIGHT
2223  ED11 C3 00 01             ADDD  #1                          ADD 1 TO PAINT COUNT TOWARD RIGHT - ACCD=LENGTH OF PAINTED LINE
2224  ED14 39                   RTS
2225                    * BLOCKS OF DATA ARE STORED ON THE STACK SO THAT HPAINT
2226                    * CAN REMEMBER WHERE IT SHOULD GO BACK AND PAINT UP OR DOWN
2227                    * FROM THE CURRENT LINE IT IS PAINTING. THESE BLOCKS OF DATA
2228                    * REPRESENT HORIZONTAL LINES ABOVE OR BELOW THE CURRENT LINE
2229                    * BEING PAINTED AND REQUIRE SIX BYTES OF STORAGE ON THE STACK.
2230                    * THE DATA ARE AS FOLLOWS: ,S=UP/DN FLAG; 1,S=VERTICAL COORD
2231                    * OF LINE; 2 3,S=LEFT MOST HORIZONTAL COORD OF LINE; 4 5,S=LENGTH OF LINE
2232
2233                    * SAVE A BLOCK OF PAINT DATA FOR A LINE IN THE OPPOSITE DIRECTION OF THE CURREN UP/DN FLAG
2234  ED15 DD CB        LED15   STD   VCB                         SAVE NUMBER OF PIXELS PAINTED
2235  ED17 35 20                PULS  Y                           GET RETURN ADDRESS IN Y
2236  ED19 DC BD                LDD   HORBEG                      GET HORIZONTAL START COORD
2237  ED1B 34 16                PSHS  X,B,A                       PUT ON STACK
2238  ED1D 96 D7                LDA   VD7                         GET UP/DN FLAG
2239  ED1F 40                   NEGA                              REVERSE IT
2240  ED20 D6 C0        LED20   LDB   VERBEG+1                    GET VERTICAL START COORDINATE
2241  ED22 34 06                PSHS  B,A                         SAVE VERTICAL START COORD AND UP/DN FLAG
2242  ED24 34 20                PSHS  Y                           PUT BACK RETURN ADDRESS
2243  ED26 C6 06                LDB   #$06                        GET NUMBER OF FREE BYTES TO CHECK FOR
2244  ED28 BD ED 3F             JSR   LED3F                       GO SEE IF THERE IS ENOUGH RAM
2245  ED2B DC CB                LDD   VCB                         GET LENGTH OF RIGHT PAINTED LINE
2246  ED2D 39                   RTS
2247
2248                    * SAVE A BLOCK OF PAINT DATA FOR A LINE IN THE SAME DIRECTION AS THE CURRENT UP/DN FLAG
2249  ED2E DD CB        LED2E   STD   VCB                         SAVE THE LENGTH OF RIGHT HORIZONTAL PAINTED LINE
2250  ED30 35 20                PULS  Y                           SAVE RETURN ADDRESS IN Y
2251  ED32 DC C3                LDD   HOREND                      GET HORIZONTAL START COORD
2252  ED34 34 16                PSHS  X,B,A                       SAVE START COORD AND LENGTH
2253  ED36 96 D7                LDA   VD7                         GET UP/DN FLAG (1 OR -1)
2254  ED38 20 E6                BRA   LED20                       SAVE THE PAINT DATA ON THE STACK
2255  ED3A 9E BD        LED3A   LDX   HORBEG                      GET CURRENT HORIZONTAL COORD
2256  ED3C 9F C3                STX   HOREND                      SAVE IT
2257  ED3E 39                   RTS
2258
2259                    * CHECK ACCB (ONLY 0-127) BYTES OF FREE RAM ON THE STACK
2260  ED3F 50           LED3F   NEGB
2261  ED40 32 E5                LEAS  B,S                         MOVE THE STACK POINTER DOWN ACCB BYTES
2262  ED42 11 8C BF F1          CMPS  #TMPSTACK-($2000+14)        COMPARE TO THE BOTTOM OF THE STACK AREA - THE 14 EXTRA BYTES ARE
2263                                                              GENERATED BY THE FACT THAT THE SEVEN INTERRUPT VECTORS ARE GOTTEN FROM
2264                                                              THE ROM BY THE GIME CHIP.  THE 14 BYTES IN RAM ARE UNUSED BY BASIC.
2265  ED46 10 25 00 04          LBCS  LED4E                       'OM' ERROR IF PAST THE BOTTOM
2266  ED4A 50                   NEGB                              MAKE ACCB POSITIVE AGAIN
2267  ED4B 32 E5                LEAS  B,S                         PUT THE STACK POINTER BACK WHERE IT BELONGS
2268  ED4D 39                   RTS
2269  ED4E 10 CE DF FD  LED4E   LDS   #TMPSTACK-2                 PUT THE STACK POINTER AT THE TOP OF THE TEMPORARY STACK BUFFER
2270  ED52 BD E0 FF             JSR   SELTASK0                    ENABLE TASK REGISTER 0
2271  ED55 7E AC 44             JMP   LAC44                       GO DO AN 'OM' ERROR
2272
2273                    * HBUFF
2274                    * THE HBUFF COMMAND WILL RESERVE SPACE IN THE HPUT/HGET BUFFER.  THERE MUST BE ENOUGH FREE RAM
2275                    * IN THE BUFFER FOR THE REQUESTED BUFFER SIZE AND A FIVE BYTE HEADER.  EACH BUFFER HAS A FIVE BYTE
2276                    * HEADER WHICH IS DESCRIBED AS FOLLOWS:
2277                    * BYTES 0,1: ADDRESS OF THE NEXT HPUT/HGET BUFFER IN THE BUFFER SPACE.  IF ZERO, THERE ARE
2278                    * NO MORE BUFFERS IN THE BUFFER SPACE.  IF $FFFF, THEN THERE ARE NO
2279                    * BUFFERS ALLOCATED AND THE ENTIRE BUFFER SPACE IS FREE.
2280                    * BYTE 2: BUFFER NUMBER; BYTES 3,4: SIZE OF THE BUFFER
2281
2282                    * HBUFF
2283  ED58 BD B7 3D     HBUFF   JSR   LB73D                       EVALUATE BUFFER NUMBER ARGUMENT; RETURN VALUE IN X
2284  ED5B 10 21 12 A1          LBRN  RAMLINK                     RAM HOOK
2285  ED5F 8C 00 FF             CMPX  #255                        MAXIMUM OF 255 BUFFERS ALLOWED
2286  ED62 10 22 C6 E4          LBHI  ILLFUNC                     ILLEGAL FUNCTION CALL ERROR IF BUFFER NUMBER > 255
2287  ED66 9F D1                STX   VD1                         SAVE THE BUFFER NUMBER
2288  ED68 27 08                BEQ   LED72                       DON'T GET THE SIZE OF THE BUFFER IF BUFFER 0 SELECTED
2289  ED6A BD B2 6D             JSR   SYNCOMMA                    DO A SYNTAX CHECK FOR A COMMA
2290  ED6D BD B7 3D             JSR   LB73D                       EVALUATE THE BUFFER SIZE ARGUMENT
2291  ED70 9F D3                STX   VD3                         SAVE THE BUFFER SIZE
2292  ED72 BD E0 CB     LED72   JSR   LE0CB                      PUT BLOCK 6.4 INTO LOGICAL BLOCK 6 ($C000) OF TASK REGISTER 1
2293  ED75 BD E1 19             JSR   SELTASK1                    ENABLE TASK REGISTER 1
2294  ED78 DC D1                LDD   VD1                         GET THE NEW BUFFER NUMBER
2295  ED7A 5D                   TSTB                              CHECK FOR BUFFER ZERO
2296  ED7B 26 08                BNE   LED85                       BRANCH IF NOT BUFFER ZERO
2297  ED7D CC FF FF             LDD   #$FFFF                      EMPTY BUFFER FLAG
2298  ED80 FD C0 00             STD   HRESBUFF                    RESET BUFFER SPACE TO EMPTY
2299  ED83 20 38                BRA   LEDBD                       EXIT COMMAND
2300  ED85 10 8E C0 00  LED85   LDY   #HRESBUFF                   POINT TO THE START OF THE BUFFER SPACE
2301  ED89 EC A4                LDD   ,Y                          GET THE FIRST TWO BYTES OF THE HEADER BLOCK (HB.ADDR)
2302  ED8B 10 83 FF FF          CMPD  #$FFFF                      IS THE BUFFER EMPTY?
2303  ED8F 26 04                BNE   LED95                       NO; CHECK FOR FIRST EMPTY HEADER SPOT
2304  ED91 8D 31                BSR   LEDC4                       CHECK FOR ENOUGH FREE RAM IN THE BUFFER SPACE FOR THIS BUFFER
```

```
2305   ED93 20 1B                 BRA   LEDB0
2306   ED95 D6 D2          LED95  LDB   VD1+1                    GET THE BUFFER NUMBER
2307   ED97 E1 22          LED97  CMPB  $02,Y                    COMPARE TO THE BUFFER NUMBER OF THE CURRENT HEADER (HB.NUM)
2308   ED99 27 37                 BEQ   LEDD2                    RE-DIMENSIONED ARRAY (DD) ERROR IF THE SAME
2309   ED9B EE A4                 LDU   ,Y                       GET THE ADDRESS OF THE NEXT BUFFER
2310   ED9D 27 04                 BEQ   LEDA3                    BRANCH IF THIS IS THE LAST HEADER
2311   ED9F 1F 32                 TFR   U,Y                      POINT Y TO THE START OF THE NEXT BUFFER HEADER
2312   EDA1 20 F4                 BRA   LED97                    KEEP SEARCHING FOR THE LAST HEADER
2313   EDA3 1F 23          LEDA3  TFR   Y,U                      SAVE THE START ADDRESS OF THE LAST HEADER IN U
2314   EDA5 EC 23                 LDD   $03,Y                    GET THE SIZE OF THE LAST HEADER (HB.SIZE)
2315   EDA7 31 25                 LEAY  $05,Y                    SKIP PAST THE HEADER DATA (HB.LEN)
2316   EDA9 31 AB                 LEAY  D,Y                      NOW Y POINTS TOT THE START OF FREE BUFFER SPACE
2317   EDAB 8D 17                 BSR   LEDC4                    CHECK FOR ENOUGH FREE RAM IN THE BUFFER SPACE FOR THIS BUFFER
2318   EDAD 10 AF C4              STY   ,U                       SAVE THE ADDRESS OF THIS HEADER IN THE PREVIOUS HEADER
2319   EDB0 CC 00 00       LEDB0  LDD   #0                       LAST ENTRY FLAG
2320   EDB3 ED A4                 STD   ,Y                       MAKE THIS HEADER THE LAST ENTRY
2321   EDB5 D6 D2                 LDB   VD1+1                    GET THE BUFFER NUMBER AND
2322   EDB7 E7 22                 STB   $02,Y                    SAVE IT IN THE HEADER
2323   EDB9 DC D3                 LDD   VD3                      GET THE SIZE OF THE BUFFER AND
2324   EDBB ED 23                 STD   $03,Y                    SAVE IT IN THE HEADER TOO
2325   EDBD BD E0 FF       LEDBD  JSR   SELTASK0                 ENABLE TASK REGISTER 0
2326   EDC0 BD E0 97              JSR   SETMMU                   RESET MMU REGISTERS
2327   EDC3 39                    RTS
2328   EDC4 1F 21          LEDC4  TFR   Y,X                      USE X A TEMPORARY POINTER TO THE START OF BUFFER
2329   EDC6 30 05                 LEAX  $05,X                    SKIP PAST THE HEADER
2330   EDC8 DC D3                 LDD   VD3                      GET THE SIZE OF THE BUFFER AND
2331   EDCA 30 8B                 LEAX  D,X                      ADD IT TO THE BUFFER POINTER WHICH NOW POINTS TO THE BUFFER END
2332   EDCC 8C DF 00              CMPX  #HRESBUFF+$1F00          PAST THE END OF THE BUFFER SPACE?
2333   EDCF 22 05                 BHI   LEDD6                    'OM' ERROR IF PAST END OF BUFFER SPACE
2334   EDD1 39                    RTS
2335   EDD2 C6 12          LEDD2  LDB   #9*2                     REDIMENSIONED ARRAY ERROR (DD)
2336   EDD4 20 02                 BRA   LEDD8
2337   EDD6 C6 0C          LEDD6  LDB   #6*2                     OUT OF MEMORY ERROR (OM)
2338   EDD8 10 CE DF FD    LEDD8  LDS   #TMPSTACK-2              RESET THE STACK TO TEMPORARY LOCATON
2339   EDDC BD E0 FF              JSR   SELTASK0                 ENABLE TASK REGISTER 0
2340   EDDF BD E0 97              JSR   SETMMU                   RESET THE MMU REGISTERS
2341   EDE2 7E AC 46              JMP   LAC46                    JUMP TO THE ERROR HANDLER
2342
2343                       * HGET
2344   EDE5 8E EE C0       HGET   LDX   #LEEC0                   POINT X TO THE HGET MOVEMENT ROUTINE
2345   EDE8 9F D5                 STX   VD5                      SAVE THE MOVEMENT ROUTINE ADDRESS
2346   EDEA 5F                    CLRB                           HGET FLAG
2347   EDEB 20 07                 BRA   LEDF4
2348
2349                       * HPUT
2350   EDED 8E EE EF       HPUT   LDX   #LEEEF                   HPUT MOVEMENT ROUTINE ADDRESS
2351   EDF0 9F D5                 STX   VD5                      SAVE THE MOVEMENT ROUTINE ADDRESS
2352   EDF2 C6 01                 LDB   #$01                     HPUT FLAG
2353   EDF4 0D E6          LEDF4  TST   HRMODE                   IS THE HI-RES GRAPHICS MODE ENABLED?
2354   EDF6 10 27 F8 F5           LBEQ  LE6EF                    'HR' ERROR IF NOT IN HI-RES MODE
2355   EDFA 10 21 12 02           LBRN  RAMLINK                  RAM HOOK
2356   EDFE D7 D8                 STB   VD8                      SAVE THE GET/PUT FLAG
2357   EE00 81 40                 CMPA  #'@'                     ALLOW HGET@, HPUT@ AS LEGAL SYNTAX
2358   EE02 26 02                 BNE   LEE06                    BRANCH IF NOT @
2359   EE04 9D 9F                 JSR   GETNCH                   GET THE NEXT CHARACTER FROM BASIC'S INPUT LINE IF @ WAS THERE
2360   EE06 BD E9 E1       LEE06  JSR   LE9E1                    EVALUATE THE RECTANGLE BOUNDS
2361   EE09 BD B2 6D              JSR   SYNCOMMA                 SYNTAX CHECK FOR COMMA
2362   EE0C BD B7 0B              JSR   EVALEXPB                 EVALUATE EXPRESSION, RETURN VALUE IN ACCB
2363   EE0F D7 D3                 STB   VD3                      SAVE THE BUFFER NUMBER
2364   EE11 0F D4                 CLR   VD4                      SET THE ACTION FLAG TO SHOW AN ACTION SPECIFIED
2365   EE13 9D A5                 JSR   GETCCH                   GET BASIC'S CURRENT INPUT CHARACTER
2366   EE15 27 21                 BEQ   LEE38                    BRANCH IF END OF LINE - NO ACTION SPECIFIED
2367   EE17 03 D4                 COM   VD4                      SET THE ACTION FLAG TO SHOW THAT AN ACTION WAS SPECIFIED
2368   EE19 BD B2 6D              JSR   SYNCOMMA                 SYNTAX CHECK FOR COMMA
2369   EE1C 0D D8                 TST   VD8                      CHECK THE GET/PUT FLAG
2370   EE1E 26 03                 BNE   LEE23                    BRANCH IF PUT
2371   EE20 16 C4 54              LBRA  LB277                    'SYNTAX' ERROR - HGET MAY NOT SPECIFY AN ACTION
2372   EE23 C6 05          LEE23  LDB   #$05                     FIVE POSSIBLE ACTIONS
2373   EE25 8E EE E0              LDX   #LEEE0                   POINT TO THE ACTION ROUTINE ADDRESS
2374   EE28 EE 81          LEE28  LDU   ,X++                     GET THE ACTION ROUTINE ADDRESS
2375   EE2A A1 80                 CMPA  ,X+                      COMPARE THE DESIRED ACTION TO THIS ROUTINE'S TOKEN
2376   EE2C 27 06                 BEQ   LEE34                    SEARCH NO MORE - A MATCH WAS FOUND
2377   EE2E 5A                    DECB                           DECREMENT COUNTER
2378   EE2F 26 F7                 BNE   LEE28                    LOOP UNTIL ALL ACTIONS CHECKED
2379   EE31 7E B2 77              JMP   LB277                    'SYNTAX' ERROR IF ILLEGAL ACTION DESIRED
2380   EE34 DF D5          LEE34  STU   VD5                      SAVE THE ACTION ADDRESS
2381   EE36 9D 9F                 JSR   GETNCH                   GET A CHARACTER FROM BASIC'S INPUT LINE
2382   EE38 BD E0 CB       LEE38  JSR   LE0CB                    PUT THE HBUFF BLOCK INTO LOGICAL ADDRESS SPACE OF TASK REGISTER 1
2383   EE3B BD E1 19              JSR   SELTASK1                 SELECT TASK REGISTER 1
2384   EE3E D6 D3                 LDB   VD3                      GET THE BUFFER NUMBER
2385   EE40 BD EF 18              JSR   LEF18                    GET THE START AND END OF THIS BUFFER'S DATA
2386   EE43 DC BD                 LDD   HORBEG                   GET THE STARING HORIZONTAL COORDINATE
2387   EE45 10 93 C3              CMPD  HOREND                   COMPARE IT TO THE ENDING COORDINATE
2388   EE48 2F 06                 BLE   LEE50                    BRANCH IF START <= END COORDINATE
2389   EE4A 9E C3                 LDX   HOREND                   GET THE ENDING COORDINATE
2390   EE4C 9F BD                 STX   HORBEG                   SAVE IT AS THE STARTING COORDINATE
2391   EE4E DD C3                 STD   HOREND                   NOW SAVE THE STARTING COORDINATE AS THE ENDING COORDINATE
2392   EE50 DC BF          LEE50  LDD   VERBEG                   GET THE VERTICAL STARTING COORDINATE
2393   EE52 10 93 C5              CMPD  VEREND                   COMPARE IT TO THE ENDING COORDINATE
2394   EE55 2F 06                 BLE   LEE5D                    BRANCH IF START <= END COORDINATE
2395   EE57 9E C5                 LDX   VEREND                   GET THE ENDING COORDINATE
2396   EE59 9F BF                 STX   VERBEG                   SAVE IT AS THE STARTING COORDINATE
2397   EE5B DD C5                 STD   VEREND                   NOW SAVE THE STARTING COORDINATE AS THE ENDING COORDINATE
2398                       * ROUND OFF THE HORIZONTAL START AND END COORDINATES TO AN EVEN NUMBER OF BYTES
2399   EE5D 96 E6          LEE5D  LDA   HRMODE                   GET THE HI-RES GRAPHICS MODE
2400   EE5F C6 F8                 LDB   #$F8                     ROUND OFF MASK FOR HSCREEN 3 (EIGHT PIXELS PER BYTE)
```

```
2401   EE61 81 Ø3                  CMPA  #$Ø3                          HSCREEN 3?
2402   EE63 27 Ø8                  BEQ   LEE6D
2403   EE65 C6 FC                  LDB   #$FC                          ROUND OFF MASK FOR HSCREEN 1 OR 4 (FOUR PIXELS PER BYTE)
2404   EE67 81 Ø2                  CMPA  #$Ø2                          HSCREEN 2?
2405   EE69 26 Ø2                  BNE   LEE6D                         NO IT'S HSCREEN 1 OR 4
2406   EE6B C6 FE                  LDB   #$FE                          ROUND OFF MASK FOR HSCREEN 2 (TWO PIXELS PER BYTE)
2407   EE6D 1F 98           LEE6D  TFR   B,A                           SAVE MASK IN BOTH ACCA AND ACCB
2408   EE6F 94 BE                  ANDA  HORBEG+1                      ROUND OFF HORIZONTAL START COORDINATE
2409   EE71 97 BE                  STA   HORBEG+1                      SAVE NEW START COORDINATE
2410   EE73 D4 C4                  ANDB  HOREND+1                      ROUND OFF HORIZONTAL END COORDINATE
2411   EE75 D7 C4                  STB   HOREND+1                      SAVE NEW END COORDINATE
2412   EE77 BD E9 DB               JSR   LE9DB                         CALCULATE THE DIFFERENCE BETWEEN THE HORIZONTAL START AND END
2413   EE7A DD C3                  STD   HOREND                        SAVE THE HORIZONTAL DIFFERENCE
2414   EE7C BD E9 CD               JSR   LE9CD                         CALCULATE THE DIFFERENCE BETWEEN THE VERTICAL START AND END
2415   EE7F C3 Ø0 Ø1               ADDD  #1                            ADD ONE TO THE VERTICAL DIFFERENCE (INCLUSIVE START AND END)
2416   EE82 DD C5                  STD   VEREND                        SAVE THE VERTICAL DIFFERENCE
2417
2418                        * CONVERT THE HORIZONTAL DIFFERENCE (IN PIXELS) INTO A BYTE DIFFERENCE
2419   EE84 96 E6                  LDA   HRMODE                        GET THE HI-RES GRAPHICS MODE
2420   EE86 81 Ø2                  CMPA  #$Ø2                          HSCREEN 2?
2421   EE88 27 ØC                  BEQ   LEE96                         YES; DIVIDE PIXEL COUNT BY TWO (TWO PIXELS PER BYTE)
2422   EE8A 81 Ø3                  CMPA  #$Ø3                          HSCREEN 3?
2423   EE8C 26 Ø4                  BNE   LEE92                         NO; DIVIDE PIXEL COUNT BY FOUR (FOUR PIXELS PER BYTE)
2424   EE8E Ø4 C3                  LSR   HOREND                        * HSCREEN 3; DIVIDE PIXEL COUNT BY EIGHT (EIGHT PIXELS PER BYTE)
2425   EE9Ø Ø6 C4                  ROR   HOREND+1                      DIVIDE THE HORIZONTAL DIFFERENCE BY 2
2426   EE92 Ø4 C3           LEE92  LSR   HOREND
2427   EE94 Ø6 C4                  ROR   HOREND+1                      DIVIDE THE HORIZONTAL DIFFERENCE BY 2
2428   EE96 Ø4 C3           LEE96  LSR   HOREND
2429   EE98 Ø6 C4                  ROR   HOREND+1                      DIVIDE THE HORIZONTAL DIFFERENCE BY 2
2430   EE9A DC C3                  LDD   HOREND
2431   EE9C C3 Ø0 Ø1               ADDD  #1                            ADD ONE TO THE HORIZONTAL DIFFERENCE (INCLUSIVE START AND END)
2432   EE9F DD C3                  STD   HOREND                        SAVE THE HORIZONTAL DIFFERENCE
2433   EEA1 BD E7 DA               JSR   HCALPOS                       POINT X TO THE FIRST BYTE TO MOVE
2434   EEA4 1Ø 9E D5               LDY   VD5                           POINT Y TO THE ACTION ADDRESS
2435   EEA7 D6 C4           LEEA7  LDB   HOREND+1                      GET THE LS BYTE OF HORIZONTAL DIFFERENCE
2436   EEA9 34 1Ø                  PSHS  X                             SAVE THE MOVEMENT POINTER
2437   EEAB AD A4           LEEAB  JSR   ,Y                            PERFORM THE APPROPRIATE MOVEMENT ACTION
2438   EEAD 5A                     DECB                                DECREMENT THE HORIZONTAL MOVEMENT COUNTER
2439   EEAE 26 FB                  BNE   LEEAB                         LOOP UNTIL ALL BYTES ON THIS ROW MOVED
2440   EEBØ 35 1Ø                  PULS  X                             RESTORE THE MOVEMENT POINTER
2441   EEB2 BD EA 45               JSR   LEA45                         MOVE THE MOVEMENT POINTER DOWN ONE ROW
2442   EEB5 ØA C6                  DEC   VEREND+1                      DECREMENT THE VERTICAL DIFFERENCE (ROW COUNTER)
2443   EEB7 26 EE                  BNE   LEEA7                         LOOP UNTIL ALL ROWS MOVED
2444   EEB9 BD E Ø FF              JSR   SELTASKØ                      SELECT TASK REGISTER Ø AS THE ACTIVE TASK
2445   EEBC BD E Ø 97              JSR   SETMMU                        SET UP THE MMU REGISTERS
2446   EEBF 39                     RTS                                 WHY NOT MAKE THE JSR ABOVE A JMP
2447
2448                        * HGET'S BYTE MOVEMENT ROUTINE
2449   EEC Ø A6 8Ø          LEECØ  LDA   ,X+                           GET A BBYTE FROM THE HI-RES SCREEN
2450   EEC2 8D Ø3                  BSR   LEEC7                         POINT U TO PROPER BUFFER LOCATION
2451   EEC4 A7 C4                  STA   ,U                            SAVE THE BYTE IN THE BUFFER
2452   EEC6 39                     RTS
2453   EEC7 DE CF           LEEC7  LDU   VCF                           GET THE BUFFER POINTER
2454   EEC9 33 41                  LEAU  $Ø1,U                         BUMP IT UP BY ONE
2455   EECB DF CF                  STU   VCF                           SAVE IT
2456   EECD 11 93 D1               CMPU  VD1                           COMPARE THE NEW POINTER TO THE END OF THE BUFFER SPACE
2457   EED Ø 22 Ø1                 BHI   LEED3                         'FC' FUNCTION CALL ERROR IF PAST THE END OF THE BUFFER
2458   EED2 39                     RTS
2459   EED3 1Ø CE DF FD     LEED3  LDS   #TMPSTACK-2                   RESET THE TEMPORARY STACK POINTER
2460   EED7 BD E Ø FF              JSR   SELTASKØ                      SELECT TASK REGISTER Ø AS THE ACTIVE TASK
2461   EEDA BD E Ø 97              JSR   SETMMU                        SET UP THE MMU REGISTERS
2462   EEDD 7E B4 4A               JMP   ILLFUNC                       ILLEGAL FUNCTION CALL ERROR
2463
2464   EEE Ø EE EF          LEEEØ  FDB   LEEEF                         ADDRESS OF PSET ACTION ROUTINE
2465   EEE2 BD              LEEE2  FCB   $BD                           TOKEN FOR PSET
2466   EEE3 EE F6           LEEE3  FDB   LEEF6                         ADDRESS OF PRESET ACTION ROUTINE
2467   EEE5 BE              LEEE5  FCB   $BE                           TOKEN FOR PRESET
2468   EEE6 EF Ø7           LEEE6  FDB   LEFØ7                         ADDRESS OF OR ACTION ROUTINE
2469   EEE8 B1              LEEE8  FCB   $B1                           TOKEN FOR OR
2470   EEE9 EE FE           LEEE9  FDB   LEEFE                         ADDRESS OF AND ACTION ROUTINE
2471   EEEB BØ              LEEEB  FCB   $BØ                           TOKEN FOR AND
2472   EEEC EF 1Ø           LEEEC  FDB   LEF1Ø                         ADDRESS OF NOT ACTION ROUTINE
2473   EEEE A8              LEEEE  FCB   $A8                           TOKEN FOR NOT
2474
2475                        * HPUT'S MOVEMENT ROUTINES
2476                        * PSET (DEFAULT ROUTINE)
2477   EEEF 8D D6           LEEEF  BSR   LEEC7                         POINT U TO THE PROPER BUFFER LOCATION
2478   EEF1 A6 C4                  LDA   ,U                            GET A BYTE FROM THE BUFFER
2479   EEF3 A7 8Ø                  STA   ,X+                           PUT IT BACK ON THE SCREEN
2480   EEF5 39                     RTS
2481                        * PRESET
2482   EEF6 8D CF           LEEF6  BSR   LEEC7                         POINT U TO THE PROPER BUFFER LOCATION
2483   EEF8 A6 C4                  LDA   ,U                            GET A BYTE FROM THE BUFFER
2484   EEFA 43                     COMA
2485   EEFB A7 8Ø                  STA   ,X+                           PUT IT BACK ON THE SCREEN
2486   EEFD 39                     RTS
2487                        * AND
2488   EEFE 8D C7           LEEFE  BSR   LEEC7                         POINT U TO THE PROPER BUFFER LOCATION
2489   EF ØØ A6 C4                 LDA   ,U                            GET A BYTE FROM THE BUFFER
2490   EF Ø2 A4 84                 ANDA  ,X                            'AND' IT WITH THE SCREEN DATA
2491   EF Ø4 A7 8Ø                 STA   ,X+                           PUT IT BACK ON THE SCREEN
2492   EF Ø6 39                    RTS
2493                        * OR
2494   EF Ø7 8D BE          LEFØ7  BSR   LEEC7                         POINT U TO THE PROPER BUFFER LOCATION
2495   EF Ø9 A6 C4                 LDA   ,U                            GET A BYTE FROM THE BUFFER
2496   EF ØB AA 84                 ORA   ,X                            'OR' IT WITH THE SCREEN DATA
```

```
2497   EF0D A7 80                      STA   ,X+                              PUT IT BACK ON THE SCREEN
2498   EF0F 39                         RTS
2499                           * NOT
2500   EF10 8D B5              LEF10   BSR   LEEC7                             POINT U TO THE PROPER BUFFER LOCATION
2501                           * THIS IS A MAJOR BUG - SHOULD BE LDA ,U
2502   EF12 A6 84                      LDA   ,X                                GET A BYTE FROM THE SCREEN, SHOULD BE FROM THE BUFFER
2503   EF14 43                         COMA                                    COMPLEMENT THE BYTE
2504   EF15 A7 80                      STA   ,X+                               PUT IT BACK ON THE SCREEN
2505   EF17 39                         RTS
2506
2507   EF18 10 8E C0 00        LEF18   LDY   #HRESBUFF                         POINT Y TO THE START OF THE BUFFER SPACE
2508   EF1C A6 A4                      LDA   ,Y                                GET THE FIRST BYTE
2509   EF1E 81 FF                      CMPA  #$FF                              ARE ANY BUFFERS ACTIVE?
2510   EF20 26 0A                      BNE   LEF2C                             YES, SEARCH FOR THE CORRECT BUFFER
2511   EF22 7E EE D3                   JMP   LEED3                             'FC' ERROR IF NO BUFFERS ACTIVE
2512   EF25 10 AE A4           LEF25   LDY   ,Y                                SKIP TO NEXT BUFFER
2513   EF28 10 27 FF A7                LBEQ  LEED3                             'FC' ERROR IF THERE ARE NO MORE ACTIVE BUFFERS
2514   EF2C E1 22              LEF2C   CMPB  $02,Y                             COMPARE THE DESIRED BUFFER TO THE CURRENT BUFFER NUMBER
2515   EF2E 26 F5                      BNE   LEF25                             NO, MATCH, CHECK THE NEXT BUFFER
2516   EF30 EC 23                      LDD   $03,Y                             GET THE SIZE OF THE SELECTED BUFFER
2517   EF32 31 24                      LEAY  $04,Y                             SKIP TO ONE BYTE BEFORE THE START BUFFER DATA
2518   EF34 10 9F CF                   STY   VCF                               SAVE THE START OF THE BUFFER DATA
2519   EF37 31 21                      LEAY  $01,Y                             MOVE TO THE ACTUAL START OF DATA
2520   EF39 31 AB                      LEAY  D,Y                               ADD IN THE SIZE OF THE DATA
2521   EF3B 10 9F D1                   STY   VD1                               SAVE THE ADDRESS OF THE END OF THE DATA
2522   EF3E 39                         RTS
2523
2524                           * HPRINT
2525   EF3F 0D E6              HPRINT  TST   HRMODE                             CHECK THE HI-RES GRAPHICS MODE
2526   EF41 10 27 F7 AA                LBEQ  LE6EF                             'HR' ERROR IF NOT HI-RES GRAPHICS MODE
2527   EF45 10 21 10 B7                LBRN  RAMLINK                           RAM HOOK
2528   EF49 BD B2 6A                   JSR   LB26A                             SYNTAX CHECK FOR '('
2529   EF4C BD E7 B2                   JSR   LE7B2                             EVALUATE HORIZONTAL AND VERTICAL COORDINATE
2530   EF4F BD B2 67                   JSR   LB267                             SYNTAX CHECK FOR ')'
2531   EF52 BD B2 6D                   JSR   SYNCOMMA                          SYNTAX CECK FOR COMMA
2532   EF55 BD B1 56                   JSR   LB156                             EVALUATE EXPRESSION
2533   EF58 0D 06                      TST   VALTYP                            CHECK THE TYPE OF VARIABLE EVALUATED
2534   EF5A 26 06                      BNE   LEF62                             BRANCH IF NOT NUMERIC - REALLY SHOULD BE BMI
2535   EF5C BD BD D9                   JSR   LBDD9                             CONVERT FLOATING POINT NUMBER INTO A STRING
2536   EF5F BD B5 16                   JSR   LB516                             SAVE THE STRING IN STRING SPACE
2537   EF62 BD B6 57           LEF62   JSR   LB657                             CALCULATE THE LENGTH AND ADDRESS OF THE STRING
2538   EF65 F7 FE 18                   STB   H.PCOUNT                          SAVE THE LENGTH OF THE STRING
2539   EF68 10 8E FE 19                LDY   #H.PBUF                           POINT TO THE HPRINT BUFFER
2540   EF6C 5A                 LEF6C   DECB                                    DECREMENT THE CHARACTER COUNT
2541   EF6D 2B 06                      BMI   LEF75                             BRANCH IF ALL CHARACTERS PRINTED
2542   EF6F A6 80                      LDA   ,X+                               GET A CHARACTER FROM THE STRING
2543   EF71 A7 A0                      STA   ,Y+                               SAVE IT IN THE HPRINT BUFFER
2544   EF73 20 F7                      BRA   LEF6C                             KEEP GOING UNTIL DONE
2545   EF75 96 E6              LEF75   LDA   HRMODE                            GET THE HI-RES GRAPHICS MODE
2546   EF77 C6 28                      LDB   #40                               40 CHARACTERS MAX IN THE 320 PIXEL WIDE MODE
2547   EF79 81 03                      CMPA  #$03                              CHECK THE HSCREEN MODE
2548   EF7B 25 02                      BCS   LEF7F                             BRANCH IF 40 COLUMN RESOLUTION
2549   EF7D C6 50                      LDB   #80                               80 CHARACTERS MAX IN THE 640 PIXEL WIDE MODE
2550   EF7F 4F                 LEF7F   CLRA                                    CLEAR THE MOST SIGNIFICANT BYTE OF ACCD
2551   EF80 93 BD                      SUBD  HORBEG                            SUBTRACT THE HORIZONTAL PRINT POSITION
2552   EF82 2B 7D                      BMI   LF001                             EXIT IF HORIZONTAL PRINT POSITION > LINE LENGTH
2553   EF84 F1 FE 18                   CMPB  H.PCOUNT                          IS THE PRINT CHARACTER COUNT > LINE LENGTH?
2554   EF87 22 05                      BHI   LEF8E                             BRANCH IF NOT
2555   EF89 F7 FE 18                   STB   H.PCOUNT                          FORCE THE PRINT CHARACTER COUNT TO EQUAL THE LINE LENGTH
2556   EF8C 27 73                      BEQ   LF001                             EXIT IF LINE LENGTH = 0
2557   EF8E 86 17              LEF8E   LDA   #ROWMAX-1                         GET THE HIGHEST POSSIBLE ROW NUMBER
2558   EF90 91 C0                      CMPA  VERBEG+1                          AND COMPARE IT TO THE PRINT ROW
2559   EF92 2C 02                      BGE   LEF96                             BRANCH IF PRINTING ON A LEGAL ROW NUMBER
2560   EF94 97 C0                      STA   VERBEG+1                          PRINT ON BOTTOM ROW (HIGHEST NUMBER) IF ILLEGAL ROW SPECIFIED
2561   EF96 BD F0 8C           LEF96   JSR   LF08C                             ADJUST ROW AND COLUMN NUMBERS FOR PRINTING ON HI-RES SCREEN
2562   EF99 BD E7 DA                   JSR   HCALPOS                           POINT X TO THE SCREEN ADDRESS; ACCA = PIXEL MASK
2563   EF9C 10 8E FE 19                LDY   #H.PBUF                           POINT TO THE HPRINT BUFFER
2564   EFA0 F6 FE 18                   LDB   H.PCOUNT                          GET THE NUMBER OF CHARACTERS IN THE PRINT BUFFER
2565   EFA3 A6 A4              LEFA3   LDA   ,Y                                GET A CHARACTER FROM THE PRINT BUFFER
2566   EFA5 84 7F                      ANDA  #$7F                              MASK OFF THE GRAPHICS BIT (BIT 7)
2567   EFA7 80 20                      SUBA  #$20                              SUBTRACT OUT THE CONTROL CODES
2568   EFA9 2A 02                      BPL   LEFAD                             BRANCH IF IT WAS NOT A CONTROL CODE
2569   EFAB 86 00                      LDA   #$00                              FORCE A CONTROL CODE TO PRINT A BLANK
2570   EFAD A7 A0              LEFAD   STA   ,Y+                               PUT THE 'MASSAGED' CHARACTER BACK INTO THE BUFFER
2571   EFAF 5A                         DECB                                    BUMP CHARACTER DOWN ONE
2572   EFB0 2E F1                      BGT   LEFA3                             LOOP UNTIL ALL CHARACTERS DONE
2573   EFB2 96 E6                      LDA   HRMODE                            GET THE HI-RES GRAPHICS MODE
2574   EFB4 4A                         DECA                                    CONVERT 1-4 TO 0-3
2575   EFB5 48                         ALSA                                    MULTIPLY BY TWO - THE LOOKUP TABLE HAS TWO BYTES/ENTRY
2576   EFB6 10 8E F0 02                LDY   #LF002                            POINT TO THE CHARACTER PRINT ROUTINE ADDRESS TABLE
2577   EFBA 10 AE A6                   LDY   A,Y                               GET THE ADDRESS OF THE CHARACTER PRINT ROUTINE
2578   EFBD 10 9F D1                   STY   VD1                               AND SAVE IT IN VD1
2579
2580                           * THIS SECTION OF CODE WILL PRINT THE BUFFER TO THE HI-RES SCREEN
2581   EFC0 86 08                      LDA   #$08                              8 ROWS PER HI-RES CHARACTER
2582   EFC2 97 D3                      STA   VD3                               TEMP SAVE THE ROW COUNTER
2583   EFC4 10 8E FE 19                LDY   #H.PBUF                           POINT TO THE PRINT BUFFER
2584   EFC8 CE F0 9D                   LDU   #LF09D                            POINT TO THE HI-RES CHARACTER GENERATOR 'ROM'
2585   EFCB F6 FE 0A                   LDB   H.FCOLOR                          GET THE FOREGROUND COLOR
2586   EFCE BD E7 42                   JSR   PIXELFIL                          FILL ACCB WITH ALL FOREGROUND COLOR PIXELS
2587   EFD1 D7 B5                      STB   ALLCOL                            SAVE THE PIXEL-FILLED BYTE
2588   EFD3 BD E1 19                   JSR   SELTASK1                          SWITCH IN TASK REGISTER 1
2589   EFD6 B6 FE 18                   LDA   H.PCOUNT                          GET THE CHARACTER COUNT
2590   EFD9 34 32              LEFD9   PSHS  Y,X,A
2591   EFDB E6 A0              LEFDB   LDB   ,Y+                               GET A CHARACTER FROM THE PRINT BUFFER
2592   EFDD 4F                         CLRA                                    CLEAR THE MOST SIGNIFICANT BYTE OF ACCD
```

```
2593   EFDE 58                        ALSB
2594   EFDF 58                        ALSB
2595   EFE0 49                        ROLA
2596   EFE1 58                        ALSB
2597   EFE2 49                        ROLA                            MULTIPLY ACCD BY 8 - 8 BYTES PER CHARACTER
2598   EFE3 A6 CB                     LDA   D,U                       GET THE FIRST BYTE OF THE GRAPHIC CHARACTER FROM THE 'ROM'
2599   EFE5 AD 9F 00 D1               JSR   [VD1]                     PUT THE BYTE ON THE HI-RES SCREEN
2600   EFE9 7A FE 18                  DEC   H.PCOUNT                  DECREMENT CHARACTER COUNT
2601   EFEC 2E ED                     BGT   LEFDB                     KEEP GOING UNTIL ALL CHARACTERS DONE
2602   EFEE 35 32                     PULS  A,X,Y                     RESTORE THE PRINT BUFFER POINTER, 'ROM' POINTER & CHAR COUNT
2603   EFF0 0A D3                     DEC   VD3                       BUMP THE ROW COUNTER DOWN ONE
2604   EFF2 27 0A                     BEQ   LEFFE                     BRANCH IF ALL ROWS DONE
2605   EFF4 B7 FE 18                  STA   H.PCOUNT                  RESET THE CHARACTER COUNT
2606   EFF7 33 41                     LEAU  $01,U                     ADJUST THE 'ROM' POINTER DOWN ONE ROW
2607   EFF9 BD EA 45                  JSR   LEA45                     MOVE X DONW ONE HI-RES ROW
2608   EFFC 20 DB                     BRA   LEFD9                     KEEP LOOPING UNTIL THE WHOLE ROW IS DONE
2609   EFFE BD E0 FF          LEFFE   JSR   SELTASK0                  MAKE TASK REGISTER 0 ACTIVE
2610   F001 39                LF001   RTS
2611
2612                          * TABLE OF ADDRESSES OF HI-RES PRINT DRIVERS
2613   F002 F0 1A             LF002   FDB   LF01A                     MODE 1
2614   F004 F0 45             LF004   FDB   LF045                     MODE 2
2615   F006 F0 0A             LF006   FDB   LF00A                     MODE 3
2616   F008 F0 1A             LF008   FDB   LF01A                     MODE 4
2617
2618                          * MODE 3 PRINT DRIVER
2619   F00A 34 02             LF00A   PSHS  A                         SAVE THE CHARACTER MASK
2620   F00C 43                        COMA                            INVERT THE MASK
2621   F00D A4 84                     ANDA  ,X                        'AMD' IT WITH THE SCREEN DATA - CREATE A HOLE FOR THE CHARACTER
2622   F00F A7 84                     STA   ,X                        DATA AND THEN PUT IT BACK ON THE SCREEN
2623   F011 35 02                     PULS  A                         GET THE CHARACTER MASK BACK
2624   F013 94 B5                     ANDA  ALLCOL                    'AND' IT WITH THE PIXEL COLOR BYTE - 'COLOR' THE DATA
2625   F015 AA 84                     ORA   ,X                        'OR' IT WITH THE SCREEN DATA - FILL THE 'HOLE' CREATED ABOVE
2626   F017 A7 80                     STA   ,X+                       WITH THE 'COLORED' CHARACTER DATA AND PUT THE DATA ON THE SCREEN
2627   F019 39                        RTS
2628
2629                          * MODES 1,4 PRINT DRIVER
2630   F01A 34 20             LF01A   PSHS  Y                         SAVE THE PRINT BUFFER POINTER
2631   F01C 10 8E F0 35               LDY   #LF035                    POINT TO THE TABLE OF 4 COLOR PIXEL MASKS
2632   F020 1F 89                     TFR   A,B                       COPY CHARACTER DATA TO ACCB
2633   F022 44                        LSRA
2634   F023 44                        LSRA
2635   F024 44                        LSRA
2636   F025 44                        LSRA                            SHIFT THE HIGH ORDER NIBBLE TO THE LOW ORDER NIBBLE
2637   F026 A6 A6                     LDA   A,Y                       GET THE PIXEL MASK FOR THE HIGH NIBBLE
2638   F028 BD F0 0A                  JSR   LF00A                     DISPLAY THE HIGH ORDER NIBBLE DATA ON THE SCREEN
2639   F02B C4 0F                     ANDB  #$0F                      MASK OFF THE HIGH ORDER NIBBLE
2640   F02D A6 A5                     LDA   B,Y                       GET THE PIXEL MASK FOR THE LOW NIBBLE
2641   F02F BD F0 0A                  JSR   LF00A                     DISPLAY THE LOW ORDER NIBBLE DATA ON THE SCREEN
2642   F032 35 20                     PULS  Y                         RESTORE THE PRINT BUFFER POINTER
2643   F034 39                        RTS
2644
2645                          * FOUR COLOR PIXEL MASKS
2646   F035 00 03 0C 0F 30 33 LF035   FCB   $00,$03,$0C,$0F,$30,$33     4 COLOR PIXEL MASKS
2647   F03B 3C 3F C0 C3 CC CF         FCB   $3C,$3F,$C0,$C3,$CC,$CF
2648   F041 F0 F3 FC FF               FCB   $F0,$F3,$FC,$FF
2649
2650                          * MODE 2 PRINT DRIVER
2651   F045 34 22             LF045   PSHS  Y,A                       SAVE THE PRINT BUFFER POINTER AND THE CHARACTER DATA
2652   F047 10 8E F0 6C               LDY   #LF06C                    POINT TO THE TABLE OF 16 COLOR MASKS
2653   F04B 44                        LSRA
2654   F04C 44                        LSRA
2655   F04D 44                        LSRA
2656   F04E 44                        LSRA                            SHIFT THE HIGH ORDER PIXEL TO BITS 0-3
2657   F04F 48                        ALSA                            MULTIPLY BY 2, THERE ARE 2 BYTES PER TABLE ENTRY
2658   F050 EC A6                     LDD   A,Y                       GET THE FIRST FOUR PIXEL MASKS FROM THE TABLE
2659   F052 BD F0 0A                  JSR   LF00A                     DISPLAY THE FIRST TWO PIXELS
2660   F055 1F 98                     TFR   B,A                       PUT THE NEXT TWO PIXELS' DATA INTO ACCA
2661   F057 BD F0 0A                  JSR   LF00A                     DISPLAY THE NEXT TWO PIXELS
2662   F05A 35 02                     PULS  A                         GET THE CHARACTER DATA BACK
2663   F05C 84 0F                     ANDA  #$0F                      MASK OFF THE HIGH NIBBLE
2664   F05E 48                        ALSA                            MULTIPLY BY 2, THERE ARE 2 BYTES PER TABLE ENTRY
2665   F05F EC A6                     LDD   A,Y                       GET THE LAST FOUR PIXEL MASKS FROM THE TABLE
2666   F061 BD F0 0A                  JSR   LF00A                     DISPLAY THE NEXT TWO PIXELS
2667   F064 1F 98                     TFR   B,A                       PUT THE NEXT TWO PIXELS' DATA INTO ACCA
2668   F066 BD F0 0A                  JSR   LF00A                     DISPLAY THE LAST TWO PIXELS
2669   F069 35 20                     PULS  Y                         RESTORE THE PRINT BUFFER POINTER
2670   F06B 39                        RTS                             WASTED; THIS AND ABOVE INSTRUCTION SHOULD BE PULS Y,PC
2671
2672                          * 16 COLOR PIXEL MASKS - DOUBLE BYTE WIDE
2673   F06C 00 00 00 0F 00 F0 LF06C   FDB   $0000,$000F,$00F0
2674   F072 00 FF 0F 00 0F 0F         FDB   $00FF,$0F00,$0F0F
2675   F078 0F F0 0F FF F0 00         FDB   $0FF0,$0FFF,$F000
2676   F07E F0 0F F0 F0 F0 FF         FDB   $F00F,$F0F0,$F0FF
2677   F084 FF 00 FF 0F FF F0         FDB   $FF00,$FF0F,$FFF0
2678   F08A FF FF                     FDB   $FFFF
2679
2680                          * CONVERT THE PRINT POSITION FROM CHARACTER ROWS AND COLUMNS TO PIXEL ROWS
2681                          * AND COLUMNS; EACH CHARACTER IS 8 PIXELS WIDE AND 8 PIXELS DEEP.
2682   F08C DC BD             LF08C   LDD   HORBEG                    GET THE PRINT COLUMN POSITION
2683   F08E 58                        ALSB
2684   F08F 58                        ALSB
2685   F090 49                        ROLA
2686   F091 58                        ALSB
2687   F092 49                        ROLA                            SHIFT ACCD LEFT THREE TIMES; MULTIPLY COLUMN POSITION BY EIGHT
2688   F093 DD BD                     STD   HORBEG                    SAVE NEW COLUMN POSITION IN TERMS OF PIXELS (8 PIXELS/CHARACTER)
```

```
2689   F095 96 C0                    LDA   VERBEG+1                    GET THE PRINT ROW NUMBER
2690   F097 48                       ALSA
2691   F098 48                       ALSA
2692   F099 48                       ALSA                              SHIFT ACCA LEFT THREE TIMES; MULTIPLY ROW POSITION BY EIGHT
2693   F09A 97 C0                    STA   VERBEG+1                    SAVE NEW ROW POSITION IN TERMS OF PIXELS (8 PIXELS/CHARACTER)
2694   F09C 39                       RTS
2695
2696                          * HI-RES CHARACTER GENERATOR 'ROM'
2697                          * SPECIAL CHARACTERS AND NUMBERS
2698   F09D 00 00 00 00 00 00 LF09D  FCB   $00,$00,$00,$00,$00,$00      BLANK
2699   F0A3 00 00                    FCB   $00,$00
2700   F0A5 10 10 10 10 10 00 LF0A5  FCB   $10,$10,$10,$10,$10,$00      !
2701   F0AB 10 00                    FCB   $10,$00
2702   F0AD 28 28 28 00 00 00 LF0AD  FCB   $28,$28,$28,$00,$00,$00      "
2703   F0B3 00 00                    FCB   $00,$00
2704   F0B5 28 28 7C 28 7C 28 LF0B5  FCB   $28,$28,$7C,$28,$7C,$28      #
2705   F0BB 28 00                    FCB   $28,$00
2706   F0BD 10 3C 50 38 14 78 LF0BD  FCB   $10,$3C,$50,$38,$14,$78      $
2707   F0C3 10 00                    FCB   $10,$00
2708   F0C5 60 64 08 10 20 4C LF0C5  FCB   $60,$64,$08,$10,$20,$4C      %
2709   F0CB 0C 00                    FCB   $0C,$00
2710   F0CD 20 50 50 20 54 48 LF0CD  FCB   $20,$50,$50,$20,$54,$48      &
2711   F0D3 34 00                    FCB   $34,$00
2712   F0D5 10 10 20 00 00 00 LF0D5  FCB   $10,$10,$20,$00,$00,$00      '
2713   F0DB 00 00                    FCB   $00,$00
2714   F0DD 08 10 20 20 20 10 LF0DD  FCB   $08,$10,$20,$20,$20,$10      (
2715   F0E3 08 00                    FCB   $08,$00
2716   F0E5 20 10 08 08 08 10 LF0E5  FCB   $20,$10,$08,$08,$08,$10      )
2717   F0EB 20 00                    FCB   $20,$00
2718   F0ED 00 10 54 38 38 54 LF0ED  FCB   $00,$10,$54,$38,$38,$54      *
2719   F0F3 10 00                    FCB   $10,$00
2720   F0F5 00 10 10 7C 10 10 LF0F5  FCB   $00,$10,$10,$7C,$10,$10      +
2721   F0FB 00 00                    FCB   $00,$00
2722   F0FD 00 00 00 00 00 10 LF0FD  FCB   $00,$00,$00,$00,$00,$10      ,
2723   F103 10 20                    FCB   $10,$20
2724   F105 00 00 00 7C 00 00 LF105  FCB   $00,$00,$00,$7C,$00,$00      -
2725   F10B 00 00                    FCB   $00,$00
2726   F10D 00 00 00 00 00 00 LF10D  FCB   $00,$00,$00,$00,$00,$00      .
2727   F113 10 00                    FCB   $10,$00
2728   F115 00 04 08 10 20 40 LF115  FCB   $00,$04,$08,$10,$20,$40      /
2729   F11B 00 00                    FCB   $00,$00
2730   F11D 38 44 4C 54 64 44 LF11D  FCB   $38,$44,$4C,$54,$64,$44      0
2731   F123 38 00                    FCB   $38,$00
2732   F125 10 30 10 10 10 10 LF125  FCB   $10,$30,$10,$10,$10,$10      1
2733   F12B 38 00                    FCB   $38,$00
2734   F12D 38 44 04 38 40 40 LF12D  FCB   $38,$44,$04,$38,$40,$40      2
2735   F133 7C 00                    FCB   $7C,$00
2736   F135 38 44 04 08 04 44 LF135  FCB   $38,$44,$04,$08,$04,$44      3
2737   F13B 38 00                    FCB   $38,$00
2738   F13D 08 18 28 48 7C 08 LF13D  FCB   $08,$18,$28,$48,$7C,$08      4
2739   F143 08 00                    FCB   $08,$00
2740   F145 7C 40 78 04 04 44 LF145  FCB   $7C,$40,$78,$04,$04,$44      5
2741   F14B 38 00                    FCB   $38,$00
2742   F14D 38 40 40 78 44 44 LF14D  FCB   $38,$40,$40,$78,$44,$44      6
2743   F153 38 00                    FCB   $38,$00
2744   F155 7C 04 08 10 20 40 LF155  FCB   $7C,$04,$08,$10,$20,$40      7
2745   F15B 40 00                    FCB   $40,$00
2746   F15D 38 44 44 38 44 44 LF15D  FCB   $38,$44,$44,$38,$44,$44      8
2747   F163 38 00                    FCB   $38,$00
2748   F165 38 44 44 38 04 04 LF165  FCB   $38,$44,$44,$38,$04,$04      9
2749   F16B 38 00                    FCB   $38,$00
2750   F16D 00 00 10 00 00 10 LF16D  FCB   $00,$00,$10,$00,$00,$10      :
2751   F173 00 00                    FCB   $00,$00
2752   F175 00 00 10 00 00 10 LF175  FCB   $00,$00,$10,$00,$00,$10      ;
2753   F17B 10 20                    FCB   $10,$20
2754   F17D 08 10 20 40 20 10 LF17D  FCB   $08,$10,$20,$40,$20,$10      >
2755   F183 08 00                    FCB   $08,$00
2756   F185 00 00 7C 00 7C 00 LF185  FCB   $00,$00,$7C,$00,$7C,$00      =
2757   F18B 00 00                    FCB   $00,$00
2758   F18D 20 10 08 04 08 10 LF18D  FCB   $20,$10,$08,$04,$08,$10      <
2759   F193 20 00                    FCB   $20,$00
2760   F195 38 44 04 08 10 00 LF195  FCB   $38,$44,$04,$08,$10,$00      ?
2761   F19B 10 00                    FCB   $10,$00
2762
2763                          * UPPER CASE CHARACTERS
2764   F19D 38 44 04 34 4C 4C LF19D  FCB   $38,$44,$04,$34,$4C,$4C      @
2765   F1A3 38 00                    FCB   $38,$00
2766   F1A5 10 28 44 44 7C 44 LF1A5  FCB   $10,$28,$44,$44,$7C,$44      A
2767   F1AB 44 00                    FCB   $44,$00
2768   F1AD 78 24 24 38 24 24 LF1AD  FCB   $78,$24,$24,$38,$24,$24      B
2769   F1B3 78 00                    FCB   $78,$00
2770   F1B5 38 44 40 40 40 44 LF1B5  FCB   $38,$44,$40,$40,$40,$44      C
2771   F1BB 38 00                    FCB   $38,$00
2772   F1BD 78 24 24 24 24 24 LF1BD  FCB   $78,$24,$24,$24,$24,$24      D
2773   F1C3 78 00                    FCB   $78,$00
2774   F1C5 7C 40 40 70 40 40 LF1C5  FCB   $7C,$40,$40,$70,$40,$40      E
2775   F1CB 7C 00                    FCB   $7C,$00
2776   F1CD 7C 40 40 70 40 40 LF1CD  FCB   $7C,$40,$40,$70,$40,$40      F
2777   F1D3 40 00                    FCB   $40,$00
2778   F1D5 38 44 40 40 4C 44 LF1D5  FCB   $38,$44,$40,$40,$4C,$44      G
2779   F1DB 38 00                    FCB   $38,$00
2780   F1DD 44 44 44 7C 44 44 LF1DD  FCB   $44,$44,$44,$7C,$44,$44      H
2781   F1E3 44 00                    FCB   $44,$00
2782   F1E5 38 10 10 10 10 10 LF1E5  FCB   $38,$10,$10,$10,$10,$10      I
2783   F1EB 38 00                    FCB   $38,$00
2784   F1ED 04 04 04 04 04 44 LF1ED  FCB   $04,$04,$04,$04,$04,$44      J
```

```
2785   F1F3 38 00                     FCB   $38,$00
2786   F1F5 44 48 50 60 50 48 LF1F5   FCB   $44,$48,$50,$60,$50,$48          K
2787   F1FB 44 00                     FCB   $44,$00
2788   F1FD 40 40 40 40 40 40 LF1FD   FCB   $40,$40,$40,$40,$40,$40          L
2789   F203 7C 00                     FCB   $7C,$00
2790   F205 44 6C 54 54 44 44 LF205   FCB   $44,$6C,$54,$54,$44,$44          M
2791   F20B 44 00                     FCB   $44,$00
2792   F20D 44 44 64 54 4C 44 LF20D   FCB   $44,$44,$64,$54,$4C,$44          N
2793   F213 44 00                     FCB   $44,$00
2794   F215 38 44 44 44 44 44 LF215   FCB   $38,$44,$44,$44,$44,$44          O
2795   F21B 38 00                     FCB   $38,$00
2796   F21D 78 44 44 78 40 40 LF21D   FCB   $78,$44,$44,$78,$40,$40          P
2797   F223 40 00                     FCB   $40,$00
2798   F225 38 44 44 44 54 48 LF225   FCB   $38,$44,$44,$44,$54,$48          Q
2799   F22B 34 00                     FCB   $34,$00
2800   F22D 78 44 44 78 50 48 LF22D   FCB   $78,$44,$44,$78,$50,$48          R
2801   F233 44 00                     FCB   $44,$00
2802   F235 38 44 40 38 04 44 LF235   FCB   $38,$44,$40,$38,$04,$44          S
2803   F23B 38 00                     FCB   $38,$00
2804   F23D 7C 10 10 10 10 10 LF23D   FCB   $7C,$10,$10,$10,$10,$10          T
2805   F243 10 00                     FCB   $10,$00
2806   F245 44 44 44 44 44 44 LF245   FCB   $44,$44,$44,$44,$44,$44          U
2807   F24B 38 00                     FCB   $38,$00
2808   F24D 44 44 44 28 28 10 LF24D   FCB   $44,$44,$44,$28,$28,$10          V
2809   F253 10 00                     FCB   $10,$00
2810   F255 44 44 44 44 54 6C LF255   FCB   $44,$44,$44,$44,$54,$6C          W
2811   F25B 44 00                     FCB   $44,$00
2812   F25D 44 44 28 10 28 44 LF25D   FCB   $44,$44,$28,$10,$28,$44          X
2813   F263 44 00                     FCB   $44,$00
2814   F265 44 44 28 10 10 10 LF265   FCB   $44,$44,$28,$10,$10,$10          Y
2815   F26B 10 00                     FCB   $10,$00
2816   F26D 7C 04 08 10 20 40 LF26D   FCB   $7C,$04,$08,$10,$20,$40          Z
2817   F273 7C 00                     FCB   $7C,$00
2818   F275 38 20 20 20 20 20 LF275   FCB   $38,$20,$20,$20,$20,$20          ]
2819   F27B 38 00                     FCB   $38,$00
2820   F27D 00 40 20 10 08 04 LF27D   FCB   $00,$40,$20,$10,$08,$04          \
2821   F283 00 00                     FCB   $00,$00
2822   F285 38 08 08 08 08 08 LF285   FCB   $38,$08,$08,$08,$08,$08          [
2823   F28B 38 00                     FCB   $38,$00
2824   F28D 10 38 54 10 10 10 LF28D   FCB   $10,$38,$54,$10,$10,$10          UP ARROW
2825   F293 10 00                     FCB   $10,$00
2826   F295 00 10 20 7C 20 10 LF295   FCB   $00,$10,$20,$7C,$20,$10          LEFT ARROW
2827   F29B 00 00                     FCB   $00,$00
2828
2829                          * LOWER CASE CHARACTERS
2830   F29D 10 28 44 00 00 00 LF29D   FCB   $10,$28,$44,$00,$00,$00          ^
2831   F2A3 00 00                     FCB   $00,$00
2832   F2A5 00 00 38 04 3C 44 LF2A5   FCB   $00,$00,$38,$04,$3C,$44          a
2833   F2AB 3C 00                     FCB   $3C,$00
2834   F2AD 40 40 58 64 44 64 LF2AD   FCB   $40,$40,$58,$64,$44,$64          b
2835   F2B3 58 00                     FCB   $58,$00
2836   F2B5 00 00 38 44 40 44 LF2B5   FCB   $00,$00,$38,$44,$40,$44          c
2837   F2BB 38 00                     FCB   $38,$00
2838   F2BD 04 04 34 4C 44 4C LF2BD   FCB   $04,$04,$34,$4C,$44,$4C          d
2839   F2C3 34 00                     FCB   $34,$00
2840   F2C5 00 00 38 44 7C 40 LF2C5   FCB   $00,$00,$38,$44,$7C,$40          e
2841   F2CB 38 00                     FCB   $38,$00
2842   F2CD 08 14 10 38 10 10 LF2CD   FCB   $08,$14,$10,$38,$10,$10          f
2843   F2D3 10 00                     FCB   $10,$00
2844   F2D5 00 00 34 4C 4C 34 LF2D5   FCB   $00,$00,$34,$4C,$4C,$34          g
2845   F2DB 04 38                     FCB   $04,$38
2846   F2DD 40 40 58 64 44 44 LF2DD   FCB   $40,$40,$58,$64,$44,$44          h
2847   F2E3 44 00                     FCB   $44,$00
2848   F2E5 00 10 00 30 10 10 LF2E5   FCB   $00,$10,$00,$30,$10,$10          i
2849   F2EB 38 00                     FCB   $38,$00
2850   F2ED 00 04 00 04 04 04 LF2ED   FCB   $00,$04,$00,$04,$04,$04          j
2851   F2F3 44 38                     FCB   $44,$38
2852   F2F5 40 40 48 50 60 50 LF2F5   FCB   $40,$40,$48,$50,$60,$50          k
2853   F2FB 48 00                     FCB   $48,$00
2854   F2FD 30 10 10 10 10 10 LF2FD   FCB   $30,$10,$10,$10,$10,$10          l
2855   F303 38 00                     FCB   $38,$00
2856   F305 00 00 68 54 54 54 LF305   FCB   $00,$00,$68,$54,$54,$54          m
2857   F30B 54 00                     FCB   $54,$00
2858   F30D 00 00 58 64 44 44 LF30D   FCB   $00,$00,$58,$64,$44,$44          n
2859   F313 44 00                     FCB   $44,$00
2860   F315 00 00 38 44 44 44 LF315   FCB   $00,$00,$38,$44,$44,$44          o
2861   F31B 38 00                     FCB   $38,$00
2862   F31D 00 00 78 44 44 78 LF31D   FCB   $00,$00,$78,$44,$44,$78          p
2863   F323 40 40                     FCB   $40,$40
2864   F325 00 00 3C 44 44 3C LF325   FCB   $00,$00,$3C,$44,$44,$3C          q
2865   F32B 04 04                     FCB   $04,$04
2866   F32D 00 00 58 64 40 40 LF32D   FCB   $00,$00,$58,$64,$40,$40          r
2867   F333 40 00                     FCB   $40,$00
2868   F335 00 00 3C 40 38 04 LF335   FCB   $00,$00,$3C,$40,$38,$04          s
2869   F33B 78 00                     FCB   $78,$00
2870   F33D 20 20 70 20 20 24 LF33D   FCB   $20,$20,$70,$20,$20,$24          t
2871   F343 18 00                     FCB   $18,$00
2872   F345 00 00 44 44 44 4C LF345   FCB   $00,$00,$44,$44,$44,$4C          u
2873   F34B 34 00                     FCB   $34,$00
2874   F34D 00 00 44 44 44 28 LF34D   FCB   $00,$00,$44,$44,$44,$28          v
2875   F353 10 00                     FCB   $10,$00
2876   F355 00 00 44 54 54 28 LF355   FCB   $00,$00,$44,$54,$54,$28          w
2877   F35B 28 00                     FCB   $28,$00
2878   F35D 00 00 44 28 10 28 LF35D   FCB   $00,$00,$44,$28,$10,$28          x
2879   F363 44 00                     FCB   $44,$00
2880   F365 00 00 44 44 44 3C LF365   FCB   $00,$00,$44,$44,$44,$3C          y
```

```
2881   F36B 04 38                     FCB   $04,$38
2882   F36D 00 00 7C 08 10 20 LF36D   FCB   $00,$00,$7C,$08,$10,$20          z
2883   F373 7C 00                     FCB   $7C,$00
2884   F375 08 10 10 20 10 10 LF375   FCB   $08,$10,$10,$20,$10,$10          {
2885   F37B 08 00                     FCB   $08,$00
2886   F37D 10 10 10 00 10 10 LF37D   FCB   $10,$10,$10,$00,$10,$10          |
2887   F383 10 00                     FCB   $10,$00
2888   F385 20 10 10 08 10 10 LF385   FCB   $20,$10,$10,$08,$10,$10          }
2889   F38B 20 00                     FCB   $20,$00
2890   F38D 20 54 08 00 00 00 LF38D   FCB   $20,$54,$08,$00,$00,$00          ~
2891   F393 00 00                     FCB   $00,$00
2892   F395 00 00 00 00 00 00 LF395   FCB   $00,$00,$00,$00,$00,$00          underline
2893   F39B 7C 00                     FCB   $7C,$00
2894
2895                          * HDRAW
2896   F39D 0D E6             HDRAW   TST   HRMODE                          CHECK HI-RES GRAPHICS MODE
2897   F39F 10 27 F3 4C               LBEQ  LE6EF                          'HR' ERROR IF HI-RES MODE NOT ENABLED
2898   F3A3 10 21 0C 59               LBRN  RAMLINK                        RAM HOOK
2899   F3A7 8E 00 00                  LDX   #0                             * X=0, ACCB=1; END OF DRAW COMMAND LINE VALUES
2900   F3AA C6 01                     LDB   #$01                           * WHEN THESE VALUES ARE PULLED OFF THE STACK,
2901   F3AC 34 14                     PSHS  X,B                            * THE DRAW COMMAND WILL END
2902   F3AE D7 C2                     STB   SETFLG                         SET PSET/PRESET FLAG TO PSET
2903   F3B0 9F D5                     STX   VD5                            CLEAR UPDATE AND DRAW FLAGS
2904   F3B2 BD E7 31                  JSR   LE731                          SET ACTIVE COLOR BYTE
2905   F3B5 BD B1 56                  JSR   LB156                          EVALUATE EXPRESSION
2906   F3B8 BD B6 54          LF3B8   JSR   LB654                          GET LENGTH AND ADDRESS OF COMMAND STRING
2907   F3BB 20 08                     BRA   LF3C5                          INTERPRET THE COMMAND STRING
2908   F3BD BD F5 91          LF3BD   JSR   LF591                          GET THE NEXT CHARACTER FROM THE COMMAND LINE
2909   F3C0 7E F5 A7                  JMP   LF5A7                          EVALUATE A DECIMAL VALUE IN COMMAND LINE
2910   F3C3 35 14             LF3C3   PULS  B,X                            GET NEXT COMMAND LINE TO BE INTERPRETED FROM THE STACK
2911   F3C5 D7 D8             LF3C5   STB   VD8                            SET COMMAND LENGTH COUNTER
2912   F3C7 27 FA                     BEQ   LF3C3                          GET NEW COMMAND LINE IF ZERO
2913   F3C9 9F D9                     STX   VD9                            SET COMMAND LINE ADDRESS
2914   F3CB 10 27 01 01               LBEQ  LF4D0                          EXIT ROUTINE IF ADDRESS = 0
2915   F3CF 0D D8             LF3CF   TST   VD8                            TEST COMMAND LENGTH COUNTER
2916   F3D1 27 F0                     BEQ   LF3C3                          GET NEW LINE IF 0
2917   F3D3 BD F5 91                  JSR   LF591                          GET A COMMAND CHARACTER
2918   F3D6 81 3B                     CMPA  #';'                           CHECK FOR A SEMI-COLON
2919   F3D8 27 F5                     BEQ   LF3CF                          IGNORE SEMI-COLONS
2920   F3DA 81 27                     CMPA  #''                            CHECK FOR APOSTROPHE
2921   F3DC 27 F1                     BEQ   LF3CF                          IGNORE APOSTROPHE
2922   F3DE 81 4E                     CMPA  #'N'                           UPDATE CHECK?
2923   F3E0 26 04                     BNE   LF3E6                          BRANCH IF NOT
2924   F3E2 03 D5                     COM   VD5                            TOGGLE UPDATE FLAG; 0 = UPDATE, FF = NO UPDATE
2925   F3E4 20 E9                     BRA   LF3CF                          GET NEXT COMMAND
2926   F3E6 81 42             LF3E6   CMPA  #'B'                           CHECK DRAW FLAG?
2927   F3E8 26 04                     BNE   LF3EE                          BRANCH IF NOT
2928   F3EA 03 D6                     COM   VD6                            TOGGLE DRAW FLAG; 0 = DRAW LINE, FF = DON'T DRAW LINE
2929   F3EC 20 E1                     BRA   LF3CF                          GET ENXT COMMAND
2930   F3EE 81 58             LF3EE   CMPA  #'X'                           SUBSTRING?
2931   F3F0 10 27 00 AD               LBEQ  LF4A1                          GO EXECUTE A COMMAND SUBSTRING
2932   F3F4 81 4D                     CMPA  #'M'                           MOVE THE DRAW POSITION?
2933   F3F6 10 27 01 52               LBEQ  LF54C                          BRANCH IF YES, GO MOVE IT
2934   F3FA 34 02                     PSHS  A                              SAVE CURRENT COMMAND
2935   F3FC C6 01                     LDB   #$01                           DEFAULT VALUE IF NO NUMBER FOLLOWS COMMAND
2936   F3FE 0F D3                     CLR   VD3                            CLEAR MS BYTE OF SUBCOMMAND VALUE
2937   F400 D7 D4                     STB   VD4                            SAVE LS BYTE OF SUBCOMMAND VALUE
2938   F402 0D D8                     TST   VD8                            CHECK COMMAND LENGTH COUNTER
2939   F404 27 11                     BEQ   LF417                          BRANCH IF NO COMMANDS LEFT
2940   F406 BD F5 91                  JSR   LF591                          GET A COMMAND CHARACTER
2941   F409 BD B3 A2                  JSR   LB3A2                          SET CARRY IF NOT ALPHA
2942   F40C 34 01                     PSHS  CC                             SAVE CARRY FLAG
2943   F40E BD F5 F2                  JSR   LF5F2                          MOVE COMMAND POINTER BACK ONE
2944   F411 35 01                     PULS  CC                             RESTORE CARRY FLAG
2945   F413 24 02                     BCC   LF417                          BRANCH IF NEXT COMMAND IS ALPHA
2946   F415 8D A6                     BSR   LF3BD                          EVALUATE DECIMAL COMMAND LINE VALUE - RETURN VALUE IN ACCD & VD3
2947   F417 35 02             LF417   PULS  A                              GET CURRENT COMMAND BACK
2948   F419 81 43                     CMPA  #'C'                           CHANGE COLOR?
2949   F41B 27 28                     BEQ   LF445                          BRANCH IF YES
2950   F41D 81 41                     CMPA  #'A'                           CHANGE ANGLE?
2951   F41F 27 30                     BEQ   LF451                          BRANCH IF YES
2952   F421 81 53                     CMPA  #'S'                           CHANGE SCALE?
2953   F423 27 37                     BEQ   LF45C                          BRANCH IF YES
2954   F425 81 55                     CMPA  #'U'                           GO UP?
2955   F427 27 6D                     BEQ   LF496                          BRANCH IF YES
2956   F429 81 44                     CMPA  #'D'                           GO DOWN?
2957   F42B 27 65                     BEQ   LF492                          BRANCH IF YES
2958   F42D 81 4C                     CMPA  #'L'                           GO LEFT?
2959   F42F 27 5B                     BEQ   LF48C                          BRANCH IF YES
2960   F431 81 52                     CMPA  #'R'                           GO RIGHT?
2961   F433 27 50                     BEQ   LF485                          BRANCH IF YES
2962   F435 80 45                     SUBA  #'E'                           MASK OFF ASCII FOR LETTER E-H COMMAND CHECKS
2963   F437 27 3A                     BEQ   LF473                          BRANCH IF E (45 DEGREES)
2964   F439 4A                        DECA                                 CHECK FOR F
2965   F43A 27 31                     BEQ   LF46D                          BRANCH IF F (135 DEGREES)
2966   F43C 4A                        DECA                                 CHECK FOR G
2967   F43D 27 3E                     BEQ   LF47D                          BRANCH IF G (225 DEGREES)
2968   F43F 4A                        DECA                                 CHECK FOR H
2969   F440 27 25                     BEQ   LF467                          BRANCH IF H (315 DEGREES)
2970   F442 7E B4 4A                  JMP   ILLFUNC                        ILLEGAL FUNCTION CALL ERROR IF ILLEGAL SUBCOMMAND
2971
2972                          * CHANGE COLOR
2973   F445 BD E7 11          LF445   JSR   LE711                          ADJUST COLOR CODE FOR PROPER GRAPHICS MODE
2974   F448 F7 FE 0A                  STB   H.FCOLOR                       SAVE NEW FOREGROUND COLOR
2975   F44B BD E7 31                  JSR   LE731                          SET UP COLOR BYTES
2976   F44E 16 FF 7E                  LBRA  LF3CF                          GO PROCESS ANOTHER COMMAND
```

```
                        * CHANGE ANGLE
F451 C1 Ø4              LF451   CMPB  #$Ø4                     ONLY ANGLES Ø-3 ARE LEGAL
F453 1Ø 24 BF F3                LBCC  ILLFUNC                  ILLEGAL FUNCTION CALL ERROR
F457 D7 E8                      STB   ANGLE                    SAVE DRAW ANGLE
F459 16 FF 73                   LBRA  LF3CF                    GO PROCESS ANOTHER COMMAND
                        * CHANGE SCALE
F45C C1 3F              LF45C   CMPB  #63                      ONLY Ø-63 ARE LEGAL
F45E 1Ø 24 BF E8                LBCC  ILLFUNC                  ILLEGAL FUNCTION CALL ERROR
F462 D7 E9                      STB   SCALE                    SAVE DRAW SCALE
F464 16 FF 68                   LBRA  LF3CF                    GO PROCESS ANOTHER COMMAND
                        * 315 DEGREES
F467 96 D3              LF467   LDA   VD3                      NOW ACCD = VALUE OF THE SUBCOMMAND
F469 8D 61                      BSR   NEGACCD                  MAKE HORIZONTAL DIFFERENCE NEGATIVE
F46B 2Ø Ø2                      BRA   LF46F                    BRANCH AROUND NEXT INSTRUCTION
                        * 135 DEGREES
F46D 96 D3              LF46D   LDA   VD3                      NOW ACCD = VALUE OF THE SUBCOMMAND
F46F 1F Ø1              LF46F   TFR   D,X                      COPY HORIZONTAL DIFFERENCE TO VERTICAL DIFFERENCE
F471 2Ø 61                      BRA   LF4D4                    GO MOVE THE DRAW POSITION
                        * 45 DEGREES
F473 96 D3              LF473   LDA   VD3                      NOW ACCD = VALUE OF THE SUBCOMMAND
F475 1F Ø1                      TFR   D,X                      COPY HORIZONTAL DIFFERENCE TO VERTICAL DIFFERENCE
F477 8D 53                      BSR   NEGACCD                  MAKE HORIZONTAL DIFFERENCE NEGATIVE
F479 1E Ø1                      EXG   D,X                      SWAP HOR AND VER DIFFERENCES
F47B 2Ø 57                      BRA   LF4D4                    GO MOVE THE DRAW POSITION
                        * 225 DEGREES
F47D 96 D3              LF47D   LDA   VD3                      NOW ACCD = VALUE OF THE SUBCOMMAND
F47F 1F Ø1                      TFR   D,X                      COPY HORIZONTAL DIFFERENCE TO VERTICAL DIFFERENCE
F481 8D 49                      BSR   NEGACCD                  MAKE HORIZONTAL DIFFERENCE NEGATIVE
F483 2Ø 4F                      BRA   LF4D4                    GO MOVE THE DRAW POSITION
                        * GO RIGHT
F485 96 D3              LF485   LDA   VD3                      NOW ACCD = VALUE OF THE SUBCOMMAND
F487 8E Ø0 Ø0           LF487   LDX   #Ø                       X=Ø; VERT DIFFERENCE = Ø
F48A 2Ø 48                      BRA   LF4D4                    GO MOVE THE DRAW POSITION
                        * GO LEFT
F48C 96 D3              LF48C   LDA   VD3                      NOW ACCD = VALUE OF THE SUBCOMMAND
F48E 8D 3C                      BSR   NEGACCD                  MAKE HORIZONTAL DIFFERENCE NEGATIVE
F49Ø 2Ø F5                      BRA   LF487                    MAKE VERTICAL DIFFERENCE ZERO AND MOVE THE DRAW POSITION
                        * GO DOWN
F492 96 D3              LF492   LDA   VD3                      NOW ACCD = VALUE OF THE SUBCOMMAND
F494 2Ø Ø4                      BRA   LF49A                    MAKE VER DIFF=Ø, EXCHANGE HOR & VER DIFFS AND MOVE DRAW POSITION
                        * GO UP
F496 96 D3              LF496   LDA   VD3                      NOW ACCD = VALUE OF THE SUBCOMMAND
F498 8D 32                      BSR   NEGACCD                  MAKE HORIZONTAL DIFFERENCE NEGATIVE
F49A 8E Ø0 Ø0           LF49A   LDX   #Ø                       X=Ø; HORIZONTAL DIFFERENCE = Ø
F49D 1E 1Ø                      EXG   X,D                      SWAP HOR AND VER DIFFERENCES
F49F 2Ø 33                      BRA   LF4D4                    GO MOVE THE DRAW POSITION
                        * EXECUTE A COMMAND SUB STRING
F4A1 BD F6 11           LF4A1   JSR   LF611                    INTERPRET CURRENT COMMAND AS IF IT WERE A BASIC VARIABLE
F4A4 C6 Ø2                      LDB   #$Ø2                     =
F4A6 BD AC 33                   JSR   LAC33                    = SEE IF AT LEAST FOUR BYTES OF FREE RAM ARE LEFT
F4A9 D6 D8                      LDB   VD8                      GET CURRENT COMMAND LENGTH
F4AB 9E D9                      LDX   VD9                      GET CURRENT COMMAND COUNTER
F4AD 34 14                      PSHS  X,B                      SAVE THEM ON THE STACK
F4AF 7E F3 B8                   JMP   LF3B8                    EVALUATE NUMERICAL VALUE IN COMMAND LINE
                        * MULTIPLY HOR OR VER DIFFERENCE BY SCALE FACTOR, DIVIDE PRODUCT BY 4 AND RETURN VALUE IN ACCD
F4B2 D6 E9              LF4B2   LDB   SCALE                    GET DRAW SCALE
F4B4 27 1B                      BEQ   LF4D1                    BRANCH IF ZERO (DEFAULT TO FULL SCALE)
F4B6 4F                         CLRA                           CLEAR THE MS BYTE
F4B7 1E Ø1                      EXG   D,X                      SWAP DIFFERENCE AND SCALE FACTOR
F4B9 A7 E2                      STA   ,-S                      SAVE MS BYTE OF DIFFERENCE ON STACK (SIGN INFORMATION)
F4BB 2A Ø2                      BPL   LF4BF                    BRANCH IF POSITIVE DIFFERENCE
F4BD 8D ØD                      BSR   NEGACCD                  FORCE THE DIFFERENCE TO BE A POSITIVE VALUE
F4BF BD EB CB           LF4BF   JSR   LEBCB                    MULT DIFFERENCE BY SCALE FACTOR
F4C2 1F 3Ø                      TFR   U,D                      SAVE 2 MS BYTES IN ACCD
F4C4 44                         LSRA
F4C5 56                         RORB                           DIVIDE ACCD BY 2
F4C6 44                         LSRA
F4C7 56                         RORB                           DO IT AGAIN, EACH SCALE INCREMENT IS 1/4 FULL SCALE
F4C8 6D E Ø                     TST   ,S+                      CHECK SIGN OF ORIGINAL DIFFERENCE
F4CA 2A Ø4                      BPL   LF4DØ                    RETURN IF IT WAS POSITIVE

                        * NEGATE ACCD
F4CC 4Ø                 NEGACCD NEGA
F4CD 5Ø                         NEGB
F4CE 82 ØØ                      SBCA  #$ØØ                     NEGATE ACCD
F4DØ 39                 LF4DØ   RTS
F4D1 1F 1Ø              LF4D1   TFR   X,D                      TRANSFER UNCHANGED DIFFERENCE TO ACCD
F4D3 39                         RTS

                        * MOVE THE DRAW POSITION - ADD THE ORTHOGONAL DIFFERENCES IN ACCD (HORIZONTAL)
                        * AND X (VERTICAL) TO THE CURRENT POSITION; DRAW A LINE AFTER THE MOVE
F4D4 34 Ø6              LF4D4   PSHS  B,A                      SAVE THE HORIZONTAL DIFFERENCE
F4D6 8D DA                      BSR   LF4B2                    APPLY SCALE FACTOR TO VERTICAL
F4D8 35 1Ø                      PULS  X                        GET HORIZONTAL DIFFERENCE
F4DA 34 Ø6                      PSHS  B,A                      SAVE THE VERTICAL DIFFERENCE
F4DC 8D D4                      BSR   LF4B2                    APPLY THE SCALE FACTOR TO HORIZONTAL
F4DE 35 1Ø                      PULS  X                        GET THE VERTICAL DIFFERENCE
F4E Ø 1Ø 9E E8                  LDY   ANGLE                    GET DRAW ANGLE AND SCALE
F4E3 34 2Ø                      PSHS  Y                        SAVE THEM ON THE STACK
F4E5 6D E4              LF4E5   TST   ,S                       CHECK DRAW ANGLE
F4E7 27 Ø8                      BEQ   LF4F1                    BRANCH IF NO ANGLE
F4E9 1E 1Ø                      EXG   X,D                      SWAP HORIZONTAL AND VERTICAL DIFFERENCES
F4EB 8D DF                      BSR   NEGACCD                  NEGATE ACCD
F4ED 6A E4                      DEC   ,S                       DECR ANGLE
F4EF 2Ø F4                      BRA   LF4E5                    CHECK ANGLE AGAIN
F4F1 35 2Ø              LF4F1   PULS  Y                        PULL ANGLE AND SCALE OFF OF THE STACK
```

```
3073   F4F3 CE 00 00              LDU   #0                          DEFAULT HORIZONTAL END POSITION TO 0
3074   F4F6 D3 C7                 ADDD  HORDEF                       ADD DIFFERENCE TO HORIZONTAL START
3075   F4F8 2B 02                 BMI   LF4FC                        GO FORCE HORIZONTAL COORD TO 0 IF RESULT IS NEGATIVE
3076   F4FA 1F 03                 TFR   D,U                          SAVE HORIZONTAL END POSITION IN U
3077   F4FC 1F 10          LF4FC  TFR   X,D                          PUT DIFFERENCE IN ACCD
3078   F4FE 8E 00 00              LDX   #0                           DEFAULT THE VERTICAL END POSITION TO 0
3079   F501 D3 C9                 ADDD  VERDEF                       ADD THE DIFFERENCE TO VERTICAL START
3080   F503 2B 02                 BMI   LF507                        VERTICAL COORD = 0 IF RESULT IS NEGATIVE
3081   F505 1F 01                 TFR   D,X                          SAVE VERTICAL POSITION IN X
3082                       * MOVE THE DRAW POSITION; ENTER WITH ABSOLUTE HORIZONTAL POSITION
3083                       * IN U REGISTER AND ABSOULTE VERTICAL POSITION IN X REGISTER.
3084   F507 11 83 02 80    LF507  CMPU  #640                          COMPARE TO MAX HORIZONTAL COORDINATE
3085   F50B 25 03                 BCS   LF510                        BRANCH IF WITHIN RANGE
3086   F50D CE 02 7F              LDU   #640-1                       FORCE MAXIMUM VALUE IF NOT
3087   F510 96 E6          LF510  LDA   HRMODE                       GET HI-RES GRAPHICS MODE
3088   F512 81 02                 CMPA  #$02                         SEE WHICH ONE
3089   F514 2E 09                 BGT   LF51F                        BRANCH IF MODE 3 OR 4
3090   F516 11 83 01 40           CMPU  #320                         MAX HORIZONTAL COORD FOR 320x192 MODES (1 AND 2)
3091   F51A 25 03                 BCS   LF51F                        BRANCH IF WITHIN LIMITS
3092   F51C CE 01 3F              LDU   #320-1                       FORCE TO MAXIMUM IF NOT
3093   F51F 8C 00 C0       LF51F  CMPX  #192                         IS VERTICAL COORD WITHIN RANGE?
3094   F522 25 03                 BCS   LF527                        BRANCH IF IT IS
3095   F524 8E 00 BF              LDX   #192-1                       FORCE TO MAXIMUM IF NOT
3096   F527 DC C7          LF527  LDD   HORDEF                       GET LAST HORIZONTAL POSITION
3097   F529 DD BD                 STD   HORBEG                       MAKE IT THE HORIZONTAL START
3098   F52B DC C9                 LDD   VERDEF                       GET LAST VERTICAL POSITION
3099   F52D DD BF                 STD   VERBEG                       MAKE IT THE VERTICAL START
3100   F52F 9F C5                 STX   VEREND                       SAVE VERTICAL END COORD
3101   F531 DF C3                 STU   HOREND                       SAVE HORIZONTAL END COORDINATE
3102   F533 0D D5                 TST   VD5                          CHECK UPDATE FLAG
3103   F535 26 04                 BNE   LF53B                        BRANCH IF NO UPDATE
3104   F537 9F C9                 STX   VERDEF                       UPDATE VERTICAL POSITION OF DRAW POINTER
3105   F539 DF C7                 STU   HORDEF                       DO THE SAME WITH THE HORIZONTAL DRAW POINTER
3106   F53B BD EA 0D       LF53B  JSR   LEA0D                        NORMALIZE COORDS IN HOREND,VEREND AND HORBEG,VERBEG
3107   F53E 0D D6                 TST   VD6                          CHECK DRAW FLAG
3108   F540 26 03                 BNE   LF545                        BRANCH IF NO DRAW
3109   F542 BD E9 4E              JSR   LE94E                        DRAWLINE FROM (HORBEG,VERBEG) TO (HOREND,VEREND)
3110   F545 0F D5          LF545  CLR   VD5                          RESET UPDATE FLAG
3111   F547 0F D6                 CLR   VD6                          RESET DRAW FLAG
3112   F549 7E F3 CF              JMP   LF3CF                        GO GET ANOTHER COMMAND
3113
3114                       * SET THE DRAW POSITION
3115   F54C BD F5 91       LF54C  JSR   LF591                        GET A CHAR FROM COMMAND LINE
3116   F54F 34 02                 PSHS  A                            SAVE IT
3117   F551 BD F5 78              JSR   LF578                        EVALUATE THE HORIZONTAL DIFFERENCE
3118   F554 34 06                 PSHS  B,A                          SAVE IT ON THE STACK
3119   F556 BD F5 91              JSR   LF591                        GET A CHAR FROM COMMAND LINE
3120   F559 81 2C                 CMPA  #','                         CHECK FOR COMMA
3121   F55B 10 26 BE EB           LBNE  ILLFUNC                      ILLEGAL FUCNTION CALL ERROR IF NO COMMA
3122   F55F BD F5 75              JSR   LF575                        EVALUATE THE VERTICAL DIFFERENCE
3123   F562 1F 01                 TFR   D,X                          SAVE VERTICAL DIFFERENCE IN X
3124   F564 35 40                 PULS  U                            GET HORIZONTAL DIFFERENCE IN U
3125   F566 35 02                 PULS  A                            GET FIRST COMMAND CHARACTER
3126   F568 81 2B                 CMPA  #'+'                         CHECK FOR PLUS
3127   F56A 27 04                 BEQ   LF570                        TREAT VALUES IN X AND U AS DIFFERENCES AND MOVE POINTER
3128   F56C 81 2D                 CMPA  #'-'                         CHECK FOR MINUS
3129   F56E 26 97                 BNE   LF507                        IF NOT '+' OR '-', MOVE THE POINTER TO THE COORDINATES IN U AND ACCD
3130   F570 1F 30          LF570  TFR   U,D                          PUT HORIZONTAL DIFFERENCE IN ACCD; X CONTAINS THE VERTICAL DIFFERENCE
3131   F572 7E F4 D4              JMP   LF4D4                        GOMOVE THE DRAW POSITION
3132   F575 BD F5 91       LF575  JSR   LF591                        GET A CHAR FROM COMMAND LINE
3133   F578 81 2B          LF578  CMPA  #'+'                         CHECK FOR LEADING + (RELATIVE MOTION)
3134   F57A 27 07                 BEQ   LF583                        BRANCH IF RELATIVE
3135   F57C 81 2D                 CMPA  #'-'                         DO THE SAME FOR THE MINUS SIGN
3136   F57E 27 04                 BEQ   LF584                        BRANCH IF RELATIVE
3137   F580 BD F5 F2              JSR   LF5F2                        MOVE COMMAND STRING BACK ONE IF NOT RELATIVE MOTION
3138   F583 4F             LF583  CLRA                               IF ACCA=0, THEN '+'; IF ACCA <> 0, THEN '-'
3139   F584 34 02          LF584  PSHS  A                            SAVE ADD/SUB FLAG; 0=ADD, <> 0 = SUBTRACT
3140   F586 BD F3 BD              JSR   LF3BD                        EVALUATE DECIMAL NUMBER IN COMMAND STRING - RETURN VALUE IN ACCD
3141   F589 6D E0                 TST   ,S+                          CHECK THE ADD/SUBTRACT FLAG AND CLEAN UP THE STACK
3142   F58B 27 03                 BEQ   LF590                        BRANCH IF ADD
3143                       * THIS IS A BUG; SHOULD BE JSR NEGACCD INSTEAD OF THE NEXT TWO INSTRUCTIONS
3144   F58D 50                    NEGB
3145   F58E 82 00                 SBCA  #$00
3146   F590 39             LF590  RTS
3147
3148                       * GET NEXT COMMAND - RETURN VALUE IN ACCA
3149   F591 34 10          LF591  PSHS  X                            SAVE X REGISTER
3150   F593 0D D8          LF593  TST   VD8                          CHECK COMMAND COUNTER
3151   F595 10 27 BE B1           LBEQ  ILLFUNC                      ILLEGAL FUNCTION CALL ERROR IF NO COMMAND DATA LEFT
3152   F599 9E D9                 LDX   VD9                          GET COMMAND ADDRESS
3153   F59B A6 80                 LDA   ,X+                          GET COMMAND
3154   F59D 9F D9                 STX   VD9                          SAVE NEW COMMAND ADDRESS
3155   F59F 0A D8                 DEC   VD8                          DECREMENT COMMAND COUNTER
3156   F5A1 81 20                 CMPA  #SPACE                       CHECK FOR BLANK
3157   F5A3 27 EE                 BEQ   LF593                        IGNORE BLANKS
3158   F5A5 35 90                 PULS  X,PC                         RESTORE X REGISTER AND RETURN
3159
3160   F5A7 81 3D          LF5A7  CMPA  #'='                         CHECK FOR A VARIABLE EQUATE
3161   F5A9 26 0B                 BNE   LF5B6                        BRANCH IF NOT VARIABLE EQUATE
3162   F5AB 34 60                 PSHS  U,Y                          SAVE REGISTERS
3163   F5AD 8D 62                 BSR   LF611                        INTERPRET THE VARIABLE IN THE COMMAND LINE
3164   F5AF BD B3 E9              JSR   LB3E9                        CONVERT VARIABLE INTO A POSITIVE INTEGER IN ACCD
3165   F5B2 DD D3                 STD   VD3                          SAVE THE SUBCOMMAND VALUE
3166   F5B4 35 E0                 PULS  Y,U,PC                       RESTORE REGISTERS AND RETURN
3167   F5B6 BD F6 08       LF5B6  JSR   LF608                        CLEAR CARRY IF NUMERIC
3168   F5B9 10 25 BE 8D           LBCS  ILLFUNC                      ILLEGAL FUNCTION CALL IF NOT NUMERIC
```

```
3169   F5BD 0F D3                      CLR   VD3                             *
3170   F5BF 0F D4                      CLR   VD4                             * INITIALIZE THE SUBCOMMAND VALUE TO ZERO
3171                                * STRIP A DECIMAL ASCII VALUE FROM THE COMMAND STRING AND RETURN THE BINARY VALUE IN VD3
3172   F5C1 80 30                LF5C1 SUBA  #'0'                            MASK OFF ASCII
3173   F5C3 97 D7                      STA   VD7                             SAVE TEMPORARILY
3174   F5C5 DC D3                      LDD   VD3                             GET THE CURRENT SUBCOMMAND VALUE
3175   F5C7 8D 34                      BSR   LF5FD                           MULTIPLY ACCD BY 10
3176   F5C9 DB D7                      ADDB  VD7                             ADD THE CURRENT DIGIT
3177   F5CB 89 00                      ADCA  #$00                            PROPAGATE THE CARRY
3178   F5CD DD D3                      STD   VD3                             SAVE THE NEW SUBCOMMAND VALUE
3179   F5CF 96 E6                      LDA   HRMODE                          GET THE HI-RES GRAPHICS MODE
3180   F5D1 81 02                      CMPA  #$02                            IS IT A 640 OR 320 BYTES/PIXEL ROW MODE?
3181   F5D3 2E 05                      BGT   LF5DA                           BRANCH IF 640 PIXELS/HORIZONTAL ROW MODE
3182   F5D5 CC 01 3F                   LDD   #320-1                          MAXIMUM HORIZONTAL PIXELS IN THE 320 PIXEL MODE
3183   F5D8 20 03                      BRA   LF5DD
3184   F5DA CC 02 7F             LF5DA LDD   #640-1                          MAXIMUM HORIZONTAL PIXELS IN THE 640 PIXEL MODE
3185   F5DD 10 93 D3             LF5DD CMPD  VD3                             COMPARE THE SUBCOMMAND TO THE MAXIMUM PERMISSABLE
3186   F5E0 10 2D BE 66                LBLT  ILLFUNC                         ILLEGAL FUNCTION CALL IF SUBCOMMAND TOO BIG
3187   F5E4 DC D3                      LDD   VD3                             THIS INSTRUCTION IS USELESS
3188   F5E6 0D D8                      TST   VD8                             CHECK THE COMMAND COUNTER
3189   F5E8 27 10                      BEQ   LF5FA                           BRANCH IF NO COMMANDS LEFT
3190   F5EA BD F5 91                   JSR   LF591                           GET ANOTHER COMMAND
3191   F5ED BD F6 08                   JSR   LF608                           CLEAR CARRY IF NUMERIC
3192   F5F0 24 CF                      BCC   LF5C1                           BRANCH IF MORE NUMERIC DATA TO CONVERT
3193   F5F2 0C D8                LF5F2 INC   VD8                             ADD ONE TO THE COMMAND COUNTER
3194   F5F4 9E D9                      LDX   VD9                             *
3195   F5F6 30 1F                      LEAX  $-01,X                          *
3196   F5F8 9F D9                      STX   VD9                             * MOVE THE COMMAND STRING BACK ONE
3197   F5FA DC D3                LF5FA LDD   VD3                             LOAD ACCD WITH THE VALUE OF THE SUBCOMMAND
3198   F5FC 39                         RTS
3199
3200                                * MULTIPLY ACCD BY TEN
3201   F5FD 58                   LF5FD ALSB
3202   F5FE 49                         ROLA                                  MULTIPLY ACCD BY 2
3203   F5FF 34 06                      PSHS  B,A                             SAVE ACCD TIME 2
3204   F601 58                         ALSB
3205   F602 49                         ROLA
3206   F603 58                         ALSB
3207   F604 49                         ROLA                                  NOW ACCD = ACCD * 8
3208   F605 E3 E1                      ADDD  ,S++                            ADD ACCD*2; THE RESULT IS NOW ACCD*10
3209   F607 39                         RTS
3210
3211                                * CLEAR THE CARRY FLAG IF ACCA CONTAINS A NUMERIC ASCII VALUE ($30-$39)
3212   F608 81 30                LF608 CMPA  #'0'
3213   F60A 25 04                      BCS   LF610                           RETURN IF LESS THAN ASCII ZERO
3214   F60C 80 3A                      SUBA  #'9'+1
3215   F60E 80 C6                      SUBA  #-('9'+1)                       SET CARRY IF NOT 0-9
3216   F610 39                   LF610 RTS
3217                                * INTERPRET THE CURRENT COMMAND STRING AS IF IT WERE A BASIC VARIABLE
3218   F611 9E D9                LF611 LDX   VD9                             GET THE COMMAND POINTER
3219   F613 34 10                      PSHS  X                               SAVE IT
3220   F615 BD F5 91                   JSR   LF591                           GET A COMMAND STRING CHARACTER
3221   F618 BD B3 A2                   JSR   LB3A2                           SET CARRY IF NOT UPPER CASE ALPHA
3222   F61B 10 25 BE 2B                LBCS  ILLFUNC                         ILLEGAL FUNCTION CALL ERROR IF NOT ALPHA - ILLEGAL VARIABLE NAME
3223   F61F BD F5 91             LF61F JSR   LF591                           GET COMMAND STRING CHARACTER
3224   F622 81 3B                      CMPA  #';'                            CHECK FOR A SEMICOLON (SUBCOMMAND SEPARATOR)
3225   F624 26 F9                      BNE   LF61F                           LOOP UNTIL SEMICOLON FOUND
3226   F626 35 10                      PULS  X                               GET THE START OF THE VARIABLE NAME
3227   F628 DE A6                      LDU   CHARAD                          GET THE CURRENT ADDRESS OF THE VARIABLE NAME
3228   F62A 34 40                      PSHS  U                               SAVE IT
3229   F62C 9F A6                      STX   CHARAD                          PUT THE COMMAND POINTER IN PLACE OF BASIC'S INPUT POINTER
3230   F62E BD B2 84                   JSR   LB284                           EVALUATE AN ALPHA EXPRESSION
3231   F631 35 10                      PULS  X                               GET BASIC'S POINTER BACK
3232   F633 9F A6                      STX   CHARAD                          RESTORE BASIC'S INPUT POINTER
3233   F635 39                         RTS
3234
3235                                * WIDTH
3236   F636 0F E6                WIDTH CLR   HRMODE                          TURN OFF HI-RES GRAPHICS MODE
3237   F638 10 21 09 C4                LBRN  RAMLINK                         RAM HOOK
3238   F63C 81 00                      CMPA  #$00                            TEST FOR END OF LINE - NO ARGUMENT GIVEN
3239   F63E 27 0F                      BEQ   LF64F                           'FC' ERROR IF NO ARGUMENT
3240   F640 BD B7 0B                   JSR   EVALEXPB                        EVALUATE EXPRESSION, RETURN VALUE IN ACCB
3241   F643 C1 20                      CMPB  #32                             32 COLUMNS
3242   F645 27 0B                      BEQ   COL32
3243   F647 C1 28                      CMPB  #40                             40 COLUMNS
3244   F649 27 11                      BEQ   COL40
3245   F64B C1 50                      CMPB  #80                             80 COLUMNS
3246   F64D 27 2A                      BEQ   COL80
3247   F64F 7E B4 4A             LF64F JMP   ILLFUNC                         ILLEGAL FUNCTION CALL ERROR
3248
3249                                * 32 COLUMNS
3250   F652 4F                   COL32 CLRA                                  32 COLUMN MODE FLAG
3251   F653 97 E7                      STA   HRWIDTH                         SAVE THE HI-RES TEXT MODE
3252   F655 BD A9 28                   JSR   LA928                           CLEAR THE 32 COLUMN SCREEN
3253   F658 17 E9 BE                   LBSR  SETTEXT                         SETUP THE VIDEO MODE REGISTERS
3254   F65B 39                         RTS
3255
3256                                * 40 COLUMNS
3257   F65C 86 01                COL40 LDA   #$01                            40 COLUMN MODE FLAG
3258   F65E 97 E7                      STA   HRWIDTH                         SAVE THE HI-RES TEXT MODE
3259   F660 17 01 0F                   LBSR  LF772                           PUT THE HI-RES TEXT SCREEN INTO THE LOGICAL ADDRESS SPACE
3260   F663 86 28                      LDA   #40                             40 COLUMNS
3261   F665 C6 18                      LDB   #ROWMAX                         MAXIMUM NUMBER OF ROWS
3262   F667 FD FE 04                   STD   H.COLUMN                        SAVE THE NUMBER OF COLUMNS AND ROWS
3263   F66A CC 27 80                   LDD   #HRSCREEN+40*ROWMAX*2           END OF THE HI-RES TEXT SCREEN
3264
```

```
3265   F66D FD FE 06          LF66D    STD   H.DISPEN                        SAVE THE END OF THE HI-RES TEXT SCREEN
3266   F670 8D 1A                      BSR   LF68C                           RESET HI-RES TEXT SCREEN
3267   F672 17 01 03                   LBSR  LF778                           PUT BLOCK 7.1 INTO LOGICAL BLOCK 1
3268   F675 17 E9 A1                   LBSR  SETTEXT                         SETUP THE VIDEO MODE REGISTERS
3269   F678 39                         RTS
3270
3271                          * 80 COLUMNS
3272   F679 86 02             COL80    LDA   #$02                            80 COLUMN MODE FLAG
3273   F67B 97 E7                      STA   HRWIDTH                         SAVE THE HI-RES TEXT MODE
3274   F67D 17 00 F2                   LBSR  LF772                           PUT THE HI-RES TEXT SCREEN INTO THE LOGICAL ADDRESS SPACE
3275   F680 86 50                      LDA   #80                             80 COLUMNS
3276   F682 C6 18                      LDB   #ROWMAX                         MAXIMUM NUMBER OF ROWS
3277   F684 FD FE 04                   STD   H.COLUMN                        SAVE THE NUMBER OF COLUMNS AND ROWS
3278   F687 CC 2F 00                   LDD   #HRSCREEN+80*ROWMAX*2           END OF THE HI-RES TEXT SCREEN
3279   F68A 20 E1                      BRA   LF66D
3280
3281   F68C 8E 20 00          LF68C    LDX   #HRESSCRN                       POINT X TO THE TOP OF THE HI-RES TEXT SCREEN
3282   F68F 10 21 09 6D                LBRN  RAMLINK                         RAM HOOK
3283   F693 BF FE 00                   STX   H.CRSLOC                        SAVE THE START OF THE HI-RES TEXT SCREEN
3284   F696 86 20                      LDA   #SPACE                          INITIALIZE CHARACTERS TO SPACES
3285   F698 F6 FE 08                   LDB   H.CRSATT                        GET THE CHARACTER ATTRIBUTES
3286   F69B ED 81             LF69B    STD   ,X++                            SAVE THE CHARACTER AND ATTRIBUTES IN HI-RES TEXT SCREEN
3287   F69D BC FE 06                   CMPX  H.DISPEN                        COMPARE TO THE END OF HI-RES TEXT SCREEN
3288   F6A0 25 F9                      BCS   LF69B                           LOOP UNTIL ALL MEMORY INITIALIZED
3289   F6A2 8E 20 00                   LDX   #HRESSCRN                       RESET X TO THE TOP OF THE SCREEN
3290   F6A5 4F                         CLRA
3291   F6A6 B7 FE 02                   STA   H.CURSX                         SET THE CURSOR X COORDINATE (COLUMN) TO ZERO
3292   F6A9 B7 FE 03                   STA   H.CURSY                         SET THE CURSOR Y COORDIANTE (ROW) TO ZERO
3293   F6AC 39                         RTS
3294
3295                          * CLS PATCH ENTERED FROM $8C4C
3296   F6AD 35 01             ALINK23  PULS  CC                              RESTORE THE ZERO FLAG
3297   F6AF 10 21 09 4D                LBRN  RAMLINK                         RAM HOOK
3298   F6B3 27 2B                      BEQ   LF6E0                           CLEAR THE SCREEN CURSOR ATTRIBUTES IF NO ARGUMENT
3299   F6B5 BD B7 0B                   JSR   EVALEXPB                        EVALUATE EXPRESSION, RETURN VALUE IN ACCB
3300   F6B8 5D                         TSTB                                  TEST ARGUMENT
3301   F6B9 27 25                      BEQ   LF6E0                           BRANCH IF CLS 0
3302   F6BB C1 08                      CMPB  #$08                            CHECK FOR CLS 8
3303   F6BD 22 28                      BHI   LF6E7                           BRANCH IF > CLS 8
3304   F6BF 5A                         DECB                                  CHANGE 1-8 TO 0-7
3305   F6C0 31 8D EF B4                LEAY  IM.PALET,PC                     POINT TO THE PALETTE REGISTER IMAGES
3306   F6C4 A6 A5                      LDA   B,Y                             GET THE COLOR IN THE PALETTE REGISTER
3307   F6C6 B7 FF 9A                   STA   V.BORDER                        AND SAVE IT AS THE NEW BORDER COLOR
3308   F6C9 17 00 9A                   LBSR  LF766                           SET THE BORDER COLOR IN THE 40 & 80 COLUMN VIDEO MODE IMAGES
3309   F6CC F7 FE 08                   STB   H.CRSATT                        SAVE THE ADJUSTED CLS ARGUMENT AS THE NEW ATTRIBUTE BYTE
3310   F6CF 86 20                      LDA   #SPACE
3311   F6D1 17 00 9E                   LBSR  LF772                           PUT THE HI-RES TEXT SCREEN INTO LOGICAL BLOCK 1
3312   F6D4 8E 20 00                   LDX   #HRESSCRN                       POINT X TO THE TOP OF THE HI-RES TEXT SCREEN
3313   F6D7 BF FE 00                   STX   H.CRSLOC                        PUT THE CURSOR AT THE TOP OF THE SCREEN
3314   F6DA 8D BF                      BSR   LF69B                           CLEAR THE SCREEN
3315   F6DC 17 00 99          LF6DC    LBSR  LF778                           REMOVE THE HI-RES TEXT SCREEN FROM THE LOGICAL ADDRESS SPACE
3316   F6DF 39                         RTS
3317   F6E0 17 00 8F          LF6E0    LBSR  LF772                           PUT THE HI-RES TEXT SCREEN INTO LOGICAL BLOCK 1
3318   F6E3 8D A7                      BSR   LF68C                           CLEAR THE HI-RES TEXT SCREEN
3319   F6E5 20 F5                      BRA   LF6DC                           PUT BLOCK 7.1 BACK INTO LOGICAL BLOCK 1
3320   F6E7 7F FE 08          LF6E7    CLR   H.CRSATT                        RESET THE ATTRIBUTE BYTE TO ZERO
3321   F6EA B6 E6 78                   LDA   IM.PALET                        GET THE COLOR IN PALETTE REGISTER 0
3322   F6ED B7 FF 9A                   STA   V.BORDER                        AND SAVE IT AS THE NEW BORDER COLOR
3323   F6F0 8D 74                      BSR   LF766                           ALSO SAVE IT IN THE 40 AND 80 COLUMN VIDEO REGISTER IMAGES
3324   F6F2 C1 64                      CMPB  #100                            CHECK FOR CLS 100
3325   F6F4 27 3A                      BEQ   LF730                           IF CLS 100, THEN PRINT THE AUTHORS' NAMES - THIS WILL ONLY BE
3326                                                                         DONE THE FIRST TIME CLS 100 IS EXECUTED, THIS CODE WILL BE
3327                                                                         OVERWRITTEN BY NOPs WHEN THE AUTHORS' NAMES ARE DISPLAYED.
3328   F6F6 8D 7A                      BSR   LF772                           PUT THE HI-RES TEXT SCREEN INTO LOGICAL BLOCK 1
3329   F6F8 8D 92                      BSR   LF68C                           CLEAR THE HI-RES TEXT SCREEN
3330   F6FA 8D 7C                      BSR   LF778                           PUT BLOCK 7.1 BACK INTO LOGICAL BLOCK 1
3331   F6FC 8E F7 01                   LDX   #MICROMS-1                      POINT TO MICROWARE'S COMMERCIAL MESSAGE
3332   F6FF 7E B9 9C                   JMP   STRINOUT                        COPY A STRING TO CONSOLE OUT ($B99C)
3333
3334                          * MICROWARE COMMERCIAL
3335   F702 4D 69 63 72 6F 77 MICROMS  FCC   'T.Harris & T.Earles'
3336   F708 61 72 65 20 53 79
3337   F70E 73 74 65 6D 73 20
3338   F714 43 6F 72 70 2E
3339   F719 0D 00             LF719    FCB   $0D,$00
3340
3341                          * NAMES OF THE AUTHORS
3342                          * THE INITIALIZATION CODE WILL COPY THE AUTHOR'S NAMES INTO THIS SPOT
3343   F71B 00 00 00 00 00 00 AUTHORMS FCB   $00,$00,$00,$00,$00,$00
3344   F721 00 00 00 00 00 00          FCB   $00,$00,$00,$00,$00,$00
3345   F727 00 00 00 00 00 00          FCB   $00,$00,$00,$00,$00,$00
3346   F72D 00 00 00                   FCB   $00,$00,$00
3347
3348   F730 8D 40             LF730    BSR   LF772                           PUT THE HI-RES TEXT SCREEN INTO LOGICAL BLOCK 1
3349   F732 17 FF 57                   LBSR  LF68C                           CLEAR THE HI-RES TEXT SCREEN
3350   F735 8D 41                      BSR   LF778                           PUT BLOCK 7.1 BACK INTO LOGICAL BLOCK 1
3351   F737 8E F7 1A                   LDX   #AUTHORMS-1                     POINT TO THE AUTHOR MESSAGE
3352   F73A BD B9 9C                   JSR   STRINOUT                        COPY A STRING TO CONSOLE OUT
3353   F73D 34 10                      PSHS  X
3354   F73F 30 8D FF B1                LEAX  LF6F4,PC                        POINT TO THE INSTRUCTION WHICH BRANCHES TO THIS ROUTINE
3355   F743 86 12                      LDA   #$12                            OP CODE OF A NOP
3356   F745 A7 80                      STA   ,X+                             REPLACE THE BRANCH TO THIS ROUTINE WITH 2 NOPs MAKING IT SO
3357   F747 A7 84                      STA   ,X                              THAT THIS ROUTINE MAY ONLY BE ENTERED ONE TIME
3358   F749 30 8D FF CE                LEAX  AUTHORMS,PC                     POINT TO THE AUTHORS CODED NAMES
3359                          * REPLACE THE AUTHORS' NAMES AND THE CODE THAT DISPLAYS THEM WITH NOPs
3360   F74D A7 80             LF74D    STA   ,X+                             SAVE A NOP
```

```
3361   F74F 8C F7 4D                  CMPX  #LF74D                          CHECK FOR END OF THE DISPLAY NAME ROUTINE
3362   F752 25 F9                     BCS   LF74D                           LOOP UNTIL DONE
3363   F754 35 10                     PULS  X                               RESTORE X; THIS AND THE RTS FOLLOWING SHOULD BE PULS X,PC
3364   F756 39                        RTS
3365
3366                          * GET AN INPUT LINE FOR BASIC PATCH ENTERED FROM $A38D
3367   F757 0D E7               ALINK27 TST   HRWIDTH                         CHECK FOR HI-RES TEXT MODE
3368   F759 26 06                     BNE   LF761                           BRANCH IF IN A HI-RES TEXT MODE
3369   F75B BD A9 28                  JSR   LA928                           CLEAR THE 32 COLUMN SCREEN
3370   F75E 7E A3 90            LF75E   JMP   LA390                         RE-ENTER THE MAIN STREAM OF CODE ($A390)
3371   F761 17 FF 7C            LF761   LBSR  LF6E0                         RESET THE HI-RES TEXT SCREEN
3372   F764 20 F8                     BRA   LF75E
3373
3374                          * SAVE THE VALUE IN ACCA AS THE BORDER COLOR IN THE 40 AND 80 COLUMN VIDEO MODE IMAGES
3375   F766 34 20               LF766   PSHS  Y
3376   F768 31 8D E8 CF               LEAY  LE8CF,PC                        POINT TO THE 40 COLUMN MODE REGISTER IMAGE
3377   F76C A7 23                     STA   $03,Y                           SAVE THE BORDER COLOR IN THE 40 COLUMN VIDEO MODE REGISTER IMAGE
3378   F76E A7 2C                     STA   $0C,Y                           SAVE THE BORDER COLOR IN THE 80 COLUMN VIDEO MODE REGISTER IMAGE
3379   F770 35 A0                     PULS  Y,PC
3380
3381   F772 1A 50               LF772   ORCC  #$50                           DISABLE THE INTERRUPTS
3382   F774 17 E9 3E                  LBSR  SELTEXT                         PUT BLOCK 6.6 INTO LOGICAL BLOCK 1
3383   F777 39                        RTS
3384   F778 17 E9 1C            LF778   LBSR  SETMMU                         COPY THE MMU IMAGES INTO THE MMU REGISTERS
3385   F77B 1C AF                     ANDCC #$AF                            ENABLE IRQ, FIRQ
3386   F77D 39                        RTS
3387                          * PATCH 24 - BLINK THE CURSOR PATCH ENTERED FROM $A0D4
3388   F77E 8D 07               ALINK24 BSR   LF787                           BLINK THE CURSOR
3389   F780 BD A1 CB                  JSR   KEYIN                           GET A KEY
3390   F783 27 F9                     BEQ   ALINK24                         LOOP UNTIL A KEY IS PRESSED
3391   F785 35 94                     PULS  B,X,PC
3392   F787 0A 94               LF787   DEC   BLKCNT                         DECREMENT THE CURSOR BLINK DELAY
3393   F789 26 1D                     BNE   LF7A8                           IT'S NOT TIME TO BLINK THE CURSOR
3394   F78B C6 0B                     LDB   #11                             CURSOR BLINK DELAY CONSTANT
3395   F78D D7 94                     STB   BLKCNT                          RESET THE CURSOR BLINK DELAY COUNTER
3396   F78F 8D E1                     BSR   LF772                           PUT THE HI-RES TEXT SCREEN INTO THE LOGICAL ADDRESS SPACE
3397   F791 BE FE 00                  LDX   H.CRSLOC                        POINT TO THE CURSOR CHARACTER
3398   F794 A6 01                     LDA   $01,X                           GET THE CURSOR CHARACTER'S ATTRIBUTES
3399   F796 85 40                     BITA  #$40                            IS THE UNDERLINE MODE ACTIVE?
3400   F798 27 05                     BEQ   LF79F                           BRANCH IF NOT ACTIVE UNDERLINE
3401   F79A B6 FE 08                  LDA   H.CRSATT                        GET THE CURSOR ATTRIBUTES RAM IMAGE
3402   F79D 20 05                     BRA   LF7A4                           PUT IT ON THE SCREEN
3403   F79F B6 FE 08            LF79F   LDA   H.CRSATT                        GET THE CURSOR ATTRIBUTES RAM IMAGE
3404   F7A2 8A 40                     ORA   #$40                            FORCE THE UNDERLINE ATTRIBUTE
3405   F7A4 A7 01               LF7A4   STA   $01,X                           SAVE THE NEW CURSOR ATTRIBUTES IN THE HI-RES TEXT SCREEN
3406   F7A6 8D D0                     BSR   LF778                           RESTORE THE NORMAL BASIC PROGRAM BLOCK TO LOGICAL BLOCK 1
3407   F7A8 8E 04 5E            LF7A8   LDX   #DEBDEL                        GET THE KEYBOARD DEBOUNCE DELAY
3408   F7AB 7E A7 D3                  JMP   LA7D3                           GO WAIT A WHILE ($A7D3)
3409
3410                          * PATCH 22 - PUT A CHARACTER ON THE SCREEN PATCH ENTERED FROM $BC3D
3411   F7AE 8D C2               ALINK22 BSR   LF772                           PUT THE HI-RES TEXT SCREEN INTO THE LOGICAL ADDRESS SPACE
3412   F7B0 10 21 08 4C               LBRN  RAMLINK                         RAM HOOK
3413   F7B4 BE FE 00                  LDX   H.CRSLOC                        POINT TO THE CURSOR CHARACTER
3414   F7B7 81 08                     CMPA  #BS                             BACKSPACE CHARACTER?
3415   F7B9 26 09                     BNE   LF7C4                           NO
3416
3417                          DO A BACKSPACE HERE
3418   F7BB 8C 20 00                  CMPX  #HRESSCRN                       ARE WE AT THE UPPER LEFT-HAND CORNER OF THE SCREEN?
3419   F7BE 27 1E                     BEQ   LF7DE                           YES, DO NOT ALLOW A BACKSPACE
3420   F7C0 8D 20                     BSR   LF7E2                           DO A BACKSPACE ON THE HI-RES SCREEN
3421   F7C2 20 1A                     BRA   LF7DE
3422   F7C4 81 0D               LF7C4   CMPA  #CR                             ENTER KEY?
3423   F7C6 26 04                     BNE   LF7CC                           NO
3424   F7C8 8D 5D                     BSR   LF827                           DO A CARRIAGE RETURN ON THE HI-RES SCREEN
3425   F7CA 20 0B                     BRA   LF7D7                           CHECK TO SEE IF THE SCREEN SHOULD BE SCROLLED
3426   F7CC 81 20               LF7CC   CMPA  #$20                           CHECK FOR A CONTROL CHARACTER
3427   F7CE 25 0E                     BCS   LF7DE                           DO NOTHING IF A CONTROL CHARACTER
3428   F7D0 F6 FE 08                  LDB   H.CRSATT                        GET THE CURSOR ATTRIBUTES RAM IMAGE
3429   F7D3 ED 84                     STD   ,X                              PUT THE NEW CHARACTER AND ATTRIBUTES INTO THE HI-RES TEXT SCREEN
3430   F7D5 8D 30                     BSR   LF807                           MOVE THE CURSOR FORWARD ONE CHARACTER
3431   F7D7 BC FE 06            LF7D7   CMPX  H.DISPEN                        CHECK FOR THE END OF THE HI-RES TEXT SCREEN
3432   F7DA 25 02                     BCS   LF7DE                           BRANCH IF NOT AT THE END
3433   F7DC 8D 76                     BSR   LF854                           SCROLL THE SCREEN UP ONE ROW
3434   F7DE 8D 98               LF7DE   BSR   LF778                           RESTORE THE NORMAL BASIC PROGRAM BLOCK TO LOGICAL BLOCK 1
3435   F7E0 35 96                     PULS  A,B,X,PC
3436
3437                          DO A HI-RES BACKSPACE HERE
3438   F7E2 34 06               LF7E2   PSHS  B,A
3439   F7E4 86 20                     LDA   #SPACE                          SPACE CHARACTER
3440   F7E6 F6 FE 08                  LDB   H.CRSATT                        GET THE ATTRIBUTES RAM IMAGE
3441   F7E9 ED 84                     STD   ,X                              SAVE A SPACE ON THE SCREEN AT THE OLD CURSOR POSITION
3442   F7EB CA 40                     ORB   #$40                            FORCE THE UNDERLINE ATTRIBUTE
3443   F7ED ED 1E                     STD   $-02,X                          SAVE AN UNDERLINED SPACE AS THE NEW CURSOR CHARACTER
3444   F7EF 30 1E                     LEAX  $-02,X                          MOVE THE CURSOR POINTER BACK TWO
3445   F7F1 BF FE 00                  STX   H.CRSLOC                        AND SAVE IT IN RAM
3446   F7F4 FC FE 02                  LDD   H.CURSX                         GET THE COLUMN AND ROW POSITION OF THE OLD CURSOR
3447   F7F7 4A                        DECA                                  BUMP THE COLUMN NUMBER DOWN ONE
3448   F7F8 2A 08                     BPL   LF802                           BRANCH IF NO WRAP-AROUND
3449   F7FA 5A                        DECB                                  BUMP THE ROW COUNTER DOWN ONE
3450   F7FB F7 FE 03                  STB   H.CURSY                         SAVE THE NEW CURSOR ROW NUMBER
3451   F7FE B6 FE 04                  LDA   H.COLUMN                        GET THE NUMBER OF CHARACTERS PER ROW
3452   F801 4A                        DECA                                  MAKE THE HIGHEST ALLOWABLE COLUMN NUMBER (ZERO IS FIRST)
3453   F802 B7 FE 02            LF802   STA   H.CURSX                         SAVE THE NEW CURSOR COLUMN NUMBER
3454   F805 35 86                     PULS  A,B,PC
3455   F807 34 06               LF807   PSHS  B,A
3456   F809 86 20                     LDA   #$20                            GET THE CURSOR CHARACTER
```

```
3457   F80B F6 FE 08            LDB   H.CRSATT                  GET THE CURSOR ATTRIBUTES RAM IMAGE
3458   F80E CA 40               ORB   #$40                      FORCE THE UNDERLINE ATTRIBUTE
3459   F810 30 02               LEAX  $02,X                     MOVE THE POINTER UP ONE CHARACTER POSITION
3460   F812 ED 84               STD   ,X                        SAVE THE NEW CHARACTER ATTRIBUTES IN THE HI-RES TEXT SCREEN
3461   F814 BF FE 00            STX   H.CRSLOC                  SAVE THE NEW CURSOR POSITION
3462   F817 FC FE 02            LDD   H.CURSX                   GET THE OLD CURSOR ROW AND COLUMN NUMBERS
3463   F81A 4C                  INCA                            BUMP THE COLUMN NUMBER UP ONE
3464   F81B B1 FE 04            CMPA  H.COLUMN                  CHECK FOR WRAP-AROUND TO NEXT ROW
3465   F81E 25 E2               BCS   LF802                     BRANCH IF NO WRAP-AROUND
3466   F820 5C                  INCB                            BUMP THE ROW NUMBER UP ONE
3467   F821 F7 FE 03            STB   H.CURSY                   SAVE THE NEW ROW NUMBER
3468   F824 4F                  CLRA                            SET THE COLUMN NUMBER TO ZERO
3469   F825 20 DB               BRA   LF802
3470                         * DO A HI-RES CARRIAGE RETURN
3471   F827 34 06             LF827 PSHS  B,A
3472   F829 86 20               LDA   #SPACE                    SPACE CHARACTER
3473   F82B F6 FE 08            LDB   H.CRSATT                  GET THE CURSOR ATTRIBUTES RAM IMAGE
3474   F82E ED 81             LF82E STD   ,X++                  SAVE A SPACE CHARACTER AND ADVANCE THE CURSOR POINTER ONE CHARACTER
3475   F830 34 02               PSHS  A
3476   F832 B6 FE 02            LDA   H.CURSX                   GET THE CURSOR'S COLUMN NUMBER
3477   F835 4C                  INCA                            BUMP IT UP ONE
3478   F836 B7 FE 02            STA   H.CURSX                   SAVE THE NEW COLUMN NUMBER
3479   F839 B1 FE 04            CMPA  H.COLUMN                  HAS IT WRAPPED AROUND?
3480   F83C 35 02               PULS  A
3481   F83E 25 EE               BCS   LF82E                     BRANCH IF NO WRAP-AROUND
3482   F840 BF FE 00            STX   H.CRSLOC                  SAVE THE NEW CURSOR POINTER
3483   F843 7F FE 02            CLR   H.CURSX                   SET THE CURSOR COLUMN NUMBER TO ZERO
3484   F846 7C FE 03            INC   H.CURSY                   BUMP THE ROW NUMBER UP ONE
3485   F849 86 20               LDA   #$20                      GET THE CURSOR CHARACTER
3486   F84B F6 FE 08            LDB   H.CRSATT                  ACCB ALREADY CONTAINS THIS VALUE
3487   F84E CA 40               ORB   #$40                      FORCE THE UNDERLINE ATTRIBUTE
3488   F850 ED 84               STD   ,X                        SAVE AN UNDERLINED CHARACTER AS THE NEW CURSOR CHARACTER
3489   F852 35 86               PULS  A,B,PC
3490
3491                         * SCROLL THE SCREEN
3492   F854 34 06             LF854 PSHS  B,A
3493   F856 8E 20 00            LDX   #HRESSCRN                 POINT TO THE START OF THE HI-RES TEXT SCREEN
3494   F859 B6 FE 04            LDA   H.COLUMN                  GET THE NUMBER OF CHARACTERS PER ROW
3495   F85C 81 28               CMPA  #40                       40 CHARACTERS PER ROW?
3496   F85E 26 0E               BNE   LF86E                     BRANCH IF 80 CHARACTERS PER ROW
3497
3498                         * SCROLL A 40 CHARACTER ROW
3499   F860 EC 88 50          LF860 LDD   2*40,X                GET A CHARACTER AND ATTRIBUTE FROM ONE ROW DOWN
3500   F863 ED 81               STD   ,X++                      AND MOVE THEM UP TO THE PRESENT ROW
3501   F865 8C 27 30            CMPX  #HRESSCRN+(ROWMAX-1)*40*2 PAST THE END OF THE HI-RES TEXT SCREEN?
3502   F868 25 F6               BCS   LF860                     NO, KEEP MOVING CHARACTERS AND ATTRIBUTES
3503   F86A 8D 0F             LF86A BSR   LF87B                 FILL THE LAST ROW WITH SPACES
3504   F86C 35 86               PULS  A,B,PC
3505
3506                         * SCROLL AN 80 CHARACTER SCREEN
3507   F86E EC 89 00 A0       LF86E LDD   80*2,X                GET A CHARACTER AND ATTRIBUTES FROM ONE ROW DOWN
3508   F872 ED 81               STD   ,X++                      AND MOVE THEM UP TO THE PRESENT ROW
3509   F874 8C 2E 60            CMPX  #HRESSCRN+(ROWMAX-1)*80*2 PAST THE END OF THE HI-RES TEXT SCREEN?
3510   F877 25 F5               BCS   LF86E                     NO, KEEP MOVING CHARACTERS AND ATTRIBUTES
3511   F879 20 EF               BRA   LF86A
3512
3513                         * FILL THE LAST ROW WITH SPACES
3514   F87B 7F FE 02          LF87B CLR   H.CURSX               RESET THE COLUMN NUMBER TO ZERO
3515   F87E 86 17               LDA   #ROWMAX-1                 GET THE HIGHEST ROW NUMBER (ZERO IS LOWEST)
3516   F880 B7 FE 03            STA   H.CURSY                   AND SAVE IT AS THE CURRENT ROW NUMBER
3517   F883 86 20               LDA   #SPACE                    SPACE CHARACTER
3518   F885 F6 FE 08            LDB   H.CRSATT                  GET THE ATTRIBUTES RAM IMAGE
3519   F888 34 10               PSHS  X                         SAVE THE CURRENT CHARACTER POINTER
3520   F88A ED 81             LF88A STD   ,X++                  SAVE A CHARACTER AND ATTRIBUTES TO THE HI-RES TEXT SCREEN
3521   F88C BC FE 06            CMPX  H.DISPEN                  CHECK FOR THE END OF THE HI-RES TEXT SCREEN
3522   F88F 26 F9               BNE   LF88A                     BRANCH IF NOT AT THE END OF THE HI-RES TEXT SCREEN
3523   F891 7F FE 02            CLR   H.CURSX                   RESET THE COLUMN NUMBER TO ZERO
3524   F894 35 10               PULS  X                         RESTORE THE CHARACTER POINTER
3525   F896 86 20               LDA   #$20                      GET THE CURSOR CHARACTER
3526   F898 F6 FE 08            LDB   H.CRSATT                  GET THE CURSOR ATTRIBUTES RAM IMAGE
3527   F89B CA 40               ORB   #$40                      FORCE THE UNDERLINE ATTRIBUTE
3528   F89D ED 84               STD   ,X                        SAVE THE NEW CURSOR CHARACTER
3529   F89F BF FE 00            STX   H.CRSLOC                  SAVE THE NEW CURSOR POINTER
3530   F8A2 39                  RTS
3531
3532                         * PRINT @ PATCH ENTERED FROM $B902
3533   F8A3 0D 6F             ALINK26 TST   DEVNUM               CHECK THE DEVICE NUMBER
3534   F8A5 26 04               BNE   LF8AB                     BRANCH IF NOT THE SCREEN
3535   F8A7 0D E7               TST   HRWIDTH                   CHECK THE HI-RES TEXT MODE
3536   F8A9 26 06               BNE   LF8B1                     BRANCH IF A HI-RES TEXT MODE IS SET
3537   F8AB BD A3 5F          LF8AB JSR   LA35F                 SET UP THE PRINT PARAMETERS
3538   F8AE 7E B9 5F            JMP   LB95F                     RE-ENTER THE MAIN STREAM OF CODE ($B95F)
3539   F8B1 17 FE BE          LF8B1 LBSR  LF772                 PUT THE HI-RES TEXT SCREEN INTO THE LOGICAL ADDRESS SPACE
3540   F8B4 7D FE 02            TST   H.CURSX                   CHECK THE CURSOR'S X COORDINATE
3541   F8B7 34 01               PSHS  CC                        SAVE THE ZERO FLAG
3542   F8B9 17 FE BC            LBSR  LF778                     RESTORE THE NORMAL BASIC PROGRAM BLOCK TO LOGICAL BLOCK 1
3543   F8BC 35 01               PULS  CC                        RESTORE THE ZERO FLAG
3544   F8BE 10 26 C0 96         LBNE  LB958                     BRANCH IF THE CURSOR IS NOT AT THE START OF THE LINE ($B958)
3545   F8C2 39                  RTS
3546
3547                         * PRINT @ PATCH ENTERED FROM $B902
3548   F8C3 0D E7             ALINK25 TST   HRWIDTH              CHECK THE HI-RES TEXT MODE
3549   F8C5 26 06               BNE   LF8CD                     'HP' ERROR IF THE HI-RES TEXT MODE IS NOT SET
3550   F8C7 BD A5 54            JSR   LA554                     MOVE THE CURSOR TO THE PROPER PRINT POSITION
3551   F8CA 7E B9 05            JMP   LB905                     RE-ENTER THE MAIN STREAM OF CODE ($B905)
3552   F8CD C6 4E             LF8CD LDB   #39*2                 'HP' ERROR
```

```
3553   F8CF 7E AC 46                JMP   LAC46                    JUMP TO ERROR HANDLER ($AC46)
3554
3555                        * LOCATE
3556   F8D2 D6 E7           LOCATE  LDB   HRWIDTH                  IS THE HI-RES TEXT MODE ENABLED?
3557   F8D4 10 21 07 28             LBRN  RAMLINK                  RAM HOOK
3558   F8D8 27 F3                   BEQ   LF8CD                    'HP' ERROR IF NOT ENABLED
3559   F8DA 34 04                   PSHS  B                        SAVE THE HI-RES TEXT MODE
3560   F8DC BD E7 B2                JSR   LE7B2                    EVALUATE TWO EXPRESSIONS
3561   F8DF 96 2C                   LDA   BINVAL+1                 GET THE FIRST OF THE TWO EXPRESSIONS (COLUMN NUMBER)
3562   F8E1 35 04                   PULS  B                        RESTORE THE FIRST ARGUMENT
3563   F8E3 C1 01                   CMPB  #$01                     GET BACK THE HI-RES TEXT MODE
3564   F8E5 26 04                   BNE   LF8EB                    BRANCH IF NOT 40 COLUMN MODE
3565   F8E7 81 28                   CMPA  #40                      40 COLUMNS MAXIMUM IN 40 COLUMN MODE
3566   F8E9 20 02                   BRA   LF8ED                    DO A RANGE CHECK
3567   F8EB 81 50           LF8EB   CMPA  #80                      80 COLUMNS MAXIMUM IN 80 COLUMN MODE
3568   F8ED 10 24 BB 59     LF8ED   LBCC  ILLFUNC                  ILLEGAL FUNCTION CALL ERROR
3569   F8F1 D6 C0                   LDB   VERBEG+1                 GET THE SECOND ARGUMENT (ROW NUMBER)
3570   F8F3 C1 18                   CMPB  #ROWMAX                  RANGE CHECK ON THE ROW NUMBER
3571   F8F5 24 F6                   BCC   LF8ED                    'FC' ERROR IF ROW NUMBER IS TOO LARGE
3572   F8F7 34 06                   PSHS  B,A                      SAVE THE COLUMN AND ROW NUMBERS
3573   F8F9 17 FE 76                LBSR  LF772                    PUT THE HI-RES TEXT SCREEN INTO THE LOGICAL ADDRESS SPACE
3574   F8FC FD FE 02                STD   H.CURSX                  SAVE THE NEW COLUMN AND ROW NUMBERS AS THOSE OF THE CURSOR
3575   F8FF BE FE 00                LDX   H.CRSLOC                 GET THE CURRENT CURSOR POINTER
3576   F902 B6 FE 08                LDA   H.CRSATT                 GET THE CURSOR ATTRIBUTES RAM IMAGE
3577   F905 A7 01                   STA   $01,X                    AND SAVE IT AS THE ATTRIBUTES IN THE OLD CURSOR POSITION
3578   F907 B6 FE 04                LDA   H.COLUMN                 GET THE NUMBER OF CHARACTERS/ROW
3579   F90A 48                      ALSA                           MULTIPLY BY TWO - TWO BYTES PER CHARACTER (CHAR AND ATTR)
3580   F90B 3D                      MUL                            GET THE ROW OFFSET TO THE PROPER CHARACTER
3581   F90C 8E 20 00                LDX   #HRESSCRN                POINT TO THE START OF THE HI-RES TEXT SCREEN
3582   F90F 30 8B                   LEAX  D,X                      ADD ROW OFFSET TO THE START OF THE HI-RES TEXT SCREEN
3583   F911 35 06                   PULS  A,B                      RESTORE THE NEW CURSOR COLUMN AND ROW NUMBERS
3584   F913 48                      ALSA                           MULTIPLY COLUMN NUMBER BY TWO - TWO BYTES PER CHARACTER (CHAR AND ATTR)
3585   F914 1F 89                   TFR   A,B                      SAVE COLUMN OFFSET IN ACCB
3586   F916 3A                      ABX                            ADD THE COLUMN OFFSET TO THE CURRENT CURSOR POINTER
3587   F917 B6 FE 08                LDA   H.CRSATT                 GET THE CURSOR ATTRIBUTES RAM IMAGE
3588   F91A 8A 40                   ORA   #$40                     FORCE UNDERLINE ATTRIBUTE
3589   F91C A7 01                   STA   $01,X                    SAVE THE NEW CURSOR ATTRIBUTE IN THE HI-RES TEXT SCREEN
3590   F91E BF FE 00                STX   H.CRSLOC                 SAVE THE NEW CURSOR POINTER
3591   F921 17 FE 54                LBSR  LF778                    RESTORE THE NORMAL BASIC PROGRAM BLOCK TO LOGICAL BLOCK 1
3592   F924 39                      RTS
3593
3594                        * HSTAT
3595   F925 0D E7           HSTAT   TST   HRWIDTH                  IS THE HI-RES TEXT MODE ENABLED?
3596   F927 10 21 06 D5             LBRN  RAMLINK                  RAM HOOK
3597   F92B 27 A0                   BEQ   LF8CD                    'HP' ERROR IF HI-RES TEXT MODE NOT ENABLED
3598   F92D 17 FE 42                LBSR  LF772                    PUT THE HI-RES TEXT SCREEN INTO THE LOGICAL ADDRESS SPACE
3599   F930 BE FE 00                LDX   H.CRSLOC                 GET THE CURRENT CURSOR POINTER
3600   F933 EC 84                   LDD   ,X                       GET THE CURSOR CHARACTER ATTRIBUTES
3601   F935 DD CB                   STD   VCB                      AND SAVE THEM
3602   F937 FC FE 02                LDD   H.CURSX                  GET THE CURRENT COLUMN AND ROW NUMBER
3603   F93A DD CD                   STD   VCD                      AND SAVE THEM
3604   F93C 17 FE 39                LBSR  LF778                    RESTORE THE NORMAL BASIC PROGRAM BLOCK TO LOGICAL BLOCK 1
3605   F93F BD B3 57                JSR   LB357                    EVALUATE A VARIABLE; RETURN X POINTING TO THE VARIABLE DESCRIPTOR
3606   F942 9F 3B                   STX   VARDES                   SAVE THE VARIABLE DESCRIPTOR
3607   F944 BD B2 6D                JSR   SYNCOMMA                 SYNTAX CHECK FOR A COMMA
3608   F947 C6 01                   LDB   #$01
3609   F949 BD B5 6D                JSR   LB56D                    RESERVE SPACE FOR A ONE CHARACTER STRING IN STRING SPACE
3610   F94C 96 CB                   LDA   VCB                      GET THE CURSOR CHARACTER
3611   F94E BD B5 11                JSR   LB511                    THIS IS REALLY A WASTE - THE JSR LB56D ABOVE SHOULD JUST BE A
3612                                                               JSR LB50D AND THE JSR LB511 WOULD NOT BE NECESSARY
3613   F951 A7 84                   STA   ,X                       SAVE THE CURSOR CHARACTER IN THE NEWLY RESERVED STRING SPACE
3614   F953 BD B5 4C                JSR   LB54C                    PUT THE STRING ONTO THE STRING STACK
3615   F956 9E 3B                   LDX   VARDES                   POINT TO THE STRING'S VARIABLE DESCRIPTOR
3616   F958 6D 1F                   TST   $-01,X                   CHECK THE SECOND CHARACTER OF THE VARIABLE NAME
3617   F95A 10 2A B7 F3             LBPL  TMERROR                  TYPE MISMATCH ERROR IF NUMERIC VARIABLE
3618   F95E 10 9E 52                LDY   FPA0+2                   POINT Y TO THE START OF THE STRING DESCRIPTOR
3619   F961 C6 05                   LDB   #$05                     VARIABLE DESCRIPTORS ARE 5 BYTES LONG
3620   F963 A6 A0           LF963   LDA   ,Y+                      * COPY THE DATA FROM THE STRING DESCRIPTOR
3621   F965 A7 80                   STA   ,X+                      * TO THE VARIABLE DESCRIPTOR
3622   F967 5A                      DECB                           DECREMENT THE DESCRIPTOR COUNTER
3623   F968 26 F9                   BNE   LF963                    LOOP UNTIL DONE
3624   F96A 9E 0B                   LDX   TEMPPT                   * THIS CODE IS DESIGNED TO REMOVE THE ABOVE ALLOCATED STRING FROM
3625   F96C 30 1B                   LEAX  $-05,X                   * THE STRING STACK - IT MAY CAUSE BUGS BECAUSE IT DOESN'T RESET
3626   F96E 9F 0B                   STX   TEMPPT                   * LASTPT; LDX LASTPT, JSR LB675 WOULD BE MUCH BETTER
3627   F970 BD B3 57                JSR   LB357                    EVALUATE A VARIABLE; RETURN X POINTING TO THE VARIABLE DESCRIPTOR
3628   F973 9F 3B                   STX   VARDES                   SAVE THE VARIABLE DESCRIPTOR
3629   F975 BD B2 6D                JSR   SYNCOMMA                 SYNTAX CHECK FOR A COMMA
3630   F978 4F                      CLRA                           ZERO OUT THE MS BYTE OF ACCD
3631   F979 D6 CC                   LDB   VCB+1                    GET THE CURSOR ATTRIBUTES
3632   F97B BD B4 F4                JSR   GIVABF                   CONVERT ACCD TO FLOATING POINT
3633   F97E 9E 3B                   LDX   VARDES                   POINT X TO THE VARIABLE DESCRIPTOR
3634   F980 6D 1F                   TST   $-01,X                   CHECK THE SECOND CHARACTER OF THE VARIABLE NAME
3635   F982 10 2B B7 CB             LBMI  TMERROR                  TYPE MISMATCH ERROR IF STRING VARIABLE
3636   F986 BD BC 35                JSR   LBC35                    PACK FPA0 AND STORE IT IN THE DESCRIPTOR POINTED TO BY X
3637   F989 BD B3 57                JSR   LB357                    EVALUATE A VARIABLE; RETURN X POINTING TO THE VARIABLE DESCRIPTOR
3638   F98C 9F 3B                   STX   VARDES                   SAVE THE VARIABLE DESCRIPTOR
3639   F98E BD B2 6D                JSR   SYNCOMMA                 SYNTAX CHECK FOR A COMMA
3640   F991 4F                      CLRA                           ZERO OUT THE MS BYTE OF ACCD
3641   F992 D6 CD                   LDB   VCD                      GET THE X COORDINATE OF THE CURSOR POSITION
3642   F994 BD B4 F4                JSR   GIVABF                   CONVERT ACCD TO FLOATING POINT
3643   F997 9E 3B                   LDX   VARDES                   POINT X TO THE VARIABLE DESCRIPTOR
3644   F999 6D 1F                   TST   $-01,X                   CHECK THE SECOND CHARACTER OF THE VARIABLE NAME
3645   F99B 10 2B B7 B2             LBMI  TMERROR                  TYPE MISMATCH ERROR IF STRING VARIABLE
3646   F99F BD BC 35                JSR   LBC35                    PACK FPA0 AND STORE IT IN THE DESCRIPTOR POINTED TO BY X
3647   F9A2 BD B3 57                JSR   LB357                    EVALUATE A VARIABLE; RETURN X POINTING TO THE VARIABLE DESCRIPTOR
3648   F9A5 9F 3B                   STX   VARDES                   SAVE THE VARIABLE DESCRIPTOR
```

```
3649   F9A7 4F                     CLRA                        ZERO OUT THE MS BYTE OF ACCD
3650   F9A8 D6 CE                  LDB   VCD+1                 GET THE Y COORDINATE OF THE CURSOR POSITION
3651   F9AA BD B4 F4               JSR   GIVABF                CONVERT ACCD TO FLOATING POINT
3652   F9AD 9E 3B                  LDX   VARDES                POINT X TO THE VARIABLE DESCRIPTOR
3653   F9AF 6D 1F                  TST   $-01,X                CHECK THE SECOND CHARACTER OF THE VARIABLE NAME
3654   F9B1 10 2B B7 9C            LBMI  TMERROR               TYPE MISMATCH ERROR IF STRING VARIABLE
3655   F9B5 BD BC 35               JSR   LBC35                 PACK FPA0 AND STORE IT IN THE DESCRIPTOR POINTED TO BY X
3656   F9B8 39                     RTS
3657
3658                       * ATTR
3659   F9B9 BD B7 0B       ATTR    JSR   EVALEXPB              EVALUATE EXPRESSION, RETURN VALUE IN ACCB (CHARACTER COLOR)
3660   F9BC 10 21 06 40            LBRN  RAMLINK               RAM HOOK
3661   F9C0 C1 08                  CMPB  #$08                  8 CHARACTER COLORS MAXIMUM
3662   F9C2 10 24 BA 84            LBCC  ILLFUNC               ILLEGAL FUNCTION CALL ERROR IF CHARACTER COLOR > 8
3663   F9C6 58                     ALSB
3664   F9C7 58                     ALSB
3665   F9C8 58                     ALSB                        SHIFT THE CHARACTER COLOR INTO BITS 3-6
3666   F9C9 34 04                  PSHS  B                     SAVE THE SHIFTED COLOR ON THE STACK
3667   F9CB 9D A5                  JSR   GETCCH                GET THE CURRENT INPUT CHARACTER
3668   F9CD BD B2 6D               JSR   SYNCOMMA              SYNTAX CHECK FOR A COMMA
3669   F9D0 BD B7 0B               JSR   EVALEXPB              EVALUATE EXPRESSION, RETURN VALUE IN ACCB (BACKGROUND COLOR)
3670   F9D3 C1 08                  CMPB  #$08                  8 MAXIMUM BACKGROUND COLORS
3671   F9D5 10 24 BA 71            LBCC  ILLFUNC               ILLEGAL FUNCTION CALL ERROR IF > 8
3672   F9D9 EA E4                  ORB   ,S                    'OR' IN THE CHARACTER COLOR
3673   F9DB 32 61                  LEAS  $01,S                 REMOVE TEMPORARY CHARACTER FROM STACK; ORB ,S+ ABOVE IS MORE EFFICIENT
3674   F9DD C4 3F                  ANDB  #3F                   MASK OFF BITS 6,7; THIS INSTRUCTION IS UNNECESSARY
3675   F9DF 34 04                  PSHS  B                     SAVE THE CHARACTER AND BACKGROUND COLORS ON THE STACK
3676   F9E1 9D A5                  JSR   GETCCH                GET THE CURRENT INPUT CHARACTER
3677   F9E3 27 21          LF9E3   BEQ   LFA06                 BRANCH IF END OF LINE
3678   F9E5 BD B2 6D               JSR   SYNCOMMA              SYNTAX CHECK FOR A COMMA
3679   F9E8 81 42                  CMPA  #'B'                  CHECK FOR THE BLINK ATTRIBUTE FLAG
3680   F9EA 26 0A                  BNE   LF9F6                 BRANCH IF NOR BLINK ATTRIBUTE FLAG
3681   F9EC 35 04                  PULS  B
3682   F9EE CA 80                  ORB   #$80                  SET BIT 7 WHICH IS THE BLINK ATTRIBUTE BIT
3683   F9F0 34 04                  PSHS  B
3684   F9F2 9D 9F                  JSR   GETNCH                GET A CHARACTER FROM BASIC'S INPUT LINE
3685   F9F4 20 ED                  BRA   LF9E3                 KEEP CHECKING FOR ATTRIBUTE FLAGS
3686   F9F6 81 55          LF9F6   CMPA  #'U'                  CHECK FOR THE UNDERLINE ATTRIBUTE
3687   F9F8 10 26 BA 4E            LBNE  ILLFUNC               ILLEGAL FUNCION CALL ERROR
3688   F9FC 35 04                  PULS  B
3689   F9FE CA 40                  ORB   #$40                  SET BIT 6 WHICH IS THE UNDERLINE ATTRIBUTE BIT
3690   FA00 34 04                  PSHS  B
3691   FA02 9D 9F                  JSR   GETNCH                GET A CHARACTER FROM BASIC'S INPUT LINE
3692   FA04 20 DD                  BRA   LF9E3                 KEEP CHECKING FOR ATTRIBUTE FLAGS
3693   FA06 35 04          LFA06   PULS  B                     GET THE NEW ATTRIBUTE BYTE FROM THE STACK
3694   FA08 F7 FE 08               STB   H.CRSATT              AND SAVE IT AS THE CURSOR ATTRIBUTES
3695   FA0B 39                     RTS
3696
3697
3698   FA0C                LFA0C   RMB   1012                  UNUSED BYTES
3699   FE00                LFE00   RMB   256                   $FE00 SECONDARY VECTORS AREA
3700   FF00                LFF00   RMB   256                   $FF00 INPUT/OUTPUT AREA
```

ALINK12	E288	EXECCART	A05E	IM.MMU	E0E1	LB657	B657	LC271	C271
ALINK14	E389	FP0EXP	004F	IM.PALET	E678	LB70E	B70E	LC275	C275
ALINK15	E429	FPA0	0050	IM.RGB	E664	LB734	B734	LC277	C277
ALINK16	E413	FRETOP	0021	IM.TEXT	E032	LB73D	B73D	LC278	C278
ALINK17	E532	FUNDIC20	E264	INIT0	FF90	LB740	B740	LC28A	C28A
ALINK18	E3B4	FUNDIS20	E27E	INIT1	FF91	LB89D	B89D	LC28C	C28C
ALINK19	E4D0	G1BITPIX	E7FF	INT.FLAG	FEED	LB8D7	B8D7	LC28D	C28D
ALINK2	E138	G2BITBIX	E820	INTCNV	B3ED	LB905	B905	LC294	C294
ALINK20	E470	G4BITPIX	E83F	INTIMAGE	C359	LB958	B958	LC296	C296
ALINK21	E502	GETBLOK0	E00A	KEYIN	A1CB	LB95C	B95C	LC297	C297
ALINK22	F7AE	GETCCH	00A5	L4000	4000	LB95F	B95F	LC299	C299
ALINK23	F6AD	GETNCH	009F	L80B8	80B8	LB9D7	B9D7	LC29B	C29B
ALINK24	F77E	GETTASK0	E00C	L80E7	80E7	LBA92	BA92	LC29C	C29C
ALINK25	F8C3	GETTASK1	E00E	L8800	8800	LBC35	BC35	LC29E	C29E
ALINK26	F8A3	GETTEXT	E008	L883F	883F	LBCC8	BCC8	LC2A0	C2A0
ALINK27	F757	GIVABF	B4F4	LA0CE	A0CE	LBDD9	BDD9	LC2A1	C2A1
ALINK28	E297	H.BCOLOR	FE0B	LA35F	A35F	LC000	C000	LC2A3	C2A3
ALINK29	E29D	H.COLUMN	FE04	LA390	A390	LC00D	C00D	LC2A5	C2A5
ALINK3	E172	H.CRSATT	FE08	LA3C6	A3C6	LC01B	C01B	LC2A6	C2A6
ALINK4	E192	H.CRSLOC	FE00	LA554	A554	LC02F	C02F	LC2A8	C2A8
ALINK5	E1A6	H.CURSX	FE02	LA7D3	A7D3	LC056	C056	LC2AA	C2AA
ALINK6A	E3F8	H.CURSY	FE03	LA928	A928	LC091	C091	LC2AB	C2AB
ALINK6B	E40C	H.DISPEN	FE06	LAC33	AC33	LC0B1	C0B1	LC2AE	C2AE
ALLCOL	00B5	H.ERLINE	FE13	LAC44	AC44	LC0C2	C0C2	LC2B0	C2B0
ANGLE	00E8	H.ERRBRK	FE17	LAC46	AC46	LC0C9	C0C9	LC2B1	C2B1
ATTR	F9B9	H.ERROR	FE10	LAC49	AC49	LC0DC	C0DC	LC2B2	C2B2
AUTHORMS	F71B	H.FCOLOR	FE0A	LAC65	AC65	LC0F1	C0F1	LC2B4	C2B4
AUTHPIC	C405	H.ONBRK	FE0C	LAC76	AC76	LC0F6	C0F6	LC2B5	C2B5
BAS20ERR	E4CC	H.ONBRKS	FE15	LACA0	ACA0	LC106	C106	LC2B8	C2B8
BASIC	A000	H.ONERR	FE0E	LAD05	AD05	LC10C	C10C	LC2BA	C2BA
BEGMOVE	C03F	H.ONERRS	FE11	LAD19	AD19	LC137	C137	LC2BB	C2BB
BINVAL	002B	H.PBUF	FE19	LAD43	AD43	LC165	C165	LC2BF	C2BF
BLKCNT	0094	H.PCOUNT	FE18	LADC4	ADC4	LC175	C175	LC2C1	C2C1
BRK	E3E6	HBUFF	ED58	LADD4	ADD4	LC180	C180	LC2C2	C2C2
BUTTON	E5B1	HCALPOS	E7DA	LADF4	ADF4	LC185	C185	LC2C6	C2C6
CALTABLE	E7DE	HCIRCLE	EA49	LAE09	AE09	LC18C	C18C	LC2C8	C2C8
CHARAD	00A6	HCLS	E6CF	LAE11	AE11	LC19A	C19A	LC2C9	C2C9
CHGFLG	00DB	HCOLOR	E6F4	LAEBB	AEBB	LC1AA	C1AA	LC2CB	C2CB
CIRCDATA	EB99	HDRAW	F39D	LAEEB	AEEB	LC1B1	C1B1	LC2CD	C2CD
CLRHIRES	E6D8	HGET	EDE5	LAF45	AF45	LC1DE	C1DE	LC2CE	C2CE
CMP	E676	HLINE	E882	LAF67	AF67	LC1E7	C1E7	LC2D1	C2D1
COL32	F652	HORBEG	00BD	LB141	B141	LC1F0	C1F0	LC2D3	C2D3
COL40	F65C	HORBYT	00B9	LB156	B156	LC20A	C20A	LC2D4	C2D4
COL80	F679	HORDEF	00C7	LB262	B262	LC22A	C22A	LC2D8	C2D8
COMDIC20	E1C5	HOREND	00C3	LB267	B267	LC24E	C24E	LC2DA	C2DA
COMDIS20	E236	HPAINT	EBF5	LB26A	B26A	LC257	C257	LC2DB	C2DB
CURLIN	0068	HPOINT	E85C	LB26F	B26F	LC259	C259	LC2DE	C2DE
DCNVEC	C004	HPRINT	EF3F	LB277	B277	LC25A	C25A	LC2E0	C2E0
DEVNUM	006F	HPUT	EDED	LB284	B284	LC25B	C25B	LC2E1	C2E1
DISK20MS	E2A3	HRESBUFF	C000	LB2CE	B2CE	LC25D	C25D	LC2E4	C2E4
DISK21MS	E316	HRESET	E765	LB357	B357	LC25E	C25E	LC2E6	C2E6
DOSBAS	C000	HRMODE	00E6	LB3A2	B3A2	LC261	C261	LC2E7	C2E7
EBCOMTAB	E162	HRWIDTH	00E7	LB3E9	B3E9	LC263	C263	LC2EA	C2EA
ENDMOVE	C36C	HSCREEN	E688	LB4F3	B4F3	LC264	C264	LC2EC	C2EC
ERLIN	E4FD	HSET	E761	LB511	B511	LC267	C267	LC2ED	C2ED
ERNO	E4E9	HSTAT	F925	LB516	B516	LC269	C269	LC2F0	C2F0
ERR	E3D4	ILLFUNC	B44A	LB54C	B54C	LC26A	C26A	LC2F2	C2F2
EVALEXPB	B70B	IM.CMP	E654	LB56D	B56D	LC26E	C26E	LC2F3	C2F3
EXBAS	8000	IM.GRAPH	E070	LB654	B654	LC270	C270	LC2FB	C2FB

LC2FD	C2FD	LE1DD	E1DD	LE36A	E36A	LE73B	E73B	LEA29	EA29
LC2FE	C2FE	LE1E1	E1E1	LE36B	E36B	LE759	E759	LEA2B	EA2B
LC301	C301	LE1E7	E1E7	LE386	E386	LE75D	E75D	LEA2D	EA2D
LC303	C303	LE1ED	E1ED	LE3C2	E3C2	LE76A	E76A	LEA33	EA33
LC304	C304	LE1F4	E1F4	LE3CF	E3CF	LE77F	E77F	LEA34	EA34
LC307	C307	LE1F9	E1F9	LE424	E424	LE782	E782	LEA3D	EA3D
LC309	C309	LE1FD	E1FD	LE426	E426	LE788	E788	LEA44	EA44
LC30A	C30A	LE201	E201	LE430	E430	LE792	E792	LEA45	EA45
LC30D	C30D	LE206	E206	LE43F	E43F	LE7AA	E7AA	LEA59	EA59
LC322	C322	LE20C	E20C	LE446	E446	LE7AD	E7AD	LEA95	EA95
LC334	C334	LE20F	E20F	LE449	E449	LE7B0	E7B0	LEA9F	EA9F
LC33B	C33B	LE212	E212	LE458	E458	LE7B2	E7B2	LEAB3	EAB3
LC349	C349	LE218	E218	LE466	E466	LE7B9	E7B9	LEAC7	EAC7
LC351	C351	LE21D	E21D	LE47D	E47D	LE7BF	E7BF	LEAE7	EAE7
LC352	C352	LE221	E221	LE488	E488	LE7CD	E7CD	LEB13	EB13
LC355	C355	LE227	E227	LE496	E496	LE7D0	E7D0	LEB1E	EB1E
LC356	C356	LE22C	E22C	LE49F	E49F	LE7D7	E7D7	LEB20	EB20
LC35A	C35A	LE22F	E22F	LE4B0	E4B0	LE7E6	E7E6	LEB32	EB32
LD67F	D67F	LE232	E232	LE4B3	E4B3	LE875	E875	LEB3D	EB3D
LDC05	DC05	LE238	E238	LE4C7	E4C7	LE87B	E87B	LEB3F	EB3F
LE010	E010	LE23A	E23A	LE4CE	E4CE	LE899	E899	LEB48	EB48
LE033	E033	LE23C	E23C	LE4F4	E4F4	LE8B3	E8B3	LEB53	EB53
LE03B	E03B	LE23E	E23E	LE4F9	E4F9	LE8B4	E8B4	LEB5C	EB5C
LE03C	E03C	LE240	E240	LE4FA	E4FA	LE8EB	E8EB	LEB5F	EB5F
LE044	E044	LE242	E242	LE51E	E51E	LE8F2	E8F2	LEB60	EB60
LE045	E045	LE244	E244	LE528	E528	LE8F4	E8F4	LEB73	EB73
LE063	E063	LE246	E246	LE52B	E52B	LE8F6	E8F6	LEB7B	EB7B
LE06D	E06D	LE248	E248	LE58E	E58E	LE905	E905	LEB91	EB91
LE06E	E06E	LE24A	E24A	LE59A	E59A	LE906	E906	LEB9B	EB9B
LE06F	E06F	LE24C	E24C	LE5AF	E5AF	LE913	E913	LEB9D	EB9D
LE071	E071	LE24E	E24E	LE5D5	E5D5	LE921	E921	LEBA1	EBA1
LE079	E079	LE250	E250	LE5D9	E5D9	LE92F	E92F	LEBA5	EBA5
LE07A	E07A	LE252	E252	LE5E3	E5E3	LE931	E931	LEBA9	EBA9
LE08B	E08B	LE254	E254	LE5EA	E5EA	LE93E	E93E	LEBAB	EBAD
LE0CB	E0CB	LE256	E256	LE5EC	E5EC	LE94E	E94E	LEBAF	EBB1
LE0E9	E0E9	LE258	E258	LE5FA	E5FA	LE95D	E95D	LEBB5	EBB5
LE0F1	E0F1	LE25A	E25A	LE600	E600	LE96C	E96C	LEBB9	EBB9
LE0F7	E0F7	LE25C	E25C	LE606	E606	LE977	E977	LEBBD	EBBD
LE148	E148	LE25E	E25E	LE60C	E60C	LE988	E988	LEBCA	EBCA
LE14D	E14D	LE260	E260	LE62A	E62A	LE98D	E98D	LEBCB	EBCB
LE152	E152	LE262	E262	LE634	E634	LE9AD	E9AD	LEBEA	EBEA
LE158	E158	LE269	E269	LE648	E648	LE9B1	E9B1	LEC05	EC05
LE15B	E15B	LE26F	E26F	LE64A	E64A	LE9B8	E9B8	LEC1D	EC1D
LE15D	E15D	LE275	E275	LE693	E693	LE9B9	E9B9	LEC47	EC47
LE163	E163	LE279	E279	LE69C	E69C	LE9BF	E9BF	LEC4A	EC4A
LE165	E165	LE280	E280	LE6A5	E6A5	LE9C6	E9C6	LEC51	EC51
LE167	E167	LE282	E282	LE6B9	E6B9	LE9CC	E9CC	LEC6A	EC6A
LE168	E168	LE284	E284	LE6BC	E6BC	LE9CD	E9CD	LEC6E	EC6E
LE16A	E16A	LE286	E286	LE6CB	E6CB	LE9D1	E9D1	LEC80	EC80
LE16C	E16C	LE2C0	E2C0	LE6D6	E6D6	LE9DB	E9DB	LEC83	EC83
LE18B	E18B	LE2C1	E2C1	LE6E4	E6E4	LE9E1	E9E1	LEC86	EC86
LE19A	E19A	LE2DA	E2DA	LE6EF	E6EF	LE9F0	E9F0	LEC88	EC88
LE19E	E19E	LE2DB	E2DB	LE705	E705	LEA04	EA04	LEC9B	EC9B
LE1AE	E1AE	LE2F7	E2F7	LE70D	E70D	LEA0A	EA0A	LECA5	ECA5
LE1B2	E1B2	LE313	E313	LE70E	E70E	LEA0D	EA0D	LECB7	ECB7
LE1BF	E1BF	LE333	E333	LE711	E711	LEA16	EA16	LECBA	ECBA
LE1CA	E1CA	LE334	E334	LE718	E718	LEA21	EA21	LECBE	ECBE
LE1D1	E1D1	LE33D	E34D	LE72F	E72F	LEA25	EA25	LECC7	ECC7
LE1D8	E1D8	LE33E	E34E	LE731	E731	LEA27	EA27	LECCD	ECCD

LECD1	ECD1	LEF7F	EF7F	LF1DD	F1DD	LF3C5	F3C5	LF6DC	F6DC
LECE8	ECE8	LEF8E	EF8E	LF1E5	F1E5	LF3CF	F3CF	LF6E0	F6E0
LECEA	ECEA	LEF96	EF96	LF1ED	F1ED	LF3E6	F3E6	LF6E7	F6E7
LECF1	ECF1	LEFA3	EFA3	LF1F5	F1F5	LF3EE	F3EE	LF719	F719
LED01	ED01	LEFAD	EFAD	LF1FD	F1FD	LF417	F417	LF730	F730
LED15	ED15	LEFD9	EFD9	LF205	F205	LF445	F445	LF74D	F74D
LED20	ED20	LEFDB	EFDB	LF20D	F20D	LF451	F451	LF75E	F75E
LED2E	ED2E	LEFFE	EFFE	LF215	F215	LF45C	F45C	LF761	F761
LED3A	ED3A	LF001	F001	LF21D	F21D	LF467	F467	LF766	F766
LED3F	ED3F	LF002	F002	LF225	F225	LF46D	F46D	LF772	F772
LED4E	ED4E	LF004	F004	LF22D	F22D	LF46F	F46F	LF778	F778
LED72	ED72	LF006	F006	LF235	F235	LF473	F473	LF787	F787
LED85	ED85	LF008	F008	LF23D	F23D	LF47D	F47D	LF79F	F79F
LED95	ED95	LF00A	F00A	LF245	F245	LF485	F485	LF7A4	F7A4
LED97	ED97	LF01A	F01A	LF24D	F24D	LF487	F487	LF7A8	F7A8
LEDA3	EDA3	LF035	F035	LF255	F255	LF48C	F48C	LF7C4	F7C4
LEDB0	EDB0	LF045	F045	LF25D	F25D	LF492	F492	LF7CC	F7CC
LEDBD	EDBD	LF06C	F06C	LF265	F265	LF496	F496	LF7D7	F7D7
LEDC4	EDC4	LF08C	F08C	LF26D	F26D	LF49A	F49A	LF7DE	F7DE
LEDD2	EDD2	LF09D	F09D	LF275	F275	LF4A1	F4A1	LF7E2	F7E2
LEDD6	EDD6	LF0A5	F0A5	LF27D	F27D	LF4B2	F4B2	LF802	F802
LEDD8	EDD8	LF0AD	F0AD	LF285	F285	LF4BF	F4BF	LF807	F807
LEDF4	EDF4	LF0B5	F0B5	LF28D	F28D	LF4D0	F4D0	LF827	F827
LEE06	EE06	LF0BD	F0BD	LF295	F295	LF4D1	F4D1	LF82E	F82E
LEE23	EE23	LF0C5	F0C5	LF29D	F29D	LF4D4	F4D4	LF854	F854
LEE28	EE28	LF0CD	F0CD	LF2A5	F2A5	LF4E5	F4E5	LF860	F860
LEE34	EE34	LF0D5	F0D5	LF2AD	F2AD	LF4F1	F4F1	LF86A	F86A
LEE38	EE38	LF0DD	F0DD	LF2B5	F2B5	LF4FC	F4FC	LF86E	F86E
LEE50	EE50	LF0E5	F0E5	LF2BD	F2BD	LF507	F507	LF87B	F87B
LEE5D	EE5D	LF0ED	F0ED	LF2C5	F2C5	LF510	F510	LF88A	F88A
LEE6D	EE6D	LF0F5	F0F5	LF2CD	F2CD	LF51F	F51F	LF8AB	F8AB
LEE92	EE92	LF0FD	F0FD	LF2D5	F2D5	LF527	F527	LF8B1	F8B1
LEE96	EE96	LF105	F105	LF2DD	F2DD	LF53B	F53B	LF8CD	F8CD
LEEA7	EEA7	LF10D	F10D	LF2E5	F2E5	LF545	F545	LF8EB	F8EB
LEEAB	EEAB	LF115	F115	LF2ED	F2ED	LF54C	F54C	LF8ED	F8ED
LEEC0	EEC0	LF11D	F11D	LF2F5	F2F5	LF570	F570	LF963	F963
LEEC7	EEC7	LF125	F125	LF2FD	F2FD	LF575	F575	LF9E3	F9E3
LEED3	EED3	LF12D	F12D	LF305	F305	LF578	F578	LF9F6	F9F6
LEEE0	EEE0	LF135	F135	LF30D	F30D	LF583	F583	LFA06	FA06
LEEE2	EEE2	LF13D	F13D	LF315	F315	LF584	F584	LFA0C	FA0C
LEEE3	EEE3	LF145	F145	LF31D	F31D	LF590	F590	LFE00	FE00
LEEE5	EEE5	LF14D	F14D	LF325	F325	LF591	F591	LFF00	FF00
LEEE6	EEE6	LF155	F155	LF32D	F32D	LF593	F593	LOCATE	F8D2
LEEE8	EEE8	LF15D	F15D	LF335	F335	LF5A7	F5A7	LPEEK	E573
LEEE9	EEE9	LF165	F165	LF33D	F33D	LF5B6	F5B6	LPOKE	E545
LEEEB	EEEB	LF16D	F16D	LF345	F345	LF5C1	F5C1	MICROMS	F702
LEEEC	EEEC	LF175	F175	LF34D	F34D	LF5DA	F5DA	MMUIMAGE	C246
LEEEE	EEEE	LF17D	F17D	LF355	F355	LF5DD	F5DD	MMUREG	FFA0
LEEEF	EEEF	LF185	F185	LF35D	F35D	LF5F2	F5F2	MOVE.XY	C1D6
LEEF6	EEF6	LF18D	F18D	LF365	F365	LF5FA	F5FA	MWAREMS	E2F8
LEEFE	EEFE	LF195	F195	LF36D	F36D	LF5FD	F5FD	NEGACCD	F4CC
LEF07	EF07	LF19D	F19D	LF375	F375	LF608	F608	NOWARM	C0DE
LEF10	EF10	LF1A5	F1A5	LF37D	F37D	LF610	F610	OLDPTR	002D
LEF18	EF18	LF1AD	F1AD	LF385	F385	LF611	F611	PALETTE	E5F0
LEF25	EF25	LF1B5	F1B5	LF38D	F38D	LF61F	F61F	PALETREG	FFB0
LEF2C	EF2C	LF1BD	F1BD	LF395	F395	LF64F	F64F	PALIMAGE	C236
LEF62	EF62	LF1C5	F1C5	LF3B8	F3B8	LF66D	F66D	PATCH28	C0C6
LEF6C	EF6C	LF1CD	F1CD	LF3BD	F3BD	LF68C	F68C	PATCH29	C0D9
LEF75	EF75	LF1D5	F1D5	LF3C3	F3C3	LF69B	F69B	PATCH30	C8B4

PATCHTAB	C256	VEREND	ØØC5
PIAØ	FFØØ	VIDEOREG	FF98
PIA1	FF2Ø	VIDIMAGE	C22E
PIX1MASK	E7F1	VIDRAM	Ø4ØØ
PIX2MASK	E7F9	WAITLOOP	C22C
PIX4MASK	E7FD	WCOLOR	ØØB4
PIXELFIL	E742	WIDTH	F636
PRGGRAPH	EØØ4	ZERO	ØØ8A
PRGMMU	EØØ6		
PRGTEXT	EØØ2		
RAMLINK	ØØØØ		
RESTABLE	EØ6C		
RESVEC	AØ27		
RGB	E674		
RSTFLG	Ø071		
RSTVEC	Ø072		
SAM	FFCØ		
SCALE	ØØE9		
SELBLOKØ	EØA1		
SELTASKØ	EØFF		
SELTASK1	E119		
SELTEXT	EØB5		
SETFLG	ØØC2		
SETGRAPH	EØ4D		
SETMMU	EØ97		
SETTEXT	EØ19		
SETVIDEO	EØ82		
SPAREØ	EØ13		
SPARE1	EØ15		
SPARE2	EØ17		
STRINOUT	B99C		
SUPERVAR	EØØØ		
SYNCOMMA	B26D		
TEMPPT	ØØØB		
TMERROR	B151		
TMPSTACK	DFFF		
TMPSTK	ØØDC		
TXTTAB	Ø019		
V.BORDER	FF9A		
V.TIMER	FF94		
VØ	ØØ4Ø		
V41	Ø041		
V42	Ø042		
V44	Ø044		
VALTYP	Ø006		
VARDES	Ø03B		
VCB	ØØCB		
VCD	ØØCD		
VCF	ØØCF		
VD1	ØØD1		
VD3	ØØD3		
VD4	ØØD4		
VD5	ØØD5		
VD6	ØØD6		
VD7	ØØD7		
VD8	ØØD8		
VD9	ØØD9		
VERBEG	ØØBF		
VERDEF	ØØC9		

The major functions of the GIME chip are controlled by the chip control register which are mapped into the I/O page ($FF00-$FFFF) which is always present in the logical address space regardless of the status of the MMU registers. The area from $FF90-$FFBF in particular is a direct link to the GIME chip.

FF90 Initialization Register 0 (INIT0)

BIT7	COCO	1=Color Computer Compatible
BIT6	MMUEN	1=MMU Enabled (COCO = 0)
BIT5	IEN	1=Chip IRQ output enabled
BIT4	FEN	1=Chip FIRQ output enabled
BIT3	MC3	1=RAM at XFEFF is constant
BIT2	MC2	1=$FF40-4F external; 0=internal
BIT1	MC1	ROM map control (see table below)
BIT0	MC0	ROM map control (see table below)

MC1	MC0	ROM mapping
0	X	16K Internal, 16K external
1	0	32K Internal
1	1	32K External (except vectors)

FF91 Initialization Register 1 (INIT1)

BIT6-7		Unused
BIT5	TINS	Timer clock: 1 = 70nsec, 0 = 63.5 usec
BIT1-4		Unused
BIT0	TR	MMU task register select

FF92 IRQ Interrupt Enable/Status Register (IRQENR)

BIT6-7		Unused
BIT5	TMR	Timer
BIT4	HBORD	Horizontal Border
BIT3	VBORD	Vertical Border
BIT2	EI2	RS-232 serial port
BIT1	EI1	Keyboard
BIT0	EI0	Cartridge Port

FF93 FIRQ Interrupt Enable/Status Register (FIRQENR)

BIT6-7		Unused
BIT5	TMR	Timer
BIT4	HBORD	Horizontal Border
BIT3	VBORD	Vertical Border
BIT2	EI2	RS-232 serial port
BIT1	EI1	Keyboard
BIT0	EI0	Cartridge Port

FF94 Timer Register (MSB)

BIT4-7		Unused
BIT0-3		Most Significant 4 bits of timer

FF95 Timer Register (LSB)

BIT0-7		Least Significant 8 bits of timer

FF96,7 Reserved

FF98 Video Mode Register

BIT7	BP	0=text, 1=bit plane graphics
BIT6		Unused
BIT5	BPI	Burst Phase Invert (Color Set)
BIT4	MOCH	1=Monochrome (Composite Monitor)
BIT3	H50	1=50Hz vertical sync
BIT0-2	LPR	Lines per Row

FF99 Video Resolution Register

BIT7		Undefined
BIT5-6	LPF	Lines per Field (number of rows)
BIT2-4	HRES	Horizontal Resolution
BIT0-1	CRES	Color Resolution

FF9A Border Register

BIT6-7		Unused
BIT0-5		Border Color

FF9B Unused

FF9C Vertical Scroll Register

BIT4-7		Unused
BIT0-3	VSC	Vertical Scroll Bits

FF9D Vertical Offset Register (MSB)

BIT0-7	Y8-Y15	Vertical offset high order byte

FF9E Vertical Offset Register (LSB)

BITØ-7	Y0-Y7	Vertical offset low order byte

FF9F Horizontal Offset Register

BIT7	HVEN	Horizontal Virtual Enable
BITØ-6	HOFF	Horizontal Offset

FFA0-FFA7 Memory Management Unit Task Register Ø

FFA8-FFAF Memory Management Unit Task Register 1

FFB0-FFBF Palette Registers

Listed below are the 64 different colors available on the Color Computer 3. In order to use the colors, first decide which color you wish to display and find the color closest to your desired color in the table below. Then you must know whether the color will be viewed on an RGB monitor (RGB) or a composite monitor or a television set (CMP). Get the color number from the appropriate monitor column and store that number into a palette register or the border color register.

The names assigned to these colors are based upon the names given to the RGB colors. Since the methods in which the RGB and composite colors are generated are not the same (as explained in Chapter Five) the name of the color may not agree whith what you personally see the color to be. This table is presented in order to provide a universal conversion between the RGB and composite colors. All you need to do is have a short program such as the one shown in Figure 16 to allow easy conversion of colors in your program based upon the type of monitor being used to view the program. The table of colors given below is the conversion used in OS-9 Level Two.

Monitor		Color	Monitor		Color
RGB	CMP		RGB	CMP	
ØØ	ØØ	Black	32	23	Medium intensity red
Ø1	12	Low intensity blue	33	8	Blue tint red
Ø2	Ø2	Low intensity green	34	21	Light Orange
Ø3	14	Low intensity cyan	35	6	Cyan tint red
Ø4	Ø7	Low intensity red	36	39	Full intensity red
Ø5	Ø9	Low intensity magenta	37	24	Magenta tint red
Ø6	Ø5	Low intensity brown	38	38	Brown tint red
Ø7	16	Low intensity white	39	54	Faded red
Ø8	28	Medium intensity blue	4Ø	25	Medium intensity magenta
Ø9	44	Full intensity blue	41	42	Blue tint magenta
1Ø	13	Green tint blue	42	26	Green tint magenta
11	29	Cyan tint blue	43	58	Cyan tint magenta
12	11	Red tint blue	44	24	Red tint magenta
13	27	Magenta tint blue	45	41	Full intensity magenta
14	1Ø	Brown tint blue	46	4Ø	Brown tint magenta
15	43	Faded blue	47	56	Faded magenta
16	34	Medium intensity green	48	2Ø	Medium intensity yellow
17	17	Blue tint green	49	4	Blue tint yellow
18	18	Full intensity green	5Ø	35	Green tint yellow
19	33	Cyan tint green	51	51	Cyan tint yellow
2Ø	Ø3	Red tint green	52	37	Red tint yellow
21	Ø1	Magenta tint green	53	53	Magenta tint yellow
22	19	Brown tint green	54	36	Full intensity yellow
23	5Ø	Faded green	55	52	Faded yellow
24	3Ø	Medium intensity cyan	56	32	Medium intensity white
25	45	Blue tint cyan	57	59	Light blue
26	31	Green tint cyan	58	49	Light green
27	46	Full intensity cyan	59	62	Light cyan
28	15	Red tint cyan	6Ø	55	Light red
29	6Ø	Magenta tint cyan	61	57	Light magenta
3Ø	47	Brown tint cyan	62	63	Light yellow
31	61	Faded cyan	63	48	White

Converting RGB Colors to Composite Colors

It will often be beneficial to allow a graphic display to appear the same on a composite monitor as it does on an RGB monitor. The following Basic routine, which will convert an RGB color code into its closest similar composite color code, can be used. Since RGB colors are derived in a different manner than composite colors, no conversion will be exact.

```
10  'SET UP CONVERSION FACTORS FOR COMPOSITE COLORS
20  DIM C(63) : FOR X = 0 TO 63 : READ C(X) : NEXT X
30  GOTO 100
40  DATA 0,12,2,14,7,9,5,16,28,44,13,29,11,27,10,43
50  DATA 34,17,18,33,3,1,19,50,30,45,31,46,15,60,47,61
60  DATA 23,8,21,6,39,24,38,54,25,42,26,58,24,41,40,56
70  DATA 20,4,35,51,37,53,36,52,32,59,49,62,55,57,63,48
80  'CONVERT C TO COMPOSITE COLOR IF NOT RGB MON. (R=0)
90  IF R=0 THEN C=C(C) : RETURN ELSE RETURN
100 'MAIN BODY OF PROGRAM
```

FIGURE 16 - BASIC PROGRAM TO CONVERT RGB COLORS TO COMPOSITE

The above routine should be near the beginning of the program (but after the CLEAR statement). The program must ask for the type of monitor being used. If an RGB monitor is in use, set R=1, otherwise set R=0. Then, when it is time to set a color, make C equal to the RGB color desired and execute a GOSUB 90. The color will be converted to the appropriate composite color. Then simply set the palette to the value in C. For example (assuming R has already been set up), a line might read

```
270 C=27 : GOSUB 90 : PALETTE 3,C
```

Listed below are all of the data and ASCII tables found in the last half (C000-FDFF) of the Super Extended Basic ROM.

START	END	DESCRIPTION
C22E	C235	VIDEO CONTROL REGISTERS' IMAGE (INITIALIZATION)
C236	C245	PALETTE REGISTERS' IMAGE (INITIALIZATION)
C246	C255	MMU REGISTERS' IMAGE (INITIALIZATION)
C256	C30C	COLOR/EXTENDED BASIC PATCH TABLE
C30D	C321	CODED AUTHORS' NAMES
C351	C358	DISK BASIC PATCH TABLE
C359	C36B	INTERRUPT JUMP TABLE IMAGE
C405	DC04	DIGITIZED PICTURE OF THE AUTHORS
E000	E018	ROM ROUTINES' ADDRESS VECTORS
E032	E04C	VIDEO REGISTERS' TEXT MODE IMAGES
E06C	E06F	VIDEO RESOLUTION REGISTER ($FF99) VALUES
E070	E081	VIDEO REGISTERS' GRAPHICS MODE IMAGES
E0E1	E0F0	MMU IMAGES
E162	E16B	COMMAND INTERPRETATION TABLE
E1C5	E235	COMMANDS DICTIONARY
E236	E263	COMMANDS DISPATCH TABLE
E264	E27D	FUNCTIONS DICTIONARY
E27E	E287	FUNCTIONS DISPATCH TABLE
E2A3	E2F7	DISK BASIC 2.0 COPYRIGHT MESSAGE
E2F8	E315	'AND MICROWARE SYSTEMS CORP.' MESSAGE
E316	E388	DISK BASIC 2.1 COPYRIGHT MESSAGE
E4CC	E4CF	SUPER EXTENDED BASIC ERROR CODES
E654	E663	TABLE OF 'OFFICIAL' COMPOSITE COLORS
E664	E673	TABLE OF 'OFFICIAL' RGB COLORS
E678	E687	PALETTE REGISTERS' RAM IMAGE
E6CB	E6CE	TABLE OF HOW MANY BYTES PER HORIZONTAL ROW IN THE HIGH RESOLUTION GRAPHICS MODES
E759	E75C	TABLE OF SINGLE PIXEL MASKS FOR THE HI-RES GRAPHICS MODES
E75D	E760	TABLE OF MULTIPLIERS TO SPREAD A SINGLE PIXEL MASK THROUGH AN ENTIRE BYTE
E7F1	E7FE	TABLE OF 1,2 AND 4 BIT SHIFTED PIXEL MASKS
EA25	EA2C	TABLE OF PIXEL MOVE ADDRESSES
EB99	EBBC	TABLE OF SINES AND COSINES FOR THE HCIRCLE COMMAND
EEE0	EEEE	LOOKUP TABLE FOR PSET, PRESET, AND, OR AND NOT ROUTINES FOR THE HPUT COMMAND
F002	F00B	TABLE OF ADDRESSES FOR THE HI-RES PRINT DRIVERS
F035	F044	TABLE OF ALL POSSIBLE 2 BIT PIXEL MASKS
F06C	F08B	TABLE OF ALL POSSIBLE DOUBLE BYTE 4 BIT PIXEL MASKS
F09D	F39C	HIGH RESOLUTION SOFTWARE CHARACTER GERNERATOR 'ROM'
F702	F71A	'MICROWARE SYSTEMS CORP.' MESSAGE
F71B	F72F	AUTHORS' NAMES - THIS AREA IS ALL ZEROS IN THE ROM. AFTER INITIALIZATION THE AUTHORS' NAMES WILL BE FOUND HERE (IN RAM) UNTIL YOU EXECUTE A CLS 100 COMMAND, AFTER WHICH TIME YOU WILL FIND NOPs IN THIS AREA.

At the back of the Color Computer 3 Extended Basic manual, you will find a section called ROM ROUTINES. In this section you will find a summary of the official ROM calls which may be made. These official calls are located at $A000 and are made by indirect subroutine calls to the addresses given. No mention is made of a suspicious looking table of addresses located at address $E000 which is the beginning of the new code added to the Basic ROM by the Color Computer 3. These addresses are similar to a table of addresses at the beginning of the Disk Basic ROM which is only partially documented. It is the opinion of the author that the Super Extended table, as well as the Disk Basic table, will be maintained by Tandy and should be used as if they were supported by Tandy. The reader is cautioned that this is just the OPINION of the author and is by no means the official stance of Tandy.

SUPERVAR = [E000]
This is the address of the direct page variables used by Super Extended Basic. It is not the address of a routine.

ENTRY CONDITIONS
Not applicable

EXIT CONDITIONS
Not applicable

PRGTEXT = [E002]
Program INIT0 and the video control registers with their RAM images according to the value contained in HRWIDTH. Basic (unmodified by the user) will do the following: 1) If HRWIDTH = 0, set up 32 column CoCo compatible mode, 2) If HRWIDTH = 1, set up the 40 column hi-res text mode, 3) If HRWIDTH = anything else, set up the 80 column hi-res text mode. A RAM hook exists in this routine which will allow the user to modify it.

ENTRY CONDITIONS
HRWIDTH should be set to a valid value.

EXIT CONDITIONS
INIT0 and the video control registers are modified. All CPU registers, except CC are preserved.

PRGGRAPH = [E004]
Program INIT0 and the video control registers with their RAM images according to the value contained in HRMODE. Basic (unmodified by the user) will do the following: 1) If HRMODE is 1,2,3 or 4, set the proper HSCREEN graphics mode or 2) if HRMODE is any other value, cause invalid and potentially disastrous data to be programmed into INIT0 and the video control registers. A RAM hook exists in this routine which will allow the user to modify it.

ENTRY CONDITIONS
HRMODE should be set to a valid value (1-4)

EXIT CONDITIONS
INITØ and the video control registers are modified. All CPU registers, except CC are preserved.

PRGMMU = [EØØ6]
Program the MMU registers with their RAM images.

ENTRY CONDITIONS
None

EXIT CONDITIONS
The MMU registers are modifed. All CPU registers, except CC are preserved.

GETTEXT = [EØØ8]
Place block 6.6 into the RAM image of the Task Register Ø MMU register which controls logical block 1. Then copy the RAM image of the MMU registers to the MMU registers. Finally, replace block 6.6 (as saved above) with block 7.1 in the RAM image of the MMU registers. This is a very special purpose routine used by Basic to replace the hi-res text screen into the logical address space so that they may be modified.

ENTRY CONDITIONS
None

EXIT CONDIIIONS
The RAM image of MMU register one of task register Ø and the MMU registers are modified. All CPU registers, except CC, are preserved.

GETBLOKØ = [EØØA]

Place a block into the RAM image of Task Register Ø MMU register which controls logical block Ø. Then copy the RAM image of the MMU registers to the MMU registers. Finally, replace the block (as saved above) with block 7.Ø in the RAM image of the MMU registers. This is a very special purpose routine used by Basic to place any block into the logical address space so that it may be modified.

ENTRY CONDITIONS
B contains the block (Ø-$3F) to be loaded.

EXIT CONDITIONS
The RAM image of the MMU register Ø of task register Ø and the MMU registers are modifed. Akk CPU registers, except CC, are preserved.

GETTASKØ = [EØØC]
Restore task register Ø as the active task register.

ENTRY CONDITIONS

The new address for the stack pointer must be the first two bytes on the stack.

EXIT CONDITIONS
The stack pointer is reset to the first two bytes on the old stack. All other CPU registers, except CC, are preserved. V40-V45 are modified. INIT1 is cleared which may affect the timer input clock. FIRQ and IRQ are masked on at the CPU.

GETTASK1 = [E00E]
Select Task Register 1 as the active task register

ENTRY CONDITIONS
None

EXIT CONDITIONS
The stack pointer has been reset to $DFFF and the old stack pointer has been saved on the new stack. All other CPU registers, except CC, have been saved. V40-V45 have been modified. INIT1 has been forced to 1 which may affect the timer input clock. FIRQ and IRQ have been masked off at the CPU.

GOCART = E010
This address is used to execute a ROM cartridge (on the expansion port) if the cartridge does not autostart.

SPARE0 = [E013]
This address is undefined.

SPARE1 = [E015]
This address is undefined.

SPARE2 = [E017]
This address is undefined.

Listed below are several routines in the Super Extended Basic ROM (which run in RAM). These routines should be used with great care since they usually expect that some of Basic's variables have been initialized to a certain range of values in order to function. If the routines encounter an error, they will exit to Basic's error processing code. The user must be aware of this fact and intercept Basic's error routines if you are to stay in control while using these routines. Some of these routines may also change the MMU registers - BEWARE!

MODIFIED* REGISTERS	ADDRESS	DESCRIPTION
none	E0CB	ENABLE HGET/HPUT BUFFER - Put the HPUT/HGET buffer block (6.4) into the logical address space. Exit with the MMU register images restored to 'normal'.
A,B,X,Y	E0F1	PROGRAM MMU REGISTERS - Program the 16 MMU registers with the 16 bytes pointed to by the X register.
B,X	E58E	GET LONG ADDRESS - Convert FPA0 into a 'long' (512K) address. Return the block number of the address in ACCB and the remaining 13 bits of the address in X.
A,B,X	BUTTON+13 (E5BE)	READ JOYSTICK BUTTON - Read the joystick button specified in ACCB (0-3) and return the status in FPA0.
A,B,X,Y	E5FA	DISPLAY DEFAULT RGB COLORS - Copy Basic's default RGB palette register colors into the palette registers.
A,B,X,Y	E606	DISPLAY DEFAULT CMP COLORS - Copy Basic's default CMP palette register colors into the palette registers.
A,B,X,Y	E634	COPY PALETTE IMAGES - Copy the palette register color RAM images into the palette registers.
A,B,X	CLRHIRES (E6D8)	CLEAR THE HI-RES GRAPHICS SCREEN - Clear the hi-res graphics screen to the palette register number in ACCB.
A,B	PIXELFIL E742	FILL ACCB WITH PIXELS - Fill ACCB with pixels composed of a specific palette register. Enter with ACCB Containing the palette register number used to fill ACCB.
A,B	E792	TURN ON A PIXEL - Turn on the pixel which is being pointed to by the X register (screen address) and bit position specified by ACCA (pixel mask) to the color in ALLCOL. Set CHGFLG <> 0 if pixel was unchanged by the action.
A,B,X,Y	E7B2	EVALUATE HI-RES COORDINATES - Evaluate two expressions in a Basic Line. Perform hi-res coordinate range checks

		on the values returned and store the tested values in the address pointed to by Y.
A,X,U	CALPOS E7DA	CALPOS FOR CURRENT HSCREEN MODE - Jump to the correct CALculate POSition routine depending upon the current HSCREEN mode.
A,X	G1BITPIX (E7FF)	CALPOS 2 COLOR MODE - Calculate the screen address and pixel mask for the 2 color hi-res graphics mode. Enter with X,Y coordinates in HORBEG and VERBEG and exit with the address in the X Register and the pixel mask in ACCA.
A,X	G2BITPIX (E82Ø)	CALPOS 4 COLOR MODE - Calculate the screen address and pixel mask for the 4 color hi-res graphics mode. Enter with X,Y coordinates in HORBEG and VERBEG and exit with the address in the X Register and the pixel mask in ACCA.
A,X	G4BITPIX (E83F)	CALPOS 16 COLOR MODE - Calculate the screen address and pixel mask for the 16 color hi-res graphics mode. Enter with X,Y coordinates in HORBEG and VERBEG and exit with the address in the X Register and the pixel mask in ACCA.
A,B,X,Y,U	E8D3	DRAW A HI-RES BOX - Enclose a diagonal line with a box (box function of HLINE). Enter with the start and end coordinates of the original line in HORBEG,VERBEG, HOREND and VEREND.
A,B,X,Y,U	E8F6	FILL A HI-RES BOX - Draw a series of horizontal lines from VERBEG to VEREND.
A,B,X,Y,U	E9Ø6	DRAW A HORIZONTAL HI-RES LINE - Draw a horizontal hi-res line from HOREND to HORBEG at the vertical coordinate VERBEG with the palette register number in ALLCOL.
A,B,X,Y,U	E931	DRAW A VERTICAL HI-RES LINE - Draw a vertical hi-res line from VEREND to VERBEG at the horizontal coordinate HORBEG with the palette register number in ALLCOL.
A,B,X,Y,U	E94E	DRAW A HI-RES LINE - Draw a hi-res line from (HORBEG, VERBEG) to (HOREND, VEREND).
X	E9B1	INCREMENT HORIZONTAL HI-RES POSITION - Increment the horizontal hi-res position (HORBEG).
X	E9B8	INCREMENT VERTICAL HI-RES POSITION - Increment the vertical hi-res position (VERBEG).
X	E9BF	DEREMENT HORIZONTAL HI-RES POSITION - Decrement the horizontal hi-res position (HORBEG).
X	E9C6	DECREMENT VERTICAL HI-RES POSITION - Decrement the vertical hi-res position (VERBEG).

A,B	E9CD	CALCULATE HI-RES ABS (VEREND-VERBEG) - Calculate the absolute value of the distance between VEREND and VERBEG. The carry flag will indicate which was the larger coordinate.
A,B	E9DB	CALCULATE HI-RES ABS (HOREND-HORBEG) - Calculate the absolute value of the distance between HOREND and HORBEG. The carry flag will indicate which was the larger coordinate.
B,U	EA16	POINT TO HI-RES PIXEL MOVE ROUTINE - Point the U register to the routine which will move the current pointer (X) to the right one pixel position for the current HSCREEN mode.
A,X	EA2D	MOVE A HI-RES PIXEL TO THE RIGHT - Adjust the X register and ACCA one pixel position to the right in the 2 color hi-res graphics mode. Enter with the screen address in the X register and the pixel mask in ACCA.
A,X	EA34	MOVE A HI-RES PIXEL TO THE RIGHT - Adjust the X register and ACCA one pixel position to the right in the 4 color hi-res graphics mode. Enter with the screen address in the X register and the pixel mask in ACCA.
A,X	EA3D	MOVE A HI-RES PIXEL TO THE RIGHT - Adjust the X register and ACCA one pixel position to the right in the 16 color hi-res graphics mode. Enter with the screen address in the X register and the pixel mask in ACCA.
B,X	EA45	ADJUST HI-RES SCREEN POINTER DOWN A ROW - Move the X register down one hi-res graphic row. The number of bytes per horizontal graphic row must be in HORBYT.
U,Y	EBCB	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.
A,X	FØØA	2 COLOR HI-RES PRINT DRIVER - Convert the bit pattern in ACCA into a hi-res 2 color pixel pattern and put that pixel pattern into the screen address pointed to by X. ALLCOL contains the palette register used.
A,B,X	FØ1A	4 COLOR HI-RES PRINT DRIVER - Convert the bit pattern in ACCA into a hi-res 4 color pixel pattern and put that pixel pattern into the screen address pointed to by X. ALLCOL contains the palette register used.
A,B,X	FØ45	16 COLOR HI-RES PRINT DRIVER - Convert the bit pattern in ACCA into a hi-res 16 color pixel pattern and put that pixel pattern into the screen address pointed to by X. ALLCOL contains the palette register used.

A,B	NEGACCD (F4CC)	NEGATE ACCD - Negate the value contained in ACCD.
A,B	F5FD	MULTIPLY ACCD BY 1Ø - Multiply the value contained in ACCD by 1Ø.
none	F6Ø8	NUMERIC ASCII TEST - Test ACCA to see if it contains a numeric (Ø-9) character. Return the carry flag clear if numeric, set if not.
A,B,X	COL32 (F652)	SET TO 32 COLUMN MODE - Set up the 32 column CoCo compatible text mode and clear the text screen. This routine will enable IRQ and FIRQ at the CPU level.
A,B,X	COL4Ø (F65C)	SET TO 4Ø COLUMN MODE - Set up the 4Ø column hi-res text mode and clear the text screen. This routine will enable IRQ and FIRQ at the CPU level.
A,B,X	COL8Ø (F679)	SET TO 8Ø COLUMN MODE - Set up the 8Ø column hi-res text mode and clear the text screen. This routine will enable IRQ and FIRQ at the CPU level.
A,B,X	F68C	CLEAR THE HI-RES TEXT SCREEN - Clear the hi-res text screen and home the cursor. The text screen must be in logical block 1 for this routine to function.
A,B,X	F8F7	MOVE THE HI-RES CURSOR - Move the hi-res cursor to the column and row numbers specified in ACCA and ACCB respectively.

* The CC register is modified by all routines

BASIC 1.2/EXTENDED 1.1 vs COLOR EXTENDED 2.0 DIFFERENCES

Listed below are all of the sections of code where the Basic 1.2 and Extended Basic 1.1 ROMs differ from the bottom half of the CoCo 3 ROM. If these changes are made in the Color Basic Unravelled and Extended Basic Unravelled books, those books can then be used with the new CoCo 3 ROM. The code below is CoCo 3 code.

```
                    * EXBAS WARM START ENTRY POINT
                    PATCH1
80C0                XBWMST  FCB   $FF                          SET TO NOT ALLOW A RESET TO WARM START HERE

80E8                L80E8   FCC   'EXTENDED COLOR BASIC 2.0'
8100                        FCB   CR
8101                        FCC   'COPR. 1982, 1986 BY TANDY  '
811C                        FCB   CR
811D                        FCC   'UNDER LICENSE FROM MICROSOFT'
8139                        FCB   CR

813A                PATCH13 FCB   CR,0

                    * DLOAD COMMAND
8C18                DLOAD   JSR   LA429                         CLOSE FILES

                    * PRESSING THE RESET WILL BRING YOU HERE
8C1B                INT.RSET ORCC  #$50                         DISABLE IRQ, FIRQ INTERRUPTS
8C1D                        LDA   #MC3+MC1                      32K INTERNAL ROM, MMU DISABLED, NON COCO COMPATIBLE
8C1F                        STA   INIT0
8C22                        CLR   SAM+$1E
8C25                        JMP   $C000
                    * FIRQ SERVICING ROUTINE ADDITIONS
8C28                        CLR   INT.FLAG                      SET THE INTERRUPT FLAG TO NOT VALID
8C2B                        CLR   PIA1+3                        DISABLE PIA 1. PORT B INTERRUPTS
                    * NON SELF-STARTING ROM CARTRIDGE INITIALIZATION CODE
8C2E                        LDA   #COCO+MMUEN+MC3+MC2           ENABLE MMU, 16K INTERNAL/16K EXTERNAL ROM
8C30                        STA   INIT0                         ALSO ENABLE STANDARD SCS, CONSTANT RAM AT FE00
8C33                        CLR   SAM+$1E                       FORCE THE ROM MODE
8C36                        RTS
                    * PUT A CHARACTER ON THE SCREEN PATCH
                    PATCH22A
8C37                L8C37   PSHS  A,B,X                        SAVE REGISTERS
8C39                        LDX   CURPOS                        POINT X TO THE CURRENT CHARACTER POSITION
8C3B                        LDB   HRWIDTH                       GET THE HI-RES TEXT MODE
8C3D                        LBNE  $F7AE                         BRANCH IF IN A HI-RES TEXT MODE (ALINK22)
8C41                L8C41   LDB   1,S                          RESTORE ACCB TO ITS FORMER GLORY
8C43                        JMP   LA30E                         GO BACK TO THE NON HI-RES CHARACTER DISPLAY ROUTINE

                    * CLS PATCH
                    PATCH23A
8C46                L8C46   PSHS  CC                           SAVE THE ZERO FLAG
8C48                        TST   HRWIDTH                       CHECK THE HI-RES TEXT MODE
8C4A                        BEQ   L8C4F                         BRANCH IF NOT IN A HI-RES TEXT MODE
8C4C                        JMP   $F6AD                         GO DO A HI-RES CLS (ALINK23)
8C4F                L8C4F   PULS  CC                           RESTORE THE ZERO FLAG
8C51                        JMP   LA913                         GO DO A NON HI-RES CLS
8C54                        NOP

                    * NEW 2.0 INITIALIZATION CODE
A02A                LA02A   LDA   #BLOCK7.2                    * PUT THE 'NORMAL' BLOCK BACK INTO LOGICAL BLOCK 2;
A02C                        STA   MMUREG+2                      * THE INITIALIZATION CODE AT $C000 USES BLOCK 6.4.
A02F                        LDX   #PIA1                         POINT X TO PIA1
A032                        LDD   #$FF34                        *
A035                        CLR   1,X                           CLEAR CONTROL REGISTER A ON PIA1
A037                        CLR   3,X                           CLEAR CONTROL REGISTER B ON PIA1
A039                        DECA                                A REG NOW HAS $FE
A03A                        STA   ,X                            BITS 1-7 ARE OUTPUTS, BIT 0 IS INPUT ON PIA1 SIDE A
A03C                        LDA   #$F8                          =
A03E                        STA   2,X                           = BITS 0-2 ARE INPUTS, BITS 3-7 ARE OUTPUTS ON B SIDE
A040                        STB   1,X                           * ENABLE PERIPHERAL REGISTERS, DISABLE PIA1 MPU
A042                        STB   3,X                           * INTERRUPTS AND SET CA2, CB2 AS OUTPUTS
A044                        CLR   2,X                           SET 6847 MODE TO ALPHA-NUMERIC
A046                        LDA   #2                            *
A048                        STA   ,X                            * MAKE RS232 OUTPUT MARKING
A04A                        LDA   #$FF34
A04C                        LDX   #PIA0                         POINT X TO PIA0
A04F                        CLR   1,X                           CLEAR PIA0 CONTROL REGISTER A
A051                        CLR   3,X                           CLEAR PIA0 CONTROL REGISTER B
A053                        CLR   ,X                            SET PIA0 SIDE A TO INPUT
A055                        STA   2,X                           * SET PIA0 SIDE B TO OUTPUT
```

```
A057                    STB   1,X                       * ENABLE PIAØ PERIPHERAL REGISTERS, DISABLE PIAØ
A059                    STB   3,X                       * MPU INTERRUPTS, SET CA2, CA1 TO OUTPUTS
A05B                    JMP   LAØ72
                  * THE MANUAL ROM CARTRIDGE START (EXEC &HEØ1Ø) JUMPS HERE
A05E                    JSR   LBC2E                     SET UP THE SYSTEM FOR A ROM CARTRIDGE
A061                    JMP   ROMPAK                    JUMP TO THE ROM-PAK


A084              LAØ84 LDX   #$7FFF                    FORCE THE TOP OF RAM TO BE $7FFF
A087                    BRA   LAØ93
A089                    NOP                             THESE 1Ø NOPs ARE JUST SPACE FILLERS
A08A                    NOP
A08B                    NOP
A08C                    NOP
A08D                    NOP
A08E                    NOP
A08F                    NOP
A09Ø                    NOP
A091                    NOP
A092                    NOP


A0CB                    JMP   EXBAS+2                   JUMP TO EXTENDED BASIC
A0CE              LAØCE PSHS  B,X
A0DØ                    TST   HRWIDTH                   CHECK FOR HI-RES TEXT MODE
A0D2                    LBNE  $F77E                     BRANCH IF A HI-RES TEXT MODE IS ENABLED (ALINK24)
A0D6              LAØD6 JSR   LA199                     BLINK THE CURSOR
A0D9                    JSR   >KEYIN                    GET A KEY
A0DC                    BEQ   LAØD6                     KEEP GOING UNTIL A KEY IS DEPRESSED
A0DE              LAØDE JMP   LA1B9                     REMOVE THE CURSOR FROM THE SCREEN AND RETURN

A0F3              LAØF3 JMP   LAC73                     GO TO MAIN LOOP OF BASIC

A0FC              LAØFC JSR   L8C28                     PREPARE TO USE THE CARTRIDGE ROM; FORCE THE ROM MODE


                  * THIS ROUTINE GETS A KEYSTRIKE FROM THE KEYBOARD IF A KEY
                  * IS DOWN. IT RETURNS A ZERO TRUE IF THERE WAS NO KEY DOWN.

A1C1              LA1C1 JMP   KEYIN
A1C4                    RTS                             * THESE RTS's ARE WHERE A CHECK WAS PERFORMED TO
A1C5                    RTS                             * SEE IF A KEY WAS DOWN. IF THE CHECK REVEALED THAT
A1C6                    RTS                             * A KEY WAS NOT DOWN, THEN THE KEYIN
A1C7                    RTS                             * ROUTINE WAS NOT CHECKED. WHICH MAKES BASIC RUN
A1C8                    RTS                             * FASTER
A1C9                    RTS
A1CA                    RTS

                  * INTERRUPT VECTORS
BFFØ                    FDB   $A681                     RESERVED FOR FUTURE USE (FILLED WITH GARBAGE BYTES)
BFF2                    FDB   INT.SWI3                  SOFTWARE INTERRUPT 3          ($FEEE)
BFF4                    FDB   INT.SWI2                  SOFTWARE INTERRUPT 2          ($FEF1)
BFF6                    FDB   INT.FIRQ                  FAST INTERRUPT REQUEST        ($FEF4)
BFF8                    FDB   INT.IRQ                   INTERRUPT REQUEST             ($FEF7)
BFFA                    FDB   INT.SWI                   SOFTWARE INTERRUPT            ($FEFA)
BFFC                    FDB   INT.NMI                   NON-MASKABLE INTERRUPT        ($FEFD)
BFFE                    FDB   INT.RESET                 RESET BUTTON                  ($8C1B)
```

HI RESOLUTION CHARACTER SET

Listed below is the character set available when in the high resolution text modes (WIDTH 40,80). The character set is repeated for character values $80-$FF.

00	Ç	10	ó	20		30	0	40	@	50	P	60	^	70	p
01	ü	11	æ	21	!	31	1	41	A	51	Q	61	a	71	q
02	é	12	Æ	22	"	32	2	42	B	52	R	62	b	72	r
03	â	13	ô	23	#	33	3	43	C	53	S	63	c	73	s
04	ä	14	ö	24	$	34	4	44	D	54	T	64	d	74	t
05	à	15	ø	25	%	35	5	45	E	55	U	65	e	75	u
06	å	16	û	26	&	36	6	46	F	56	V	66	f	76	v
07	ç	17	ù	27	'	37	7	47	G	57	W	67	g	77	w
08	ê	18	Ø	28	(	38	8	48	H	58	X	68	h	78	x
09	ë	19	Ö	29	)	39	9	49	I	59	Y	69	I	79	y
0A	è	1A	Ü	2A	*	3A	:	4A	J	5A	Z	6A	j	7A	z
0B	ï	1B	§	2B	+	3B	;	4B	K	5B	[	6B	k	7B	{
0C	î	1C	£	2C	,	3C	<	4C	L	5C	\	6C	l	7C	|
0D	ß	1D	±	2D	-	3D	=	4D	M	5D	]	6D	m	7D	}
0E	Ä	1E	°	2E	.	3D	>	4E	N	5E	↑	6E	n	7E	~
0F	Å	1F	ƒ	2F	/	3F	?	4F	O	5F	←	6F	o	7F	_

LOW RESOLUTION COCO COMPATIBLE CHARACTER SET

Listed below is the character set available when in the CoCo compatible text mode (WIDTH 32). Graphics blocks are printed for character values $80-$FF. The character set given below assumes that bit 4 of $FF22 is set. If that bit is clear, then the characters in the range of 0-$1F must be replaced by the corresponding characters in the range $40-$5F in inverse video.

00	^	10	p	20		30	0	40	@	50	P	60		70	0
01	a	11	q	21	!	31	1	41	A	51	Q	61	!	71	1
02	b	12	r	22	"	32	2	42	B	52	R	62	"	72	2
03	c	13	s	23	#	33	3	43	C	53	S	63	#	73	3
04	d	14	t	24	$	34	4	44	D	54	T	64	$	74	4
05	e	15	u	25	%	35	5	45	E	55	U	65	%	75	5
06	f	16	v	26	&	36	6	46	F	56	V	66	&	76	6
07	g	17	w	27	'	37	7	47	G	57	W	67	'	77	7
08	h	18	x	28	(	38	8	48	H	58	X	68	(	78	8
09	I	19	y	29	)	39	9	49	I	59	Y	69	)	79	9
0A	j	1A	z	2A	*	3A	:	4A	J	5A	Z	6A	*	7A	:
0B	k	1B	{	2B	+	3B	;	4B	K	5B	[	6B	+	7B	;
0C	l	1C	|	2C	,	3C	<	4C	L	5C	\	6C	,	7C	<
0D	m	1D	}	2D	-	3D	=	4D	M	5D	]	6D	-	7D	=
0E	n	1E	~	2E	.	3D	>	4E	N	5E	↑	6E	.	7E	>
0F	o	1F	_	2F	/	3F	?	4F	O	5F	←	6F	/	7F	?

Note: The characters defined by $20-$3F are inverse video.