the frob™

A Connoisseur's Guide

Read Me First

As with any computer-based product we recommend you read the accompanying manual. However, if you're like most of us at FROBCO, you won't. In that case scan through the manual first, paying particular attention to the sections marked with the heading "NOTE:".

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

COPYRIGHT NOTICE

TRADEMARK

The names FROB, FROB MON, EXPLORER and FROBCO are trademarks of FROBCO.

References are made throughout this manual to the Video Computer System, commonly known as VCS. Video Computer System is a trademark of Atari, Inc.

Atari is a registered trademark of Atari, Inc.

Apple is a registered trademark of Apple Computers, Inc.

DISCLAIMER OF ALL WARRANTIES AND LIABILITY

FROBCO makes no warranties, either express or implied, with respect to this manual or with respect to the software described in this manual, its quality, performance, merchantability, or fitness for any particular purpose. FROBCO software is sold or licensed "as is." The entire risk as to its quality and performance is with the buyer. In no event will FROBCO be liable for direct, indirect, incidental, or consequential damages resulting from any defect in the software, or use of this product in any way, even if FROBCO has been advised of the possibility of such damages. Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you.

LIMITED WARRANTY

FROBCO warrants this FROBCO Software Development Product to be in good working order for a period of one year from the date of purchase from FROBCO or an authorized FROBCO dealer. Should this Product fail to be in good working order at any time during this one year warranty period, FROBCO will, at its option, repair or replace this Product at no additional charge except as set forth below. Repair parts and replacement Products will be furnished on an exchange basis and will be either reconditioned or new. All replaced parts and Products become the property of FROBCO. This limited warranty does not include service to repair damage to the Product resulting from accident, disaster, misuse, abuse, or non-FROBCO modification of the Product.

Limited Warranty service may be obtained by delivering the Product during the one year warranty period to an authorized FROBCO dealer or FROBCO Service Center and providing proof of purchase date. If this Product is delivered by mail, you agree to insure the Product or assume the risk of loss or damage in transit, to prepay shipping charges to the warranty service location and to use the original shipping container or equivalent. Contact an authorized FROBCO dealer or write to FROBCO, Sales and Service, 603 Mission Street, Santa Cruz, California 95060, for further information.

ALL EXPRESS AND IMPLIED WARRANTIES FOR THIS PRODUCT INCLUDING THE WARRANTIES OF MERCHANT ABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED IN DURATION TO A PERIOD OF ONE YEAR FROM THE DATE OF PURCHASE, AND NO WARRANTIES, WHETHER EXPRESS OR IMPLIED, WILL APPLY AFTER THIS PERIOD. SOME STATES DO NOT ALLOW LIMITATIONS ON HOW LONG AN IMPLIED WARRANTY LASTS, SO THE ABOVE LIMITATIONS MAY NOT APPLY TO YOU.

IF THIS PRODUCT IS NOT IN GOOD WORKING ORDER AS WARRANTED ABOVE, YOUR SOLE REMEMDY SHALL BE REPAIR OR REPLACEMENT AS PROVIDED ABOVE. IN NO EVENT WILL FROBCO BE LIABLE TO YOU FOR ANY DAMAGES, INCLUDING ANY LOST PROFITS, LOST SAVINGS OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF OR INABILITY TO USE SUCH PRODUCT, EVEN IF FROBCO OR AN AUTHORIZED FROBCO DEALER HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, OR FOR ANY CLAIM BY ANY OTHER PARTY.

SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES FOR CONSUMER PRODUCTS, SO THE ABOVE LIMITATIONS OR EXCLUSIONS MAY NOT APPLY TO YOU.

THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS, AND YOU MAY ALSO HAVE OTHER RIGHTS WHICH MAY VARY FROM STATE TO STATE.

STANDARD TERMS AND CONDITIONS
FOR END USER SOFTWARE LICENSE AGREEMENT

DEFINITIONS

(a) "Licensed Programs" shall mean all software as defined herein, developed or furnished by FROBCO and utilized by licensee, and all printed documentation for implementing and use of the software.

(b) "Software" shall mean the alterable programs and routines developed by or for FROBCO and furnished to Licensee for the internal operation of a computer, all Derivative Works or such software as defined herein, and the media supplied by FROBCO on which the software is delivered.

(c) "Derivative Work" shall mean licensee's alterations of Licensed Programs, or licensee's inclusion of all or part of Licensed Programs with other program material.

(d) "Industrial Property Rights" shall mean all patents, licenses, trademarks, trade names, inventions and copyrights relating to the orgin, design, manufacture, programming, operation or service of Licensed Programs.

ACCEPTANCE

This agreement becomes effective when executed by authorized representatives of licensee and FROBCO.

COPYING OR TRANSFERRING LICENSED PROGRAMS

(a) Licensee may make no more that (5) five copies of Licensed Programs for meeting Licensee's internal requirements. Licensee agrees to maintain appropriate records of the location and number of all such copies.

(b) Licensee may nqt copy the printed materials furnished with Licensed Programs. Additional copies of these materials are available from FROBCO.

(c) Licensee may use Licensed Programs in machine-readable form on one (1) computer system only, as denoted in this license.

(d) Licensee recognizes the proprietary nature of the Licensed Programs and Industrial Property Rights and agrees to preserve and protect FROBCO's therein. All information relating to the Licensed Programs provided to Licensee or its agents shall be retained and shall not be used or disclosed except for the purpose of meeting Licensee's internal needs. Licensee agrees to take reasonable caution to protect FROBCO's interest in the Licensed Programs and Industrial Property Rights.

(e) Licensee agrees to reproduce and include any and all copyright notices on all copies, in any form, including Derivative Works, of Licensed Programs.

TITLE AND PROPRIETARY RIGHTS

Title and ownership to the Licensed Programs and Industrial Property Rights is not transfered to licensee.

MODIFICATION

Licensee may produce Derivative Works provided licensee reproduces the copyright notice on all copies. Upon discontinuance of this agreement all portions of the Licensed Programs supplied by FROBCO will be completely removed from all such Derivative Works.

TERMINATION

In the event Licensee shall fail to keep, observe, or perform and covenant or condition set forth herein, FROBCO may, at its option, terminate this agreement on fifteen (15) days written notice to Licensee. Within thirty (30) days after termination and/or discontinued use, Licensee shall certify in writing that through its best efforts and to the best of its knowledge the orginal and all copies, in whole or in part, of the terminated and/or discontinued Licensed Programs and any material relating thereto have been destroyed or returned to FROBCO.

WARRANTY

FROBCO warrants that the Licensed Programs shall be free of material defects and conform to current FROBCO specifications for a period of one year from the date of delivery to Licensee. FROBCO's sole obligation, and Licensee's sole remedy, shall be for FROBCO to exert its best efforts to correct such defects and to supply Licensee with a corrected version within a reasonable time after Licensee notifies FROBCO in writing of any defect. This warranty does not cover any modifications to the Licensed Programs developed by any party other than FROBCO, or any defects caused by or otherwise related to such modifications.

LIMITATION OF LIABILITY

THIS WARRANTY IS IN LIEU OF, AND LICENSEE HEREBY WAIVES, ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. FROBCO NEITHER ASSUMES NOR AUTHORIZES LICENSEE OR ANY OTHER PERSON TO ASSUME FOR FROBCO ANY OTHER WARRANTY. FROBCO SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER PERSON AND LICENSEE AGREES TO INDEMNIFY FROBCO WITH RESPECT TO ANY CLAIMS AGAINST FROBCO FOR INCIDENTAL, SPECIAL, INDIRECT OR CONSQUENTIAL DAMAGES, INCLUDING LOSS OF PROFIT.

GENERAL

(a) This agreement constitutes the entire agreement between the parties and supersedes all prior agreements and understandings between them relating to the subject matter hereunder and no modification of the agreement shall be binding on either party unless it is in writing and signed by both parties.

(b) No waiver of any portion of the agreement shall be effective unless made in writing. No waiver of any breach of any provision of this agreement shall constitute a waiver of any subsequent breach of the same or any other provision of this agreement.

(c) The rights and obligations of the parties to this agreement shall be governed and construed in accordance with the laws of the State of California.

(d) Section headings are for convenience only and shall not be considered in the interpretation of this agreement.

(e) The plural shall include the singular, and the singular shall include the plural, whenever used.

(f) The provisions of this agreement are severable and if any one or more such provisions are judicially determined to be illegal or otherwise unenforceable by and between the parties hereto.

(g) In the event it becomes necessary for FROBCO to retain the services of an attorney to enforce any of the provisions of this agreement the Licensee agrees to pay the costs thereof, including any court costs.

SOURCE CODE LISTINGS
OF
PROM DEVELOPMENT SOFTWARE

CHAPTER 1
Introduction

1.1 Purpose Of The FROB

The FROB is an periperal printed circuit board for the Apple II computer that provides a development interface to the Atari VCS game player. The purpose of the FROB is to provide a professional software development system for the Atari Video Computer System (VCS). The FROB system accomplishes this goal with hardware, software and user support. The FROB hardware is the FROB card, cable and FROB cartridge adapter. The software is the FROB tool kit and FROB MON, and Explorer. The support is a team of people at FROBCO who have been developing games for over two years and are willing to answer questions, especially in the areas of game production and marketing.

The FROB is being marketed so that individual programmers can have the same advantages as programmers working for the game companies. It is our hope that the FROB will lead to many new creative ideas that will be made available to the general public. We see the FROB as a way of opening up new product areas, beyond games, in education and business.

One of the key elements of a good development system for the professional is access to the source code for all development tools. In most cases the only way to gain access is to write the code yourself; if someone sells you an assembler they are not about to hand you the source code. But we want to make the FROB as good a development system as possible, so we are releasing all of the source code. With this information users can tailor the system to fit their needs.

1.2 Organization Of This Manual

This manual is organized to provide a complete reference guide for the FROB system and its various components.

This manual will provide a good grounding in the use of the FROB system as a VCS software development tool. It is not a guide to game design. This and other topics will be covered in future manuals from FROBCO.

CHAPTER 2

Hardware Overview

2.1 Architecture

The principal hardware component of the FROB system, the FROB card, is an Apple peripheral card designed to be inserted in any one of the Apple slots from 1 to 7 (not slot Ø). This card is attached to the FROB cartridge adapter via a flat cable. The FROB Cartridge Adapter is designed to be inserted in the cartridge slot of the VCS with no modification to the VCS. The FROB card is a specialized ROM emulator that emulates the 4K address space of the VCS. The FROB also has a Bi-Directional Port that allows a program running in the VCS logical address space to communicate with a program running in the Apple logical address space.

2.1.1 Memory Map

The FROB resides in the Apple slot memory space, and utilizes both the Apple Peripheral Card I/O locations and the Peripheral Card PROM locations (refer to Apple reference Manual). The FROB does not require any of the Apple main memory. The control and status port for both the FROB memory and the Bi-Directional Port is the first location of the Peripheral Card I/O space for the particular slot the FROB card is inserted into. The read/write port for the Bi-Directional Port is the second location of the Peripheral Card I/O space.

EXAMPLE:

For Slot 1 the control and status port is the Apple address $CØ9Ø.

For Slot 2 the read/write port is Apple address $CØ91.

The 4K bytes of FROB memory is mapped into 16 pages of 256 bytes each, of which the Apple can only access one page at a time. The Control Port can select a FROB page which is then available for reading or modifying in the 256 byte Peripheral Card PROM space of the Apple. The FROB memory is selected by writing the desired page number to the FROB Control Port.

EXAMPLE:

If the FROB is in Slot 1, the user selects page 9 of the FROB memory by writing a $Ø9 to location $CØ9Ø of the Apple. The user can then access page 9 in the Slot 1 Peripheral Card PROM Space locations $C1ØØ to $C1FF. The user can then access page 15 by writing a $ØF to location $CØ9Ø and again view the data at locations $C1ØØ to $C1FF.

NOTE: The FROB cannot write to the memory in the logical VCS memory space; only the Apple can. Thus the VCS continues to treat FROB memory as if it were ROM. The only exception to this is the Bi-Directional Port as described in section 2.1.3.

2.1.2 FROB Hardware Operation

The operation of the FROB is controlled through the FROB control port. The most significant four bits of the control port are used for control information. The least significant four bits of the control port contain memory map information as described in section 2.1.1.

The FROB card can be under control of the Apple, the VCS, or both when the Bi-Directional Port is in use.

When the FROB is under the control of the Apple, the user may read, or modify the FROB memory. While the FROB is under control of the Apple, the VCS cannot read the FROB memory, even though the FROB Cartridge Adapter is inserted in the VCS.

NOTE: APPLE CONTROL IS SELECTED BY WRITING A Ø TO BIT 4 OF THE FROB CONTROL REGISTER.

EXAMPLE: If the FROB is in Slot 1 then writing $ØØ to address $CØ9Ø will place the FROB in Apple control and also select page Ø of the FROB memory for access.

When the FROB is under VCS control the user cannot access FROB memory, but can do anything else on the Apple as long as a new control value is not written to the FROB Control Port. The VCS will appear to have a normal cartridge inserted and if there is a valid program image in the FROB it will run on the VCS.

NOTE: VCS CONTROL IS SELECTED BY WRITING A 1 TO BIT 4 OF THE CONTROL REGISTER. THE LEAST SIGNIFICANT BITS HAVE NO EFFECT.

EXAMPLE: If the FROB is in Slot 1, it may be placed under VCS control by writing a $1Ø to location $CØ9Ø. The lower four bits have no effect so writing a $1F to location $CØ9Ø will have the same result.

When the FROB is used with the Bi-Directional Port both the Apple and the VCS can talk to each other through their respective status ports and read/write ports as described in section 2.1.3 below.

NOTE: BI-DIRECTIONAL PORT CONTROL IS SELECTED BY WRITING A 1 TO BIT 5 OF THE FROB CONTROL PORT. THIS CAN BE DONE WHEN EITHER THE Apple OR THE VCS HAVE CONTROL OF THE VCS ROM MEMORY SPACE. HOWEVER TURNING THE BI-DIRECTIONAL PORT ON WHILE THE Apple HAS CONTROL (BIT 4 SET TO Ø) WILL HAVE NO EFFECT.

EXAMPLE: If the FROB is in Slot 1 then Bi-Directional Port operation can be turned on with VCS control of FROB memory by writing a $3Ø to location $CØ9Ø. The lower 4 bits are not significant, so writing a $3F to $CØ9Ø will have the same effect.

2.1.3 Bi-Directional Port

The Bi-Directional Port is used mainly by the FROB MON interactive debugger. It allows software in the logical VCS memory space to communicate with software in the Apple space in real time. There are two sides to the Bi-Directional Port, the Apple side and the VCS side. Each side has its own Status and Read/Write Ports.

On the Apple side there are two ports; the Status Port and Read/Write (R/W) Port. The Status Port is the same location as the FROB control port as described in section 2.1.1. The R/W Port is the next location after the Status Port as described in section 2.1.1.

On the VCS side there are three ports; a Status Port, a Read Port, and a Write Port. They are located in the VCS address space as follows:

```
NOTE: STATUS PORT - $FFF1
      WRITE PORT  - $FFFØ
      READ PORT   - $FFF2
```

The only two significant bits of the Status Port on either side are the most significant bits, bits 6 and 7. The other bits are not significant and may "float" during actual operation.

Bit 7, the most significant bit of the Status register is "Write OK" bit. If this bit is set, i.e., is a logical 1 then one side may write to the other. If this bit is not set, i.e., a logical Ø, then it is not ok to write to the other side. In the Apple, if bit 7 of the Status register is set then the user may transmit a byte of data to the VCS by writing it to the R/W Port. Bit 7 of the Status Port then goes low and remains low until the data is successfully read by the VCS, at which time it is set again. The same is true on the VCS side with the exception that there are separate Read and Write Ports.

Bit 6 is the "Read OK" bit. If this bit is set, i.e. is a logical 1, then there is a valid data byte in the Read Port which was sent there from the other side. If bit 6 is not set, i.e., it is a logical Ø, the contents of the Read Port are not valid. In the VCS if bit 6 of the Status Port at $FFF1 is set, then there is a valid data byte in the Read Port at location $FFF2. The same is true on the Apple side, with the exception that there is only one R/W Port, which is used for both purposes.

For an example of Bi-Directional Port usage, see the FROB MON listing in this manual and the AMON code on the FROB development disk.

CHAPTER 3

Development System Overview

This section provides an overview of the development environment you should have while using the FROB. The development system environment has three major components: the development hardware, development software, and the developer. The FROB system provides the first two; the user provides the last one.

FROBCO's experience has shown that a VCS programmer should have a good working knowledge of the 6502 processor. The VCS actually has a 6507 processor, which is functionally identical to the 6502 with the exception that it can only address 8K of memory and it does not have the IRQ or NMI interrupts. The developer should also have experience with limited memory, ROM based, computer system development. The computational power of the VCS is minimal compared to other target game systems such as the Apple or the Atari 800. It is, therefore, imperative for the developer to have a good development system.

3.1 Hardware

The basic VCS development system hardware set consists of:

1 Apple II with 48K and at least one disk drive

1 Atari VCS and a color TV

1 FROB card

1 PROM Programmer capable of programming a 2732 EPROM

The Apple, VCS, and the FROB provide the basic hardware set for designing, implementing, and debugging VCS programs. The PROM Programmer is needed to translate the software into the finished product by programming an 2732 EPROM. The 2732 EPROM is simply inserted into the FROB Cartridge Adapter in place of the cable from the FROB card. A FROB Cartridge Adapter with a working 2732 is a direct replacement for a VCS cartridge with a masked ROM.

The PROM Programmer can be either a stand-alone device or another peripheral card for the Apple. FROBCO uses the FROB BURNER, a peripheral card for the Apple. The FROB BURNER has a VCS cartridge slot, which allows the user to place a blank 2732 into a FROB Cartridge Adapter and then insert into the FROB BURNER to be programmed. Once the 2732 has been programmed, the FROB BURNER can read and verify the contents. The FROB BURNER can also be used to read a previously programmed cartridge to verify its contents. Please refer to the FROB BURNER data sheet and manual for more information.

3.2 Software

There are three major components to VCS software development: 65Ø2 development tools, FROB development tools, and the software being developed.

The Apple was chosen as a host for the FROB because of the availability of many 65Ø2 assemblers, editors, and 65Ø2 programming aids that exist on the Apple. We have chosen EDASM and APPLESOFT for all FROB development tools because they are the most common and are the assembler and BASIC supported by Apple, Inc. Choice of assembler and editor is entirely up to the developer.

The second major component of VCS program development is the FROB development software. This software is used during the debugging phase after the basic target software has been completed. The FROB development software has three major components; an interactive debugging system (AMON + FMON), a software laboratory program (EXPLORER), and the FROB TOOL KIT. AMON and FMON are described in chapter 6 of this manual. The FROB TOOL KIT is described in chapter 7 of this manual. The FROB TOOL KIT is a collection of utilities which aid the developer in loading, listing, saving, and playing around with target software in the FROB. The EXPLORER is described in chapter 8 of this manual. The EXPLORER allows the user to conduct software experiments with the VCS and to try out various video and audio effects.

The third major software component of VCS program development is the target software itself; currently a game of one type or another. It is FROBCO's hope that, now that there is a good VCS development system with the FROB, we will begin to see a more diverse set of VCS cartridges, such as educational and business related products. The FROB system is not designed to be a "game only" development system. It is a general-purpose software development system with the capabilities that an experienced software developer comes to expect. We will continue to add new development software and hardware to the FROB family to meet the needs of VCS developers.

CHAPTER 4

Installation

It is assumed that the FROB user is an experienced Apple user and is therefore familiar with the dos and don'ts of inserting Apple peripheral cards. It is further assumed that the user is familiar with Apple disk software. If these assumptions are not true for you, we strongly advise you to spend some time getting to know your Apple and reading the Apple reference manuals, or enlist help from a friend who knows the Apple.

4.1 Hardware

The FROB card may be placed in Slots 1 thru 7 of the Apple Peripheral Card slots. Slot 6 is generally used for the disk controller and Slot 1 for a printer card, so the FROB can be placed in any of slots 2 through 5, or slot 7. FROB development software is designed to ask the user which slot the FROB card is in, so you needn't always put it in the same place.

NOTE: The current version of the FROB Cartridge Adapter must always be inserted in the Atari with the cable from the FROB card UP!! If you put it in backwards it could damage the VCS. Also, you should always turn the VCS off when inserting or removing the cartridge adapter. It is not necessary to turn the Apple off when inserting the cartridge adapter into the VCS.

NOTE: The user should always take care when inserting the FROB card cable into either the FROB card or into a FROB Cartridge Adapter. The cable is always inserted into the FROB card with the cable coming out the back of the Apple (where there are openings for it when the cover is on the Apple). The cable is always inserted into the cartridge adapter with the cable extending away from the gold contacts.

4.2 Software

The FROB development software is supplied on a standard Apple DOS diskette. This disk is not protected, so several copies should be made and the original stored in a safe place.

The user should become familiar with the tools available on the FROB development disk and then make up a working disk with the tools needed for a particular project.

CHAPTER 5

Using the FROB

This chapter is a short tutorial for using the FROB. The best way to begin is to just sit down and start playing with it.

When the FROB card has been installed in the Apple, the cartridge adapter inserted into the VCS and a disk with the FROB development tools in the currently selected drive, we can begin.

There are two modes for the FROB card: you can use the FROB TOOL KIT utility routines to talk to the FROB card, or the Apple Monitor program to control the FROB card directly by writing to its control port.

First, load a target program into the FROB. This can be done with the FLOAD program in the tool kit or by entering the monitor and loading it in by hand a byte at a time.

The second step is to make sure that the reset interrupt vector of the 6507 in the VCS is pointing at the starting location of your target software. In most cases this was taken care of when you loaded the file into the FROB card. If you also loaded FROB MON into the FROB card, it pointed the reset interrupt vector at its starting location. The following example applies to either case:

1) Enter the Apple Monitor by typing "CALL -151" to the APPLESOFT prompt "]".

2) Determine the address of the FROB card Control Port (Section 2.1.2). For this example we will assume the FROB card is in slot 2. Therefore, the control port is $C0A0 and the FROB memory will be available at locations $C200 through $C2FF.

3) Write a $0F to location $C0A0 by replying to the monitor prompt with:

    *C0A0:0F

    This selects page 15 of the FROB memory space which corresponds to address locations $FF00 through $FFFF in the VCS 6507 address space.

4) As we already know the reset interrupt vector is at locations $FFFC and $FFFD in the VCS 6507 address space. This translates to locations $C2FC and $C2FD in the Apple space, assuming the FROB card is in slot 2 and we have selected page 15 with step

5) Now we can look at what is in these two locations by giving the monitor command:

```
*C2FC.C2FD
```

The monitor should respond with two bytes of information which should be the desired reset interrupt vector in the 6502 byte reversed format; that is if the desired reset vector was $F000 then location $C2FC should be $00 and location $C2FD should be $F0.

6) Let's assume that the reset interrupt vector was not what we wanted. So let's change it. If we want the vector to be $F000 then give the monitor the command:

```
*C2FC: 00 F0
```

All this assumes that you haven't written anything to the FROB card Control Port at $C0A0 to change the fact that we have selected page 15.

7) Now that we have the target program ready to go, let's do it! We can turn the FROB card memory over to the VCS by writing a $10 to the FROB card Control Port at location $C0A0 (Assuming the FROB card is still in Slot 2 of the Apple). We can do this with the monitor command:

```
*C0A0:10
```

8) We now need to turn on (or off and then on again) the VCS power switch. This causes a power interrupt and the VCS 6507 will restart its program at the location that we went to all the trouble of stuffing into the reset interrupt vector. You can now use the VCS to do whatever it is supposed to be doing with the software you loaded into the FROB in the first place.

9) After you get tired of playing with the VCS you can stop the program and return the FROB card to Apple control by writing a $0n to the FROB card Control Port at location $C0A0 (n represents the page number you wish to view). So if we want to stop the program and look at page 0 we need to give the monitor the command:

*C0A0:00

10) Now let's assume that there is a "LDA #00" instruction at location $F01A in the target software. Let's also assume that our diligent research in step 7 indicated that this instruction should be "LDA #01". So let's change it. We have already selected page 0 in step 9 so we should see the "LDA #00" instruction by asking the monitor to disassemble the memory, starting with the Apple location $C21A which corresponds to location $F01A in the VCS space.

*C21AL

If we verify that this is the right place, we can change the "LDA #00" to "LDA #01" by writing a $01 to location $C21B.

*C21B:01

We can now try out this patch by repeating steps 8 and 9 above.

CHAPTER 6

FROB MON - Interactive Debugging

FROB MON is an interactive debugger for use with the FROB and its Apple host.

Since FROB MON is an interactive debugger designed to run in a host/slave environment, it has two separate software components: FMON, the "Atari Side"(target), which resides in the FROB card memory and is logically running on the Atari VCS, and AMON, the "Apple Side"(host) which runs in Apple memory.

The Atari side is a collection of 6502 assembly routines that perform basic debugging tasks. The major constraint in these routines is size. They are designed to be as small as possible to debug a large program in the 4K VCS memory space.

The Apple side is AMON, an APPLESOFT program that calls several assembly subroutines. We chose APPLESOFT because it is the most common language available for the Apple. The Apple side is the command monitor and allows the user to control the routines running in the Atari Side. The design is similar to the Apple Monitor.

Chapter 6

6.1 FMON Operation

This section describes the functions available on the Atari side. Please refer to the source code for more documentation.

6.1.1 Summary

The Atari side routines are a collection of individual modules. In order to use as little space as possible in the VCS, a command interpreter communicates over the FROB's Bi-Directional Port with the Apple side monitor. So as not to alter any of the RAM memory in page zero of the VCS 6507 address space, FMON saves any zero page RAM data it may need to alter by first sending it to AMON for safekeeping. Whenever FMON is asked to execute the target software, it asks AMON for the data back and restores any zero page locations it may have altered. This handshake is necessary because the VCS cannot write to the memory in its normal ROM address space; only the Apple can write to this memory when it has control of the FROB card.

NOTE: THE FOLLOWING FORMAT APPLIES TO ALL SUBSEQUENT SOFTWARE COMMANDS WITHIN THIS CHAPTER.

SYNTAX: All FMON commands consists of a command byte optionally followed by parameter bytes. Some of the commands also return one or more bytes in a pre-ordained order.

ERROR HANDLING: If FMON gets a command byte that is invalid, it ignores it and looks at the next byte. The Atari side command interpreter is designed to be called exactly by the Apple side and, therefore, does not waste space with elaborate error checking.

NOTATION CONVENTIONS:

CB - Command code byte

IPn - Input parameter n to command

RPn - Return parameter n from command

NOTE: All values will in HEX will be denoted by a "$" prefix.

6.1.2 Read Memory

READ MEMORY reads from any memory location in the VCS address space and returns the values in sequentially increasing address order.

SYNTAX: CB,IP1,IP2,IP3

CB = $10

IP1 = HIGH ORDER OF TARGET ADDRESS

IP2 = LOW ORDER OF TARGET ADDRESS

IP3 = NUMBER OF BYTES TO READ (MUST BE $00<IP3<=$FF)

RETURNS: RP1 thru RPn

RP1 - RPn = RETURN DATA READ FROM MEMORY, n = IP3

NOTES:

1) All addresses are 16 bits.
2) Can only read up to 255 bytes for each call.
3) Will always return IP3 number of bytes and will wrap around from $FFFF to $0000.
4) Must be called to read at least 1 byte.

6.1.3 Write Memory

WRITE MEMORY writes a series of bytes to any location within the VCS address space in sequential order.

SYNTAX: CB,IP1,IP2,IP3,IP4,...IPn

CB = $20

IP1 = HIGH ORDER OF TARGET ADDRESS

IP2 = LOW ORDER OF TARGET ADDRESS

IP3 = NUMBER OF BYTES TO WRITE (MUST BE $00<IP3<=$FF)

IP4 - IPn = DATA BYTES TO WRITE (n = 3 + IP3)

NOTES:

1) No return bytes.
2) Must be called with at least 1 byte to write.
3) Will wrap from $FFFF to $0000.
4) If the Apple side does not supply IP3 number bytes as data, this command will continue to consume the next command bytes until IP3 number of bytes have been delivered to it.

6.1.4 Go

GO allows the user to start or restart the program being debugged from an arbitrary address. This is to start a program or to restart it after a break point.

SYNTAX: CB

CB = $3Ø

RETURNS: RP1 - RP38

RP1 = A REG
RP2 = FLAG CODE
RP3 = X REG
RP4 = Y REG
RP5 = S REG
RP6 - RP37 = LOCATIONS $EØ - $FF
RP38 = P REG

NOTES:

1) If no break point (BP) was set, or if one is not encountered, then the Atari side will not return control of the VCS to the Apple side. If a BP was encountered, then the GO command will return 36 bytes to save its state after its return from a BP.
2) If the GO command does return from a BP it does NOT automatically reset the BP; this must be done by the AMON CLEAR BP command.
3) The FLAG CODE is a code that gives the true state of the N and Z flags. The P REG when it is returned may not. Please refer to the FMON source code to see how FLAG CODE is used.

6.1.5 Go Indirect

GO INDIRECT allows the FMON caller to direct FMON to jump to another address in memory. AMON uses this command to make FMON jump to the SLEEP routine at location $E0.

SYNTAX: CB,IP1,IP2

CB = $40

IP1 = HIGH ORDER ADDRESS BYTE OF JUMP VECTOR

IP2 = LOW ORDER ADDRESS BYTE OF JUMP VECTOR

NOTES:

1) This command will use the jump vector given to it as an indirect jump vector.
2) It does not restore the state of RAM before it makes the jump.

6.2 AMON Operation

The Apple side of the FROB MON system is the user interface to the debugger, AMON.

AMON is an APPLESOFT program invoked by the APPLESOFT RUN command. AMON sets up the debug environment and then prompts the user for commands.

6.2.1 Summary

AMON is invoked with the APPLESOFT RUN command. The following files must be present on the development disk:

AMON
FMON.OBJØ

AMON will ask the user which slot the FROB card is in. AMON will then ask if the user wishes to load a program file into the FROB.

Next, AMON asks the user if the user wants to load FMON into high memory. If the user wants to do any interactive debugging FMON must be loaded. FMON is uneccessary for simply examining the target software particularly if the target software is very large and uses the memory where FMON lives (refer to FMON source code). Use of AMON without FMON is analogous to controlling the FROB directly from the Apple monitor except that AMON handles the handshake with FMON automatically.

AMON then displays its prompt character ">" and the AMON Command Interpreter begins accepting commands from the user.

COMMAND CONVENTIONS:

1) The command syntax is usually the argument followed by the one character command. For example:

    >1000.10

    Read memory starting at $1000 for 16 bytes

2) All values are in hex.
3) All hex numbers will be four digits or less.

NOTE: AMON accepts addresses in a form that corresponds to the VCS 6507 address space and then translates them into their FROB memory space equivalents. As an example AMON would take the request to read location $F310 and translate that into a series of commands that would first select page 3 of the FROB address space and then it would read location $C210 (assuming the FROB is in slot 2) of the Apple Peripheral PROM space.

6.2.2 Read Memory

READ MEMORY allows the user to read a single location or a block of locations.

SYNTAX:

addr	- Reads one location at "addr".
addr.val	- Reads a range of locations starting at "addr" for "val" number of bytes.

EXAMPLE:

F003	- Reads the contents of location $F003 and displays it.
F003.10	- Reads 16 bytes starting at location $F003, and displays them.

6.2.3 Alter Memory

ALTER MEMORY allows the user to change the contents of a memory location.

SYNTAX:

addr<val	- Replace the contents of memory at "addr" with "val".

EXAMPLE:

82<FF	- Stores an $FF in location $82.
F100<00 F101<00 F102<00	- Zeros out locations $F100 through $F102.

NOTES:

1) ALTER MEMORY will try to store the specified location, whether or not it is a valid data storage location.

6.2.4 Disassemble And List Memory

DISASSEMBLE and LIST will list a range of memory and will attempt to disassemble it into 6502 code rather than display its contents as in section 6.2.2.

SYNTAX:

addrLval	- Will list and disassemble val number of bytes. It will always disassemble at least one instruction which may be one or two bytes more than specified by val.
L	- Will disassemble 30 bytes from the last location that was listed.

EXAMPLE:

>F340L10	- Will list 16 bytes of code starting at location $F340.
>L	- Will disassemble an additional 30 bytes of code starting at location $F350.

6.2.5 Register Display

REGISTER DISPLAY allows the user to display the registers of the 6502 in the VCS. It will display the A,X,Y,S, and P registers.

SYNTAX:

R	- Displays the values of the registers A,X,Y,S, and P.

EXAMPLES:

>R	- Display the registers.

6.2.6 Modify Registers

MODIFY REGISTERS allows the user to modify the contents of a particular register.

SYNTAX:

M - Alter the contents of a register. The command then asks the user for which register and the value to place in it.

EXAMPLE:

>M - Will then ask:

```
WHICH REGISTER? A
NEW VALUE? FF
```

- This will place an $FF in register A.

NOTES:

1) The user may choose not to modify the register by entering a carriage return to either the register or the value question.
2) The user will need to repeat this command for each register that is to be modified.
3) If the user attempts to modify the P REG by setting both the N and Z flags to true AMON will complain because this is not a possible condition with the 6502.

6.2.7 Go From An Address

GO FROM AN ADDRESS allows the user to run programs and continue from break points from AMON.

SYNTAX:

addrG - Start executing from location "addr". AMON will ask the user: "DO YOU WISH TO WAIT FOR A BREAK POINT?". If the user answers "Y" to this question, then AMON will wait for FMON to respond when a Break Point is encountered. If the user answers "N" then AMON will immediately return with a prompt.

EXAMPLE 1:

```
>F332G              - Begin executing with the
                      instruction at address $F332.

     *DO YOU WISH TO WAIT FOR A BREAK POINT? N

>
```

EXAMPLE 2:

```
>F823P              - Place a break point at $F823.
>F400G              - Begin executing with the
                      instruction at address $F400.

     *DO YOU WISH TO WAIT FOR A BREAK POINT? Y

     *WAITING FOR BREAK POINT*
     *BREAK POINT ENCOUNTERED*

>                   - You are now at location $F823.
```

6.2.8 Go From Break Point

GO FROM BREAK POINT allows the user to start the target software from the location of the last break point. Upon returning from a break point AMON always performs a KILL BREAK POINT COMMAND (section 6.2.10) for the user. AMON also remembers where the last break point was. This allows the user to set a break point, go from an address (section 6.2.7), return from that break point, place another break point (section 6.2.9), and then go from the location of the previous break point which has had its original code restored by the Kill Break Point Command.

SYNTAX:

```
G                    - Go from the last break point.
                       Go from break point also asks the
                       user if they want to wait for
                       another break point(see 3.2.9).
```

EXAMPLE:

```
>F02AP               - Place a break point at $F02A.

>F000G               - Go from location $F000.

     *DO YOU WISH TO WAIT FOR A BREAK POINT? Y
     *WAITING FOR BREAK POINT*
     *BREAK POINT ENCOUNTERED*

>F100P               - Place a new break point at $F100.

>G                   - Go from location $F02A, which has
                       now had its original values
                       restored by the return from
                       break point.

     *DO YOU WISH TO WAIT FOR A BREAK POINT? Y
     *WAITING FOR BREAK POINT*
     *BREAK POINT ENCOUNTERED*

>                    - You are now at location $F100.
```

NOTES:

1) If the user issues a G command before setting a break point AMON will complain with an error message.
2) If the user issues a G command after setting the first break point but before returning from it, AMON will also complain with an error message.

6.2.9 Place Break Point

PLACE BREAK POINT allows the user to set a break point in the code in the VCS memory space.

SYNTAX:

addrP	- Place a break point at address "addr".

EXAMPLE:

>F454P	- Places a break point at location $F454.

NOTES:

1) Setting a break point saves the 3 bytes of code that were at that location into a "safe" place in the monitor.
2) There can only be one break point set at a time.

6.2.10 Kill Break Point

KILL A BREAK POINT allows the user to remove a break point and to replace it with the original code at that location.

SYNTAX:

K	- Kills the last break point.

EXAMPLE:

>K	- Kills the last break point.

NOTES:

1) The monitor remembers if a break point has been set and where in memory it was. If the user tries to remove a break point when one has not yet been set, an error message will appear.
2) The monitor always performs a K command when it returns from a break point(see section 3.2.9).
3) If the user places a break point with a P command and then desires to change it and set another before executing any code, the user should perform a K command to erase the first break point.

6.2.11 Start

START allows the user to start the target software from its reset interrupt vector location. This command is used mainly when FMON is not being used. It turns the VCS ROM memory over to the VCS - WITHOUT THE BI-DIRECTIONAL PORT ENABLED - and then prompts the user to throw the power switch of the VCS on. This causes the VCS to start executing its code at the address contained in the reset interrupt vector at location $FFFC in the VCS 6507 address space.

The user may stop execution of the VCS software at any time by either issuing a HALT command from AMON, or by issuing a command that causes AMON to take back control of the VCS ROM memory such as the READ MEMORY command.

SYNTAX:

S - Gives control to the VCS

EXAMPLE:

```
>S

    * TURN THE VCS ON *

>
```

6.2.12 Halt

AMON can easily stop the execution of the target software in the VCS by retaking control of the FROB card memory. This may be done at any time. If FMON was loaded the user will need to reach over and power up the VCS so that FMON will restart itself after the user caused a HALT.

SYNTAX:

H - Halt the VCS now.

EXAMPLE:

>H - Halts execution of the VCS and then prompts the user for the next command.

6.2.13 Exit AMON

The user may exit AMON by typing a control C (^C) character.

CHAPTER 7

FROB Tool Kit

The FROB TOOL KIT is a collection of utility programs to be used with AMON and FMON to complete the development system software for the FROB system.

7.1 Theory Of The Tool Kit

Each of the utilities in the TOOL KIT has a specific function as described in the sections below. They have certain syntax conventions in common. Each of the utilities first asks which slot the FROB card is in. This allows the user to place the FROB card in any convenient slot. In all cases, the utilities expect their input in HEX digits rather than decimal digits.

7.2 FROB Load

FLOAD allows the user to load the FROB memory with the data contained in an Apple disk file or to move data from Apple memory into FROB memory.

7.2.1 Summary

FLOAD has two main modes: disk to FROB or Apple memory to FROB. After asking the slot question, FLOAD asks the question: "ENTER FILE NAME". If the user answers this question with the special filename "MEMORY" then FLOAD will assume this is a load from Apple memory to FROB memory as described in section 7.2.3. If the user answered the filename question with anything other than the key name "MEMORY", FLOAD will assume this is a load from Apple disk to FROB memory as described in section 7.2.2.

7.2.2 Load From Disk

If the user answers the filename question with a filename then FLOAD asks the question: "READY FOR DISK LOAD(Y/N)?". This allows the user time to change disks if desired. If the user answers this question with a "N" then FLOAD will quit without doing anything. If the user answers this question with a "Y", then FLOAD will load the file into its work buffer.

NOTE: ALL FILES FOR FLOAD MUST BE IN APPLE DOS "BLOAD" FORMAT.

After the file is loaded into the work buffer then FLOAD will ask the question: "IS IT A 4K FILE(Y/N)?". If the user answers this question with a "Y", then FLOAD will start loading the file into FROB memory in the first location of page Ø. It will load 4K bytes from its buffer and thus load all 16 pages. This corresponds to locations $FØØØ through $FFFF in the VCS 65Ø7 address space. If the file is longer than 4K bytes then only the first 4K bytes will be loaded. If the file was is than 4K then whatever data is in the remaining bytes of the FLOAD work buffer will be loaded into the FROB memory.

If the user answers the "IS IT A 4K FILE(Y/N)?" question with a "N" then FLOAD will assume it is a 2K file. It will then ask the user "LOAD TO HIGH OR LOW MEMORY(H/L)?". If the user answers this question with a "L", then FLOAD will load the first 2K bytes in the work buffer into pages Ø through 7 of the FROB. This corresponds to locations $FØØØ through $F7FF in the VCS 65Ø7 address space. If the file is more than 2K bytes long, only the first 2K bytes will be loaded. If the file is less than 2K bytes long, then whatever data was in the end of the FLOAD work buffer will be loaded up to the 2K boundary.

If the user answers the "LOAD TO HIGH OR LOW MEMORY(H/L)?" question with a "H", then FLOAD will load the first 2K bytes in the work buffer into pages 8 - F of the FROB. This corresponds to locations $F8ØØ through $FFFF in the VCS 65Ø7 address space. If the file is more than 2K bytes long, only the first 2K bytes will be loaded. If the file is less than 2K bytes long, then whatever data was in the end of the FLOAD work buffer will be loaded up to the 2K boundary.

7.2.3 Load From Memory

If the user answers the "ENTER FILE NAME" question with the key name "MEMORY" then FLOAD will assume the user wants to move data from Apple memory to FROB memory. FLOAD will then ask the user for three data values; it will assume all three are hex digits:

"START MOVING FROM Apple ADDRESS?"	(up to 4 digits)
"MOVE TO FROB PAGE?"	(1 hex digit)
"HOW MANY PAGES?"	(1 hex digit)

FLOAD will move data in 256 byte blocks. The Apple starting address must be on an even page boundary (i.e. $1ØØØ, $12ØØ, etc.). If the user asks FLOAD to move 2 pages to FROB page "F" or any other such error, FLOAD will complain.

7.3 FROB Save

FSAVE allows the user to save data in the FROB memory in an Apple disk file or move data to Apple memory from the FROB memory.

7.3.1 Summary

FSAVE purpose is to save program images in the FROB to an Apple disk.

7.3.2 Save To Disk

The user is first prompted for a file name. This must be a legal Apple DOS file name. When the user has answered with a file name, then FSAVE will ask the question "READY FOR DISK SAVE(Y/N)?". This allows the user time to change disks if desired. If the user answers this question with a "N", then FSAVE will quit without doing anything.

NOTE: FSAVE WILL ALWAYS CREATE AN Apple DOS "BLOAD" FORMAT FILE.

CHAPTER 8

The EXPLORER

8.1 Theory Of The Explorer

The EXPLORER is a software laboratory tool. It allows the user to conduct experiments with the video and audio functions of the VCS. It is divided into three parts; XCONTROL, EXPLORER and EXPLOR. EXPLORER is the program that resides in the VCS. It acts as a real time monitor of the VCS and accepts commands from the Apple side. XCONTROL is the program that resides in the Apple. EXPLOR is an Applesoft program that loads EXPLORER into the FROB and XCONTROL into the Apple.

XCONTROL maintains a screen menu that displays the current contents of various hardware registers within the VCS. These are divided into three categories; EXPLORER software registers, VCS control registers, and VCS read-only registers. These are discussed in section 8.3.

The user can move a cursor around the various fields in the XCONTROL menu and insert new HEX data in those fields that are not read-only. This will cause XCONTROL to send a message to EXPLORER which will then update the corresponding register in the VCS.

By using EXPLORER the user may change various values in the VCS registers and observe the resulting effects. This provides a software laboratory in which to conduct experiments for special effects that may be then used in target software.

8.2 EXPLORER Operation

To start the EXPLORER system, run EXPLOR.

```
RUN EXPLOR
```

EXPLOR will then ask which slot the FROB is in. EXPLOR will load EXPLORER into the FROB and ask the user to turn the VCS on. It then loads XCONTROL into the Apple and starts it.

XCONTROL displays its menu screen and places the cursor into the upper left hand corner of the screen. The user may move the cursor with the right and left arrow keys and the ";" and "/" keys.

CURSOR MOVMENT:

KEY	MOVEMENT
->	RIGHT
<-	LEFT
;	UP
/	DOWN

XCONTROL only allows the user to insert data within a valid two character data field. If the user attempts to write elsewhere on the screen nothing will happen with the exception that the cursor may disappear until the user moves the cursor again.

NOTE: ALL INPUT TO XCONTROL MUST BE A VALID HEX DIGIT (Ø-F). IF THE USER ENTERS A NON-HEX CHARACTER XCONTROL WILL IGNORE IT.

8.3 Explorer Registers

There are three classes of registers on the XCONTROL menu: EXPLORER software control registers, VCS hardware control registers, and VCS read-only registers.

8.3.1 Software Control Registers

There are several EXPLORER software control registers that allow the user to change the operation of the EXPLORER software in the VCS. These control the shape of Player 1, Player 2, A object, B object, and C object images. They control the vertical position of all of the above objects on the screen. The most important of these is the Enable register. The Enable register is the the first register in the upper left hand corner of the XCONTROL menu. The Enable register allows the user to select which of the 5 main objects to display on the screen at any one time. If the user sets the Enable register to display all 5 it will cause EXPLORER to lose vertical sync because it is too much to display at one time.

ENABLE REGISTER CONTROL BITS:

BIT Ø	SELECT PLAYER ONE
BIT 1	SELECT PLAYER TWO
BIT 2	SELECT OBJECT A
BIT 3	SELECT OBJECT B
BIT 4	SELECT OBJECT C
BIT 5-6	NOT USED
BIT 7	DISPLAY CONTROL, BIT 7 = Ø causes object display every screen BIT 7 = 1 causes object display every other screen

EXAMPLE:

WRITING A $Ø1 TO THE ENABLE REG DISPLAYS PLAYER 1 ONLY

WRITING A $ØC TO THE ENABLE REG DISPLAYS THE A & B OBJECTS

WRITING A $82 TO THE ENABLE REG DISPLAYS PLAYER 2 EVERY OTHER SCREEN

8.3.2 VCS Hardware Control Registers

The VCS hardware registers map directly to the hardware registers on the VCS. Writing a value to the Player 1 horizontal register causes XCONTROL to send a message to EXPLORER which then writes the new value into this register. Not all of the hardware registers are available on the XCONTROL screen. XCONTROL has a general write/read register in the center of the screen that allows the user to write to any address within the VCS and/or to read its contents. When XCONTROL starts, this register is pointing to location $81 in RAM which is used by EXPLORER to hold a timer that is incremented every second.

NOTE: THE READ VALUE IN THE XCONTROL GENERAL READ/WRITE REGISTER IS ONLY VALID WHEN THE CURSOR IS NOT IN THE ADDRESS FIELD OF THIS REGISTER.

8.3.3 VCS Read-Only Hardware Registers

For almost all of the read-only VCS hardware registers only the top bit (7) is significant. XCONTROL will not allow the user to write into them. They provide various status bits of information such as collision detection. Bits 1 - 6 have a tendency to "Float" or to change their value randomly. This appears as changing values on the XCONTROL display screen.

For a more detailed view of the EXPLORER system the user is advised to study the source code supplied with the FROB system.

```
8
]LIST

1  HIMEM: 32767
2  HOME : PRINT "FROB MONITOR PROGRAM"
3  PRINT : REM  VERSION 1.1
4  DIM DIG%(4),S%(32),BP%(3),PG%(18),BIT%(8)
5 FM% = 0
6  DIM OCODE%(255),ACODE%(255)
7 PM% = 768:BUF$ = "8":BUF = 8 * 4096
8  PRINT  CHR$ (4);"BLOAD PMOVE.OBJ"
10  INPUT "WHAT IS THE FROB SLOT NUMBER (1,7)?";FSLOT%
12  IF FSLOT% < 1 OR FSLOT% > 7 THEN 10
14 DEV = 4096 * 12 + 16 * (FSLOT% + 8)
20  INPUT "DO YOU WISH TO LOAD A FILE? (Y/N)";A$
22  IF A$ = "N" THEN 30
24  IF A$ <  > "Y" THEN 20
26  GOSUB 11000: REM  GO LOAD A FILE
30  INPUT "DO YOU WISH TO USE FMON? (Y/N)";A$
32  IF A$ = "N" THEN 40
34  IF A$ <  > "Y" THEN 30
36  GOSUB 12000: REM  GO LOAD FMON
40 BP% = 0: REM    SET TO 1 ON BREAK POINT HIT
60  GOSUB 4000: REM  LOAD THE SHADOW
100  IF FM% = 0 THEN 200
110  PRINT "PLEASE TURN ON VCS AND HIT RETURN";
120  INPUT A$
130  IF  PEEK (DEV) < 128 THEN 110
140  POKE DEV + 1,0
150  IF  PEEK (DEV) < 128 THEN 110
200 OPCODES$ = "***ADCANDASLBCCBCSBEQBITBMIBNEBPLBRKBVCBVSCLCCLDCLICLVCMPCPXCPY"
201 OPCODES$ = OPCODES$ + "DECDEXDEYEORINCINXINYJMPJSRLDALDXLDYLSRNOPORAPHAPHPPLA"
202 OPCODES$ = OPCODES$ + "PLPROLRORRTIRTSSBCSECSEDSEISTASTXSTYTAXTAYTSXTXATXSTYA"
210  FOR I = 0 TO 255: READ OCODE%(I): NEXT I
211  DATA  11,35,0,0,0,35,3,0
212  DATA  37,35,3,0,0,35,3,0
213  DATA  10,35,0,0,0,35,3,0
214  DATA  14,35,0,0,0,35,3,0
215  DATA  29,2,0,0,7,2,40,0
216  DATA  39,2,40,0,7,2,40,0
217  DATA  8,2,0,0,0,2,40,0
218  DATA  45,2,0,0,0,2,40,0
219  DATA  42,24,0,0,0,24,33,0
220  DATA  36,24,33,0,28,24,33,0
221  DATA  12,24,0,0,0,24,33,0
222  DATA  16,24,0,0,0,24,33,0
223  DATA  43,1,0,0,0,1,41,0
224  DATA  38,1,41,0,28,1,41,0
225  DATA  13,1,0,0,0,1,41,0
226  DATA  47,1,0,0,0,1,41,0
227  DATA  0,48,0,0,50,48,49,0
228  DATA  23,0,54,0,50,48,49,0
229  DATA  4,48,0,0,50,48,49,0
230  DATA  56,48,55,0,0,48,0,0
231  DATA  32,30,31,0,32,30,31,0
232  DATA  52,30,51,0,32,30,31,0
233  DATA  5,30,0,0,32,30,31,0
234  DATA  17,30,53,0,32,30,31,0
235  DATA  20,18,0,0,20,18,21,0
236  DATA  27,18,22,0,20,18,21,0
```

```
237  DATA  9,18,0,0,0,18,21,0
238  DATA  15,18,0,0,0,18,21,0
239  DATA  19,44,0,0,19,44,25,0
240  DATA  26,44,34,0,19,44,25,0
241  DATA  6,44,0,0,0,44,25,0
242  DATA  46,44,0,0,0,44,25,0
250  FOR I = 0 TO 255: READ ACODE%(I): NEXT I
260  DATA  1,10,1,1,1,3,3,1
261  DATA  1,5,2,1,1,4,4,1
262  DATA  13,11,1,1,1,6,6,1
263  DATA  1,9,1,1,1,8,8,1
264  DATA  4,10,1,1,3,3,3,1
265  DATA  1,5,2,1,4,4,4,1
266  DATA  13,11,1,1,1,6,6,1
267  DATA  1,9,1,1,1,8,8,1
268  DATA  1,10,1,1,1,3,3,1
269  DATA  1,5,2,1,4,4,4,1
270  DATA  13,11,1,1,1,6,6,1
271  DATA  1,9,1,1,1,8,8,1
272  DATA  1,10,1,1,1,3,3,1
273  DATA  1,5,2,1,12,4,4,1
274  DATA  13,11,1,1,1,6,6,1
275  DATA  1,9,1,1,1,8,8,1
276  DATA  1,10,1,1,3,3,3,1
277  DATA  1,1,1,1,4,4,4,1
278  DATA  13,11,1,1,6,6,7,1
279  DATA  1,9,1,1,1,8,1,1
280  DATA  5,10,5,1,3,3,3,1
281  DATA  1,5,1,1,4,4,4,1
282  DATA  13,11,1,1,6,6,7,1
283  DATA  1,9,1,1,8,8,9,1
284  DATA  5,10,1,1,3,3,3,1
285  DATA  1,5,1,1,4,4,4,1
286  DATA  13,11,1,1,1,6,6,1
287  DATA  1,9,1,1,1,8,8,1
288  DATA  5,10,1,1,3,3,3,1
289  DATA  1,5,1,1,4,4,4,1
290  DATA  13,11,1,1,1,6,6,1
291  DATA  1,9,1,1,1,8,8,1
400  FOR I = 1 TO 18: READ PG%(I): NEXT I
401  DATA   198,241,208,252
402  DATA   198,240,208,248
403  DATA   198,239,208,244
404  DATA   76,99,255,2,0,0
490 PC = 1000
500 FAIL% = 0
510  GOSUB 1000
520  IF  LEN (B$) > 4 THEN 900
540  IF  LEN (C$) > 0 THEN 600
550  IF  LEN (B$) = 0 THEN 500
560 D$ = "1"
570  GOTO 20000: REM  THE READ MEM ROUTINE
600  REM  DELIM FIELD NOT BLANK
610  IF C$ = "," THEN 20000: REM  READ MEMORY
620  IF C$ = "<" THEN 21000: REM  WRITE MEMORY
630  IF C$ = "M" THEN 22000: REM  MODIFY REGISTER
640  IF C$ = "R" THEN 23000: REM  DISPLAY REGISTERS
```

```
650  IF C$ = "P" THEN 24000: REM  PLACE BREAKPOINT
660  IF C$ = "K" THEN 25000: REM  KILL BREAK POINT
670  IF C$ = "H" THEN 26000: REM  HALT THE VCS
675  IF C$ = "S" THEN 31000: REM  START VCS AT RESET
680  IF C$ = "G" AND  LEN (B$) = 0 THEN 27000: REM   GO FROM BREAK
690  IF C$ = "G" THEN 28000: REM  GOTO ADDR
700  IF C$ = "L" AND  LEN (B$) = 0 THEN 29000: REM  CONTINUE DISASM
710  IF C$ = "L" THEN 30000: REM  DISASM
750  PRINT "NOT A VALID COMMAND"
760  GOTO 500
900  PRINT "ERROR"
910  GOTO 500
1000  REM  ROUTINE TO RETURN STRING
1005 B$ = "":C$ = "":D$ = ""
1010  INPUT ">";A$
1015 K% = 1
1020  FOR L = 1 TO  LEN (A$)
1025 X$ =  MID$ (A$,L,1)
1027  IF X$ = " " THEN 1100
1030  GOSUB 2000: REM  IS THIS A HEX DIG?
1031  IF FLAG% > 0 AND K% = 2 THEN K% = 3
1032  IF FLAG% = 0 THEN K% = K% + 1
1033  IF K% > 3 THEN 1100
1035  IF K% = 1 THEN B$ = B$ + X$
1040  IF K% = 2 THEN C$ = C$ + X$
1050  IF K% = 3 THEN D$ = D$ + X$
1100  NEXT L
1110  RETURN
2000  REM  RETURN FLAG% SET IF X$ IS HEX
2005 FLAG% = 1
2010  IF X$ < "0" OR X$ > "F" THEN FLAG% = 0
2020  IF X$ > "9" AND X$ < "A" THEN FLAG% = 0
2100  RETURN
2500  REM   CHANGE B$ TO HEX ADDR
2505 ADDR = 0
2510  FOR L = 1 TO  LEN (B$)
2520 X% =  ASC ( MID$ (B$,L,1)) - 48
2540  IF X% > 9 THEN X% = X% - 7
2550 ADDR = ADDR * 16 + X%
2560  NEXT L
2570  RETURN
2630  GOTO 500
3000  REM  ROUTINE TO TALK TO FMON
3100  IF  PEEK (DEV) < 128 THEN 3100
3110  POKE DEV + 1,WR%
3120  RETURN
3200  IF  PEEK (DEV) < 196 THEN 3200
3210 X% =  PEEK (DEV + 1)
3220  RETURN
3300  REM  ROUTINE TO GET REGISTERS
3310  GOSUB 3200: POKE BUF + 4086,X%
3311  GOSUB 3200: POKE BUF + 4085,X%
3312  GOSUB 3200: POKE BUF + 4087,X%
3313  GOSUB 3200: POKE BUF + 4088,X%
3314  GOSUB 3200: POKE BUF + 4089,X%
3320  FOR I = 1 TO 32
3330  GOSUB 3200:S%(I) = X%
```

```
3340  NEXT I
3345  GOSUB 3200
3350 NZ% = 0: IF  PEEK (BUF + 4085) = 128 THEN NZ% = 1
3360 ZZ% = 1: IF NZ% = 1 THEN ZZ% = 0
3370  IF  PEEK (BUF + 4085) = 1 THEN ZZ% = 0
3380  GOSUB 13000: REM  TAKE FLAGS APART
3385  POKE BUF + 4084,NZ% * 128 + BITZ%(2) * 64 + BITZ%(3) * 32 + BITZ%(4) * 16 + BITZ%(5)
                                                    * 8 + BITZ%(6) * 4 + ZZ% * 2 + BITZ%(8)
3390  RETURN
3400  REM  ROUTINE TO RESTORE REGISTERS
3410  FOR I = 32 TO 1 STEP  - 1
3420 WRZ% = SZ%(I): GOSUB 3000
3430  NEXT I
3440  RETURN
4000  REM  LOAD SHADOW TO FROB
4010  POKE PMZ% + 18,FSLOTZ%
4020  POKE PMZ% + 19, INT (BUF / 256)
4030  POKE PMZ% + 20,16
4040  POKE PMZ% + 21,32
4050  CALL PMZ% + 22
4060  POKE DEV,48
4100  RETURN
5000  REM  CONVERT D$ TO LENGTH FIELD
5010 NUMBYTZ% = 0
5020  IF  LEN (D$) = 0 THEN  RETURN
5030  IF  LEN (D$) > 3 THEN D$ =  RIGHT$ (D$,3)
5040  FOR L = 1 TO  LEN (D$)
5050 XZ% =  ASC ( MID$ (D$,L,1)) - 48
5060  IF XZ% > 9 THEN XZ% = XZ% - 7
5070 NUMBYTZ% = NUMBYTZ% * 16 + XZ%
5080  NEXT L
5100  RETURN
6000  REM    PRINT LOWER 3 HEX DIGITS OF A1
6010 NZ% = 3
6020  FOR I = 1 TO NZ%
6030 DIGZ%(I) = A1 - ( INT (A1 / 16) * 16)
6040 A1 = A1 / 16
6050  NEXT I
6060  FOR I = NZ% TO 1 STEP  - 1
6070 KZ% = DIGZ%(I)
6080  IF KZ% > 9 THEN KZ% = KZ% + 7
6090  PRINT  CHR$ (KZ% + 48);
6100  NEXT I
6200  RETURN
6500  REM   PRINT 2 HEX DIGITS FROM A1
6510 NZ% = 2
6520  GOTO 6020
7000  REM  FMON READ
7010 WRZ% = 16: GOSUB 3000
7020 WRZ% = ADDR / 256: GOSUB 3000
7030 WRZ% = ADDR - (WRZ% * 256): GOSUB 3000
7040 WRZ% = 1: GOSUB 3000
7050  GOSUB 3200
7060 A1 = XZ%
7070  RETURN
8000  REM  DO AN FMON WRITE
8010 WRZ% = 32: GOSUB 3000
8020 WRZ% = ADDR / 256: GOSUB 3000
```

```
8030 WR% = ADDR - (WR% * 256): GOSUB 3000
8040 WR% = 1: GOSUB 3000
8050 WR% = A1: GOSUB 3000
8100  RETURN
9000  REM  ROUTINE TO PUT TO SLEEP
9010 WR% = 32: GOSUB 3000
9020 WR% = 0: GOSUB 3000
9030 WR% = 224: GOSUB 3000
9040 WR% = 18: GOSUB 3000
9045  FOR I = 1 TO 18
9050 WR% = PG%(I): GOSUB 3000
9060  NEXT I
9062 WR% = 64: GOSUB 3000
9064 WR% = 0: GOSUB 3000
9066 WR% = 224: GOSUB 3000
9080  GOSUB 4000: REM  LOAD SHADOW
9100  RETURN
11000  REM   LOAD A FILE
11010  INPUT "WHAT IS THE FILE NAME?";A$
11020  INPUT "LOAD TO WHAT VCS HEX ADDRESS?";B$
11030  IF  LEN (B$) <  > 4 THEN 11020
11031 X$ =  LEFT$ (B$,1)
11032  IF X$ = "2" OR X$ = "4" OR X$ = "6" OR X$ = "0" OR X$ = "8" OR X$ = "A" OR X$ =
11035 OK% = 1                                                          "C" OR X$ = "E" THEN 11020
11040  FOR L = 1 TO 4
11050 X$ =  MID$ (B$,L,1)
11060  GOSUB 2000: REM  SEE IF A HEX DIGIT
11070  IF FLAG% = 0 THEN OK% = 0
11080  NEXT L
11090  IF OK% = 0 THEN 11020
11100  PRINT  CHR$ (4);"BLOAD ";A$;",A$";BUF$; RIGHT$ (B$,3
11110  RETURN
12000  REM : LOAD FMON
12010  PRINT  CHR$ (4);"BLOAD FMON.OBJ,A$";BUF$;"F11"
12020 FM% = 1
12030  RETURN
13000  REM  CONVERT X% TO BITS
13005 X1% = X%:XB% = 256
13010  FOR I = 1 TO 8
13020 BIT%(I) = 0:XB% = XB% / 2
13030  IF X1% < XB% THEN 13060
13040 BIT%(I) = 1
13050 X1% = X1% - XB%
13060  NEXT I
13070  RETURN
14000  REM  PRINT DADDR AND ":"
14010 X = DADDR: GOSUB 14300
14040  PRINT ":";
14050  RETURN
14100  REM  PRINT 2 HEX FROM X%
14110 A1 = X%
14120  GOTO 6500
14200  REM  PRINT OPCODE
14210 N% = OCODE%(OP%) * 3 + 1
14220  PRINT  MID$ (OPCODES$,N%,3);
14230  RETURN
14300  REM  PRINT DADDR RELATIVE
```

```
14310  PRINT FHR$;
14320 A1 = X: GOSUB 6000
14330  RETURN
20000  REM  READ MEMORY
20002 RD% = 1: REM  FOLLOW THE READ PATH
20005  GOSUB 5000: REM  CONVERT D$ TO NUMBER
20010  IF  LEN (B$) = 4 THEN 20050
20020  FOR L = 1 TO 4 -  LEN (B$)
20030 B$ = "0" + B$
20040  NEXT L
20050 X$ =  LEFT$ (B$,1)
20055 FHR$ = X$
20060  IF X$ = "0" OR X$ = "2" OR X$ = "4" OR X$ = "6" OR X$ = "8" OR X$ = "A" OR X$ =
20070 B$ = BUF$ +  RIGHT$ (B$,3)                                        "C" OR X$ = "E" THEN 20100
20080  GOTO 20200
20100 B$ = "0" +  RIGHT$ (B$,3)
20110  IF FM% > 0 THEN 20200
20120  PRINT "ERROR -- FMON NOT LOADED"
20130  GOTO 500
20200  GOSUB 2500: REM  CONVERT ADDRESS
20201  IF RD% = 0 THEN 21100: REM  DO WRITE
20203  PRINT
20205  PRINT FHR$;
20210 A1 = ADDR: GOSUB 6000
20212  PRINT ":";
20220  IF ADDR > 4095 THEN A1 =  PEEK (ADDR)
20225  IF ADDR < 256 THEN 20600
20230  IF ADDR < 4096 THEN  GOSUB 7000: REM     GO DO FMON READ
20240  GOSUB 6500
20250  PRINT " ";
20260 ADDR = ADDR + 1:NUMBYT% = NUMBYT% - 1
20270  IF NUMBYT% = 0 THEN 20500
20280 B% = ADDR - ( INT (ADDR / 8) * 8)
20290  IF B% = 0 THEN 20203
20300  GOTO 20220
20500  PRINT : PRINT
20510  GOTO 500
20600  IF ADDR < 224 THEN 20230
20610 A1 = S%(ADDR - 223)
20620  GOTO 20240
21000  REM  WRITE MEMORY
21010 RD% = 0: REM  CHOOSE WRITE PATH
21020  GOTO 20005
21100  IF NUMBYT% > 255 THEN 900
21110 A1 = NUMBYT%
21120  IF ADDR > 4095 THEN  POKE ADDR,A1
21125  IF ADDR < 256 THEN 21200
21130  IF ADDR < 4096 THEN  GOSUB 8000
21140 NUMBYT% = 1
21150  GOTO 20203
21200  IF ADDR < 224 THEN 21130
21210 S%(ADDR - 223) = A1
21220  GOTO 21140
22000  REM  MODIFY REGISTER
22010  INPUT "WHICH REGISTER? (A,X,Y,S,P)";A$
22020  IF A$ < > "A" AND A$ < > "X" AND A$ < > "Y" AND A$ < > "S" AND A$ < > "P" THEN 22010
22030  INPUT "NEW VALUE (HEX)=";D$
```

```
22040  IF  LEN (D$) > 2 THEN 22030
22050 A1 = 0:FLAGZ = 1
22060  FOR I = 1 TO  LEN (D$)
22065  IF FLAGZ = 0 THEN 22100
22070 X$ =  MID$ (D$,I,1): GOSUB 2000
22080 XZ =  ASC (X$) - 48: IF XZ > 9 THEN XZ = XZ - 7
22090 A1 = A1 * 16 + XZ
22100  NEXT I
22110  IF FLAGZ = 0 THEN 22030
22120  IF A$ < > "A" THEN 22130
22125  POKE BUF + 4086,A1: GOTO 500
22130  IF A$ < > "X" THEN 22140
22135  POKE BUF + 4087,A1: GOTO 500
22140  IF A$ < > "Y" THEN 22150
22145  POKE BUF + 4088,A1: GOTO 500
22150  IF A$ < > "S" THEN 22160
22155  POKE BUF + 4089,A1: GOTO 500
22160 FZ = A1:FZ = FZ / 2
22165 ZZ = FZ - ( INT (FZ / 2) * 2)
22170 NZ =  INT (FZ / 64)
22175  IF ZZ = 0 OR NZ = 0 THEN 22200
22180  PRINT "??? EITHER Z OR N OR BOTH MUST BE 0"
22190  GOTO 22030
22200  POKE BUF + 4084,A1
22210  IF ZZ = 0 THEN 22250
22220  POKE BUF + 4085,255
22230  GOTO 500
22250  IF NZ THEN 22280
22260  POKE BUF + 4085,1
22270  GOTO 500
22280  POKE BUF + 4085,128
22290  GOTO 500
23000  REM  DISPLAY REGISTERS
23010  PRINT
23020  PRINT "A=";:A1 =  PEEK (BUF + 4086): GOSUB 6500
23030  PRINT " X=";:A1 =  PEEK (BUF + 4087): GOSUB 6500
23040  PRINT " Y=";:A1 =  PEEK (BUF + 4088): GOSUB 6500
23050  PRINT " S=";:A1 =  PEEK (BUF + 4089): GOSUB 6500
23060  PRINT " P=";:A1 =  PEEK (BUF + 4084): GOSUB 6500
23070  PRINT : GOTO 500
24000  REM  PLACE BREAK POINT
24010  IF  LEN (B$) < > 4 THEN 900
24020 B$ = BUF$ +  RIGHT$ (B$,3)
24030  GOSUB 2500
24040 BREAK = ADDR
24050  FOR I = 1 TO 3
24060 BPZ(I) =  PEEK (ADDR + I - 1)
24070  NEXT I
24080  POKE ADDR,76: POKE ADDR + 1,17: POKE ADDR + 2,255
24090  GOTO 500
25000  REM  KILL BREAK POINT
25010  IF BREAK = 0 THEN 500
25020  FOR I = 1 TO 3
25030  POKE BREAK + I - 1,BPZ(I)
25040  NEXT I
25050 BREAK = 0: GOTO 500
26000  REM  HALT THE VCS
```

```
26010  POKE DEV,0
26020  PRINT "FROB MEMORY NOW UNDER APPLE CONTROL"
26030  GOTO 500
27000  REM   GO FROM BREAK
27003  IF PC = 1000 THEN 27200
27005  IF PC = BREAK - BUF + 61440 THEN 27100
27006  INPUT "DO YOU WISH TO WAIT FOR A BREAK? (Y/N)";A$
27007  IF A$ < > "Y" AND A$ < > "N" THEN 27006
27010  POKE BUF + 4000,PC - ( INT (PC / 256) * 256)
27015  POKE BUF + 4001, INT (PC / 256)
27020  GOSUB 9000: REM  SLEEP AND SET UP
27030 WRX = 48: GOSUB 3000
27040  GOSUB 3400: REM  RESTORE REGISTERS
27042  IF A$ = "N" THEN 500
27045  PRINT "WAITING FOR BREAK POINT"
27050  GOSUB 3300: REM   WAIT FOR BREAK
27052  PRINT "BREAK POINT ENCOUNTERED"
27055 PC = BREAK - BUF + 61440
27060  GOTO 25000
27100  PRINT "ERROR -- YOU MUST MOVE THE BREAKPOINT"
27110  PRINT "BEFORE YOU CONTINUE"
27120  GOTO 500
27200  PRINT "ERROR -- NO BREAK POINT FROM"
27210  PRINT "WHICH TO PROCEED"
27220  GOTO 500
28000  REM  GOTO ADDR
28010  GOSUB 2500
28020 PC = ADDR
28030  GOTO 27006
29000  REM  CONTINUE DISASM
29005 DLIM = DADDR + 30
29010 OP% =  PEEK (DADDR)
29015  GOSUB 14000: REM   PRINT DADDR:
29016 DADDR = DADDR + 1
29017 X% = OP%: GOSUB 14100: PRINT " ";
29020  ON ACODE%(OP%) GOTO 29100,29150,29200,29250,29300,29350,29400,29450,29500,29550,29600,
29100  REM  IMPLIED (NO ADDRESS)                                                    29650,29700
29110  PRINT "      ";
29120  GOSUB 14200: REM  PRINT OP STRING
29130  GOTO 29900
29150  REM  ACCUMULATOR
29155  PRINT "      ";
29160  GOSUB 14200
29165  PRINT " A";
29170  GOTO 29900
29200  REM  PAGE 0 ADDRESS
29205 X1% =  PEEK (DADDR):DADDR = DADDR + 1
29210 X% = X1%: GOSUB 14100: PRINT " ";
29215  PRINT "   ";
29220  GOSUB 14200: PRINT " $";
29225 X% = X1%: GOSUB 14100
29230  GOTO 29900
29250  REM  16 BIT ABSOLUTE
29252 X1% =  PEEK (DADDR):DADDR = DADDR + 1
29254 X% = X1%: GOSUB 14100: PRINT " ";
29256 X2% =  PEEK (DADDR):DADDR = DADDR + 1
29258 X% = X2%: GOSUB 14100: PRINT " ";
```

```
29260  GOSUB 14200: PRINT " $";
29262 X% = X2%: GOSUB 14100
29264 X% = X1%: GOSUB 14100
29270  GOTO 29900
29300  REM  IMMEDIATE
29305 X1% =  PEEK (DADDR):DADDR = DADDR + 1
29310 X% = X1%: GOSUB 14100: PRINT " ";
29315  PRINT "   ";
29320  GOSUB 14200: PRINT " #$";
29325 X% = X1%: GOSUB 14100
29330  GOTO 29900
29350  REM  ZERO PAGE INDEXED X
29355 X1% =  PEEK (DADDR):DADDR = DADDR + 1
29360 X% = X1%: GOSUB 14100: PRINT "    ";
29365  GOSUB 14200: PRINT " $";
29370 X% = X1%: GOSUB 14100: PRINT ",X";
29375  GOTO 29900
29400  REM  ZERO PAGE INDEXED Y
29405 X1% =  PEEK (DADDR):DADDR = DADDR + 1
29410 X% = X1%: GOSUB 14100: PRINT "    ";
29415  GOSUB 14200: PRINT " $";
29420 X% = X1%: GOSUB 14100: PRINT ",Y";
29425  GOTO 29900
29450  REM  ABSOLUTE INDEXED X
29455 X1% =  PEEK (DADDR):DADDR = DADDR + 1
29460 X% = X1%: GOSUB 14100: PRINT " ";
29465 X2% =  PEEK (DADDR):DADDR = DADDR + 1
29470 X% = X2%: GOSUB 14100: PRINT " ";
29475  GOSUB 14200: PRINT " $";
29480 X% = X2%: GOSUB 14100
29485 X% = X1%: GOSUB 14100
29490  PRINT ",X";: GOTO 29900
29500  REM  ABSOLUTE INDEXED Y
29505 X1% =  PEEK (DADDR):DADDR = DADDR + 1
29510 X% = X1%: GOSUB 14100: PRINT " ";
29515 X2% =  PEEK (DADDR):DADDR = DADDR + 1
29520 X% = X2%: GOSUB 14100: PRINT " ";
29525  GOSUB 14200: PRINT " $";
29530 X% = X2%: GOSUB 14100
29535 X% = X1%: GOSUB 14100
29540  PRINT ",Y";: GOTO 29900
29550  REM  INDEXED INDIRECT X
29555 X1% =  PEEK (DADDR):DADDR = DADDR + 1
29560 X% = X1%: GOSUB 14100: PRINT "    ";
29565  GOSUB 14200: PRINT " ($";
29570 X% = X1%: GOSUB 14100: PRINT ",X)";
29575  GOTO 29900
29600  REM  INDIRECT INDEXED Y
29605 X1% =  PEEK (DADDR):DADDR = DADDR + 1
29610 X% = X1%: GOSUB 14100: PRINT "    ";
29615  GOSUB 14200: PRINT " ($";
29620 X% = X1%: GOSUB 14100: PRINT "),Y";
29625  GOTO 29900
29650  REM  JUMP INDIRECT
29655 X1% =  PEEK (DADDR):DADDR = DADDR + 1
29660 X% = X1%: GOSUB 14100: PRINT " ";
29661 X2% =  PEEK (DADDR):DADDR = DADDR + 1
```

```
29662 X% = X2%: GOSUB 14100: PRINT " ";
29665  GOSUB 14200: PRINT " ($";
29670 X% = X2%: GOSUB 14100
29671 X% = X1%: GOSUB 14100: PRINT ")";
29675  GOTO 29900
29700  REM  RELATIVE BRANCH
29705 X1% =  PEEK (DADDR):DADDR = DADDR + 1
29710 X% = X1%: GOSUB 14100: PRINT "    ";
29715  GOSUB 14200: PRINT " $";
29720  IF X1% < 128 THEN 29750
29730 X1% =  - 1 * (256 - X1%)
29750 X = DADDR + X1%
29760  GOSUB 14300: REM  PRINT 16 BITS SPECIAL FIELD
29900  PRINT
29910  IF DADDR < DLIM THEN 29010
29920  GOTO 500
30000  REM  DISASSEMBLE
30010  IF  LEN (B$) = 4 THEN 30020
30012  PRINT "ERROR -- ADDRESS MUST POINT TO ROM"
30015  GOTO 500
30020  IF  LEN (D$) = 0 THEN D$ = "1"
30025  GOSUB 5000: REM  CONVERT D$
30030 X$ =  LEFT$ (B$,1)
30040  IF X$ = "0" OR X$ = "2" OR X$ = "4" OR X$ = "6" OR X$ = "8" OR X$ = "A" OR X$ =
30050 FHR$ = X$                                                    "C" OR X$ = "E" THEN 30012
30060 B$ = BUF$ +  RIGHT$ (B$,3)
30070  GOSUB 2500:DADDR = ADDR
30080 DLIM = DADDR + NUMBYT%
30090  GOTO 29010
31000  REM  START THE VCS
31010  POKE DEV,16
31020  PRINT "FROB MEMORY NOW UNDER VCS CONTROL"
31030  PRINT "PLEASE POWER UP THE VCS TO START IT"
31040  GOTO 500

]
```

```
SOURCE FILE: XCONTROL
0000:            1 * EXPLORER CONTROL PROGRAM
0000:            2 * COPYRIGHT 1982 BY FROBCO -- ALL RIGHTS RESERVED
0000:            3 *
0000:            4 * VERSION  1.1 -- LAST MODIFIED 10/19/82
0000:            5 *
0000:            6 *
0000:            7 * SYSTEM DEFINITIONS
C000:            8 KBD     EQU  $C000     ; APPLE KEYBOARD REGISTER
C010:            9 KSTRB   EQU  $C010     ; APPLE KEYBOARD STROBE LOCATION
0010:           10 SACMD   EQU  $10       ; SET ADDRESS COMMAND FOR FROB
0020:           11 RDCMD   EQU  $20       ; READ REGISTER COMMAND
0040:           12 WRCMD   EQU  $40       ; WRITE REGISTER COMMAND
0000:           13 *
0000:           14 *
0000:           15 *
0000:           16 * PAGE 0 RAM ALLOCATION
0000:           17 *
0085:           18 RLOC    EQU  $85       ; READ REGISTER ADDRESS
0086:           19 FSTAT   EQU  RLOC+1    ; POINTER TO FROB STATUS REGISTER
0088:           20 FDATA   EQU  FSTAT+2   ; POINTER TO FROB DATA REGISTER
008A:           21 RBASE   EQU  FDATA+2   ; SCREEN ROW BASE POINTER FOR READ REGISTERS
008C:           22 CRBASE  EQU  RBASE+2   ; SCREEN ROW BASE POINTER USED BY CURSOR
008E:           23 FBASE   EQU  CRBASE+2  ; FIELD ARRAY ROW BASE POINTER
0090:           24 CROW    EQU  FBASE+2   ; CURSOR ROW
0091:           25 CCOL    EQU  CROW+1    ; CURSOR COLUMN
0092:           26 RRCOL   EQU  CCOL+1    ; READ REG COLUMN
0093:           27 OLDCR   EQU  RRCOL+1   ; OLD CURSOR ROW
0094:           28 OLDCCL  EQU  OLDCR+1   ; OLD CURSOR COLUMN
0095:           29 ROW     EQU  OLDCCL+1  ; PARAMETER TO ROW BASE LOOKUP ROUTINE
0096:           30 TMP1    EQU  ROW+1     ; RANDOM TEMP STORE
0097:           31 RDHGH   EQU  TMP1+1    ; READ ADDRESS HIGH POINTER
0098:           32 RDLOW   EQU  RDHGH+1   ; LOW PART OF ABOVE
0099:           33 LSTKEY  EQU  RDLOW+1   ; LAST KEY PRESSED
009A:           34 LSTRD   EQU  LSTKEY+1  ; LAST READ REGISTER VALUE
0000:           35 *
0000:           36 *
0000:           37 * START OF PROGRAM -- LOAD AT $8000
0000:           38 *
----- NEXT OBJECT FILE NAME IS XCONTROL.OBJ0
8000:           39         ORG  $8000
8000:           40 *
8000:20 93 82   41 START   JSR  INIT      ; THIS PROGRAM IS CALLED FROM BASIC
8003:20 B5 80   42 MLOOP   JSR  INCRR     ; GO INCREMENT THE READ REGISTER
8006:90 03      43         BCC  MLP1      ; CHECK FOR FULL SET DONE
8008:20 72 80   44         JSR  SPECR     ; IF SO GO DO SPECIAL READ
800B:20 E5 80   45 MLP1    JSR  RDREG     ; GO DO STANDARD READ
800E:85 9A      46         STA  LSTRD     ; SAVE RESULT
8010:20 F9 80   47         JSR  ADRCNV    ; DO AN ADDRESS CONVERSION SO WE KNOW WHERE THIS REGISTER
8013:           48 *        SHOULD GO ON THE SCREEN.
8013:A5 9A      49         LDA  LSTRD     ; GET VALUE BACK
8015:20 32 81   50         JSR  DSPBYT    ; PUT IT UP
8018:20 56 81   51         JSR  CHKKEY    ; CHECK FOR KEY PRESSED (GET IT IF SO)
801B:90 E6      52         BCC  MLOOP     ; END OF MAIN LOOP IF NO KEY
801D:C9 83      53         CMP  #$83      ; SEE IF CONTROL C
```

```
801F:D0 03      54          BNE  MLP2      ; IF SO EXIT TO BASIC
8021:4C D0 03   55          JMP  $3D0
8024:85 99      56 MLP2     STA  LSTKEY    ; SAVE KEY
8026:C9 BB      57          CMP  #$BB      ; IS IT THE ";" (UP)?
8028:D0 06      58          BNE  NTUP      ; IF NOT GO ON
802A:20 65 81   59          JSR  MOVUP     ; ELSE MOVE CURSOR UP
802D:4C 03 80   60          JMP  MLOOP
8030:C9 AF      61 NTUP     CMP  #$AF      ; IS IT THE "/" (DOWN)?
8032:D0 06      62          BNE  NTDWN     ; IF NOT GO ON
8034:20 7A 81   63          JSR  MOVDWN    ; ELSE MOVE CURSOR DOWN
8037:4C 03 80   64          JMP  MLOOP
803A:C9 88      65 NTDWN    CMP  #$88      ; IS IT THE LEFT ARROW?
803C:D0 06      66          BNE  NTLFT     ; IF NOT GO ON
803E:20 A6 81   67          JSR  MOVLFT    ; ELSE MOVE CURSOR LEFT
8041:4C 03 80   68          JMP  MLOOP
8044:C9 95      69 NTLFT    CMP  #$95      ; IS IT THE RIGHT ARROW?
8046:D0 06      70          BNE  NTRGH     ; IF NOT GO ON
8048:20 8E 81   71          JSR  MOVRGH    ; ELSE MOVE CURSOR RIGHT
804B:4C 03 80   72          JMP  MLOOP
804E:           73 *
804E:20 ED 81   74 NTRGH    JSR  NUMFLD    ; CHECK TO SEE IF CURSOR IN NUMBER FIELD
8051:90 B0      75          BCC  MLOOP     ; IF NOT THE KEY IS MEANINGLESS
8053:A5 99      76          LDA  LSTKEY    ; GET KEY BACK
8055:20 1D 81   77          JSR  HEXKEY    ; IS IT A HEXIDECIMAL DIGIT?
8058:90 A9      78          BCC  MLOOP     ; IF NOT THE KEY IS MEANINGLESS
805A:A5 99      79          LDA  LSTKEY    ; GET KEY BACK AGAIN
805C:20 1F 82   80          JSR  DSPCHR    ; GO PUT IT AT CURSOR AND ADVANCE CURSOR
805F:20 8E 81   81          JSR  MOVRGH
8062:20 ED 81   82          JSR  NUMFLD    ; SEE IF STILL IN NUMBER FIELD
8065:B0 9C      83          BCS  MLOOP     ; IF SO WE ARE DONE FOR NOW
8067:20 02 82   84          JSR  SPECCK    ; ELSE TIME TO DO SOMETHING SO CHECK FOR SPECIAL CASE
806A:B0 97      85          BCS  MLOOP     ; IF TRUE THEN THE SPECIAL CASE HAS BEEN DONE BEFORE RETURN
806C:20 2D 82   86          JSR  DOWRT     ; GO DO THE VCS REGISTER WRITE
806F:4C 03 80   87          JMP  MLOOP     ; END OF TOP LEVEL
8072:           88 *
8072:           89 *
8072:           90 * ROUTINE TO DO A SPECIAL READ
8072:           91 *
8072:A5 93      92 SPECR    LDA  OLDCR     ; WE WILL NEED TO USE
8074:48         93          PHA             ; OLDCR AND OLDCCL AS PARAMETERS
8075:A5 94      94          LDA  OLDCCL    ; TO GETVAL
8077:48         95          PHA
8078:A9 12      96          LDA  #$12      ; ROW OF SPECIAL OPERATIONS
807A:85 93      97          STA  OLDCR
807C:A9 13      98          LDA  #$13      ; COLUMN OF SECOND DIGIT OF THE HIGH ORDER ADDRESS
807E:85 94      99          STA  OLDCCL
8080:20 51 82  100          JSR  GETVAL    ; GET HIGH ADDRESS PART
8083:85 97     101          STA  RDHGH     ; PUT IN HIGH ADDR POINTER
8085:A9 15     102          LDA  #$15      ; COLUMN OF SECOND DIGIT OF THE LOW ORDER ADDRESS
8087:85 94     103          STA  OLDCCL    ; PASS ON TO GETVAL
8089:20 51 82  104          JSR  GETVAL
808C:85 98     105          STA  RDLOW     ; NOW THE ADDRESS IS SET UP
808E:68        106          PLA             ; GET OLDCCL BACK
808F:85 94     107          STA  OLDCCL
8091:68        108          PLA             ; GET OLDCR BACK
8092:85 93     109          STA  OLDCR
8094:20 E5 80  110          JSR  RDREG     ; GO DO THE VCS READ
```

```
8097:48        111         PHA             ; SAVE THE RESULT
8098:A9 24     112         LDA  #$24       ; COLUMN OF THE RESULT
809A:85 92     113         STA  RRCOL
809C:A2 12     114         LDX  #$12       ; ROW OF THE RESULT
809E:BD 04 83  115         LDA  CHLIST,X   ; GET HIGH PART OF SCREE ROW
80A1:85 8B     116         STA  RBASE+1    ; PUT IN HIGH PART
80A3:BD 1C 83  117         LDA  CLLIST,X   ; GET LOW PART
80A6:85 8A     118         STA  RBASE
80A8:68        119         PLA             ; GET VALUE TO BE DISPLAYED
80A9:20 32 81  120         JSR  DSPBYT     PUT IT UP
80AC:A9 00     121         LDA  #0         ; CLEAR RDHGH
80AE:85 97     122         STA  RDHGH
80B0:A5 85     123         LDA  RLOC       ; RESTORE RDLOW
80B2:85 98     124         STA  RDLOW
80B4:60        125         RTS
80B5:          126 *
80B5:          127 *
80B5:          128 * ROUTINE TO INCREMENT READ REGISTER NUMBER AND RETURN CARRY SET
80B5:          129 * WHEN THROUGH THE ENTIRE LIST
80B5:          130 *
80B5:E6 85     131 INCRR   INC  RLOC
80B7:A5 85     132         LDA  RLOC       ; GET THE NEW VALUE
80B9:C9 40     133         CMP  #$40       ; GONE TOO FAR?
80BB:90 04     134         BCC  INCR1      ; BRANCH IF NOT AT END
80BD:A9 30     135         LDA  #$30       ; ELSE RESET COUNTER
80BF:85 85     136         STA  RLOC
80C1:A9 00     137 INCR1   LDA  #0          ; UPDATE THE READ ADDRESS POINTERS
80C3:85 97     138         STA  RDHGH      ; HIGH PART
80C5:A5 85     139         LDA  RLOC
80C7:85 98     140         STA  RDLOW      ; LOW PART
80C9:60        141         RTS                  ; DONE
80CA:          142 *
80CA:          143 *
80CA:          144 * ROUTINE TO DO A SET READ/WRITE ADDRESS OVER IN THE VCS
80CA:          145 *
80CA:A0 00     146 SETADR  LDY  #0          ; SO WE CAN LOAD INDIRECT
80CC:B1 86     147 SA1     LDA  (FSTAT),Y ; GO GET FROB STATUS REGISTER
80CE:10 FC     148         BPL  SA1        ; WAIT FOR OK TO WRITE
80D0:A9 10     149         LDA  #SACMD     ; GET THE SET ADDRESS COMMAND CODE
80D2:91 88     150         STA  (FDATA),Y ; SEND IT TO THE FROB EXPLORER
80D4:B1 86     151 SA2     LDA  (FSTAT),Y ; GO GET FROB STATUS REGISTER
80D6:10 FC     152         BPL  SA2        ; WAIT FOR OK FOR NEXT WRITE
80D8:A5 97     153         LDA  RDHGH      ; GET THE HIGH ORDER PART
80DA:91 88     154         STA  (FDATA),Y ; SEND IT
80DC:B1 86     155 SA3     LDA  (FSTAT),Y ; SAME AS ABOVE
80DE:10 FC     156         BPL  SA3
80E0:A5 98     157         LDA  RDLOW      ; GET THE LOW ORDER PART
80E2:91 88     158         STA  (FDATA),Y ; THAT FINISHES THE JOB.
80E4:60        159         RTS
80E5:          160 *
80E5:          161 *
80E5:          162 * ROUTINE TO READ A VCS REGISTER
80E5:          163 *
80E5:20 CA 80  164 RDREG   JSR  SETADR     ; SEND THE VCS THE ADDRESS OF THE REGISTER WE WANT
80E8:B1 86     165 RDR0    LDA  (FSTAT),Y ; GET FROB STATUS
80EA:10 FC     166         BPL  RDR0       ; WAIT TIL OK TO WRITE
80EC:A9 20     167         LDA  #RDCMD     ; SEND IT A READ REGISTER COMMAND
```

```
80EE:91 88     168         STA  (FDATA),Y
80F0:B1 86     169 RDR1    LDA  (FSTAT),Y ; GET FROB STATUS REGISTER
80F2:29 40     170         AND  #$40      ; LOOK AT DATA COMING BACK BIT
80F4:F0 FA     171         BEQ  RDR1      ; WAIT FOR DATA TO COME BACK
80F6:B1 88     172         LDA  (FDATA),Y ; NOW READ DATA
80F8:60        173         RTS
80F9:          174 *
80F9:          175 *
80F9:          176 * ROUTINE TO CONVERT READ REGISTER NUMBER INTO SCREEN ADDRESS
80F9:          177 *
80F9:A5 85     178 ADRCNV  LDA  RLOC      ; GET READ REGISTER NUMBER
80FB:29 0F     179         AND  #$0F      ; LOOK AT LOWER BITS
80FD:4A        180         LSR  A          ; DIVIDE BY 4
80FE:4A        181         LSR  A
80FF:09 14     182         ORA  #$14      ; PUT IN ROW OFFSET
8101:AA        183         TAX            ; SAVE IN X
8102:BD 04 83  184         LDA  CHLIST,X  ; LOOKUP THE ROW BASE ADDRESS HIGH
8105:85 8B     185         STA  RBASE+1   ; SAVE IT IN POINTER
8107:BD 1C 83  186         LDA  CLLIST,X  ; LOOKUP THE ROW BASE ADRESS LOW
810A:85 8A     187         STA  RBASE     ; SAVE IT IN POINTER
810C:A5 85     188         LDA  RLOC      ; GET REGISTER NUMBER AGAIN
810E:29 03     189         AND  #$3       ; LOOK AT LAST TWO BITS
8110:85 96     190         STA  TMP1      ; MULTIPLY BY 9
8112:0A        191         ASL  A
8113:0A        192         ASL  A
8114:0A        193         ASL  A
8115:05 96     194         ORA  TMP1
8117:18        195         CLC               ; NOW ADD COLUMN OFFSET
8118:69 06     196         ADC  #6
811A:85 92     197         STA  RRCOL
811C:60        198         RTS
811D:          199 *
811D:          200 *
811D:          201 * ROUTINE TO CHECK KEY NUMBER FOR LEGAL HEX DIGIT
811D:          202 *
811D:C9 BA     203 HEXKEY  CMP  #$BA      ; SEE IF IT IS IN ALPHA RANGE
811F:B0 05     204         BCS  HK1
8121:C9 B0     205         CMP  #$B0
8123:90 0B     206         BCC  NOHEX     ; ALSO FAIL IF TOO LOW
8125:60        207         RTS               ; ELSE OK
8126:          208 *
8126:C9 C1     209 HK1     CMP  #$C1      ; SEE IF OUT OF BA>C0 RANGE
8128:90 06     210         BCC  NOHEX
812A:C9 C7     211         CMP  #$C7      ; SEE IF OFF THE TOP
812C:B0 02     212         BCS  NOHEX
812E:38        213         SEC               ; ELSE IT IS OK
812F:60        214         RTS
8130:          215 *
8130:18        216 NOHEX   CLC            ; RETURN CARRY CLEAR ON FAIL
8131:60        217         RTS
8132:          218 *
8132:          219 *
8132:          220 * ROUTINE TO PUT UP HEX BYTE AT RBASE AND RRCOL
8132:          221 *
8132:85 96     222 DSPBYT  STA  TMP1      ; SAVE BYTE TO BE DISPALYED
8134:4A        223         LSR  A          ; GET UPPER 4 BITS
8135:4A        224         LSR  A
```

```
8136:4A        225         LSR  A
8137:4A        226         LSR  A
8138:20 4A 81  227         JSR  TOHEX     ; CONVERT NIBBLE TO ALPHANUMERIC FOR THE HEX DIGIT
813B:A4 92     228         LDY  RRCOL     ; GET THE COLUMN ADDRESS
813D:91 8A     229         STA  (RBASE),Y ; PUT THE CHAR ON THE SCREEN
813F:A5 96     230         LDA  TMP1      ; GET BYTE BACK
8141:20 4A 81  231         JSR  TOHEX     ; CONVERT LOWER NIBBLE
8144:A4 92     232         LDY  RRCOL     ; GET THE COLUMN ADDRESS
8146:C8        233         INY              ; MOVE TO NEXT COLUMN
8147:91 8A     234         STA  (RBASE),Y ; PUT UP CHAR
8149:60        235         RTS
814A:          236 *
814A:          237 *
814A:          238 * ROUTINE TO RETURN HEX ALPHA CHAR FOR LOW NIBBLE IN A
814A:          239 *
814A:29 0F     240 TOHEX   AND  #$0F      ; GET LOW NIBBLE
814C:C9 0A     241         CMP  #$0A      ; SEE IF IN THE ALPHA GROUP
814E:90 02     242         BCC  TH1       ; BRANCH IF NOT ALPHA
8150:69 06     243         ADC  #6        ; IF SO ADD 7 (CARRY + 6)
8152:18        244 TH1     CLC            ; PUT IN UPPER BITS
8153:69 B0     245         ADC  #$B0
8155:60        246         RTS
8156:          247 *
8156:          248 * ROUTINE TO SEE IF KEY PRESSED (RETURNS VALUE AND CARRY SET IF SO)
8156:          249 *
8156:2C 00 C0  250 CHKKEY  BIT  KBD       ; LOOK AT KBD STROBE BIT
8159:30 02     251         BMI  CKY1      ; BRANCH IF KEY PRESSED
815B:18        252         CLC              ; ELSE CLEAR CARRY AND EXIT
815C:60        253         RTS
815D:          254 *
815D:AD 00 C0  255 CKY1    LDA  KBD       ; GET KEY CODE
8160:2C 10 C0  256         BIT  KSTRB     ; RESET THE STROBE BIT
8163:38        257         SEC              ; SHOW WE GOT IT
8164:60        258         RTS
8165:          259 *
8165:          260 *
8165:          261 * ROUTINE TO MOVE THE CURSOR UP
8165:          262 *
8165:A5 90     263 MOVUP   LDA  CROW      ; GET THE CURSOR ROW POSITION
8167:85 93     264         STA  OLDCR     ; THIS IS NOW THE OLD POSITION
8169:C6 90     265         DEC  CROW      ; MOVE NEW POSITION UP
816B:A5 90     266         LDA  CROW      ; CHECK FOR WRAP OFF TOP
816D:10 04     267         BPL  MUP1      ; BRANCH IF OK
816F:A9 17     268         LDA  #$17      ; ELSE MOVE TO BOTTOM ROW
8171:85 90     269         STA  CROW
8173:A5 91     270 MUP1    LDA  CCOL      ; UPDATE OLD COLUMN
8175:85 94     271         STA  OLDCCL
8177:4C BB 81  272         JMP  CUPDTE    ; THEN GO UPDATE THE CURSOR
817A:          273 *
817A:          274 *
817A:          275 * ROUTINE TO MOVE CURSOR DOWN
817A:          276 *
817A:A5 90     277 MOVDWN  LDA  CROW      ; GET THE CURSOR ROW
817C:85 93     278         STA  OLDCR     ; UPDATE THE OLD POSITION
817E:C9 17     279         CMP  #$17      ; AT THE BOTTOM?
8180:F0 05     280         BEQ  MDN1      ; IF SO WRAP TO TOP
8182:E6 90     281         INC  CROW      ; ELSE MOVE DOWN
```

```
8184:4C 73 81  282          JMP  MUP1       ; AND HANDLE AS ABOVE
8187:          283 *
8187:A9 00     284 MDN1     LDA  #0         ; WRAP TO TOP OF SCREEN
8189:85 90     285          STA  CROW
818B:4C 73 81  286          JMP  MUP1
818E:          287 *
818E:          288 *
818E:          289 * ROUTINE TO MOVE THE CURSOR TO THE RIGHT
818E:          290 *
818E:A5 91     291 MOVRGH   LDA  CCOL       ; GET THE CURSOR COLUMN
8190:85 94     292          STA  OLDCCL     ; UPDATE THE OLD POSITION
8192:C9 27     293          CMP  #39        ; SEE IF AT RIGHT EDGE
8194:D0 07     294          BNE  MRT0       ; BRANCH IF SIMPLE MOVE
8196:A9 00     295          LDA  #0         ; ELSE RESET COLUMN
8198:85 91     296          STA  CCOL
819A:4C 9F 81  297          JMP  MRT1
819D:          298 *
819D:E6 91     299 MRT0     INC  CCOL       ; ELSE MOVE TO THE RIGHT
819F:A5 90     300 MRT1     LDA  CROW       ; GET THE ROW
81A1:85 93     301          STA  OLDCR       ; UPDATE OLD POSITION
81A3:4C B8 81  302          JMP  CUPDTE     ; GO UPDATE THE CURSOR
81A6:          303 *
81A6:          304 *
81A6:          305 * ROUTINE TO MOVE THE CURSOR LEFT
81A6:          306 *
81A6:A5 91     307 MOVLFT   LDA  CCOL       ; GET THE COLUMN
81A8:85 94     308          STA  OLDCCL     ; UPDATE OLD POSITION
81AA:D0 07     309          BNE  MLF1       ; BRANCH IF ROOM FOR SIMPLE MOVE
81AC:A9 27     310          LDA  #39        ; ELSE MOVE TO LAST COLUMN
81AE:85 91     311          STA  CCOL
81B0:4C 9F 81  312          JMP  MRT1
81B3:          313 *
81B3:C6 91     314 MLF1     DEC  CCOL
81B5:4C 9F 81  315          JMP  MRT1       ; TAKE CARE OF AS ABOVE
81B8:          316 *
81B8:          317 *
81B8:          318 * ROUTINE TO UPDATE CURSOR POSITION
81B8:          319 *
81B8:A4 94     320 CUPDTE   LDY  OLDCCL     ; GET OLD CURSOR COLUMN POSITION
81BA:A5 93     321          LDA  OLDCR      ; GET OLD CURSOR ROW
81BC:85 95     322          STA  ROW        ; SAVE AS PARAMETER TO CADDR
81BE:20 E0 81  323          JSR  CRADDR     ; LOOKUP CURSOR ROW BASE ADDRESS GIVEN 'ROW'
81C1:B1 8C     324          LDA  (CRBASE),Y ; GET WHAT IS AT THE OLD CURSOR POSITION
81C3:09 80     325          ORA  #$80       ; TAKE OUT OF INVERSE VIDEO
81C5:85 96     326          STA  TMP1       ; HERE IS THE MAPPING
81C7:29 20     327          AND  #$20       ; ALGORITHM
81C9:0A        328          ASL  A
81CA:49 40     329          EOR  #$40
81CC:05 96     330          ORA  TMP1
81CE:91 8C     331          STA  (CRBASE),Y ; PUT IT BACK
81D0:A4 91     332          LDY  CCOL       ; GET THE NEW CURSOR COLUMN
81D2:A5 90     333          LDA  CROW       ; GET THE NEW CURSOR ROW
81D4:85 95     334          STA  ROW        ; SAVE AS PARAMETER TO CRADDR
81D6:20 E0 81  335          JSR  CRADDR     ; LOOKUP NEW CURSOR ROW BASE ADDRESS GIVEN 'ROW'
81D9:B1 8C     336          LDA  (CRBASE),Y ; GET NEW CURSOR CHARACTOR
81DB:29 3F     337          AND  #$3F       ; MAKE INVERSE VIDEO
81DD:91 8C     338          STA  (CRBASE),Y ; PUT IT BACK
```

```
81DF:60        339         RTS
81E0:          340 *
81E0:          341 *
81E0:          342 * ROUTINE TO LOOKUP CURSOR ROW BASE ADDRESS FROM INDEX 'ROW'
81E0:          343 *
81E0:A6 95     344 CRADDR  LDX  ROW       ; GET INDEX
81E2:BD 04 83  345         LDA  CHLIST,X  ; LOOKUP HIGH PART
81E5:85 8D     346         STA  CRBASE+1  ; SAVE IN HIGH POINTER
81E7:BD 1C 83  347         LDA  CLLIST,X  ; LOOKUP LOW PART
81EA:85 8C     348         STA  CRBASE     ; SAVE IN LOW POINTER
81EC:60        349         RTS
81ED:          350 *
81ED:          351 *
81ED:          352 * ROUTINE TO SEE IF CURSOR IS IN A NUMERITC FIELD
81ED:          353 *
81ED:A6 90     354 NUMFLD  LDX  CROW      ; LOOKUP ROW BASE ADDRESS
81EF:BD B4 8A  355         LDA  FHLIST,X  ; HIGH PART OF FIELD LIST
81F2:85 8F     356         STA  FBASE+1   ; SAVE IN HIGH POINTER
81F4:BD CC 8A  357         LDA  FLLIST,X  ; LOOKUP LOW PART
81F7:85 8E     358         STA  FBASE     ; SAVE IN LOW POINTER
81F9:A4 91     359         LDY  CCOL      ; GET THE COLUMN NUMBER
81FB:B1 8E     360         LDA  (FBASE),Y ; GET THE FIELD FLAG
81FD:4C 1D 81  361         JMP  HEXKEY    ; GO SEE IF LEGAL HEX DIGIT
8200:          362 *
8200:18        363 NONUM   CLC            ; CLEAR CARRY TO RETURN 'FALSE'
8201:60        364         RTS
8202:          365 *
8202:          366 *
8202:          367 * ROUTINE TO HANDLE SPECIAL CASE
8202:A5 93     368 SPECCK  LDA  OLDCR     ; SEE IF IN SPECIAL ROW
8204:C9 12     369         CMP  #$12
8206:D0 15     370         BNE  NOSPEC    ; IF NOT RETURN CARRY CLEAR
8208:A5 94     371         LDA  OLDCCL    ; ELSE CHECK COLUMN FOR LAST ADDRESS DIGIT
820A:C9 15     372         CMP  #$15
820C:F0 0D     373         BEQ  SPDONE    ; IF SO WE HAVE DONE THE ADDRESS
820E:C9 1C     374         CMP  #$1C      ; THIS IS THE WRITE OPERATION COLUMN
8210:D0 0B     375         BNE  NOSPEC    ; IF NOT THIS THEN NOT SPECIAL
8212:20 72 80  376         JSR  SPECR     ; DO A SPECIAL READ TO SET UP THE ADDRESS
8215:20 51 82  377         JSR  GETVAL    ; GET THE VALUE TO WRITE
8218:20 80 82  378         JSR  WTREG     ; DO THE WRITE
821B:38        379 SPDONE  SEC            ; DONE SO SET CARRY AND RETURN
821C:60        380         RTS
821D:          381 *
821D:18        382 NOSPEC  CLC            ; IF NOT SPECIAL THEN CLEAR CARRY AND RETURN
821E:60        383         RTS
821F:          384 *
821F:          385 * ROUTINE TO PUT CHARACTOR AT CURSOR
821F:          386 *
821F:A5 90     387 DSPCHR  LDA  CROW      ; GET THE CURSOR ROW
8221:85 95     388         STA  ROW       ; SETUP FOR ROW ADDRESS COMPUTATION
8223:20 E0 81  389         JSR  CRADDR    ; DO LOOKUP
8226:A4 91     390         LDY  CCOL      ; GET THE COLUMN
8228:A5 99     391         LDA  LSTKEY    ; GET THE CHARACTOR
822A:91 8C     392         STA  (CRBASE),Y ; PUT IT ON THE SCREEN
822C:60        393         RTS
822D:          394 *
822D:          395 *
```

```
822D:          396 * ROUTINE TO DO THE VCS REG WRITE
822D:          397 *
822D:20 3D 82  398 DOWRT   JSR  GETFLD    ; GET THE FIELD BYTE FOR THE OLD CURSOR
8230:20 4A 82  399         JSR  SETREG    ; TURN THAT INTO A VCS REGISTER ADDRESS
8233:20 CA 80  400         JSR  SETADR    ; SET IT UP IN THE VCS
8236:20 51 82  401         JSR  GETVAL    ; GO GET THE CURRENT VALUE OF THE OLD CURSOR FIELD
8239:20 80 82  402         JSR  WTREG     ; GO SEND IT TO THE VCS
823C:60        403         RTS              ; DONE
823D:          404 *
823D:          405 *
823D:          406 * ROUTINE TO GET FIELD BYTE
823D:          407 *
823D:A6 93     408 GETFLD  LDX  OLDCR     ; GET THE OLD CURSOR ROW
823F:BD B4 8A  409         LDA  FHLIST,X  ; GET THE HIGH PART OF THE FIELD ROW BASE ADDRESS
8242:85 8D     410         STA  CRBASE+1  ; USE THIS POINTER SO WE CAN DROP IN BELOW
8244:BD CC 8A  411         LDA  FLLIST,X  ; NOW THE LOW PART
8247:4C 5B 82  412         JMP  GVAL1       ; DO THE REST BELOW
824A:          413 *
824A:          414 *
824A:          415 * ROUTINE TO GO FROM FIELD BYTE TO VCS ADDRESS
824A:          416 *
824A:85 98     417 SETREG  STA  RDLOW     ; SAVE IN LOW ORDER REGISTER
824C:A9 00     418         LDA  #0        ; CLEAR OUT HIGH PART
824E:85 97     419         STA  RDHGH
8250:60        420         RTS
8251:          421 *
8251:          422 *
8251:          423 * ROUTINE TO GET VALUE OF THE FIELD AT OLD CURSOR
8251:          424 *
8251:A6 93     425 GETVAL  LDX  OLDCR     ; GET THE OLD CURSOR ROW
8253:BD 04 83  426         LDA  CHLIST,X  ; GET HIGH PART OF ROW BASE ADDRESS
8256:85 8D     427         STA  CRBASE+1, ; SAVE IN HIGH PART OF POINTER
8258:BD 1C 83  428         LDA  CLLIST,X  ; GET LOW PART OF ROW BASE ADDRESS
825B:85 8C     429 GVAL1   STA  CRBASE    ; SAVE IT
825D:A4 94     430         LDY  OLDCCL    ; GET THE OLD CURSOR COLUMN POSITION
825F:B1 8C     431         LDA  (CRBASE),Y ; THIS GETS THE LOW DIGIT OF THE VALUE
8261:48        432         PHA              ; SAVE IT
8262:88        433         DEY              ; BACK UP TO THE HIGH DIGIT
8263:B1 8C     434         LDA  (CRBASE),Y ; GET THAT
8265:20 75 82  435         JSR  TOBIN     ; CONVERT FROM HEX DIGIT TO BINARY
8268:0A        436         ASL  A         ; MAKE HIGH PART
8269:0A        437         ASL  A
826A:0A        438         ASL  A
826B:0A        439         ASL  A
826C:85 96     440         STA  TMP1      ; SAVE FOR LATER
826E:68        441         PLA            ; GET LOW ORDER HEX DIGIT BACK
826F:20 75 82  442         JSR  TOBIN     ; CONVERT TO BINARY
8272:05 96     443         ORA  TMP1      ; THIS PUTS IT TOGETHER
8274:60        444         RTS
8275:          445 *
8275:          446 *
8275:          447 * ROUTINE TO CONVERT ASCII FOR HEX DIGIT BACK TO BINARY
8275:          448 *
8275:29 7F     449 TOBIN   AND  #$7F      ; GET RID OF TOP BIT
8277:C9 3A     450         CMP  #$3A      ; SEE IF IN NUMERIC RANGE
8279:90 02     451         BCC  TOB1      ; BRANCH IF SO
827B:E9 07     452         SBC  #7        ; ELSE SUBTRACT 7 TO CORRECT ALPHA RANGE
```

```
827D:29 0F     453 TOB1    AND  #$0F      ; NOW WE JUST NEED THE LAST 4 BITS
827F:60        454         RTS
8280:          455 *
8280:          456 *
8280:          457 * ROUTINE TO DO A VCS REGISTER WRITE
8280:          458 *
8280:48        459 WTREG   PHA            ; SAVE VALUE
8281:A0 00     460         LDY  #0        ; SET UP TO INDIRECT
8283:B1 86     461 WTR1    LDA  (FSTAT),Y ; LOOK AT FROB STATUS REGISTER
8285:10 FC     462         BPL  WTR1      ; WAIT TIL OK TO WRITE
8287:A9 40     463         LDA  #WRCMD    ; SEND IT A FROB EXPLORER WRITE COMMAND
8289:91 88     464         STA  (FDATA),Y
828B:B1 86     465 WTR2    LDA  (FSTAT),Y ; WAIT FOR IT TO GET IT
828D:10 FC     466         BPL  WTR2
828F:68        467         PLA            ; GET VALUE BACK
8290:91 88     468         STA  (FDATA),Y SEND IT TO VCS
8292:60        469         RTS
8293:          470 *
8293:          471 *
8293:          472 * THE INITILAIZATION ROUTINE
8293:          473 *
8293:20 D6 82  474 INIT    JSR  SLOAD     ; PUT UP THE SCREEN
8296:A9 C0     475         LDA  #$C0      ; SET UP FSTAT AND FDATA
8298:85 87     476         STA  FSTAT+1
829A:85 89     477         STA  FDATA+1
829C:A9 A0     478         LDA  #$A0
829E:85 86     479         STA  FSTAT
82A0:A9 A1     480         LDA  #$A1
82A2:85 88     481         STA  FDATA
82A4:A9 00     482         LDA  #0        ; CLEAR SOME VARIABLES
82A6:85 94     483         STA  OLDCCL
82A8:85 93     484         STA  OLDCR
82AA:85 91     485         STA  CCOL
82AC:85 90     486         STA  CROW
82AE:85 99     487         STA  LSTKEY
82B0:          488 *
82B0:          489 * NOW A LOOP TO DO THE INITIAL REGISTER WRITES
82B0:          490 *
82B0:20 ED 81  491 ILP     JSR  NUMFLD    ; IS CURSOR AT A NUMBER FIELD?
82B3:B0 1A     492              BCS       INFLD ; IF YES, JUST PUT THE FLAG IN LSTKEY AND MOVE ON
82B5:24 99     493         BIT  LSTKEY    ; IF NOT, SEE IF WE JUST WERE
82B7:F0 07     494         BEQ  NOCHNG    ; IF NOT THEN THE CHANGE IS NOT IMPORTANT
82B9:A9 00     495         LDA  #0        ; CLEAR LSTKEY FLAG
82BB:85 99     496         STA  LSTKEY
82BD:20 2D 82  497         JSR  DOWRT     ; UPDATE THE REGISTER
82C0:          498 *
82C0:20 8E 81  499 NOCHNG  JSR  MOVRGH    ; MOVE TO THE NEXT POSITION
82C3:A5 91     500         LDA  CCOL      ; GET NEW CURSOR COLUMN
82C5:D0 E9     501         BNE  ILP       ; LOOP IF NOT READY FOR NEW LINE
82C7:20 7A 81  502         JSR  MOVDWN    ; ELSE MOVE DOWN TO NEXT LINE
82CA:A5 90     503         LDA  CROW      ; SEE IF AT THE END OF THE DISPLAY
82CC:D0 E2     504         BNE  ILP       ; IF NOT KEEP GOING
82CE:60        505         RTS              ; BUT IF SO THEN WE ARE DONE
82CF:          506 *
82CF:A9 FF     507 INFLD   LDA  #$FF      ; SET LSTKEY TO INDICATE WE ARE IN A FIELD
82D1:85 99     508         STA  LSTKEY
82D3:4C C0 82  509         JMP  NOCHNG    ; GET BACK IN
```

```
82D6:           510 *
82D6:           511 *
82D6:           512 *
82D6:           513 * ROUTINE TO LOAD DISPLAY FROM ARRAY "SCREEN"
82D6:           514 *
82D6:A2 17      515 SLOAD   LDX  #$17      ; DO FOR 24 LINES
82D8:A9 86      516         LDA  #<SCREEN+920 ; SCREEN+(23*40) POINTS TO THE LAST LINE
82DA:85 8B      517         STA  RBASE+1   ; STORE THE HIGH PART
82DC:A9 CC      518         LDA  #>SCREEN+920 ; NOW THE LOW PART
82DE:85 8A      519         STA  RBASE
82E0:BD 04 83   520 SL1     LDA  CHLIST,X  ; GET HIGH POINTER TO DISPLAY ROW
82E3:85 8D      521         STA  CRBASE+1  ; PUT IN POINTER
82E5:BD 1C 83   522         LDA  CLLIST,X  ; GET LOW PART
82E8:85 8C      523         STA  CRBASE    ; PUT THAT IN POINTER
82EA:A0 27      524         LDY  #39       ; WORK BACK FROM END OF LINE
82EC:B1 8A      525 SL2     LDA  (RBASE),Y ; GET CHARACTOR FROM ARRAY
82EE:91 8C      526         STA  (CRBASE),Y ; PUT UP ON SCREEN
82F0:88         527         DEY
82F1:10 F9      528         BPL  SL2       ; DO FOR Y=39 TO 0
82F3:38         529         SEC            ; SUBTRACT 40 FROM RBASE
82F4:A5 8A      530         LDA  RBASE
82F6:E9 28      531         SBC  #40
82F8:85 8A      532         STA  RBASE
82FA:A5 8B      533         LDA  RBASE+1   ; 16 BIT
82FC:E9 00      534         SBC  #0
82FE:85 8B      535         STA  RBASE+1
8300:CA         536         DEX              ; THIS IS THE MAJOR LOOP COUNT
8301:10 DD      537         BPL  SL1       ; WORK THROUGH 24 LINES
8303:60         538         RTS
8304:           539 *
8304:           540 *
8304:           541 *
8304:           542 * SCREEN ROW BASE LOOKUP TABLES
8304:           543 *
8304:04 04 05   544 CHLIST  DFB  4,4,5,5,6,6,7,7,4,4,5,5,6,6,7,7,4,4,5,5,6,6,7,7
8307:05 06 06
830A:07 07 04
830D:04 05 05
8310:06 06 07
8313:07 04 04
8316:05 05 06
8319:06 07 07
831C:00 80 00   545 CLLIST  DFB  0,$80,0,$80,0,$80,0,$80
831F:80 00 80
8322:00 80
8324:28 A8 28   546         DFB  $28,$A8,$28,$A8,$28,$A8,$28,$A8
8327:A8 28 A8
832A:28 A8
832C:50 D0 50   547         DFB  $50,$D0,$50,$D0,$50,$D0,$50,$D0
832F:D0 50 D0
8332:50 D0
8334:           548 *
8334:           549 *
8334:           550 * HERE IS THE PAGE SETUP
8334:           551 *
8334:           552         MSB  ON        ; DO STRINGS WITH MSB ON
8334:           553 *
```

```
8334:A0 A0 A0  554 SCREEN  ASC  /              FROB EXPLORER CONTROL SCREEN      /
8337:A0 A0 A0
833A:C6 D2 CF
833D:C2 A0 C5
8340:D8 D0 CC
8343:CF D2 C5
8346:D2 A0 C3
8349:CF CE D4
834C:D2 CF CC
834F:A0 D3 C3
8352:D2 C5 C5
8355:CE A0 A0
8358:A0 A0 A0
835B:A0
835C:C5 CE C1  555          ASC  /ENABLE:0C                              /
835F:C2 CC C5
8362:BA B0 C3
8365:A0 A0 A0
8368:A0 A0 A0
836B:A0 A0 A0
836E:A0 A0 A0
8371:A0 A0 A0
8374:A0 A0 A0
8377:A0 A0 A0
837A:A0 A0 A0
837D:A0 A0 A0
8380:A0 A0 A0
8383:A0
8384:D0 CC C1  556          ASC  /PLAYER1:COLOR 82 VERT 14 HORZ 30 FLP 00 /
8387:D9 C5 D2
838A:B1 BA C3
838D:CF CC CF
8390:D2 A0 B8
8393:B2 A0 D6
8396:C5 D2 D4
8399:A0 B1 B4
839C:A0 C8 CF
839F:D2 DA A0
83A2:B3 B0 A0
83A5:C6 CC D0
83A8:A0 B0 B0
83AB:A0
83AC:A0 A0 A0  557          ASC  /             IMAGE: FE 7F 63 62 7C 78 60 F2    /
83AF:A0 C9 CD
83B2:C1 C7 C5
83B5:BA A0 C6
83B8:C5 A0 B7
83BB:C6 A0 B6
83BE:B3 A0 B6
83C1:B2 A0 B7
83C4:C3 A0 B7
83C7:B8 A0 B6
83CA:B0 A0 C6
83CD:B2 A0 A0
83D0:A0 A0 A0
83D3:A0
83D4:A0 A0 A0  558          ASC  /             SOUND: A:00 B:00 C:00             /
```

```
83D7:A0 D3 CF
83DA:D5 CE C4
83DD:BA A0 C1
83E0:BA B0 B0
83E3:A0 C2 BA
83E6:B0 B0 A0
83E9:C3 BA B0
83EC:B0 A0 A0
83EF:A0 A0 A0
83F2:A0 A0 A0
83F5:A0 A0 A0
83F8:A0 A0 A0
83FB:A0
83FC:D0 CC C1  559        ASC  /PLAYER2:COLOR 00 VERT 1E HORZ 30 FLP 00 /
83FF:D9 C5 D2
8402:B2 BA C3
8405:CF CC CF
8408:D2 A0 B0
840B:B0 A0 D6
840E:C5 D2 D4
8411:A0 B1 C5
8414:A0 C8 CF
8417:D2 DA A0
841A:B3 B0 A0
841D:C6 CC D0
8420:A0 B0 B0
8423:A0
8424:A0 A0 A0  560        ASC  /          IMAGE: FE 7F 63 62 7C 78 60 F6      /
8427:A0 C9 CD
842A:C1 C7 C5
842D:BA A0 C6
8430:C5 A0 B7
8433:C6 A0 B6
8436:B3 A0 B6
8439:B2 A0 B7
843C:C3 A0 B7
843F:B8 A0 B6
8442:B0 A0 C6
8445:B6 A0 A0
8448:A0 A0 A0
844B:A0
844C:A0 A0 A0  561        ASC  /          SOUND: A:00 B:00 C:00               /
844F:A0 D3 CF
8452:D5 CE C4
8455:BA A0 C1
8458:BA B0 B0
845B:A0 C2 BA
845E:B0 B0 A0
8461:C3 BA B0
8464:B0 A0 A0
8467:A0 A0 A0
846A:A0 A0 A0
846D:A0 A0 A0
8470:A0 A0 A0
8473:A0
8474:C2 C1 C3  562        ASC  /BACKGROUND COLOR EA  OBJECT COLOR 64    /
8477:CB C7 D2
```

```
847A:CF D5 CE
847D:C4 A0 C3
8480:CF CC CF
8483:D2 A0 C5
8486:C1 A0 A0
8489:CF C2 CA
848C:C5 C3 D4
848F:A0 C3 CF
8492:CC CF D2
8495:A0 B6 B4
8498:A0 A0 A0
849B:A0
849C:CF C2 CA  563        ASC  /OBJECT   A IMAGE: 18 24 42 42 7E 42 42 CE /
849F:C5 C3 D4
84A2:A0 C1 A0
84A5:C9 CD C1
84A8:C7 C5 BA
84AB:A0 B1 B8
84AE:A0 B2 B4
84B1:A0 B4 B2
84B4:A0 B4 B2
84B7:A0 B7 C5
84BA:A0 B4 B2
84BD:A0 B4 B2
84C0:A0 C3 C5
84C3:A0
84C4:CF C2 CA  564        ASC  /OBJECT   B IMAGE: FE 61 61 7E 7E 61 61 FE /
84C7:C5 C3 D4
84CA:A0 C2 A0
84CD:C9 CD C1
84D0:C7 C5 BA
84D3:A0 C6 C5
84D6:A0 B6 B1
84D9:A0 B6 B1
84DC:A0 B7 C5
84DF:A0 B7 C5
84E2:A0 B6 B1
84E5:A0 B6 B1
84E8:A0 C6 C5
84EB:A0
84EC:CF C2 CA  565        ASC  /OBJECT   C IMAGE: 7E FF FE F0 F0 FE FF 7E /
84EF:C5 C3 D4
84F2:A0 C3 A0
84F5:C9 CD C1
84F8:C7 C5 BA
84FB:A0 B7 C5
84FE:A0 C6 C6
8501:A0 C6 C5
8504:A0 C6 B0
8507:A0 C6 B0
850A:A0 C6 C5
850D:A0 C6 C6
8510:A0 B7 C5
8513:A0
8514:D2 C5 C7  566        ASC  /REG03:00 REG04:00 REG05:00 REG0A:05     /
8517:B0 B3 BA
851A:B0 B0 A0
```

```
851D:D2 C5 C7
8520:B0 B4 BA
8523:B0 B0 A0
8526:D2 C5 C7
8529:B0 B5 BA
852C:B0 B0 A0
852F:D2 C5 C7
8532:B0 C1 BA
8535:B0 B5 A0
8538:A0 A0 A0
853B:A0
853C:D2 C5 C7  567          ASC  /REG10:00 REG11:00 REG12:00 REG13:00     /
853F:B1 B0 BA
8542:B0 B0 A0
8545:D2 C5 C7
8548:B1 B1 BA
854B:B0 B0 A0
854E:D2 C5 C7
8551:B1 B2 BA
8554:B0 B0 A0
8557:D2 C5 C7
855A:B1 B3 BA
855D:B0 B0 A0
8560:A0 A0 A0
8563:A0
8564:D2 C5 C7  568          ASC  /REG14:00 REG1F:00 REG24:00 REG25:00     /
8567:B1 B4 BA
856A:B0 B0 A0
856D:D2 C5 C7
8570:B1 C6 BA
8573:B0 B0 A0
8576:D2 C5 C7
8579:B2 B4 BA
857C:B0 B0 A0
857F:D2 C5 C7
8582:B2 B5 BA
8585:B0 B0 A0
8588:A0 A0 A0
858B:A0
858C:D2 C5 C7  569          ASC  /REG26:00 REG27:00 REG28:00 REG29:00     /
858F:B2 B6 BA
8592:B0 B0 A0
8595:D2 C5 C7
8598:B2 B7 BA
859B:B0 B0 A0
859E:D2 C5 C7
85A1:B2 B8 BA
85A4:B0 B0 A0
85A7:D2 C5 C7
85AA:B2 B9 BA
85AD:B0 B0 A0
85B0:A0 A0 A0
85B3:A0
85B4:D2 C5 C7  570          ASC  /REG2A:00 REG2B:00 REG2C:00 REG2D:00     /
85B7:B2 C1 BA
85BA:B0 B0 A0
85BD:D2 C5 C7
```

```
85C0:B2 C2 BA
85C3:B0 B0 A0
85C6:D2 C5 C7
85C9:B2 C3 BA
85CC:B0 B0 A0
85CF:D2 C5 C7
85D2:B2 C4 BA
85D5:B0 B0 A0
85D8:A0 A0 A0
85DB:A0
85DC:D2 C5 C7  571          ASC  /REG2E:00 REG2F:00                    /
85DF:B2 C5 BA
85E2:B0 B0 A0
85E5:D2 C5 C7
85E8:B2 C6 BA
85EB:B0 B0 A0
85EE:A0 A0 A0
85F1:A0 A0 A0
85F4:A0 A0 A0
85F7:A0 A0 A0
85FA:A0 A0 A0
85FD:A0 A0 A0
8600:A0 A0 A0
8603:A0
8604:C7 C5 CE  572          ASC  /GENERAL  WRITE LOC:0081 VAL:00  READ=00  /
8607:C5 D2 C1
860A:CC A0 D7
860D:D2 C9 D4
8610:C5 A0 CC
8613:CF C3 BA
8616:B0 B0 B8
8619:B1 A0 D6
861C:C1 CC BA
861F:B0 B0 A0
8622:A0 D2 C5
8625:C1 C4 BD
8628:B0 B0 A0
862B:A0
862C:D2 C5 C1  573          ASC  /READ     ONLY LOCATIONS:                /
862F:C4 A0 CF
8632:CE CC D9
8635:A0 CC CF
8638:C3 C1 D4
863B:C9 CF CE
863E:D3 BA A0
8641:A0 A0 A0
8644:A0 A0 A0
8647:A0 A0 A0
864A:A0 A0 A0
864D:A0 A0 A0
8650:A0 A0 A0
8653:A0
8654:D2 C5 C7  574          ASC  /REG30:00 REG31:00 REG32:00 REG33:00     /
8657:B3 B0 BA
865A:B0 B0 A0
865D:D2 C5 C7
8660:B3 B1 BA
```

```
8663:B0 B0 A0
8666:D2 C5 C7
8669:B3 B2 BA
866C:B0 B0 A0
866F:D2 C5 C7
8672:B3 B3 BA
8675:B0 B0 A0
8678:A0 A0 A0
867B:A0
867C:D2 C5 C7  575          ASC  /REG34:00 REG35:00 REG36:00 REG37:00     /
867F:B3 B4 BA
8682:B0 B0 A0
8685:D2 C5 C7
8688:B3 B5 BA
868B:B0 B0 A0
868E:D2 C5 C7
8691:B3 B6 BA
8694:B0 B0 A0
8697:D2 C5 C7
869A:B3 B7 BA
869D:B0 B0 A0
86A0:A0 A0 A0
86A3:A0
86A4:D2 C5 C7  576          ASC  /REG38:00 REG39:00 REG3A:00 REG3B:00     /
86A7:B3 B8 BA
86AA:B0 B0 A0
86AD:D2 C5 C7
86B0:B3 B9 BA
86B3:B0 B0 A0
86B6:D2 C5 C7
86B9:B3 C1 BA
86BC:B0 B0 A0
86BF:D2 C5 C7
86C2:B3 C2 BA
86C5:B0 B0 A0
86C8:A0 A0 A0
86CB:A0
86CC:D2 C5 C7  577          ASC  /REG3C:00 REG3D:00 REG3E:00 REG3F:00     /
86CF:B3 C3 BA
86D2:B0 B0 A0
86D5:D2 C5 C7
86D8:B3 C4 BA
86DB:B0 B0 A0
86DE:D2 C5 C7
86E1:B3 C5 BA
86E4:B0 B0 A0
86E7:D2 C5 C7
86EA:B3 C6 BA
86ED:B0 B0 A0
86F0:A0 A0 A0
86F3:A0
86F4:          578 *
86F4:          579 *
86F4:A0 A0 A0  580 SR0      ASC  /              XXXX XXXXXXXX XXXXXXX XXXXXX      /
86F7:A0 A0 A0
86FA:D8 D8 D8
86FD:D8 A0 D8
```

```
8700:D8 D8 D8
8703:D8 D8 D8
8706:D8 A0 D8
8709:D8 D8 D8
870C:D8 D8 D8
870F:A0 D8 D8
8712:D8 D8 D8
8715:D8 A0 A0
8718:A0 A0 A0
871B:A0
871C:D8 D8 D8  581 SR1     ASC  /XXXXXX:AA                          /
871F:D8 D8 D8
8722:BA C1 C1
8725:A0 A0 A0
8728:A0 A0 A0
872B:A0 A0 A0
872E:A0 A0 A0
8731:A0 A0 A0
8734:A0 A0 A0
8737:A0 A0 A0
873A:A0 A0 A0
873D:A0 A0 A0
8740:A0 A0 A0
8743:A0
8744:D8 D8 D8  582 SR2     ASC  /XXXXXXX:XXXXX AB XXXX AF XXXX B9 XXX 0B /
8747:D8 D8 D8
874A:D8 BA D8
874D:D8 D8 D8
8750:D8 A0 C1
8753:C2 A0 D8
8756:D8 D8 D8
8759:A0 C1 C6
875C:A0 D8 D8
875F:D8 D8 A0
8762:B8 B9 A0
8765:D8 D8 D8
8768:A0 B0 C2
876B:A0
876C:A0 A0 A0  583 SR3     ASC  /           XXXXX: B3 B4 B5 B6 B7 B8 B9 BA      /
876F:A0 D8 D8
8772:D8 D8 D8
8775:BA A0 C2
8778:B3 A0 C2
877B:B4 A0 C2
877E:B5 A0 C2
8781:B6 A0 C2
8784:B7 A0 C2
8787:B8 A0 C2
878A:B9 A0 C2
878D:C1 A0 A0
8790:A0 A0 A0
8793:A0
8794:A0 A0 A0  584 SR4     ASC  /           XXXXX: X:15 X:17 X:19               /
8797:A0 D8 D8
879A:D8 D8 D8
879D:BA A0 D8
87A0:BA B1 B5
```

```
87A3:A0 D8 BA
87A6:B1 B7 A0
87A9:D8 BA B1
87AC:B9 A0 A0
87AF:A0 A0 A0
87B2:A0 A0 A0
87B5:A0 A0 A0
87B8:A0 A0 A0
87BB:A0
87BC:D8 D8 D8  585 SR5     ASC  /XXXXXXX:XXXXX AC XXXX B0 XXXX 8A XXX 0C /
87BF:D8 D8 D8
87C2:D8 BA D8
87C5:D8 D8 D8
87C8:D8 A0 C1
87CB:C3 A0 D8
87CE:D8 D8 D8
87D1:A0 C2 B0
87D4:A0 D8 D8
87D7:D8 D8 A0
87DA:B8 C1 A0
87DD:D8 D8 D8
87E0:A0 B0 C3
87E3:A0
87E4:A0 A0 A0  586 SR6     ASC  /           XXXXX: BB BC BD BE BF C0 C1 C2      /
87E7:A0 D8 D8
87EA:D8 D8 D8
87ED:BA A0 C2
87F0:C2 A0 C2
87F3:C3 A0 C2
87F6:C4 A0 C2
87F9:C5 A0 C2
87FC:C6 A0 C3
87FF:B0 A0 C3
8802:B1 A0 C3
8805:B2 A0 A0
8808:A0 A0 A0
880B:A0
880C:A0 A0 A0  587 SR7     ASC  /           XXXXX: X:16 X:18 X:1A               /
880F:A0 D8 D8
8812:D8 D8 D8
8815:BA A0 D8
8818:BA B1 B6
881B:A0 D8 BA
881E:B1 B8 A0
8821:D8 BA B1
8824:C1 A0 A0
8827:A0 A0 A0
882A:A0 A0 A0
882D:A0 A0 A0
8830:A0 A0 A0
8833:A0
8834:D8 D8 D8  588 SR8     ASC  /XXXXXXXXXX XXXXX AD  XXXXXX XXXXX AE    /
8837:D8 D8 D8
883A:D8 D8 D8
883D:D8 A0 D8
8840:D8 D8 D8
8843:D8 A0 C1
```

```
8846:C4 A0 A0
8849:D8 D8 D8
884C:D8 D8 D8
884F:A0 D8 D8
8852:D8 D8 D8
8855:A0 C1 C5
8858:A0 A0 A0
885B:A0
885C:D8 D8 D8  589 SR9     ASC  /XXXXXX   X XXXXX: C4 C5 C6 C7 C8 C9 CA CB /
885F:D8 D8 D8
8862:A0 D8 A0
8865:D8 D8 D8
8868:D8 D8 BA
886B:A0 C3 B4
886E:A0 C3 B5
8871:A0 C3 B6
8874:A0 C3 B7
8877:A0 C3 B8
887A:A0 C3 B9
887D:A0 C3 C1
8880:A0 C3 C2
8883:A0
8884:D8 D8 D8  590 SRA     ASC  /XXXXXX   X XXXXX: CD CE CF D0 D1 D2 D3 D4 /
8887:D8 D8 D8
888A:A0 D8 A0
888D:D8 D8 D8
8890:D8 D8 BA
8893:A0 C3 C4
8896:A0 C3 C5
8899:A0 C3 C6
889C:A0 C4 B0
889F:A0 C4 B1
88A2:A0 C4 B2
88A5:A0 C4 B3
88A8:A0 C4 B4
88AB:A0
88AC:D8 D8 D8  591 SRB     ASC  /XXXXXX   X XXXXX: D6 D7 D8 D9 DA DB DC DD /
88AF:D8 D8 D8
88B2:A0 D8 A0
88B5:D8 D8 D8
88B8:D8 D8 BA
88BB:A0 C4 B6
88BE:A0 C4 B7
88C1:A0 C4 B8
88C4:A0 C4 B9
88C7:A0 C4 C1
88CA:A0 C4 C2
88CD:A0 C4 C3
88D0:A0 C4 C4
88D3:A0
88D4:D8 D8 D8  592 SRC     ASC  /XXXXX:03 XXXXX:04 XXXXX:05 XXXXX:A5     /
88D7:D8 D8 BA
88DA:B0 B3 A0
88DD:D8 D8 D8
88E0:D8 D8 BA
88E3:B0 B4 A0
88E6:D8 D8 D8
```

```
88E9:D8 D8 BA
88EC:D0 B5 A0
88EF:D8 D8 D8
88F2:D8 D8 BA
88F5:C1 B5 A0
88F8:A0 A0 A0
88FB:A0
88FC:D8 D8 D8  593 SRD     ASC  /XXXXX:10 XXXXX:11 XXXXX:12 XXXXX:13     /
88FF:D8 D8 BA
8902:B1 B0 A0
8905:D8 D8 D8
8908:D8 D8 BA
890B:B1 B1 A0
890E:D8 D8 D8
8911:D8 D8 BA
8914:B1 B2 A0
8917:D8 D8 D8
891A:D8 D8 BA
891D:B1 B3 A0
8920:A0 A0 A0
8923:A0
8924:D8 D8 D8  594 SRE     ASC  /XXXXX:14 XXXXX:1F XXXXX:24 XXXXX:25     /
8927:D8 D8 BA
892A:B1 B4 A0
892D:D8 D8 D8
8930:D8 D8 BA
8933:B1 C6 A0
8936:D8 D8 D8
8939:D8 D8 BA
893C:B2 B4 A0
893F:D8 D8 D8
8942:D8 D8 BA
8945:B2 B5 A0
8948:A0 A0 A0
894B:A0
894C:D8 D8 D8  595 SRF     ASC  /XXXXX:26 XXXXX:27 XXXXX:28 XXXXX:29     /
894F:D8 D8 BA
8952:B2 B6 A0
8955:D8 D8 D8
8958:D8 D8 BA
895B:B2 B7 A0
895E:D8 D8 D8
8961:D8 D8 BA
8964:B2 B8 A0
8967:D8 D8 D8
896A:D8 D8 BA
896D:B2 B9 A0
8970:A0 A0 A0
8973:A0
8974:D8 D8 D8  596 SR10    ASC  /XXXXX:2A XXXXX:2B XXXXX:2C XXXXX:2D     /
8977:D8 D8 BA
897A:B2 C1 A0
897D:D8 D8 D8
8980:D8 D8 BA
8983:B2 C2 A0
8986:D8 D8 D8
8989:D8 D8 BA
```

```
898C:B2 C3 A0
898F:D8 D8 D8
8992:D8 D8 BA
8995:B2 C4 A0
8998:A0 A0 A0
899B:A0
899C:D8 D8 D8  597 SR11    ASC  /XXXXX:2E XXXXX:2F                        /
899F:D8 D8 BA
89A2:B2 C5 A0
89A5:D8 D8 D8
89A8:D8 D8 BA
89AB:B2 C6 A0
89AE:A0 A0 A0
89B1:A0 A0 A0
89B4:A0 A0 A0
89B7:A0 A0 A0
89BA:A0 A0 A0
89BD:A0 A0 A0
89C0:A0 A0 A0
89C3:A0
89C4:D8 D8 D8  598 SR12    ASC  /XXXXXXX  XXXXX XXX:0000 XXX:00  XXXX=XX  /
89C7:D8 D8 D8
89CA:D8 A0 D8
89CD:D8 D8 D8
89D0:D8 A0 D8
89D3:D8 D8 BA
89D6:B0 B0 B0
89D9:B0 A0 D8
89DC:D8 D8 BA
89DF:B0 B0 A0
89E2:A0 D8 D8
89E5:D8 D8 BD
89E8:D8 D8 A0
89EB:A0
89EC:D8 D8 D8  599 SR13    ASC  /XXXX     XXXX XXXXXXXXX:                  /
89EF:D8 A0 D8
89F2:D8 D8 D8
89F5:A0 D8 D8
89F8:D8 D8 D8
89FB:D8 D8 D8
89FE:D8 BA A0
8A01:A0 A0 A0
8A04:A0 A0 A0
8A07:A0 A0 A0
8A0A:A0 A0 A0
8A0D:A0 A0 A0
8A10:A0 A0 A0
8A13:A0
8A14:D8 D8 D8  600 SR14    ASC  /XXXXX:30 XXXXX:31 XXXXX:32 XXXXX:33     /
8A17:D8 D8 BA
8A1A:B3 B0 A0
8A1D:D8 D8 D8
8A20:D8 D8 BA
8A23:B3 B1 A0
8A26:D8 D8 D8
8A29:D8 D8 BA
8A2C:B3 B2 A0
```

```
8A2F:D8 D8 D8
8A32:D8 D8 BA
8A35:B3 B3 A0
8A38:A0 A0 A0
8A3B:A0
8A3C:D8 D8 D8  601 SR15    ASC  /XXXXX:34 XXXXX:35 XXXXX:36 XXXXX:37     /
8A3F:D8 D8 BA
8A42:B3 B4 A0
8A45:D8 D8 D8
8A48:D8 D8 BA
8A4B:B3 B5 A0
8A4E:D8 D8 D8
8A51:D8 D8 BA
8A54:B3 B6 A0
8A57:D8 D8 D8
8A5A:D8 D8 BA
8A5D:B3 B7 A0
8A60:A0 A0 A0
8A63:A0
8A64:D8 D8 D8  602 SR16    ASC  /XXXXX:38 XXXXX:39 XXXXX:3A XXXXX:3B     /
8A67:D8 D8 BA
8A6A:B3 B8 A0
8A6D:D8 D8 D8
8A70:D8 D8 BA
8A73:B3 B9 A0
8A76:D8 D8 D8
8A79:D8 D8 BA
8A7C:B3 C1 A0
8A7F:D8 D8 D8
8A82:D8 D8 BA
8A85:B3 C2 A0
8A88:A0 A0 A0
8A8B:A0
8A8C:D8 D8 D8  603 SR17    ASC  /XXXXX:3C XXXXX:3D XXXXX:3E XXXXX:3F     /
8A8F:D8 D8 BA
8A92:B3 C3 A0
8A95:D8 D8 D8
8A98:D8 D8 BA
8A9B:B3 C4 A0
8A9E:D8 D8 D8
8AA1:D8 D8 BA
8AA4:B3 C5 A0
8AA7:D8 D8 D8
8AAA:D8 D8 BA
8AAD:B3 C6 A0
8AB0:A0 A0 A0
8AB3:A0
8AB4:          604 *
8AB4:          605 *
8AB4:86 87 87  606 FHLIST  DFB  <SR0,<SR1,<SR2,<SR3,<SR4,<SR5
8AB7:87 87 87
8ABA:87 88 88  607         DFB  <SR6,<SR7,<SR8,<SR9,<SRA,<SRB
8ABD:88 88 88
8AC0:88 88 89  608         DFB  <SRC,<SRD,<SRE,<SRF,<SR10,<SR11
8AC3:89 89 89
8AC6:89 89 8A  609         DFB  <SR12,<SR13,<SR14,<SR15,<SR16,<SR17
8AC9:8A 8A 8A
```

```
8ACC:           610 *
8ACC:F4 1C 44   611 FLLIST  DFB  >SR0,>SR1,>SR2,>SR3,>SR4,>SR5
8ACF:6C 94 BC
8AD2:E4 0C 34   612         DFB  >SR6,>SR7,>SR8,>SR9,>SRA,>SRB
8AD5:5C 84 AC
8AD8:D4 FC 24   613         DFB  >SRC,>SRD,>SRE,>SRF,>SR10,>SR11
8ADB:4C 74 9C
8ADE:C4 EC 14   614         DFB  >SR12,>SR13,>SR14,>SR15,>SR16,>SR17
8AE1:3C 64 8C
8AE4:           615 *
8AE4:           616 *

*** SUCCESSFUL ASSEMBLY: NO ERRORS
```

80F9 ADRCNV
815D CKY1
90 CROW
821F DSPCHR
8ACC FLLIST
825B GVAL1
80C1 INCR1
C000 KBD
8187 MDN1
8024 MLP2
8165 MOVUP
82C0 NOCHNG
803A NTDWN
81ED NUMFLD
20 RDCMD
80F0 RDR1
92 RRCOL
10 SACMD
82E0 SL1
8202 SPECCK
89C4 SR12
899C SR11
8744 SR2
87E4 SR6
8884 SRA
8924 SRE
96 TMP1
40 WRCMD
91 CCOL
831C CLLIST
81B8 CUPDTE
8E FBASE
86 FSTAT
811D HEXKEY
80B5 INCRR
C010 KSTRB
81B3 MLF1
817A MOVDWN
819D MRT0
8130 NOHEX
8044 NTLFT
94 OLDCCL
97 RDHGH
80E5 RDREG
80CC SA1
8334 SCREEN
82EC SL2
8072 SPECR
89EC SR13
8A3C SR15
876C SR3
880C SR7
88AC SRB
894C SRF
827D TOB1
8283 WTR1
8156 CHKKEY
81E0 CRADDR
822D DOWRT
88 FDATA
823D GETFLD
8126 HK1
82CF INFLD
99 LSTKEY
8003 MLOOP
81A6 MOVLFT
819F MRT1
?8200 NONUM
804E NTRGH
93 OLDCR
98 RDLOW
85 RLOC
80D4 SA2
80CA SETADR
82D6 SLOAD
86F4 SR0
8A14 SR14
8A64 SR16
8794 SR4
8834 SR8
88D4 SRC
78000 START
8275 TOBIN
828B WTR2
8304 CHLIST
8C CRBASE
8132 DSPBYT
8AB4 FHLIST
8251 GETVAL
82B0 ILP
8293 INIT
9A LSTRD
800B MLP1
818E MOVRGH
8173 MUP1
821D NOSPEC
8030 NTUP
8A RBASE
80E8 RDR0
95 ROW
80DC SA3
824A SETREG
821B SPDONE
8974 SR10
871C SR1
8A8C SR17
87BC SR5
885C SR9
88FC SRD
8152 TH1
814A TOHEX
8280 WTREG

10 SACMD 20 RDCMD 40 WRCMD

85 RLOC

86 FSTAT	88 FDATA	8A RBASE	8C CRBASE
8E FBASE	90 CROW	91 CCOL	92 RRCOL
93 OLDCR	94 OLDCCL	95 ROW	96 TMP1
97 RDHGH	98 RDLOW	99 LSTKEY	9A LSTRD
?8000 START	8003 MLOOP	800B MLP1	8024 MLP2
8030 NTUP	803A NTDWN	8044 NTLFT	804E NTRGH
8072 SPECR	80B5 INCRR	80C1 INCR1	80CA SETADR
80CC SA1	80D4 SA2	80DC SA3	80E5 RDREG
80E8 RDR0	80F0 RDR1	80F9 ADRCNV	811D HEXKEY
8126 HK1	8130 NOHEX	8132 DSPBYT	814A TOHEX
8152 TH1	8156 CHKKEY	815D CKY1	8165 MOVUP
8173 MUP1	817A MOVDWN	8187 MDN1	818E MOVRGH
819D MRT0	819F MRT1	81A6 MOVLFT	81B3 MLF1
81B8 CUPDTE	81E0 CRADDR	81ED NUMFLD	?8200 NONUM
8202 SPECCK	821B SPDONE	821D NOSPEC	821F DSPCHR
822D DOWRT	823D GETFLD	824A SETREG	8251 GETVAL
825B GVAL1	8275 TOBIN	827D TOB1	8280 WTREG
8283 WTR1	828B WTR2	8293 INIT	82B0 ILP
82C0 NOCHNG	82CF INFLD	82D6 SLOAD	82E0 SL1
82EC SL2	8304 CHLIST	831C CLLIST	8334 SCREEN
86F4 SR0	871C SR1	8744 SR2	876C SR3
8794 SR4	87BC SR5	87E4 SR6	880C SR7
8834 SR8	885C SR9	8884 SRA	88AC SRB
88D4 SRC	88FC SRD	8924 SRE	894C SRF
8974 SR10	899C SR11	89C4 SR12	89EC SR13
8A14 SR14	8A3C SR15	8A64 SR16	8A8C SR17
8AB4 FHLIST	8ACC FLLIST	C000 KBD	C010 KSTRB

```
SOURCE FILE: EXPLORER
0000:          1 * EXPLORER -- A FROBCO PROGRAM TO EXPLORE THE OPERATION OF THE VCS GAME PLAYER
0000:          2 * COPYRIGHT 1982 BY FROBCO -- ALL RIGHTS RESERVED
0000:          3 *
0000:          4 * VERSION 1.1 -- LAST MODIFIED 10/21/82
0000:          5 *
0000:          6 *
0000:          7 * SYSTEM DEFINITIONS
0000:          8 *
0000:          9 VSYNCH  EQU  0        ; WRITE A 2 HERE TO CAUSE VERTICAL SYNCH PULSE (ONCE PER
0001:         10 VRESET  EQU  1        ; RESET VIDEO ONCE PER SCREEN
0002:         11 LWAIT   EQU  2        ; WRITE ANYTHING HERE TO WAIT TO END OF CURRENT LINE
0003:         12 VID03   EQU  3
0004:         13 P1VMOD  EQU  4        ; PLAYER1 VIDEO MODE REGISTER
0005:         14 P2VMOD  EQU  5        ; PLAYER2 VIDEO MODE REGISTER
0006:         15 P1COLOR EQU  6        ; PLAYER 1 COLOR REGISTER
0007:         16 P2COLOR EQU  7        ; PLAYER 2 COLOR REGISTER
0008:         17 OCOLOR  EQU  8        ; OBJECT COLOR REGISTER
0009:         18 BCOLOR  EQU  9        ; BACKGROUND COLOR REGISTER
000A:         19 FVMODE  EQU  $0A      ; FIELD VIDEO MODE REGISTER
000B:         20 P1FLIP  EQU  $0B      ; PLAYER1 FLIP REGISTER & READ FIRE BUTTON
000C:         21 P2FLIP  EQU  $0C      ; PLAYER2 FLIP REGISTER & READ FIRE BUTTON
000D:         22 FLDAIM  EQU  $0D      ; OBJECT FIELD A IMAGE REGISTER
000E:         23 FLDBIM  EQU  $0E      ; OBJECT FIELD B IMAGE REGISTER
000F:         24 FLDCIM  EQU  $0F      ; OBJECT FIELD C IMAGE REGISTER
0010:         25 P1HRES  EQU  $10      ; WRITE HERE TO RESET PLAYER1 HORZ POS TO 0
0011:         26 P2HRES  EQU  $11      ; WRITE HERE TO RESET PLAYER2 HORZ POS TO 0
0012:         27 S1HRES  EQU  $12      ; WRITE HERE TO RESET PLAYER1 SHOT HORZ POS TO 0
0013:         28 S2HRES  EQU  $13      ; WRITE HERE TO RESET PALYER2 SHOT HORZ POS TO 0
0014:         29 VID14   EQU  $14
0015:         30 SND1MD  EQU  $15      ; SOUND MODE REGISTER FOR PLAYER 1
0016:         31 SND2MD  EQU  $16      ; SOUND MODE REGISTER FOR PLAYER 2
0017:         32 SND1TN  EQU  $17      ; SOUND TONE REGISTER FOR PLAYER 1
0018:         33 SND2TN  EQU  $18      ; SOUND TONE REGISTER FOR PLAYER 2
0019:         34 SND1AM  EQU  $19      ; SOUND AMPLITURE REGISTER FOR PLAYER 1
001A:         35 SND2AM  EQU  $1A      ; SOUND AMPLITUDE REGISTER FOR PLAYER 2
001B:         36 P1IMAG  EQU  $1B      ; VIDEO IMAGE FOR PLAYER 1
001C:         37 P2IMAG  EQU  $1C      ; VIDEO IMAGE FOR PLAYER 2
001D:         38 P1SHOT  EQU  $1D      ; WRITE 0 HERE TO ENABLE SHOT IMAGE THIS LINE
001E:         39 P2SHOT  EQU  $1E      ; SAME AS ABOVE BUT FOR PLAYER 2
001F:         40 VID1F   EQU  $1F
0020:         41 P1HORZ  EQU  $20      ; HORIZONTAL SPEED PLAYER 1
0021:         42 P2HORZ  EQU  $21      ; HORIZONTAL SPEED PLAYER 2
0022:         43 S1HORZ  EQU  $22      ; HORIZONTAL SPEED OF SHOT 1
0023:         44 S2HORZ  EQU  $23      ; HORIZONTAL SPEED OF SHOT 2
0024:         45 VID24   EQU  $24
0025:         46 VID25   EQU  $25
0026:         47 VID26   EQU  $26
0027:         48 VID27   EQU  $27
0028:         49 S1CONT  EQU  $28      ; SHOT 1 CONTROL REGISTER
0029:         50 S2CONT  EQU  $29      ; SHOT 2 CONTROL REGISTER
002A:         51 HZSCRL  EQU  $2A      ; WRITE HERE TO ADVANCE PLAYER AND SHOT HORZ SCROLE
002B:         52 VID2B   EQU  $2B
002C:         53 COLRES  EQU  $2C      ; WRITE HERE TO RESET COLLISION REGISTERS
002D:         54 VID2D   EQU  $2D
```

```
002E:          55 VID2E   EQU  $2E
002F:          56 VID2F   EQU  $2F
0030:          57 PS1COL  EQU  $30       ; BIT 7 SET IF S1+P2, BIT 0 SET IF S1+P1
0031:          58 PS2COL  EQU  $31       ; BIT 7 SET IF S2+P1, BIT 0 SET IF S2+P2
0032:          59 P1OCOL  EQU  $32       ; BIT 7 COMES BACK SET IF PLAYER 1 COLLIDES WITH OBJECTS
0033:          60 P2OCOL  EQU  $33       ; BIT 7 COMES BACK SET IF PLAYER 2 COLLIDES WITH OBJECTS
0034:          61 VID34   EQU  $34       ; BIT 7 SET IF SHOT1 HITS OBJECT
0035:          62 VID35   EQU  $35       ; BIT 7 SET IF SHOT2 HITS OBJECT
0036:          63 VID36   EQU  $36
0037:          64 PPCOL   EQU  $37       ; BIT 7 COMES BACK SET IF PLAYER 1 COLLIDES WITH PLAYER 2
0038:          65 VID38   EQU  $38
0039:          66 VID39   EQU  $39
003A:          67 VID3A   EQU  $3A
003B:          68 VID3B   EQU  $3B
003C:          69 LFTFR   EQU  $3C       ; BIT 7 CLEAR IF LEFT FIRE BUTTON DOWN
003D:          70 RGHFR   EQU  $3D       ; BIT 7 CLEAR IF RIGHT FIRE BUTTON DOWN
003E:          71 VID3E   EQU  $3E
003F:          72 VID3F   EQU  $3F
0000:          73 *
0000:          74 *
0000:          75 * NOW THE 6532 REGISTERS
0000:          76 *
0280:          77 CTROLS  EQU  $280      ; READ THIS FOR THE JOY STICK BITS
0281:          78 DDRA    EQU  $281      ; DATA DIRECTION REGISTER FOR THE ABOVE PORT
0282:          79 SWTCHS  EQU  $282      ; READ THIS FOR THE CONSOL SWITCH BITS
0283:          80 DDRB    EQU  $283      ; DATA DIRECTION REGISTER FOR THE ABOVE PORT
0284:          81 RTIME   EQU  $284      ; READ THE TIMER
0000:          82 *
0294:          83 TIM1    EQU  $294      ; WRITE HERE TO SET TIMER FOR 1/1
0295:          84 TIM8    EQU  $295      ; WRITE HERE TO SET TIMER FOR 1/8
0296:          85 TIM64   EQU  $296      ; WRITE HERE TO SET TIMER FOR 1/64
0297:          86 TIM1024 EQU  $297      ; WRITE HERE TO SET TIMER FOR 1/1024
0000:          87 *
0000:          88 *
0000:          89 * RAM ASSIGNMENTS
0000:          90 *
0080:          91 SCREENS EQU  $80       ; SCREEN COUNTER GOING FROM 0 TO 59
0081:          92 SECONDS EQU  SCREENS+1 ; SECOND COUNTER INCREMENTED EVERY 60 SCREENS
0082:          93 TARL    EQU  SECONDS+1 ; LOW ORDER INDIRECT USED BY FMON
0083:          94 TARH    EQU  TARL+1    ; HIGH ORDER INDERECT USED BY FMON
0084:          95 COMCOD  EQU  TARH+1    ; COMMAND CODE HOLDING REGISTER USED BY FMON
0085:          96 VLNCT   EQU  COMCOD+1  ; VERTICAL LINE COUNTER
0086:          97 SPREG   EQU  VLNCT+1   ; TEMP STORE USED FOR THE STACK POINTER
0087:          98 DIGCON  EQU  SPREG+1   ; DIGIT CONTROL REGISTER
0088:          99 TOPCON  EQU  DIGCON+1  ; CONTROL REG FOR THE TOP OF SCREEN
0089:         100 P1HS    EQU  TOPCON+1  ; HORIZONTAL POSITON OF PLAYER1
008A:         101 P2HS    EQU  P1HS+1    ; HORIZONTAL POSITION OF PLAYER2
008B:         102 S1HS    EQU  P2HS+1    ; HORIZONTAL POSITION OF PLAYER 1 SHOT
008C:         103 S2HS    EQU  S1HS+1    ; HORIZONTAL POSITION OF PLAYER 2 SHOT
008D:         104 TREG1   EQU  S2HS+1    ; THE TREG GROUP IS USED TO HOLD IMAGES TO BE DISPLAYED
0093:         105 TREG2   EQU  TREG1+6   ; AT THE TOP OF THE SCREEN.
0099:         106 TREG3   EQU  TREG2+6
009F:         107 TREG4   EQU  TREG3+6
00A5:         108 SMODE   EQU  TREG4+6   ; SCREEN MODE REGISTER
00A6:         109 MODE1   EQU  SMODE+1   ; VID MODE 1 REGISTER
00A7:         110 MODE2   EQU  MODE1+1   ; VID MODE 2 REGISTER
00A8:         111 EBITS   EQU  MODE2+1   ; TEMP STORAGE FOR THE ENABLE BITS
```

```
00A9:          112 TMCNT   EQU  EBITS+1   ; IN BETWEEN SCREEN TIME COUNT
00AA:          113 SPTS    EQU  TMCNT+1   ; PARTS OF THE SCREEN FUNCTIONS TO DO
00AB:          114 P1CR    EQU  SPTS+1    ; PLAYER1 COLOR (TOP 4 BITS CHROMA LOWER 4 BITS LUMA)
00AC:          115 P2CR    EQU  P1CR+1    ; PLAYER2 COLOR
00AD:          116 BKCR    EQU  P2CR+1    ; BACKGROUND COLOR
00AE:          117 OBJCR   EQU  BKCR+1    ; OBJECT COLOR
00AF:          118 P1VERT  EQU  OBJCR+1   ; PLAYER1 VERTICAL POSITION REGISTER
00B0:          119 P2VERT  EQU  P1VERT+1  ; PLAYER2 VERTICAL POSITION REGISTER
00B1:          120 S1VERT  EQU  P2VERT+1  ; SHOT 1 VERTICAL REGISTER
00B2:          121 S2VERT  EQU  S1VERT+1  ; SHOT 2 VERTICAL POSITION REGISTER
00B3:          122 P1IMG   EQU  S2VERT+1  ; PLAYER1 IMAGE ARRAY
00BB:          123 P2IMG   EQU  P1IMG+8   ; PLAYER2 IMAGE ARRAY
00C3:          124 OBJAVT  EQU  P2IMG+8   ; OBJECT A VERTICAL POSITION
00C4:          125 OBJAIM  EQU  OBJAVT+1  ; OBJECT A IMAGE ARRAY
00CC:          126 OBJBVT  EQU  OBJAIM+8  ; OBJECT B VERTICAL POSITION
00CD:          127 OBJBIM  EQU  OBJBVT+1  ; OBJECT B IMAGE ARRAY
00D5:          128 OBJCVT  EQU  OBJBIM+8  ; OBJECT C VERTICAL POSITION
00D6:          129 OBJCIM  EQU  OBJCVT+1  ; OBJECT C IMAGE ARRAY
0000:          130 *
0000:          131 *
0000:          132 *
0000:          133 * START OF PROGRAM - LOAD AT $F000
----- NEXT OBJECT FILE NAME IS EXPLORER.OBJ0
F000:          134         ORG  $F000
F000:          135 *
F000:A2 FF     136 START   LDX  #$FF      ; HERE ON RESET SO SET UP THE STACK POINTER
F002:9A        137         TXS
F003:D8        138         CLD              ; CLEAR DECIMAL MODE
F004:A2 00     139         LDX  #0        ; DO A MASSIVE MEMORY CLEAR
F006:A9 00     140         LDA  #0
F008:95 00     141 CLOOP   STA  0,X
F00A:CA        142         DEX
F00B:D0 FB     143         BNE  CLOOP
F00D:          144 *
F00D:          145 *
F00D:          146 *     MAIN RAM INITIALIZATION
F00D:          147 *
F00D:A9 03     148         LDA  #3        ; SET UP FOR  PLAYER1 AND PLAYER2 DISPLAY AT FIRST
F00F:85 AA     149         STA  SPTS
F011:A9 02     150         LDA  #2        ; SET UP TOPCON TO DO THE DISPLAY OF TOP SECTION
F013:85 88     151         STA  TOPCON
F015:A9 00     152         LDA  #0        ; INIT THE HORZ POSITION SHADOW REGISTER FOR P1, P2, AND SHOT
F017:85 89     153         STA  P1HS
F019:85 8A     154         STA  P2HS
F01B:85 8B     155         STA  S1HS
F01D:85 8C     156         STA  S2HS
F01F:          157 *
F01F:A9 14     158         LDA  #20       ; SET UP PLAYER1  VERTICAL
F021:85 AF     159         STA  P1VERT
F023:A9 1E     160         LDA  #30       ; SET UP PLAYER2 VERTICAL
F025:85 B0     161         STA  P2VERT
F027:A9 0E     162         LDA  #14       ; SET UP PLAYER1 SHOT VERTICAL
F029:85 B1     163         STA  S1VERT
F02B:A9 12     164         LDA  #18       ; SET UP PLAYER2 SHOT VERTICAL
F02D:85 B2     165         STA  S2VERT
F02F:A9 2D     166         LDA  #45       ; SET UP OBJECT A VERITCAL
F031:85 C3     167         STA  OBJAVT
```

```
F033:A9 37     168          LDA  #55       ; SET UP OBJECT B VERTICAL
F035:85 CC     169          STA  OBJBVT
F037:A9 41     170          LDA  #65       ; SET UP OBJECT C VERTICAL
F039:85 D5     171          STA  OBJCVT
F03B:          172 *
F03B:A9 FE     173          LDA  #$FE      ; MAKE PLAYER IMAGES
F03D:85 B3     174          STA  P1IMG
F03F:85 BB     175          STA  P2IMG
F041:A9 7F     176          LDA  #$7F
F043:85 B4     177          STA  P1IMG+1
F045:85 BC     178          STA  P2IMG+1
F047:A9 63     179          LDA  #$63
F049:85 B5     180          STA  P1IMG+2
F04B:85 BD     181          STA  P2IMG+2
F04D:A9 62     182          LDA  #$62
F04F:85 B6     183          STA  P1IMG+3
F051:85 BE     184          STA  P2IMG+3
F053:A9 7C     185          LDA  #$7C
F055:85 B7     186          STA  P1IMG+4
F057:85 BF     187          STA  P2IMG+4
F059:A9 78     188          LDA  #$78
F05B:85 B8     189          STA  P1IMG+5
F05D:85 C0     190          STA  P2IMG+5
F05F:A9 60     191          LDA  #$60
F061:85 B9     192          STA  P1IMG+6
F063:85 C1     193          STA  P2IMG+6
F065:A9 F2     194          LDA  #$F2
F067:85 BA     195          STA  P1IMG+7
F069:A9 F6     196          LDA  #$F6
F06B:85 C2     197          STA  P2IMG+7
F06D:          198 *
F06D:          199 *
F06D:4C A0 F0  200          JMP  TINIT     ; JUMP AROUND THIS DATA
F070:          201 *
F070:FE 62 F8  202 LETF     DFB  $FE,$62,$F8,$60,$60,$F0
F073:60 60 F0
F076:3F 66 66  203 LETR     DFB  $3F,$66,$66,$3E,$26,$4F
F079:3E 26 4F
F07C:7E FF C3  204 LETO     DFB  $7E,$FF,$C3,$C3,$FF,$7E
F07F:C3 FF 7E
F082:3F 26 7E  205 LETB     DFB  $3F,$26,$7E,$7E,$26,$3F
F085:7E 26 3F
F088:          206 *
F088:18 24 42  207 LETA     DFB  $18,$24,$42,$42,$7E,$42,$42,$CE
F08B:42 7E 42
F08E:42 CE
F090:FE 61 61  208 LETB8    DFB  $FE,$61,$61,$7E,$7E,$61,$61,$FE
F093:7E 7E 61
F096:61 FE
F098:7E FF FE  209 LETC     DFB  $7E,$FF,$FE,$F0,$F0,$FE,$FF,$7E
F09B:F0 F0 FE
F09E:FF 7E
F0A0:          210 *
F0A0:          211 *   LOOP TO LOAD IMAGES AT TOP OF SCREEN
F0A0:A2 00     212 TINIT    LDX  #0
F0A2:BD 70 F0  213 TLP      LDA  LETF,X
F0A5:95 8D     214          STA  TREG1,X
```

```
F0A7:E8        215          INX
F0A8:E0 18     216          CPX  #$18
F0AA:D0 F6     217          BNE  TLP
F0AC:          218 *
F0AC:A2 00     219          LDX  #0         ; INITIALIZE OBJECT A IMAGE
F0AE:BD 88 F0  220 AIMLP    LDA  LETA,X
F0B1:95 C4     221          STA  OBJAIM,X
F0B3:E8        222          INX
F0B4:E0 08     223          CPX  #8
F0B6:D0 F6     224          BNE  AIMLP
F0B8:          225 *
F0B8:A2 00     226          LDX  #0         ; INITIALIZE OBJECT B IMAGE
F0BA:BD 90 F0  227 BIMLP    LDA  LETB8,X
F0BD:95 CD     228          STA  OBJBIM,X
F0BF:E8        229          INX
F0C0:E0 08     230          CPX  #8
F0C2:D0 F6     231          BNE  BIMLP
F0C4:          232 *
F0C4:A2 00     233          LDX  #0         ; INITIALIXE OBJECT C IMAGE
F0C6:BD 98 F0  234 CIMLP    LDA  LETC,X
F0C9:95 D6     235          STA  OBJCIM,X
F0CB:E8        236          INX
F0CC:E0 08     237          CPX  #8
F0CE:D0 F6     238          BNE  CIMLP
F0D0:          239 *
F0D0:A9 82     240          LDA  #$82       ; SET UP PLAYER1 COLOR
F0D2:85 AB     241          STA  P1CR
F0D4:A9 DA     242          LDA  #$DA       ; SET UP PLAYER2 COLOR
F0D6:85 AC     243          STA  P2CR
F0D8:A9 EA     244          LDA  #$EA       ; SET UP BACKGROUND COLOR
F0DA:85 AD     245          STA  BKCR
F0DC:A9 64     246          LDA  #$64       ; SET UP COLOR FOR OBJECTS
F0DE:85 AE     247          STA  OBJCR
F0E0:A9 05     248          LDA  #5         ; SET UP THE SCREEN MODE PARAMETER
F0E2:85 A5     249          STA  SMODE
F0E4:A9 02     250          LDA  #2         ; SET UP DIGCON
F0E6:85 B7     251          STA  DIGCON
F0E8:A9 2B     252          LDA  #$2B       ; NUMBER OF TIMER CLICKS BETWEEN SCREENS
F0EA:85 A9     253          STA  TMCNT
F0EC:          254 *
F0EC:          255 *     MAIN PROGRAM LOOP
F0EC:          256 *
F0EC:          257 *
F0EC:          258 *
F0EC:20 42 F2  259 MAIN     JSR  SYNCH      ; DO THE VERTICAL SYNCH
F0EF:20 6C F2  260          JSR  FMON       ; GO SEE IF USER HAS SOMETHING TO DO
F0F2:20 FB F0  261          JSR  SSETUP     ; SET UP THE ONCE/SCREEN REGISTERS
F0F5:20 14 F1  262          JSR  DISPLAY    ; DO THE LINE BY LINE DISPLAY
F0F8:4C EC F0  263          JMP  MAIN       ; LOOP ALWAYS
F0FB:          264 *
F0FB:          265 *
F0FB:          266 SSETUP  EQU  *
F0FB:A5 89     267          LDA  P1HS       ; PUT HORIZONTAL SHADOW REGISTERS INTO
F0FD:85 20     268          STA  P1HORZ     ; THE VIDEO REGISTERS
F0FF:A5 8A     269          LDA  P2HS
F101:85 21     270          STA  P2HORZ
F103:A5 8B     271          LDA  S1HS
```

```
F105:85 22     272         STA  S1HORZ
F107:A5 8C     273         LDA  S2HS
F109:85 23     274         STA  S2HORZ
F10B:          275 *
F10B:A5 AD     276         LDA  BKCR      ; GET THE BACKGROUND COLOR
F10D:85 09     277         STA  BCOLOR
F10F:A5 AE     278         LDA  OBJCR     ; UPDATE THE OBJECT COLOR
F111:85 08     279         STA  OCOLOR
F113:          280 *
F113:          281 *       DONE WITH THE ONCE PER SCREEN STUFF
F113:60        282         RTS
F114:          283 *
F114:          284 *
F114:          285 *       DISPLAY
F114:          286 *
F114:          287 * THE DISPLAY ROUTINE IS CALLED ONCE PER SCREEN TO DO A LINE BY LINE UPDATE
F114:          288 * OF THE VIDEO REGISTERS.
F114:          289 *
F114:          290 *
F114:          291 * NOTE: THIS LOOP LETS TWO HORIZONTAL SYNCH PULSES GO BEFORE
F114:          292 * INCREMENT OF VERTICAL LINE COUNT.
F114:          293 * THIS CAUSES ALL SCREEEN PATTERNS TO BE TWO LINES HIGH FOR EACH LOOP
F114:          294 * THROUGH DISPLAY.
F114:          295 *
F114:          296 *
F114:          297 *
F114:A9 00     298 DISPLAY LDA  #0        ; RESET THE LINE COUNTER
F116:85 85     299         STA  VLNCT
F118:85 02     300         STA  LWAIT     ; WAIT FOR END OF LINE
F11A:85 2A     301         STA  HZSCRL    ; DONE AT END OF LINE THIS CAUSES THE
F11C:          302 *                     HORIZONTAL REGISTERS TO BE UPDATED FOR MOTION
F11C:          303 *
F11C:AD 84 02  304 TWAIT   LDA  RTIME     ; WAIT FOR THE TIMER TO GET TO ZERO
F11F:D0 FB     305         BNE  TWAIT
F121:          306 *
F121:85 02     307         STA  LWAIT
F123:85 01     308         STA  VRESET    ; TURN SCREEN ON
F125:          309 *
F125:BA        310         TSX            ; SAVE STACK POINTER SO WE CAN USE A PHP INSTRUCTION TO GET
F126:          311 *                 FLAG REGISTER BELOW
F126:86 86     312         STX  SPREG
F128:A5 87     313         LDA  DIGCON    ; GET THE DIGIT CONTROL BYTE
F12A:85 0A     314         STA  FVMODE    ; SEND IT TO THE VIDEO
F12C:A6 88     315         LDX  TOPCON    ; GET THE CONTROL BYTE FOR THE TOP OF SCREEN
F12E:85 02     316 TOPWT   STA  LWAIT     ; WAIT THAT MANY LINES
F130:CA        317         DEX
F131:D0 FB     318         BNE  TOPWT
F133:          319 *
F133:          320 * DISPLAY TOP PART OF SCREEN
F133:          321 *
F133:A2 00     322         LDX  #0        ; DO A SIX LINE LOOP
F135:          323 *
F135:85 02     324 TOPLP   STA  LWAIT     ; WAIT FOR EOL
F137:B5 8D     325         LDA  TREG1,X   ; FIRST IMAGE BLOCK
F139:85 0E     326         STA  FLDBIM    ; PUT ON SCREEN
F13B:B5 93     327         LDA  TREG2,X   ; NEXT FIELD
F13D:85 0F     328         STA  FLDCIM
```

```
F13F:EA       329         NOP              ; WAIT A WHILE
F140:EA       330         NOP
F141:EA       331         NOP
F142:EA       332         NOP
F143:EA       333         NOP
F144:EA       334         NOP
F145:EA       335         NOP
F146:EA       336         NOP
F147:EA       337         NOP
F148:B5 99    338         LDA  TREG3,X
F14A:85 0E    339         STA  FLDBIM    ; PUT UP NEXT FIELD
F14C:B5 9F    340         LDA  TREG4,X
F14E:85 0F    341         STA  FLDCIM    ; PUT IN 4TH FIELD
F150:         342 *
F150:85 02    343         STA  LWAIT     ; WAIT FOR THE NEXT LINE
F152:B5 8D    344         LDA  TREG1,X   ; GET NEXT IMAGE
F154:85 0E    345         STA  FLDBIM    ; PUT IT UP
F156:B5 93    346         LDA  TREG2,X   ; NOW SECOND FIELD AGAIN
F158:85 0F    347         STA  FLDCIM
F15A:EA       348         NOP              ; WAIT THE SAME WHILE
F15B:EA       349         NOP
F15C:EA       350         NOP
F15D:EA       351         NOP
F15E:EA       352         NOP
F15F:EA       353         NOP
F160:EA       354         NOP
F161:EA       355         NOP
F162:EA       356         NOP
F163:B5 99    357         LDA  TREG3,X
F165:85 0E    358         STA  FLDBIM    ; PART 3
F167:B5 9F    359         LDA  TREG4,X   ; GET PART 4
F169:85 0F    360         STA  FLDCIM    ; PUT UP LAST PART
F16B:         361 *
F16B:E8       362         INX              ; MOVE TO NEXT GROUP
F16C:E0 06    363         CPX  #6
F16E:D0 C5    364         BNE  TOPLP
F170:         365 *
F170:A9 00    366 PART2   LDA  #0        ; CLEAR OUT VID REGISTER
F172:85 02    367         STA  LWAIT     ; WAIT FOR LINE TO END
F174:85 0E    368         STA  FLDBIM
F176:85 0F    369         STA  FLDCIM
F178:         370 *
F178:A5 A5    371         LDA  SMODE     ; GET THE SCREEN MODE PARAMETER
F17A:85 0A    372         STA  FVMODE    ; WHAT?
F17C:A5 AB    373         LDA  P1CR      ; GET THE COLOR FOR PLAYER 1
F17E:85 06    374         STA  P1COLOR
F180:A5 AC    375         LDA  P2CR      ; GET THE COLOR FOR PLAYER 2
F182:85 07    376         STA  P2COLOR
F184:A5 AA    377         LDA  SPTS      ; GET ENABLE BITS
F186:85 A8    378         STA  EBITS     ; FEED AS PARAMETERS TO NEXT SECTION
F188:10 08    379         BPL  SLOOP     ; SHOW EACH SCREEN
F18A:A5 80    380         LDA  SCREENS   ; ELSE ONLY ENABLE EVERY OTHER SCREEN
F18C:29 01    381         AND  #1
F18E:D0 02    382         BNE  SLOOP
F190:85 A8    383         STA  EBITS     ; CLEAR ENABLE BITS
F192:         384 *
F192:         385 *
```

```
F192:A2 1E     386 SLOOP  LDX  #P2SHOT  ; USE THE STACK MECH TO SEND OUT SHOTS
F194:9A        387        TXS
F195:          388 *
F195:85 02     389        STA  LWAIT    ; THIS IS A LINE BY LINE LOOP
F197:          390 *
F197:A5 B2     391        LDA  S2VERT   ; SEE IF TIME TO DISPLAY SHOT
F199:45 85     392        EOR  VLNCT
F19B:08        393        PHP           ; THIS WILL DISPLAY IF Z FLAG SET
F19C:A5 B1     394        LDA  S1VERT
F19E:45 85     395        EOR  VLNCT
F1A0:08        396        PHP
F1A1:          397 *
F1A1:A5 A8     398        LDA  EBITS    ; SEE IF WE SHOULD DO THIS SECTION
F1A3:29 01     399        AND  #1
F1A5:F0 14     400        BEQ  P2TST    ; NOT IF THE BIT IS ZERO
F1A7:38        401        SEC           ; TEST FOR VERT CLOSE TO PLAYER 1
F1A8:A5 85     402        LDA  VLNCT    ; GET THE VERTICAL POSITION AT THIS LINE
F1AA:E5 AF     403        SBC  P1VERT   ; SUBTRACT THE PLAYER1 VERTICAL POSITION
F1AC:90 04     404        BCC  NOP1     ; IF CARRY CLEAR THEN WE ARE NOT THERE YET
F1AE:C9 08     405        CMP  #8       ; A NOW HOLDS THE INDEX TO THE IMAGE ARRAY
F1B0:90 04     406        BCC  INP1     ; IF CARRY CLEAR WE ARE IN RANGE TO SHOW A LINE OF P1 IMAGE
F1B2:A9 00     407 NOP1   LDA  #0       ; IF HERE, NO SHOW
F1B4:F0 03     408        BEQ  P1DSP
F1B6:AA        409 INP1   TAX           ; GO GET LINE OF IMAGE
F1B7:B5 B3     410        LDA  P1IMG,X
F1B9:85 1B     411 P1DSP  STA  P1IMAG   ; DISPLAY PLAYER1 IMAGE
F1BB:          412 *
F1BB:A5 A8     413 P2TST  LDA  EBITS    ; SEE IF WE SHOULD DO SECTION 2
F1BD:29 02     414        AND  #2
F1BF:F0 14     415        BEQ  OATST
F1C1:38        416        SEC           ; GET READY FOR SUBTRACT
F1C2:A5 85     417        LDA  VLNCT    ; GET THE VERTICAL POSITION AT THIS LINE
F1C4:E5 B0     418        SBC  P2VERT   ; SUBTRACT THE PLAYER2 VERTICAL POSITION
F1C6:90 04     419        BCC  NOP2     ; IF CARRY CLEAR THEN WE ARE NOT THERE YET
F1C8:C9 08     420        CMP  #8       ; A NOW HOLDS THE INDEX TO THE IMAGE ARRAY
F1CA:90 04     421        BCC  IMP2     ; IF CARRY CLEAR WE ARE IN RANGE TO SHOW A LINE OF P2 IMAGE
F1CC:A9 00     422 NOP2   LDA  #0       ; IF HERE, NO P2 SHOW
F1CE:F0 03     423        BEQ  P2DSP
F1D0:AA        424 IMP2   TAX           ; GO GET LINE OF IMAGE
F1D1:B5 BB     425        LDA  P2IMG,X
F1D3:85 1C     426 P2DSP  STA  P2IMAG   ; DISPLAY PLAYER2 IMAGE
F1D5:          427 *
F1D5:A5 A8     428 OATST  LDA  EBITS    ; SEE IF WE DO SECTION 3
F1D7:29 04     429        AND  #4
F1D9:F0 14     430        BEQ  OBTST
F1DB:38        431        SEC           ; TEST FOR VERT CLOSE TO OBJECT A
F1DC:A5 85     432        LDA  VLNCT
F1DE:E5 C3     433        SBC  OBJAVT   ; SUBTRACT THE OBJECT A VERTICAL POSITION
F1E0:90 04     434        BCC  NOA      ; IF CLEAR THEN NOT THERE
F1E2:C9 08     435        CMP  #8       ; IF INDEX PAST 8 THEN
F1E4:90 04     436        BCC  INOA     ; KEEP GOING, ELSE DISPLAY
F1E6:A9 00     437 NOA    LDA  #0       ; NOT AT OBJECT A
F1E8:F0 03     438        BEQ  OADSP    ; GO FILL WITH 0
F1EA:AA        439 INOA   TAX           ; GO GET LINE OF IMAGE
F1EB:B5 C4     440        LDA  OBJAIM,X
F1ED:85 0D     441 OADSP  STA  FLDAIM   ; SEND TO OBJECT A
F1EF:          442 *
```

```
F1EF:A5 A8     443 OBTST   LDA  EBITS     ; SEE IF WE DO 4
F1F1:29 08     444         AND  #8
F1F3:F0 14     445         BEQ  OCTST
F1F5:38        446         SEC                  ; TEST FOR VERT CLOSE TO OBJECT B
F1F6:A5 85     447         LDA  VLNCT
F1F8:E5 CC     448         SBC  OBJBVT    ; SUB OBJ B VERTICAL
F1FA:90 04     449         BCC  NOB        ; IF CLEAR, NOT THERE YET
F1FC:C9 08     450         CMP  #8         ; IF > 7, GONE TOO FAR
F1FE:90 04     451         BCC  INOB
F200:A9 00     452 NOB     LDA  #0         ; NOT AT OBJECT B
F202:F0 03     453         BEQ  OBDSP      ; GO FILL WITH 0
F204:AA        454 INOB    TAX               ; GO GET LINE OF IMAGE
F205:B5 CD     455         LDA  OBJBIM,X
F207:85 0E     456 OBDSP   STA  FLDBIM    ; SEND TO OBJECT B
F209:          457 *
F209:A5 AB     458 OCTST   LDA  EBITS     ; SEE IF WE DO 5
F20B:29 10     459         AND  #16
F20D:F0 14     460         BEQ  ENDLP
F20F:38        461         SEC                  ; TEST FOR VERT CLOSE TO OBJECT C
F210:A5 85     462         LDA  VLNCT
F212:E5 D5     463         SBC  OBJCVT    ; SUB OBJ C VERTICAL
F214:90 04     464         BCC  NOC        ; IF CLEAR, NOT THERE YET
F216:C9 08     465         CMP  #8         ; IF > 7, GONE TOO FAR
F218:90 04     466         BCC  INOC
F21A:A9 00     467 NOC     LDA  #0         ; NOT AT OBJECT C
F21C:F0 03     468         BEQ  OCDSP      ; GO FILL WITH 0
F21E:AA        469 INOC    TAX               ; GO GET LINE OF IMAGE
F21F:B5 D6     470         LDA  OBJCIM,X
F221:85 0F     471 OCDSP   STA  FLDCIM    ; SEND TO OBJECT C
F223:          472 *
F223:E6 85     473 ENDLP   INC  VLNCT     ; MOVE TO NEXT LINE
F225:A5 85     474         LDA  VLNCT     ; CHECK TO SEE IF DONE
F227:C9 66     475         CMP  #$66
F229:F0 03     476         BEQ  LPDONE    ; WITH THE MAIN SCREEN LOOP
F22B:4C 92 F1  477         JMP  SLOOP     ; IF NOT GO BACK FOR ANOTHER LINE
F22E:          478 *
F22E:          479 *
F22E:A6 B6     480 LPDONE  LDX  SPREG     ; DONE WITH SCREEN SO PUT THE STACK POINTER BACK
F230:9A        481         TXS
F231:          482 *    NOW SOME END OF PAGE REGISTER STUFFING
F231:A9 00     483         LDA  #0
F233:85 1D     484         STA  P1SHOT
F235:85 1E     485         STA  P2SHOT
F237:85 1B     486         STA  P1IMAG
F239:85 1C     487         STA  P2IMAG
F23B:85 0D     488         STA  FLDAIM
F23D:85 0E     489         STA  FLDBIM
F23F:85 0F     490         STA  FLDCIM
F241:          491 *            DONE WITH DISPLAY
F241:60        492         RTS
F242:          493 *
F242:          494 *
F242:          495 *
F242:          496 *
F242:          497 *
F242:          498 *
F242:          499 *
```

```
F242:E6 80     500 SYNCH   INC  SCREENS   ; UPDATE SCREEN COUNT
F244:A9 3C     501         LDA  #60       ; TEST FOR OVERFLOW INTO SECONDS
F246:45 80     502         EOR  SCREENS
F248:D0 04     503         BNE  SYN1      ; BRANCH IF NO OVERFLOW
F24A:85 80     504         STA  SCREENS   ; ELSE ZERO SCREENS AND
F24C:E6 81     505         INC  SECONDS   ; UPDATE THE SECONDS COUNT
F24E:          506 *
F24E:A9 02     507 SYN1    LDA  #2        ; GET READY FOR VERTICAL SYNCH PULSE GENERATION
F250:85 02     508         STA  LWAIT     ; WAIT FOR END OF LINE
F252:85 01     509         STA  VRESET    ; RESET VIDEO
F254:85 02     510         STA  LWAIT     ; WAIT THREE LINES
F256:85 02     511         STA  LWAIT
F258:85 02     512         STA  LWAIT
F25A:85 00     513         STA  VSYNCH    ; START SYNCH PULSE
F25C:A9 00     514         LDA  #0        ; WRITE 0 TO END PULSE
F25E:85 02     515         STA  LWAIT     ; BUT FIRST WAIT THREE LINES
F260:85 02     516         STA  LWAIT
F262:85 02     517         STA  LWAIT
F264:85 00     518         STA  VSYNCH    ; END PULSE
F266:A5 A9     519         LDA  TMCNT     ; HAVE TIMER COUNT THIS DOWN
F268:8D 96 02  520         STA  TIM64     ; IN DIVIDE BY 64 MODE TO GIVE US
F26B:          521 *   TIME TO DO COME STUFF BEFORE WE NEED TO DISPLAY SCREEN.
F26B:          522 *   WHEN THAT TIME COMES WE WILL BE DISPLAYING LINE BY LINE
F26B:          523 *   AND TIME WILL BE VERY SHORT.
F26B:          524 *
F26B:60        525         RTS            ; DONE WITH VERTICAL SYNCH
F26C:          526 *
F26C:          527 *
F26C:          528 *
F26C:          529 *
F26C:          530 * FMON
F26C:          531 *  COMMANDS
F26C:          532 *
F26C:          533 *    10 - SET ADDRESS (2 BYTE)
F26C:          534 *    20 - READ BYTE
F26C:          535 *    40 - WRITE BYTE
F26C:          536 *
F26C:          537 *  SUB COMMANDS
F26C:          538 *
F26C:          539 *    X0 - DO NOTHING
F26C:          540 *    X1 - POST INCREMENT
F26C:          541 *    X2 - POST DECREMENT
F26C:          542 *
F26C:          543 *  MEMORY USAGE
F26C:          544 *
F26C:          545 *   TARL - TARGET ADDRESS LOW
F26C:          546 *   TARH - TARGET ADDRESS HIGH
F26C:          547 *   COMCOD - COMMAND CODE STORE
F26C:          548 * DECLARATIONS
FFF0:          549 AR1     EQU  $FFF0     ; FMON WRITE REG
FFF1:          550 AR2     EQU  $FFF1     ; FMON STATUS
FFF2:          551 AR3     EQU  $FFF2     ; FMON READ REG
F26C:          552 *
F26C:          553 *
F26C:AD F1 FF  554 FMON    LDA  AR2       ; LOOK TO SEE IF COMMAND WAITING
F26F:29 40     555         AND  #$40
F271:F0 0F     556         BEQ  NOCOM     ; IF NOT JUST EXIT
```

```
F273:AD F2 FF  557          LDA  AR3       ; GET THE COMMAND
F276:85 84     558          STA  COMCOD
F278:29 10     559          AND  #$10
F27A:F0 07     560          BEQ  L1
F27C:20 AD F2  561          JSR  SETADDR
F27F:4C 6C F2  562          JMP  FMON
F282:60        563 NOCOM    RTS              ; IF NO COMMAND JUST RETURN
F283:A5 84     564 L1       LDA  COMCOD
F285:29 20     565          AND  #$20
F287:F0 06     566          BEQ  L2
F289:20 BE F2  567          JSR  READ
F28C:4C 6C F2  568          JMP  FMON
F28F:A5 84     569 L2       LDA  COMCOD
F291:29 40     570          AND  #$40
F293:F0 03     571          BEQ  L3
F295:20 E0 F2  572          JSR  WRITE
F298:4C 6C F2  573 L3       JMP  FMON
F29B:          574 *
F29B:          575 *  READOK
F29B:          576 *
F29B:          577 * CHECK IF IT IS OK TO READ APPLE
F29B:          578 *
F29B:AD F1 FF  579 READOK   LDA  AR2       ; GET STATUS BYTE
F29E:29 40     580          AND  #$40      ; OK TO READ?
F2A0:F0 F9     581          BEQ  READOK    ; NO KEEP LOOKING
F2A2:60        582          RTS             ; YES
F2A3:68        583          PLA
F2A4:60        584 DOREAD   RTS
F2A5:          585 *
F2A5:          586 *
F2A5:AD F1 FF  587 WRITEOK  LDA  AR2
F2A8:29 80     588          AND  #$80
F2AA:F0 F9     589          BEQ  WRITEOK
F2AC:60        590          RTS
F2AD:          591 *
F2AD:          592 * SETADDR
F2AD:          593 *
F2AD:          594 *
F2AD:20 9B F2  595 SETADDR  JSR  READOK
F2B0:AD F2 FF  596          LDA  AR3
F2B3:85 83     597          STA  TARH
F2B5:20 9B F2  598          JSR  READOK
F2B8:AD F2 FF  599          LDA  AR3
F2BB:85 82     600          STA  TARL
F2BD:60        601          RTS
F2BE:          602 *
F2BE:          603 * READ
F2BE:          604 *
F2BE:          605 *
F2BE:20 A5 F2  606 READ     JSR  WRITEOK
F2C1:A0 00     607          LDY  #0
F2C3:B1 82     608          LDA  (TARL),Y
F2C5:8D F0 FF  609          STA  AR1
F2C8:A5 84     610          LDA  COMCOD
F2CA:29 07     611          AND  #$07
F2CC:F0 09     612          BEQ  RL1
F2CE:29 01     613          AND  #$01
```

```
F2D0:F0 08     614          BEQ  RL2
F2D2:C6 82     615          DEC  TARL
F2D4:4C DF F2  616          JMP  RXT
F2D7:4C DF F2  617 RL1      JMP  RXT
F2DA:E6 82     618 RL2      INC  TARL
F2DC:4C DF F2  619          JMP  RXT
F2DF:60        620 RXT      RTS
F2E0:          621 *
F2E0:          622 * WRITE
F2E0:          623 *
F2E0:20 9B F2  624 WRITE    JSR  READOK
F2E3:AD F2 FF  625          LDA  AR3
F2E6:A0 00     626          LDY  #0
F2E8:91 82     627          STA  (TARL),Y
F2EA:A5 84     628          LDA  COMCOD
F2EC:29 07     629          AND  #$07
F2EE:F0 09     630          BEQ  WL1
F2F0:29 01     631          AND  #$01
F2F2:F0 08     632          BEQ  WL2
F2F4:C6 82     633          DEC  TARL
F2F6:4C 01 F3  634          JMP  WXT
F2F9:4C 01 F3  635 WL1      JMP  WXT
F2FC:E6 82     636 WL2      INC  TARL
F2FE:4C 01 F3  637          JMP  WXT
F301:60        638 WXT      RTS
F302:          639 *
F302:          640 *

*** SUCCESSFUL ASSEMBLY: NO ERRORS
```

```
 F0AE AIMLP
   09 BCOLOR
 F008 CLOOP
?0281 DDRA
?F2A4 DOREAD
   0E FLDBIM
   2A HZSCRL
 F21E INOC
 F298 L3
 F098 LETC
?  3C LFTFR
   A6 MODE1
 F282 NOCON
 F1ED OADSP
   C3 OBJAVT
   AE OBJCR
   08 OCOLOR
 F1B9 P1DSP
   89 P1HS
   1D P1SHOT
   AC P2CR
?  11 P2HRES
?  33 P2OCOL
?  05 P2VMOD
?  31 PS2COL
 F2D7 RL1
?  28 S1CONT
   B1 S1VERT
   8C S2HS
 F2AD SETADDR
?  15 SND1MD
?  18 SND2TN
?F000 START
   83 TARH
 0296 TIM64
   A9 TMCNT
   8D TREG1
 F11C TWAIT
?  24 VID24
?  2B VID2B
?  34 VID34
?  39 VID39
?  3F VID3F
 F2F9 WL1
 F301 WXT

 FFF0 AR1
 F0BA BIMLP
?  2C COLRES
?0283 DDRB
   A8 EBITS
   0F FLDCIM
 F1D0 IMP2
 F1B6 INP1
 F088 LETA
 F070 LETF
 F22E LPDONE
   A7 MODE2
 F21A NOC
 F1D5 OATST
   CD OBJBIM
   D5 OBJCVT
 F209 OCTST
?  0B P1FLIP
   1B P1IMAG
   AF P1VERT
 F1D3 P2DSP
   8A P2HS
   1E P2SHOT
?F170 PART2
 F2BE READ
 F2DA RL2
   22 S1HORZ
?  29 S2CONT
   B2 S2VERT
 F192 SLOOP
?  17 SND1TN
   86 SPREG
?0282 SWTCHS
   82 TARL
?0295 TIM8
   88 TOPCON
   93 TREG2
?  03 VID03
?  25 VID25
?  2D VID2D
?  35 VID35
?  3A VID3A
   85 VLNCT
 F2FC WL2

 FFF1 AR2
   AD BKCR
   84 COMCOD
   87 DIGCON
 F223 ENDLP
 F26C FMON
 F1EA INOA
 F283 L1
?F082 LETB
?F07C LETO
   02 LWAIT
 F1E6 NOA
 F1B2 NOP1
 F207 OBDSP
   CC OBJBVT
 F1EF OBTST
   06 P1COLOR
   20 P1HORZ
   B3 P1IMG
?  04 P1VMOD
?  0C P2FLIP
   1C P2IMAG
 F1BB P2TST
?  37 PPCOL
 F29B READOK
 0284 RTIME
?  12 S1HRES
   23 S2HORZ
   80 SCREENS
   A5 SMODE
?  1A SND2AM
   AA SPTS
 F24E SYN1
?0294 TIM1
 F0A0 TINIT
 F135 TOPLP
   99 TREG3
?  14 VID14
?  26 VID26
?  2E VID2E
?  36 VID36
?  3B VID3B
   01 VRESET
 F2E0 WRITE

 FFF2 AR3
 F0C6 CIMLP
?0280 CTROLS
 F114 DISPLAY
   0D FLDAIM
   0A FVMODE
 F204 INOB
 F28F L2
 F090 LETB8
?F076 LETR
 F0EC MAIN
 F200 NOB
 F1CC NOP2
   C4 OBJAIM
   D6 OBJCIM
 F221 OCDSP
   AB P1CR
?  10 P1HRES
?  32 P1OCOL
   07 P2COLOR
   21 P2HORZ
   BB P2IMG
   B0 P2VERT
?  30 PS1COL
?  3D RGHFR
 F2DF RXT
   8B S1HS
?  13 S2HRES
   81 SECONDS
?  19 SND1AM
?  16 SND2MD
 F0FB SSETUP
 F242 SYNCH
?0297 TIM1024
 F0A2 TLP
 F12E TOPWT
   9F TREG4
?  1F VID1F
?  27 VID27
?  2F VID2F
?  38 VID38
?  3E VID3E
   00 VSYNCH
 F2A5 WRITEOK
```

```
                00 VSYNCH       01 VRESET       02 LWAIT      ? 03 VID03
? 04 P1VMOD   ? 05 P2VMOD       06 P1COLOR      07 P2COLOR
  08 OCOLOR     09 BCOLOR       0A FVMODE     ? 0B P1FLIP
? 0C P2FLIP     0D FLDAIM       0E FLDBIM       0F FLDCIM
? 10 P1HRES   ? 11 P2HRES     ? 12 S1HRES   ? 13 S2HRES
? 14 VID14    ? 15 SND1MD     ? 16 SND2MD   ? 17 SND1TN
? 18 SND2TN   ? 19 SND1AM     ? 1A SND2AM     1B P1IMAG
  1C P2IMAG     1D P1SHOT       1E P2SHOT   ? 1F VID1F
  20 P1HORZ     21 P2HORZ       22 S1HORZ     23 S2HORZ
? 24 VID24    ? 25 VID25      ? 26 VID26    ? 27 VID27
? 28 S1CONT   ? 29 S2CONT       2A HZSCRL   ? 2B VID2B
? 2C COLRES   ? 2D VID2D      ? 2E VID2E    ? 2F VID2F
? 30 PS1COL   ? 31 PS2COL     ? 32 P10COL   ? 33 P20COL
? 34 VID34    ? 35 VID35      ? 36 VID36    ? 37 PPCOL
? 38 VID38    ? 39 VID39      ? 3A VID3A    ? 3B VID3B
? 3C LFTFR    ? 3D RGHFR      ? 3E VID3E    ? 3F VID3F
  80 SCREENS      81 SECONDS      82 TARL       83 TARH
  84 COMCOD       85 VLNCT        86 SPREG      87 DIGCON
  88 TOPCON       89 P1HS         8A P2HS       8B S1HS
  8C S2HS         8D TREG1        93 TREG2      99 TREG3
  9F TREG4        A5 SMODE        A6 MODE1      A7 MODE2
  A8 EBITS        A9 TMCNT        AA SPTS       AB P1CR
  AC P2CR         AD BKCR         AE OBJCR      AF P1VERT
  B0 P2VERT       B1 S1VERT       B2 S2VERT     B3 P1IMG
  B8 P2IMG        C3 OBJAVT       C4 OBJAIM     CC OBJBVT
  CD OBJBIM       D5 OBJCVT       D6 OBJCIM   ?0280 CTROLS
?0281 DDRA      ?0282 SWTCHS    ?0283 DDRB     0284 RTIME
?0294 TIM1      ?0295 TIM8       0296 TIM64   ?0297 TIM1024
?F000 START      F008 CLOOP     ?F076 LETR
?F07C LETO      ?F082 LETB       F090 LETB0
 F098 LETC       F0A0 TINIT      F0A2 TLP
 F0BA BIMLP      F0C6 CIMLP      F0EC MAIN
 F114 DISPLAY    F11C TWAIT      F12E TOPWT
?F170 PART2      F192 SLOOP      F1B2 NOP1
 F1B9 P1DSP      F1BB P2TST      F1CC NOP2
 F1D3 P2DSP      F1D5 DATST      F1E6 NOA
 F1ED DADSP      F1EF DBTST      F200 NOB
 F207 DBDSP      F209 DCTST      F21A NOC
 F221 DCDSP      F223 ENDLP      F22E LPDONE
 F24E SYN1       F26C FMON       F282 NOCOM
 F28F L2         F298 L3         F29B READOK
 F2A5 WRITEOK    F2AD SETADDR   ?F2A4 DOREAD
 F2DA RL2        F2DF RXT        F2D7 RL1
 F2FC WL2        F301 WXT        F2F9 WL1
 FFF2 AR3                        FFF0 AR1
                                 FFF1 AR2
```

```
SOURCE FILE: PMOVE
0000:            1 * ************************
0000:            2 * * COPYRIGHT 1982       *
0000:            3 * * FROBCO  ALL RIGHTS RESERVED *
0000:            4 * *                      *
0000:            5 * ************************
0000:            6 *
0000:            7 * AUTHOR: KEN CLEMENTS
0000:            8 * DATE: 10/13/82
0000:            9 * LAST MODIFIED: 10/13/82
0000:           10 * VERSION 1.1
0000:           11 * PAGE MOVE
0000:           12 *
----- NEXT OBJECT FILE NAME IS PMOVE.OBJ0
0300:           13          ORG  $300
0300:A2 00      14 PMOVE    LDX  $0
0302:BD 00 10   15 PM1      LDA  $1000,X
0305:9D 00 10   16 PM2      STA  $1000,X
0308:E8         17          INX
0309:D0 F7      18          BNE  PM1
0304:           19 PFROM    EQU  PM1+2
0307:           20 PTO      EQU  PM2+2
030B:8E 03 03   21          STX  PFROM-1
030E:8E 06 03   22          STX  PTO-1
0311:60         23          RTS
0312:           24 *
0312:           25 *
0312:           26 * HERE IS A ROUTINE TO LOAD OR
0312:           27 * UNLOAD THE FROB
0312:           28 *
0312:00         29 FSLOT    DFB  0          ; SLOT NUMBER FOR THE FROB
0313:00         30 LFROM    DFB  0          ; WHERE IN MEMORY TO LOAD TO/FROM
0314:00         31 LNPGS    DFB  0          ; HOW MANY PAGES TO TRANSFER
0315:00         32 STPAGE   DFB  0          ; WHAT FROB PAGE TO START WITH
0316:           33 *
0316:A9 03      34 DNLOAD   LDA  #<PTO      ; PICKUP HIGH ORDER ADDRESS PART
0318:8D 65 03   35          STA  TX+2 ; SETUP FOR DOWNLOAD
031B:A9 07      36          LDA  #>PTO ; PICKUP LOW ORDER ADDRESS PART
031D:8D 64 03   37          STA  TX+1
0320:A9 03      38          LDA  #<PFROM
0322:8D 77 03   39          STA  TX1+2
0325:8D 89 03   40          STA  TX2+2
0328:A9 04      41          LDA  #>PFROM
032A:8D 76 03   42          STA  TX1+1
032D:8D 88 03   43          STA  TX2+1
0330:4C 4D 03   44          JMP   FXFER
0333:           45 *
0333:A9 03      46 UPLOAD   LDA  #<PFROM
0335:8D 65 03   47          STA  TX+2
0338:A9 04      48          LDA  #>PFROM
033A:8D 64 03   49          STA  TX+1
033D:A9 03      50          LDA  #<PTO
033F:8D 77 03   51          STA  TX1+2
0342:8D 89 03   52          STA  TX2+2
0345:A9 07      53          LDA  #>PTO
```

```
0347:8D 76 03   54          STA  TX1+1
034A:8D 88 03   55          STA  TX2+1
034D:           56 *
034D:AD 14 03   57 FXFER    LDA  LNPGS
0350:F0 34      58          BEQ  EXIT       ; DON'T DO FOR 0 PAGES
0352:A9 00      59          LDA  #0
0354:8D 06 03   60          STA  PTO-1
0357:8D 03 03   61          STA  PFROM-1
035A:A9 C0      62          LDA  #$C0
035C:8D 7D 03   63          STA  PPOINT+2
035F:18         64          CLC
0360:6D 12 03   65          ADC  FSLOT
0363:8D 07 03   66 TX       STA  PTO
0366:29 07      67          AND  #7         ; GET SLOT NUMBER BACK
0368:18         68          CLC             ; ADD IN DEV SELECT OFFSET
0369:69 08      69          ADC  #8
036B:0A         70          ASL  A          ; SHIFT IT OVER TO
036C:0A         71          ASL  A          ; FORM LOWER BYTE OF
036D:0A         72          ASL  A          ; DEVSEL ADDRESS FOR SLOT
036E:0A         73          ASL  A
036F:8D 7C 03   74          STA  PPOINT+1
0372:AD 13 03   75          LDA  LFROM      ; MEM PAGE ADDRESS
0375:8D 04 03   76 TX1      STA  PFROM
0378:           77 *
0378:AD 15 03   78 LDLOOP   LDA  STPAGE     ; FROB START PG
037B:8D E8 03   79 PPOINT   STA  1000       ; FROB PAGE SELECT
037E:20 00 03   80          JSR  PMOVE
0381:CE 14 03   81          DEC  LNPGS
0384:D0 01      82          BNE  TX2          ; KEEP GOING UNTIL 0
0386:60         83 EXIT     RTS
0387:EE 04 03   84 TX2      INC  PFROM
038A:EE 15 03   85          INC  STPAGE
038D:4C 78 03   86          JMP  LDLOOP

*** SUCCESSFUL ASSEMBLY: NO ERRORS
```

?0316 DNLOAD
0378 LDLOOP
0302 PM1
0307 PTO
0387 TX2
0386 EXIT
0313 LFROM
0305 PM2
0315 STPAGE
?0333 UPLOAD
0312 FSLOT
0314 LNPGS
0300 PMOVE
0363 TX
034D FXFER
0304 PFROM
037B PPOINT
0375 TX1

0304 PFROM 0305 PM2

0300 PMOVE 0302 PM1
0307 PTO 0312 FSLOT 0313 LFROM 0314 LNPGS
0315 STPAGE ?0316 DNLOAD ?0333 UPLOAD 034D FXFER
0363 TX 0375 TX1 0378 LDLOOP 037B PPOINT
0386 EXIT 0387 TX2

```
SOURCE FILE: FMON
0000:            1 * FMON -- THE VCS SIDE OF THE DEBUGGER SYSTEM
0000:            2 * COPYRIGHT 1982 BY FROBCO
0000:            3 * ALL RIGHTS RESERVED
0000:            4 *
0000:            5 * VERSION 1.1 -- LAST MODIFIED 10/28/82
0000:            6 *
0000:            7 *
0000:            8 * SYSTEM DEFINITIONS
0000:            9 *
FFF1:           10 PSTAT   EQU  $FFF1     ; STATUS PORT
FFF2:           11 RDATA   EQU  $FFF2     ; DATA PORT FROM APPLE
FFF0:           12 WDATA   EQU  $FFF0     ; DATA PORT TO APPLE
0000:           13 *
FFF4:           14 F1REST  EQU  $FFF4     ; FIRST PART OF FLAG RESTORE
FFF5:           15 F2REST  EQU  $FFF5     ; LAST PART OF FLAG RESTORE
FFF6:           16 AREST   EQU  $FFF6     ; RESTORE A FROM HERE
FFF7:           17 XREST   EQU  $FFF7     ; RESTORE X FROM HERE
FFF8:           18 YREST   EQU  $FFF8     ; RESTORE Y FORM HERE
FFF9:           19 SREST   EQU  $FFF9     ; RESTORE S FORM HERE
0000:           20 *
0000:           21 *
0000:           22 *
0000:           23 * RAM DEFINITIONS
0000:           24 *
00E0:           25 RSAVE   EQU  $E0       ; PLACE TO START SAVEING RAM LOCATIONS
00F8:           26 RBASE   EQU  $F8       ; GENERAL 16 BIT POINTER
0000:           27 *
0000:           28 *
----- NEXT OBJECT FILE NAME IS FMON.OBJ0
FF11:           29         ORG  $FF11     ; FMON ADDRESS SPACE
FF11:           30 *
FF11:           31 *
FF11:           32 * THIS IS THE ENTRY POINT UPON BREAK POINT
FF11:           33 * THE SYSTEM SAVE THE STATE BY PASSING INFORMATION
FF11:           34 * OVER TO THE APPLE.
FF11:           35 *
FF11:           36 * THE TRICK HERE IS TO SAVE THE STATE OF THE N AND Z
FF11:           37 * FLAGS WITHOUT USING THE STACK WHICH MAY OR MAY NOT
FF11:           38 * BE SET UP AS A STACK AT THE TIME OF THE BREAK POINT
FF11:           39 *
FF11:           40 *
FF11:8D F0 FF   41 BREAK   STA  WDATA     ; SEND THE A REGISTER TO THE APPLE
FF14:D0 09      42         BNE  NOTZ      ; BRANCH IF Z FLAG NOT SET
FF16:AD F1 FF   43 BRKW1   LDA  PSTAT     ; ELSE SEND A CODE THAT WILL
FF19:10 FB      44         BPL  BRKW1     ; RESTORE N=0 AND Z=1 WHEN WE GET BACK
FF1B:A9 FF      45         LDA  #$FF      ; AND DO AN INC MEM. (FF WILL INC TO 0)
FF1D:D0 12      46         BNE  BRK1      ; BRANCH ALWAYS
FF1F:           47 *
FF1F:30 09      48 NOTZ    BMI  FNEG      ; KEEP GOING IF N=1
FF21:AD F1 FF   49 BRKW2   LDA  PSTAT     ; ELSE CHOOSE CODE THAT WILL RESTORE
FF24:10 FB      50         BPL  BRKW2
FF26:A9 01      51         LDA  #$01      ; NON ZERO AND POSITIVE
FF28:D0 07      52         BNE  BRK1      ; BRANCH ALWAYS
FF2A:           53 *
```

```
FF2A:           54 *
FF2A:AD F1 FF   55 FNEG    LDA  PSTAT     ; LOOK AT THE OK TO WRITE FLAG
FF2D:10 FB      56         BPL  FNEG      ; WAIT FOR IT TO GO HIGH
FF2F:A9 80      57         LDA  #$80      ; THIS WILL INC TO NEG
FF31:8D F0 FF   58 BRK1    STA  WDATA     ; SEND FLAG CODE
FF34:AD F1 FF   59 BRKW4   LDA  PSTAT     ; WAIT TO STORE X REG
FF37:10 FB      60         BPL  BRKW4
FF39:8E F0 FF   61         STX  WDATA     ; SEND THE X REGISTER
FF3C:AD F1 FF   62 BRK2    LDA  PSTAT     ; LOOK AT THE OK TO WRITE FLAG
FF3F:10 FB      63         BPL  BRK2      ; KEEP WAITING
FF41:8C F0 FF   64         STY  WDATA     ; SEND THE Y REGISTER
FF44:AD F1 FF   65 BRK3    LDA  PSTAT     ; CHECK FLAG AGAIN
FF47:10 FB      66         BPL  BRK3      ; AND WAIT FOR IT
FF49:BA         67         TSX            ; GET THE STACK POINTER
FF4A:8E F0 FF   68         STX  WDATA     ; SEND IT OVER
FF4D:A2 E0      69         LDX  #RSAVE    ; SET UP LOOP TO SAVE RAM LOCATIONS
FF4F:AD F1 FF   70 BRK4    LDA  PSTAT     ; CHECK OK TO WRITE FLAG
FF52:10 FB      71         BPL  BRK4      ; LOOP ON IT
FF54:B5 00      72         LDA  0,X       ; GET RAM VALUE
FF56:8D F0 FF   73         STA  WDATA     ; SEND IT TO THE APPLE
FF59:E8         74         INX              ; MOVE TO NEXT
FF5A:30 F3      75         BMI  BRK4
FF5C:           76 *
FF5C:CA         77         DEX            ; NOW RESET THE STACK POINTER
FF5D:9A         78         TXS
FF5E:08         79         PHP            ; PUSH THE FLAGS
FF5F:68         80         PLA            ; GET THEM IN A
FF60:20 A2 FF   81         JSR  WBA       ; GO SEND FLAGS
FF63:           82 *
FF63:           83 *
FF63:           84 *
FF63:           85 *
FF63:           86 * THIS IS THE COMMAND LEVEL
FF63:           87 *
FF63:20 AB FF   88 CI      JSR  RBA       ; GET A POSSIBLE COMMAND BYTE
FF66:C9 10      89         CMP  #$10      ; DO WE READ MEM?
FF68:F0 5B      90         BEQ  RM        ; IF SO DO IT
FF6A:C9 20      91         CMP  #$20      ; DO WE WRITE MEM?
FF6C:F0 66      92         BEQ  WM        ; IF SO DO IT
FF6E:C9 30      93         CMP  #$30      ; DO WE GO FROM BREAK?
FF70:F0 07      94         BEQ  GO        ; IF SO DO IT
FF72:C9 40      95         CMP  #$40      ; DO WE JUMP INDIRECT?
FF74:D0 ED      96         BNE  CI        ; IF NOT KEEP LOOPING
FF76:4C E3 FF   97         JMP  GOI       ; ELSE DO THE JMP INDIRECT
FF79:           98 *
FF79:           99 * ROUTINE TO RESTORE STATE AFTER BREAK POINT AND JMP
FF79:          100 * BACK INTO THE VCS PROGRAM
FF79:          101 *
FF79:          102 *
FF79:AD F4 FF  103 GO      LDA  F1REST    ; GET PART 1 OF THE FLAG RESTORE PROCESS
FF7C:48        104         PHA            ; PUT IN FLAGS BY WAY OF STACK
FF7D:28        105         PLP
FF7E:          106 *
FF7E:A2 20     107         LDX  #256-RSAVE ; LOOP COUNT FOR RAM STUFF BACK
FF80:AD F1 FF  108 GO1     LDA  PSTAT     ; LOOK AT STATUS
FF83:29 40     109         AND  #$40      ; SEE IF STUFF THERE
FF85:F0 F9     110         BEQ  GO1       ; IF NOT THEN WAIT
```

```
FF87:AD F2 FF  111          LDA  RDATA     ; THEN GET VALUE
FF8A:95 DF     112          STA  RSAVE-1,X ; PUT BACK IN RAM
FF8C:CA        113          DEX            ; MOVE TO NEXT
FF8D:D0 F1     114          BNE  GO1       ; LOOP
FF8F:          115 *
FF8F:          116 *
FF8F:AE F9 FF  117          LDX  SREST     ; GET VALUE IN X TO RESTORE STACK POINTER
FF92:9A        118          TXS            ; RESTORE STACK POINTER
FF93:AC F8 FF  119          LDY  YREST     ; GET VALUE TO RESTORE Y
FF96:AE F7 FF  120          LDX  XREST     ; GET VALUE TO RESTORE X
FF99:AD F6 FF  121          LDA  AREST     ; GET VALUE TO RESTORE A
FF9C:EE F5 FF  122          INC  F2REST    ; RESTORE Z AND N FLAGS
FF9F:          123 * NOW ALL IS RESTORED SO JMP BACK
FF9F:4C 11 FF  124 BJUMP    JMP  BREAK
FFA2:          125 *
FFA2:          126 *
FFA2:          127 *
FFA2:          128 *
FFA2:          129 * HERE IS THE COMM PACKAGE
FFA2:          130 *
FFA2:2C F1 FF  131 WBA      BIT  PSTAT     ; GET READY TO WRITE A BYTE TO THE APPLE
FFA5:10 FB     132          BPL  WBA       ; WAIT FOR WRITE OK FLAG
FFA7:8D F0 FF  133          STA  WDATA     ; THEN DO IT AND RETURN
FFAA:60        134          RTS
FFAB:          135 *
FFAB:          136 *
FFAB:2C F1 FF  137 RBA      BIT  PSTAT     ; GET READY TO READ SOMETHING
FFAE:50 FB     138          BVC  RBA       ; WAIT FOR READ READY BIT
FFB0:AD F2 FF  139          LDA  RDATA     ; READ THE DATA
FFB3:60        140          RTS              ; THEN DONE
FFB4:          141 *
FFB4:          142 *
FFB4:          143 *
FFB4:          144 * ROUTINE TO SET UP A 16 BIT ADDRESS AND 8 BIT COUNT
FFB4:          145 *
FFB4:20 AB FF  146 GETADR   JSR  RBA       ; GO GET HIGH ADDRESS PART
FFB7:85 F9     147          STA  RBASE+1   ; PUT INTO BASE POINTER
FFB9:20 AB FF  148          JSR  RBA       ; GET LOWER PART
FFBC:85 F8     149          STA  RBASE     ; PUT IN POINTER
FFBE:20 AB FF  150          JSR  RBA       ; GET THE LENGTH
FFC1:AA        151          TAX            ; PUT IN X REG
FFC2:A0 00     152          LDY  #0        ; USE Y FOR INDEX
FFC4:60        153          RTS            ; NOW ADDRESS AND COUNT SETUP
FFC5:          154 *
FFC5:          155 *
FFC5:          156 *
FFC5:          157 * ROUTINE TO READ OUT MEMORY
FFC5:20 B4 FF  158 RM       JSR  GETADR    ; SET UP ADDRESS AND COUNT
FFC8:B1 F8     159 RM1      LDA  (RBASE),Y ; GET VALUE
FFCA:20 A2 FF  160          JSR  WBA
FFCD:C8        161          INY            ; MOVE INDEX
FFCE:CA        162          DEX            ; DECREMENT COUNT
FFCF:D0 F7     163          BNE  RM1       ; LOOP TILL DONE
FFD1:4C 63 FF  164          JMP  CI        ; DONE
FFD4:          165 *
FFD4:          166 *
FFD4:          167 *
```

```
FFD4:           168 * ROUTINE TO WRITE MEMORY
FFD4:           169 *
FFD4:20 B4 FF   170 WM      JSR  GETADR    ; GO GET ADDRESS AND COUNT
FFD7:20 AB FF   171 WM1     JSR  RBA       ; GO GET BYTE
FFDA:91 F8      172         STA  (RBASE),Y ; STORE IT
FFDC:C8         173         INY            ; MOVE INDEX
FFDD:CA         174         DEX            ; DECREMENT COUNT
FFDE:D0 F7      175         BNE  WM1       ; LOOP TILL DONE
FFE0:4C 63 FF   176         JMP  CI        ; DONE
FFE3:           177 *
FFE3:           178 *
FFE3:           179 *
FFE3:           180 * ROUTINE TO TAKE A JMP INDIRECT
FFE3:           181 *  (USED TO PUT THINGS TO SLEEP)
FFE3:20 AB FF   182 GOI     JSR  RBA       ; GET HIGH ORDER JMP ADDR
FFE6:85 F9      183         STA  RBASE+1   ; PUT IN POINTER HIGH
FFE8:20 AB FF   184         JSR  RBA       ; GET LOW ORDER JMP ADDR
FFEB:85 F8      185         STA  RBASE     ; PUT IN POINTER LOW
FFED:6C F8 00   186         JMP  (RBASE)   ; GO THERE
FFF0:           187 *
FFF0:           188 *
FFF0:           189 * TOP OF MEMORY STUFF -- THIS ADDRESS MUST BE $FF00
FFF0:           190 *
FFF0:00 00      191         DW   0
FFF2:00 00      192         DW   0
FFF4:00 00      193         DW   0
FFF6:00 00      194         DW   0         ; THIS IS FUTURE PATCH AREA
FFF8:00 00      195         DW   0
FFFA:00 00      196         DW   0
FFFC:63 FF      197         DW   CI        ; RESET JUMP VECTOR
FFFE:00 00      198         DW   0         ; END OF ADDRESS SPACE (FFFE)
0000:           199 *
0000:           200 *
0000:           201 *
0000:           202 * END OF FMON

*** SUCCESSFUL ASSEMBLY: NO ERRORS
```

```
FFF6 AREST      ?FF9F BJUMP     FF11 BREAK      FF31 BRK1
FF3C BRK2        FF44 BRK3      FF4F BRK4       FF16 BRKW1
FF21 BRKW2       FF34 BRKW4     FF63 CI         FFF4 F1REST
FFF5 F2REST      FF2A FNEG      FFB4 GETADR     FF79 GO
FF80 GO1         FFE3 GOI       FF1F NOTZ       FFF1 PSTAT
FFAB RBA           F8 RBASE     FFF2 RDATA      FFC5 RM
FFC8 RM1           E0 RSAVE     FFF9 SREST      FFA2 WBA
FFF0 WDATA      FFD4 WM         FFD7 WM1        FFF7 XREST
FFF8 YREST
```

```
                E0 RSAVE     F8 RBASE    FF11 BREAK    FF16 BRKW1
FF1F NOTZ       FF21 BRKW2   FF2A FNEG   FF31 BRK1
FF34 BRKW4      FF3C BRK2    FF44 BRK3   FF4F BRK4
FF63 CI         FF79 GO      FF80 GO1   ?FF9F BJUMP
FFA2 WBA        FFAB RBA     FFB4 GETADR  FFC5 RM
FFC8 RM1        FFD4 WM      FFD7 WM1     FFE3 GOI
FFF0 WDATA      FFF1 PSTAT   FFF2 RDATA   FFF4 F1REST
FFF5 F2REST     FFF6 AREST   FFF7 XREST   FFF8 YREST
FFF9 SREST
```

```
]LIST

1  HIMEM: 32767
2  REM  VERSION 1.1
3  HOME
5 D$ = ""
10  REM  PROGRAM TO BRING UP EXPLORER
11  REM  VERSION 1.0
12 PM = 768
15  PRINT D$;"BLOAD PMOVE.OBJ"
20  PRINT "PROGRAM TO LOAD THE FROB EXPLORER"
22  PRINT "ENTER SLOT FROB SLOT NUMBER (1-7)";
23  INPUT FSLOT
24  IF FSLOT < 1 THEN 1
25  IF FSLOT > 7 THEN 1
50 DEV = (FSLOT + 8) * 16 + 49152
70  PRINT
75  REM   LOAD THE FILE AT $8000
80  PRINT D$"BLOAD EXPLOR.OBJ,A$8000"
90  PRINT
95 STPAGE = 0
96 LNPGS = 16
97 LFROM = 128
160  GOSUB 1000
200  PRINT "DOWNLOAD PROCESS COMPLETE"
300  POKE DEV,48
310  PRINT "TURN ON VCS AND HIT RETURN";
320  INPUT A$
330  PRINT D$;"BLOAD XCONTROL.OBJ"
340  POKE 33437,(FSLOT + 8) * 16
350  POKE 33441,(FSLOT + 8) * 16 + 1
360  CALL 32768
500  END
1000  REM  FROB DOWNLOAD ROUTINE
1010  POKE PM + 18,FSLOT
1020  POKE PM + 19,LFROM
1030  POKE PM + 20,LNPGS
1040  POKE PM + 21,STPAGE
1050  CALL PM + 22
1060  RETURN

]
```

```
LOAD FLOAD
]LIST

1  HIMEM: 32767
2  HOME
3  REM  VERSION 1.1
5 D$ = ""
10  REM  TO LOAD FROM DISK TO 'THE FROB'
15  PRINT D$;"BLOAD PMOVE.OBJ"
17 PM = 768: REM  START ADDRESS OF PMOVE
20  PRINT "PROGRAM TO LOAD THE FROB"
21  PRINT : PRINT : PRINT
24  GOSUB 3000
25  PRINT "ENTER NAME OF FILE";
30  INPUT FILE$
40  IF FILE$ = "MEMORY" THEN 800
60  PRINT "READY FOR DISK LOAD (Y/N)";
70  INPUT A$
72  IF A$ = "N" THEN  END
75  REM   LOAD THE FILE AT $8000
80  PRINT D$;"BLOAD ",FILE$,",A$8000
90  PRINT ""
95 STPAGE = 0
96 LNPGS = 16
97 LFROM = 128
125  PRINT "IS IT A 4K FILE (Y/N)";
126  INPUT A$
127  IF A$ = "Y" THEN 160
128  IF A$ = "N" THEN 130
129  GOTO 125
130 LNPGS = 8
132  PRINT "LOAD TO HIGH OR LOW MEMORY (H/L)"
133  INPUT A$
134  IF A$ = "H" THEN 140
135  IF A$ = "L" THEN 150
136  GOTO 132
137  REM
140 STPAGE = 8
144  GOTO 160
150 STPAGE = 0
160  GOSUB 1000
200  PRINT "DOWNLOAD PROCESS COMPLETE"
300  POKE 4096 * 12 + 16 * (FSLOT + 8),16
500  END
800  PRINT "ENTER STARTING PAGE NUMBER (HEX)";
810  GOSUB 4000
820  IF FLAG = 0 AND V < 256 THEN 850
830  PRINT "IT MUST BE A ONE OR TWO DIGIT"
832  PRINT "HEXIDECIMAL NUMBER, TRY AGAIN"
835  GOTO 800
850 LFROM = V
860  PRINT "ENTER THE FROB STARTING PAGE (HEX)";
870  GOSUB 4000
880  IF FLAG = 0 AND V < 16 THEN 900
885  PRINT "IT MUST BE A HEXIDECIMAL DIGIT"
890  GOTO 860
900 STPAGE = V
```

```
 910  PRINT "HOW MANY PAGES (1-16 DEC)";
 920  INPUT LNPGS
 925  IF LNPGS + STPAGE > 16 THEN 935
 930  IF LNPGS > 0 AND LNPGS < 17 THEN 950
 935  PRINT "ARE YOU SURE?"
 940  GOTO 910
 950  GOTO 160
1000  REM  FROB DOWNLOAD ROUTINE
1010  POKE PM + 18,FSLOT
1020  POKE PM + 19,LFROM
1030  POKE PM + 20,LNPGS
1040  POKE PM + 21,STPAGE
1050  CALL PM + 22
1060  RETURN
3000  REM    COMPUTE ADDRESSES OF            CARD IO STUFF
3001  PRINT "WHAT SLOT IS THE FROB IN";
3005  INPUT FSLOT
3010  IF FSLOT < 1 THEN 3100
3020  IF FSLOT > 7 THEN 3100
3030  RETURN
3100  PRINT "THE FROB MUST BE IN A SLOT NUMBER 1-7"
3110  GOTO 3001
4000  REM  THIS ROUTINE GETS A HEX NUMBER
4001  REM  AND SETS FLAG IF ERROR
4005 FLAG = 0
4010 V = 0
4020  INPUT H$
4030  IF  LEN (H$) > 4 THEN 4500
4040  FOR I = 1 TO  LEN (H$)
4050 CHAR =  ASC ( MID$ (H$,I,1))
4080 CHAR = CHAR -  ASC ("0")
4090  IF CHAR > 9 THEN CHAR = CHAR - 7
4100  IF CHAR >  - 1 AND CHAR < 16 THEN 4150
4110 CHAR = 0
4120 FLAG = FLAG + 1
4150 V = V * 16 + CHAR
4200  NEXT I
4300  RETURN
4500 FLAG = FLAG + 1
4600  RETURN

]
```

```
 LOAD FSAVE
]LIST

1  HIMEM: 32767
2  HOME : REM  VERSION 1.1
5 D$ = ""
10  REM  PROGRAM TO LOAD TO DISK FROM THE FROB
15  PRINT D$;"BLOAD PMOVE.OBJ"
17 PM = 768: REM  START OF PMOVE
20  PRINT "PROGRAM TO SAVE FROB FILES"
21  PRINT "TO A BSAVE DISK FILE"
22  PRINT
23  PRINT
24  GOSUB 3000
25  PRINT "ENTER NAME OF FILE";
30  INPUT FILE$
95 STPAGE = 0
96 LNPGS = 16
97 LFROM = 128
160  GOSUB 1000
170  PRINT "PRESS RETURN WHEN READY TO WRITE DISK";
180  INPUT A$
190  PRINT D$;"BSAVE",FILE$,",A$8000,L$1000"
195  PRINT
200  PRINT "SAVE PROCESS COMPLETE"
500  END
1000  REM  FROB UPLOAD ROUTINE
1010  POKE PM + 18,FSLOT
1020  POKE PM + 19,LFROM
1030  POKE PM + 20,LNPGS
1040  POKE PM + 21,STPAGE
1050  CALL PM + 51
1060  RETURN
3000  REM    COMPUTE ADDRESSES OF             CARD TO STUFF
3001  PRINT "WHAT SLOT IS THE FROB IN";
3005  INPUT FSLOT
3010  IF FSLOT < 1 THEN 3100
3020  IF FSLOT > 7 THEN 3100
3030  RETURN
3100  PRINT "THE FROB MUST BE IN A SLOT NUMBER 1-7"
3110  GOTO 3001

]
```

frobco

603 Mission Street
Santa Cruz, CA 95060