Welcome to Apple][Redbook

30th
1978 • 2008 • ANNIVERSARY

DIGITAL
GRAPHICS

iDonate

This recreation of the 1978 Apple][Redbook is courtesy of Gerry Doire.
gerrydoire@yahoo.ca for any comments or suggestions.

If you wish to donate monetary or computer hardware
for the continuation of this project email: gerrydoire@yahoo.ca

PayPal™

Current Donators to this project are:
Dan Chisarick, David Schmidt

Apple II Reference Manual

January 1978

APPLE II
Reference Manual

Apple II Reference Manual

January 1978

APPLE II

Reference Manual

APPLE II
Reference Manual

January 1978

APPLE Computer Inc.
10260 Brandley Dr.
Cupertino, CA
95014

APPLE II Reference Manual
TABLE OF CONTENTS

GETTING STARTED WITH YOUR APPLE II

Unpacking

Don't throw away the packing material. Save it for the unlikely event that you may need to return your Apple II for warrantee repair. If you bought an Apple II Board only, see hardware section in this manual on how to get started. You should have received the following:

1. Apple II system including mother printed circuit board with specified amount of RAM memory and 8K of ROM memory, switching power supply, keyboard, and case assembly.

2. Accessories Box including the following:
   a. This manual including warranty card.
   b. Pair of Game Paddles
   c. A.C. Power Cord
   d. Cassette tape with "Breakout"on one side and "Color Demos" on the other side.
   e. Cassette recorder interface cable (miniature phone jack type)

3. If you purchased a 16K or larger system, your accessory box should also contain:
   a. 16K Startrek game cassette with High Resolution Graphics Demo ("HIRES") on the flipside.
   b. Applesoft Floating Point Basic Language Cassette with an example program on the other side.
   c. Applesoft reference manual

4. In addition other items such as a vinyl carrying case or hobby board peripherial may have been included if specifically ordered as "extras".

Notify your dealer or Apple Computer, Inc. immediately if you are missing any items.

Warranty Registration Card

Fill this card out immediately and completely and mail to Apple in order to register for one year warranty and to be placed on owners club mailing list. Your Apple II's serial number is located on the bottom near the rear edge. You model number is:

A2S00MMX

MM is the amount of memory you purchased. For Example:

A2S0008X

is an 8K Byte Apple II system.

Check for Damage

Inspect the outside case of your Apple for shipping damage. Gently lift up on the top rear of the lid of the case to release the lid snaps and remove the lid. Inspect the inside. Nothing should be loose and rattling around. Gently press down on each integrated circuit to make sure that each is still firmly seated in its socket. Plug in your game paddles into the Apple II board at the socket marked "GAME I/O" at location J14. See hardware section of this manual for additional detail. The white dot on the connector should be face forward. Be careful as this connector is fragile. Replace the lid and press on the back top of it to re-snap it into place.

Power Up

First, make sure that the power ON/OFF switch on the rear power supply panel on your Apple II is in the "OFF" position. Connect the A.C. power cord to the Apple and to a 3 wire 12Ø volt A.C. outlet. Make sure that you connect the third wire to ground if you have only a two conductor house wiring system. This ground is for your safety if there is an internal failure in the Apple power supply, minimizes the chance of static damage to the Apple, and minimizes RFI problems.

Connect a cable from the video output jack on the back of the Apple to a TV set with a direct video input jack. This type of set is commonly called a "Monitor". If your set does not have a direct video input, it is possible to modify your existing set. Write for Apple's Application note on this. Optionally you may connect the Apple to the antenna terminals of your TV if you use a modulator. See additional details in the hardware section of this manual under "Interfacing with the Home TV".

Now turn on the power switch on the back of the Apple. The indicator light (it's not a switch) on the keyboard should now be ON. If not, check A.C. connections. Press and release the "Reset" button on the keyboard. The following should happen: the Apple's internal speaker should beep, an asterisk ("*") prompt character should appear at the lower left hand corner of your TV, and a flashing white square should appear just to the right of the asterisk. The rest of the TV screen will be made up of radom text characters (typically question marks).

If the Apple beeps and garbage appears but you cannot see an "*" and the cursor, the horizontal or vertical height settings on the TV need to be adjusted. Now depress and release the "ESC" key, then hold down the "SHIFT" key while depressing and releasing the P key. This should clear your TV screen to all black. Now depress and release the "RESET" key again. The "*" prompt character and the cursor should return to the lower left of your TV screen.

Apple Speaks Several Languages

The prompt character indicates which language your Apple is currently in. The current prompt character, an asterisk ("*"),indicates that you are in the "Monitor" language, a powerful machine level language for advanced programmers. Details of this language are in the "Firmware" section of this manual.

Apple Integer BASIC

Apple also contains a high level English oriented language called Integer BASIC, permanently in its ROM memory. To switch to this language hold down the "CTRL" key while depressing and releasing the "B" key. This is called a control-B function and is similiar to the use of the shift key in that it indicates a different function to the Apple. Control key functions are not displayed on your TV screen but the Apple still gets the message. Now depress and release the "RETURN" key to tell Apple that you have finished typing a line on the keyboard. A right facing arrow (">") called a caret will now appear as the prompt character to indicate that Apple is now in its Interger BASIC language mode.

Running Your First and Second Program

Read through the next three sections that include:

1. Loading a BASIC program Tape
2. Breakout Game Tape
3. Color Demo Tape

Then load and run each program tape. Additional information on Apple II's interger BASIC is in the next section of this manual.

Running 16K Startrek

If you have 16K Bytes or larger memory in your Apple, you will also receive a "STARTREK" game tape. Load this program just as you did the previous two, but <u>before</u> you "RUN" it, type in "HIMEM: 16384" to set exactly where in memory this program is to run.

LOADING A PROGRAM TAPE

INTRODUCTION

This section describes a procedure for loading BASIC programs successfully into the Apple II. The process of loading a program is divided into three section; System Checkout, Loading a Tape and What to do when you have Loading Problems. They are discussed below.

When loading a tape, the Apple II needs a signal of about 2 1/2 to 5 volts peak-to-peak. Commonly, this signal is obtained from the "Monitor" or "earphone" output jack on the tape recorder. Inside most tape recorders, this signal is derived from the tape recorder's speaker. One can take advantage of this fact when setting the volume levels. Using an Apple Computer pre-recorded tape, and with all cables disconnected, play the tape and adjust the volume to a loud but un-distorted level. You will find that this volume setting will be quite close to the optimum setting.

Some tape recorders (mostly those intended for use with hi-fi sets) do not have an "earphone" or high-level "monitor" output. These machines have outputs labeled"line output" for connection to the power amplifier. The signal levels at these outputs are too low for the Apple II in most cases.

Cassette tape recorders in the $4Ø - $5Ø range generally have ALC (Automatic Level Control) for recording from the microphone input. This feature is useful since the user doesn't have to set any volume controls to obtain a good recording. If you are using a recorder which must be adjusted, it will have a level meter or a little light to warn of excessive recording levels. Set the recording level to just below the level meter's maximum, or to just a dim indication on the level lamp. Listen to the recorded tape after you've saved a program to ensure that the recording is "loud and clear".

Apple Computer has found that an occasional tape recorder will not function properly when both Input and Output cables are plugged in at the same time. This problem has been traced to a ground loop in the tape recorder itself which prevents making a good recording when saving a program. The easiest solution is to unplug the "monitor" output when recording. This ground loop does not influence the system when loading a pre-recorded tape.

Tape recorder head alignment is the most common source of tape recorder problems. If the playback head is skewed, then high frequency information on pre-recorded tapes is lost and all sorts of errors will result. To confirm that head alignment is the problem, write a short program in BASIC. >10 END is sufficient. Then save this program. And then rewind and load the program. If you can accomplish this easily but cannot load pre-recorded tapes, then head alignment problems are indicated.

Apple Computer pre-recorded tapes are made on the highest quality professional duplicating machines, and these tapes may be used by the service technician to align the tape recorder's heads. The frequency response of the tape recorder should be fairly good; the 6 KHz tone should be not more than 3 db down from a 1 KHz tone, and a 9 KHz tone should be no more than 9 db down. Note that recordings you have made yourself with mis-aligned heads may not not play properly with the heads properly aligned. If you made a recording with a skewed record head, then the tiny magnetic fields on the tape will be skewed as well, thus playing back properly only when the skew on the tape exactly matches the skew of the tape recorder's heads. If you have saved valuable programs with a skewed tape recorder, then borrow another tape recorder, load the programs with the old tape recorder into the Apple, then save them on the borrowed machine. Then have your tape recorder properly aligned.

Listening to the tape can help solve other problems as well. Flaws in the tape, excessive speed variations, and distortion can be detected this way. Saving a program several times in a row is good insurance against tape flaws. One thing to listen for is a good clean tone lasting for at least 3 1/2 seconds is needed by the computer to "set up" for proper loading. The Apple puts out this tone for anout 1Ø seconds when saving a program, so you normally have 6 1/2 seconds of leeway. If the playback volume is too high, you may pick up tape noise before getting to the set-up tone. Try a lower playback volume.

SYSTEM CHECKOUT

A quick check of the Apple II computer system will help you spot any problems that might be due to improperly placed or missing connections between the Apple II, the cassette interface, the Video display, and the game paddles. This checkout procedure takes just a few seconds to perform and is a good way of insuring that everything is properly connected before the power is turned on.

1. POWER TO APPLE - check that the AC power cord is plugged into an appropriate wall socket, which includes a "true" ground and is connected to the Apple II.

2. CASSETTE INTERFACE - check that at least one cassette cable double ended with miniature phone tip jacks is connected between the Apple II cassette Input port and the tape recorder's MONITOR plug socket.

3. VIDEO DISPLAY INTERFACE -
   a) for a video monitor - check that a cable connects the monitor to the Apple's video output port.
   b) for a standard television - check that an adapter (RF modulator) is plugged into the Apple II (either in the video output (K 14) or the video auxiliary socket (J148), and that a cable runs between the television and the Adapter's output socket.

4. GAME PADDLE INTERFACE - if paddles are to be used, check that they are connected into the Game I/O connector (J14) on the right-hand side of the Apple II mainboard.

5. POWER ON - flip on the power switch in back of the Apple II, the "power" indicator on the keyboard will light. Also make sure the video monitor (or TV set) is turned on.

After the Apple II system has been powered up and the video display presents a random matrix of question marks or other text characters the following procedure can be followed to load a BASIC program tape:

1. Hit the RESET key.
   An asterick, "*",should appear on the lefthand side of the screen below the random text pattern. A flashing white cursor will appear to the right of the asterick.

2. Hold down the CTRL key, depress and release the B key, then depress the "RETURN" key and release the "CTRL" key. A right facing arrow should appear on the lefthand side of the screen with a flashing cursor next to it. If it doesn't, repeat steps 1 and 2.

3. Type in the word "LOAD" on the keyboard. You should see the word in between the right facing arrow and the flashing cursor. <u>Do not depress the "RETURN" key yet</u>.

4. Insert the program cassette into the tape recorder and rewind it.

5. If not already set, adjust the Volume control to 50-70% maximum. If present, adjust the Tone control to 80-100% maximum.

6. Start the tape recorder in "PLAY" mode and now depress the "RETURN" key on the Apple II.
7. The cursor will disappear and Apple II will beep in a few seconds when it finds the beginning of the program. If an error message is flashed on the screen, proceed through the steps listed in the Tape Problem section of this paper.
8. A second beep will sound and the flashing cursor will reappear after the program has been successfully loaded into the computer.
9. Stop the tape recorder. You may want to rewind the program tape at this time.
10. Type in the word "RUN" and depress the "RETURN" key.

The steps in loading a program have been completed and if everying has gone satisfactorily the program will be operating now.

LOADING PROBLEMS

Occasionally, while attempting to load a BASIC program Apple II beeps and a memory full error is written on the screen. At this time you might wonder what is wrong with the computer, with the program tape, or with the cassette recorder. Stop. This is the time when you need to take a moment and checkout the system rather than haphazardly attempting to resolve the loading problem. Thoughtful action taken here will speed in a program's entry. If you were able to successfully turn on the computer, reset it, and place it into BASIC then the Apple II is probably operating correctly. Before describing a procedure for resolving this loading problem, a discussion of what a memory full error is in order.

The memory full error displayed upon loading a program indicates that not enough (RAM) memory workspace is available to contain the incoming data. How does the computer know this? Information contained in the beginning of the program tape declares the record length of the program. The computer reads this data first and checks it with the amount of free memory. If adequate workspace is available program loading continues. If not, the computer beeps to indicate a problem, displays a memory full error statement, stops the loading procedure, and returns command of the system to the keyboard. Several reasons emerge as the cause of this problem.

Memory Size too Small

Attempting to load a 16K program into a 4K Apple II will generate this kind of error message. It is called loading too large of a program. The solution is straight forward: only load appropriately sized programs into suitably sized systems.

Another possible reason for an error message is that the memory pointers which indicate the bounds of available memory have been preset to a smaller capacity. This could have happened through previous usage of the "HIMEN:" and "LOMEN:" statements. The solution is to reset the pointers by BC (CTRL B) command. Hold the CTRL key down, depress and release the B key, then depress the RETURN key and release the CTRL key. This will reset the system to maximum capacity.

Cassette Recorder Inadjustment

If the Volume and Tone controls on the cassette recorder are not properly set a memory full error can occur. The solution is to adjust the Volume to 50-70% maximum and the Tone (if it exists) to 80-100% maximum.*

A second common recorder problem is skewed head azimuth. When the tape head is not exactly perpendicular to the edges of the magnetic tape some of the high frequency data on tape can be skipped. This causes missing bits in the data sent to the computer. Since the first data read is record length an error here could cause a memory full error to be generated because the length of the record is inaccurate. The solution: adjust tape head azimuth. It is recommended that a competent technician at a local stereo shop perform this operation.
Often times new cassette recorders will not need this adjustment.

*Apple Computer Inc. has tested many types of cassette recorders and so far the Panasonic RQ-309 DS (less than $40.00) has an excellent track record for program loading.

Tape Problems

A memory full error can result from unintentional noise existing in a program tape. This can be the result of a program tape starting on its header which sometimes causes a glitch going from a nonmagnetic to magnetic recording surface and is interpreted by the computer as the record length. Or, the program tape can be defective due to false erasure, imperfections in the tape, or physical damage. The solution is to take a moment and listen to the tape. If any imperfections are heard then replacement of the tape is called for. Listening to the tape assures that you know what a "good" program tape sounds like. If you have any questions about this please contact your local dealer or Apple for assistance.

If noise or a glitch is heard at the beginning of a tape advance the tape to the start of the program and re-Load the tape.

Dealing with the Loading Problem

With the understanding of what a memory full error is an efficient way of dealing with program tape loading problems is to perform the following procedure:

1. Check the program tape for its memory requirements. Be sure that you have a large enough system.

2. Before loading a program reset the memory pointers with the B_C (control B) command.

3. In special cases have the tape head azimuth checked and adjusted.

4. Check the program tape by listening to it.
   a) Replace it if it is defective, or
   b) start it at the beginning of the program.

5. Then re-LOAD the program tape into the Apple II.

In most cases if the preceeding is followed a good tape load will result.

UNSOLVED PROBLEMS

If you are having any unsolved loading problems, contact your nearest local dealer or Apple Computer Inc.

BREAKOUT GAME TAPE

PROGRAM DESCRIPTION

Breakout is a color graphics game for the Apple II computer. The object of the game is to "knock-out' all 16Ø colored bricks from the playing field by hitting them with the bouncing ball. You direct the ball by hitting it with a paddle on the left side of the screen. You control the paddle with one of the Apple's Game Paddle controllers. But watch out: you can only miss the ball five times:

There are eight columns of bricks. As you penetrate through the wall the point value of the bricks increases. A perfect game is 72Ø points; after five balls have been played the computer will display your score and a rating such as "Very Good". "Terrible!", etc. After ten hits of the ball, its speed with double, making the game more difficult. If you break through to the back wall, the ball will rebound back and forth, racking up points.

Breakout is a challenging game that tests your concentration, dexterity, and skill.

REQUIREMENTS

This program will fit into a 4K or greater system.
BASIC is the programming language used.

PLAYING BREAKOUT

1. Load Breakout game following instructions in the "Loading a BASIC Program from Tape" section of this manual.
2. Enter your name and depress RETURN key.
3. If you want standard BREAKOUT colors type in Y or Yes and hit RETURN. The game will then begin.
4. If the answer to the previous questions was N or No then the available colors will be displayed. The player will be asked to choose colors, represented by a number from Ø to 15, for background, even bricks, odd bricks, paddle and ball colors. After these have been chosen the game will begin.

5. At the end of the game you will be asked if they want to play again. A Y or Yes response will start another game. A N or No will exit from the program.

NOTE: A game paddle (15Øk ohm potentiometer) must be connected to PDL (Ø) of the Game I/O connector for this game.

COLOR DEMO TAPE

PROGRAM DESCRIPTION

COLOR DEMO demonstrates some of the Apple II video graphics capabilities. In it are ten examples: Lines, Cross, Weaving, Tunnel, Circle, Spiral, Tones, Spring, Hyperbola, and Color Bars. These examples produce various combinations of visual patterns in fifteen colors on a monitor or television screen. For example, Spiral combines colorgraphics with tones to produce some amusing patterns. Tones illustrates various sounds that you can produce with the two inch Apple speaker. These examples also demonstrate how the paddle inputs (PDL(X)) can be used to control the audio and visual displays. Ideas from this program can be incorporated into other programs with a little modification.

REQUIREMENTS

4K or greater Apple II system, color monitor or television, and paddles are needed to use this program. BASIC is the programming language used.

BREAKOUT GAME
PROGRAM LISTING

PROGRAM LISTING

```
 5 GOTO 15
10 Q=( PDL (0)-20)/6: IF Q<0 THEN
   Q=0: IF Q>=34 THEN Q=34: COLOR=
   D: VLIN Q,Q+5 AT 0: COLOR=A:
   IF P>Q THEN 175: IF Q THEN
   VLIN 0,Q-1 AT 0:P=Q:RETURN

15 DIM A$(15),B$(10):A=1:B=13:
   C=9:D=6:E=15: TEXT : CALL -
   936: VTAB 4: TAB 10: PRINT
   "*** BREAKOUT ***":PRINT
20 PRINT "  OBJECT IS TO DESTROY
   ALL BRICKS": PRINT : INPUT
   "HI, WHAT'S YOUR NAME? ",A$

25 PRINT "STANDARD COLORS ";A$
   ;: INPUT "Y/N? ",B$: GR: CALL
   -936: IF B$(1,1)#"N" THEN 40
   : FOR I=0 TO 39: COLOR=I/2*
   (I(32): VLIN 0,39 AT I
30 NEXT I: POKE 34,20: PRINT :
   PRINT : PRINT : FOR I=0 TO
   15: VTAB 21+I MOD 2: TAB I+
   I+1: PRINT I;: NEXT I: POKE
   34,22: YTAB 24: PRINT : PRINT
   "BACKGROUND";
35 GOSUB 95:A=E: PRINT "EVEN BRICK"
   ;:GOSUB 95:B=E: PRINT "ODD BRIC
   K";: GOSUB 95:C=E: PRINT "PADDLE
   ";: GOSUB 95:D=E: PRINT "BALL"
   ;:GOSUB 95
40 POKE 34,20: COLOR=A: FOR I=
   0 TO 39: VLIN 0,39 AT I: NEXT
   I: FOR I=20 TO 34 STEP 2: TAB
   I+1: PRINT I/2-9;: COLOR=B:
   VLIN 0,39 AT I: COLOR=C: FOR
   J=I MOD 4 TO 39 STEP 4
45 VLIN J,J+1 AT I: NEXT J, I: TAB
   5: PRINT "SCORE=0":PRINT
   : PRINT : POKE 34,21:S=0:P=
   S:L=S:X=19:Y=19:L=6
50 COLOR=A: PLOT X,Y/3:X=19:Y=
   RND (120):V=-1:W= RND (5)-
   2:L=L-1: IF L<1 THEN 120: TAB
   6: IF L>1 THEN PRINT L;"BALLS L
   EFT"
55 IF L=1 THEN PRINT "LAST BALL, "
   ;A$: PRINT : FOR I=1 TO 100
   : GOSUB 10: NEXT I:M=1:N=0
60 J=Y+W: IF J>=0 AND J<120 THEN
   65:W=-W:J=Y: FOR I-1 TO 6:K=
    PEEK (-16336): NEXT I
65 I=X+V: IF I<0 THEN 180: GOSUB
   170: COLOR=A:K=J/3: IF I>39
   THEN 75: IF SCRN(I,K)=A THEN
   85: IF I THEN 100:N=N+1:V=(
   N>5)+1:W=(K-P)*2-5:M=1
70 Z= PEEK (-16336)-PEEK (-16336
   )+ PEEK (-16336)- PEEK (-16336
   )+ PEEK (-16336)- PEEK (-16336
   )+ PEEK (-16336): GOTO 85
75 FOR I=1 TO 6:M= PEEK (-16336
   ): NEXT I:I=X:M=0
80 V=-V
85 PLOT X,Y/3: COLOR=E: PLOT I,
   K:X=I:Y=J: GOTO 60
90 PRINT "INVALID, REENTER";
95 INPUT " COLOR (0 TO 15)",E:
   IF E<0 OR E>15 THEN 90: RETURN
100 IF M THEN V= ABS (V): VLIN
    K/2*2,K/2*2+1 AT I:S=S+I/2-
    9: VTAB 21: TAB 13: PRINT S
105 Q= PEEK (-16336)- PEEK (-16336
    )+ PEEK (-16336)- PEEK (-16336
    )+ PEEK (-16336)- PEEK (-16336
    )+ PEEK (-16336)- PEEK (-16336
    )+ PEEK (-16336)- PEEK (-16336
    )
110 IF S<720 THEN 80
115 PRINT "CONGRATULATONS, ";A$
    ;" YOU WIN!": GOTO 165
120 PRINT "YOUR SCORE OF ";S;" IS "
    ;: GOTO 125+(S/100)*5
125 PRINT "TERRIBLE!": GOTO 165

130 PRINT "LOUSY.": GOTO 165
135 PRINT "POOR.": GOTO 165
140 PRINT "FAIR.": GOTO 165
145 PRINT "GOOD.": GOTO 165
150 PRINT "VERY GOOD.": GOTO 165

155 PRINT "EXCELLENT.": GOTO 165

160 PRINT "NEARLY PERFECT."
165 PRINT "ANOTHER GAME ";A$;" (Y/N)
    ";: INPUT A$: IF A$(1,1)="Y"
    THEN 25: TEXT : CALL -936:
    VTAB 10: TAB 10: PRINT "GAME OV
    ER": END
170 Q=( PDL (0)-20)/6: IF Q<0 THEN
    Q=0: IF Q>=34 THEN Q=34: COLOR=
    D: VLIN Q,Q+5 AT 0: COLOR=A:
    IF P>Q THEN 175: IF Q THEN
    VLIN 0,Q-1 AT 0:P=Q: RETURN

175 IF P=Q THEN RETURN : IF Q#34
    THEN VLIN Q+6,39 AT 0:P=Q:
    RETURN
180 FOR I=1 TO 80:Q= PEEK (-16336
    ): NEXT I: GOTO 50
```

COLOR DEMO PROGRAM
LISTING

PROGRAM LISTING

```
10 DIM C(4): POKE 2,173: POKE
   3,48: POKE 4,192: POKE 5,165
   : POKE 6,0: POKE 7,32: POKE
   8,168: POKE 9,252: POKE 10,
   165: POKE 11,1: POKE 12,208

20 POKE 13,4: POKE 14,198: POKE
   15,24: POKE 16,240: POKE 17
   ,5: POKE 18,198: POKE 19,1:
    POKE 20,76: POKE 21,2: POKE
   22,0: POKE 23,96
30 TEXT : CALL -936: VTAB 4: TAB
   8: PRINT "4K COLOR DEMOS": PRINT
   : PRINT "1 LINES": PRINT "2 CROS
   S": PRINT "3 WEAVING"
40 PRINT "4 TUNNEL": PRINT "5 CIRCL
   E": PRINT "6 SPIRAL **": PRINT
   "7 TONES ** ": PRINT "8 SPRING"

50 PRINT "9 HYPERBOLA": PRINT
   "10 COLOR BARS": PRINT : PRINT
   "** NEEDS PDL(0) CONNECTED"
   : PRINT
60 PRINT "HIT ANY KEY FOR NEW DEMO"
   :Z=0: PRINT : INPUT "WHICH DEMO
   # ",I: GR : IF I>0 AND I<11
    THEN GOTO 100*I: GOTO 30
70 INPUT "WHICH DEMO WOULD YOU LIKE
   ",I: GR : IF I AND I<20 THEN
   GOTO 100*I: GOTO 30
100 I=1+I MOD 79:J=I+(I>39)*(79
   -I-I): GOSUB 2000: GOSUB 10000
   : GOTO 100
200 I=1+I MOD 39:J=I: GOSUB 2000
   :J=39-I: GOSUB 2000: GOSUB
   10000: GOTO 200
300 J=J+1:J=J MOD 22+1: FOR I=1
    TO 1295: COLOR=I MOD J+7: PLOT
    (2*I) MOD 37,(3*I) MOD 35: NEXT
    I: GOSUB 10000: GOTO 300
400 FOR I=1 TO 4:C(I)= RND (16)
    : NEXT I
410 FOR I=3 TO 1 STEP -1:C(I+1)
    =C(I): NEXT I:C(1)= RND (16
    ): FOR I=1 TO 5: FOR J=1 TO
    4
420 COLOR=C(J):L=J*5+14+I:K=39-
    L: HLIN K,L AT K: VLIN K,L AT
    L: HLIN K,L AT L: VLIN K,L AT
    K: NEXT J,I: GOSUB 10000: GOTO
    410
500 Z=20: GOTO 900
600 COLOR= RND (16): FOR I=0 TO
    18 STEP 2:J=39-I: HLIN I,J AT
    I: GOSUB 640: VLIN I,J AT J:
     GOSUB 640
610 HLIN I+2,J AT J: GOSUB 640:
     VLIN I+2,J AT I+2: GOSUB 640
    : NEXT I
620 COLOR= RND (16): FOR I=18 TO
    0 STEP -2:J=39-I: VLIN I+2,
    J AT I+2: GOSUB 640: HLIN I+
    2,J AT J: GOSUB 640
630 VLIN I,J AT J: GOSUB 640: HLIN
    I,J AT I: GOSUB 640: NEXT I:
     GOSUB 10000: GOTO 600
640 K=I+7:L=K*K*5+K*26+70:L=32767
    /L*( PDL (0)/10): POKE 0,K:
     POKE 1,L MOD 256: POKE 24,
    L/256+1: CALL 2: RETURN
700 I= RND (30)+3:J=I*I*5+I*26+
    70:K=32767/J*( PDL (0)/10):
     POKE 0,I: POKE 1,K MOD 256
    : POKE 24,(K>255)+1: CALL 2
    : GOSUB 10000: GOTO 700
800 X=3:A=1000:P=A:L=20:W=4:Y=0
    :J=1: COLOR=6: HLIN 0,39 AT
    4: COLOR=9: GOSUB 880: COLOR=
    12: VLIN 5,M-2 AT X
810 N=2*A-P-A/W: COLOR=0: GOSUB
    880: VLIN 5,39 AT X:X=X+1: IF
    X<39 THEN 820:X=3: VLIN 5,39
     AT 1: VLIN 5,39 AT 2
820 P=A:A=N:Y=A/100: COLOR=12: GOSUB
    880: COLOR=9: VLIN 5,M-2 AT
    X: COLOR=15: PLOT X-2,M: FOR
    I=0 TO J: NEXT I: GOSUB 10000
    : GOTO 810
880 M=L-Y:L1=M-1:L2=M+1: VLIN L1,
    L2 AT X-1: VLIN L1,L2 AT X:
     VLIN L1,L2 AT X+1: RETURN

900 I=1+I MOD 15: FOR Y=0 TO 39
    : FOR X=0 TO 39: COLOR=I+( ABS
    (20-X)-Z)*( ABS (20-Y)-Z)/25
    : PLOT X,Y: NEXT X,Y: GOSUB
    10000: GOTO 900
1000 CALL -936
1010 J=1+J MOD 32: COLOR=J/2: VLIN
    0,39 AT 3+J: VTAB 21+(J/2) MOD
    2: TAB 3+J: IF J MOD 2 THEN
    PRINT J/2;: GOSUB 10000: GOTO
    1010
2000 COLOR= RND (16): HLIN 0,39 AT
    J: COLOR= RND (16): VLIN 0,
    39 AT J: RETURN
10000 IF PEEK (-16384)<128 THEN RETURN
    : POKE -16368,0: POP : GOTO
    30
```

-.-.-.-.-.-.-.-.-.-   APPLE II STARTREK VERSION   -.-.-.-.-.-.-.-.-.-.-

THIS IS A SHORT DESCRIPTION OF HOW TO PLAY STARTREK ON THE APPLE COMPUTER.

THE UNIVERSE IS MADE UP OF 64 QUADRANTS IN AN 8 BY 8 MATRIX. THE QUADRANT IN WHICH YOU THE ENTERPRISE ' ARE, IS IN WHITE, AND A BLOW UP OF THAT QUADRANT IS FOUND IN THE LOWER LEFT CORNER. YOUR SPACE SHIP STATUS IS FOUND IN A TABLE TO THE RIGHT SIDE OF THE QUADRANT BLOW UP.

THIS IS A SEARCH AND DESTROY MISSION. THE OBJECT IS TO LONG-RANGE SENSE FOR INFORMATION AS TO WHERE KLINGONS (K) ARE MOVE TO THAT QUADRANT, AND DESTROY.

NUMBERS DISPLAYED FOR EACH QUADRANT DENOTE:

* OF STARS IN THE ONES PLACE
* OF BASES IN THE TENS PLACE
* OF KLINGONS IN THE HUNDREDS PLACE

AT ANY TIME DURING THE GAME, FOR INSTANCE BEFORE ONE TOTALLY RUNS OUT OF ENERGY, OR NEEDS TO REGENERATE ALL SYSTEMS, ONE MOVES TO A QUADRANT WHICH INCLUDES A BASE, IONS NEXT TO THAT BASE (B) AT WHICH TIME THE BASE SELF-DESTRUCTS AND THE ENTERPRISE (E) HAS ALL SYSTEMS 'GO' AGAIN.

TO PLAY:

1.THE COMMANDS CAN BE OBTAINED BY TYPING A '0' (ZERO) AND RETURN. THEY ARE:

1. PROPULSION	2.REGENERATE
3. LONG RANGE SENSORS	4. PHASERS
5. PHOTON TORPEDOES	6. GALAXY RECORD
7. COMPUTER	8. PROBE
9. SHIELD ENERGY	10.DAMAGE REPORT
11.LOAD PHOTON TORPEDOES	

2.THE COMANDS ARE INVOKED BY TYPING 1HE NUMBER REFERING TO THEM FOLLOWED BY A 'RETURN'.

A.IF RESPONSE IS 1 THE COMPUTER WILL ASK WARP OR ION AND EXPECTS 'W' IF ONE WANTS TO TRAVEL IN THE GALAXY BETWEEN QUADRANTS AND AN 'I' IF ONE WANTS ONLY INTERNAL QUADRANT TRAVEL.
DURATION OF WARP FACTOR IS THE NUMBER OF SPACES OR QUADRANTS THE ENTERPRISE WILL MOVE.
COURSE IS COMPASS READING IN DEGREES FOR THE DESI-RED DESTINATION.

B.A 2 REGENERATES THE ENERGY AT 1HE EXPENSE OF TIME.

C.A 3 GIVES THE CONTENTS OF THE IMMEDIATE. ADJACENT QUADRANTS. THE GALAXY IS WRAP-AROUND IN ALL DIRECTIONS.

D.4 FIRES PHASERS AT THE EXPENSE OF AVAILABLE ENERGY.

E.5 INITIATES A SET OF QUESTIONS FOR TORPEDO FIRING. THEY CAN BE FIRED AUTOMATICALLY IF THEY HAVE BEEN LOCKED ON TARGET WHILE IN THE COMPUTER MODE, OR MAY BE FIRED MANUALLY IF THE TRAGECTORY ANGLE ISKNOWN.

F.6, 8 AND 10 ALL GIVE INFORMATION ABOUT THE STATUS OF THE SHIP AND ITS ENVIRONMENT.

G.9 SETS THE SHIELD ENERGY/AVAILABLE ENERGY RATIO.

H.11 ASKS FOR INFORMATION ON LOADING AND UNLOADING OF PHOTON TORPEDOES AT THE ESPENSE OF AVAILABLE ENERGY.
THE ANSWER SHOULD BE A SIGNED NUMBER. FOR EXAMPLE +5 OR -2.

I. 7 ENTERS A COMPUTER WHICH WILL RESPOND TO THE FOLLOWING INSTRUCTIONS:

1. COMPUTE COURSE 2.LOCK PHASERS
3.LOCK PHOTON TORPEDOES
4.LOCK COURSE   5. COMPUTE TREJECTORY
6.STATUS   7. RETURN TO COMAND MODE

IN THE FIRST FIVE ONE WILL HAVE TO GIVE COORDINATES. COORDINATES ARE GIVEN IN MATHMATICAL NOTATION WITH THE EXCEPTION THAT THE 'Y' VALUE IS GIVEN FIRST. AN EXAMPLE WOULD BE 'Y,X'

COURSE OR TRAJECTORY:

```
                  0
                  |
                  |
                  |
                  |
270---------------+---------------90
                  |
                  |
                  |
                  |
                 180
```

-.-.-.-.-.-.-.- THIS EXPLANATION WAS WRITTEN BY ELWOOD -.-.-.-.-.-.-.-
NOT RESPONSIBLE FOR
ERRORS

LOADING THE HI-RES DEMO TAPE

PROCEDURE

1. Power up system - turn the AC power switch in the back of the Apple II on. You should see a random matrix of question marks and other text characters. If you don't, consult the operator's manual for system checkout procedures.

2. Hit the RESET key. On the left hand side of the screen you should see an asterisk and a flashing cursor next to it below the text matrix.

3. Insert the HI-RES demo tape into the cassette and rewind it. Check Volume (5Ø-7Ø%) and Tone (8Ø-1ØØ%) settings.

4. Type in "CØØ.FFFR" on the Apple II keyboard. This is the address range of the high resolution machine language subprogram. It extends from $CØØ to $FFF. The R tells the computer to read in the data. <u>Do not</u> depress the "RETURN" key yet.

5. Start the tape recorder in playback mode and depress the "RETURN" key. The flashing cursor disappears.

6. A beep will sound after the program has been read in. STOP the tape recorder. <u>Do not</u> rewind the program tape yet.

7. Hold down the "CTRL" key, depress and release the B key, then depress the "RETURN" key and release the "CTRL" key. You should see a right facing arrow and a flashing cursor. The Bc command places the Apple into BASIC initializing the memory pointers.

8. Type in "LOAD", restart the tape recorder in playback mode and hit the "RETURN" key. The flashing cursor disappears. This begins the loading of the BASIC subprogram of the HI-RES demo tape.

9. A beep will sound to indicate the program is being loaded.

10. A second beep will sound, and the right facing arrow will reappear with the flashing cursor. STOP the tape recorder. Rewind the tape.

11. Type in "HIMEM:8192" and hit the "RETURN" key. This sets up memory for high resolution graphics.

12. Type in "RUN" and hit the "RETURN" key. The screen should clear and momentarily a HI-RES demo menu table should appear. The loading sequence is now completed.

SUMMARY OF HI-RES DEMO TAPE LOADING

1. RESET

2. Type in CØØ.FFFR

3. Start tape recorder, hit RETURN

4. Asterick or flashing cursor reappear
   Bc (CTRL B) into BASIC

5. Type in "LOAD", hit RETURN

6. BASIC prompt (7) and flashing cursor reappear. Type in "HIMEN:8192", hit RETURN

7. Type in "RUN", hit RETURN

8. STOP tape recorder, rewind tape.

APPLE II INTEGER BASIC

BASIC COMMANDS

Commands are executed immediately; they do not require line numbers.Most Statements (see Basic Statements Section) may also be used as commands. Remember to press Return key after each command so that Apple knows that you have finished that line. Multiple commands (as opposed to statements) on same line separated by a " : " are NOT allowed.

COMMAND NAME

AUTO *num*	Sets automatic line numbering mode. Starts at line number *num* and increments line numbers by 10. To exit AUTO mode, type a control X*, then type the letters "MAN" and press the return key.
AUTO *num1*, *num2*	Same as above execpt increments line numbers by number *num2*.
CLR	Clears current BASIC variables; undimensions arrays. Program is unchanged.
CON	Continues program execution after a stop from a control C*. Does not change variables.
DEL *num1*,	Deletes line number *num1*.
DEL *num1*, *num2*	Deletes program from line number *num1* through line number *num2*.
DSP *var*	Sets debug mode that will display variable *var* every time that it is changed along with the line number that caused the change. (NOTE: RUN command clears DSP mode so that DSP command is effective only if program is continued by a CON or GOTO command.)
HIMEM *expr*	Sets highest memory location for use by BASIC at location specified by expression *expr* in decimal. HIMEM: may not be increased without destroying program. HIMEM: is automatically set at maximum RAM memory when BASIC is entered by a control B*.
GOTO *expr*	Causes immediate jump to line number specified by expression *expr*.
GR	Sets mixed color graphics display mode. Clears screen to black. Resets scrolling window. Displays 4Ø x4Ø squares in 15 colors on top of screen and 4 lines of text at bottom.
LIST	Lists entire program on screen.
LIST *num1*	Lists program line number *num1*.
LIST *num1*, *num2*	Lists program line number *num1* through line number *num2*.

LOAD *expr.*	Reads (Loads) a BASIC program from cassette tape. Start tape recorder before hitting return key. Two beeps and a " > " indicate a good load. "ERR" or "MEM" FULL ERR" message indicates a bad tape or poor recorder performance.
LOMEM: *expr*	Similar to HIMEM: except sets lowest memory location available to BASIC. Automatically set at 2Ø48 when BASIC is entered with a control B*. Moving LOMEM: destroys current variable values.
MAN	Clears AUTO line numbering mode to all manual line numbering after a control C* or control X*.
NEW	Clears (Scratches) current BASIC program.
NO DSP *var*	Clears DSP mode for variable var.
NO TRACE	Clears TRACE mode.
RUN	Clears variables to zero, undimensions all arrays and executes program starting at lowest statement line number.
RUN *expr*	Clears variables and executes program starting at line number specified by expression *expr*.
SAVE	Stores (saves) a BASIC program on a cassette tape. Start tape recorder in record mode prior to hitting return key.
TEXT	Sets all text mode. Screen is formated to display alpha-numeric characters on 24 lines of 4Ø characters each. TEXT resets scrolling window to maximum.
TRACE	Sets debug mode that displays line number of each statement as it is executed.

* Control characters such as control X or control C are typed by holding down the CTRL key while typing the specified letter. This is similiar to how one holds down the shift key to type capital letters. Control characters are NOT displayed on the screen but are accepted by the computer. For example, type several control G's. We will also use a superscript C to indicate a control character as in X^C.

BASIC Operators

Symbol	Sample Statement	Explanation
Prefix Operators		
( )	1Ø X= 4*(5 + X)	Expressions within parenthesis ( ) are always evaluated first.
+	2Ø X= 1+4*5	Optional; +1 times following expression.
-	3Ø ALPHA = -(BETA +2)	Negation of following expression.
NOT	4Ø IF A NOT B THEN 2ØØ	Logical Negation of following expression; Ø if expression is true (non-zero), 1 if expression is false (zero).
Arithmetic Operators		
↑	6Ø Y = X 3	Exponentiate as in X^3 . NOTE: ↑ is shifted letter N.
*	7Ø LET DOTS=A*B*N2	Multiplication. NOTE: Implied multiplication such as (2 + 3)(4) is not allowed thus N2 in example is a variable not N * 2.
/	8Ø PRINT GAMMA/S	Divide
MOD	9Ø X = 12 MOD 7 1ØØ X = X MOD(Y+2)	Modulo: Remainder after division of first expression by second expression.
+	11Ø P = L + G	Add
-	12Ø XY4 = H-D	Substract
=	13Ø HEIGHT=15 14Ø LET SIZE=7*5 15Ø A(8) = 2 155 ALPHA$ = "PLEASE"	Assignment operator; assigns a value to a variable. LET is optional

Relational and Logical Operators

The numeric values used in logical evaluation are "true" if non-zero, "false" if zero.

Symbol	Sample Statement	Explanation
=	16Ø IF D = E THEN 5ØØ	Expression "equals" expression.
=	17Ø IF A$(1,1)= "Y" THEN 500	String variable "equal'string variable.
# or < >	18Ø IF ALPHA #X*Y THEN 5ØØ	Expression "does not equal" expression.
#	19Ø IF A$ # "NO" THEN 5ØØ	String variable "does not equal" string variable. NOTE: If strings are not the same length, they are considered un-equal. < > not allowed with strings.
>	2ØØ IF A>B THEN GO TO 5Ø	Expression "is greater than" expression.
<	21Ø IF A+1<B-5 THEN 1ØØ	Expression "is less than" expression.
>=	22Ø IF A>=B THEN 1ØØ	Expression "is greater than or equal to" expression.
<=	23Ø IF A+1<=B-6 THEN 2ØØ	Expression "is less than or equal to" expression.
AND	24Ø IF A>B AND C<D THEN 2ØØ	Expression 1 "and" expression 2 must both be "true" for statements to be true.
OR	25Ø IF ALPHA OR BETA+1 THEN 2ØØ	If either expression 1 or expression 2 is "true", statement is "true".

BASIC FUNCTIONS

Functions return a numeric result. They may be used as expressions or as part of expressions. PRINT is used for examples only, other statements may be used. Expressions following function name must be enclosed between two parenthesis signs.

FUNCTION NAME

ABS (*expr*)	300 PRINT ABS(X)	Gives absolute value of the expression *expr*.
ASC (*str$*)	310 PRINT ASC("BACK") 320 PRINT ASC(B$) 330 PRINT ASC(B$(4,4)) 335 PRINT ASC(B$(Y))	Gives decimal ASCII value of designated string variable str. If more than one character is in designated string or sub-string, it gives decimal ASCII value of first character.
LEN (*str$*)	340 PRINT LEN(B$)	Gives current length of designated string variable *str$*;i.e., number of characters.
PDL (*expr*)	350 PRINT PDL(X)	Gives number between Ø and 255 corresponding ponding to paddle position on game paddle number designated by expression expr and must be legal paddle (Ø,1,2,or 3) or else 255 is returned.
PEEK (*expr*)	360 PRINT PEEK(X)	Gives the decimal value of number stored of decimal memory location specified by expression *expr*. For MEMORY locations above 32676, use negative number; i.e., HEX location FFFØ is -16
RND (*expr*)	370 PRINT RND(X)	Gives random number between V and (expression expr -1) if expression expr is positive; if minus, it gives random number between Ø and (expression expr +1).
SCRN(*expr1*, *expr2*)	380 PRINT SCRN (X1,Y1)	Gives color (number between Ø and 15) of screen at horizontal location designated by expression *exprl* and vertical location designated by expression *expr2* Range of expression *exprl* is Ø to 39. Range of expression expr2 is Ø to 39 if in standar mixed colorgraphics display mode as set by GR command or Ø to 47 if in all color mode set by POKE -163Ø4 ,Ø: POKE - 163Ø2,Ø'.
SGN (*expr*)	39Ø PRINT SGN(X)	Gives sign (not sine) of expression *expr* i.e., -1 if expression *expr* is negative,zero zero and +1 if *expr* is positive.

BASIC STATEMENTS

Each BASIC statement must have a line number between Ø and 32767. Variable names must start with an alpha character and may be any number of alpha-numeric characters up to 1ØØ. Variable names may not contain buried any of the following words: AND, AT, MOD, OR, STEP, or THEN. Variable names may not begin with the letters END, LET, or REM. String variables names must end with a $ (dollar sign). Multiple statements may appear under the same line number if separated by a : (colon) as long as the total number of characters in the line (including spaces) is less than approximately 15Ø characters
Most statements may also be used as commands. BASIC statements are executed by RUN or GOTO commands.

NAME		
CALL *expr*	1Ø CALL-936	Causes execution of a machine level language subroutine at decimal memory location specified by expression *expr* Locations above 32767 are specified using negative numbers; i.e., location in example 1Ø is hexidecimal number $FC53
COLOR=*expr*	3Ø COLOR=12	In standard resolution color (GR) graphics mode, this command sets screen TV color to value in expression *expr* in the range Ø to 15 as described in Table A. Actually expression *expr* may be in the range Ø to 255 without error message since it is implemented as if it were expression *expr* MOD 16.
DIM *var1 (expr1)* *str$ (expr2)* *var2 (expr3)*	5Ø DIM A(2Ø),B(1Ø) 6Ø DIM B$(3Ø) 7Ø DIM C (2) Illegal: 8Ø DIM A(3Ø) Legal: 85 DIM C(1ØØØ)	The DIM statement causes APPLE II to reserve memory for the specified variables. For number arrays APPLE reserves approximately 2 times *expr* bytes of memory limited by available memory. For string arrays *-str$-(expr)* must be in the range of 1 to 255. Last defined variable may b'e redimensioned at any time; thus, example in line is illegal but 85 is allowed.
DSP*var*	Legal: 9Ø DSP AX: DSP L Illegal: 1ØØ DSP AX,B 1Ø2 DSP AB$ 1Ø4 DSP A(5) Legal: 1Ø5 A=A(5): DSP A	Sets debug mode that DSP variable *var* each time it changes and the line number where the change occured.

NAME	EXAMPLE	DESCRIPTION
END	11Ø END	Stops program execution. Sends carriage return and "> " BASIC prompt) to screen.
FOR *var=* *exp'21* TO*expr2* STEP*expr3*	11Ø FOR L=Ø to 39 12Ø FOR X=Y1 TO Y3 13Ø FOR 1=39 TO 1 15Ø GOSUB 1ØØ *J2	Begins FOR...NEXT loop, initializes variable var to value of expression *expr1* then increments it by amount in expression *expr3* each time the corresponding "NEXT" statement is encountered, until value of expression *expr2* is reached. If STEP *expr3* is omitted, a STEP of +1 is assumed. Negative numbers are allowed.
GOSUB *expr*	14Ø GOSUB 5ØØ	Causes branch to BASIC subroutine starting at legal line number specified by expression *expr* Subroutines may be nested up to 16 levels.
GOTO *expr*	16Ø GOTO 2ØØ 17Ø GOTO ALPHA+1ØØ	Causes immediate jump to legal line number specified by expression *expr.*
GR	18Ø GR 19Ø GR: POKE -163Ø2,Ø	Sets mixed standard resolution color graphics mode. Initializes COLOR = Ø (Black) for top 4Øx4Ø of screen and sets scrolling window to lines 21 through 24 by 4Ø characters for four lines of text at bottom of screen. Example 19Ø sets all color mode (4Øx48 field) with no text at bottom of screen.
HLIN *expr1,* *expr2*AT*expr3*	2ØØ HLIN Ø,39 AT 2Ø 21Ø HLIN Z,Z+6 AT I	In standard resolution color graphics mode, this command draws a horizontal line of a predefined color (set by COLOR=) starting at horizontal position defined by expression exprl and ending at position expr2 at vertical position defined by expression e*xpr3.expr1* and *expr2* must be in the range of Ø to 39 and *expr1* < = *expr2* . *expr3* be in the range of Ø to 39 (or Ø to 47 if not in mixed mode).

Note:

HLIN Ø, 19 AT Ø is a horizontal line at the top of the screen extending from left corner to center of screen and HLIN 2Ø,39 AT 39 is a horizontal line at the bottom of the screen extending from center to right corner.

IF expression THEN statement	22Ø IF A> B THEN PRINT A 23Ø IF X=Ø THEN C=1 24Ø IF A#1Ø THEN GOSUB 2ØØ 25Ø IF A$(1,1)# "Y" THEN 1ØØ Illegal: 26Ø IF L> 5 THEN 5Ø: ELSE 6Ø Legal: 27Ø IF L> 5 THEN 5Ø GO TO 6Ø	If *expression* is true (non-zero) then execute statement; if false do not execute statement. If *statement* is an expression, then a GOTO *expr* type of statement is assumed to be implied. The "ELSE" in example 26Ø is illegal but may be implemented as shown in example 27Ø.
INPUT *var1*, *var2*, *str$*	28Ø INPUT X,Y,Z(3) 29Ø INPUT "AMT", DLLR 3ØØ INPUT "Y or N?", A$	Enters data into memory from I/O device. If number input is expected, APPLE wil output "?"; if string input is expected no "?" will be outputed. Multiple numeric inputs to same statement may be separated by a comma or a carriage return. String inputs must be separated by a carriage return only. One pair of " " may be used immediately after INPUT to output prompting text enclosed within the quotation marks to the screen.
IN# *expr*	31Ø IN# 6 32Ø IN# Y+2 33Ø IN# 0	Transfers source of data for subsequent INPUT statements to peripheral I/O slot (1-7) as specified as by expression *expr.* Slot Ø is not addressable from BASIC. IN#Ø (Example 33Ø) is used to return data source from peripherial I/O to keyboard connector.
LET	34Ø LET X=5	Assignment operator. "LET" is optional
LIST *num1*, *num2*	35Ø IF X > 6 THEN LIST 5Ø	Causes program from line number *num1* through line number num2 to be displayed on screen.
NEXT *var1*, *var2*	36Ø NEXT I 37Ø NEXT J,K	Increments corresponding "FOR" variable and loops back to statement following "FOR" until variable exceeds limit.
NO DSP *var*	38Ø NO DSP I	Turns-off DSP debug mode for variable
NO TRACE	39Ø NO TRACE	Turns-off TRACE debug mode

<u>PLOT</u> *expr1, expr2*	4ØØ PLOT 15, 25 4ØØ PLT XV,YV	In standard resolution color graphics, this command plots a small square of a predefined color (set by COLOR=) at horizontal location specified by expression *expr1* in range Ø to 39 and vertical location specified by expression *expr2* in range Ø to 39 (or Ø to 47 if in all graphics mode) NOTE: PLOT Ø Ø is upper left and PLOT 39, 39 (or PLOT 39, 47) is lower right corner.
<u>POKE</u> *expr1, expr2*	42Ø POKE 2Ø, 4Ø 430 POKE 7*256, XMOD255	Stores <u>decimal</u> number defined by expression *expr2* in range of Ø 255 at <u>decimal</u> memory location specified by expression *expr1* Locations above 32767 are specified by negative numbers.
<u>POP</u>	44Ø POP	"POPS" nested GOSUB return stack address by one.
<u>PRINT</u> *var1, var, str$*	45Ø PRINT L1 46Ø PRINT Li, X2 47Ø PRINT "AMT=";DX 48Ø PRINT A$;B$; 49Ø PRINT 492 PRINT "HELLO" 494 PRINT 2+3	Outputs data specified by variable *var* or string variable *str$* starting at current cursor location. If there is not trailing "," or ";" (Ex 45Ø) a carriage return will be generated. Commas (Ex. 46Ø) outputs data in 5 left justified columns. Semi-colon (Ex. 47Ø) inhibits print of any spaces. Text imbedded in " " will be printed and may appear multiple times.
<u>PR#</u> *expr*	500 PR# 7	Like IN#, transfers output to I/O slot defined by expression *expr* PR# Ø is video output not I/O slot Ø.
<u>REM</u>	510 REM REMARK	No action. All characters after REM are treated as a remark until terminated by a carriage return.
<u>RETURN</u>	52Ø RETURN 53Ø IFX= 5 THEN RETURN	Causes branch to statement following last GOSUB; i.e., RETURN ends a subroutine. Do not confuse "RETURN" <u>statement</u> with Return <u>key</u> on keyboard.

TAB *expr*	53Ø TAB 24 54Ø TAB 1+24 55Ø IF A#B THEN TAB 2Ø	Moves cursor to absolute horizontal position specified by expression *expr* in the range of 1 to 4Ø. Position is left to right
TEXT	55Ø TEXT 56Ø TEXT: CALL-936	Sets all text mode. Resets scrolling window to 24 lines by 4Ø characters. Example 56Ø also clears screen and homes cursor to upper left corner
TRACE	570 TRACE 580 IFN >32ØØØ THEN TRACE	Sets debug mode that displays each line number as it is executed.
VLIN *expr1, expr2* AT *expr3*	59Ø VLIN Ø, 39AT15 6ØØ VLIN Z,Z+6ATY	Similar to HLIN except draws vertical line starting at *expr1* and ending at *expr2* at horizontal position *expr3*.
VTAB *expr*	61Ø VTAB 18 62Ø VTAB Z+2	Similar to TAB. Moves cursor to absolute vertical position specified by expression expr in the range 1 to 24. VTAB 1 is top line on screen; VTAB24 is bottom.

SPECIAL CONTROL AND EDITING CHARACTERS

"Control" characters are indicated by a super-scripted "C" such as G^C. They are obtained by holding down the CTRL key while typing the letter. Control characters are NOT displayed on the TV screen. B and C must be followed by a carriage return. Screen editing characters are indicated by a sub-scripted "E" such as D_E. They are obtained by pressing and releasing the ESC key then typing specified letter. Edit characters send information only to display screen and does not send data to memory. For example, U^C moves to cursor to right and copies text while A_E moves cursor to right but does not copy text.

CHARACTER	DESCRIPTION OF ACTION
RESET key	Immediately interrupts any program execution and resets computer. Also sets all text mode with scrolling window at maximum. Control is transfered to System Monitor and Apple prompts with a "*" (asterisk) and a bell. Hitting RESET key does NOT destroy existing BASIC or machine language program.
Control B	If in System Monitor (as indicated by a "*"), a control B and a carriage return will transfer control to BASIC, <u>scratching (killing) any existing BASIC program</u> and set HIMEM: to maximum installed user memory and LOMEM: to 2048.
Control C	If in BASIC, halts program and displays line number where stop occurred*. Program may be continued with a CON command. If in <u>System</u> Monitor, (as indicated by "*"), control C and a carraige return will enter BASIC <u>without</u> killing current program.
Control G	Sounds bell (beeps speaker)
Control H	Backspaces cursor and deletes any overwritten characters from computer but not from screen. Apply supplied keyboards have special key "÷" on right side of keyboard that provides this functions without using control button.
Control J	Issues line feed only
Control V	Compliment to H^C. Forward spaces cursor and copies over written characters. Apple keyboards have + key on right side which also performs this function.
Control X	Immediately deletes current line.

* If BASIC program is expecting keyboard input, you will have to hit carriage return key after typing control C.

CHARACTER	DESCRIPTION OF ACTION
A_E	Move cursor to right
B_E	Move cursor to left
C_E	Move cursor down
D_E	Move cursor up
E_E	Clear text from cursor to end of line
F_E	Clear text from cursor to end of page
$@_E$	Home cursor to top of page, clear text to end of page.

Table A: APPLE II COLORS AS SET BY COLOR =

Note: Colors may vary depending on TV tint (hue) setting and may also be changes by adjusting trimmer capacitor C3 on APPLE II P.C. Board.

0 = Black	8 = Brown
1 = Magnenta	9 = Orange
2 = Bark Blue	10 = Grey
3 = Light Purple	11 = Pink
4 = Dark Green	12 = Green
5 = Grey	13 = Yellow
6 = Medium Blue	14 = Blue/Green
7 = Light Blue	15 = White

Special Controls and Features

Hex	BASIC Example	Description
Display Mode Controls		
C05Ø	1Ø POKE -163Ø4,Ø	Set color graphics mode
C051	2Ø POKE -163Ø3,Ø	Set text mode
C052	3Ø POKE -163Ø2,Ø	Clear mixed graphics
C053	4Ø POKE -163Ø1,Ø	Set mixed graphics (4 lines text)
C054	5Ø POKE -163ØØ,Ø	Clear display Page. 2 (BASIC commands use Page 1 only)
C055	6Ø POKE -16299,Ø	Set display to Page 2 (alternate)
C056	7Ø POKE -16298,Ø	Clear HIRES graphics mode
C057	8Ø POKE -16297,Ø	Set HIRES graphics mode
TEXT Mode Controls		
ØØ2Ø	9Ø POKE 32,L1	Set left side of scrolling window to location specified by L1 in range of Ø to 39.
ØØ21	1ØØ POKE 33,W1	Set window width to amount specified by WI. L1+W1<4Ø. W1>Ø
ØØ22	11Ø POKE 34,T1	Set window top to line specified by T1 in range of Ø to 23
ØØ23	12Ø POKE 35,B1	Set window bottom to line specified by B1 in the range of Ø to 23. B1>T1
ØØ24	13Ø CH=PEEK(36) 14Ø POKE 36,CH 15Ø TAB(CH+1)	Read/set cusor horizontal position in the range of Ø to 39. If using TAB, you must add "1" to cusor positior read value; Ex. 14Ø and 15Ø perform identical function.
ØØ25	16Ø CV=PEEK (37) 17Ø POKE 37,CV 18Ø VTAB(CV+1)	Similar to above. Read/set cusor vertical position in the range Ø to 23.
ØØ32	19Ø POKE 5Ø,127 2ØØ POKE 5Ø,255	Set inverse flag if 127 (Ex. 19Ø) Set normal flag if 255(Ex. 2ØØ)
FC58	21Ø CALL -936	(@$_E$) Home cusor, clear screen
FC42	22Ø CALL -958	(F$_E$) Clear from cusor to end of page

Hex	BASIC Example	Description
FC9C	230 CALL -868	(E_E) Clear from cusor to end of line
FC66	240 CALL -922	(J^C) Line feed
FC70	250 CALL -912	Scroll up text one line

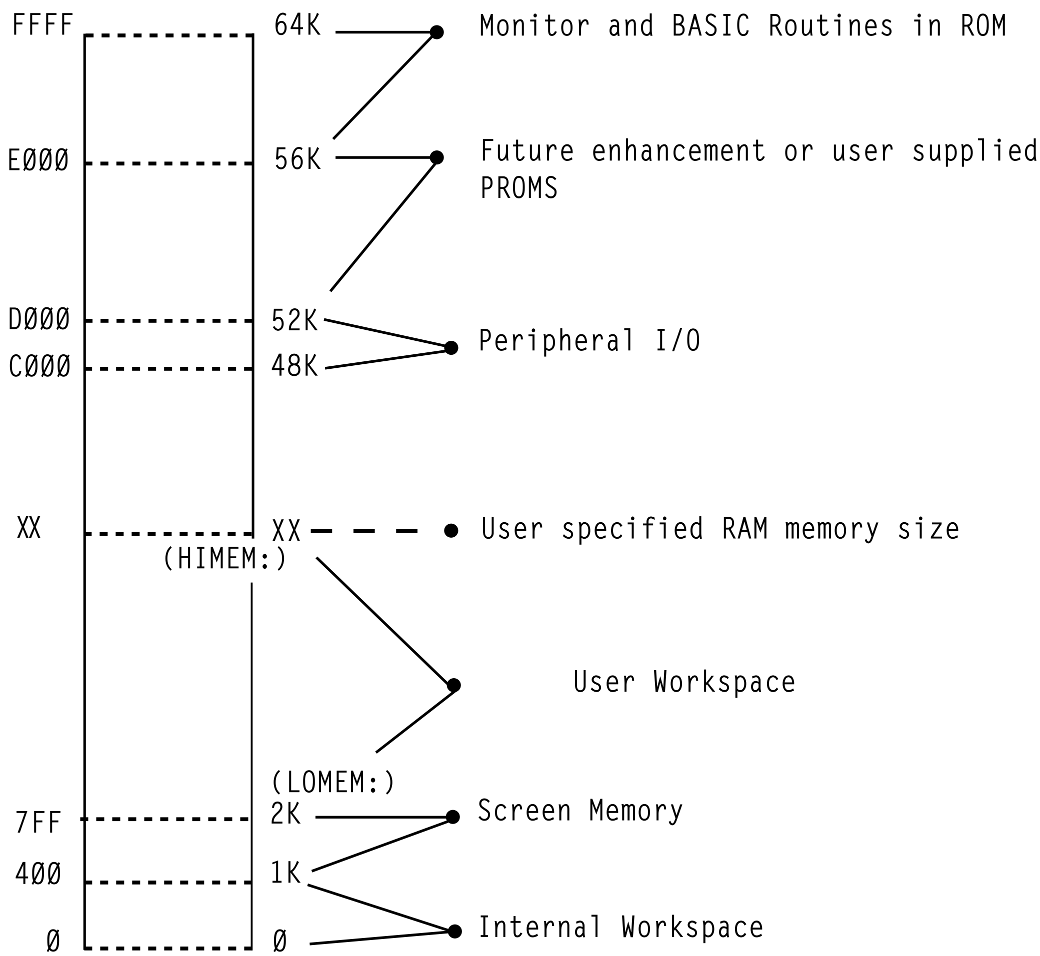
Miscellaneous

C030	360 X=PEEK(-16336) 365 POKE -16336,0	Toggle speaker
C000	370 X=PEEK(-16384)	Read keyboard; if X>127 then key was pressed.
C010	380 POKE -16368,0	Clear keyboard strobe - always after reading keyboard.
C061	390 X=PEEK(16287)	Read PDL(0) push button switch. If X>127 then switch is "on".
C062	400 X=PEEK(-16286)	Read PDL(1) push button switch.
C063	410 X=PEEK(-16285)	Read PDL(2) push button switch.
C058	420 POKE -16296,0	Clear Game I/O AN0 output
C059	430 POKE -16295,0	Set Game I/O AN0 output
C05A	440 POKE -16294,0	Clear Game I/O AN1 output
C05B	450 POKE -16293,0	Set Game I/O AN1 output
C05C	460 POKE -16292,0	Clear Game I/O AN2 output
C05D	470 POKE -16291,0	Set Game I/O AN2 output
C05E	480 POKE -16290,0	Clear Game I/O AN3 output
C05F	490 POKE -16289,0	Set Game I/O AN3 output

APPLE II BASIC ERROR MESSAGES

*** SYNTAX ERR	Results from a syntactic or typing error.
*** > 32767 ERR	A value entered or calculated was less than -32767 or greater than 32767.
*** > 255 ERR	A value restricted to the range Ø to 255 was outside that range.
*** BAD BRANCH ERR	Results from an attempt to branch to a non-existant line number.
*** BAD RETURN ERR	Results from an attempt to execute more RETURNs than previously executed GOSUBs.
*** BAD NEXT ERR	Results from an attempt to execute a NEXT statement for which there was not a corresponding FOR statement.
*** 16 GOSUBS ERR	Results from more than 16 nested GOSUBs.
*** 16 FORS ERR	Results from more than 16 nested FOR loops.
*** NO END ERR	The last statement executed was not an END.
*** MEM FULL ERR	The memory needed for the program has exceeded the memory size allotted.
*** TOO LONG ERR	Results from more than 12 nested parentheses or more than 128 characters in input line.
*** DIM ERR	Results from an attempt to DIMension a string array which has been previously dimensioned.
*** RANGE ERR	An array was larger than the DIMensioned value or smaller than 1 or HLIN,VLIN, PLOT, TAB, or VTAB arguments are out of range.
*** STR OVFL ERR	The number of characters assigned to a string exceeded the DIMensioned value for that string.
*** STRING ERR	Results from an attempt to execute an illegal string operation.
RETYPE LINE	Results from illegal data being typed in response to an INPUT statement. This message also requests that the illegal item be retyped.

Simplified Memory Map



READ/SAVE DATA SUBROUTINE

INTRODUCTION

Valuable data can be generated on the Apple II computer and sometimes it is useful to have a software routine that will allow making a permanent record of this information. This paper discusses a simple subroutine that serves this purpose.

Before discussing the Read/Save routines a rudimentary knowledge of how variables are mapped into memory is needed.

Numeric variables are mapped into memory with four attributes. Appearing in order sequentially are the Variable Name, the Display Byte, the Next Variable Address, and the Data of the Variable. Diagramatically this is represented as:

YN	DSP	NVA	DATA(0)	DATA(1)	,	DATA(N)
1			h_1	h_2		h_n+1

VARIABLE NAME - up to 100 characters represented in memory as ASCII equivalents with the high order bit set.

DSP (DISPLAY) BYTE - set to 01 when DSP set in BASIC initiates a process that displays this variable with the line number every time it is changed within a program.

NVA (NEXT VARIABLE ADDRESS) - two bytes (first low order, the second high order) indicating the memory location of the next variable.

DATA - hexadecimal equivalent of numeric information, represented in pairs of bytes, low order byte first.

String variables are formatted a bit differently than numeric ones. These variables have one extra attribute - a string terminator which designates the end of a string. A string variable is formatted as follows:

VN	DSP	NVA	DATA(Ø)	DATA(1)....	DATA(n)	ST
1			h_1	h_2	h_{n+1}	

VARIABLE NAME - up to 1ØØ characters represented in memory as ASCII equivalents with the high order bit set.

DSP (DISPLAY) BYTE - set to Ø1 when DSP set in BASIC, initiates a process that displays this variable with the line number every time it is changed within a program.

NVA (NEXT VARIABLE ADDRESS) - two bytes (first low order, the second high order) indicating the memory location of the next variable.

DATA - ASCII equivalents with high order bit set.

STRING TERMINATOR (ST) - none high order bit set character indicating END of string.

There are two parts of any BASIC program represented in memory. One is the location of the variables used for the program, and the other is the actual BASIC program statements. As it turns out, the mapping of these within memory is a straightforward process. Program statements are placed into memory starting at the top of RAM memory* unless manually shifted by the "HIMEM:." command, and are pushed down as each new (numerically larger) line numbered statement is entered into the system. Figure 1a illustrates this process diagramatically. Variables on the other hand are mapped into memory starting at the lowest position of RAM memory - hex $8ØØ (2Ø48) unless manually shifted by the"LOMEM:" command. They are laid down from there (see Figure 1b) and continue until all the variables have been mapped into memory or until they collide with the program statements. In the event of the latter case a memory full error will be generated

*Top of RAM memory is a function of the amount of memory. 16384 will be the value of "HIMEM:" for a 16K system.

The computer keeps track of the amount of memory used for the variable table and program statements. By placing the end memory location of each into \$CC-CD(2Ø4-2Ø5) and \$CA-CB(2Ø3-2Ø4), respectively. These are the BASIC memory program pointers and their values can be found by using the statements in Figure 2. CM defined in Figure 1 as the location of the end of the variable tape is equal to the number resulting from statement a of Figure 2. PP, the program pointer, is equal to the value resulting from statement 2b. These statements(Figure 2) can then be used on any Apple II computer to find the limits of the program and variable table.

FINDING THE VARIABLE TABLE FROM BASIC

First, power up the Apple II, reset it, and use the CTRL B (control B) command to place the system into BASIC initializing the memory pointers. Using the statements from Figure 2 it is found that for a 16K Apple II CM is equal to 2Ø48 and PP is equal to 16384. These also happen to be the values of OMEN and HIMEN: But this is expected because upon using the Bc command both memory pointers are initialized indicating no program statements and no variables.

To illustrate what a variable table looks like in Apple II memory suppose we want to assign the numeric variable A (\$C1 is the ASCII equivalent of a with the high order bit set) the value of -1 (FF FF in hex) and then examine the memory contents. The steps in this process are outlined in example I. Variable A is defined as equal to -1 (step 1). Then for convenience another variable - B - is defined as equal to Ø (step 2). Now that the variable table has been defined use of statement 2a indicates that CM is equal to 2Ø6Ø (step 3). LOMEN has not been readjusted so it is equal to 2Ø48. Therefore the variable table resides in memory from 2Ø48 (\$8ØØ hex) to 2Ø6Ø (\$88C). Depressing the "RESET" key places the Apple II into the monitor mode (step 4).

We are now ready to examine the memory contents of the variable table. Since the variable table resides from \$8ØØ hex to \$8ØC hex typing in "8ØØ.8ØC" and then depressing the "RETURN" key (step 5) will list the memory contents of this range. Figure 3 lists the contents with each memory location labelled. Examining these contents we see that Cl is equal to the variable name and is the memory equivalent of "A" and that FF FF is the equivalent of -1. From this, since the variable name is at the beginning of the table and the data is at the end, the variable table representation of A extends from \$8ØØ to \$8O5. We have then found

the memory range of where the variable A is mapped into memory. The reason forthis will become clear in the next section.

READ/SAVE ROUTINE

The READ/SAVE subroutine has three parts. The first section (lines Ø-1Ø) defines variable A and transfers control to the main program. Lines 2Ø through 26 represents the Write data to tape routine and lines 3Ø-38 represent the Read data from tape subroutine. Both READ and SAVE routines are executable by the BASIC "GOSUB X" (where X is 2Ø for write and 3Ø is for read) command. And as listed these routines can be directly incorporated into almost any BASIC program for read and saving a variable table. The limitation of these routines is that the whole part of a variable table is processed so it is necessary to maintain exactly the dimension statements for the variables used.

The variables used in this subroutine are defined as follows:

A =	record length, must be the first variable defined
CM=	the value obtained from statement a of figure 2
LM=	is equal to the value of "LOMEM:" Nominally 2Ø48

SAVING A DATA TABLE

The first step in a hard copy routine is to place the desired data onto tape. This is accomplished by determining the length of the variable table and setting A equal to it. Next within the main program when it is time to write the data a GOSUB2Ø statement will execute the write to tape process. Record length, variable A, is written to tape first (line 22) followed by the desired data (line 24). When this process is completed control is returned to the main program.

READING A DATA TABLE

The second step is to read the data from tape. When it is time a GOSUB3Ø statement will initiate the read process. First, the record length is read in and checked to see if enough memory is available (line 32-34). If exactly the same dimension statements are used it is almost guaranteed that there will be enough memory available. After this the variable table is read in (line 34) and control is then returned to the main program (line 36). If not enough memory is available then an error is generated and control is returned to the main program (line 38)

EXAMPLE OF READ/SAVE USAGE

The Read/Save routines may be incorporated directly into a main program. To illustrate this a test program is listed in example 2. This program dimensions a variable array of twenty by one, fills the array with numbers, writes the data table to tape, and then reads the data from tape listing the data on the video display. To get a feeling for how to use these routines enter this program and explore how the Read/Save routines work.

CONCLUSION

Reading and Saving data in the format of a variable table is a relatively straight forward process with the Read/Save subroutine listed in figure 4. This routine will increase the flexibility of the Apple II by providing a permanent record of the data generated within a program. This program can be reprocessed. The Read/Save routines are a valuable addition to any data processing program.

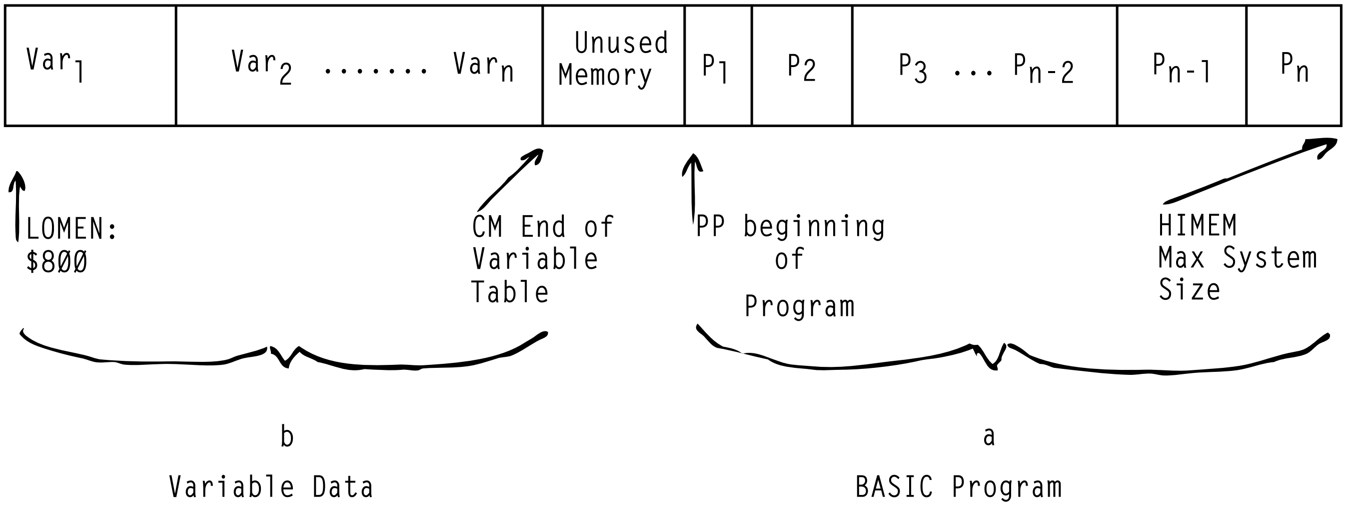


Figure 1

a) PRINT PEEK(2Ø4) + PEEK(2Ø5)*256 → PP

b) PRINT PEEK(2Ø2) + PEEK(2Ø3)*256 → CM

Figure 2

```
8ØØ  8Ø1  8Ø2  8Ø3  8Ø4  8Ø5  8Ø6  8Ø7  8Ø8  8Ø9  8ØA  8ØB  8ØC
 C1   ØØ   Ø6   Ø8   FF   FF   C2   ØØ   OC   Ø8   ØØ   ØØ   ØØ
            L    H    L    H              L    H
VAR  DSP    NVA      DATA     VAR  DSP    NVA      DATA
NAM                           NAM
```

Figure 3
$8ØØ.8ØC rewritten with labelling

FIGURE 4b

READ/SAVE PROGRAM	COMMENTS
Ø A=Ø	This must be the first statement in the program. It is initially Ø, but if data is to be saved, it will equal the length of the data base.
1Ø GOTO 1ØØ	This statement moves command to the main program.
2Ø PRINT "REWIND TAPE THEN START TAPE RECORDER": INPUT "THEN HIT RETURN", B$	Lines 20-26 are the write data to tape subroutine.
22 A=CM-LM: POKE 6Ø,4: POKE 61,8: POKE 62,5: POKE 63,8: CALL -3Ø7	
24 POKE 6Ø,LM MOD 256: POKE 61, LM/256: POKE 62, CM MOD 256: POKE 63, CM/256: CALL -3Ø7	Writing data table to tape
26 PRINT "DATA TABLE SAVED": RETURN	Returning control to main program.
3Ø PRINT "REWIND THE TAPE THEN START TAPE RECORDER": INPUT "AND HIT RETURN", B$	Lines 30-38 are the READ data from tape subroutine.
32 POKE 6Ø,4: POKE 61,8: POKE 62,5: POKE 63,8: CALL -259	
34 IF A<0 THEN 38: P=LM+A: IF P>HM THEN 38: CM=P: POKE 6Ø, LM MOD 256: POKE 61, LM/256: POKE 62, CM MOD 256: POKE 63, CM/256: CALL -259	Checking the record length (A) for memory requirements if everything is satisfactory the data is READ in.
36 PRINT "DATA READ IN": RETURN	
38 PRINT "***TOO MUCH DATA BASE***": RETURN	Returning control to main program.

NOTE: CM, LM and A must be defined within the main program.

1	>A=1 >	Define variable A=-1, then hit RETURN
2	>B=Ø >	Define variable B=Ø, then hit RETURN
3	>PRINT PEEK (2Ø4) + PEEK (2Ø5) * 256 computer responds with= 2Ø6Ø	Use statement 2a to find the end of the VARIABLE TABLE
4	> *	Hit the RESET key, Apple moves into Monitor mode.
5	*8ØØ.8ØC	Type in VARIABLE TABLE RANGE and HIT the RETURN KEY.

Computer responds with:

```
Ø8ØØ- C1 ØØ 86 Ø8 FF FF C2 ØØ

Ø8Ø8  ØC Ø8 ØØ ØØ ØØ
```

Example 1

Example 2

```
>LIST
  0 A=0
 10 GOTO 100
 20 REM WRITE DATA TO TAPE ROUTINE
 22 A=CM-LM: POKE 60,4: POKE 61
    ,8: POKE 62,5: POKE 63,8: CALL
    -307
 24 POKE 60,LM MOD 256: POKE 61
    ,LM/256: POKE 62,CM MOD 256
    : POKE 63, CM/256: CALL -307

 26 RETURN
 30 REM READ DATA SUBROUTINE
 32 POKE 60,4: POKE 61,8: POKE
    62,5: POKE 63,8: CALL -259
 34 IF A<0 THEN 38:P=LM+A: IF P>
    HM THEN 38: CM=P: POKE 60,LM MOD
    256: POKE 61,LM/256: POKE 62
    ,CM MOD 256: POKE 63,CM/256
    : CALL - 259
 36 RETURN
 38 PRINT "*** TOO MUCH DATA BASE **
    *":END
100 DIM A$(1),X(20)
105 FOR I=1 TO 20:X(I)=I: NEXT
    I
108 LM=2048:CM=2106:A=58:HM=16383
110 PRINT "20 NUMBERS GENERATED"

120 PRINT "NOW WE ARE GOING TO SAVE
    THE DATA": PRINT "WHEN YOU ARE R
    EADY START THE RECORDER IN RECOR
    D MORE": INPUT "AND HIT RETURN"
    ,A$
130 CALL -936: PRINT "NOW WRITING DA
    TA TO TAPE": GOSUB 20
135 PRINT "NOW THE DATA IS SAVE"

140 PRINT "NOW WE ARE GOING TO CLEAR
    THE X(20) TABLE AND READ THE DA
    TA FROM TAPE"
150 FOR I=1 TO 20:X(I): NEXT I
    "X(";I;")= ";X(I): NEXT I
160 PRINT "NOW START TAPE RECORDER"
    :INPUT "AND THEN HIT RETURN"
    ,A$
165 PRINT "A ",A
170 GOSUB 30
180 PRINT "ALL THE DATA READ IN"

190 FOR I-1 TO 20: PRINT "X(";I;
    ")=";X(I): NEXT I
195 PRINT "THIS IS THE END"
200 END
```

A SIMPLE TONE SUBROUTINE

INTRODUCTION

Computers can perform marvelous feats of mathematical computation at well beyond the speed capable of most human minds. They are fast, cold and accurate; man on the other hand is slower, has emotion, and makes errors. These differences create problems when the two interact with one another. So to reduce this problem humanizing of the computer is needed. Humanizing means incorporating within the computer procedures that aid in a program's usage. One such technique is the addition of a tone subroutine. This paper discusses the incorporation and usage of a tone subroutine within the Apple II computer.

Tone Generation

To generate tones in a computer three things are needed: a speaker, a circuit to drive the speaker, and a means of triggering the circuit. As it happens the Apple II computer was designed with a two-inch speaker and an efficient speaker driving circuit. Control of the speaker is accomplished through software.

Toggling the speaker is a simple process, a mere PEEK - 16336 ($CØ3Ø) in BASIC statement will perform this operation. This does not, however, produce tones, it only emits clicks. Generation of tones is the goal, so describing frequency and duration is needed, This is accomplished by toggling the speaker at regular intervals for a fixed period of time. Figure 1 lists a machine language routine that satisfies these requirements.

Machine Language Program

This machine language program resides in page Ø of memory from $Ø2 (2) to $14 (2Ø). $ØØ (ØØ) is used to store the relative period (P) between toggling of the speaker and $Ø1 (Ø1) is used as the memory location for the value of relative duration (Ø). Both P and D can range in value from $ØØ (Ø) to $FF (255). After the values for frequency and duration are placed into memory a CALL2 statement from BASIC will activate this routine. The speaker is toggled with the machine language statement residing at $Ø2 and then a

delay in time equal to the value in $ØØ occurs. This process is repeated until the tone has lasted a relative period of time equal to the duration (value in $Ø1) and then this program is exited (statement $14).

Basic Program

The purpose of the machine language routine is to generate tones controllable from BASIC as the program dictates. Figure 2 lists the appropriate statement that will deposit the machine language routine into memory. They are in the form of a subroutine and can be activated by a GOSUB 32ØØØ statement. It is only necessary to use this statement once at the beginning of a program. After that the machine language program will remain in memory unless a later part of the main program modifies the first 2Ø locations of page Ø.

After the GOSUB 32ØØØ has placed the machine language program into memory it may be activated by the statement in Figure 3. This statement is also in the form of a GOSUB because it can be used repetitively in a program. Once the frequency and duration have been defined by setting P and D equal to a value between Ø and 255 a GOSUB 25 statement is used to initiate the generation of a tone. The values of P and D are placed into $ØØ and $Ø1 and the CALL2 command activates the machine language program that toggles the speaker. After the tone has ended control is returned to the main program.

The statements in Figures 2 and 3 can be directly incorporated into BASIC programs to provide for the generation of tones. Once added to a program an infinite variety of tone combinations can be produced. For example, tones can be used to prompt, indicate an error in entering or answering questions, and supplement video displays on the Apple II computer system.

Since the computer operates at a faster rate than man does, prompting can be used to indicate when the computer expects data to be entered. Tones can be generated at just about any time for any reason in a program. The programmer's imagination can guide the placement of these tones.

CONCLUSION

The incorporation of tones through the routines discussed in this paper will aid in the humanizing of software used in the Apple computer. These routines can also help in transforming a dull program into a lively one. They are relatively easy to use and are a valuable addition to any program.

```
0000-   FF          ???
0000-   FF          ???
0002-   AD 30 C0    LDA   $C030
0005-   88          DEY
0006-   D0 04       BNE   $000C
0008-   C6 01       DEC   $01
000A-   F0 08       BEQ   $0014
000C-   CA          DEX
000D-   D0 F6       BNE   $0005
000F-   A6 00       LDX   $00
0011-   4C 02 00    JMP   $0002
0014-   60          RTS
```

FIGURE 1. Machine Language Program adapted from a program by P. Lutas.

```
32000 POKE 2,173: POKE 3,48: POKE
      4,192: POKE 5,136: POKE 6,208
      : POKE 7,4: POKE 8,198: POKE
      9,1: POKE 10,240
32005 POKE 11,8: POKE 12,202: POKE
      13,208: POKE 14,246: POKE 15
      ,166: POKE 16,0: POKE 17,76
      : POKE 18,2: POKE 19,0: POKE
      20,96: RETURN
```

FIGURE 2. BASIC "POKES"

```
25 POKE 0,P: POKE 1,D: CALL 2:
   RETURN
```

FIGURE 3. GOSUB

High-Resolution Operating Subroutines

These subroutines were created to make programming for High-Resolution Graphics easier, for both BASIC and machine. language programs. These subroutines occupy 757 bytes of memory and are available on either cassette tape or Read-Only Memory (ROM). This note describes use and care of these subroutines.

There are seven subroutines in this package. With these, a programmer can initialize High-Resolution mode, clear the screen, plot a point, draw a line, or draw and animate a predefined shape. on the screen. There are also some other general-purpose subroutines to shorten and simplify programming.

BASIC programs can access these subroutines by use of ,the CALL statement, and can pass information by using the POKE statement. There are special entry points for most of the subroutines that will perform the same functions as the original subroutines without modifying any BASIC pointers or registers. For machine language programming, a JSR to the appropriate subroutine address will perform the same function as a BASIC CALL.

In the following subroutine descriptions, all addresses given will be in decimal. The hexadecimal substitutes will be preceded by a dollar sign ($). All entry points given are for the cassette tape subroutines, which load into addresses CØØ to FFF (hex). Equivalent addresses for the ROM subroutines will be in *italic type face.*

High-Resolution Operating Subroutines

INIT Initiates High-Resolution Graphics mode.

From BASIC: CALL 3072 (or CALL -12288)

From machine language: JSR $C00 (or JSR $D000)

This subroutine sets High-Resolution Graphics mode with a 280 x 160 matrix of dots in the top portion of the screen and four lines of text in the bottom portion of the screen. INIT also clears the screen.

CLEAR Clears the screen.

From BASIC: CALL 3886 (or CALL -12274)

From machine language: JSR SCOE (or JSR $L000E)

This subroutine clears the High-Resolution screen without resetting the High-Resblution Graphics mode.

PLOT Plots a point on the screen.

From BASIC: CALL 3780 (or CALL -21589)

From machine language: JSR $C7C (or JSR $L107C)

This subroutine plots a single point on the screen. The X and Y coodinates of the point are passed in locations 800, 801, and 802 from BASIC, or in the A, X, and Y registers from machine language. The Y (vertical) coordinate can be from 0

PLOT (continued)

(top of screen) to 159 (bottom of screen) and is passed in location 8Ø2 or the A-register; but the X (horizontal) coordinate can range from Ø (left side of screen) to 279 (right side of screen) and must be split between locations 8ØØ (X MOD 256) and 8Ø1 (X/256).or, from machine language, between registers X (X LO) and Y (X HI). The color of the point to be plotted must be set in location 812 ($32C). Four colors are possible: Ø is BLACK, 85 ($55) is GREEN, 17Ø ($AA) is VIOLET, and 255 ($FF) is WHITE.

POSN Positions a point on the screen.

From BASIC: CALL 3761 *(or CALL -11599)*

From machine language: JSR $C26 *(or JSR $DØ26)*

This subroutine does all calculations for a PLOT, but does not plot a point (it leaves the screen unchanged). This is useful when used in conjumction with LINE or SHAPE (described later). To use this subroutine, set up the X and Y coordinates just the same as for PLOT. The color in location 812 ($32C) is ignored.

LINE Draw a line on the screen.

<u>High-Resolution Operating Routines</u>

<u>LINE</u>   Draws a line on the screen.

From BASIC: CALL 3786 *(or CALL -11574)*

From machine language: JSR $C95 *(or JSR $DØ95)*

This subroutine draws a line from the last point PLOTted or POSN'ed to the point specified. One endpoint is the last point PLOTted or POSN'ed; the other endpoint is passed in the same manner as for a PLOT or POSN. The color of the line is set in location 812 ($32C). After the line is drawn, the new endpoint becomes the base endpoint for the next line drawn.

<u>SHAPE</u>   Draws a predefined shape on the screen.

From BASIC: CALL 38Ø5 *(or CALL -11555)*

From machine language: JSR $DBC *(or JSR $D1BC)*

This subroutine draws a predefined shape on the screen at the point previously PLOTted or POSN'ed. The shape is defined by a *table of vectors* in memory. (How to create a vector table will be described later). The starting address of this table should be passed in locations 804 and 805 from BASIC or in the Y and X registers from machine language. The color of the shape should be passed in location 28 ($1C).

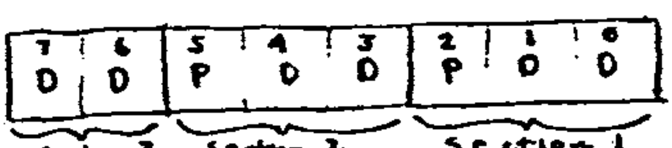
There are two special variables that are used only with shapes: the <u>scaling factor</u> and the <u>rotation factor</u>. The scaling factor determines the relative size of the shape. A scaling factor of

SHAPE (continued)

1 will cause the shape to be drawn true size, while a scaling factor of 2 will draw the shape double size, etc. The scaling factor is passed in location 8Ø6 from BASIC or $32F from machine language. The rotation factor specifies one of 64 possible angles of rotation for the shape. A rotation factor of Ø will cause the shape to be drawn right-side up, where a rotation factor if 16 will draw the shape rotated 90° clockwise, etc. The rotation factor is passed in location 8Ø7 from BASIC of in the A-register from machine language.

The table of vectors which defines the shape to be drawn is a series of bytes stored in memory. Each byte is divided into three sections, and each section specifies whether or not to plot a point and also a direction to move (up, down, left, or right). The SHAPE subroutine steps through the vector table byte by byte, and then through each byte section by section. When it reaches a ØØ byte, it is finished.

The three sections are arranged in a byte like this:



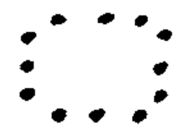
DD = ØØ Move ↑
Ø1 " →
1Ø " ↓
11 " ←

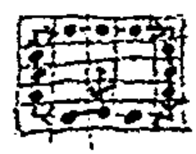
Each bit pair DD specifies a direction to move, and the two bits P specify whether or not to plot a point before moving. Notice that the last section (most significant bits) does not have a P field, so it can only be a move without plotting. The SHAPE

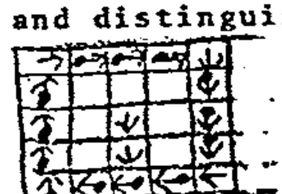
SHAPE (continued)

subroutine processes the sections from right to left (least significant bit to most significant bit). IF THE REMAINING SECTIONS OF THE BYTE ARE ZERO, THEN THEY ARE IGNORED. Thus, the byte cannot end with sections of ØØ (move up without plotting).

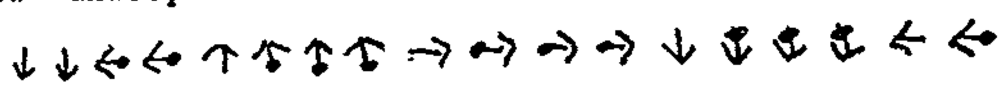
Here is an example of how to create a vector table:

Suppose we want to draw a shape like this: 

First, draw it on graph paper, one dot per square. Then decide where to start drawing the shape. Let's start this one in the center. Next, we must draw a path through each point in the shape, using only 90° angles on the turns:

Next, re-draw the shape as a series of vectors, each one moving one place up, down, left, or right, and distinguish the vectors that plot a point before moving:



Now "unwrap" those vectors and write them in a straight line.



Now draw a table like the one in Figure 1. For each vector in the line, figure the bit code and place it in the next available section in the table. If it will not fit or is a ØØ at the end of a byte, then skip that section and go on to the next. When you have finished

SHAPE (continued)

coding all vectors, check your work to make sure it is accurate. Then make another table (as in figure 2) and re-copy the coded vectors from the first table. Then decode the vector information into a series of hexadecimal bytes, using the hexidecimal code table in figure 3. This series of hexadecimal bytes is your shape definition table, which you can now put into the Apple II's memory and use to draw that shape on the screen.

Shape vectors: ↓↓←←↑↑↑↑→→→→↓↓↓↓←←

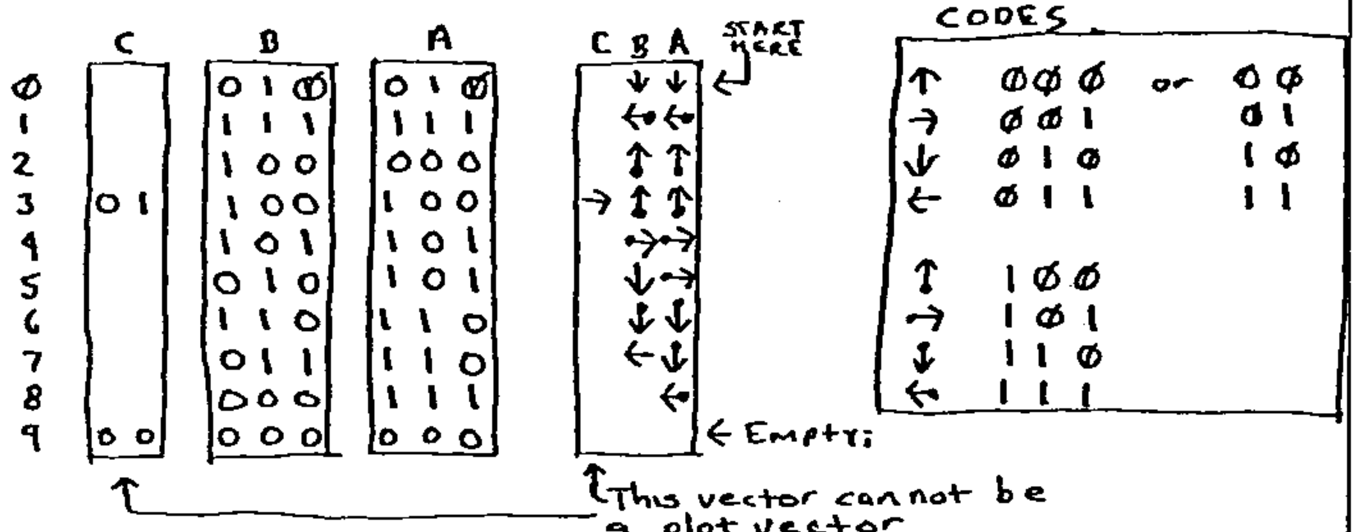


Figure 1.

	C | B | A		=
Ø	0 0 0 1	Ø 0 1 Ø	1 2
1	0 0 1 1	1 1 1 1	3 F
2	0 0 1 0	0 0 0 0	2 Ø
3	0 1 1 0	0 1 0 0	6 4
4	0 0 1 0	1 1 0 1	2 D
5	0 0 0 1	0 1 0 1	1 5
6	0 0 1 1	0 1 1 0	3 6
7	0 0 0 1	1 1 1 0	1 E
8	0 0 0 0	0 1 1 1	Ø 7
9	0 0 0 0	0 0 0 0	Ø Ø ← Empty; denotes end of vector table.

Figure 2.

Hex-decimal Codes

0000	→	0
0001	→	1
0010	→	2
0011	→	3
0100	→	4
0101	→	5
0110	→	6
0111	→	7
1000	→	8
1001	→	9
1010	→	A
1011	→	B
1100	→	C
1101	→	D
1110	→	E
1111	→	F

```
>REM HIRES DEMO-BASIC LISTING

>LIST
  1 INIT=3072:CLEAR=3086:POSN=3761
    :PLOT=3780:LINE=3786:SHAPE=
    3805:FIND=3667:SINTBL=3840
  5 DIM X(10),Y(10)
 10 TEXT : CALL -936: VTAB 4: TAB
    10: PRINT "*** 16K APPLE II ***"
    : PRINT "  *** HIGH RESOLUTION G
    RAPHICS DEMOS ***": PRINT
 15 PRINT "1 RANDOM LINE DRAW AT BAS
    IC SPEED": PRINT "2 RANDOM SHAPE
     PROJECTED INTO CORNER"
 20 PRINT "3 CHRIS' MAD FOLLY":
     PRINT "4 RANDOM SHAPE SPIRALING
     INTO POINT": PRINT "5 SPIROGRAP
    H"
 25 PRINT "6 HI-RES DONUT": PRINT
    "7 RANDOM WAVE FORM": PRINT
    "8 SUM OF TWO SINE WAVES"
 30 PRINT : PRINT "HIT ANY KEY FOR N
    EW DEMO": PRINT "TYPE 'CONTROL C
    ' ; RETURN BUTTON THEN   TYPE 'T
    EXT AND RETURN BUTTON TO STOP"

 50 PRINT : INPUT "WHICH DEMO # DO Y
    OU WANT ",X1
 90 IF X1<1 OR X1>8 THEN 10: CALL
    INIT: GOTO 100*X1
100 CALL INIT:X=40:Y=X: GOSUB 2000
    : POKE 812,255: CALL PLOT
110 X= RND (280):Y= RND (160): GOSUB
    2000: CALL LINE: IF NOT RND
    (300) THEN POKE 28,( PEEK (
    28)+ RND (3)+1) MOD 4*85: GOSUB
    3000: GOTO 110
200 GOSUB 1000:X= RND (2)*279:Y=
     RND (2)*159: CALL PLOT: FOR
    J=1 TO 30: FOR I=1 TO R: POKE
    800,X(I) MOD 256: POKE 801,
    X(I)>255: POKE 802,Y(I): CALL
    LINE
210 X(I)=(X(I)-X)*9/10+X:Y(I)=(
    Y(I)-Y)*9/10+Y: NEXT I,J: GOSUB
    3000: GOTO 200
300 CALL INIT:X= RND (24)*10+20
    :Y= RND (14)*10+20: POKE 812
    , RND (3)*85+85: GOSUB 2000
    : CALL PLOT
310 IF RND (1000)<1 THEN 300: IF
    NOT RND (200) THEN POKE 28,
     RND (4)*85
320 X1=X+( RND (3)-1)*25:Y1=Y+(
     RND (3)-1)*15: IF X1<0 OR
    X1>279 OR Y1<0 OR Y1>159 THEN
    320
330 X=X1:Y=Y1: GOSUB 2000: CALL
    LINE: GOSUB 3000: GOTO 310
400 GOSUB 1000: POKE 812, RND (
    3)*85+85: CALL PLOT
410 FOR J=1 TO 25: FOR I=1 TO R:
     POKE 800,X(I) MOD 256: POKE
    801,X>255: POKE 802,Y(I): CALL
    LINE
420 X=(X(I)-80+(Y(I)-80)/8)*9/10
    +80:Y(I)=(Y(I)-80-(X(I)-80)
    /8)*9/10+80:X(I)=X: NEXT I,
    J: GOSUB 3000: GOTO 400
500 CALL INIT: POKE 800,0: CALL
    PLOT:X=0:Y=0:XDIR=1:YDIR=1:
    A=5:B=3:C=8
510 POKE 800,0: POKE 801,0: POKE
    802,Y: CALL LINE: POKE 800,
    (279-X) MOD 256: POKE 801,X<
    24: POKE 802,159: CALL LINE:
     POKE 800,23: POKE 801,1: POKE
    802,159-Y: CALL LINE
515 IF RND (500) THEN 520:A=1+ RND
    (13):B=2+ RND (8):C=4+ RND
    (7)
520 POKE 800,X MOD 256: POKE 801
    ,X>255: POKE 802,0: CALL LINE:
    X=X+XDIR*A: IF X>=0 AND X<280
     THEN 530:XDIR=-XDIR:X=-X: IF
    X<0 THEN X=X+558
530 IF RND (500)<C THEN POKE 28
    , RND (4)*85:Y=Y+YDIR*B: IF
    Y>=0 AND Y<160 THEN 510:YDIR=
    -YDIR:Y=-Y: IF Y<0 THEN Y=Y+
    318: GOSUB 3000: GOTO 510
600 POKE -16302,0: POKE 768,5: POKE
    769,0: POKE 800,140: POKE 801
    ,0: POKE 802,0: POKE 804,0:
     POKE 805,3: POKE 812,255: CALL
    POSN
610 FOR R=0 TO 4160: POKE 807,R MOD
    64: POKE 806,2+6* NOT (R MOD
    65): CALL SHAPE: NEXT R: GOSUB
    3000: GOTO 610
700 J= RND (10)+ RND (10):K= RND
    (33)+ RND (31)+ RND (60):L=
     RND (9)/8: PRINT "FREQ#1= "
    ;J;"  FREQ#2= ";K
710 GOSUB 4000: GOSUB 3000: GOTO
    700
800 INPUT "REL FREQ #1=",J: INPUT
    "REL FREQ #2=",K: INPUT "MODE (0
    =SOLID, 1=POINTS)",L
810 GOSUB 4000: GOSUB 3000: GOTO
    800
1000 CALL CLEAR: POKE 812, RND (
    3)*85+85:R= RND (3)+2+ RND
    (2): FOR I=1 TO R:X(I)= RND
    (160):Y(I)= RND (160): NEXT
    I
1010 X=X(1):Y=Y(1): GOSUB 2000: RETURN
2000 POKE 800,X MOD 256: POKE 801
    ,X>255: POKE 802,Y: RETURN

3000 IF PEEK (-16384)<128 THEN RETURN
    : POKE -16368,0: POP : GOTO
    10
4000 CALL INIT: POKE 812,255:A=0
    :B=0: FOR I=0 TO 279:A=(A+J)
     MOD 256:B=(B+K) MOD 256:Y=
    ( PEEK (SINTBL+A)+ PEEK (SINTBL+
    B))*5/16
4010 POKE 800,I MOD 256: POKE 801
    ,I>255: POKE 802,Y: CALL LINE-
    6*( NOT I OR L): NEXT I: RETURN
```

ROD'S COLOR PATTERN

PROGRAM DESCRIPTION

ROD'S COLOR PATTERN is a simple but eloquent program. It generates a continuous flow of colored mosaic-like patterns in a 40 high by 40 wide block matrix. Many of the patterns generated by this program are pleasing to the eye and will dazzle the mind for minutes at a time.

REQUIREMENTS

4K or greater Apple II system with a color video display.
BASIC is the programming language used.

PROGRAM LISTING

```
100 GR
105 FOR W=3 TO 50
110 FOR I=1 TO 19
115 FOR J=0 TO 19
120 K=I+J
130 COLOR=J*3/(I+3)+I*W/12
135 PLOT I,K: PLOT K,I: PLOT 40
   -I,40-K
136 PLOT 40-K,40-I: PLOT K,40-I:
    PLOT 40-I,K: PLOT I,40-K: PLOT
   40-K,I
140 NEXT J,I
145 NEXT W: GOTO 105
```

PROGRAM LISTING: PONG

```
  5 REM  PONG BY WENDELL BITTER
 10 REM       7/7/77
 15 REM  PADDLE SWITCHES CONTROL
         PADDLE SIZE AFTER A MISS
         OR DURING A HIT
 20 GR
 25 DIM P(3): DIM HP$(10)
 30 A=38:B=1:C=-1
 35 COLOR=13: HLIN 1,38 AT 0: HLIN
    1,38 AT 39
 40 CALL -936: VTAB 23: INPUT "HANDB
    ALL OR PONG ? ",HP$
 45 INPUT "PADDLE SIZE (1-6) ",
    PS: IF PS<1 OR PS>6 THEN 45
    :S=PS-1
 50 CALL -936
 55 IF HP$(1)#"H" THEN 205
 60 H=1: COLOR=13: VLIN 0,39 AT
    39: GOTO 205
 65 FOR X=A TO B STEP C
 70 Y=YY+V: IF Y>1 AND Y<38 THEN
    80: IF Y<1 THEN Y=1: IF Y>38
     THEN Y=38
 75 V=-V: FOR T=1 TO 5:N= PEEK
    (-16336): NEXT T
 80 IF X=C OR X=39+C THEN 85: COLOR=
    0: PLOT X-C,YY: COLOR=15: PLOT
    X,Y
 85 YY=Y: IF X MOD 2=0 THEN GOSUB
    235: NEXT X
 90 GOSUB 235
 95 IF SCRN(X,Y+V*(Y+V<40 AND Y+
    V>-1))=0 THEN 165
100 FOR T=1 TO 10:N= PEEK (-16336
    ): NEXT T
105 IF H AND C>0 THEN 130
110 PP=P(X/38)
115 IF Y=PP THEN V=3: IF Y=PP+1
     THEN V=2: IF Y=PP+2 THEN V=
    1
120 IF Y=PP+3 THEN V=-1: IF Y=PP+
    4 THEN V=-2: IF Y=PP+5 THEN
    V=-3
125 IF S=0 THEN V=3- RND (7)
130 COLOR=0: PLOT X-C,Y
135 IF (H AND C>0) OR (VYO= ABS
    (V) AND X=0) THEN V=4- RND
    (9)
140 IF X=0 THEN VYO= ABS (V)
145 A=39-A:B=39-B:C=-C
150 IF PEEK (-16286)>127 AND S#
    5 THEN S=S+1
155 IF PEEK (-16287)>127 AND S#
    0 THEN S=S-1
160 GOTO 65
165 COLOR=0: PLOT X-C,Y
170 COLOR=15: PLOT X,Y+V*(Y+V>-
    1 AND Y+V<40)
175 FOR T=1 TO 75:N= PEEK (-16336
    )+ PEEK (-16336)- PEEK (-16336
    ): NEXT T
180 IF X=0 THEN SR=SR+1: IF X=39
     THEN SL=SL+1
185 VTAB 23: TAB 7: PRINT SL;: TAB
    33: PRINT SR
190 COLOR=0: PLOT X-C,Y
195 IF SL=15 OR SR=15 THEN 260
200 COLOR=0: PLOT X,Y+V*(Y+V)-1
     AND Y+V<40)
205 FOR T=1 TO 75: IF T MOD 5#0
     THEN 210: IF PEEK (-16286)
    >127 AND S#5 THEN S=S+1: IF
    PEEK (-16287)>127 AND S#0 THEN
    S=S-1
210 GOSUB 235: NEXT T
215 YY=P(0): IF X=0 THEN YY=P(1
    )
220 IF H THEN YY= RND (37)+1
225 V=1- RND (3)
230 GOTO 65
235 IF H THEN 245:P(1)=(( PDL (
    1)-24)*20)/115: IF P(1)=P(3
    ) THEN 245: IF P(1)<0 THEN
    P(1)=0: IF P(1)+S>39 THEN P(
    1)=39-S
240 COLOR=6: VLIN P(1),P(1)+S AT
    39: COLOR=0: IF P(1)>P(3) THEN
    VLIN 0,P(1)-1 AT 39: IF P(1
    )<P(3) THEN VLIN P(1)+S+1,39
     AT 39:P(3)=P(1)
245 P(0)=(( PDL (0)-24)*20)/115
    : IF P(0)<0 THEN P(0)=0: IF
    P(0)=P(2) THEN RETURN : IF
    P(0)+S>39 THEN P(0)=39-S
250 COLOR=6: VLIN P(0),P(0)+S AT
    0: COLOR=0: IF P(0)>P(2) THEN
    VLIN 0,P(0)-1 AT 0: IF P(0)
    <P(2) THEN VLIN P(0)+S+1,39
     AT 0
255 COLOR=0: IF P(0)>P(2) THEN
    VLIN 0,P(0)-1 AT 0: IF P(0)
    <P(2) THEN VLIN P(0)+S+1,39
     AT 0:P(2)=P(0): RETURN
260 PRINT "": END
265 END
```

COLOR SKETCH

PROGRAM DESCRIPTION

Color Sketch is a little program that transforms the Apple II into an artist's easel, the screen into a sketch pad. The user as an artist has a 40 high by 40 wide (1600 blocks) sketching pad to fill with a rainbow of fifteen colors. Placement of colors is determined by controlling paddle inputs; one for the horizontal and the other for the vertical. Colors are selected by depressing a letter from <u>A</u> through <u>P</u> on the keyboard.

An enormous number of distinct pictures can be drawn on the sketch pad and this program will provide many hours of visual entertainment.

REQUIREMENTS

This program will fit into a 4K system in the BASIC mode.

PROGRAM LISTING: COLOR SKETCH

```
5 POKE 2,173: POKE 3,48: POKE
   4,192: POKE 5,165: POKE 6,0
   : POKE 7,32: POKE 8,168: POKE
   9,252: POKE 10,165: POKE 11
   ,1: POKE 12,208: POKE 13,4
10 POKE 14,198: POKE 15,24: POKE
   16,240: POKE 17,5: POKE 18,
   198: POKE 19,1: POKE 20,76:
   POKE 21,2: POKE 22,0: POKE
   23,96
15 DIM B$(40): TEXT : CALL -936
   : GOTO 90
20 CALL -936: GOTO 90
25 A= LEN(B$): FOR Z=1 TO A: GOSUB
   65: PRINT B$(Z,Z);: NEXT Z:
   GOSUB 70: RETURN
30 B$="***************************
   *************": RETURN
35 B$="COLOR SKETCH": RETURN
40 B$="COPYRIGHT APPLE COMPUTER 197
   7": RETURN
45 B$="THIS PROGRAM ALLOWS YOU TO "
   : RETURN
50 B$="SKETCH COLORED FIGURES IN"
   : RETURN
55 B$="LOW RESOLUTION GRAPHICS WITH
   PADDLES": RETURN
60 KK=20:TON=20: GOSUB 85: RETURN

65 KK=10:TON=10: GOSUB 85: RETURN

70 KK=20:TON=50: GOSUB 85:KK=30
   :TON=90: GOSUB 85: RETURN
75 KK=20:TON=20: GOSUB 85: RETURN

80 KK=8:TON=250: GOSUB 85:KK=9
   :TON=250: GOSUB 85: RETURN
85 POKE 1,TON MOD 256: POKE 24
   ,TON/256+1: POKE 0,KK: CALL
   2: RETURN
90 GOSUB 30: GOSUB 25: PRINT :
   TAB 13: GOSUB 35: GOSUB 25
   : PRINT : GOSUB 30: GOSUB 25
   : PRINT : TAB 5: GOSUB 40: GOSUB
   25: PRINT : GOSUB 30: GOSUB
   25
95 PRINT : GOSUB 70: GOSUB 45:
   GOSUB 25: PRINT : GOSUB 50
   : GOSUB 25: PRINT : GOSUB 55
   : GOSUB 25: PRINT
100 PRINT : PRINT : GOSUB 70: INPUT
   "WHEN READY HIT RETURN",B$
105 GR
110 B$="ABCDEFGHIJKLMNOP": CALL
   -936
115 FOR Z=0 TO 15: COLOR=Z: PLOT
   Z*2+4,39: VTAB 21: GOSUB 75
   : TAB Z*2+5: PRINT B$(Z+1,Z+
   1);: GOSUB 75: NEXT Z: TAB
   1
120 VTAB 22:B$="TYPE A LETTER TO  CH
   ANGE COLOR.": GOSUB 25: PRINT
   :B$="TYPE SPACE BAR TO STOP PLOT
   .": GOSUB 25: PRINT
125 Y= PDL (1)*38/255:X= PDL (0
   )*39/255: VTAB 24: TAB 1: PRINT
   "CURSOR POSITION: X=";X;"   Y="
   ;Y;"    ";
130 IF PEEK (-16384)>127 THEN 145
   : IF X1=X AND Y1=Y THEN 125
   : COLOR=C2: PLOT X1,Y1: IF
   NOT FLAG THEN 135: COLOR=C:
   PLOT X,Y
135 C2= SCRN(X,Y):C3=15: IF C2=
   15 THEN C3=5: COLOR=C3: PLOT
   X,Y:X1=X:Y1=Y
140 GOTO 125
145 IF PEEK (-16384)#160 THEN 155
   :FLAG=0: POKE -16368,0: POKE
   34,20: COLOR=0: HLIN 0,39 AT
   39: CALL -936
150 PRINT :B$="CONTINUE OR STOP"
   : VTAB 24: GOSUB 25: INPUT
   " (C/S) ",B$: IF B$(1,1)="C"
   THEN 110: PRINT "END": END

155 FLAG=1:C= PEEK (-16384)-193
   : POKE -16368,0: GOTO 125
```

MASTERMIND PROGRAM

PROGRAM DESCRIPTION

MASTERMIND is a game of strategy that matches your wits against Apple's. The object of the game is to choose correctly which 5 colored bars have been secretly chosen by the computer. Eight different colors are possible for each bar - Red (R), Yellow (Y), Violet (V), Orange (O), White (W), and Black (B). A color may be used more than once. Guesses for a turn are made by selecting a color for each of the five hidden bars. After hitting the RETURN key Apple will indicate the correctness of the turn. Each white square to the right of your turn indicates a correctly colored and positioned bar. Each grey square acknowledges a correctly colored but improperly positioned bar. No squares indicate you're way off.

Test your skill and challenge the Apple II to a game of MASTERMIND.

REQUIREMENTS

8K or greater Apple II computer system.
BASIC is the programming language.

PROGRAM LISTING: MASTERMIND

```
0 REM GAME OF MASTERMIND  8-25-77
   WOZ (APPLE COMPUTER)
10 DIM A(6),C(8),D(5),X(8),X$(
   8):X(1)=2:X(2)=12:X(3)=1:X(
   4)=13:X(5)=3:X(6)=9:X(7)=15
   :X(8)=5:X$="BGRYVOWX"
20 TEXT : CALL -936: PRINT "


WELCO
   ME TO THE GAME OF MASTERMIND!

    YOUR OBJECT IS TO GUESS 5 COLOR
   S (WHICH"
30 PRINT "I WILL MAKE UP) IN THE MI
   NIMUM NUMBER   OF GUESSES.  THER
   E ARE EIGHT DIFFERENT  COLORS TO
    CHOSE FROM."
40 PRINT "

FEWER THAN 7 GUESSES--EXC
   ELLENT": PRINT "   7 TO 9 GUESSE
   S-----GOOD": PRINT "  10 TO 14 G
   UESSES----AVERAGE"
50 PRINT "MORE THAN 14 GUESSES--POO
   R


": CALL -384: TAB 7: PRINT
   "HIT ANY KEY TO BEGIN PLAY"

100 CALL -380: IF PEEK (-16384)
   <132 THEN 100: POKE -16368,
   0: GR : PRINT : FOR I=1 TO
   8:C(I)= RND (8)+1: COLOR=X(
   I): HLIN I*4-2,I*4 AT 39: PRINT
   "   ";X$(I,I);: NEXT I
110 TRY=0: PRINT : PRINT "   LETTER
   KEYS FOR COLOR CHANGE": PRINT
   "  ARROW KEYS FOR ADVANCE AND BA
   CK": PRINT "   HIT RETURN TO ACC
   EPT GUESS #";
200 Y=TRY*2 MOD 36+1:TRY=TRY+1:
    TAB 32: PRINT TRY;: COLOR=
   0: HLIN 0,39 AT Y:FLASH=1: FOR
   N=1 TO 5:A(N)=8: GOSUB 1000
   : NEXT N:N=1
300 FOR WAIT=1 TO 10:KEY= PEEK
   (-16384): IF KEY<132 THEN 310
   : POKE -16368,0:FLASH=1: FOR
   I=1 TO 8: IF KEY<> ASC(X$(I)
   ) THEN NEXT I: IF I=9 THEN
   310:A(N)=I:KEY=149
310 GOSUB 1000: IF KEY=141 THEN
   400: IF KEY=136 AND N>1 OR
   KEY=149 AND N<6 THEN N=N+KEY/
   5-28: NEXT WAIT:FLASH=1-FLASH:
    GOTO 300
400 COLOR=15:M=0: FOR I=1 TO 5:
   D(I)=C(I):J=I: GOSUB 2000: NEXT
   I: IF M=5 THEN 500: COLOR=5
   : FOR J=1 TO 5: FOR I=1 TO
   5: GOSUB 2000: NEXT I,J: GOTO
   200
500 PRINT : PRINT "
YOU GOT IT IN "
   ;TRY;" TRIES (";: IF TRY<7 THEN
   PRINT "EXCELLENT";: IF TRY>
   6 AND TRY<10 THEN PRINT "GOOD"
   ;
510 IF TRY>9 AND TRY<15 THEN PRINT
   "AVERAGE";: IF TRY>14 THEN
   PRINT "POOR";: PRINT ")": CALL
   -384: TAB 5: PRINT "HIT ANY KEY
   TO PLAY AGAIN": GOTO 100
1000 IF N=6 THEN RETURN : COLOR=
   X(A(N))*FLASH: HLIN N*4-2,N*
   4 AT Y: RETURN
2000 IF A(I)<>D(J) THEN RETURN :
   M=M+1: PLOT 21+M+M,Y: PRINT
   " ";:A(I)=0:D(J)=9: RETURN
3000 REM CALL -384 SETS INVERSE VID
3010 REM CALL -380 SETS NORMAL VID
3020 REM PEEK(-16384) IS KBD (ASCII)
     (IF > 127 THEN STROBE SET)
3030 REM POKE-16368 CLRS KBD STROBE
3040 REM CALL-936 CLEARS SCREEN AND
     TABS CURSOR TO UPPER LEFT.
3050 REM IN 310, KEY/5-28= -1 OR +1
     (ARROW KEY=136 OR 149 ASCII)
4000 REM STMTS 10-50 INTRO
4010 REM STMTS 100-110 NEW SETUP
4020 REM STMT 200 NEW GUESS
4030 REM STMTS 300-310 USER INPUT
4040 REM STMT 400 GUESS EVAL
4050 REM STMTS 500-510 WIN
4060 REM SUBR 1000 COLOR LINE
4070 REM SUBR 2000 MATCH TEST
```

PROGRAM DESCRIPTION

This program plots three Biorhythm functions: Physical (P), Emotional (E), and Mental (M) or intellectual. All three functions are plotted in the color graphics display mode.

Biorhythm theory states that aspects of the mind run in cycles. A brief description of the three cycles follows:

Physical

The Physical Biorhythm takes 23 days to complete and is an indirect indicator of the physical state of the individual. It covers physical well-being, basic bodily functions, strength, coordination, and resistance to disease.

Emotional

The Emotional Biorhythm takes 28 days to complete. It indirectly indicates the level of sensitivity, mental health, mood, and creativity.

Mental

The mental cycle takes 33 days to complete and indirectly indicates the level of alertness, logic and analytic functions of the individual, and mental receptivity.

Biorhythms

Biorhythms are thought to affect behavior. When they cross a "baseline" the functions change phase - become unstable - and this causes Critical Days. These days are, according to the theory, our weakest and most vulnerable times. Accidents, catching colds, and bodily harm may occur on physically critical days. Depression, quarrels, and frustration are most likely on emotionally critical days. Finally, slowness of the mind, resistance to new situations and unclear thinking are likely on mentally critical days.

REQUIREMENTS

This program fits into a 4K or greater system.
BASIC is the programming language used.

PROGRAM LISTING: BIORHYTHM

```
 5 POKE 2,173: POKE 3,48: POKE
   4,192: POKE 5,165: POKE 6,0
   : POKE 7,32: POKE 8,168: POKE
   9,252: POKE 10,165: POKE 11
   ,1: POKE 12,208: POKE 13,4
10 POKE 14,198: POKE 15,24: POKE
   16,240: POKE 17,5: POKE 18,
   198: POKE 19,1: POKE 20,76:
    POKE 21,2: POKE 22,0: POKE
   23,96 .
15 GOTO 85
20 TT=3: GOSUB 30: RETURN
25 PRINT "************************
   **************": RETURN
30 KK=8:TON=500: GOSUB 45: RETURN

35 KK=9:TON=250: GOSUB 45: RETURN

40 KK=8:TON=250: GOSUB 45:KK=9
   :TON=250: GOSUB 45: RETURN

45 POKE 1,TON MOD 256: POKE 24
   ,TON/256+1: POKE 0,KK: CALL
   2: RETURN
50 A=(19-(P*B(I)/100))*(P*100<
   C(I))+(P*100>C(I))*(P*100<=
   3*C(I))*((P*100-C(I))/100*B(
   I)/100)
55 A=A+(P*100>3*C(I))*(38-((P*
   100-3*C(I))/100*B(I)/100)):
   A=39*(A>39)+A*(A<40): RETURN

60 KK=8:TN=500: GOSUB 70:KK=9:
   TN=250: GOSUB 70: RETURN
65 KK=7:TN=10: GOSUB 70: RETURN
70 POKE 1,TN MOD 256: POKE 24,
   TN/256+1: POKE 0,KK: CALL 2
   : RETURN
75 GOSUB 60: INPUT "DATE (M,D,Y) "
   ,M,D,Y:Y=Y+(Y<100)*1900
80 A=Y-(M<3):N=Y MOD 58*365-Y/
   58*82+A/4-A/400+M*31-M/12-M/
   7-M/5-3*(M>2)+D: IF N<0 THEN
   N=N+21252: RETURN
85 DIM N$(10),B$(3),B(3),C(3),
   BV(3):B(1)=348:B(2)=286:B(3
   )=242:C(1)=575:C(2)=700:C(3
   )=825:BV(1)=23:BV(2)=28
90 BV(3)=33: TEXT : CALL -936:
    POKE 34,20: GOSUB 20: GOSUB
   25: GOSUB 20: PRINT : TAB 10
   : PRINT "APPLE II BIORHYTHM (4K)
    ": TAB 15: PRINT
95 GOSUB 25: TAB 5: PRINT "COPYRIGH
   T 1977 APPLE COMPUTER INC."
   : POKE 34,24: VTAB 24
100 GOSUB 60: INPUT "NAME ",N$:
    VTAB 22: PRINT N$: VTAB 24
   : PRINT "BIRTH ";: GOSUB 75
   : VTAB 22: TAB 21: PRINT "BIRTH
   DATE ";M;",";D;",";Y: VTAB
   24:N1=N: CALL -868
105 PRINT "FORECAST ";: GOSUB 75
   :N=N-N1: IF N<0 THEN N=N+21252
   : VTAB 23: TAB 10: PRINT "FORECA
   ST DATE ";M;",";D;",";Y: VTAB
   24: CALL -868
110 J=1: GR : POKE 34,23: FOR X=
   18 TO 20: COLOR=3: HLIN 0,31
    AT X: NEXT X: HLIN 1,3 AT
   3: HLIN 1,3 AT 37: VLIN 2,4
    AT 2: VTAB 21
115 FOR Y=1 TO 31 STEP 3: PRINT
   Y;: IF Y<10 THEN PRINT " ";
   : PRINT " ";: NEXT Y: PRINT
   "  P E M": VTAB 24
120 VTAB 23: PRINT "DAYS LIVED "
   ;N: FOR I=1 TO 3: COLOR=1*(
   I=1)+6*(I=2)+8*(I=3): VLIN
   0,39 AT 33+I+I: VTAB 24
125 FOR X=0 TO 31:P=(N MOD BV(I)
   +X) MOD BV(I): GOSUB 50: PLOT
   X,A: GOSUB 65: NEXT X: NEXT
   I
130 PRINT : INPUT "ANOTHER PLOT (Y/N
   ) ",B$: IF B$(1,1)="Y" THEN
   90: END
>
```

DRAGON MAZE PROGRAM

PROGRAM DESCRIPTION

DRAGON MAZE is a game that will test your skill and memory. A mazeis constructed on the video screen. You watch carefully as it is completed. After it is finished the maze is hidden as if the lights were turned out. The object of the game is to get out of the maze before the dragon eats you. A reddish-brown square indicates your position and a purple square represents the dragon's.* You move by hitting a letter on the keyboard; U for up, D for down, R for right, and L for left. As you advance so does the dragon. The scent of humans drives the dragon crazy; when he is enraged he breaks through walls to get at you. DRAGON MAZE is not a game for the weak at heart. Try it if you dare to attempt out-smarting the dragon.

REQUIREMENTS

8K or greater Apple II computer system.
BASIC is the programming language.

* Color tints may vary depending upon video monitor or television adjustments.

```
1 TEXT : CALL -936
2 PRINT "WELCOME TO THE DRAGON'S MAZE!"
3 PRINT "YOU MAY WATCH WHILE I BUILD A MAZE,"
4 PRINT "BUT WHEN IT'S COMPLETE, I'LL ERASE"
5 PRINT "THE PICTURE. THEN YOU'LL ONLY SEE THE WALLS AS YOU BUMP INTO THEM."
6 PRINT "TO MOVE, YOU HIT 'R' FOR RIGHT,"
7 PRINT "'L' FOR LEFT, 'U' FOR UP, AND"
8 PRINT "'D' FOR DOWN. DO NOT HIT RETURN!"
9 PRINT
10 PRINT "THE OBJECT IS FOR YOU (THE GREEN DOT"
11 PRINT "TO GET TO THE DOOR ON THE RIGHT SIDE"
12 PRINT "BEFORE THE DRAGON (THE RED DOT) EATS"
13 PRINT "YOU."
14 PRINT "BEWARE!!!!!!!!! SOMETIMES THE DRAGON"
15 PRINT "GETS REAL MAD, AND CLIMBS OVER A WALL."
16 PRINT "BUT MOST OF THE TIME, HE CAN'T GO OVER"
17 PRINT "AND HAS TO GO AROUND."
18 PRINT
19 PRINT "(HINT: YOU CAN OFTEN TELL WHERE A WALL"
20 PRINT "IS, EVEN BEFORE YOU CAN SEE IT, BY"
21 PRINT "THE FACT THAT THE DRAGON CAN'T GET"
22 PRINT "THROUGH IT!)"
23 PRINT
89 DIM A$(3)
90 PRINT "TYPE 'GO' TO BEGIN ";: INPUT A$
100 GR : COLOR=15
105 CALL -936: PRINT "DRAGON MAZE";: TAB (25): PRINT "GARY J. SHANNON"
110 FOR I=0 TO 39 STEP 3: VLIN 0,39 AT I: HLIN 0,39 AT I: NEXT I
120 COLOR=0
130 S=1000
1000 DIM M(169),T(169)
1001 FOR I=1 TO 169:T(I)=0: NEXT I
1010 FOR I=1 TO 169:M(I)=11: NEXT I
1030 X= RND (13)+1:Y= RND (13)+1:C=169
1035 IF C=1 THEN 1200
1040 R=0:D=0:L=0:U=0:K=X+13*(Y-1):M(K)=- ABS (M(K)):C=C-1
1050 IF X=13 THEN 1060:R=M(K+1)>0
1060 IF Y=13 THEN 1070:D=M(K+13)>0
1070 IF X=1 THEN 1080:L=M(K-1)>0
1080 IF Y=1 THEN 1090:U=M(K-13)>0
1090 Q=R+D+L+U
1100 IF (Q<3 AND RND (10)<2) OR Q=0 THEN 1170
1110 DR= RND (4)
1120 GOTO 1130+10*DR
1130 IF NOT R THEN 1110:M(K)=M(K)+1:X=X+1
1135 VLIN 3*Y-2,3*Y-1 AT 3*(X-1)
1136 GOTO 1035
1140 IF NOT D THEN 1110:M(K)=M(K)+10:Y=Y+1
1145 HLIN 3*X-2,3*X-1 AT 3*(Y-1)
1146 GOTO 1035
1150 IF NOT L THEN 1110:M(K-1)=M(K-1)-1:X=X-1
1155 VLIN 3*Y-2,3*Y-1 AT 3*X
1156 GOTO 1035
1160 IF NOT U THEN 1110:M(K-13)=M(K-13)-10:Y=Y-1
1165 HLIN 3*X-2,3*X-1 AT 3*Y: GOTO 1035
1170 X= RND (13)+1:Y= RND (13)+1
1180 IF M(X+13*(Y-1))>0 THEN 1170
1190 C=C+1: GOTO 1035
1200 GOSUB 5000: PRINT "THE MAZE IS READY"
1205 GR : COLOR=15
1210 VLIN 0,39 AT 0: VLIN 0,39 AT 39: HLIN 0,39 AT 0: HLIN 0,39 AT 39
1220 X=1:Y= RND (13)+1: COLOR=8: PLOT 3*X-2,3*Y-2
```

```
1225 HX=3*X-2:HY=3*Y-2
1230 WY= RND (13)+1
1240 COLOR=0: VLIN 3*WY-2,3*WY-1
     AT 39
1250 SX=13:SY=WY
1260 QX=3*SX-2:QY=3*SY-2
1270 RD=1
1500 K= PEEK (-16384): IF K<128 THEN
     1500
1510 POKE -16368,0
1515 QQ=K: GOSUB 7000:K=QQ
1516 IF SX=X AND SY=Y THEN 8000
1520 IF K= ASC("R") THEN 2000
1530 IF K= ASC("L") THEN 2500
1540 IF K= ASC("U") THEN 3000
1550 IF K= ASC("D") THEN 3500
1560 GOSUB 5000: GOTO 1500
2000 DX=1:DY=0
2010 IF M(X+13*(Y-1)) MOD 10 THEN
     4000
2020 FX=3*X-2:FY=3*Y-2: FOR I=1 TO
     3
2030 FX=FX+DX:FY=FY+DY
2040 COLOR=0
2060 FOR K=0 TO 1: FOR L=0 TO 1:
      PLOT HX+K,HY+L: NEXT L,K: COLOR=
     8: FOR K=0 TO 1: FOR L=0 TO
     1: PLOT FX+K,FY+L: NEXT L,K:
     HX=FX:HY=FY
2110 NEXT I
2115 X=X+DX:Y=Y+DY
2116 IF X=13 AND Y=WY THEN 6000
2120 GOTO 1500
2500 DX=-1:DY=0
2510 IF M(X+13*(Y-1)-1) MOD 10 THEN
     4100
2520 GOTO 2020
3000 DX=0:DY=-1
3010 IF M(X+13*(Y-2))/10 THEN 4200

3020 GOTO 2020
3500 DX=0:DY=1
3510 IF M(X+13*(Y-1))/10 THEN 4300

3520 GOTO 2020
4000 GOSUB 5000
4010 COLOR=15
4020 VLIN 3*(Y-1),3*Y AT 3*X
4030 GOTO 1500
4100 GOSUB 5000
4110 COLOR=15
4120 VLIN 3*(Y-1),3*Y AT 3*(X-1)

4130 GOTO 1500
4200 GOSUB 5000
4210 COLOR=15
4220 HLIN 3*(X-1),3*X AT 3*(Y-1)

4230 GOTO 1500
4300 GOSUB 5000
4310 COLOR=15
4320 HLIN 3*(X-1),3*X AT 3*Y
4330 GOTO 1500
5000 S=S-1: FOR I=1 TO 20:A= PEEK
     (-16336)+ PEEK (-16336)+ PEEK
     (-16336)+ PEEK (-16336): NEXT
     I: RETURN
6000 PRINT "YOU WIN!"
6010 GOSUB 5000: GOSUB 5000: GOSUB
     5000
6020 PRINT "SCORE=";S+3
6030 END
7000 IF X>SX THEN 7005: IF Y>SY THEN
     7050
7001 IF X<SX THEN 7100: IF Y<SY THEN
     7150
7005 IF SX=13 THEN 7050: IF T(SX+
     13*(SY-1))>9 THEN 7010: IF
     M(SX+13*(SY-1)) MOD 10 THEN
     7050
7010 DX=1:DY=0
7020 COLOR=0
7022 RX=3*SX-2:RY=3*SY-2
7023 FOR I=1 TO 3:RX=RX+DX:RY=RY+
     DY
7024 COLOR=0
7025 FOR K=0 TO 1: FOR L=0 TO 1:
      PLOT QX+K,QY+L: NEXT L,K: COLOR=
     RD: FOR K=0 TO 1: FOR L=0 TO
     1: PLOT RX+K,RY+L: NEXT L,K:
     QX=RX:QY=RY
7030 NEXT I
7035 SX=SX+DX:SY=SY+DY
7040 T(SX+13*(SY-1))=T(SX+13*(SY-
     1))+1
7045 RETURN
7050 IF SY=13 THEN 7100: IF T(SX+
     13*(SY-1))>9 THEN 7060: IF
     M(SX+13*(SY-1))/10 THEN 7100

7060 DX=0:DY=1: GOTO 7020
7100 IF SX=1 THEN 7150: IF T(SX+
     13*(SY-1))>9 THEN 7110: IF
     M(SX+13*(SY-1)-1) MOD 10 THEN
     7150
```

```
7110 DX=-1:DY=0: GOTO 7020
7150 IF SY=1 THEN 7005: IF T(SX+
     13*(SY-1))>9 THEN 7160: IF
     W(SX+13*(SY-1)-13)/10 THEN
     7005
7160 DX=0:DY=-1: GOTO 7020
8000 GOSUB 5000: GOSUB 5000: GOSUB
     5000: GOSUB 5000: PRINT "THE DRA
     GON GOT YOU!"
1999 END
```

APPLE II FIRMWARE

1. System Monitor Commands
2. Control and Editing Characters
3. Special Controls and Features
4. Annotated Monitor and Dis-assembler Listing
5. Binary Floating Point Package
6. Sweet 16 Interpreter Listing
7. 6502 Op Codes

System Monitor Commands

Apple II contains a powerful machine level monitor for use by the advanced programmer. To enter the monitor either press RESET button on keyboard or CALL-151 (Hex FF65) from Basic. Apple II will respond with an "*" (asterisk) prompt character on the TV display. This action will not kill current BASIC program which may be re-entered by a C^C (control C). NOTE: "adrs" is a four digit hexidecimal number and "data" is a two digit hexidecimal number. Remember to press "return" button at the end of each line.

Command Format	Example	Description
Examine Memory		
adrs	*CØF2	Examines (displays) single memory location of (adrs)
adrs1.adrs2	*1Ø24.1Ø48	Examines (displays) range of memory from (adrs1) thru (adrs2)
(return)	*(return)	Examines (displays) next 8 memory locations.
.adrs2	*.4Ø96	Examines (displays) memory from current location through location (adrs2)
Change Memory		
adrs:data data data	*A256:EF 2Ø 43	Deposits data into memory starting at location (adrs).
:data data data	*:FØ A2 12	Deposits data into memory starting after (adrs) last used for deposits.
Move Memory		
adrs1<adrs2. adrs3M	*1ØØ<BØ1Ø.B41ØM	Copy the data now in the memory range from (adrs2) to (adrs3) into memory locations starting at (adrs1).
Verify Memory		
adsr1<adrs2 adrs3V	*1ØØ<BØ1Ø.B41ØV	Verify that block of data in memory range from (adrs2) to (adrs3) exactly matches data block starting at memory location (adrs1)and displays differences if any.

Command Format	Example	Description
Cassette I/O		
adrs1.adrs2R	*3ØØ.4FFR	Reads cassette data into specified memory (adrs) range. Record length must be same as memory range or an error will occur.
adrs1.adrs2W	*8ØØ.9FFW	Writes onto cassette data from specified memory (adrs) range.
Display		
I	*I	Set inverse video mode. (Black characters on white background)
M	*N	Set normal video mode. (White characters on black background)
Dis-assembler		
adrsL	*C8ØØL	Decodes 2Ø instructions starting at memory (adrs) into 65Ø2 assembly nmenonic code.
L	*L	Decodes next 2Ø instructions starting at current memory address.
Mini-assembler		
(Turn-on)	*F666G	Turns-on mini-assembler. Prompt character is now a "!" (exclamation point).
$(monitor command)	!$C8ØØL	Executes any monitor command from mini-assembler then returns control to mini-assembler. Note that many monitor commands change current memory address reference so that it is good practice to retype desired address reference upon return to mini-assembler.
adrs:(65Ø2 MNEMONIC instruction)	!CØ1Ø:STA 23FF	Assembles a mnemonic 65Ø2 instruction into machine codes. If error, machine will refuse instruction, sound bell, and reprint line with up arrow under error.

Command Format	Example	Description
(space) (65Ø2 mnemonic instruction)	! STA Ø1FF	Assembles instruction into next available memory location. (Note space between "f" and instruction)
(TURN-OFF)	! (Reset Button)	Exits mini-assembler and returns to system monitor.

Monitor Program Execution and Debuging

adrsG	*3ØØG	Runs machine level program starting at memory (adrs).
adrsT	*8ØØT	Traces a program starting at memory location (adrs) and continues trace until hitting a breakpoint. Break occurs on instruction ØØ (BRK), and returns control to system monitor. Opens 65Ø2 status registers (see note 1)
asrdS	*CØ5ØS	Single steps through program beginning at memory location (adrs). Type a letter S for each additional step that you want displayed. Opens 65Ø2 status registers (see Note 1).
(Control E)	*E^C	Displays 65Ø2 status registers and opens them for modification (see Note 1)
(Control Y)	*Y^C	Executes user specified machine language subroutine starting at memory location (3F8).

Note 1:

65Ø2 status registers are open if they are last line displayed on screen. To change them type ":" then "data" for each register.

```
Example:  A = 3C  X = FF  Y = ØØ  P = 32  S = F2
          *: FF               Changes A register only
          *:FF ØØ 33          Changes A, X, and Y registers
```

To change S register, you must first retype data for A, X, Y and P.

Hexidecimal Arithmetic

data1+data2	*78+34	Performs hexidecimal sum of data1 plus data2.
data1-data2	*AE-34	Performs hexidecimal difference of data1 minus data2.

Command Format	Example	Description
Set Input/Output Ports		
(X) (Control P)	*5P^{C}	Sets printer output to I/O slot number (X). (see Note 2 below)
(X) (Control K)	*2K^{C}	Sets keyboard input to I/O slot number (X). (see Note 2 below)

Note 2:

Only slots 1 through 7 are addressable in this mode. Address Ø (Ex: ØP^{C} or ØK^{C}) resets ports to internal video display and keyboard. These commands will not work unless Apple II interfaces are plugged into specificed I/O slot.

Multiple Commands

Command Format	Example	Description
	*1ØØL 4ØØG AFFT	Multiple monitor commands may be given on same line if separated by a "space".
	*LLLL	Single letter commands may be repeated without spaces.

SPECIAL CONTROL AND EDITING CHARACTERS

"Control" characters are indicated by a super-scripted "C" such as G^C. They are obtained by holding down the CTRL key while typing the specified letter. Control characters are NOT displayed on the TV screen. B^C and C^C must be followed by a carriage return. Screen editing characters are indicated by a sub-scripted "E" such as D_E. They are obtained by pressing <u>and releasing</u> the ESC key then typing specified letter. Edit characters send information only to display screen and does not send data to memory. For example, U^C moves to cursor to right and copies text while A_E moves cursor to right but does not copy text.

CHARACTER	DESCRIPTION OF ACTION
RESET key	Immediately interrupts any program execution and resets computer. Also sets all text mode with scrolling window at maximum. Control is transferred to System Monitor and Apple prompts with a "*" (asterisk) and a bell. Hitting RESET key does NOT destroy existing BASIC or machine language program.
Control B	If in System Monitor (as indicated by a "*"), a control B and a carriage return will transfer control to BASIC, <u>scratching (killing) any existing BASIC program</u> and set HIMEM: to maximum installed user memory and LOMEM: to 2048.
Control C	If in BASIC, halts program and displays line number where stop occurred*. Program may be continued with a CON command. If in <u>System</u> Monitor, (as indicated by "*"), control C and a carriage return will enter BASIC <u>without</u> killing current program.
Control G	Sounds bell (beeps speaker)
Control H	Backspaces cursor and deletes any overwritten characters from computer but not from screen. Apply supplied keyboards have special key "4-." on right side of keyboard that provides this functions without using control button.
Control J	Issues line feed only
Control V	Compliment to H^C. Forward spaces cursor and copies over written characters. Apple keyboards have "+" key on right side which also performs this function.
Control X	Immediately deletes current line.

* If BASIC program is expecting keyboard input, you will have to hit carriage return key after typing control C.

SPECIAL CONTROL AND EDITING CHARACTERS

(continued)

CHARACTER	DESCRIPTION OF ACTION
A_E	Move cursor to right
B_E	Move cursor to left
C_E	Move cursor down
D_E	Move cursor up
E_E	Clear text from cursor to end of line
F_E	Clear text from cursor to end of page
$@_E$	Home cursor to top of page, clear text to end of page.

Special Controls and Features

Display Mode Controls

Hex	BASIC Example	Description
C050	10 POKE -16304,0	Set color graphics mode
C051	20 POKE -16303,0	Set text mode
C052	30 POKE -16302,0	Clear mixed graphics
C053	40 POKE -16301,0	Set mixed graphics (4 lines text)
C054	50 POKE -16300,0	Clear display Page 2 (BASIC commands use Page 1 only)
C055	60 POKE -16299,0	Set display to Page 2 (alternate)
C056	70 POKE -16298,0	Clear HIRES graphics mode
C057	80 POKE -16297,0	Set HIRES graphics mode

TEXT Mode Controls

Hex	BASIC Example	Description
0020	90 POKE 32,L1	Set left side of scrolling window to location specified by L1 in range of 0 to 39.
0021	100 POKE 33,W1	Set window width to amount specified by W1. L1+W1<40. W1>0
0022	110 POKE 34,T1	Set window top to line specified by T1 in range of 0 to 23
0023	120 POKE 35,B1	Set window bottom to line specified by B1 in the range of 0 to 23. B1>T1
0024	130 CH=PEEK(36) 140 POKE 36,CH 150 TAB(CH+1)	Read/set cusor horizontal position in the range of 0 to 39. If using TAB, you must add "1" to cusor position read value; Ex. 140 and 150 perform identical function.
0025	160 CV=PEEK(37) 170 POKE 37,CV 180 VTAB(CV+1)	Similar to above. Read/set cusor vertical position in the range 0 to 23.
0032	190 POKE 50,127 200 POKE 50,255	Set inverse flag if 127 (Ex. 190) Set normal flag if 255(Ex. 200)
FC58	210 CALL -936	($@_E$) Home cusor, clear screen
FC42	220 CALL -958	(F_E) Clear from cusor to end of page

Hex	BASIC Example	Description
FC9C	23Ø CALL -868	(E_E) Clear from cusor to end of line
FC66	24Ø CALL -922	(J^C) Line feed
FC7Ø	25Ø CALL -912	Scroll up text one line

Miscellaneous

CØ3Ø	36Ø X=PEEK(-16336) 365 POKE -16336,Ø	Toggle speaker
CØØØ	37Ø X=PEEK(-16384	Read keyboard; if X>127 then key was pressed.
CØ1Ø	38Ø POKE -16368,Ø	Clear keyboard strobe - always after reading keyboard.
CØ61	39Ø X=PEEK(16287)	Read PDL(Ø) push button switch. If X>127 then switch is "on".
CØ62	4ØØ X=PEEK(-16286)	Read PDL(1) push button switch.
CØ63	41Ø X=PEEK(-16285	Read PDL(2) push button switch.
CØ58	42Ø POKE -16296,Ø	Clear Game I/O ANØ output
CØ59	43Ø POKE -16295,Ø	Set Game I/O ANØ output
CØ5A	44Ø POKE -16294,Ø	Clear Game I/O AN1 output
CØ5B	45Ø POKE -16293,Ø	Set Game I/O AN1 output
CØ5C	46Ø POKE -16292,Ø	Clear Game I/O AN2 output
CØ5D	47Ø POKE -16291,Ø	Set Game I/O AN2 output
CØ5E	48Ø POKE -1629Ø,Ø	Clear Game I/O AN3 output
CØ5F	49Ø POKE -16289,Ø	Set Game I/O AN3 output

```
***************************
*                         *
*        APPLE II         *
*     SYSTEM MONITOR      *
*                         *
*    COPYRIGHT 1977 BY    *
*   APPLE COMPUTER, INC.  *
*                         *
*   ALL RIGHTS RESERVED   *
*                         *
*       S. WOZNIAK        *
*        A. BAUM          *
*                         *
***************************
        TITLE           "APPLE II SYSTEM MONITOR"
LOC0     EPZ   $00
LOC1     EPZ   $01
WNDLFT   EPZ   $20
WNDWDTH  EPZ   $21
WNDTOP   EPZ   $22
WNDBTM   EPZ   $23
CH       EPZ   $24
CV       EPZ   $25
GBASL    EPZ   $26
GBASH    EPZ   $27
BASL     EPZ   $28
BASH     EPZ   $29
BAS2L    EPZ   $2A
BAS2H    EPZ   $2B
H2       EPZ   $2C
LMNEM    EPZ   $2C
RTNL     EPZ   $2C
V2       EPZ   $2D
RMNEM    EPZ   $2D
RTNH     EPZ   $2D
MASK     EPZ   $2E
CHKSUM   EPZ   $2E
FORMAT   EPZ   $2E
LASTIN   EPZ   $2F
LENGTH   EPZ   $2F
SIGN     EPZ   $2F
COLOR    EPZ   $30
MODE     EPZ   $31
INVFLG   EPZ   $32
PROMPT   EPZ   $33
YSAV     EPZ   $34
YSAV1    EPZ   $35
CSWL     EPZ   $36
CSWH     EPZ   $37
KSWL     EPZ   $38
KSWH     EPZ   $39
PCL      EPZ   $3A
PCH      EPZ   $3B
XQT      EPZ   $3C
A1L      EPZ   $3C
A1H      EPZ   $3D
A2L      EPZ   $3E
A2H      EPZ   $3F
A3L      EPZ   $40
A3H      EPZ   $41
A4L      EPZ   $42
A4H      EPZ   $43
A5L      EPZ   $44
A5H      EPZ   $45
```

```
                   ACC      EQU  $45
                   XREG     EQU  $46
                   YREG     EQU  $47
                   STATUS   EQU  $48
                   SPNT     EQU  $49
                   RNDL     EQU  $4E
                   RNDH     EQU  $4F
                   ACL      EQU  $50
                   ACH      EQU  $51
                   XTNDL    EQU  $52
                   XTNDH    EQU  $53
                   AUXL     EQU  $54
                   AUXH     EQU  $55
                   PICK     EQU  $95
                   IN       EQU  $0200
                   USRADR   EQU  $03F8
                   NMI      EQU  $03FB
                   IRQLOC   EQU  $03FE
                   IOADR    EQU  $C000
                   KBD      EQU  $C000
                   KBDSTRB  EQU  $C010
                   TAPEOUT  EQU  $C020
                   SPKR     EQU  $C030
                   TXTCLR   EQU  $C050
                   TXTSET   EQU  $C051
                   MIXCLR   EQU  $C052
                   MIXSET   EQU  $C053
                   LOWSCR   EQU  $C054
                   HISCR    EQU  $C055
                   LORES    EQU  $C056
                   HIRES    EQU  $C057
                   TAPEIN   EQU  $C060
                   PADDL0   EQU  $C064
                   PTRIG    EQU  $C070
                   BASIC    EQU  $E000
                   BASIC2   EQU  $E003
                            ORG  $F800      ROM START ADDRESS
F800: 4A           PLOT     LSR             Y-COORD/2
F801: 08                    PHP             SAVE LSB IN CARRY
F802: 20 47 F8              JSR  GBASCALC   CALC BASE ADR IN GBASL,H
F805: 28                    PLP             RESTORE LSB FROM CARRY
F806: A9 0F                 LDA  #$0F       MASK $0F IF EVEN
F808: 90 02                 BCC  RTMASK
F80A: 69 E0                 ADC  #$E0       MASK $F0 IF ODD
F80C: 85 2E        RTMASK   STA  MASK
F80E: B1 26        PLOT1    LDA  (GBASL),Y DATA
F810: 45 30                 EOR  COLOR      EOR COLOR
F812: 25 2E                 AND  MASK        AND MASK
F814: 51 26                 EOR  (GBASL),Y    XOR DATA
F816: 91 26                 STA  (GBASL),Y     TO DATA
F818: 60                    RTS
F819: 20 00 F8     HLINE    JSR  PLOT       PLOT SQUARE
F81C: C4 2C        HLINE1   CPY  H2         DONE?
F81E: B0 11                 BCS  RTS1        YES, RETURN
F820: C8                    INY             NO, INCR INDEX (X-COORD)
F821: 20 0E F8              JSR  PLOT1      PLOT NEXT SQUARE
F824: 90 F6                 BCC  HLINE1     ALWAYS TAKEN
F826: 69 01        VLINEZ   ADC  #$01       NEXT Y-COORD
F828: 48           VLINE    PHA             SAVE ON STACK
F829: 20 00 F8              JSR  PLOT       PLOT SQUARE
F82C: 68                    PLA
F82D: C5 2D                 CMP  V2         DONE?
F82F: 90 F5                 BCC  VLINEZ      NO, LOOP
F831: 60           RTS1     RTS
F832: A0 2F        CLRSCR   LDY  #$2F       MAX Y, FULL SCRN CLR
F834: D0 02                 BNE  CLRSC2     ALWAYS TAKEN
F836: A0 27        CLRTOP   LDY  #$27       MAX Y, TOP SCREEN CLR
F838: 84 2D        CLRSC2   STY  V2         STORE AS BOTTOM COORD
                             FOR VLINE CALLS
F83A: A0 27                 LDY  #$27       RIGHTMOST X-COORD (COLUMN)
F83C: A9 00        CLRSC3   LDA  #$00       TOP COORD FOR VLINE CALLS
F83E: 85 30                 STA  COLOR      CLEAR COLOR (BLACK)
F840: 20 28 F8              JSR  VLINE      DRAW VLINE
F843: 88                    DEY             NEXT LEFTMOST X-COORD
F844: 10 F6                 BPL  CLRSC3     LOOP UNTIL DONE
F846: 60                    RTS
F847: 48           GBASCALC PHA             FOR INPUT 000DEFGH
F848: 4A                    LSR
F849: 29 03                 AND  #$03
F84B: 09 04                 ORA  #$04         GENERATE GBASH=000001FG
F84D: 85 27                 STA  GBASH
F84F: 68                    PLA             AND GBASL=HDEDE000
F850: 29 18                 AND  #$18
F852: 90 02                 BCC  GBCALC
F854: 69 7F                 ADC  #$7F
F856: 85 26        GBCALC   STA  GBASL
```

```
F858: 0A                ASL   A
F859: 0A                ASL   A
F85A: 05 26             ORA   GBASL
F85C: 85 26             STA   GBASL
F85E: 60                RTS
F85F: A5 30    NXTCOL   LDA   COLOR       INCREMENT COLOR BY 3
F861: 18                CLC
F862: 69 03             ADC   #$03
F864: 29 0F    SETCOL   AND   #$0F        SETS COLOR=17*A MOD 16
F866: 85 30             STA   COLOR
F868: 0A                ASL   A           BOTH HALF BYTES OF COLOR EQUAL
F869: 0A                ASL   A
F86A: 0A                ASL   A
F86B: 0A                ASL   A
F86C: 05 30             ORA   COLOR
F86E: 85 30             STA   COLOR
F870: 60                RTS
F871: 4A       SCRN     LSR   A           READ SCREEN Y-COORD/2
F872: 08                PHP               SAVE LSB (CARRY)
F873: 20 47 F8          JSR   GBASCALC    CALC BASE ADDRESS
F876: B1 26             LDA   (GBASL),Y   GET BYTE
F878: 28                PLP               RESTORE LSB FROM CARRY
F879: 90 04    SCRN2    BCC   RTMSKZ      IF EVEN, USE LO H
F87B: 4A                LSR   A
F87C: 4A                LSR   A
F87D: 4A                LSR   A           SHIFT HIGH HALF BYTE DOWN
F87E: 4A                LSR   A
F87F: 29 0F    RTMSKZ   AND   #$0F        MASK 4-BITS
F881: 60                RTS
F882: A6 3A    INSDS1   LDX   PCL         PRINT PCL,H
F884: A4 3B             LDY   PCH
F886: 20 96 FD          JSR   PRYX2
F889: 20 48 F9          JSR   PRBLNK      FOLLOWED BY A BLANK
F88C: A1 3A             LDA   (PCL,X)     GET OP CODE
F88E: A8       INSDS2   TAY
F88F: 4A                LSR   A           EVEN/ODD TEST
F890: 90 09             BCC   IEVEN
F892: 6A                ROR               BIT 1 TEST
F893: B0 10             BCS   ERR         XXXXXX11 INVALID OP
F895: C9 A2             CMP   #$A2
F897: F0 0C             BEQ   ERR         OPCODE $89 INVALID
F899: 29 87             AND   #$87        MASK BITS
F89B: 4A       IEVEN    LSR   A           LSB INTO CARRY FOR L/R TEST
F89C: AA                TAX
F89D: BD 62 F9          LDA   FMT1,X      GET FORMAT INDEX BYTE
F8A0: 20 79 F8          JSR   SCRN2       R/L H-BYTE ON CARRY
F8A3: D0 04             BNE   GETFMT
F8A5: A0 80    ERR      LDY   #$80        SUBSTITUTE $80 FOR INVALID OPS
F8A7: A9 00             LDA   #$00        SET PRINT FORMAT INDEX TO 0
F8A9: AA       GETFMT   TAX
F8AA: BD A6 F9          LDA   FMT2,X      INDEX INTO PRINT FORMAT TABLE
F8AD: 85 2E             STA   FORMAT      SAVE FOR ADR FIELD FORMATTING
F8AF: 29 03             AND   #$03        MASK FOR 2-BIT LENGTH
                             (P=1 BYTE, 1=2 BYTE, 2=3 BYTE)
F8B1: 85 2F             STA   LENGTH
F8B3: 98                TYA               OPCODE
F8B4: 29 8F             AND   #$8F        MASK FOR 1XXX1010 TEST
F8B6: AA                TAX               SAVE IT
F8B7: 98                TYA               OPCODE TO A AGAIN
F8B8: A0 03             LDY   #$03
F8BA: E0 8A             CPX   #$8A
F8BC: F0 0B             BEQ   MNNDX3
F8BE: 4A       MNNDX1   LSR   A
F8BF: 90 08             BCC   MNNDX3      FORM INDEX INTO MNEMONIC TABLE
F8C1: 4A                LSR   A
F8C2: 4A       MNNDX2   LSR   A           1) 1XXX1010-&gt00101XXX
F8C3: 09 20             ORA   #$20           2) XXXYYY01-&gt00111XXX
F8C5: 88                DEY               3) XXXYYY10-&gt00110XXX
F8C6: D0 FA             BNE   MNNDX2         4) XXXYY100-&gt00100XXX
F8C8: C8                INY               5) XXXXX000-&gt000XXXXX
F8C9: 88       MNNDX3   DEY
F8CA: D0 F2             BNE   MNNDX1
F8CC: 60                RTS
F8CD: FF FF FF          DFB   $FF,$FF,$FF
F8D0: 20 82 F8 INSTDSP  JSR   INSDS1      GEN FMT, LEN BYTES
F8D3: 48                PHA               SAVE MNEMONIC TABLE INDEX
F8D4: B1 3A    PRNTOP   LDA   (PCL),Y
F8D6: 20 DA FD          JSR   PRBYTE
F8D9: A2 01             LDX   #$01        PRINT 2 BLANKS
F8DB: 20 4A F9 PRNTBL   JSR   PRBL2
F8DE: C4 2F             CPY   LENGTH      PRINT INST (1-3 BYTES)
F8E0: C8                INY               IN A 12 CHR FIELD
F8E1: 90 F1             BCC   PRNTOP
F8E3: A2 03             LDX   #$03        CHAR COUNT FOR MNEMONIC PRINT
F8E5: C0 04             CPY   #$04
```

```
F8E7: 90 F2             BCC   PRNTBL
F8E9: 68                PLA               RECOVER MNEMONIC INDEX
F8EA: A8                TAY
F8EB: B9 C0 F9          LDA   MNEML,Y
F8EE: 85 2C             STA   LMNEM       FETCH 3-CHAR MNEMONIC
F8F0: B9 00 FA          LDA   MNEMR,Y       (PACKED IN 2-BYTES)
F8F3: 85 2D             STA   RMNEM
F8F5: A9 00    PRMN1    LDA   #$00
F8F7: A0 05             LDY   #$05
F8F9: 06 2D    PRMN2    ASL   RMNEM       SHIFT 5 BITS OF
F8FB: 26 2C             ROL   LMNEM         CHARACTER INTO A
F8FD: 2A                ROL                   (CLEARS CARRY)
F8FE: 88                DEY
F8FF: D0 F8             BNE   PRMN2
F901: 69 BF             ADC   #$BF        ADD "?" OFFSET
F903: 20 ED FD          JSR   COUT        OUTPUT A CHAR OF MNEM
F906: CA                DEX
F907: D0 EC             BNE   PRMN1
F909: 20 48 F9          JSR   PRBLNK      OUTPUT 3 BLANKS
F90C: A4 2F             LDY   LENGTH
F90E: A2 06             LDX   #$06        CNT FOR 6 FORMAT BITS
F910: E0 03    PRADR1   CPX   #$03
F912: F0 1C             BEQ   PRADR5      IF X=3 THEN ADDR.
F914: 06 2E    PRADR2   ASL   FORMAT
F916: 90 0E             BCC   PRADR3
F918: BD B3 F9          LDA   CHAR1-1,X
F91B: 20 ED FD          JSR   COUT
F91E: BD B9 F9          LDA   CHAR2-1,X
F921: F0 03             BEQ   PRADR3
F923: 20 ED FD          JSR   COUT
F926: CA       PRADR3   DEX
F927: D0 E7             BNE   PRADR1
F929: 60                RTS
F92A: 88       PRADR4   DEY
F92B: 30 E7             BMI   PRADR2
F92D: 20 DA FD          JSR   PRBYTE
F930: A5 2E    PRADR5   LDA   FORMAT
F932: C9 E8             CMP   #$E8        HANDLE REL ADR MODE
F934: B1 3A             LDA   (PCL),Y     SPECIAL (PRINT TARGET,
F936: 90 F2             BCC   PRADR4        NOT OFFSET)
F938: 20 56 F9 RELADR   JSR   PCADJ3
F93B: AA                TAX               PCL,PCH+OFFSET+1 TO A,Y
F93C: E8                INX
F93D: D0 01             BNE   PRNTYX      +1 TO Y,X
F93F: C8                INY
F940: 98       PRNTYX   TYA
F941: 20 DA FD PRNTAX   JSR   PRBYTE      OUTPUT TARGET ADR
F944: 8A       PRNTX    TXA                 OF BRANCH AND RETURN
F945: 4C DA FD          JMP   PRBYTE
F948: A2 03    PRBLNK   LDX   #$03        BLANK COUNT
F94A: A9 A0    PRBL2    LDA   #$A0        LOAD A SPACE
F94C: 20 ED FD PRBL3    JSR   COUT        OUTPUT A BLANK
F94F: CA                DEX
F950: D0 F8             BNE   PRBL2       LOOP UNTIL COUNT=0
F952: 60                RTS
F953: 38       PCADJ    SEC               0=1-BYTE, 1=2-BYTE
F954: A5 2F    PCADJ2   LDA   LENGTH        2=3-BYTE
F956: A4 3B    PCADJ3   LDY   PCH
F958: AA                TAX               TEST DISPLACEMENT SIGN
F959: 10 01             BPL   PCADJ4        (FOR REL BRANCH)
F95B: 88                DEY               EXTEND NEG BY DEC PCH
F95C: 65 3A    PCADJ4   ADC   PCL
F95E: 90 01             BCC   RTS2        PCL+LENGTH(OR DISPL)+1 TO A
F960: C8                INY                 CARRY INTO Y (PCH)
F961: 60       RTS2     RTS
               *         FMT1 BYTES:               XXXXXXY0 INSTRS
               *         IF Y=0                    THEN LEFT HALF BYTE
               *         IF Y=1                    THEN RIGHT HALF BYTE
               *                                        (X=INDEX)
F962: 04 20 54
F965: 30 0D    FMT1     DFB   $04,$20,$54,$30,$0D
F967: 80 04 90
F96A: 03 22             DFB   $80,$04,$90,$03,$22
F96C: 54 33 0D
F96F: 80 04             DFB   $54,$33,$0D,$80,$04
F971: 90 04 20
F974: 54 33             DFB   $90,$04,$20,$54,$33
F976: 0D 80 04
F979: 90 04             DFB   $0D,$80,$04,$90,$04
F97B: 20 54 3B
F97E: 0D 80             DFB   $20,$54,$3B,$0D,$80
F980: 04 90 00
F983: 22 44             DFB   $04,$90,$00,$22,$44
F985: 33 0D C8
F988: 44 00             DFB   $33,$0D,$C8,$44,$00
```

```
F98A: 11 22 44
F98D: 33 0D              DFB   $11,$22,$44,$33,$0D
F98F: C8 44 A9
F992: 01 22              DFB   $C8,$44,$A9,$01,$22
F994: 44 33 0D
F997: 80 04              DFB   $44,$33,$0D,$80,$04
F999: 90 01 22
F99C: 44 33              DFB   $90,$01,$22,$44,$33
F99E: 0D 80 04
F9A1: 90                 DFB   $0D,$80,$04,$90
F9A2: 26 31 87
F9A5: 9A                 DFB   $26,$31,$87,$9A $ZZXXXY01 INSTR'S
F9A6: 00        FMT2     DFB   $00      ERR
F9A7: 21                 DFB   $21      IMM
F9A8: 81                 DFB   $81      Z-PAGE
F9A9: 82                 DFB   $82      ABS
F9AA: 00                 DFB   $00      IMPLIED
F9AB: 00                 DFB   $00      ACCUMULATOR
F9AC: 59                 DFB   $59      (ZPAG,X)
F9AD: 4D                 DFB   $4D      (ZPAG),Y
F9AE: 91                 DFB   $91      ZPAG,X
F9AF: 92                 DFB   $92      ABS,X
F9B0: 86                 DFB   $86      ABS,Y
F9B1: 4A                 DFB   $4A      (ABS)
F9B2: 85                 DFB   $85      ZPAG,Y
F9B3: 9D                 DFB   $9D      RELATIVE
F9B4: AC A9 AC
F9B7: A3 A8 A4
                CHAR1    ASC   ",),#($"
F9BA: D9 00 D8
F9BD: A4 A4 00  CHAR2    DFB   $D9,$00,$D8,$A4,$A4,$00
                *CHAR2:  "Y",0,"X$$",0
                *         MNEML         IS OF FORM:
                *         (A) XXXXX000
                *         (B) XXXYY100
                *         (C) 1XXX1010
                *         (D) XXXYYY10
                *         (E) XXXYYY01
                *             (X=INDEX)
F9C0: 1C 8A 1C
F9C3: 23 5D 8B  MNEML    DFB   $1C,$8A,$1C,$23,$5D,$
F9C6: 1B A1 9D
F9C9: 8A 1D 23           DFB   $1B,$A1,$9D,$8A,$1D,$23
F9CC: 9D 8B 1D
F9CF: A1 00 29           DFB   $9D,$8B,$1D,$A1,$00,$29
F9D2: 19 AE 69
F9D5: A8 19 23           DFB   $19,$AE,$69,$A8,$19,$23
F9D8: 24 53 1B
F9DB: 23 24 53           DFB   $24,$53,$1B,$23,$24,$53
F9DE: 19 A1              DFB   $19,$A1    (A) FORMAT ABOVE
F9E0: 00 1A 5B
F9E3: 5B A5 69           DFB   $00,$1A,$5B,$5B,$A5,$69
F9E6: 24 24              DFB   $24,$24    (B) FORMAT
F9E8: AE AE A8
F9EB: AD 29 00           DFB   $AE,$AE,$A8,$AD,$29,$00
F9EE: 7C 00              DFB   $7C,$00    (C) FORMAT
F9F0: 15 9C 6D
F9F3: 9C A5 69           DFB   $15,$9C,$6D,$9C,$A5,$69
F9F6: 29 53              DFB   $29,$53    (D) FORMAT
F9F8: 84 13 34
F9FB: 11 A5 69           DFB   $84,$13,$34,$11,$A5,$69
F9FE: 23 A0              DFB   $23,$A0    (E) FORMAT
FA00: D8 62 5A
FA03: 48 26 62  MNEMR    DFB   $D8,$62,$5A,$48,$26,$62
FA06: 94 88 54
FA09: 44 C8 54           DFB   $94,$88,$54,$44,$C8,$54
FA0C: 68 44 E8
FA0F: 94 00 B4           DFB   $68,$44,$E8,$94,$00,$B4
FA12: 08 84 74
FA15: B4 28 6E           DFB   $08,$84,$74,$B4,$28,$6E
FA18: 74 F4 CC
FA1B: 4A 72 F2           DFB   $74,$F4,$CC,$4A,$72,$F2
FA1E: A4 8A              DFB   $A4,$8A    (A) FORMAT
FA20: 00 AA A2
FA23: A2 74 74           DFB   $00,$AA,$A2,$A2,$74,$74
FA26: 74 72              DFB   $74,$72    (B) FORMAT
FA28: 44 68 B2
FA2B: 32 B2 00           DFB   $44,$68,$B2,$32,$B2,$00
FA2E: 22 00              DFB   $22,$00    (C) FORMAT
FA30: 1A 1A 26
FA33: 26 72 72           DFB   $1A,$1A,$26,$26,$72,$72
FA36: 88 C8              DFB   $88,$C8    (D) FORMAT
FA38: C4 CA 26
FA3B: 48 44 44           DFB   $C4,$CA,$26,$48,$44,$44
FA3E: A2 C8              DFB   $A2,$C8    (E) FORMAT
```

```
FA40: FF FF FF          DFB   $FF,$FF,$FF
FA43: 20 D0 F8  STEP    JSR   INSTDSP    DISASSEMBLE ONE INST
FA46: 68                PLA              AT (PCL,H)
FA47: 85 2C             STA   RTNL       ADJUST TO USER
FA49: 68                PLA              STACK. SAVE
FA4A: 85 2D             STA   RTNH         RTN ADR.
FA4C: A2 08             LDX   #$08
FA4E: BD 10 FB  XQINIT  LDA   INITBL-1,X INIT XEQ AREA
FA51: 95 3C             STA   XQT,X
FA53: CA                DEX
FA54: D0 F8             BNE   XQINIT
FA56: A1 3A             LDA   (PCL,X)    USER OPCODE BYTE
FA58: F0 42             BEQ   XBRK       SPECIAL IF BREAK
FA5A: A4 2F             LDY   LENGTH     LEN FROM DISASSEMBLY
FA5C: C9 20             CMP   #$20
FA5E: F0 59             BEQ   XJSR       HANDLE JSR, RTS, JMP,
FA60: C9 60             CMP   #$60         JMP (), RTI SPECIAL
FA62: F0 45             BEQ   XRTS
FA64: C9 4C             CMP   #$4C
FA66: F0 5C             BEQ   XJMP
FA68: C9 6C             CMP   #$6C
FA6A: F0 59             BEQ   XJMPAT
FA6C: C9 40             CMP   #$40
FA6E: F0 35             BEQ   XRTI
FA70: 29 1F             AND   #$1F
FA72: 49 14             EOR   #$14
FA74: C9 04             CMP   #$04       COPY USER INST TO XEQ AREA
FA76: F0 02             BEQ   XQ2          WITH TRAILING NOPS
FA78: B1 3A     XQ1     LDA   (PCL),Y    CHANGE REL BRANCH
FA7A: 99 3C 00  XQ2     STA   XQT,Y        DISP TO 4 FOR
FA7D: 88                DEY                JMP TO BRANCH OR
FA7E: 10 F8             BPL   XQ1          NBRANCH FROM XEQ.
FA80: 20 3F FF          JSR   RESTORE    RESTORE USER REG CONTENTS.
FA83: 4C 3C 00          JMP   XQT        XEQ USER OP FROM RAM
FA86: 85 45     IRQ     STA   ACC          (RETURN TO NBRANCH)
FA88: 68                PLA
FA89: 48                PHA              **IRQ HANDLER
FA8A: 0A                ASL   A
FA8B: 0A                ASL   A
FA8C: 0A                ASL   A
FA8D: 30 03             BMI   BREAK      TEST FOR BREAK
FA8F: 6C FE 03          JMP   (IRQLOC)   USER ROUTINE VECTOR IN RAM
FA92: 28        BREAK   PLP
FA93: 20 4C FF          JSR   SAV1       SAVE REG'S ON BREAK
FA96: 68                PLA              INCLUDING PC
FA97: 85 3A             STA   PCL
FA99: 68                PLA
FA9A: 85 3B             STA   PCH
FA9C: 20 82 F8  XBRK    JSR   INSDS1     PRINT USER PC.
FA9F: 20 DA FA          JSR   RGDSP1       AND REG'S
FAA2: 4C 65 FF          JMP   MON        GO TO MONITOR
FAA5: 18        XRTI    CLC
FAA6: 68                PLA              SIMULATE RTI BY EXPECTING
FAA7: 85 48             STA   STATUS       STATUS FROM STACK, THEN RTS
FAA9: 68        XRTS    PLA              RTS SIMULATION
FAAA: 85 3A             STA   PCL          EXTRACT PC FROM STACK
FAAC: 68                PLA              AND UPDATE PC BY 1 (LEN=0)
FAAD: 85 3B     PCINC2  STA   PCH
FAAF: A5 2F     PCINC3  LDA   LENGTH     UPDATE PC BY LEN
FAB1: 20 56 F9          JSR   PCADJ3
FAB4: 84 3B             STY   PCH
FAB6: 18                CLC
FAB7: 90 14             BCC   NEWPCL
FAB9: 18        XJSR    CLC
FABA: 20 54 F9          JSR   PCADJ2     UPDATE PC AND PUSH
FABD: AA                TAX              ONTO STACH FOR
FABE: 98                TYA              JSR SIMULATE
FABF: 48                PHA
FAC0: 8A                TXA
FAC1: 48                PHA
FAC2: A0 02             LDY   #$02
FAC4: 18        XJMP    CLC
FAC5: B1 3A     XJMPAT  LDA   (PCL),Y
FAC7: AA                TAX              LOAD PC FOR JMP,
FAC8: 88                DEY              (JMP) SIMULATE.
FAC9: B1 3A             LDA   (PCL),Y
FACB: 86 3B             STX   PCH
FACD: 85 3A     NEWPCL  STA   PCL
FACF: B0 F3             BCS   XJMP
FAD1: A5 2D     RTNJMP  LDA   RTNH
FAD3: 48                PHA
FAD4: A5 2C             LDA   RTNL
FAD6: 48                PHA
FAD7: 20 8E FD  REGDSP  JSR   CROUT      DISPLAY USER REG
FADA: A9 45     RGDSP1  LDA   #ACC         CONTENTS WITH
FADC: 85 40             STA   A3L          LABELS
```

```
FADE: A9 00             LDA   #ACC/256
FAE0: 85 41             STA   A3H
FAE2: A2 FB             LDX   #$FB
FAE4: A9 A0     RDSP1   LDA   #$A0
FAE6: 20 ED FD          JSR   COUT
FAE9: BD 1E FA          LDA   RTBL-$FB,X
FAEC: 20 ED FD          JSR   COUT
FAEF: A9 BD             LDA   #$BD
FAF1: 20 ED FD          JSR   COUT
FAF4: B5 4A             LDA   ACC+5,X
FAF6: 20 DA FD          JSR   PRBYTE
FAF9: E8                INX
FAFA: 30 E8             BMI   RDSP1
FAFC: 60                RTS
FAFD: 18        BRANCH  CLC               BRANCH TAKEN,
FAFE: A0 01             LDY   #$01          ADD LEN+2 TO PC
FB00: B1 3A             LDA   (PCL),Y
FB02: 20 56 F9          JSR   PCADJ3
FB05: 85 3A             STA   PCL
FB07: 98                TYA
FB08: 38                SEC
FB09: B0 A2             BCS   PCINC2
FB0B: 20 4A FF  NBRNCH  JSR   SAVE        NORMAL RETURN AFTER
FB0E: 38                SEC                 XEQ USER OF
FB0F: B0 9E             BCS   PCINC3      GO UPDATE PC
FB11: EA        INITBL  NOP
FB12: EA                NOP               DUMMY FILL FOR
FB13: 4C 0B FB          JMP   NBRNCH        XEQ AREA
FB16: 4C FD FA          JMP   BRANCH
FB19: C1        RTBL    DFB   $C1
FB1A: D8                DFB   $D8
FB1B: D9                DFB   $D9
FB1C: D0                DFB   $D0
FB1D: D3                DFB   $D3
FB1E: AD 70 C0  PREAD   LDA   PTRIG       TRIGGER PADDLES
FB21: A0 00             LDY   #$00        INIT COUNT
FB23: EA                NOP               COMPENSATE FOR 1ST COUNT
FB24: EA                NOP
FB25: BD 64 C0  PREAD2  LDA   PADDL0,X    COUNT Y-REG EVERY
FB28: 10 04             BPL   RTS2D         12 USEC
FB2A: C8                INY
FB2B: D0 F8             BNE   PREAD2        EXIT AT 255 MAX
FB2D: 88                DEY
FB2E: 60        RTS2D   RTS
FB2F: A9 00     INIT    LDA   #$00        CLR STATUS FOR DEBUG
FB31: 85 48             STA   STATUS        SOFTWARE
FB33: AD 56 C0          LDA   LORES
FB36: AD 54 C0          LDA   LOWSCR      INIT VIDEO MODE
FB39: AD 51 C0  SETTXT  LDA   TXTSET      SET FOR TEXT MODE
FB3C: A9 00             LDA   #$00          FULL SCREEN WINDOW
FB3E: F0 0B             BEQ   SETWND
FB40: AD 50 C0  SETGR   LDA   TXTCLR      SET FOR GRAPHICS MODE
FB43: AD 53 C0          LDA   MIXSET        LOWER 4 LINES AS
FB46: 20 36 F8          JSR   CLRTOP        TEXT WINDOW
FB49: A9 14             LDA   #$14
FB4B: 85 22     SETWND  STA   WNDTOP      SET FOR 40 COL WINDOW
FB4D: A9 00             LDA   #$00          TOP IN A-REG,
FB4F: 85 20             STA   WNDLFT        BTTM AT LINE 24
FB51: A9 28             LDA   #$28
FB53: 85 21             STA   WNDWDTH
FB55: A9 18             LDA   #$18
FB57: 85 23             STA   WNDBTM        VTAB TO ROW 23
FB59: A9 17             LDA   #$17
FB5B: 85 25     TABV    STA   CV          VTABS TO ROW IN A-REG
FB5D: 4C 22 FC          JMP   VTAB
FB60: 20 A4 FB  MULPM   JSR   MD1         ABS VAL OF AC AUX
FB63: A0 10     MUL     LDY   #$10        INDEX FOR 16 BITS
FB65: A5 50     MUL2    LDA   ACL         ACX * AUX + XTND
FB67: 4A                LSR   A            TO AC, XTND
FB68: 90 0C             BCC   MUL4        IF NO CARRY,
FB6A: 18                CLC                NO PARTIAL PROD.
FB6B: A2 FE             LDX   #$FE
FB6D: B5 54     MUL3    LDA   XTNDL+2,X   ADD MPLCND (AUX)
FB6F: 75 56             ADC   AUXL+2,X     TO PARTIAL PROD
FB71: 95 54             STA   XTNDL+2,X    (XTND)
FB73: E8                INX
FB74: D0 F7             BNE   MUL3
FB76: A2 03     MUL4    LDX   #$03
FB78: 76        MUL5    DFB   $76
FB79: 50                DFB   $50
FB7A: CA                DEX
FB7B: 10 FB             BPL   MUL5
FB7D: 88                DEY
FB7E: D0 E5             BNE   MUL2
FB80: 60                RTS
```

```
FB81: 20 A4 FB  DIVPM    JSR   MD1        ABS VAL OF AC, AUX.
FB84: A0 10     DIV      LDY   #$10       INDEX FOR 16 BITS
FB86: 06 50     DIV2     ASL   ACL
FB88: 26 51              ROL   ACH
FB8A: 26 52              ROL   XTNDL      XTND/AUX
FB8C: 26 53              ROL   XTNDH        TO AC.
FB8E: 38                 SEC
FB8F: A5 52              LDA   XTNDL
FB91: E5 54              SBC   AUXL       MOD TO XTND.
FB93: AA                 TAX
FB94: A5 53              LDA   XTNDH
FB96: E5 55              SBC   AUXH
FB98: 90 06              BCC   DIV3
FB9A: 86 52              STX   XTNDL
FB9C: 85 53              STA   XTNDH
FB9E: E6 50              INC   ACL
FBA0: 88        DIV3     DEY
FBA1: D0 E3              BNE   DIV2
FBA3: 60                 RTS
FBA4: A0 00     MD1      LDY   #$00       ABS VAL OF AC, AUX
FBA6: 84 2F              STY   SIGN         WITH RESULT SIGN
FBA8: A2 54              LDX   #AUXL        IN LSB OF SIGN.
FBAA: 20 AF FB           JSR   MD3
FBAD: A2 50              LDX   #ACL
FBAF: B5 01     MD3      LDA   LOC1,X     X SPECIFIES AC OR AUX
FBB1: 10 0D              BPL   MDRTS
FBB3: 38                 SEC
FBB4: 98                 TYA
FBB5: F5 00              SBC   LOC0,X     COMPL SPECIFIED REG
FBB7: 95 00              STA   LOC0,X       IF NEG.
FBB9: 98                 TYA
FBBA: F5 01              SBC   LOC1,X
FBBC: 95 01              STA   LOC1,X
FBBE: E6 2F              INC   SIGN
FBC0: 60        MDRTS    RTS
FBC1: 48        BASCALC  PHA              CALC BASE ADR IN BASL,H
FBC2: 4A                 LSR   A            FOR GIVEN LINE NO
FBC3: 29 03              AND   #$03         0&lt=LINE NO.&lt=$17
FBC5: 09 04              ORA   #$04       ARG=000ABCDE, GENERATE
FBC7: 85 29              STA   BASH         BASH=000001CD
FBC9: 68                 PLA              AND
FBCA: 29 18              AND   #$18         BASL=EABAB000
FBCC: 90 02              BCC   BSCLC2
FBCE: 69 7F              ADC   #$7F
FBD0: 85 28     BSCLC2   STA   BASL
FBD2: 0A                 ASL
FBD3: 0A                 ASL
FBD4: 05 28              ORA   BASL
FBD6: 85 28              STA   BASL
FBD8: 60                 RTS
FBD9: C9 87     BELL1    CMP   #$87       BELL CHAR? (CNTRL-G)
FBDB: D0 12              BNE   RTS2B        NO, RETURN
FBDD: A9 40              LDA   #$40       DELAY .01 SECONDS
FBDF: 20 A8 FC           JSR   WAIT
FBE2: A0 C0              LDY   #$C0
FBE4: A9 0C     BELL2    LDA   #$0C       TOGGLE SPEAKER AT
FBE6: 20 A8 FC           JSR   WAIT         1 KHZ FOR .1 SEC.
FBE9: AD 30 C0           LDA   SPKR
FBEC: 88                 DEY
FBED: D0 F5              BNE   BELL2
FBEF: 60        RTS2B    RTS
FBF0: A4 24     STOADV   LDY   CH         CURSOR H INDEX TO Y-REG
FBF2: 91 28              STA   (BASL),Y   STORE CHAR IN LINE
FBF4: E6 24     ADVANCE  INC   CH         INCREMENT CURSOR H INDEX
FBF6: A5 24              LDA   CH           (MOVE RIGHT)
FBF8: C5 21              CMP   WNDWDTH    BEYOND WINDOW WIDTH?
FBFA: B0 66              BCS   CR           YES CR TO NEXT LINE
FBFC: 60        RTS3     RTS                NO,RETURN
FBFD: C9 A0     VIDOUT   CMP   #$A0       CONTROL CHAR?
FBFF: B0 EF              BCS   STOADV       NO,OUTPUT IT.
FC01: A8                 TAY              INVERSE VIDEO?
FC02: 10 EC              BPL   STOADV       YES, OUTPUT IT.
FC04: C9 8D              CMP   #$8D       CR?
FC06: F0 5A              BEQ   CR           YES.
FC08: C9 8A              CMP   #$8A       LINE FEED?
FC0A: F0 5A              BEQ   LF           IF SO, DO IT.
FC0C: C9 88              CMP   #$88       BACK SPACE? (CNTRL-H)
FC0E: D0 C9              BNE   BELL1        NO, CHECK FOR BELL.
FC10: C6 24     BS       DEC   CH         DECREMENT CURSOR H INDEX
FC12: 10 E8              BPL   RTS3       IF POS, OK. ELSE MOVE UP
FC14: A5 21              LDA   WNDWDTH    SET CH TO WNDWDTH-1
FC16: 85 24              STA   CH
FC18: C6 24              DEC   CH         (RIGHTMOST SCREEN POS)
FC1A: A5 22     UP       LDA   WNDTOP     CURSOR V INDEX
FC1C: C5 25              CMP   CV
```

```
FC1E: B0 0B              BCS   RTS4       IF TOP LINE THEN RETURN
FC20: C6 25              DEC   CV         DEC CURSOR V-INDEX
FC22: A5 25     VTAB     LDA   CV         GET CURSOR V-INDEX
FC24: 20 C1 FB  VTABZ    JSR   BASCALC    GENERATE BASE ADR
FC27: 65 20              ADC   WNDLFT     ADD WINDOW LEFT INDEX
FC29: 85 28              STA   BASL       TO BASL
FC2B: 60        RTS4     RTS
FC2C: 49 C0     ESC1     EOR   #$C0       ESC?
FC2E: F0 28              BEQ   HOME         IF SO, DO HOME AND CLEAR
FC30: 69 FD              ADC   #$FD       ESC-A OR B CHECK
FC32: 90 C0              BCC   ADVANCE      A, ADVANCE
FC34: F0 DA              BEQ   BS           B, BACKSPACE
FC36: 69 FD              ADC   #$FD       ESC-C OR D CHECK
FC38: 90 2C              BCC   LF           C, DOWN
FC3A: F0 DE              BEQ   UP           D, GO UP
FC3C: 69 FD              ADC   #$FD       ESC-E OR F CHECK
FC3E: 90 5C              BCC   CLREOL       E, CLEAR TO END OF LINE
FC40: D0 E9              BNE   RTS4         NOT F, RETURN
FC42: A4 24     CLREOP   LDY   CH         CURSOR H TO Y INDEX
FC44: A5 25              LDA   CV         CURSOR V TO A-REGISTER
FC46: 48        CLEOP1   PHA              SAVE CURRENT LINE ON STK
FC47: 20 24 FC           JSR   VTABZ      CALC BASE ADDRESS
FC4A: 20 9E FC           JSR   CLEOLZ     CLEAR TO EOL, SET CARRY
FC4D: A0 00              LDY   #$00       CLEAR FROM H INDEX=0 FOR REST
FC4F: 68                 PLA              INCREMENT CURRENT LINE
FC50: 69 00              ADC   #$00       (CARRY IS SET)
FC52: C5 23              CMP   WNDBTM     DONE TO BOTTOM OF WINDOW?
FC54: 90 F0              BCC   CLEOP1       NO, KEEP CLEARING LINES
FC56: B0 CA              BCS   VTAB         YES, TAB TO CURRENT LINE
FC58: A5 22     HOME     LDA   WNDTOP     INIT CURSOR V
FC5A: 85 25              STA   CV           AND H-INDICES
FC5C: A0 00              LDY   #$00
FC5E: 84 24              STY   CH         THEN CLEAR TO END OF PAGE
FC60: F0 E4              BEQ   CLEOP1
FC62: A9 00     CR       LDA   #$00       CURSOR TO LEFT OF INDEX
FC64: 85 24              STA   CH         (RET CURSOR H=0)
FC66: E6 25     LF       INC   CV         INCR CURSOR V(DOWN 1 LINE)
FC68: A5 25              LDA   CV
FC6A: C5 23              CMP   WNDBTM     OFF SCREEN?
FC6C: 90 B6              BCC   VTABZ        NO, SET BASE ADDR
FC6E: C6 25              DEC   CV         DECR CURSOR V (BACK TO BOTTOM)
FC70: A5 22     SCROLL   LDA   WNDTOP     START AT TOP OF SCRL WNDW
FC72: 48                 PHA
FC73: 20 24 FC           JSR   VTABZ      GENERATE BASE ADR
FC76: A5 28     SCRL1    LDA   BASL       COPY BASL,H
FC78: 85 2A              STA   BAS2L        TO BAS2L,H
FC7A: A5 29              LDA   BASH
FC7C: 85 2B              STA   BAS2H
FC7E: A4 21              LDY   WNDWDTH    INIT Y TO RIGHTMOST INDEX
FC80: 88                 DEY                OF SCROLLING WINDOW
FC81: 68                 PLA
FC82: 69 01              ADC   #$01       INCR LINE NUMBER
FC84: C5 23              CMP   WNDBTM     DONE?
FC86: B0 0D              BCS   SCRL3        YES, FINISH
FC88: 48                 PHA
FC89: 20 24 FC           JSR   VTABZ      FORM BASL,H (BASE ADDR)
FC8C: B1 28     SCRL2    LDA   (BASL),Y   MOVE A CHR UP ON LINE
FC8E: 91 2A              STA   (BAS2L),Y
FC90: 88                 DEY              NEXT CHAR OF LINE
FC91: 10 F9              BPL   SCRL2
FC93: 30 E1              BMI   SCRL1      NEXT LINE (ALWAYS TAKEN)
FC95: A0 00     SCRL3    LDY   #$00       CLEAR BOTTOM LINE
FC97: 20 9E FC           JSR   CLEOLZ     GET BASE ADDR FOR BOTTOM LINE
FC9A: B0 86              BCS   VTAB       CARRY IS SET
FC9C: A4 24     CLREOL   LDY   CH         CURSOR H INDEX
FC9E: A9 A0     CLEOLZ   LDA   #$A0
FCA0: 91 28     CLEOL2   STA   (BASL),Y   STORE BLANKS FROM 'HERE'
FCA2: C8                 INY                TO END OF LINES (WNDWDTH)
FCA3: C4 21              CPY   WNDWDTH
FCA5: 90 F9              BCC   CLEOL2
FCA7: 60                 RTS
FCA8: 38        WAIT     SEC
FCA9: 48        WAIT2    PHA
FCAA: E9 01     WAIT3    SBC   #$01
FCAC: D0 FC              BNE   WAIT3      1.0204 USEC
FCAE: 68                 PLA              (13+27/2*A+5/2*A*A)
FCAF: E9 01              SBC   #$01
FCB1: D0 F6              BNE   WAIT2
FCB3: 60                 RTS
FCB4: E6 42     NXTA4    INC   A4L        INCR 2-BYTE A4
FCB6: D0 02              BNE   NXTA1        AND A1
FCB8: E6 43              INC   A4H
FCBA: A5 3C     NXTA1    LDA   A1L        INCR 2-BYTE A1.
FCBC: C5 3E              CMP   A2L
FCBE: A5 3D              LDA   A1H          AND COMPARE TO A2
```

```
FCC0: E5 3F              SBC   A2H
FCC2: E6 3C              INC   A1L          (CARRY SET IF &gt=)
FCC4: D0 02              BNE   RTS4B
FCC6: E6 3D              INC   A1H
FCC8: 60        RTS4B    RTS
FCC9: A0 4B     HEADR    LDY   #$4B       WRITE A*256 'LONG 1'
FCCB: 20 DB FC           JSR   ZERDLY       HALF CYCLES
FCCE: D0 F9              BNE   HEADR        (650 USEC EACH)
FCD0: 69 FE              ADC   #$FE
FCD2: B0 F5              BCS   HEADR      THEN A 'SHORT 0'
FCD4: A0 21              LDY   #$21         (400 USEC)
FCD6: 20 DB FC  WRBIT    JSR   ZERDLY     WRITE TWO HALF CYCLES
FCD9: C8                 INY                OF 250 USEC ('0')
FCDA: C8                 INY                OR 500 USEC ('0')
FCDB: 88        ZERDLY   DEY
FCDC: D0 FD              BNE   ZERDLY
FCDE: 90 05              BCC   WRTAPE     Y IS COUNT FOR
FCE0: A0 32              LDY   #$32         TIMING LOOP
FCE2: 88        ONEDLY   DEY
FCE3: D0 FD              BNE   ONEDLY
FCE5: AC 20 C0  WRTAPE   LDY   TAPEOUT
FCE8: A0 2C              LDY   #$2C

FCEA: CA                 DEX
FCEB: 60                 RTS
FCEC: A2 08     RDBYTE   LDX   #$08       8 BITS TO READ
FCEE: 48        RDBYT2   PHA              READ TWO TRANSITIONS
FCEF: 20 FA FC           JSR   RD2BIT       (FIND EDGE)
FCF2: 68                 PLA
FCF3: 2A                 ROL              NEXT BIT
FCF4: A0 3A              LDY   #$3A       COUNT FOR SAMPLES
FCF6: CA                 DEX
FCF7: D0 F5              BNE   RDBYT2
FCF9: 60                 RTS
FCFA: 20 FD FC  RD2BIT   JSR   RDBIT
FCFD: 88        RDBIT    DEY              DECR Y UNTIL
FCFE: AD 60 C0           LDA   TAPEIN      TAPE TRANSITION
FD01: 45 2F              EOR   LASTIN
FD03: 10 F8              BPL   RDBIT
FD05: 45 2F              EOR   LASTIN
FD07: 85 2F              STA   LASTIN
FD09: C0 80              CPY   #$80       SET CARRY ON Y
FD0B: 60                 RTS
FD0C: A4 24     RDKEY    LDY   CH
FD0E: B1 28              LDA   (BASL),Y   SET SCREEN TO FLASH
FD10: 48                 PHA
FD11: 29 3F              AND   #$3F
FD13: 09 40              ORA   #$40
FD15: 91 28              STA   (BASL),Y
FD17: 68                 PLA
FD18: 6C 38 00           JMP   (KSWL)     GO TO USER KEY-IN
FD1B: E6 4E     KEYIN    INC   RNDL
FD1D: D0 02              BNE   KEYIN2     INCR RND NUMBER
FD1F: E6 4F              INC   RNDH
FD21: 2C 00 C0  KEYIN2   BIT   KBD        KEY DOWN?
FD24: 10 F5              BPL   KEYIN        LOOP
FD26: 91 28              STA   (BASL),Y   REPLACE FLASHING SCREEN
FD28: AD 00 C0           LDA   KBD        GET KEYCODE
FD2B: 2C 10 C0           BIT   KBDSTRB    CLR KEY STROBE
FD2E: 60                 RTS
FD2F: 20 0C FD  ESC      JSR   RDKEY      GET KEYCODE
FD32: 20 2C FC           JSR   ESC1         HANDLE ESC FUNC.
FD35: 20 0C FD  RDCHAR   JSR   RDKEY      READ KEY
FD38: C9 9B              CMP   #$9B       ESC?
FD3A: F0 F3              BEQ   ESC          YES, DON'T RETURN
FD3C: 60                 RTS
FD3D: A5 32     NOTCR    LDA   INVFLG
FD3F: 48                 PHA
FD40: A9 FF              LDA   #$FF
FD42: 85 32              STA   INVFLG     ECHO USER LINE
FD44: BD 00 02           LDA   IN,X         NON INVERSE
FD47: 20 ED FD           JSR   COUT
FD4A: 68                 PLA
FD4B: 85 32              STA   INVFLG
FD4D: BD 00 02           LDA   IN,X
FD50: C9 88              CMP   #$88       CHECK FOR EDIT KEYS
FD52: F0 1D              BEQ   BCKSPC       BS, CTRL-X
FD54: C9 98              CMP   #$98
FD56: F0 0A              BEQ   CANCEL
FD58: E0 F8              CPX   #$F8       MARGIN?
FD5A: 90 03              BCC   NOTCR1
FD5C: 20 3A FF           JSR   BELL         YES, SOUND BELL
FD5F: E8        NOTCR1   INX              ADVANCE INPUT INDEX
FD60: D0 13              BNE   NXTCHAR
FD62: A9 DC     CANCEL   LDA   #$DC       BACKSLASH AFTER CANCELLED LINE
FD64: 20 ED FD           JSR   COUT
```

```
FD67: 20 8E FD  GETLNZ   JSR   CROUT      OUTPUT CR
FD6A: A5 33     GETLN    LDA   PROMPT
FD6C: 20 ED FD           JSR   COUT       OUTPUT PROMPT CHAR
FD6F: A2 01              LDX   #$01       INIT INPUT INDEX
FD71: 8A        BCKSPC   TXA                WILL BACKSPACE TO 0
FD72: F0 F3              BEQ   GETLNZ
FD74: CA                 DEX
FD75: 20 35 FD  NXTCHAR  JSR   RDCHAR
FD78: C9 95              CMP   #PICK      USE SCREEN CHAR
FD7A: D0 02              BNE   CAPTST       FOR CTRL-U
FD7C: B1 28              LDA   (BASL),Y

FD7E: C9 E0     CAPTST   CMP   #$E0
FD80: 90 02              BCC   ADDINP     CONVERT TO CAPS
FD82: 29 DF              AND   #$DF
FD84: 9D 00 02  ADDINP   STA   IN,X       ADD TO INPUT BUF
FD87: C9 8D              CMP   #$8D
FD89: D0 B2              BNE   NOTCR
FD8B: 20 9C FC           JSR   CLREOL     CLR TO EOL IF CR
FD8E: A9 8D     CROUT    LDA   #$8D
FD90: D0 5B              BNE   COUT
FD92: A4 3D     PRA1     LDY   A1H        PRINT CR,A1 IN HEX
FD94: A6 3C              LDX   A1L
FD96: 20 8E FD  PRYX2    JSR   CROUT
FD99: 20 40 F9           JSR   PRNTYX
FD9C: A0 00              LDY   #$00
FD9E: A9 AD              LDA   #$AD       PRINT '-'
FDA0: 4C ED FD           JMP   COUT
FDA3: A5 3C     XAM8     LDA   A1L
FDA5: 09 07              ORA   #$07       SET TO FINISH AT
FDA7: 85 3E              STA   A2L          MOD 8=7
FDA9: A5 3D              LDA   A1H
FDAB: 85 3F              STA   A2H
FDAD: A5 3C     MODSCHK  LDA   A1L
FDAF: 29 07              AND   #$07
FDB1: D0 03              BNE   DATAOUT
FDB3: 20 92 FD  XAM      JSR   PRA1
FDB6: A9 A0     DATAOUT  LDA   #$A0
FDB8: 20 ED FD           JSR   COUT       OUTPUT BLANK
FDBB: B1 3C              LDA   (A1L),Y
FDBD: 20 DA FD           JSR   PRBYTE     OUTPUT BYTE IN HEX
FDC0: 20 BA FC           JSR   NXTA1
FDC3: 90 E8              BCC   MODSCHK    CHECK IF TIME TO,
FDC5: 60        RTS4C    RTS                PRINT ADDR
FDC6: 4A        XAMPM    LSR   A          DETERMINE IF MON
FDC7: 90 EA              BCC   XAM          MODE IS XAM
FDC9: 4A                 LSR   A            ADD, OR SUB
FDCA: 4A                 LSR   A
FDCB: A5 3E              LDA   A2L
FDCD: 90 02              BCC   ADD
FDCF: 49 FF              EOR   #$FF       SUB: FORM 2'S COMPLEMENT
FDD1: 65 3C     ADD      ADC   A1L
FDD3: 48                 PHA
FDD4: A9 BD              LDA   #$BD
FDD6: 20 ED FD           JSR   COUT       PRINT '=', THEN RESULT
FDD9: 68                 PLA
FDDA: 48        PRBYTE   PHA              PRINT BYTE AS 2 HEX
FDDB: 4A                 LSR   A            DIGITS, DESTROYS A-REG
FDDC: 4A                 LSR   A
FDDD: 4A                 LSR   A
FDDE: 4A                 LSR   A
FDDF: 20 E5 FD           JSR   PRHEXZ
FDE2: 68                 PLA
FDE3: 29 0F     PRHEX    AND   #$0F       PRINT HEX DIG IN A-REG
FDE5: 09 B0     PRHEXZ   ORA   #$B0         LSB'S
FDE7: C9 BA              CMP   #$BA
FDE9: 90 02              BCC   COUT
FDEB: 69 06              ADC   #$06
FDED: 6C 36 00  COUT     JMP   (CSWL)     VECTOR TO USER OUTPUT ROUTINE
FDF0: C9 A0     COUT1    CMP   #$A0
FDF2: 90 02              BCC   COUTZ      DON'T OUTPUT CTRL'S INVERSE
FDF4: 25 32              AND   INVFLG     MASK WITH INVERSE FLAG
FDF6: 84 35     COUTZ    STY   YSAV1      SAV Y-REG
FDF8: 48                 PHA              SAV A-REG
FDF9: 20 FD FB           JSR   VIDOUT     OUTPUT A-REG AS ASCII
FDFC: 68                 PLA              RESTORE A-REG
FDFD: A4 35              LDY   YSAV1        AND Y-REG
FDFF: 60                 RTS                THEN RETURN
FE00: C6 34     BL1      DEC   YSAV
FE02: F0 9F              BEQ   XAM8
FE04: CA        BLANK    DEX              BLANK TO MON
FE05: D0 16              BNE   SETMDZ     AFTER BLANK
FE07: C9 BA              CMP   #$BA       DATA STORE MODE?
FE09: D0 BB              BNE   XAMPM        NO, XAM, ADD, OR SUB
FE0B: 85 31     STOR     STA   MODE       KEEP IN STORE MODE
FE0D: A5 3E              LDA   A2L
```

```
FE0F: 91 40              STA   (A3L),Y    STORE AS LOW BYTE AS (A3)
FE11: E6 40              INC   A3L
FE13: D0 02              BNE   RTS5       INCR A3, RETURN
FE15: E6 41              INC   A3H
FE17: 60        RTS5     RTS
FE18: A4 34     SETMODE  LDY   YSAV       SAVE CONVERTED ':', '+',
FE1A: B9 FF 01           LDA   IN-1,Y       '-', '.' AS MODE.
FE1D: 85 31     SETMDZ   STA   MODE
FE1F: 60                 RTS
FE20: A2 01     LT       LDX   #$01
FE22: B5 3E     LT2      LDA   A2L,X      COPY A2 (2 BYTES) TO
FE24: 95 42              STA   A4L,X        A4 AND A5
FE26: 95 44              STA   A5L,X
FE28: CA                 DEX
FE29: 10 F7              BPL   LT2
FE2B: 60                 RTS
FE2C: B1 3C     MOVE     LDA   (A1L),Y    MOVE (A1 TO A2) TO
FE2E: 91 42              STA   (A4L),Y      (A4)
FE30: 20 B4 FC           JSR   NXTA4
FE33: 90 F7              BCC   MOVE
FE35: 60                 RTS
FE36: B1 3C     VFY      LDA   (A1L),Y    VERIFY (A1 TO A2) WITH
FE38: D1 42              CMP   (A4L),Y      (A4)
FE3A: F0 1C              BEQ   VFYOK
FE3C: 20 92 FD           JSR   PRA1
FE3F: B1 3C              LDA   (A1L),Y
FE41: 20 DA FD           JSR   PRBYTE
FE44: A9 A0              LDA   #$A0
FE46: 20 ED FD           JSR   COUT
FE49: A9 A8              LDA   #$A8
FE4B: 20 ED FD           JSR   COUT
FE4E: B1 42              LDA   (A4L),Y
FE50: 20 DA FD           JSR   PRBYTE
FE53: A9 A9              LDA   #$A9
FE55: 20 ED FD           JSR   COUT
FE58: 20 B4 FC  VFYOK    JSR   NXTA4
FE5B: 90 D9              BCC   VFY
FE5D: 60                 RTS
FE5E: 20 75 FE  LIST     JSR   A1PC       MOVE A1 (2 BYTES) TO
FE61: A9 14              LDA   #$14         PC IF SPEC'D AND
FE63: 48        LIST2    PHA                DISEMBLE 20 INSTRS
FE64: 20 D0 F8           JSR   INSTDSP
FE67: 20 53 F9           JSR   PCADJ      ADJUST PC EACH INSTR
FE6A: 85 3A              STA   PCL
FE6C: 84 3B              STY   PCH
FE6E: 68                 PLA
FE6F: 38                 SEC
FE70: E9 01              SBC   #$01       NEXT OF 20 INSTRS
FE72: D0 EF              BNE   LIST2
FE74: 60                 RTS
FE75: 8A        A1PC     TXA              IF USER SPEC'D ADR
FE76: F0 07              BEQ   A1PCRTS      COPY FROM A1 TO PC
FE78: B5 3C     A1PCLP   LDA   A1L,X
FE7A: 95 3A              STA   PCL,X
FE7C: CA                 DEX
FE7D: 10 F9              BPL   A1PCLP
FE7F: 60        A1PCRTS  RTS
FE80: A0 3F     SETINV   LDY   #$3F       SET FOR INVERSE VID
FE82: D0 02              BNE   SETIFLG     VIA COUT1
FE84: A0 FF     SETNORM  LDY   #$FF       SET FOR NORMAL VID
FE86: 84 32     SETIFLG  STY   INVFLG
FE88: 60                 RTS
FE89: A9 00     SETKBD   LDA   #$00       SIMULATE PORT #0 INPUT
FE8B: 85 3E     INPORT   STA   A2L          SPECIFIED (KEYIN ROUTINE)
FE8D: A2 38     INPRT    LDX   #KSWL
FE8F: A0 1B              LDY   #KEYIN
FE91: D0 08              BNE   IOPRT
FE93: A9 00     SETVID   LDA   #$00       SIMULATE PORT #0 OUTPUT
FE95: 85 3E     OUTPORT  STA   A2L          SPECIFIED (COUT1 ROUTINE)
FE97: A2 36     OUTPRT   LDX   #CSWL
FE99: A0 F0              LDY   #COUT1
FE9B: A5 3E     IOPRT    LDA   A2L        SET RAM IN/OUT VECTORS
FE9D: 29 0F              AND   #$0F
FE9F: F0 06              BEQ   IOPRT1
FEA1: 09 C0              ORA   #IOADR/256
FEA3: A0 00              LDY   #$00
FEA5: F0 02              BEQ   IOPRT2
FEA7: A9 FD     IOPRT1   LDA   #COUT1/256
FEA9: 94 00     IOPRT2   STY   LOC0,X
FEAB: 95 01              STA   LOC1,X
FEAD: 60                 RTS
FEAE: EA                 NOP
FEAF: EA                 NOP
FEB0: 4C 00 E0  XBASIC   JMP   BASIC      TO BASIC WITH SCRATCH
FEB3: 4C 03 E0  BASCONT  JMP   BASIC2     CONTINUE BASIC
```

```
FEB6: 20 75 FE  GO       JSR   A1PC      ADR TO PC IF SPEC'D
FEB9: 20 3F FF           JSR   RESTORE   RESTORE META REGS
FEBC: 6C 3A 00           JMP   (PCL)     GO TO USER SUBR
FEBF: 4C D7 FA  REGZ     JMP   REGDSP    TO REG DISPLAY
FEC2: C6 34     TRACE    DEC   YSAV
FEC4: 20 75 FE  STEPZ    JSR   A1PC      ADR TO PC IF SPEC'D
FEC7: 4C 43 FA           JMP   STEP      TAKE ONE STEP
FECA: 4C F8 03  USR      JMP   USRADR    TO USR SUBR AT USRADR
FECD: A9 40     WRITE    LDA   #$40
FECF: 20 C9 FC           JSR   HEADR     WRITE 10-SEC HEADER
FED2: A0 27              LDY   #$27
FED4: A2 00     WR1      LDX   #$00
FED6: 41 3C              EOR   (A1L,X)
FED8: 48                 PHA
FED9: A1 3C              LDA   (A1L,X)
FEDB: 20 ED FE           JSR   WRBYTE
FEDE: 20 BA FC           JSR   NXTA1
FEE1: A0 1D              LDY   #$1D
FEE3: 68                 PLA
FEE4: 90 EE              BCC   WR1
FEE6: A0 22              LDY   #$22
FEE8: 20 ED FE           JSR   WRBYTE
FEEB: F0 4D              BEQ   BELL
FEED: A2 10     WRBYTE   LDX   #$10
FEEF: 0A        WRBYT2   ASL   A
FEF0: 20 D6 FC           JSR   WRBIT
FEF3: D0 FA              BNE   WRBYT2
FEF5: 60                 RTS
FEF6: 20 00 FE  CRMON    JSR   BL1       HANDLE A CR AS BLANK
FEF9: 68                 PLA               THEN POP STACK
FEFA: 68                 PLA               AND RTN TO MON
FEFB: D0 6C              BNE   MONZ
FEFD: 20 FA FC  READ     JSR   RD2BIT    FIND TAPEIN EDGE
FF00: A9 16              LDA   #$16
FF02: 20 C9 FC           JSR   HEADR     DELAY 3.5 SECONDS
FF05: 85 2E              STA   CHKSUM    INIT CHKSUM=$FF
FF07: 20 FA FC           JSR   RD2BIT    FIND TAPEIN EDGE
FF0A: A0 24     RD2      LDY   #$24      LOOK FOR SYNC BIT
FF0C: 20 FD FC           JSR   RDBIT       (SHORT 0)
FF0F: B0 F9              BCS   RD2         LOOP UNTIL FOUND
FF11: 20 FD FC           JSR   RDBIT     SKIP SECOND SYNC H-CYCLE
FF14: A0 3B              LDY   #$3B      INDEX FOR 0/1 TEST
FF16: 20 EC FC  RD3      JSR   RDBYTE    READ A BYTE
FF19: 81 3C              STA   (A1L,X)   STORE AT (A1)
FF1B: 45 2E              EOR   CHKSUM
FF1D: 85 2E              STA   CHKSUM    UPDATE RUNNING CHKSUM
FF1F: 20 BA FC           JSR   NXTA1     INC A1, COMPARE TO A2
FF22: A0 35              LDY   #$35      COMPENSATE 0/1 INDEX
FF24: 90 F0              BCC   RD3       LOOP UNTIL DONE
FF26: 20 EC FC           JSR   RDBYTE    READ CHKSUM BYTE
FF29: C5 2E              CMP   CHKSUM
FF2B: F0 0D              BEQ   BELL      GOOD, SOUND BELL AND RETURN
FF2D: A9 C5     PRERR    LDA   #$C5
FF2F: 20 ED FD           JSR   COUT      PRINT "ERR", THEN BELL
FF32: A9 D2              LDA   #$D2
FF34: 20 ED FD           JSR   COUT
FF37: 20 ED FD           JSR   COUT
FF3A: A9 87     BELL     LDA   #$87      OUTPUT BELL AND RETURN
FF3C: 4C ED FD           JMP   COUT
FF3F: A5 48     RESTORE  LDA   STATUS    RESTORE 6502 REG CONTENTS
FF41: 48                 PHA               USED BY DEBUG SOFTWARE
FF42: A5 45              LDA   ACC
FF44: A6 46     RESTR1   LDX   XREG
FF46: A4 47              LDY   YREG
FF48: 28                 PLP
FF49: 60                 RTS
FF4A: 85 45     SAVE     STA   ACC       SAVE 6502 REG CONTENTS
FF4C: 86 46     SAV1     STX   XREG
FF4E: 84 47              STY   YREG
FF50: 08                 PHP
FF51: 68                 PLA
FF52: 85 48              STA   STATUS
FF54: BA                 TSX
FF55: 86 49              STX   SPNT
FF57: D8                 CLD
FF58: 60                 RTS
FF59: 20 84 FE  RESET    JSR   SETNORM   SET SCREEN MODE
FF5C: 20 2F FB           JSR   INIT        AND INIT KBD/SCREEN
FF5F: 20 93 FE           JSR   SETVID      AS I/O DEV'S
FF62: 20 89 FE           JSR   SETKBD
FF65: D8        MON      CLD             MUST SET HEX MODE!
FF66: 20 3A FF           JSR   BELL
FF69: A9 AA     MONZ     LDA   #$AA      '*' PROMPT FOR MON
FF6B: 85 33              STA   PROMPT
FF6D: 20 67 FD           JSR   GETLNZ    READ A LINE
```

```
FF70: 20 C7 FF          JSR   ZMODE      CLEAR MON MODE, SCAN IDX
FF73: 20 A7 FF  NXTITM  JSR   GETNUM     GET ITEM, NON-HEX
FF76: 84 34             STY   YSAV         CHAR IN A-REG
FF78: A0 17             LDY   #$17         X-REG=0 IF NO HEX INPUT
FF7A: 88        CHRSRCH DEY
FF7B: 30 E8             BMI   MON        NOT FOUND, GO TO MON
FF7D: D9 CC FF          CMP   CHRTBL,Y   FIND CMND CHAR IN TEL
FF80: D0 F8             BNE   CHRSRCH
FF82: 20 BE FF          JSR   TOSUB      FOUND, CALL CORRESPONDING
FF85: A4 34             LDY   YSAV         SUBROUTINE
FF87: 4C 73 FF          JMP   NXTITM
FF8A: A2 03     DIG     LDX   #$03
FF8C: 0A                ASL   A
FF8D: 0A                ASL   A          GOT HEX DIG,
FF8E: 0A                ASL   A            SHIFT INTO A2
FF8F: 0A                ASL   A
FF90: 0A        NXTBIT  ASL   A
FF91: 26 3E             ROL   A2L
FF93: 26 3F             ROL   A2H
FF95: CA                DEX              LEAVE X=$FF IF DIG
FF96: 10 F8             BPL   NXTBIT
FF98: A5 31     NXTBAS  LDA   MODE
FF9A: D0 06             BNE   NXTBS2     IF MODE IS ZERO
FF9C: B5 3F             LDA   A2H,X       THEN COPY A2 TO
FF9E: 95 3D             STA   A1H,X       A1 AND A3
FFA0: 95 41             STA   A3H,X
FFA2: E8        NXTBS2  INX
FFA3: F0 F3             BEQ   NXTBAS
FFA5: D0 06             BNE   NXTCHR
FFA7: A2 00     GETNUM  LDX   #$00       CLEAR A2
FFA9: 86 3E             STX   A2L
FFAB: 86 3F             STX   A2H
FFAD: B9 00 02  NXTCHR  LDA   IN,Y       GET CHAR
FFB0: C8                INY
FFB1: 49 B0             EOR   #$B0
FFB3: C9 0A             CMP   #$0A
FFB5: 90 D3             BCC   DIG        IF HEX DIG, THEN
FFB7: 69 88             ADC   #$88
FFB9: C9 FA             CMP   #$FA
FFBB: B0 CD             BCS   DIG
FFBD: 60                RTS
FFBE: A9 FE     TOSUB   LDA   #GO/256    PUSH HIGH-ORDER
FFC0: 48                PHA                SUBR ADR ON STK
FFC1: B9 E3 FF          LDA   SUBTBL,Y   PUSH LOW-ORDER
FFC4: 48                PHA                SUBR ADR ON STK
FFC5: A5 31             LDA   MODE
FFC7: A0 00     ZMODE   LDY   #$00       CLR MODE, OLD MODE

FFC9: 84 31             STY   MODE         TO A-REG
FFCB: 60                RTS               GO TO SUBR VIA RTS
FFCC: BC        CHRTBL  DFB   $BC        F("CTRL-C")
FFCD: B2                DFB   $B2        F("CTRL-Y")
FFCE: BE                DFB   $BE        F("CTRL-E")
FFCF: ED                DFB   $ED        F("T")
FFD0: EF                DFB   $EF        F("V")
FFD1: C4                DFB   $C4        F("CTRL-K")
FFD2: EC                DFB   $EC        F("S")
FFD3: A9                DFB   $A9        F("CTRL-P")
FFD4: BB                DFB   $BB        F("CTRL-B")
FFD5: A6                DFB   $A6        F("-")
FFD6: A4                DFB   $A4        F("+")
FFD7: 06                DFB   $06        F("M") (F=EX-OR $B0+$89)
FFD8: 95                DFB   $95        F("&lt")
FFD9: 07                DFB   $07        F("N")
FFDA: 02                DFB   $02        F("I")
FFDB: 05                DFB   $05        F("L")
FFDC: F0                DFB   $F0        F("W")
FFDD: 00                DFB   $00        F("G")
FFDE: EB                DFB   $EB        F("R")
FFDF: 93                DFB   $93        F(":")
FFE0: A7                DFB   $A7        F(".")
FFE1: C6                DFB   $C6        F("CR")
FFE2: 99                DFB   $99        F(BLANK)
FFE3: B2        SUBTBL  DFB   BASCONT-1
FFE4: C9                DFB   USR-1
FFE5: BE                DFB   REGZ-1
FFE6: C1                DFB   TRACE-1
FFE7: 35                DFB   VFY-1
FFE8: 8C                DFB   INPRT-1
FFE9: C3                DFB   STEPZ-1
FFEA: 96                DFB   OUTPRT-1
FFEB: AF                DFB   XBASIC-1
FFEC: 17                DFB   SETMODE-1
FFED: 17                DFB   SETMODE-1
FFEE: 2B                DFB   MOVE-1
FFEF: 1F                DFB   LT-1
```

```
FFF0: 83                DFB   SETNORM-1
FFF1: 7F                DFB   SETINV-1
FFF2: 5D                DFB   LIST-1
FFF3: CC                DFB   WRITE-1
FFF4: B5                DFB   GO-1
FFF5: FC                DFB   READ-1
FFF6: 17                DFB   SETMODE-1
FFF7: 17                DFB   SETMODE-1
FFF8: F5                DFB   CRMON-1
FFF9: 03                DFB   BLANK-1
FFFA: FB                DFB   NMI        NMI VECTOR
FFFB: 03                DFB   NMI/256
FFFC: 59                DFB   RESET      RESET VECTOR
FFFD: FF                DFB   RESET/256
FFFE: 86                DFB   IRQ        IRQ VECTOR
FFFF: FA                DFB   IRQ/256
                XQTNZ   EQU   $3C
```

```
                    **********************
                    *                    *
                    *      APPLE-II      *
                    *   MINI-ASSEMBLER   *
                    *                    *
                    *  COPYRIGHT 1977 BY *
                    * APPLE COMPUTER INC. *
                    *                    *
                    * ALL RIGHTS RESERVED *
                    *                    *
                    *     S. WOZNIAK     *
                    *      A. BAUM       *
                    **********************
                     TITLE "APPLE-II MINI-ASSEMBLER"
                    FORMAT   EQU   $2E
                    LENGTH   EQU   $2F
                    MODE     EQU   $31
                    PROMPT   EQU   $33
                    YSAV     EQU   $34
                    L        EQU   $35
                    PCL      EQU   $3A
                    PCH      EQU   $3B
                    A1H      EQU   $3D
                    A2L      EQU   $3E
                    A2H      EQU   $3F
                    A4L      EQU   $42
                    A4H      EQU   $43
                    FMT      EQU   $44
                    IN       EQU   $200
                    INSDS2   EQU   $F88E
                    INSTDSP  EQU   $F8D0
                    PRBL2    EQU   $F94A
                    PCADJ    EQU   $F953
                    CHAR1    EQU   $F9B4
                    CHAR2    EQU   $F9BA
                    MNEML    EQU   $F9C0
                    MNEMR    EQU   $FA00
                    CURSUP   EQU   $FC1A
                    GETLNZ   EQU   $FD67
                    COUT     EQU   $FDED
                    BL1      EQU   $FE00
                    A1PCLP   EQU   $FE78
                    BELL     EQU   $FF3A
                    GETNUM   EQU   $FFA7
                    TOSUB    EQU   $FFBE
                    ZMODE    EQU   $FFC7
                    CHRTBL   EQU   $FFCC
                             ORG   $F500
F500: E9 81         REL      SBC   #$81       IS FMT COMPATIBLE
F502: 4A                     LSR              WITH RELATIVE MODE?
F503: D0 14                  BNE   ERR3       NO.
F505: A4 3F                  LDY   A2H
F507: A6 3E                  LDX   A2L        DOUBLE DECREMENT
F509: D0 01                  BNE   REL2
F50B: 88                     DEY
F50C: CA            REL2     DEX
F50D: 8A                     TXA
F50E: 18                     CLC
F50F: E5 3A                  SBC   PCL        FORM ADDR-PC-2
F511: 85 3E                  STA   A2L
F513: 10 01                  BPL   REL3
F515: C8                     INY
F516: 98            REL3     TYA
```

```
F517: E5 3B              SBC   PCH
F519: D0 6B     ERR3     BNE   ERR          ERROR IF >1-BYTE BRANCH
F51B: A4 2F     FINDOP   LDY   LENGTH
F51D: B9 3D 00  FNDOP2   LDA   A1H,Y        MOVE INST TO (PC)
F520: 91 3A              STA   (PCL),Y
F522: 88                 DEY
F523: 10 F8              BPL   FNDOP2
F525: 20 1A FC           JSR   CURSUP
F528: 20 1A FC           JSR   CURSUP       RESTORE CURSOR
F52B: 20 D0 F8           JSR   INSTDSP      TYPE FORMATTED LINE
F52E: 20 53 F9           JSR   PCADJ        UPDATE PC
F531: 84 3B              STY   PCH
F533: 85 3A              STA   PCL
F535: 4C 95 F5           JMP   NXTLINE      GET NEXT LINE
F538: 20 BE FF  FAKEMON3 JSR   TOSUB        GO TO DELIM HANDLER
F53B: A4 34              LDY   YSAV         RESTORE Y-INDEX
F53D: 20 A7 FF  FAKEMON  JSR   GETNUM       READ PARAM
F540: 84 34              STY   YSAV         SAVE Y-INDEX
F542: A0 17              LDY   #$17         INIT DELIMITER INDEX
F544: 88        FAKEMON2 DEY                CHECK NEXT DELIM
F545: 30 4B              BMI   RESETZ       ERR IF UNRECOGNIZED DELIM
F547: D9 CC FF           CMP   CHRTBL,Y     COMPARE WITH DELIM TABLE
F54A: D0 F8              BNE   FAKEMON2     NO MATCH
F54C: C0 15              CPY   #$15         MATCH, IS IT CR?
F54E: D0 E8              BNE   FAKEMON3     NO, HANDLE IT IN MONITOR
F550: A5 31              LDA   MODE
F552: A0 00              LDY   #$0
F554: C6 34              DEC   YSAV
F556: 20 00 FE           JSR   BL1          HANDLE CR OUTSIDE MONITOR
F559: 4C 95 F5           JMP   NXTLINE
F55C: A5 3D     TRYNEXT  LDA   A1H          GET TRIAL OPCODE
F55E: 20 8E F8           JSR   INSDS2       GET FMT+LENGTH FOR OPCODE
F561: AA                 TAX
F562: BD 00 FA           LDA   MNEMR,X      GET LOWER MNEMONIC BYTE
F565: C5 42              CMP   A4L          MATCH?
F567: D0 13              BNE   NEXTOP       NO, TRY NEXT OPCODE.
F569: BD C0 F9           LDA   MNEML,X      GET UPPER MNEMONIC BYTE
F56C: C5 43              CMP   A4H          MATCH?
F56E: D0 0C              BNE   NEXTOP       NO, TRY NEXT OPCODE
F570: A5 44              LDA   FMT
F572: A4 2E              LDY   FORMAT       GET TRIAL FORMAT
F574: C0 9D              CPY   #$9D         TRIAL FORMAT RELATIVE?
F576: F0 88              BEQ   REL          YES.
F578: C5 2E     NREL     CMP   FORMAT       SAME FORMAT?
F57A: F0 9F              BEQ   FINDOP       YES.
F57C: C6 3D     NEXTOP   DEC   A1H          NO, TRY NEXT OPCODE
F57E: D0 DC              BNE   TRYNEXT
F580: E6 44              INC   FMT          NO MORE, TRY WITH LEN=2
F582: C6 35              DEC   L            WAS L=2 ALREADY?
F584: F0 D6              BEQ   TRYNEXT      NO.
F586: A4 34     ERR      LDY   YSAV         YES, UNRECOGNIZED INST.
F588: 98        ERR2     TYA
F589: AA                 TAX
F58A: 20 4A F9           JSR   PRBL2        PRINT ^ UNDER LAST READ
F58D: A9 DE              LDA   #$DE         CHAR TO INDICATE ERROR
F58F: 20 ED FD           JSR   COUT         POSITION.
F592: 20 3A FF  RESETZ   JSR   BELL
F595: A9 A1     NXTLINE  LDA   #$A1         '!'
F597: 85 33              STA   PROMPT       INITIALIZE PROMPT
F599: 20 67 FD           JSR   GETLNZ       GET LINE.
F59C: 20 C7 FF           JSR   ZMODE        INIT SCREEN STUFF
F59F: AD 00 02           LDA   IN           GET CHAR
F5A2: C9 A0              CMP   #$A0         ASCII BLANK?
F5A4: F0 13              BEQ   SPACE        YES
F5A6: C8                 INY
F5A7: C9 A4              CMP   #$A4         ASCII '$' IN COL 1?
F5A9: F0 92              BEQ   FAKEMON      YES, SIMULATE MONITOR
F5AB: 88                 DEY                NO, BACKUP A CHAR
F5AC: 20 A7 FF           JSR   GETNUM       GET A NUMBER
F5AF: C9 93              CMP   #$93         ':' TERMINATOR?
F5B1: D0 D5     ERR4     BNE   ERR2         NO, ERR.
F5B3: 8A                 TXA
F5B4: F0 D2              BEQ   ERR2         NO ADR PRECEDING COLON.
F5B6: 20 78 FE           JSR   A1PCLP       MOVE ADR TO PCL, PCH.
F5B9: A9 03     SPACE    LDA   #$3          COUNT OF CHARS IN MNEMONIC
F5BB: 85 3D              STA   A1H
F5BD: 20 34 F6  NXTMN    JSR   GETNSP       GET FIRST MNEM CHAR.
F5C0: 0A        NXTM     ASL   A
F5C1: E9 BE              SBC   #$BE         SUBTRACT OFFSET
F5C3: C9 C2              CMP   #$C2         LEGAL CHAR?
F5C5: 90 C1              BCC   ERR2         NO.
F5C7: 0A                 ASL   A            COMPRESS-LEFT JUSTIFY
F5C8: 0A                 ASL   A
F5C9: A2 04              LDX   #$4
F5CB: 0A        NXTM2    ASL   A            DO 5 TRIPLE WORD SHIFTS
```

```
F5CC: 26 42               ROL   A4L
F5CE: 26 43               ROL   A4H
F5D0: CA                  DEX
F5D1: 10 F8               BPL   NXTM2
F5D3: C6 3D               DEC   A1H        DONE WITH 3 CHARS?
F5D5: F0 F4               BEQ   NXTM2      YES, BUT DO 1 MORE SHIFT
F5D7: 10 E4               BPL   NXTMN      NO
F5D9: A2 05      FORM1    LDX   #$5        5 CHARS IN ADDR MODE
F5DB: 20 34 F6   FORM2    JSR   GETNSP     GET FIRST CHAR OF ADDR
F5DE: 84 34               STY   YSAV
F5E0: DD B4 F9            CMP   CHAR1,X    FIRST CHAR MATCH PATTERN?
F5E3: D0 13               BNE   FORM3      NO
F5E5: 20 34 F6            JSR   GETNSP     YES, GET SECOND CHAR
F5E8: DD BA F9            CMP   CHAR2,X    MATCHES SECOND HALF?
F5EB: F0 0D               BEQ   FORM5      YES.
F5ED: BD BA F9            LDA   CHAR2,X    NO, IS SECOND HALF ZERO?
F5F0: F0 07               BEQ   FORM4      YES.
F5F2: C9 A4               CMP   #$A4       NO,SECOND HALF OPTIONAL?
F5F4: F0 03               BEQ   FORM4      YES.
F5F6: A4 34               LDY   YSAV
F5F8: 18         FORM3    CLC              CLEAR BIT-NO MATCH
F5F9: 88         FORM4    DEY              BACK UP 1 CHAR
F5FA: 26 44      FORM5    ROL   FMT        FORM FORMAT BYTE
F5FC: E0 03               CPX   #$3        TIME TO CHECK FOR ADDR.
F5FE: D0 0D               BNE   FORM7      NO
F600: 20 A7 FF            JSR   GETNUM     YES
F603: A5 3F               LDA   A2H
F605: F0 01               BEQ   FORM6      HIGH-ORDER BYTE ZERO
F607: E8                  INX              NO, INCR FOR 2-BYTE
F608: 86 35      FORM6    STX   L          STORE LENGTH
F60A: A2 03               LDX   #$3        RELOAD FORMAT INDEX
F60C: 88                  DEY              BACKUP A CHAR
F60D: 86 3D      FORM7    STX   A1H        SAVE INDEX
F60F: CA                  DEX              DONE WITH FORMAT CHECK?
F610: 10 C9               BPL   FORM2      NO.
F612: A5 44               LDA   FMT        YES, PUT LENGTH
F614: 0A                  ASL   A           IN LOW BITS
F615: 0A                  ASL   A
F616: 05 35               ORA   L
F618: C9 20               CMP   #$20
F61A: B0 06               BCS   FORM8      ADD "$" IF NONZERO LENGTH
F61C: A6 35               LDX   L          AND DON'T ALREADY HAVE IT
F61E: F0 02               BEQ   FORM8
F620: 09 80               ORA   #$80
F622: 85 44      FORM8    STA   FMT
F624: 84 34               STY   YSAV
F626: B9 00 02            LDA   IN,Y       GET NEXT NONBLANK
F629: C9 BB               CMP   #$BB       '' START OF COMMENT?
F62B: F0 04               BEQ   FORM9      YES
F62D: C9 8D               CMP   #$8D       CARRIAGE RETURN?
F62F: D0 80               BNE   ERR4       NO, ERR.
F631: 4C 5C F5   FORM9    JMP   TRYNEXT
F634: B9 00 02   GETNSP   LDA   IN,Y
F637: C8                  INY
F638: C9 A0               CMP   #$A0       GET NEXT NON BLANK CHAR
F63A: F0 F8               BEQ   GETNSP
F63C: 60                  RTS
                          ORG   $F666
F666: 4C 92 F5   MINIASM  JMP   RESETZ
```

```
                  **********************
                  *                    *
                  *  APPLE-II FLOATING *
                  *    POINT ROUTINES  *
                  *                    *
                  *  COPYRIGHT 1977 BY *
                  * APPLE COMPUTER INC. *
                  *                    *
                  * ALL RIGHTS RESERVED *
                  *                    *
                  *     S. WOZNIAK     *
                  *                    *
                  **********************
                   TITLE "FLOATING POINT ROUTINES"
                  SIGN      EPZ  $F3
                  X2        EPZ  $F4
                  M2        EPZ  $F5
                  X1        EPZ  $F8
                  M1        EPZ  $F9
                  E         EPZ  $FC
                  OVLOC     EQU  $3F5
                            ORG  $F425
F425: 18          ADD       CLC            CLEAR CARRY
F426: A2 02                 LDX  #$2       INDEX FOR 3-BYTE ADD.
F428: B5 F9       ADD1      LDA  M1,X
F42A: 75 F5                 ADC  M2,X      ADD A BYTE OF MANT2 TO MANT1
F42C: 95 F9                 STA  M1,X
F42E: CA                    DEX            INDEX TO NEXT MORE SIGNIF. BYTE.
F42F: 10 F7                 BPL  ADD1      LOOP UNTIL DONE.
F431: 60                    RTS            RETURN
F432: 06 F3       MD1       ASL  SIGN      CLEAR LSB OF SIGN.
F434: 20 37 F4              JSR  ABSWAP    ABS VAL OF M1, THEN SWAP WITH M2
F437: 24 F9       ABSWAP    BIT  M1        MANT1 NEGATIVE?
F439: 10 05                 BPL  ABSWAP1   NO, SWAP WITH MANT2 AND RETURN.
F43B: 20 A4 F4              JSR  FCOMPL    YES, COMPLEMENT IT.
F43E: E6 F3                 INC  SIGN      INCR SIGN, COMPLEMENTING LSB.
F440: 38          ABSWAP1   SEC            SET CARRY FOR RETURN TO MUL/DIV.
F441: A2 04       SWAP      LDX  #$4       INDEX FOR 4 BYTE SWAP.
F443: 94 FB       SWAP1     STY  E-1,X
F445: B5 F7                 LDA  X1-1,X    SWAP A BYTE OF EXP/MANT1 WITH
F447: B4 F3                 LDY  X2-1,X    EXP/MANT2 AND LEAVE A COPY OF
F449: 94 F7                 STY  X1-1,X    MANT1 IN E (3 BYTES).  E+3 USED
F44B: 95 F3                 STA  X2-1,X
F44D: CA                    DEX            ADVANCE INDEX TO NEXT BYTE
F44E: D0 F3                 BNE  SWAP1     LOOP UNTIL DONE.
F450: 60                    RTS            RETURN
F451: A9 8E       FLOAT     LDA  #$8E      INIT EXP1 TO 14,
F453: 85 F8                 STA  X1        THEN NORMALIZE TO FLOAT.
F455: A5 F9       NORM1     LDA  M1        HIGH-ORDER MANT1 BYTE.
F457: C9 C0                 CMP  #$C0      UPPER TWO BITS UNEQUAL?
F459: 30 0C                 BMI  RTS1      YES, RETURN WITH MANT1 NORMALIZED
F45B: C6 F8                 DEC  X1        DECREMENT EXP1.
F45D: 06 FB                 ASL  M1+2
F45F: 26 FA                 ROL  M1+1      SHIFT MANT1 (3 BYTES) LEFT.
F461: 26 F9                 ROL  M1
F463: A5 F8       NORM      LDA  X1        EXP1 ZERO?
F465: D0 EE                 BNE  NORM1     NO, CONTINUE NORMALIZING.
F467: 60          RTS1      RTS            RETURN.
F468: 20 A4 F4    FSUB      JSR  FCOMPL    CMPL MANT1,CLEARS CARRY UNLESS 0
F46B: 20 7B F4    SWPALGN   JSR  ALGNSWP   RIGHT SHIFT MANT1 OR SWAP WITH
F46E: A5 F4       FADD      LDA  X2
F470: C5 F8                 CMP  X1        COMPARE EXP1 WITH EXP2.
F472: D0 F7                 BNE  SWPALGN   IF #,SWAP ADDENDS OR ALIGN MANTS.
F474: 20 25 F4              JSR  ADD       ADD ALIGNED MANTISSAS.
F477: 50 EA       ADDEND    BVC  NORM      NO OVERFLOW, NORMALIZE RESULT.
F479: 70 05                 BVS  RTLOG     OV: SHIFT M1 RIGHT, CARRY INTO SIGN
```

```
F47B: 90 C4     ALGNSWP   BCC  SWAP     SWAP IF CARRY CLEAR,
                *       ELSE SHIFT RIGHT ARITH.
F47D: A5 F9     RTAR      LDA  M1       SIGN OF MANT1 INTO CARRY FOR
F47F: 0A                  ASL           RIGHT ARITH SHIFT.
F480: E6 F8     RTLOG     INC  X1       INCR X1 TO ADJUST FOR RIGHT SHIFT
F482: F0 75               BEQ  OVFL     EXP1 OUT OF RANGE.
F484: A2 FA     RTLOG1    LDX  #$FA     INDEX FOR 6:BYTE RIGHT SHIFT.
F486: 76 FF     ROR1      ROR  E+3,X
F488: E8                  INX           NEXT BYTE OF SHIFT.
F489: D0 FB               BNE  ROR1     LOOP UNTIL DONE.
F48B: 60                  RTS           RETURN.
F48C: 20 32 F4  FMUL      JSR  MD1      ABS VAL OF MANT1, MANT2
F48F: 65 F8               ADC  X1       ADD EXP1 TO EXP2 FOR PRODUCT EXP
F491: 20 E2 F4            JSR  MD2      CHECK PROD. EXP AND PREP. FOR MUL
F494: 18                  CLC           CLEAR CARRY FOR FIRST BIT.
F495: 20 84 F4  MUL1      JSR  RTLOG1   M1 AND E RIGHT (PROD AND MPLIER)
F498: 90 03               BCC  MUL2     IF CARRY CLEAR, SKIP PARTIAL PROD
F49A: 20 25 F4            JSR  ADD      ADD MULTIPLICAND TO PRODUCT.
F49D: 88        MUL2      DEY           NEXT MUL ITERATION.
F49E: 10 F5               BPL  MUL1     LOOP UNTIL DONE.
F4A0: 46 F3     MDEND     LSR  SIGN     TEST SIGN LSB.
F4A2: 90 BF     NORMX     BCC  NORM     IF EVEN,NORMALIZE PROD,ELSE COMP
F4A4: 38        FCOMPL    SEC           SET CARRY FOR SUBTRACT.
F4A5: A2 03               LDX  #$3      INDEX FOR 3 BYTE SUBTRACT.
F4A7: A9 00     COMPL1    LDA  #$0      CLEAR A.
F4A9: F5 F8               SBC  X1,X     SUBTRACT BYTE OF EXP1.
F4AB: 95 F8               STA  X1,X     RESTORE IT.
F4AD: CA                  DEX           NEXT MORE SIGNIFICANT BYTE.
F4AE: D0 F7               BNE  COMPL1   LOOP UNTIL DONE.
F4B0: F0 C5               BEQ  ADDEND   NORMALIZE (OR SHIFT RT IF OVFL).
F4B2: 20 32 F4  FDIV      JSR  MD1      TAKE ABS VAL OF MANT1, MANT2.
F4B5: E5 F8               SBC  X1       SUBTRACT EXP1 FROM EXP2.
F4B7: 20 E2 F4            JSR  MD2      SAVE AS QUOTIENT EXP.
F4BA: 38        DIV1      SEC           SET CARRY FOR SUBTRACT.
F4BB: A2 02               LDX  #$2      INDEX FOR 3-BYTE SUBTRACTION.
F4BD: B5 F5     DIV2      LDA  M2,X
F4BF: F5 FC               SBC  E,X      SUBTRACT A BYTE OF E FROM MANT2.
F4C1: 48                  PHA           SAVE ON STACK.
F4C2: CA                  DEX           NEXT MORE SIGNIFICANT BYTE.
F4C3: 10 F8               BPL  DIV2     LOOP UNTIL DONE.
F4C5: A2 FD               LDX  #$FD     INDEX FOR 3-BYTE CONDITIONAL MOVE
F4C7: 68        DIV3      PLA           PULL BYTE OF DIFFERENCE OFF STACK
F4C8: 90 02               BCC  DIV4     IF M2<E THEN DON'T RESTORE M2.
F4CA: 95 F8               STA  M2+3,X
F4CC: E8        DIV4      INX           NEXT LESS SIGNIFICANT BYTE.
F4CD: D0 F8               BNE  DIV3     LOOP UNTIL DONE.
F4CF: 26 FB               ROL  M1+2
F4D1: 26 FA               ROL  M1+1     ROLL QUOTIENT LEFT, CARRY INTO LSB
F4D3: 26 F9               ROL  M1
F4D5: 06 F7               ASL  M2+2
F4D7: 26 F6               ROL  M2+1     SHIFT DIVIDEND LEFT
F4D9: 26 F5               ROL  M2
F4DB: B0 1C               BCS  OVFL     OVFL IS DUE TO UNNORMED DIVISOR
F4DD: 88                  DEY           NEXT DIVIDE ITERATION.
F4DE: D0 DA               BNE  DIV1     LOOP UNTIL DONE 23 ITERATIONS.
F4E0: F0 BE               BEQ  MDEND    NORM. QUOTIENT AND CORRECT SIGN.
F4E2: 86 FB     MD2       STX  M1+2
F4E4: 86 FA               STX  M1+1     CLEAR MANT1 (3 BYTES) FOR MUL/DIV.
F4E6: 86 F9               STX  M1
F4E8: B0 0D               BCS  OVCHK    IF CALC. SET CARRY,CHECK FOR OVFL
F4EA: 30 04               BMI  MD3      IF NEG THEN NO UNDERFLOW.
F4EC: 68                  PLA           POP ONE RETURN LEVEL.
F4ED: 68                  PLA
F4EE: 90 B2               BCC  NORMX    CLEAR X1 AND RETURN.
F4F0: 49 80     MD3       EOR  #$80     COMPLEMENT SIGN BIT OF EXPONENT.
F4F2: 85 F8               STA  X1       STORE IT.
F4F4: A0 17               LDY  #$17     COUNT 24 MUL/23 DIV ITERATIONS.
F4F6: 60                  RTS           RETURN.
F4F7: 10 F7     OVCHK     BPL  MD3      IF POSITIVE EXP THEN NO OVFL.
F4F9: 4C F5 03  OVFL      JMP  OVLOC
                          ORG  $F63D
F63D: 20 7D F4  FIX1      JSR  RTAR
F640: A5 F8     FIX       LDA  X1
F642: 10 13               BPL  UNDFL
F644: C9 8E               CMP  #$8E
F646: D0 F5               BNE  FIX1
F648: 24 F9               BIT  M1
F64A: 10 0A               BPL  FIXRTS
F64C: A5 FB               LDA  M1+2
F64E: F0 06               BEQ  FIXRTS
F650: E6 FA               INC  M1+1
F652: D0 02               BNE  FIXRTS
F654: E6 F9               INC  M1
F656: 60        FIXRTS    RTS
F657: A9 00     UNDFL     LDA  #$0
F659: 85 F9               STA  M1
F65B: 85 FA               STA  M1+1
F65D: 60                  RTS
```

```
                 **********************
                 *                    *
                 *   APPLE-II PSEUDO  *
                 * MACHINE INTERPRETER *
                 *                    *
                 *   COPYRIGHT 1977   *
                 * APPLE COMPUTER INC *
                 *                    *
                 * ALL RIGHTS RESERVED *
                 *     S. WOZNIAK     *
                 *                    *
                 **********************
                 TITLE "SWEET16 INTERPRETER"
                 R0L      EQU   $0
                 R0H      EQU   $1
                 R14H     EQU   $1D
                 R15L     EQU   $1E
                 R15H     EQU   $1F
                 SW16PAG  EQU   $F7
                 SAVE     EQU   $FF4A
                 RESTORE  EQU   $FF3F
                          ORG   $F689
F689: 20 4A FF   SW16     JSR   SAVE        PRESERVE 6502 REG CONTENTS
F68C: 68                  PLA
F68D: 85 1E               STA   R15L        INIT SWEET16 PC
F68F: 68                  PLA               FROM RETURN
F690: 85 1F               STA   R15H          ADDRESS
F692: 20 98 F6   SW16B    JSR   SW16C       INTERPRET AND EXECUTE
F695: 4C 92 F6            JMP   SW16B       ONE SWEET16 INSTR.
F698: E6 1E      SW16C    INC   R15L
F69A: D0 02               BNE   SW16D       INCR SWEET16 PC FOR FETCH
F69C: E6 1F               INC   R15H
F69E: A9 F7      SW16D    LDA   #SW16PAG
F6A0: 48                  PHA               PUSH ON STACK FOR RTS
F6A1: A0 00               LDY   #$0
F6A3: B1 1E               LDA   (R15L),Y    FETCH INSTR
F6A5: 29 0F               AND   #$F         MASK REG SPECIFICATION
F6A7: 0A                  ASL   A           DOUBLE FOR TWO BYTE REGISTERS
F6A8: AA                  TAX               TO X REG FOR INDEXING
F6A9: 4A                  LSR   A
F6AA: 51 1E               EOR   (R15L),Y    NOW HAVE OPCODE
F6AC: F0 0B               BEQ   TOBR        IF ZERO THEN NON-REG OP
F6AE: 86 1D               STX   R14H        INDICATE'PRIOR RESULT REG'
F6B0: 4A                  LSR   A
F6B1: 4A                  LSR   A           OPCODE*2 TO LSB'S
F6B2: 4A                  LSR   A
F6B3: A8                  TAY               TO Y REG FOR INDEXING
F6B4: B9 E1 F6            LDA   OPTBL-2,Y   LOW ORDER ADR BYTE
F6B7: 48                  PHA               ONTO STACK
F6B8: 60                  RTS               GOTO REG-OP ROUTINE
F6B9: E6 1E      TOBR     INC   R15L
F6BB: D0 02               BNE   TOBR2       INCR PC
F6BD: E6 1F               INC   R15H
F6BF: BD E4 F6   TOBR2    LDA   BRTBL,X     LOW ORDER ADR BYTE
F6C2: 48                  PHA               ONTO STACK FOR NON-REG OP
F6C3: A5 1D               LDA   R14H        'PRIOR RESULT REG' INDEX
F6C5: 4A                  LSR   A           PREPARE CARRY FOR BC, BNC.
F6C6: 60                  RTS               GOTO NON-REG OP ROUTINE
F6C7: 68         RTNZ     PLA               POP RETURN ADDRESS
F6C8: 68                  PLA
F6C9: 20 3F FF            JSR   RESTORE     RESTORE 6502 REG CONTENTS
F6CC: 6C 1E 00            JMP   (R15L)      RETURN TO 6502 CODE VIA PC
F6CF: B1 1E      SETZ     LDA   (R15L),Y    HIGH-ORDER BYTE OF CONSTANT
```

```
F6D1: 95 01            STA   R0H,X
F6D3: 88               DEY
F6D4: B1 1E            LDA   (R15L),Y   LOW-ORDER BYTE OF CONSTANT
F6D6: 95 00            STA   R0L,X
F6D8: 98               TYA              Y-REG CONTAINS 1
F6D9: 38               SEC
F6DA: 65 1E            ADC   R15L       ADD 2 TO PC
F6DC: 85 1E            STA   R15L
F6DE: 90 02            BCC   SET2
F6E0: E6 1F            INC   R15H
F6E2: 60        SET2   RTS
F6E3: 02        OPTBL  DFB   SET-1      1X
F6E4: F9        BRTBL  DFB   RTN-1      0
F6E5: 04               DFB   LD-1       2X
F6E6: 9D               DFB   BR-1       1
F6E7: 0D               DFB   ST-1       3X
F6E8: 9E               DFB   BNC-1      2
F6E9: 25               DFB   LDAT-1     4X
F6EA: AF               DFB   BC-1       3
F6EB: 16               DFB   STAT-1     5X
F6EC: B2               DFB   BP-1       4
F6ED: 47               DFB   LDDAT-1    6X
F6EE: B9               DFB   BM-1       5
F6EF: 51               DFB   STDAT-1    7X
F6F0: C0               DFB   BZ-1       6
F6F1: 2F               DFB   POP-1      8X
F6F2: C9               DFB   BNZ-1      7
F6F3: 5B               DFB   STPAT-1    9X
F6F4: D2               DFB   BM1-1      8
F6F5: 85               DFB   ADD-1      AX
F6F6: DD               DFB   BNM1-1     9
F6F7: 6E               DFB   SUB-1      BX
F6F8: 05               DFB   BK-1       A
F6F9: 33               DFB   POPD-1     CX
F6FA: E8               DFB   RS-1       B
F6FB: 70               DFB   CPR-1      DX
F6FC: 93               DFB   BS-1       C
F6FD: 1E               DFB   INR-1      EX
F6FE: E7               DFB   NUL-1      D
F6FF: 65               DFB   DCR-1      FX
F700: E7               DFB   NUL-1      E
F701: E7               DFB   NUL-1      UNUSED
F702: E7               DFB   NUL-1      F
F703: 10 CA     SET    BPL   SETZ       ALWAYS TAKEN
F705: B5 00     LD     LDA   R0L,X
                BK     EQU   *-1
F707: 85 00            STA   R0L
F709: B5 01            LDA   R0H,X      MOVE RX TO R0
F70B: 85 01            STA   R0H
F70D: 60               RTS
F70E: A5 00     ST     LDA   R0L
F710: 95 00            STA   R0L,X      MOVE R0 TO RX
F712: A5 01            LDA   R0H
F714: 95 01            STA   R0H,X
F716: 60               RTS
F717: A5 00     STAT   LDA   R0L
F719: 81 00     STAT2  STA   (R0L,X)    STORE BYTE INDIRECT
F71B: A0 00            LDY   #$0
F71D: 84 1D     STAT3  STY   R14H       INDICATE R0 IS RESULT NEG
F71F: F6 00     INR    INC   R0L,X
F721: D0 02            BNE   INR2       INCR RX
F723: F6 01            INC   R0H,X
F725: 60        INR2   RTS
F726: A1 00     LDAT   LDA   (R0L,X)    LOAD INDIRECT (RX)
F728: 85 00            STA   R0L        TO R0
F72A: A0 00            LDY   #$0
F72C: 84 01            STY   R0H        ZERO HIGH-ORDER R0 BYTE
F72E: F0 ED            BEQ   STAT3      ALWAYS TAKEN
F730: A0 00     POP    LDY   #$0        HIGH ORDER BYTE = 0
F732: F0 06            BEQ   POP2       ALWAYS TAKEN
F734: 20 66 F7  POPD   JSR   DCR        DECR RX
F737: A1 00            LDA   (R0L,X)    POP HIGH ORDER BYTE @RX
F739: A8               TAY              SAVE IN Y-REG
F73A: 20 66 F7  POP2   JSR   DCR        DECR RX
F73D: A1 00            LDA   (R0L,X)    LOW-ORDER BYTE
F73F: 85 00            STA   R0L        TO R0
F741: 84 01            STY   R0H
F743: A0 00     POP3   LDY   #$0        INDICATE R0 AS LAST RESULT REG
F745: 84 1D            STY   R14H
F747: 60               RTS
F748: 20 26 F7  LDDAT  JSR   LDAT       LOW-ORDER BYTE TO R0, INCR RX
F74B: A1 00            LDA   (R0L,X)    HIGH-ORDER BYTE TO R0
F74D: 85 01            STA   R0H
F74F: 4C 1F F7         JMP   INR        INCR RX
F752: 20 17 F7  STDAT  JSR   STAT       STORE INDIRECT LOW-ORDER
```

```
F755: A5 01             LDA   R0H         BYTE AND INCR RX.  THEN
F757: 81 00             STA   (R0L,X)     STORE HIGH-ORDER BYTE.
F759: 4C 1F F7          JMP   INR         INCR RX AND RETURN
F75C: 20 66 F7  STPAT   JSR   DCR         DECR RX
F75F: A5 00             LDA   R0L
F761: 81 00             STA   (R0L,X)     STORE R0 LOW BYTE @RX
F763: 4C 43 F7          JMP   POP3        INDICATE R0 AS LAST RSLT REG
F766: B5 00     DCR     LDA   R0L,X
F768: D0 02             BNE   DCR2        DECR RX
F76A: D6 01             DEC   R0H,X
F76C: D6 00     DCR2    DEC   R0L,X
F76E: 60                RTS
F76F: A0 00     SUB     LDY   #$0         RESULT TO R0
F771: 38        CPR     SEC               NOTE Y-REG = 13*2 FOR CPR
F772: A5 00             LDA   R0L
F774: F5 00             SBC   R0L,X
F776: 99 00 00          STA   R0L,Y       R0-RX TO RY
F779: A5 01             LDA   R0H
F77B: F5 01             SBC   R0H,X
F77D: 99 01 00  SUB2    STA   R0H,Y
F780: 98                TYA               LAST RESULT REG*2
F781: 69 00             ADC   #$0         CARRY TO LSB
F783: 85 1D             STA   R14H
F785: 60                RTS
F786: A5 00     ADD     LDA   R0L
F788: 75 00             ADC   R0L,X
F78A: 85 00             STA   R0L         R0+RX TO R0
F78C: A5 01             LDA   R0H
F78E: 75 01             ADC   R0H,X
F790: A0 00             LDY   #$0         R0 FOR RESULT
F792: F0 E9             BEQ   SUB2        FINISH ADD
F794: A5 1E     BS      LDA   R15L        NOTE X-REG IS 12*2!
F796: 20 19 F7          JSR   STAT2       PUSH LOW PC BYTE VIA R12
F799: A5 1F             LDA   R15H
F79B: 20 19 F7          JSR   STAT2       PUSH HIGH-ORDER PC BYTE
F79E: 18        BR      CLC
F79F: B0 0E     BNC     BCS   BNC2        NO CARRY TEST
F7A1: B1 1E     BR1     LDA   (R15L),Y    DISPLACEMENT BYTE
F7A3: 10 01             BPL   BR2
F7A5: 88                DEY
F7A6: 65 1E     BR2     ADC   R15L        ADD TO PC
F7A8: 85 1E             STA   R15L
F7AA: 98                TYA
F7AB: 65 1F             ADC   R15H
F7AD: 85 1F             STA   R15H
F7AF: 60        BNC2    RTS
F7B0: B0 EC     BC      BCS   BR
F7B2: 60                RTS
F7B3: 0A        BP      ASL   A           DOUBLE RESULT-REG INDEX
F7B4: AA                TAX               TO X REG FOR INDEXING
F7B5: B5 01             LDA   R0H,X       TEST FOR PLUS
F7B7: 10 E8             BPL   BR1         BRANCH IF SO
F7B9: 60                RTS
F7BA: 0A        BM      ASL   A           DOUBLE RESULT-REG INDEX
F7BB: AA                TAX
F7BC: B5 01             LDA   R0H,X       TEST FOR MINUS
F7BE: 30 E1             BMI   BR1
F7C0: 60                RTS
F7C1: 0A        BZ      ASL   A           DOUBLE RESULT-REG INDEX
F7C2: AA                TAX
F7C3: B5 00             LDA   R0L,X       TEST FOR ZERO
F7C5: 15 01             ORA   R0H,X       (BOTH BYTES)
F7C7: F0 D8             BEQ   BR1         BRANCH IF SO
F7C9: 60                RTS
F7CA: 0A        BNZ     ASL   A           DOUBLE RESULT-REG INDEX
F7CB: AA                TAX
F7CC: B5 00             LDA   R0L,X       TEST FOR NON-ZERO
F7CE: 15 01             ORA   R0H,X       (BOTH BYTES)
F7D0: D0 CF             BNE   BR1         BRANCH IF SO
F7D2: 60                RTS
F7D3: 0A        BM1     ASL   A           DOUBLE RESULT-REG INDEX
F7D4: AA                TAX
F7D5: B5 00             LDA   R0L,X       CHECK BOTH BYTES
F7D7: 35 01             AND   R0H,X       FOR $FF (MINUS 1)
F7D9: 49 FF             EOR   #$FF
F7DB: F0 C4             BEQ   BR1         BRANCH IF SO
F7DD: 60                RTS
F7DE: 0A        BNM1    ASL   A           DOUBLE RESULT-REG INDEX
F7DF: AA                TAX
F7E0: B5 00             LDA   R0L,X
F7E2: 35 01             AND   R0H,X       CHECK BOTH BYTES FOR NO $FF
F7E4: 49 FF             EOR   #$FF
F7E6: D0 B9             BNE   BR1         BRANCH IF NOT MINUS 1
F7E8: 60        NUL     RTS
F7E9: A2 18     RS      LDX   #$18        12*2 FOR R12 AS STACK POINTER
```

```
F7EB: 20 66 F7          JSR   DCR        DECR STACK POINTER
F7EE: A1 00             LDA   (R0L,X)    POP HIGH RETURN ADDRESS TO PC
F7F0: 85 1F             STA   R15H
F7F2: 20 66 F7          JSR   DCR        SAME FOR LOW-ORDER BYTE
F7F5: A1 00             LDA   (R0L,X)
F7F7: 85 1E             STA   R15L
F7F9: 60                RTS
F7FA: 4C C7 F6  RTN     JMP   RTNZ
```

6502 MICROPROCESSOR INSTRUCTIONS

ADC	Add Memory to Accumulator with Carry
AND	"AND" Memory with Accumulator
ASL	Shift Left One Bit (Memory or Accumulator)
BCC	Branch on Carry Clear
BCS	Branch on Carry Set
BEQ	Branch on Result Zero
BIT	Test Bits in Memory with Accumulator
BMI	Branch on Result Minus
BNE	Branch on Result not Zero
BPL	Branch on Result Plus
BRK	Force Break
BVC	Branch on Overflow Clear
BVS	Branch on Overflow Set
CLC	Clear Carry Flag
CLD	Clear Decimal Mode
CLI	Clear Interrupt Disable Bit
CLV	Clear Overflow Flag
CMP	Compare Memory and Accumulator
CPX	Compare Memory and Index X
CPY	Compare Memory and Index Y
DEC	Decrement Memory by One
DEX	Decrement index X by One
DEY	Decrement Index Y by One
EOR	"Exclusive-Or" Memory with Accumulator
INC	Increment Memory by One
INX	Increment Index X by One
INY	Increment Index `I by One
JMP	Jump to New Location
JSR	Jump to New Location Saving Return Address
LDA	Load Accumulator with Memory
LDX	Load Index X with Memory
LDY	Load Index Y with Memory
LSR	Shift Right one Bit (Memory or Accumulator)
NOP	No Operation
ORA	OR Memory with Accumulator
PHA	Push Accumulator on Stack
PHP	Push Processor Status on Stack
PLA	Pull Accumulator from Stack
PLP	Pull Processor Status from Slack
ROL	Rotate One Bit Left (Memory or Accumulator)
ROR	Rotate One Bit Right (Memory or Accumulator)
RTI	Return from Interrupt
RTS	Return from Subroutine
SBC	Subtract Memory from Accumulator with Borrow
SEC	Set Carry Flag
SED	Set Decimal Mode
SEI	Set Interrupt Disable Status
STA	Store Accumulator in Memory
STX	Store Index X in Memory
STY	Store Index Y in Memory
TAX	Transfer Accumulator to Index X
TAY	Transfer Accumulator to Index Y
TSX	Transfer Stack Pointer to Index X
TXA	Transfer Index X to Accumulator
TXS	Transfer Index X to Stack Pointer
TYA	Transfer Index Y to Accumulator

THE FOLLOWING NOTATION APPLIES TO THIS SUMMARY:

A	Accumulator
X,Y	Index Registers
M	Memory
$\bar{C}$	Borrow
P	Processor Status Register
S	Stack Pointer
✓	Change
–	No Change
+	Add
$\wedge$	Logical AND
-	Subtract
$\veebar$	Logical Exclusive OR
$\uparrow$	Transfer From Slack
$\downarrow$	Transfer To Stack
$\rightarrow$	Transfer To
$\leftarrow$	Transfer To
V	Logical OR
PC	Program Counter
PCH	Program Counter High
PCL	Program Counter low
OPER	Operand
#	Immediate Addressing Mode

FIGURE 1. ASL-SHIFT LEFT ONE BIT OPERATION

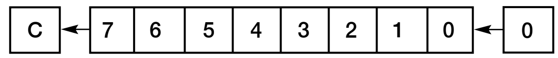


FIGURE 2 ROTATE ONE BIT LEFT (MEMORY OR ACCUMULATOR)

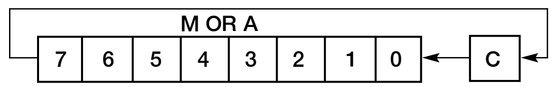
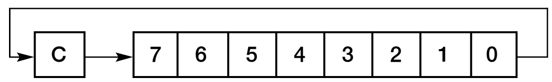


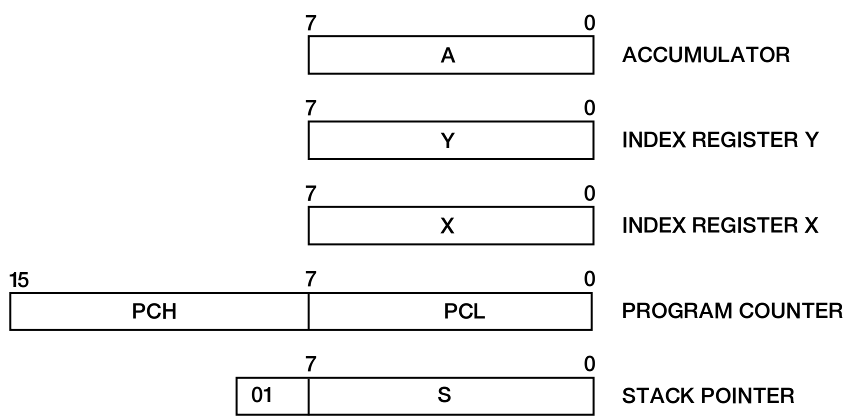
FIGURE 3.

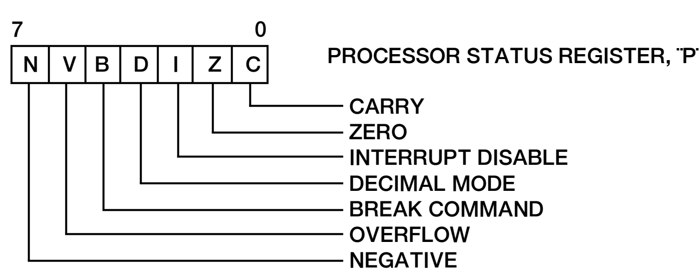


NOTE 1: BIT — TEST BITS

Bit 6 and 7 are Iranaterred to the status register. If the result of A $\wedge$ M is zero than Z=1, otherwise Z=0.

PROGRAMMING MODEL





INSTRUCTION CODES

Name Description	Operation	Addressing Mode	Assembly Language Form	HEX OP Code	No. Bytes	"P" Status Reg. N Z C I D V
ADC Add memory to accumulator with carry	A+M+C → A,C	Immediate Zero Page Zero Page,X Absolute Absolute,X Absolute,Y (Indirect,X) (Indirect),Y	ADC #Oper ADC Oper ADC Oper,X ADC Oper ADC Oper,X ADC Oper,Y ADC (Oper,X) ADC (Oper),Y	69 65 75 6D 7D 79 61 71	2 2 2 3 3 3 2 2	√√√– – √
AND "AND" memory with accumulator	A ∧ M → A	Immediate Zero Page Zero Page,X Absolute Absolute,X Absolute,Y (Indirect,X) (Indirect),Y	AND #Oper AND Oper AND Oper,X AND Oper AND Oper,X AND Oper,Y AND (Oper,X) AND (Oper),Y	29 25 35 2D 3D 39 21 31	2 2 2 3 3 3 2 2	√√– – – –
ASL Shift left one bit (Memory or Accumulator)	(See Figure 1)	Accumulator Zero Page Zero Page,X Absolute Absolute,X	ASL A ASL Oper ASL Oper,X ASL Oper ASL Oper,X	0A 06 16 0E 1E	1 2 2 3 3	√√√– – –
BCC Branch on carry clear	Branch on C=0	Relative	BCC Oper	90	2	– – – – – –
BCS Branch on carry set	Branch on C=1	Relative	BCS Oper	B0	2	– – – – – –
BEQ Branch on result zero	Branch on Z=1	Relative	BEQ Oper	F0	2	– – – – – –
BIT Test bits in memory with accumulator	A ∧ M, M_7 → N, M_6 → V	Zero Page Absolute	BIT* Oper BIT* Oper	24 2C	2 3	M_7√– – – M_6
BMI Branch on result minus	Branch on N=1	Relative	BMI Oper	30	2	– – – – – –
BNE Branch on result not zero	Branch on Z=0	Relative	BNE Oper	D0	2	– – – – – –
BPL Branch on result plus	Branch on N=0	Relative	BPL oper	10	2	– – – – – –
BRK Force Break	Forced Interrupt PC+2 ↓ P ↓	Implied	BRK*	00	1	– – – 1 – –
BVC Branch on overflow clear	Branch on V=0	Relative	BVC Oper	50	2	– – – – – –

Note 1 [illegible] 6 and 7 are transferred to the status register. If the result of A ∧ M is [illegible] then Z = 1, otherwise Z = 0

Note 2 A BRK command cannot be masked by setting I

Name Description	Operation	Addressing Mode	Assembly Language Form	HEX OP Code	No. Bytes	"P" Status Reg. N Z C I D V
BVS Branch on overflow set	Branch on V=1	Relative	BVS Oper	70	2	– – – – – –
CLC Clear carry flag	0 → C	Implied	CLC	18	1	– – 0 – – –
CLD Clear decimal mode	0 → D	Implied	CLD	D8	1	– – – – 0 –
CLI	0 → I	Implied	CLI	58	1	– – – 0 – –
CLV Clear overflow flag	0 → V	Implied	CLV	B8	1	– – – – – 0

CMP
Compare memory and accumulator
	A – M	Immediate
		Zero Page
		Zero Page, X
		Absolute
		Absolute,X
		Absolute,Y
		(Indirect,X)
		(Indirect),Y
			CMP #Oper
			CMP Oper
			CMP Oper,X
			CMP Oper
			CMP Oper,X
			CMP Oper,Y
			CMP (Oper,X)
			CMP (Oper),Y
				C9
				C5
				D5
				CD
				DD
				D9
				C1
				D1
					2
					2
					2
					3
					3
					3
					2
					2
						√√√– – –

CPX Compare memory and index X	X – M	Immediate Zero Page Absolute	CPX #Oper CPX Oper CPX Oper	E0 E4 EC	2 2 3	√√√– – –
CPY Compare memory and index Y	Y – M	Immediate Zero Page Absolute	CPY #Oper CPY Oper CPY Oper	C0 C4 CC	2 2 3	√√√– – –
DEC Decrement memory by one	M – 1 → M	Zero Page Zero Page,X Absolute Absolute,X	DEC Oper DEC Oper,X DEC Oper DEC Oper,X	C6 D6 CE DE	2 2 3 3	√√– – – –
DEX Decrement index X by one	X – 1 → X	Implied	DEX	CA	1	√√– – – –
DEY Decrement index Y by one	Y – 1 → Y	Implied	DEY	88	1	√√– – – –

INSTRUCTION CODES

Name Description	Operation	Addressing Mode	Assembly Language Form	HEX OP Code	No. Bytes	"P" Status Reg. N Z C I D V
EOR "Exclusive-Or" memory with accumulator	A V M → A	Immediate	EOR #Oper	49	2	√√————
		Zero Page	EOR Oper	45	2	
		Zero Page,X	EOR Oper,X	55	2	
		Absolute	EOR Oper	4D	3	
		Absolute,X	EOR Oper,X	5D	3	
		Absolute,Y	EOR Oper,Y	59	3	
		(Indirect,X)	EOR (Oper,X)	41	2	
		(Indirect),Y	EOR (Oper),Y	51	2	
INC Increment memory by one	M + 1 → M	Zero Page	INC Oper	E6	2	√√————
		Zero Page,X	INC Oper,X	F6	2	
		Absolute	INC Oper	EE	3	
		Absolute,X	INC Oper,X	FE	3	
INX Increment index X by one	X + 1 → X	Implied	INX	E8	1	√√————
INY Increment index Y by one	Y + 1 → Y	Implied	INY	C8	1	√√————
JMP Jump to new location	(PC+1) → PCL (PC+2) → PCH	Absolute	JMP Oper	4C	3	——————
		Indirect	JMP (Oper)	6C	3	
JSR Jump to new location saving return address	PC+2 ↓, (PC+1) → PCL (PC+2) → PCH	Absolute	JSR Oper	20	3	——————
LDA Load accumulator with memory	M → A	Immediate	LDA #Oper	A9	2	√√————
		Zero Page	LDA Oper	A5	2	
		Zero Page,X	LDA Oper,X	B5	2	
		Absolute	LDA Oper	AD	3	
		Absolute,X	LDA Oper,X	BD	3	
		Absolute,Y	LDA Oper,Y	B9	3	
		(Indirect,X)	LDA (Oper,X)	A1	2	
		(Indirect),Y	LDA (Oper),Y	B1	2	
LDX Load index X with memory	M → X	Immediate	LDX #Oper	A2	2	√√————
		Zero Page	LDX Oper	A6	2	
		Zero Page,Y	LDX Oper,Y	B6	2	
		Absolute	LDX Oper	AE	3	
		Absolute,Y	LDX Oper,Y	BE	3	
LDY Load index Y with memory	M → Y	Immediate	LDY #Oper	A0	2	√√————
		Zero Page	LDY Oper	A4	2	
		Zero Page,X	LDY Oper,X	B4	2	
		Absolute	LDY Oper	AC	3	
		Absolute,X	LDY Oper,X	BC	3	

Name Description	Operation	Addressing Mode	Assembly Language Form	HEX OP Code	No. Bytes	"P" Status Reg. N Z C I D V
LSR Shift right one bit (memory or accumulator)	(See Figure 1)	Accumulator	LSR A	4A	1	0√√———
		Zero Page	LSR Oper	46	2	
		Zero Page,X	LSR Oper,X	56	2	
		Absolute	LSR Oper	4E	3	
		Absolute,X	LSR Oper,X	5E	3	
NOP No operation	No Operation	Implied	NOP	EA	1	——————
ORA "OR" memory with accumulator	A V M → A	Immediate	ORA #Oper	09	2	√√————
		Zero Page	ORA Oper	05	2	
		Zero Page,X	ORA Oper,X	15	2	
		Absolute	ORA Oper	0D	3	
		Absolute,X	ORA Oper,X	1D	3	
		Absolute,Y	ORA Oper,Y	19	3	
		(Indirect,X)	ORA (Oper,X)	01	2	
		(Indirect),Y	ORA (Oper),Y	11	2	
PHA Push accumulator on stack	A ↓	Implied	PHA	48	1	——————
PHP Push processor status on stack	P ↓	Implied	PHP	08	1	——————
PLA Pull accumulator from stack	A ↑	Implied	PLA	68	1	√√————
PLP Pull processor status from stack	P ↑	Implied	PLP	28	1	From Stack
ROL Rotate one bit left (memory or accumulator)	(See Figure 2)	Accumulator	ROL A	2A	1	√√√———
		Zero Page	ROL Oper	26	2	
		Zero Page,X	ROL Oper,X	36	2	
		Absolute	ROL Oper	2E	3	
		Absolute,X	ROL Oper,X	3E	3	
ROR Rotate one bit right (memory or accumulator)	(See Figure 3)	Accumulator	ROR A	6A	1	√√√———
		Zero Page	ROR Oper	66	2	
		Zero Page,X	ROR Oper,X	76	2	
		Absolute	ROR Oper	6E	3	
		Absolute,X	ROR Oper,X	7E	3	

INSTRUCTION CODES

Name Description	Operation	Addressing Mode	Assembly Language Form	HEX OP Code	No. Bytes	"P" Status Reg. N Z C I D V
RTI Return from interrupt	P↑ PC↑	Implied	RTI	40	1	From Stack
RTS Return from subroutine	PC↑, PC+1 → PC	Implied	RTS		1	------
SBC Subtract memory from accumulator with borrow	A - M - $\bar{C}$ → A	Immediate Zero Page Zero Page,X Absolute Absolute,X Absolute,Y (Indirect,X) (Indirect),Y	SBC #Oper SBC Oper SBC Oper,X SBC Oper SBC Oper,X SBC Oper,Y SBC (Oper,X) SBC (Oper),Y	E9 E5 F5 ED FD F9 E1 F1	2 2 2 3 3 3 2 2	√√√--√
SEC Set carry flag	1 → C	Implied	SEC	38	1	--1---
SED Set decimal mode	1 → D	Implied	SED	F8	1	----1-
SEI Set interrupt disable status	1 → I	Implied	SEI	78	1	---1--
STA Store accumulator in memory	A → M	Zero Page Zero Page,X Absolute Absolute,X Absolute,Y (Indirect,X) (Indirect),Y	STA Oper STA Oper,X STA Oper STA Oper,X STA Oper,Y STA (Oper,X) STA (Oper),Y	85 95 8D 9D 99 81 91	2 2 3 3 3 2 2	------
STX Store index X in memory	X → M	Zero Page Zero Page,Y Absolute	STX Oper STX Oper,Y STX Oper	86 96 8E	2 2 3	------
STY Store index Y in memory	Y → M	Zero Page Zero Page,X Absolute	STY Oper STY Oper,X STY Oper	84 94 8C	2 2 3	------
TAX Transfer accumulator to index X	A → X	Implied	TAX	AA	1	√√----
TAY Transfer accumulator to index Y	A → Y	Implied	TAY	A8	1	√√----
TSX Transfer stack pointer to index X	S → X	Implied	TSX	BA	1	√√----

Name Description	Operation	Addressing Mode	Assembly Language Form	HEX OP Code	No. Bytes	"P" Status Reg. N Z C I D V
TXA Transfer index X to accumulator	X → A	Implied	TXA	8A	1	√√----
TXS Transfer index X to stack pointer	X → S	Implied	TXS	9A	1	------
TYA Transfer index Y to accumulator	Y → A	Implied	TYA	98	1	√√----

HEX OPERATION CODES

00 — BRK
01 — ORA — (Indirect. XI
02 — NOP
03 — NOR
04 — NOR
05 — ORA — Zero Page
06 — ASL — Zero Page
07 — NOP
08 — PHP
09 — ORA — Immediate
OA — ASL — Accumulator
OB — NOP
OC — NOP
OD — ORA — Absolute
OE --ASL --Absolute
OF — NOP
10 — BPL
11 — ORA — (Indirect), Y
12 — NOP
13 — NOP
14 — NOR
15 — ORA — Zero Page, X
16 — ASL — Zero Page. X
17 — NOR
18 — CLC
19 — ORA — Absolute, Y
IA — NOR
1B — NOP
1C —NOR
10 — ORA — Absolute, X
1E — ASL — Absolute. X
1F — NOP
20 — JSR
21 — AND —(Indirect, X)
22 — NOR
23 — NOP
24 — BIT — Zero Page
25 — AND — Zero Page
26 — ROL — Zero Page
27 — NOP
28 — PLP
29 — AND — Immediate
2A — ROL — Accumulator
2B — NOP
2C — BIT — Absolute
2D — AND — Absolute
2E — ROL — Absolute

2F — NOP
30 — BM!
31 — AND — (Indirect), V
32 — NOP
33 — NOP
34 — NOP
35 — AND — Zero Page, X
36 — ROL — Zero Page. X
37 — NOP
38 — SEC
39 — AND — Absolute, Y
3A — NOP
3B — NOP
3C — NOP
3D — AND — Absolute, X
3E — ROL — Absolute, X
3F — NOP
40 — RTI
41 — EOR — Indirect. X
42 — NOP
43 — NOP
44 — NOR
45 — EOR — Zero Page
46 — LSR — Zero Page
47 — NOP
48 — PHA
49 — EOR — Immediate
4A — LSR — Accumulator
4B —NOR
4C — JMP — Absolute
4D — EOR — Absolute
4E — LSR — Absolute
4F —MOP
50 — BVC
51 — EOR Indirect, Y
52 — NOP
53 — NOP
54 — NOP
55 — EOR — Zero Page, X
56 — LSR — Zero Page, X
57 — NOP
58 — CLI
59 — FOR-- Absolute, Y
5A — NOP
5B — NOP
5C — NOP
50 — EOR — Absolute, X

5E —LSR — Absolute, X
SF — NOP
60 — RTS
61 — ADC — Indirect, X
62 — NOR
63 — NOP
64 — NOR
65 — ADC — Zero Page
66 — ROR — Zero Page
67 — NOP
68 — PLA
69 — ADC — Immediate
6A — ROR — Accumulator
6B — NOP
6C — JMP — Indirect
6D — ADC — Absolute
6E — ROR — Absolute
6F — NOP
70 — BVS
71 — ADC — (Indirect), Y
72 — NOP
73 — MOP
74 — NOP
75 — ADC — Zero Page, X
76 — ROR — Zero Page. X
77 — NOP
78 — SEI
79 — ADC — Absolute, Y
7A — NOP
7B — NOP
7C — NOP
7D — ADC — Absolute, X NOP
7E — 808 — Absolute, X NOP
7F — NOP
80 — NOR
81 — STA — (Indirect, Xi
82 — NOP
83 — NOP
84 —STY — Zero Page
85 — STA — Zero Page
86 — STX — Zero Page
87 — NOP
88 — DEY
89 — NOP
8A — TXA
88 — NOP
8C — STY — Absolute

8D — STA — Absolute
BE — STX — Absolute
8F — NOP
90 — BCC
91 — STA — (Indirect), Y
92 — NOP
93 — NOR
94 — STY — Zero Page. X
95 — STA — Zero Page, X
96 — STX — Zero Page, Y
97 — NOP
98 — TVA
99 — STA — Absolute, Y
9A — TXS
9B — MOP
9C — NOP
9D — STA — Absolute, X
9E — NOP
9F — NOP
AO — LDY — Immediate
AI — LDA —(Indirect, XI
A2 —LOX — Immediate
A3 — NOR
A4 — LDY — Zero Page
AS — LDA — Zero Page
A6 — LDX — Zero Page
AI — NOP
A8 — TAY
A9 — LDA — Immediate
AA — TAX
AB — NOP
AC —LDY — Absolute
AD —Absolute
AE — LDX — Absolute
AF —NOR
BO — BCS
81 — LDA — (Indirect), Y
B2 — NOP
B3 — NOP

84 — LDY — Zero Page, X
85 — LDA — Zero Page, X
B6 — LOX — Zero Page, Y
87 — NOP
B8 — CLV
89 — LDA — Absolute. Y
BA — TSX
BB — NOP
BC — LDY — Absolute. X
BD — LDA — Absolute, X
BE — LOX — Absolute, Y
BF — NOP
CO — CPY — Immediate
C1 — CMP — (Indirect, X
C2 — NOP
C3 — NOP
C4 — CPY — Zero Page
C5 — CMP — Zero Page
C6 — DEC — Zero Page
C7 — NOP
C8 — INY
C9 — CMP — Immediate
CA — DEX
CB —MOP
CC —CPY — Absolute
CD —CMP — Absolute
CE — DEC DEC — Absolute
CF — NOP
DO — BNE
D1 — CMP — (Indirect), V
D2 — NOP
D3 — NOR
D4 — NOP
05 — CMP — Zero Page. X
D6 — DEC — Zero Page, X
07 —NOR
08 — CLD
D9 —CMP — Absolute. Y
DA — NOP

D8 — NOR
DC —MOP
DO —C CMP — Absolute X
DE — DEC — Absolute, X
OF — NOP
E0 — CPX — Immediate
EI — SBC — (Indirect, X)
E2 — NOP
E3 — NOP
E4 — CPX — Zero Page
E5 — SBC —Zero Page
E6 — INC—Zero Page
E7 — NOP
EB — INX
E9 — SBC — Immediate
EA — NOP
EB — NOP
EC — CPX — Absolute
ED — SBC — Absolute
EE — INC — Absolute
EE — NOP
FO — BM
F1 — SBC — (Indirect), Y
F2 — NOP
F3 — NOR
F4 — NOP
F5 — SBC — Zero Page, X
F6 — INC — Zero Page. X
F7 — NOP
F8 — SED
F9 — SBC — Absolute. Y
FA — NOP
FB — NOP
FC — NOP
FD — SBC — Absolute. X
FE — INC — Absolute, X
FF — NOP

APPLE II HARDWARE

GETTING STARTED WITH YOUR APPLE II BOARD

INTRODUCTION

ITEMS YOU WILL NEED:

Your APPLE II board comes completely assembled and thoroughly tested. You should have received the following:

a. 1 ea. APPLE II P.C. Board complete with specified RAM memory.

b. 1 ea. d.c. power connector with cable.

c. 1 ea. 2" speaker with cable.

d. 1 ea. Preliminary Manual

e. 1 ea. Demonstration cassette tapes. (For 4K: 1 cassette (2 programs); 16K or greater: 3 cassettes.

f. 2 ea. 16 pin headers plugged into locations A7 and J14.

In addition you will need:

g. A color TV set (or B & W) equipped with a direct video input connector for best performance or a commercially available RF modulator such as a "Pixi-verter"tm Higher channel (7-13) modulators generally provide better system performance than lower channel modulators (2-6).

h. The following power supplies (NOTE: current ratings do not include any capacity for peripheral boards.):

1. +12 Volts with the following current capacity!

   a. For 4K or 16K systems - 35ØmA.

   b. For 8K, 2ØK or 32K - 55ØmA.

   c. For 12K, 24K, 36K or 48K - 85ØmA.

2. +5 Volts at 1.6 amps

3. -5 Volts at 1ØmA.

4. OPTIONAL: If -12 Volts is reouired by your keyboard. (If using an APPLE II supplied keyboard, you will need -12V at 5ØmA.)

i. An audio cassette recorder such as a Panasonic model RQ-309 DS which is used to load and save programs.

j. An ASCII encoded keyboard equipped with a "reset" switch.

k. Cable for the following:

   1. Keyboard to APPLE II P.C.B.

   2. Video out 75 ohm cable to TV or modulator

   3. Cassette to APPLE II P.C.B. (1 or 2)

Optionally you may desire:

l. Game paddles or pots with cables to APPLE II Game I/O connector. (Several demo programs use PDL(0) and "Pong" also uses PDL(1).

m. Case to hold all the above

Final Assembly Steps

1. Using detailed information on pin functions in hardware section of manual, connect power supplies to d.c. cable assembly. Use both ground wires to miminize resistance. With cable assembly disconnected from APPLE II mother board, turn on power supplies and verify voltages on connector pins. Improper supply connections such as reverse polarity can severely damage your APPLE II.

2. Connect keyboard to APPLE II by unplugging leader in location A7 and wiring keyboard cable to it, then plug back into APPLE II P.C.B.

3. Plug in speaker cable.

4. Optionally connect one or two game paddles using leader supplied in socket located at J14.

5. Connect video cable.

6. Connect cable from cassette monitor output to APPLE II cassette input.

7. Check to see that APPLE II board is not contacting any conducting surface.

8. With power supplies turned off, plug in power connector to mother board then recheck all cableing.

POWER UP

1. Turn power on. If power supplies overload, immediately turn off and recheck power cable wiring. Verify operating supply voltages are within $\pm$3% of nominal value.

2. You should now have random video display. If not check video level pot on mother board, full clockwise is maximum video output. Also check video cables for opens and shorts. Check modulator if you are using one.

3. Press reset button. Speaker should beep and a "*" prompt character with a blinking cursor should appear in lower left on screen.

4. Press "esc" button, release and type a "@" (shift-P) to clear screen.. You may now try "Monitor" commands if you wish. See details in "Monitor" software section.

RUNNING BASIC

1. Turn power on; press reset button; type "control B" and press return button. A ">" prompt character should appear on screen indicating that you are now in BASIC.

2. Load one of the supplied demonstration cassettes into recorder. Set recorder level to approximately 5 and start recorder. Type "LOAD" and return. First beep indicates that APPLE II has found beginning of program; second indicates end of program followed by ">" character on screen. If error occurs on loading, try a different demo tape or try changing cassette volume level.

3. Type RUN and carriage return to execute demonstration program. Listings of these are included in the last section of this manual.

Switching power supplies generally have both advantages and peculiarities not generally found in conventional power supplies. The Apple II user is urged to review this section.

> Your Apple II is equipped with an AC line voltage filter and a three wire AC line cord. It is important to make sure that the third wine is returned to earth ground. Use a continuity checker or ohmmeter to ensure that the third wire is actually returned to earth. Continuity should be checked for between the power supply case and an available water pipe for example. The line filter, which is of a type approved by domestic (U.L. CSA) and international (VDE) agencies must be returned to earth to function properly and to avoid potential shock hazards.

The APPLE II power supply is of the "flyback" switching type. In this system, the AC line is rectified directly, "chopped up" by a high frequency oscillator and coupled through a small transformer to the diodes, filters, etc., and results in four low voltage DC supplies to run APPLE II. The transformer isolates the DC supplies from the line and is provided with several shields to prevent "hash" from being coupled into the logic or peripherals. In the "flyback" system, the energy transferred through from the AC line side to DC supply side is stored in the transformer's inductance on one-half of the operating cycle, then transferred to the output filter capacitors on the second half of the operating cycle. Similar systems are used in TV sets to provide horizontal deflection and the high voltages to run the CRT.

Regulation of the DC voltages is accomplished by controlling the frequency at which the converter operates; the greater the output power needed, the lower the frequency of the converter. If the converter is overloaded, the operating frequency will drop into the audible range with squeels and squawks warning the user that something is wrong.

All DC outputs are regulated at the same time and one of the four outputs (the +5 volt supply) is compared to a reference voltage with the difference error fed to a feedback loop to assist the oscillator in running at the needed frequency. Since all DC outputs are regulated together, their voltages will reflect to some extent unequal loadings.

For example; if the +5 supply is loaded very heavily, then all other supply voltages will increase in voltage slightly; conversely, very light loading on the +5 supply and heavy loading on the +12 supply will cause both it and the others to sag lightly. If precision reference voltages are needed for peripheral applications, they should be provided for in the peripheral design.

In general, the APPLE II design is conservative with respect to component ratings and operating termperatures. An over-voltage crowbar shutdown system and an auxilliary control feedback loop are provided to ensure that even very unlikely failure modes will not cause damage to theAPPLE II computer system. The over-voltage protection references to the DC output voltages only. The AC line voltage input must be within the specified limits, i.e., 1Ø7V to 132V.

> Under no circumstances, should more than 14Ø VAC be applied to the input of the power supply. Permanent damage will result.

Since the output voltages are controlled by changing the operating frequency of the converter, and since that frequency has an upper limit determined by the switching speed of power transistors, there then must be a minimum load on the supply; the Apple II board with minimum memory (4K) is well above that minimum load. However, with the board disconnected, there is no load on the supply, and the internal over-voltage protection circuitry causes the supply to turn off. A 9 watt load distributed roughly 50-50 between the +5 and +12 supply is the nominal minimum load.

Nominal load current ratios are: The +12V supply load is ½ that of the +5V.
The - 5V supply load is 1/1Ø that of the +5V.
The -12V supply load is 1/1Ø, that of the +5V.

The supply voltages are +5.Ø ± Ø.15 volts, +11.8 ± Ø.5 volts, -12.Ø ± 1V, -5.2 ± 0.5 volts. The tolerances are greatly reduced when the loads are close to nominal.

The Apple II power supply will power the Apple II board and all present and forthcoming plug-in cards, we recommend the use of low power TTL, CMOS, etc. so that the total power drawn is within the thermal limits of the entire system. In particular, the user should keep the total power drawn by any one card to less than 1.5 watts, and the total current drawn by all the cards together within the following limits:

+ 12V	-	use no more than 25Ø mA
+ 5V	-	use no more than 5ØØ mA
- 5V	-	use no more than 2ØØ mA
- 12V	-	use no more than 2ØØ mA

The power supply is allowed to run indefinetly under short circuit or open circuit conditions.

> CAUTION: There are dangerous high voltages inside the power supply case. Much of the internal circuitry is NOT isolated from the power line, and special equipment is needed for service. NO REPAIR BY THE USER IS ALLOWED.

NOTES ON INTERFACING WITH THE HOME TV

Accessories are available to aid the user in connecting the Apple II system to a home color TV with a minimum of trouble. These units are called "RF Modulators" and they generate a radio frequency signal corresponding to the carrier of one or two of the lower VHF television bands; 61.25 MHz (channel 3) or 67.25 MHz (channel 4). This RF signal is then modulated with the composite video signal generated by the Apple II.

Users report success with the following RF modulators:

the "PixieVerter" (a kit)
ATV Research
13th and Broadway
Dakota City, Nebraska 68731

the "TV-1" (a kit)
UHF Associates
6037 Haviland Ave.
Whittier, CA 90601

the "Sup-r-Mod" by (assembled & tested)
M&R Enterprises
P.O. Box 1011
Sunnyvale, CA 94088

the RF Modulator (a P.C. board)
Electronics Systems
P.O. Box 212
Burlingame, CA 94010

Most of the above are available through local computer stores.

The Apple II owner who wishes to use one of these RF Modulators should read the following notes carefully.

All these modulators have a free running transistor oscillator. The M&R Enterprises unit is pre-tuned to Channel 4. The PixieVerter and the TV-1 have tuning by means of a jumper on the P.C. board and a small trimmer capacitor. All these units have a residual FM which may cause trouble if the TV set in use has a IF pass band with excessive ripple. The unit from M&R has the least residual FM.

All the units except the M&R unit are kits to be built and tuned by the customer. All the kits are incomplete to some extent. The unit from Electronics Systems is just a printed circuit board with assembly instructions. The kits from UHF Associates and ATV do not have an RF cable or a shielded box or a balun transformer, or an antenna switch. The M&R unit is complete.

Some cautions are in order. The Apple II, by virtue of its color graphics capability, operates the TV set in a linear mode rather than the 100% contrast mode satisfactory for displaying text. For this reason, radio frequency interference (RFI) generated by a computer (or peripherals) will beat with the

carrier of the RF modulator to produce faint spurious background patterns (called "worms") This RFI "trash" must be of quite a low level if worms are to be prevented. In fact, these spurious beats must be 4Ø to 5Ødb below the signal level to reduce worms to an acceptable level. When it is remembered that only 2 to 6 mV (across 3ØØΩ, is presented to the VHF input of the TV set, then stray RFI getting into the TV must be less than 5ØØμV to obtain a clean picture. Therefore we recommend that a good, co-ax cable be used to carry the signal from any modulator to the TV set, such as RG/59u (with copper shield), Belden #8241 or an equivalent miniature type such as Belden #8218. We also recommend that the RF modulator been closed in a tight metal box (an unpainted die cast aluminum box such as Pomona #2428). Even with these precautions, some trouble may be encountered with worms, and can be greatly helped by threading the coax cable connecting the modulator to the TV set repeatedly through a Ferrite toroid core Apple Computer supplies these cores in a kit:along with a 4 circuit connector/cable assembly to match the auxilliary video connector found on the Apple II board. This kit has order number A2MØ1ØX. The M&R "Sup-r-Mod is supplied with a coax cable and toroids.

Any computer containing fast switching logic and high frequency clocks will radiate some 'radio frequency energy. Apple II is equipped with a good line filter and many other precautions have been taken to minimize radiated energy. The user is urged not to connect "antennas" to this computer; wires strung about carrying clocks and/data will act as antennas, and subsequent radiated energy may prove to be a nuisance.

Another caution concerns possible long term effects on the TV picture tube. Most home TV sets have "Brightness" and "Contrast" controls with a very wide range of adjustment. When an un-changing picture is displayed with high brightness for a long period ,a faint discoloration of the TV CRT may occur as an inverse pattern observable with the TV set turned off. This condition may be avoided by keeping the "Brightness" turned down slightly and "Contrast" moderate.

A SIMPLE SERIAL OUTPUT

The Apple II is equipped with a 16 pin DIP socket most frequently used to connect potentiometers, switches, etc. to the computer for paddle control and other game applications. This socket, located at J-14, has outputs available as well. With an appropriate machine language program, these output lines may be used to serialize data in a format suitable for a teletype. A suitable interface circuit must be built since the outputs are merely LSTTL and won't run a teletype without help. Several interface circuits are discussed below and the user may pick the one best suited to his needs.

The ASR - 33 Teletype

The ASR - 33 Teletype of recent vintage has a transistor circuit to drive its solenoids. This circuit is quite easy to interface to, since it is provided with its own power supply. (Figure 1a) It can be set up for a 20mA current loop and interfaced as follows (whether or not the teletype is strapped for full duplex or half duplex operation):

a) The yellow wire and purple wire should both go to terminal 9 of Terminal Strip X. If the purple wire is going to terminal 8, then remove it and relocate it at terminal 9. This is necessary to change from the 60mA current loop to the 20mA current loop.

b) Above Terminal Strip X is a connector socket identified as "2". Pin 8 is the input line + or high; Pin 7 is the input line - or low. This connector mates with a Molex receptacle model 1375 #Ø3-Ø9-2151 or #03-09-2153. Recommended terminals are Molex #Ø2-Ø9-2136. An alternate connection method is via spade lugs to Terminal Strip X, terminal 7 (the + input line) and 6 (the - input line).

c) The following circuit can be built on a 16 pin DIP component carrier and then plugged into the Apple's 16 pin socket found at J-14: (The junction of the 3.3k resistor and the transistor base lead is floating). Pins 16 and 9 are used as tie points as they are unconnected on the Apple board. (Figure 1a).

The "RS - 232 Interface"

For this interface to be legitimate, it is necessary to twice invert the signal appearing at J-14 pin 15 and have it swing more than 5 volts both above and below ground. The following circuit does that but requires that both +12 and -12 supplies be used. (Figure 2) Snipping off pins on the DIP-component carrier will allow the spare terminals to be used for tie points. The output ground connects to pin 7 of the DB-25 connector. The signal output connects to pin 3 of the DB-25 connector. The "protective" ground wire normally found on pin 1 of the DB-25 connector may be connected to the Apple's base plate if desired. Placing a #4 lug under one of the four power supply mounting screws is perhaps the simplest method. The +12 volt supply is easily found on the auxiliary Video connector (see Figure S-11 or Figure 7 of the manual). The -12 volt supply may be found at pin 33 of the peripheral connectors (see Figure 4) or at the power supply connector (see Figure 5 of the manual).

A Serial Out Machine Center Language Program

Once the appropriate circuit has been selected and constructed a machine language program is needed to drive the circuit. Figure 3 lists such a teletype output machine language routine. It can be used in conjunction with an Integer BASIC program that doesn't require page $300 hex of memory. This program resides in memory from $370 to $3E9. Columns three and four of the listing show the op-code used. To enter this program into the Apple II the following procedure is followed:

Entering Machine Language Program

1. Power up Apple II
2. Depress and release the "RESET" key. An asterick and flashing cursor should appear on the left hand side of the screen below the random text matrix.
3. Now type in the data from columns one, two and three for each line from $370 to 03E9. For example, type in "370: A9 82" and then depress and release the "RETURN" key. Then repeat this procedure for the data at $372 and on until you complete entering the program.

Executing this Program

1. From BASIC a CALL 880 ($370) will start the execution of this program. It will use the teletype or suitable 80 column printer as the primary output device.

2. PR#Ø will inactivate the printer transfering control back to the Video monitor as the primary output device.

3. In Monitor mode $37ØG activates the printer and hitting the "RESET" key exits the program.

Saving the Machine Language Program

After the machine language program has been entered and checked for accuracy it should, for convenience, be saved on tape - that is unless you prefer to enter it by keyboard every time you want to use it.

The way it is saved is as follows:

1. Insert a blank program cassette into the tape recorder and rewind it.

2. Hit the "RESET" key. The system should move into Monitor mode. An asterick "*" and flashing cursor should appear on the left-hand side of the screen.

3. Type in "37Ø.Ø3E9W 37Ø.Ø3E9W".

4. Start the tape recorder in record mode and depress the "RETURN" key.

5. When the program has been written to tape, the asterick and flashing cursor will reappear.

The Program

After entering, checking and saving the program perform the following procedure to get a feeling of how the program is used:

1. B^C (control B) into BASIC

2. Turn the teletype (printer on)

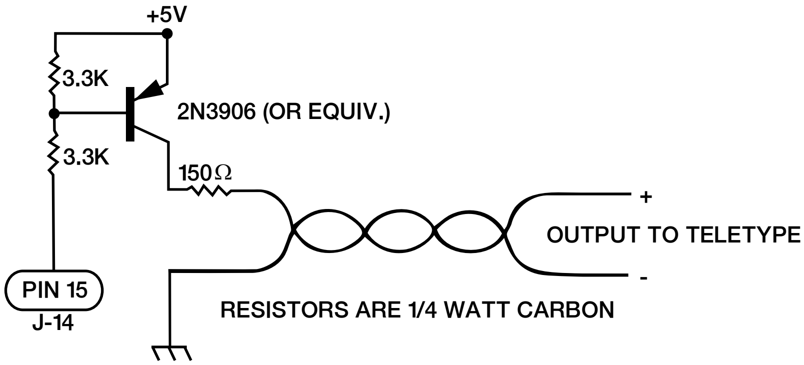
3. Type in the following
```
10 CALL 88Ø
15 PRINT "ABCD...XYZØ1123456789"
2Ø PR#Ø
25 END
```

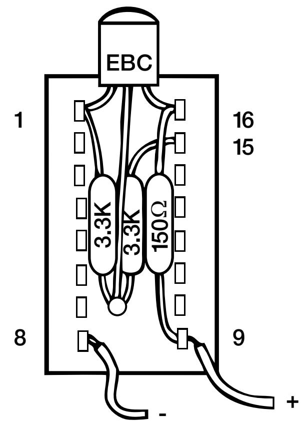
4. Type in RUN and hit the "RETURN" key. The text in line 15 should be printed on the teletype and control is returned to the keyboard and Video monitor

Line 1Ø activates the teletype machine routine and all "PRINT" statements following it will be printed to the teletype until a PR#Ø statement is encountered. Then the text in line 15 will appear on the teletype's output. Line 2Ø deactivates the printer and the program ends on line 25.

Conclusion

With the circuits and machine language program described in this paper the user may develop a relatively simple serial output interface to an ASR-3 or RS-232 compatible printers. This circuit can be activated through BASIC or monitor modes. And is a valuable addition to any users program library.





(a) (b)

FIGURE 2 ASR-33

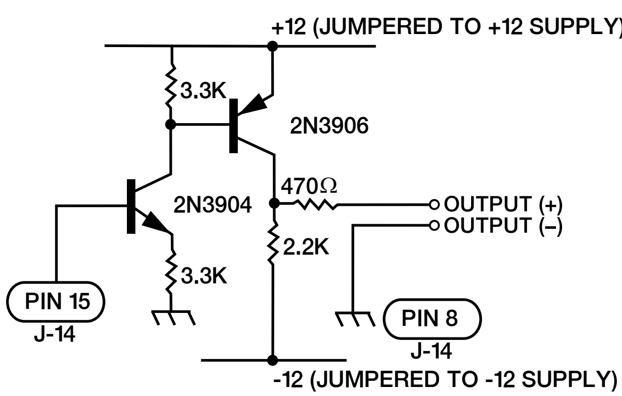


FIGURE 2 RS-232

```
                                                               PAGE: 1
                          TELETYPE DRIVER ROUTINES
   3:42 P.M., 11/18/1977
                     1   TITLE TELETYPE DRIVER ROUTINES'
                     2   ************************
                     3   *                      *
                     4   *       TTYDRIVER:     *
                     5   *     TELETYPE OUTPUT  *
                     6   *     ROUTINE FOR 72   *
                     7   *     COLUMN PRINT WITH*
                     8   *     BASIC LIST       *
                     9   *                      *
                    10   *   COPYRIGHT 1977 BY: *
                    11   *   APPLE COMPUTER INC.*
                    12   *         11/18/77     *
                    13   *                      *
                    14   *      R. WIGGINTON    *
                    15   *      S. WOZNIAK      *
                    16   *                      *
                    17   ************************
                    18   WNDWDTH  EQU   $21          ;FOR APPLE-II
                    19   CH       EQU   $24          ;CURSOR HORIZ.
                    20   CSWL     EQU   $36          ;CHAR. OUT SWITCH
                    21   YSAVE    EQU   $778
                    22   COLCNT   EQU   $7F8         ;COLUMN COUNT LOC.
                    23   MARK     EQU   $CO58
                    24   SPACE    EQU   $CO59
                    25   WAIT     EQU   $FCA8
                    26            ORG   $370
***WARNING: OPERAND OVERFLOW IN LINE 27
0370:  A9 82        27   TTINIT:  LDA   #TTOUT
0372:  85 36        28            STA   CSWL         ;POINT TO TTY ROUTINES
0374:  A9 03        29            LDA   #TTOUT/256 ;HIGH BYTE
0376:  85 37        30            STA   CSWL+1
0378:  A9 48        31            LDA   #72          ;SET WINDOW WIDTH
037A:  85 21        32            STA   WNDWDTH      ;TO NUMBER COLUMNS ONT
037C:  A5 24        33            LDA   CH
037E:  8D F8        34            STA   COLCNT       ;WHERE WE ARE NOW.
0381:  60           35            RTS
0382:  48           36   TTOUT:   PHA                ;SAVE TWICE
0383:  48           37            PHA                ;ON STACK.
0384:  AD F8        38   TTOUT2:  LDA   COLCNT       ;CHECK FOR A TAB.
0387:  C5 24        39            CMP   CH
0389:  68           40            PLA                ;RESTORE OUTPUT CHAR.
038A:  BO 03        41            BCS   TESTCTRL     ;IF C SET, NO TAB
038C:  48           42            PHA
038D:  A9 AO        43            LDA   #$A0         ;PRINT A SPACE.
038F:  2C CO        44   TESTCTRL:BIT   RTS1         ;TRICK TO DETERMINE
0392:  FO 03        45            BEQ   PRNTIT       ;IF CONTROL CHAR.
0394:  EE F8        46            INC   COLCNT       ;IF NOT, ADD ONE TO CM
0397:  20 C1        47   PRNTIT:  JSR   DOCHAR       ;PRINT THE CHAR ON TTY
039A:  68           48            PLA                ;RESTORE CHAR
0393:  48           49            PHA   TTOUT2       ;AND PUT BACK ON STAC
039C:  90 E6        50            BCC   #$OD         ;DO MORE SPACES FOR TA
039E:  49 OD        51            FOR   A            ;CHECK FOR CAR RET.
03A0:  OA           52            ASL   FINISH       ;ELIM PARITY
03A1:  DO OD        53            BNE                ;IF NOT CR, DONE.
```

FIGURE 3a

```
03A3:   8D F8 07 54          STA  COLCNT            ;CLEAR COLUMN COUNT
03A6:   A9 8A    55          LDA  #38A              ;NOW DO LINE FEED
03A8:   20 C1 03 56          JSR  DOCHAR
03AB:   A9 58    57          LDA  #153
03AD:   20 A8 FC 58          JSR  7AIT              ;200MSEC DELAY FOR LIB
0330:   AD F8 07 59          LDA  COLCNT            ;CHECK IF IN MARGIN
                    FINISH:
0333:   F0 08    60          3E0  SETCH             ;FOR CR, RESET CH
0335:   E5 21    61          S3C  7VD7DTH           ;IF SO, CARRY SET.
0337:   E9 F7    62          SSC  #SF7
0339:   90 04    63          BCC  RETURN
0393:   69 1F    64          ADC  #11F              ;ADJUST CH
033D:   85 24    65          STA  CH
                    SETCH:
033F:   68       66          PLA
                    RETURN:
03C0:   60       67          RTS                    ;RETURN TO CALLER
                    RTS1:
03C1:            68          STY  TELETYPE PRINT   A CHARACTER ROUTINE:
                    * HERE
03C4:   8C 78 07 69          PHP  YSAVE
                    DOCHAR:
03C5:   08       70          LDY                    ;SAVE STATUS.
03C7:   A0 08    71          CLC  #SOS              ;11 BITS (1 START, 1 2
03C3:   18       72          PHA                    ;BEGIN 7ITH SPACE (ST2
03C9:   48       73          3CS                    ;SAVE A REG AND SET FOI
                    TTOUT3:
03C3:   80 05    74          LDA  MARKOUT
03CE:   AD 59 C0 75          3CC  SPACE             ;SEND A SPACE
0300:   90 03    76          LDA  TTOUT4
0303:   AD 58 C0 77          LDA  MARK              ;SEND A MARK
                    MARKOUT:
0305:   A9 D7    78          PHA  #%D7              ;DELAY 9.091 MSEC FOR
                    TTOUT4:
0306:   48       79          LDA
                    DLY1:
03D8:   A9 20    80          LSR  #$20
                    DLY2:
0309:   4A       81          BCC  A
03D3:   90 FD    82          PLA  DLY2
03DC:   68       83          SBC
030E:   6A       84          3NE  #101
03E0:   88       85          PLA  DLY1
03E1:   D0 E3    86          ROR                    ;110 BAUD
03E2:   AC 78 07 87          DEY  A                 ;NEXT BIT (STOP BITS ?
03E3:   28       88          BNE                    LOOP 11 3ITS.
03E5:   60       89          LDY  TTOUT3
03E8:            90          PLP  YSAVE             ;RESTORE Y-REG.
03E9:            91          RTS                    ;RESTORE STATUS
                                                    ;RETURN
```

********SUCCESSFUL ASSEMBLY: NO ERRORS

FIGURE 3b

CROSS-REFERNCE:TELETYPE DRIVER ROUTINES

CH	0024	0033	0039	0065		
COLCNT	0718	0034	0038	0046	0054	0059
05YL	0036	0028	0030			
DLYI	0305	0085				
DLY2	0308	0082				
DOCHAR	0301	0047	0056			
FINISH	0330	0053				
MARK	CO58	0077				
MARKOUT	0300	0074				
PRNTIT	0397	0045				
RETURN	038F	0063				
RTS1	0300	0044				
SETCH	0330	0060				
SPACE	CO59	0075				
TESTCTRL	033F	0041				
TTINIT	0370					
TTOUT	0332	0027	0029			
TTOUT2	0384	0050				
TTOUT3	03C8	0089				
TTOUT4	0303	0076				
WAIT	FCAB	0058				
WNDWDTH	0021	0032	0061			
YSAVE	0778	0069	0090			

ILE:

FIGURE 3c

INTERFACING THE APPLE

This section defines the connections by which external devices are attached to the APPLE II board. Included are pin diagrams, signal descriptions, loading constraints and other useful information.

TABLE OF CONTENTS

Figure 1A APPLE II Board-Complete View

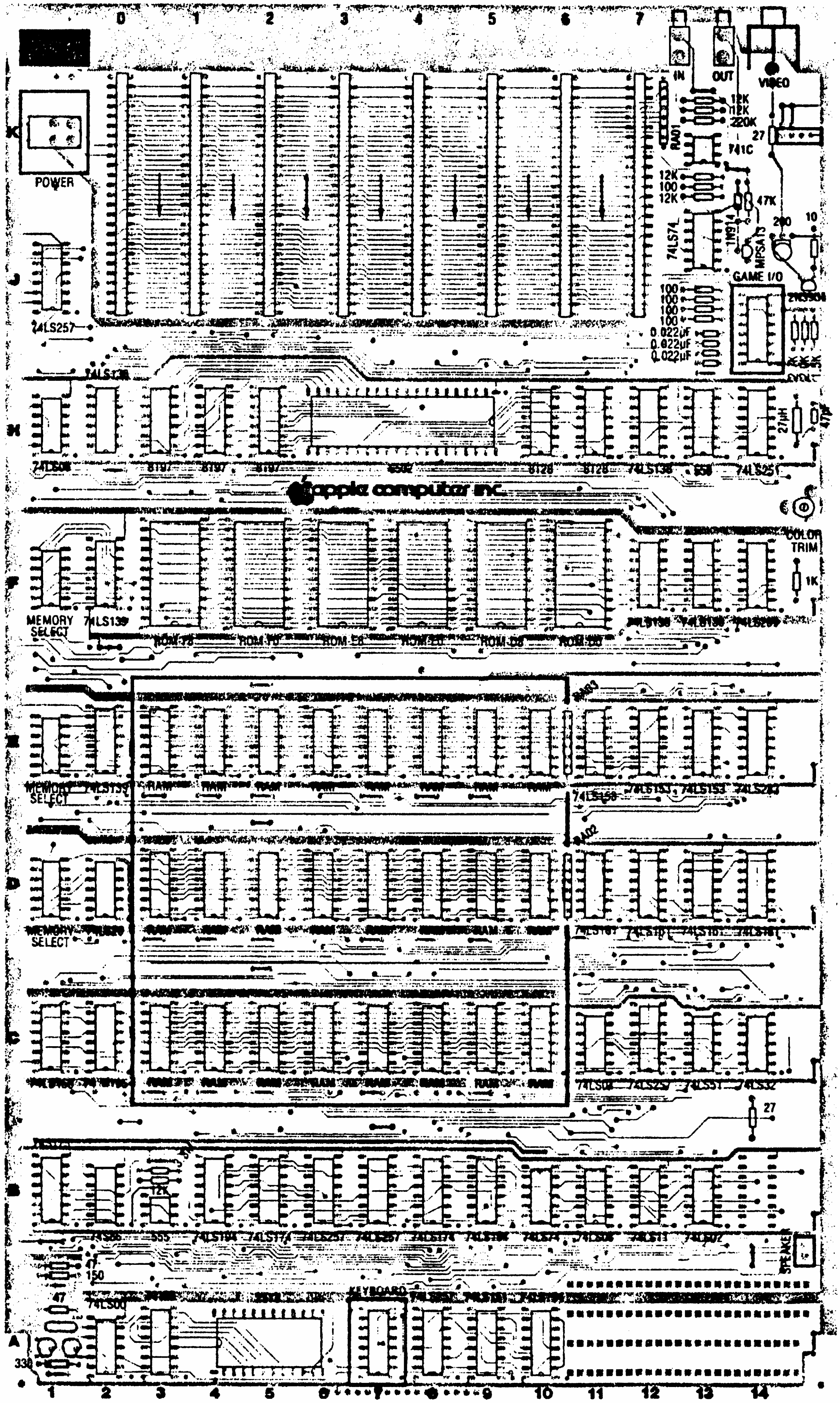
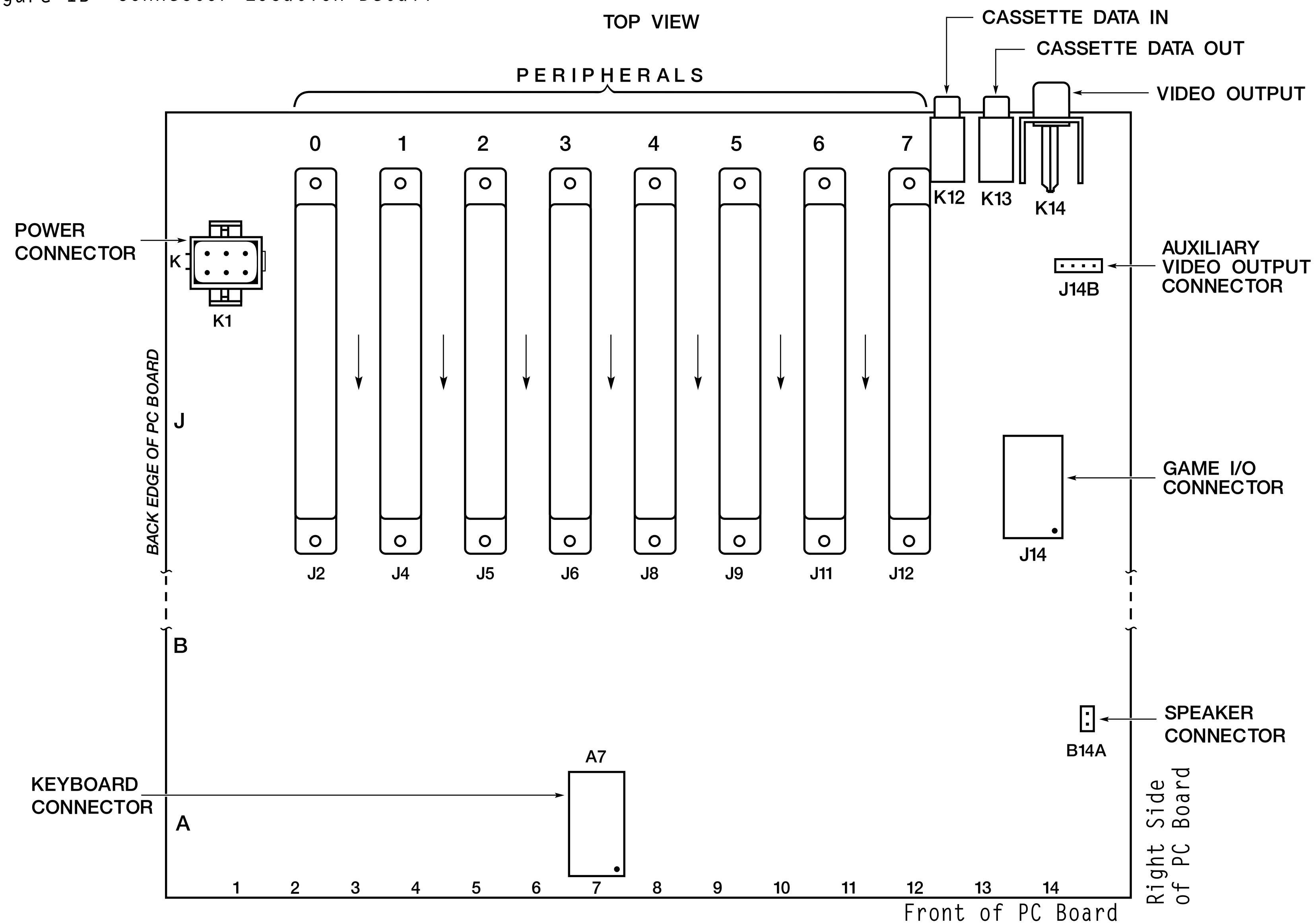


Figure 1B Connector Location Detail



CONNECTOR LOCATIONS

CASSETTE JACKS

A convenient means for interfacing an inexpensive audio cassette tape recorder to the APPLE II is provided by these two standard (3.5mm) miniature phone jacks located at the back of the APPLE II board.

CASSETTE DATA IN JACK: Designed for connection to the "EARPHONE" or "MONITOR" output found on most audio cassette tape recorders. V_{IN}=1Vpp (nominal), Z_{IN}=12K Ohms. Located at K12 as illustrated in Figure

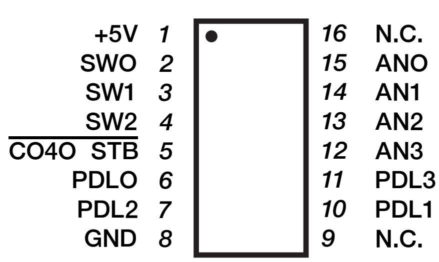
CASSETTE DATA OUT JACK: Designed for connection to the "MIC" or "MICROPHONE" input found on most audio cassette tape recorders. V_{OUT} =25 mV into 100 Ohms, Z_{OUT} =1ØØ Ohms. Located at K13 as illustrated in in Figure 1.

GAME I/O CONNECTOR

The Game I/O Connector provides a means for connecting paddle controls, lights and switches to the APPLE II for use in controlling video games, etc. It is a 16 pin IC socket located at J14 and is illustrated in Figure 1 and 2.

Figure 2 **GAME I/O CONNECTOR**

TOP VIEW
(Front Edge of PC Board)



LOCATION J14

SIGNAL DESCRIPTIONS FOR GAME I/O

AN0-AN3: 8 addresses (CØ58-CØ5F) are assigned to selectively "SET" or "CLEAR" these four "ANNUNCIATOR" outputs. Envisioned to control indicator lights, each is a 74LSxx series TTL output and must be buffered if used to drive lamps.

$\overline{\text{CØ4Ø STB}}$: A utility strobe output. Will go low during $Ø_2$ of a read or write cycle to addresses CØ4Ø-CØ4F. This is a 74LSxx series TTL output.

GND: System circuit ground. 0 Volt line from power supply.

NC: No connection.

PDLØ-PDL3: Paddle control inputs. Requires a Ø-15ØK ohm variable resistance and +5V for each paddle. Internal 1ØØ ohm resistors are provided in series with external pot to prevent excess current if pot goes completely to zero ohms.

SWØ-SW2: Switch inputs. Testable by reading from addresses CØ61-CØ63 (or CØ69-CØ6B). These are uncommitted 74LSxx series inputs.

+5V: Positive 5-Volt supply. To avoid burning out the connector pin, current drain MUST be less than 100mA.

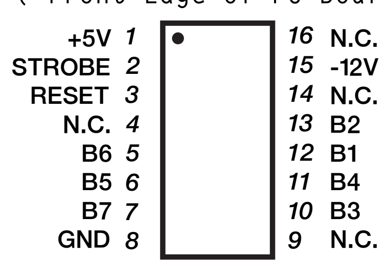
KEYBOARD CONNECTOR

This connector provides the means for connecting as ASCII keyboard to the APPLE II board. It is a 16 pin IC socket located at A7 and is illustrated in Figures 1 and 3.

Figure 3 **KEYBOARD CONNECTOR**

TOP VIEW

(Front Edge of PC Board)



LOCATION A7

SIGNAL DESCRIPTION FOR KEYBOARD INTERFACE

B1-B7: 7 bit ASCII data from keyboard, positive logic (high level= "1"), TTL logic levels expected.

GND: System circuit ground. Ø Volt line from power supply.

NC: No connection.

RESET: System reset input. Requires switch closure to ground.

STROBE: Strobe output from keyboard. The APPLE II recognizes the positive going edge of the incoming strobe.

+5V: Positive 5-Volt supply. To avoid burning out the connector pin, current drain MUST be less than 1ØØmA.

-12V: Negative 12-Volt supply. Keyboard should draw less than 50mA.

PERIPHERAL CONNECTORS

The eight Peripheral Connectors mounted near the back edge of the APPLE II board provide a convenient means of connecting expansion hardware and peripheral devices to the APPLE II I/O Bus. These are Winchester #2HW25CØ-111 (or equivalent) pin card edge connectors with pins on .1Ø" centers. Location and pin outs are illustrated in Figures 1 and 4.

SIGNAL DESCRIPTION FOR PERIPHERAL I/O

AØ-A15: 16 bit system address bus. Addresses are set up by the 65Ø2 within 3ØØnS after the beginning of $Ø_1$. These lines will drive up to a total of 16 standard TTL loads.

$\overline{\text{DEVICE SELECT}}$: Sixteen addresses are set aside for each peripheral connector. A read or write to such an address will send pin 41 on the selected connector low during $Ø_2$ (5ØØnS). Each will drive 4 standard TTL loads.

DØ-D7: 8 bit system data bus. During a write cycle data is set up by the 65Ø2 less than 3ØØnS after the beginning of $Ø_2$. During a read cycle the 65Ø2 expects data to be ready no less than 1ØØnS before the end of $Ø_2$. These lines will drive up to a total of 8 total low power schottky TTL loads.

DMA: Direct Memory Access control output. This line has a 3K Ohm pullup to +5V and should be driven with an open collector output.

DMA IN: Direct Memory Access daisy chain input from higher priority peripheral devices. Will present no more than 4 standard TTL loads to the driving device.

DMA OUT: Direct Memory Access daisy chain output to lower priority peripheral devices. This line will drive 4 standard TTL loads.

GND: System circuit ground. Ø Volt line from power supply.

INH: Inhibit Line.When a device pulls this line low, all ROM's on board are disabled (Hex addressed DØØØ through FFFF). This line has a 3K Ohm pullup to +5V and should be driven with an open collector output.

INT IN: Interrupt daisy chain input from higher priority peripheral devices. Will present no more than 4 standard TTL loads to the driving device.

INT OUT: Interrupt daisy chain output to lower priority peripheral devices. This line will drive 4 standard TTL loads.

I/O SELECT: 256 addresses are set aside for each peripheral connector (see address map in "MEMORY" section). A read or write of such an address will send pin 1 on the selected connector low during Ø2 (5ØØnS). This line will drive 4 standard TTL loads.

I/O STROBE: Pin 2Ø on all peripheral connectors will go low during $Ø_2$, of a read or write to any address C8ØØ-CFFF. This line will drive a total of 4 standard TTL loads.

IRQ: Interrupt request line to the 65Ø2. This line has a 3K Ohm pullup to +5V and should be driven with an open collector output.It is active low.

NC: No connection.

NMI: Non Maskable Interrupt request line to the 65Ø2. This line has a 3K Ohm pullup to +5V and should be driven with an open collector output.It is active low.

Q_3: A 1MHz (nonsymmetrical) general purpose timing signal. Will drive up to a total of 16 standard TTL loads.

RDY: "Ready" line to the 65Ø2. This line should change only during Ø1, and when low will halt the microprocessor at the next READ cycle. This line has a 3K Ohm pullup to +5V and should be driven with an open collector output.

RES: Reset line from "RESET" key on keyboard. Active low. Will drive 2 MOS loads per Peripheral Connector.

R/$\overline{W}$: READ/WRITE line from 65Ø2. When high indicates that a read cycle is in progress, and when low that a write cycle is in progress. This line will drive up to a total of 16 standard TTL loads.

USER 1: The function of this line will be described in a later document.

$Ø_0$: Microprocessor phase V clock. Will drive up to a total of 16 standard TTL loads.

$Ø_1$: Phase 1 clock, complement of $Ø_0$. Will drive up to a total of 16 standard TTL loads.

7M: Seven MHz high frequency clock. Will drive up to a total of 16 standard TTL loads.

+12V: Positive 12-Volt supply.

+5V: Positive 5-Volt supply

-5V: Negative 5-Volt supply.

-12V: Negative 12-Volt supply.

POWER CONNECTOR

The four voltages required by the APPLE II are supplied via this AMP #9-35Ø28-1,6 pin connector. See location and pin out in Figures 1 and 5.

PIN DESCRIPTION

GND: (2 pins) system circuit ground. Ø Volt line from power supply.

+12V: Positive 12-Volt line from power supply.

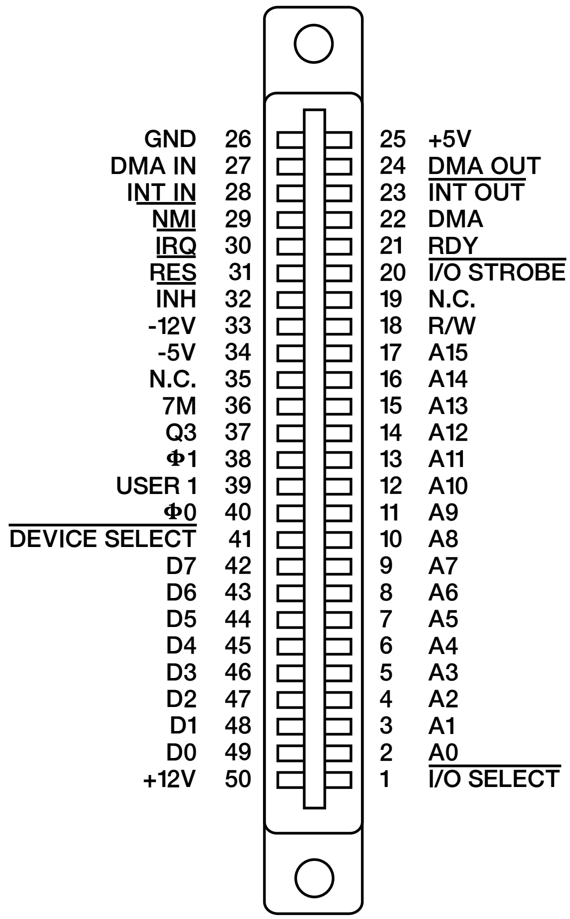
+5V: Positive 5-Volt line from power supply.

-5V: Negative 5-Volt line from power supply.

-12V: Negative 5-Volt line from power supply.

Figure 4 **PERIPHERAL CONNECTORS**
(EIGHT OF EACH)

PINOUT **TOP VIEW**
(Back Edge of PC Board)

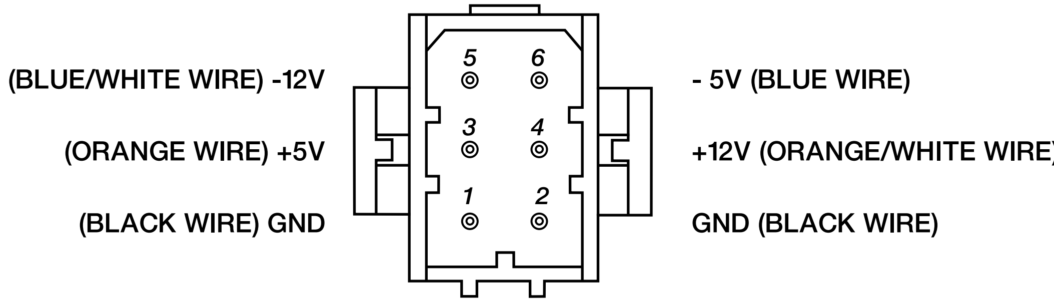


(Toward Front Edge of PC Board)

LOCATIONS J2 TO J12

Figure 5 **POWER CONNECTOR**

PINOUT **TOP VIEW**
(Toward Right side of PC Board)



LOCATION K1

SPEAKER CONNECTOR

This is a MOLEX KK 100 series connector with two .25" square pins on .10" centers. See location and pin out in Figures 1 and 6.

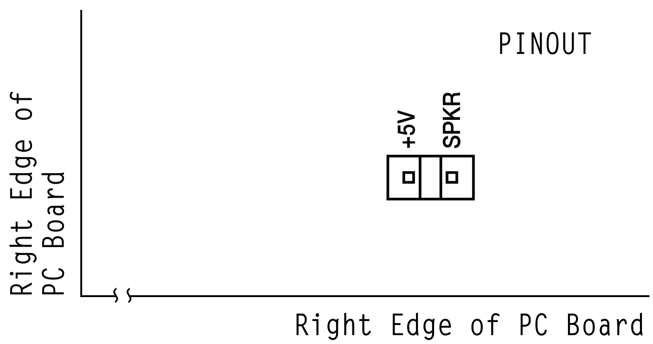
SIGNAL DESCRIPTION FOR SPEAKER

+5V: System +5 Volts

SPKR: Output line to speaker. Will deliver about .5 watt into 8 Ohms.

Figure 6

SPEAKER CONNECTIONS



LOCATION B14A

VIDEO OUTPUT JACK

This standard RCA phono jack located at the back edge of the APPLE II P.C. board will supply NTSC compatible, EIA standard, positive composite video to an external video monitor.

A video level control near the connector allows the output level to be adjusted from 0 to 1 Volt (peak) into an external 75 OHM load.

Additional tint (hue) range is provided by an adjustable trimmer capacitor.

See locations illustrated in Figure 1.

AUXILIARY VIDEO OUTPUT CONNECTOR

This is a MOLEX KK 100 series connector with four .25" square pins on .1Ø" centers. It provides composite video and two power supply voltages. Video out on this connector is not adjustable by the on board 200 Ohm trim pot. See Figures 1 and 7.

SIGNAL DESCRIPTION

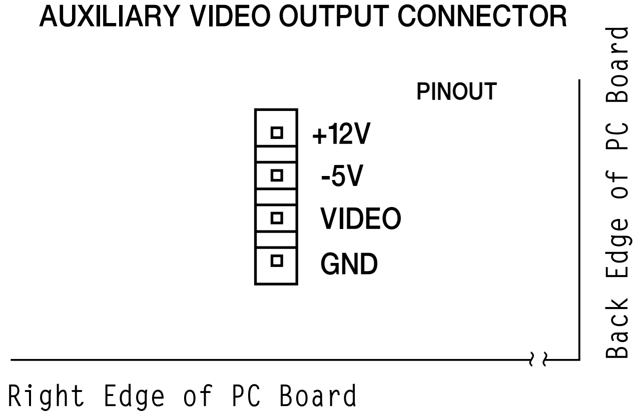
GND: System circuit ground. Ø Volt line from power supply.

VIDEO: NTSC compatible positive composite VIDEO. DC coupled emitter follower output (not short circuit protected). SYNC TIP is Ø Volts, black level is about .75 Volts, and white level is about 2.Ø Volts into 47Ø Ohms. Output level is non-adjustable.

+12V: +12 Volt line from power supply.

+5V: -5 Volt line from power supply.

Figure 7 AUXILIARY VIDEO OUTPUT CONNECTOR



LOCATION J14B

INSTALLING YOUR OWN RAM

THE POSSIBILITIES

The APPLE II computer is designed to use dynamic RAM chips organized as 4096 x 1 bit, or 16384 x 1 bit called "4K" and "16K" RAMs respectively. These must be used in sets of 8 to match the system data bus (which is 8 bits wide) and are organized into rows of 8. Thus, each row may contain either 4096 (4K) or 16384 (16K) locations of Random Access Memory depending upon whether 4K or 16K chips are used. If all three rows on the APPLE II board are filled with 4K RAM chips, then 12288 (12K) memory locations will be available for storing programs or data, and if all three rows contain 16K RAM chips then 49152 (commonly called 48K) locations of RAM memory will exist on board!

RESTRICTIONS

It is quite possible to have the three rows of RAM sockets filled with any combination of 4K RAMs, 16K RAMs or empty as long as certain rules are followed:

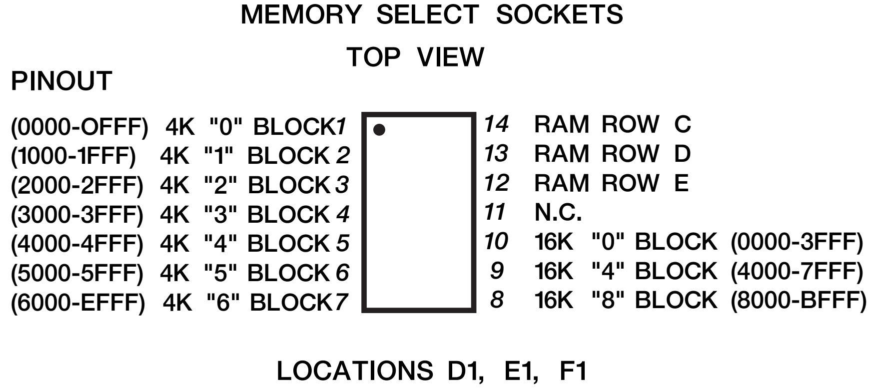
1. All sockets in a row must have the same type (4K or 16K) RAMs.

2. There <u>MUST</u> be RAM assigned to the zero block of addresses.

ASSIGNING RAM

The APPLE II has 48K addresses available for assignment of RAM memory. Since RAM can be installed in increments as small as 4K, a means of selecting which address range each row of memory chips will respond to has been provided by the inclusion of three MEMORY SELECT sockets on board.

Figure 8



MEMORY

TABLE OF CONTENTS

INTRODUCTION

APPLE II is supplied completely tested with the specified amount of RAM memory and correct memory select jumpers. There are five different sets of standard memory jumper blocks:

1. 4K 4K 4K BASIC
2. 4K 4K 4K HIRES
3. 16K 4K 4K
4. 16K 16K 4K
5. 16K 16K 16K

A set of three each of one of the above is supplied with the board. Type 1 is supplied with 4K or 8K systems. Both type 1 and 2 are supplied with 12K systems. Type 1 is a contiguous memory range for maximum BASIC program size. Type 2 is non-contiguous and allows 8K dedicated to HIRES screen memory with approximately 2K of user BASIC space. Type 3 is supplied with 16K, 2ØK and 24K systems. Type 4 with 30K and 36K systems and type 5 with 48K systems.

Additional memory may easily be added just by plugging into sockets along with correct memory jumper blocks.

The 65Ø2 microprocessor generates a 16 bit address, which allows 65536 (commonly called 65K) different memory locations to be specified. For convenience we represent each 16 bit (binary) address as a 4-digit hexadecimal number. Hexadecimal notation (hex) is explained in the Monitor section of this manual.

In the APPLE II, certain address ranges have been assigned to RAM memory, ROM memory, the I/O bus, and hardware functions. The memory and address maps give the details.

MEMORY SELECT SOCKETS

The location and pin out for memory select sockets are illustrated in Figures 1 and 8.

HOW TO USE

There are three MEMORY SELECT sockets, Thcated at D1, E1 and F1 respectively. RAM memory is assigned to various address ranges by inserting jumper wires as described below. All three MEMORY SELECT sockets MUST be jumpered identically! The easiest way to do this is to use Apple supplied memory blocks.

Let us learn by example:

If you have plugged 16K RAMs into row "C" (the sockets located at C3-C1Ø on the board), and you want them to occupy the first 16K of addresses starting at ØØØØ, jumper pin 14 to pin 1Ø on all three MEMORY SELECT sockets (thereby assigning row "C" to the ØØØØ-3FFF range of memory).

If in addition you have inserted 4K RAMs into rows "D" and "E", and you want them each to occupy the first 4K addresses starting at 4ØØØ and 5ØØØ respectively, jumper pin 13 to pin 5 (thereby assigning row "D" to the 4ØØØ-4FFF range of memory), and jumper pin 12 to pin 6 (thereby assigning row "E" to the 5ØØØ-5FFF range of memory). Remember to jumper all three MEMORY SELECT sockets the same.

Now you have a large contiguous range of addresses filled with RAM memory. This is the 24K addresses from ØØØØ-5FFF.

By following the above examples you should be able to assign each row of RAM to any address range allowed on the MEMORY SELECT sockets. Remember that to do this properly you must know three things:

1. Which rows have RAM installed?
2. Which address ranges do you want them to occupy?
3. Jumper all three MEMORY SELECT sockets the same!

If you are not sure think carefully, essentially all the necessary information is given above.

Memory Address Allocations in 4K Bytes

Address	Allocation
0000	text and color graphics display pages, 6502 stack, pointers, etc.
1000	
2000	high res graphics display primary page " " "
3000	" " "
4000	high res. graphics display secondary page " " "
5000	" " "
6000	
7000	

Address	Allocation
8000	
9000	
A000	
B000	
C000	addresses dedicated to hardware functions " "
D000	ROM socket D0: spare " ROM socket D8: spare "
E000	ROM socket E0: BASIC " ROM socket E8: BASIC "
F000	ROM socket F0: BASIC utility ROM socket F8: monitor

Memory Map Pages Ø to BFF

HEX ADDRESS(ES)	USED BY	USED FOR	COMMENTS
PAGE ZERO 0000-001F	UTILITY	register area for "sweet 16" 16 bit firmware processor.	
0020-004D	MONITOR		
004E-004F	MONITOR	holds a 16 bit number that is randomized with each key entry.	
0050-0055	UTILITY	integer multiply and divide work space.	
0055-00FF	BASIC		
00F0- 00FF	UTILITY	floating point work space.	
PAGE ONE 0100-01FF	6502	subroutine return stack.	
PAGE TWO 0200-02FF		character input buffer.	
PAGE THREE 03F8	MONITOR	Y_c (control Y) will cause a JSR to this location.	
03FB		NMI's are vectored to this location.	
03FE-03FF		IRQ's are vectored to the address pointed to by these locations.	
0400-07FF	DISPLAY	text or color graphics primary page.	
0800-0BFF	DISPLAY	text or color graphics secondary page.	BASIC initializes LOMEM to location 0800.

HEX ADDRESS	ASSIGNED FUNCTION	COMMENTS
C00X	Keyboard input.	Keyboard strobe appears in bit 7. ASCII data from keyboard appears in the 7 lower bits.
C01X	Clear keyboard strobe.	
C02X	Toggle cassette output.	
C03X	Toggle speaker output.	
C04X	"$\overline{\text{C040 STB}}$"	Output strobe to Game I/O connector.
C050	Set graphics mode	
C051	" text "	
C052	Set bottom 4 lines graphics	
C053	" " " " text	
C054	Display primary page	
C055	" secondary page	
C056	Set high res. graphics	
C057	" color "	
C058	Clear "AN0"	Annunciator 0 output to Game I/O connector.
C059	Set "	
C05A	Clear "AN1"	Annunciator 1 output to Game I/O connector.
C05B	Set "	
C05C	Clear "AN2"	Annunciator 2 output to Game I/O connector.
C05D	Set "	
C05E	Clear "AN3"	Annunciator 3 output to Game I/O connector.
C05F	Set "	

HEX ADDRESS	ASSIGNED FUNCTION	COMMENTS
C060/8	Cassette input	State of "Cassette Data In" appears in bit 7.
C061/9	"SW1"	State of Switch 1 input on Game I/O connector appears in bit 7.
C062/A	"SW2"	State of Switch 2 input on Game I/O connector appears in bit 7.
C063/B	"SW3"	State of Switch 3 input on Game I/O connector appears in bit 7.
C064/C	Paddle 0 timer output	State of timer output for Paddle 0 appears in bit 7.
C065/D	″ 1 ″ ″	State of timer output for Paddle 1 appears in bit 7.
C066/E	″ 2 ″ ″	State of timer output for Paddle 2 appears in bit 7.
C067/F	″ 3 ″ ″	State of timer output for Paddle 3 appears in bit 7.
C07X	"$\overline{\text{PDL STB}}$"	Triggers paddle timers during ϕ_2.
C08X	$\overline{\text{DEVICE SELECT}}$ 0	Pin 41 on the selected Peripheral Connector goes low during ϕ_2.
C09X	″ 1	
C0AX	″ 2	
C0BX	″ 3	
C0CX	″ 4	
C0DX	″ 5	
C0EX	″ 6	
C0FX	″ 7	
C10X	″ 8	Expansion connectors.
C11X	″ 9	″
C12X	″ A	″

HEX ADDRESS	ASSIGNED FUNCTION	COMMENTS
C13X	$\overline{\text{DEVICE SELECT}}$ B	"
C14X	" C	"
C15X	" D	"
C16X	" E	"
C17X	" F	"
C1XX	$\overline{\text{I/O SELECT}}$ 1	Pin 1 on the selected Peripheral Connector goes low during ϕ_2. NOTES: 1. Peripheral Connector 0 does not get this signal. 2. $\overline{\text{I/O SELECT}}$ 1 uses the same addresses as $\overline{\text{DEVICE SELECT}}$ 8-F.
C2XX	" 2	
C3XX	" 3	
C4XX	" 4	
C5XX	" 5	
C6XX	" 6	
C7XX	" 7	
C8XX	" 8, $\overline{\text{I/O STROBE}}$	Expansion connectors.
C9XX	" 9, "	
CAXX	" A, "	
CBXX	" B, "	
CCXX	" C, "	
CDXX	" D, "	
CEXX	" E, "	
CFXX	" F, "	
D000-D7FF	ROM socket D0	Spare.
D800-DFFF	" " D8	Spare.
E000-E7FF	" " E0	BASIC.
E800-EFFF	" " E8	BASIC.
F000-F7FF	" " F0	1K of BASIC, 1K of utility.
F800-FFFF	" " F8	Monitor.

SYSTEM TIMING

SIGNAL DESCRIPTIONS

14M: Master oscillator output, 14.318 MHz +/- 35 ppm. All other timing signals are derived from this one.

7M: Intermediate timing signal, 7.159 MHz.

COLOR REF: Color reference frequency used by video circuitry, 3.580 MHz.

$Ø_0$: Phase Ø clock to microprocessor, 1.Ø23 MHz nominal.

$Ø_1$: Microprocessor phase 1 clock, complement of $Ø_0$, 1.023 Mhz nominal.

$Ø_2$: Same as $Ø_0$. Included here because the 6502 hardware and programming manuals use the designation $Ø_2$ instead of $Ø_0$.

Q3: A general purpose timing signal which occurs at the same rate as the microprocessor clocks but is nonsymmetrical.

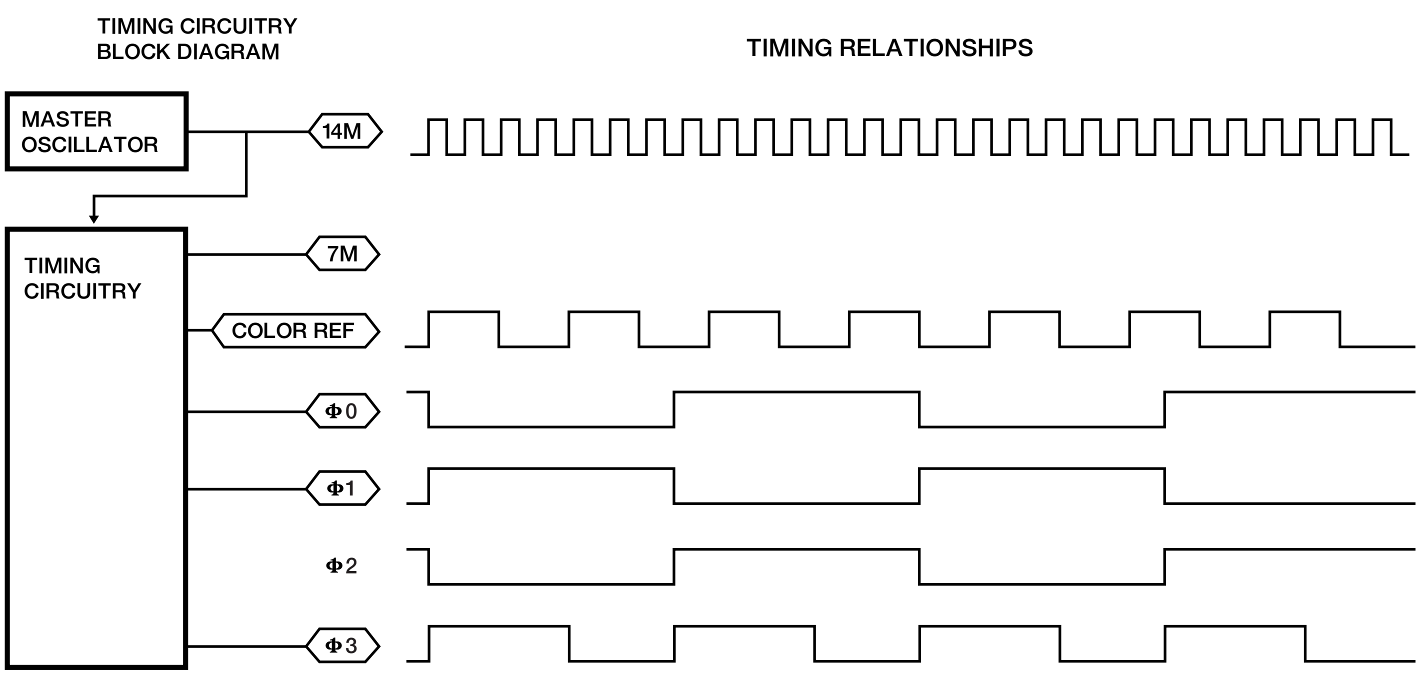
MICROPROCESSOR OPERATIONS

ADDRESS: The address from the microprocessor changes during $Ø_1$, and is stable about 300nS after the start of $Ø_1$.

DATA WRITE: During a write cycle, data from the microprocessor appears on the data bus during $Ø_2$, and is stable about 3ØØnS after the start of $Ø_2$.

DATA READ: During a read cycle, the microprocessor will expect data to appear on the data bus no less than 100nS prior to the end of $Ø_2$.

SYSTEM TIMING DIAGRAM



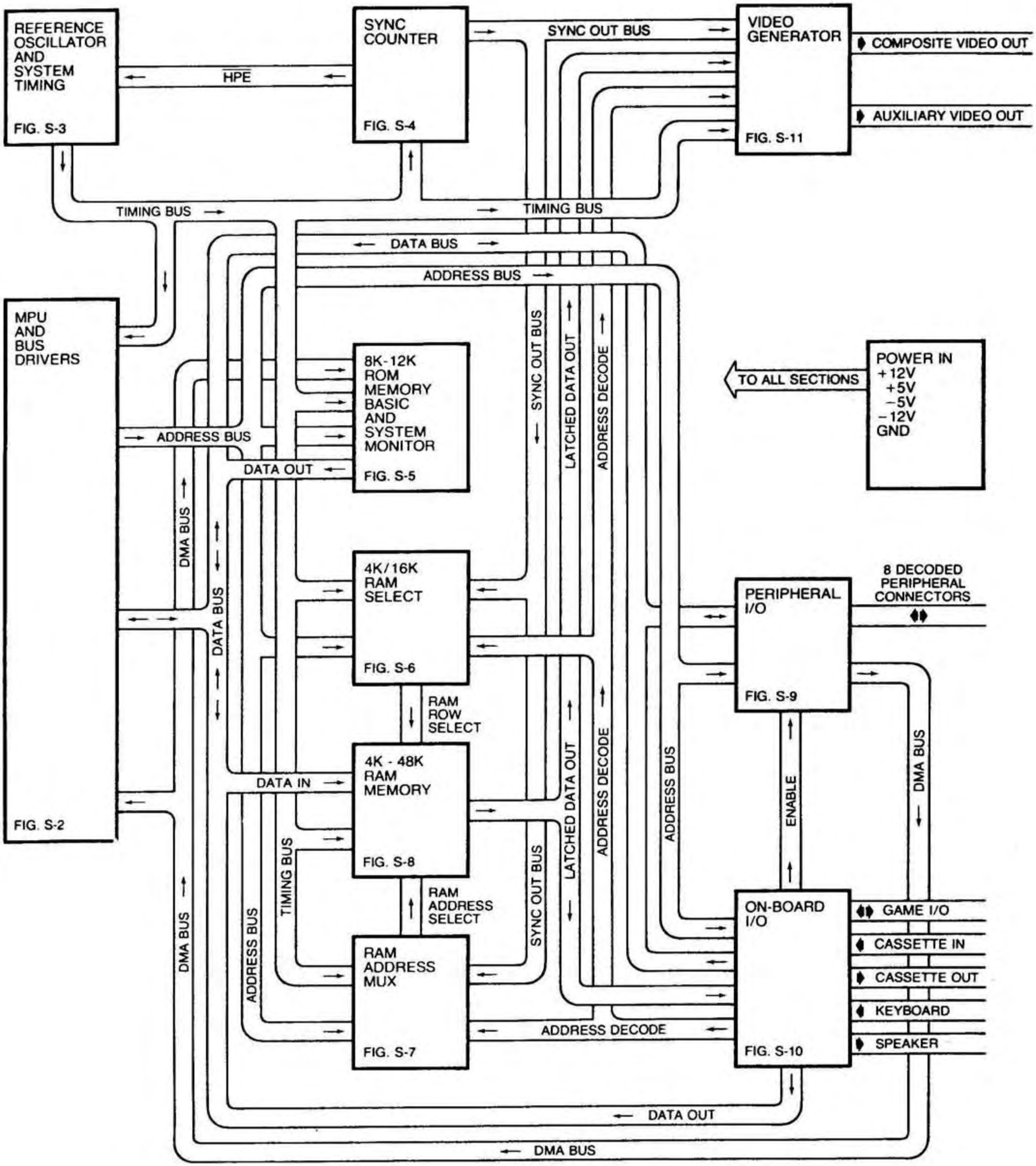


FIGURE S-1 APPLE II SYSTEM DIAGRAM

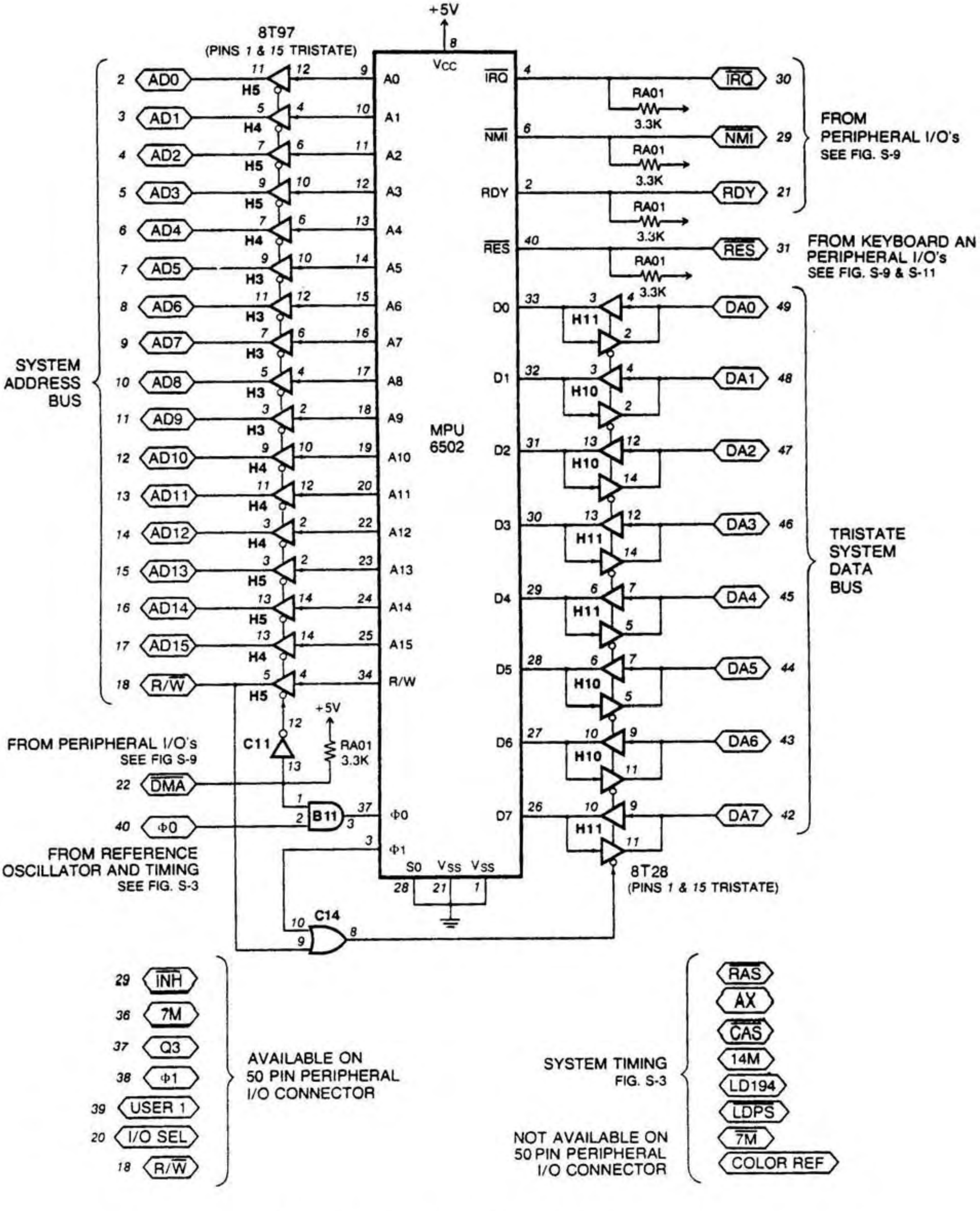


FIGURE S-2 MPU AND SYSTEM BUS

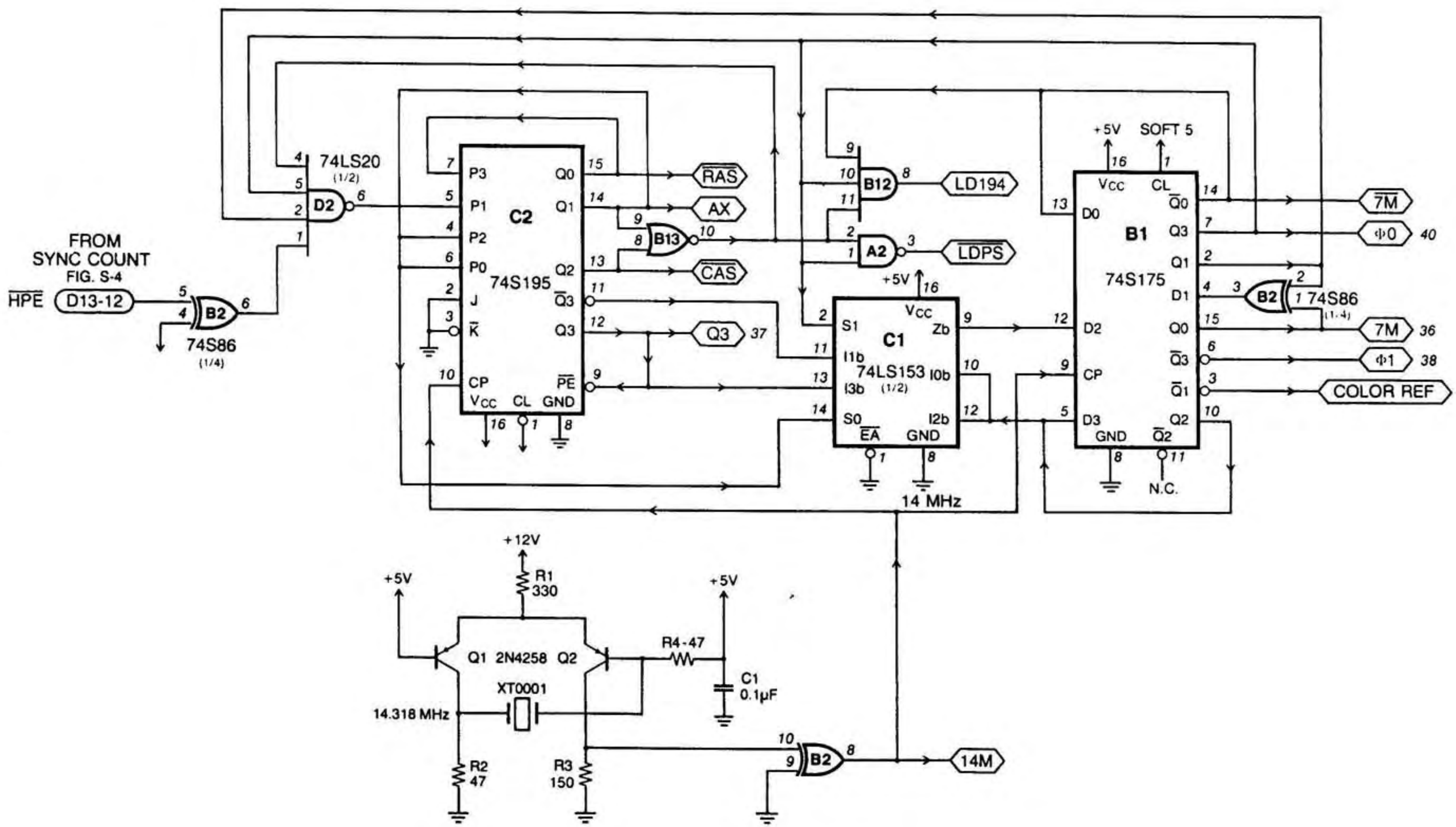


FIGURE S-3 REFERENCE OSCILLATOR AND SYSTEM TIMING

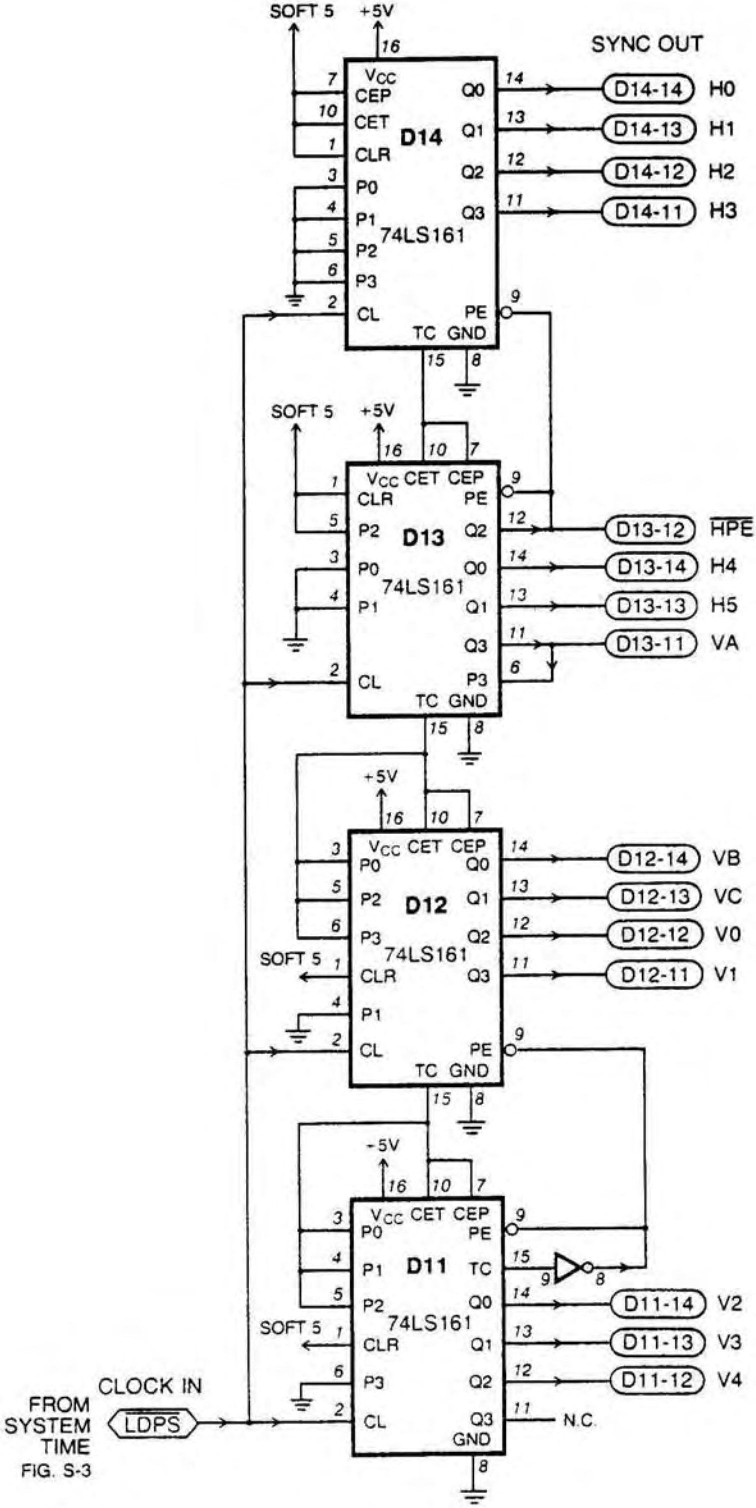


FIGURE S-4 SYNC COUNTER

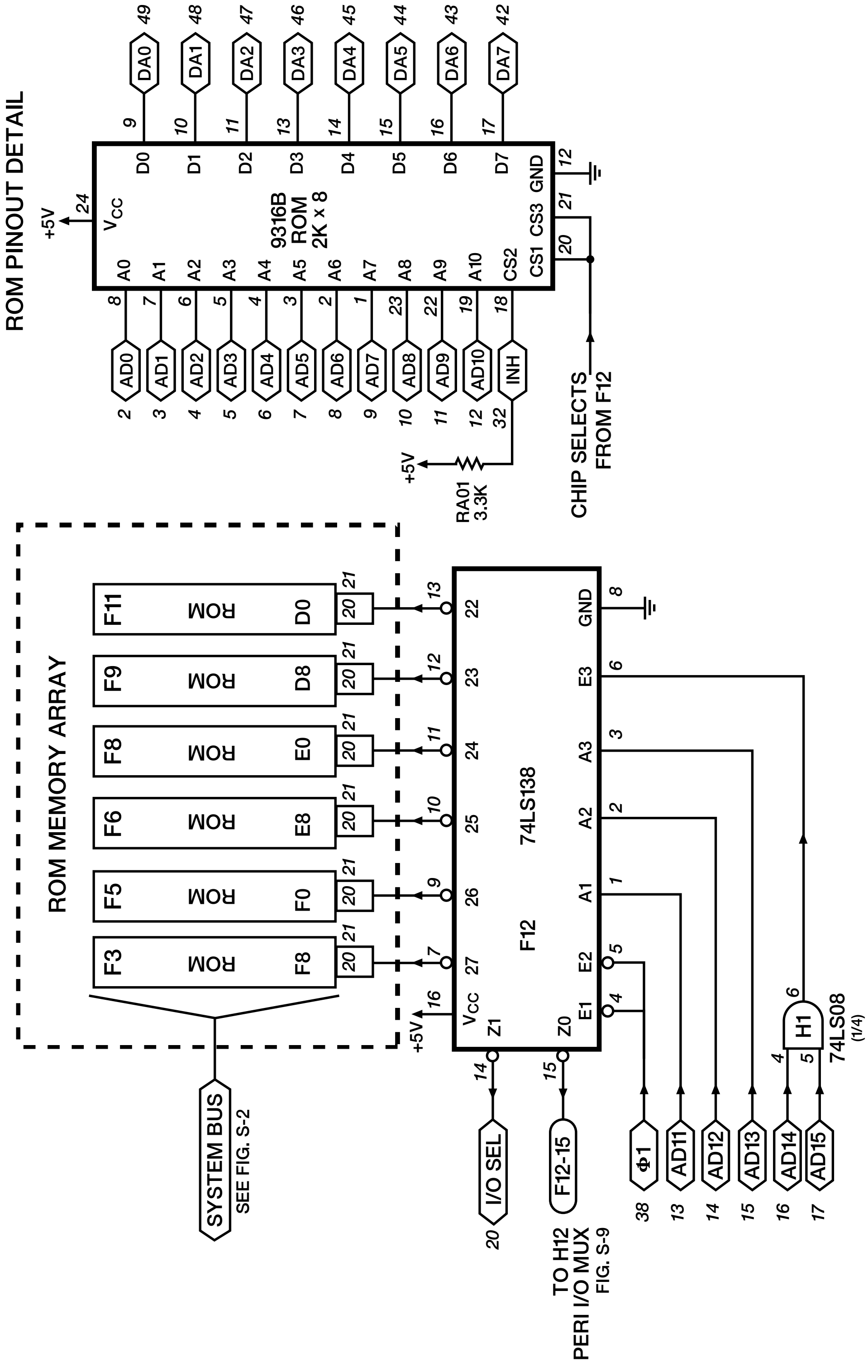


FIGURE S-5 ROM MEMORY

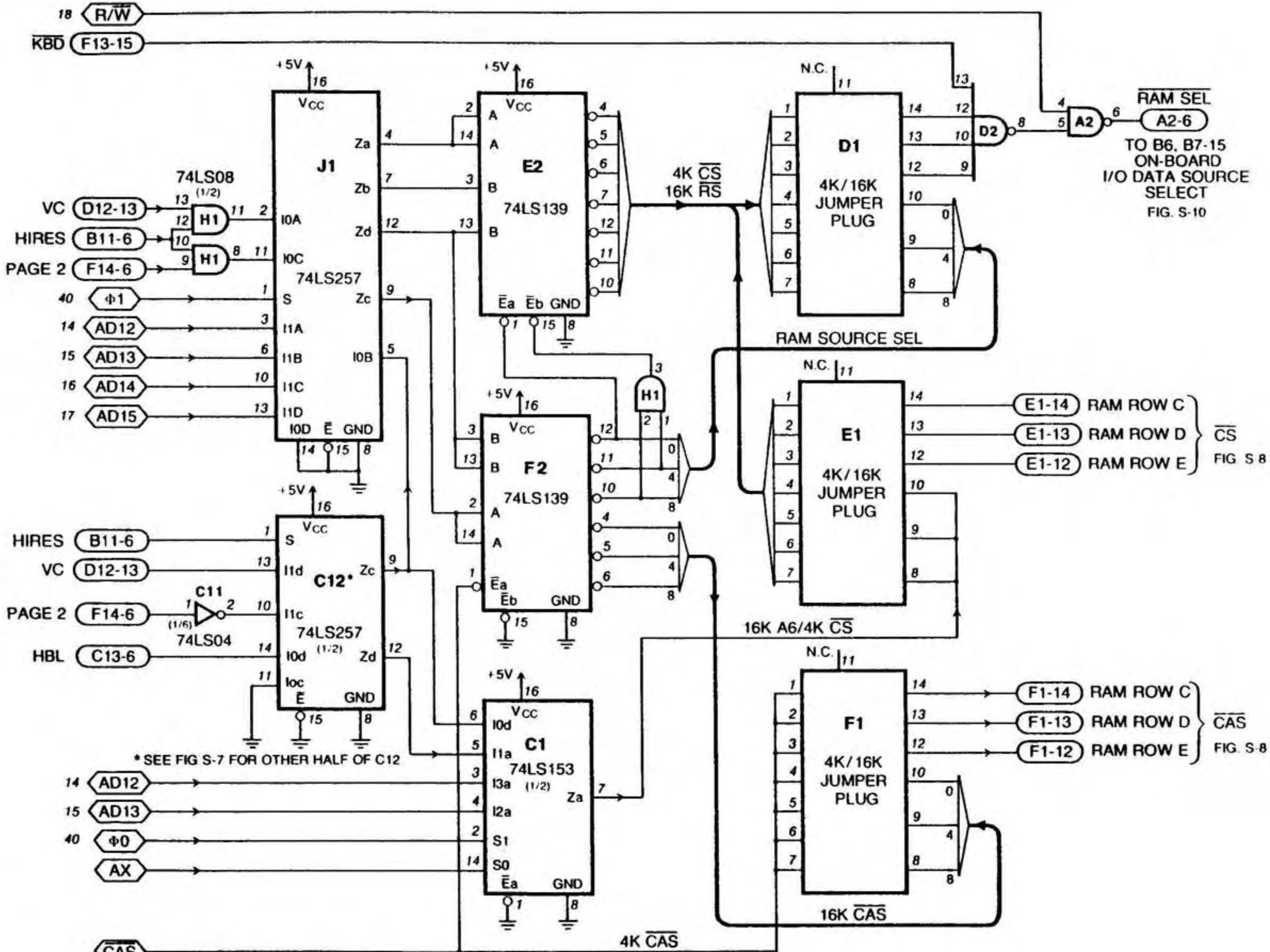


FIGURE S-6 4K/16K RAM SELECT

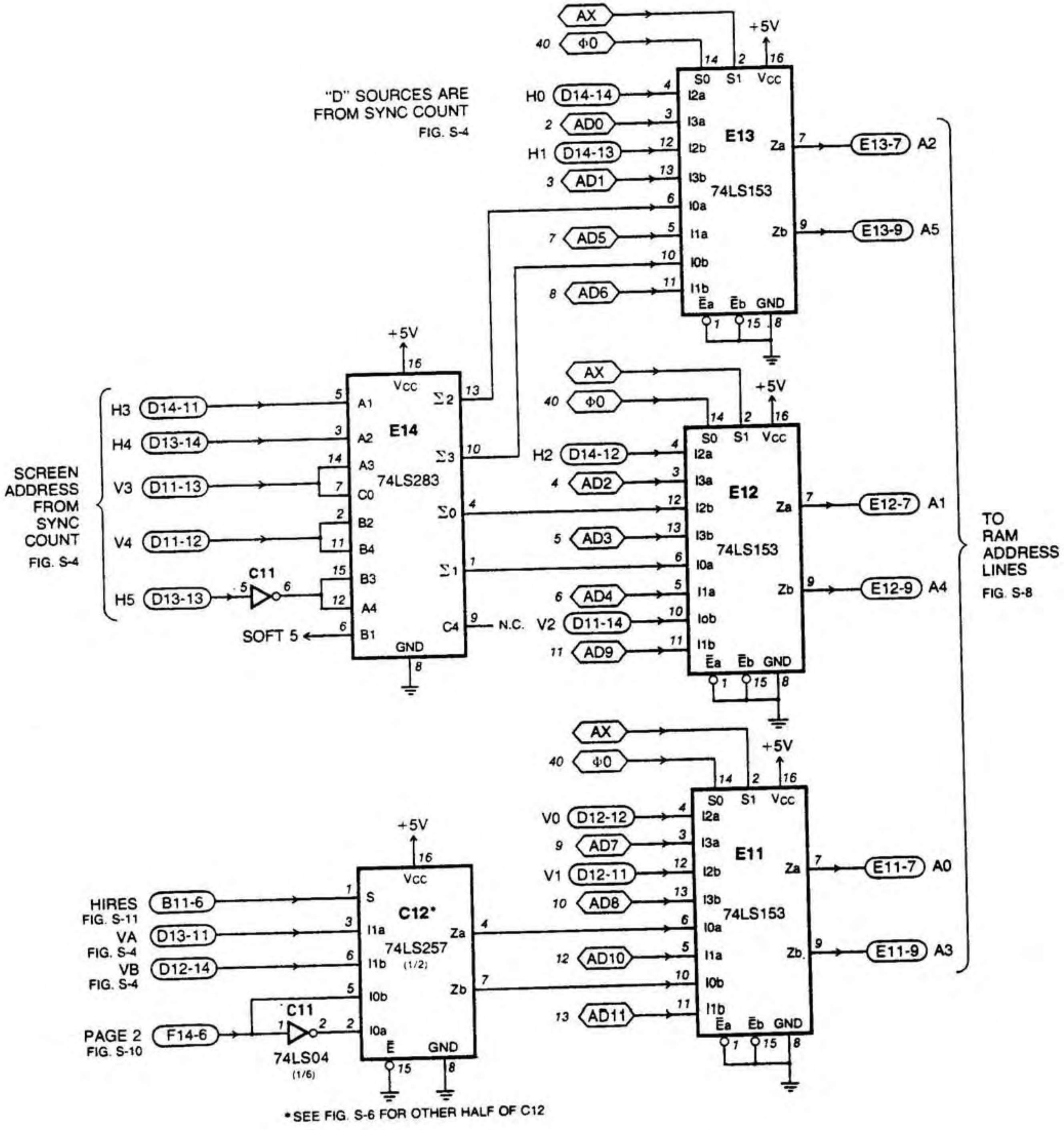


FIGURE S-7 RAM ADDRESS MUX

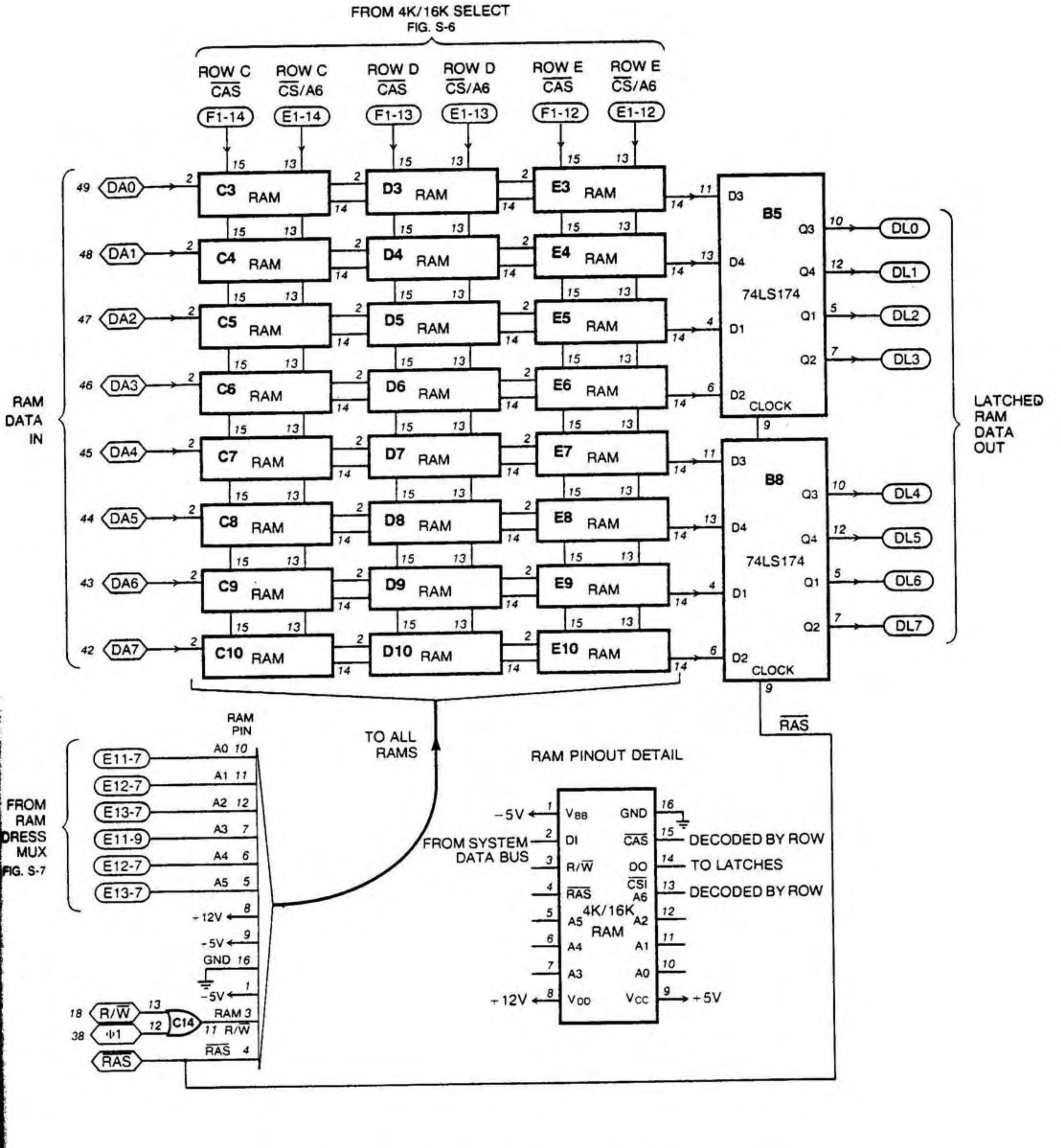


FIGURE S-8 4K TO 48K RAM MEMORY WITH DATA LATCH

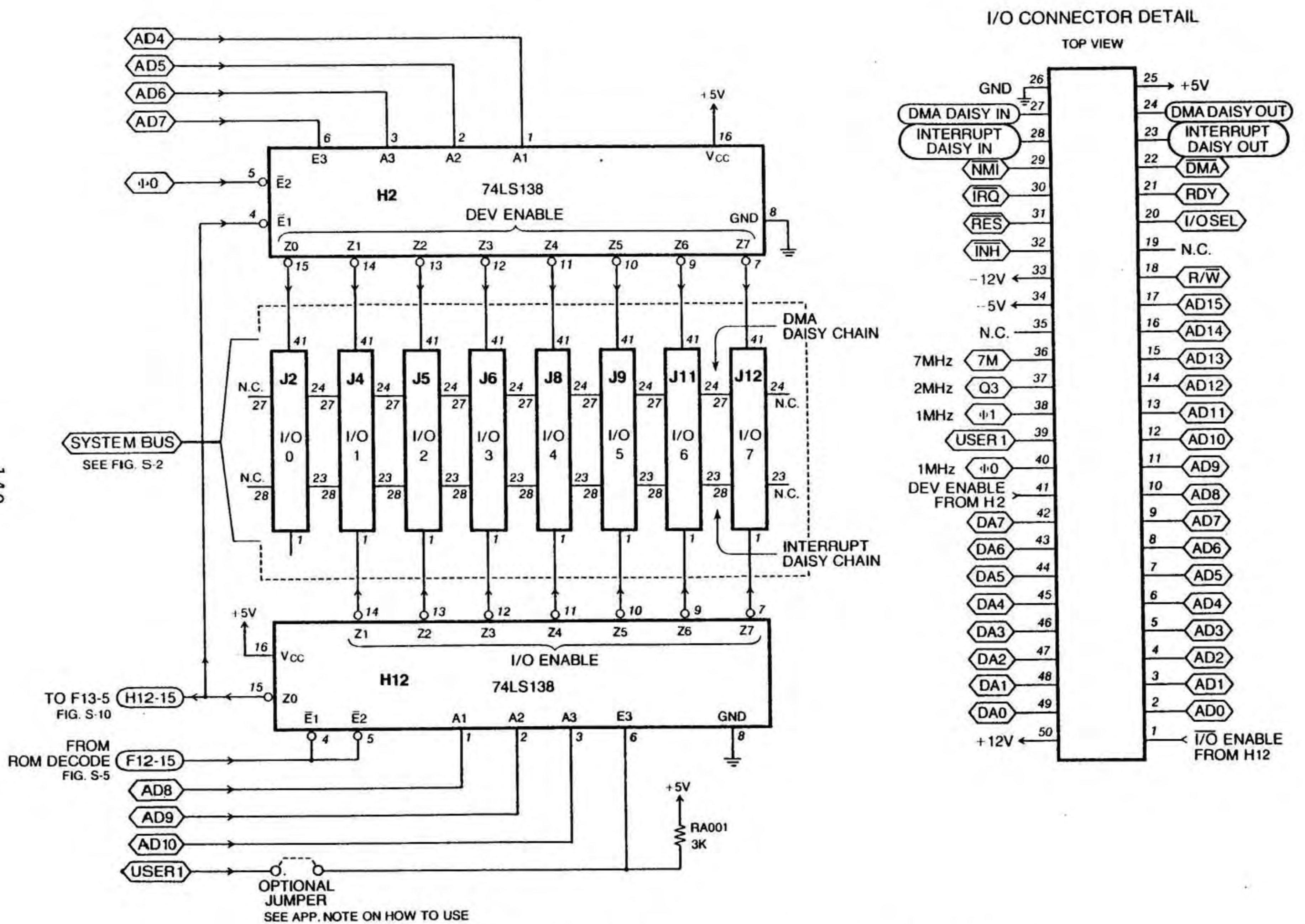


FIGURE S-9 PERIPHERIAL I/O CONNECTOR PINOUT AND CONTROL LOGIC

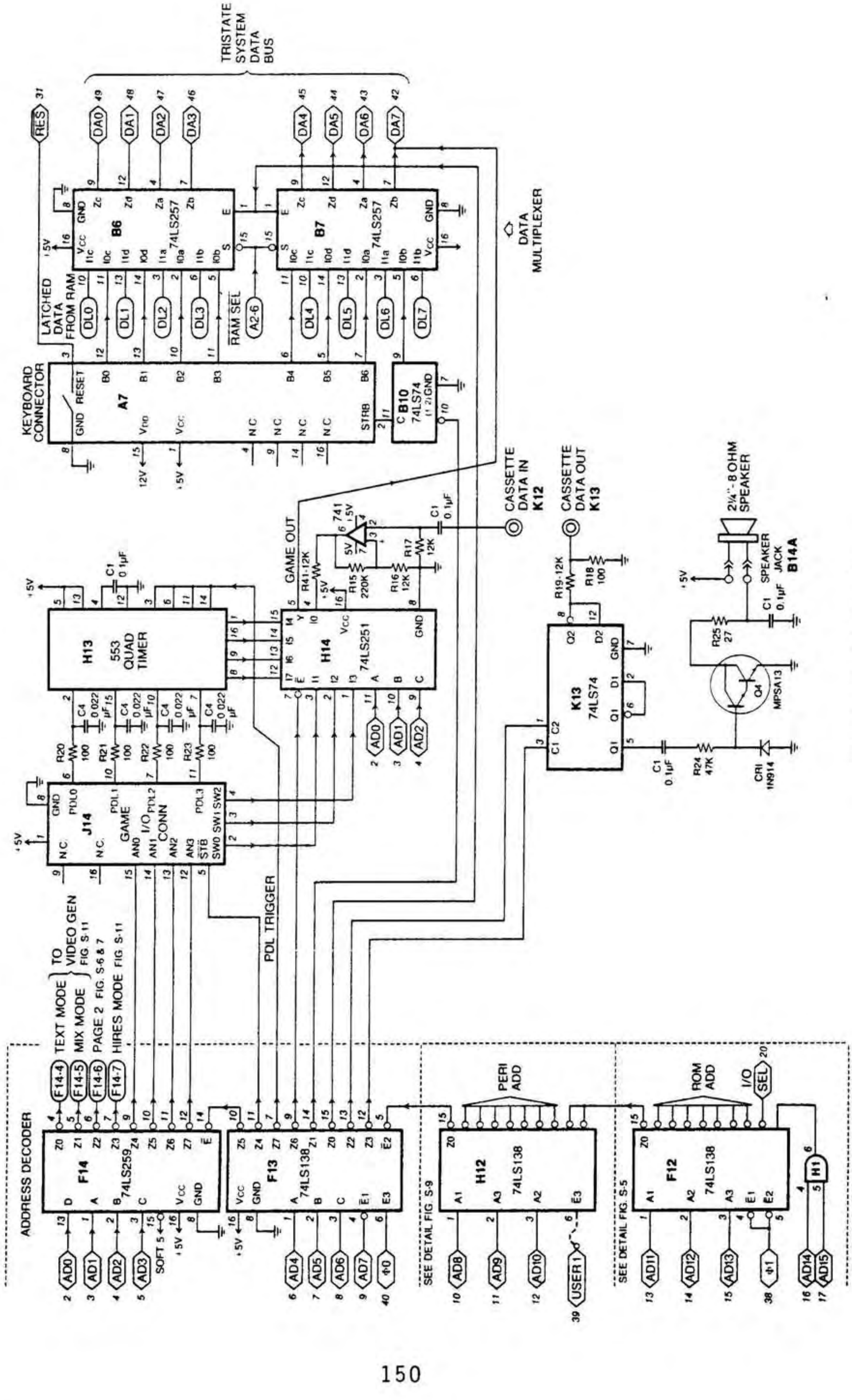


FIGURE S-10 ON-BOARD I/O

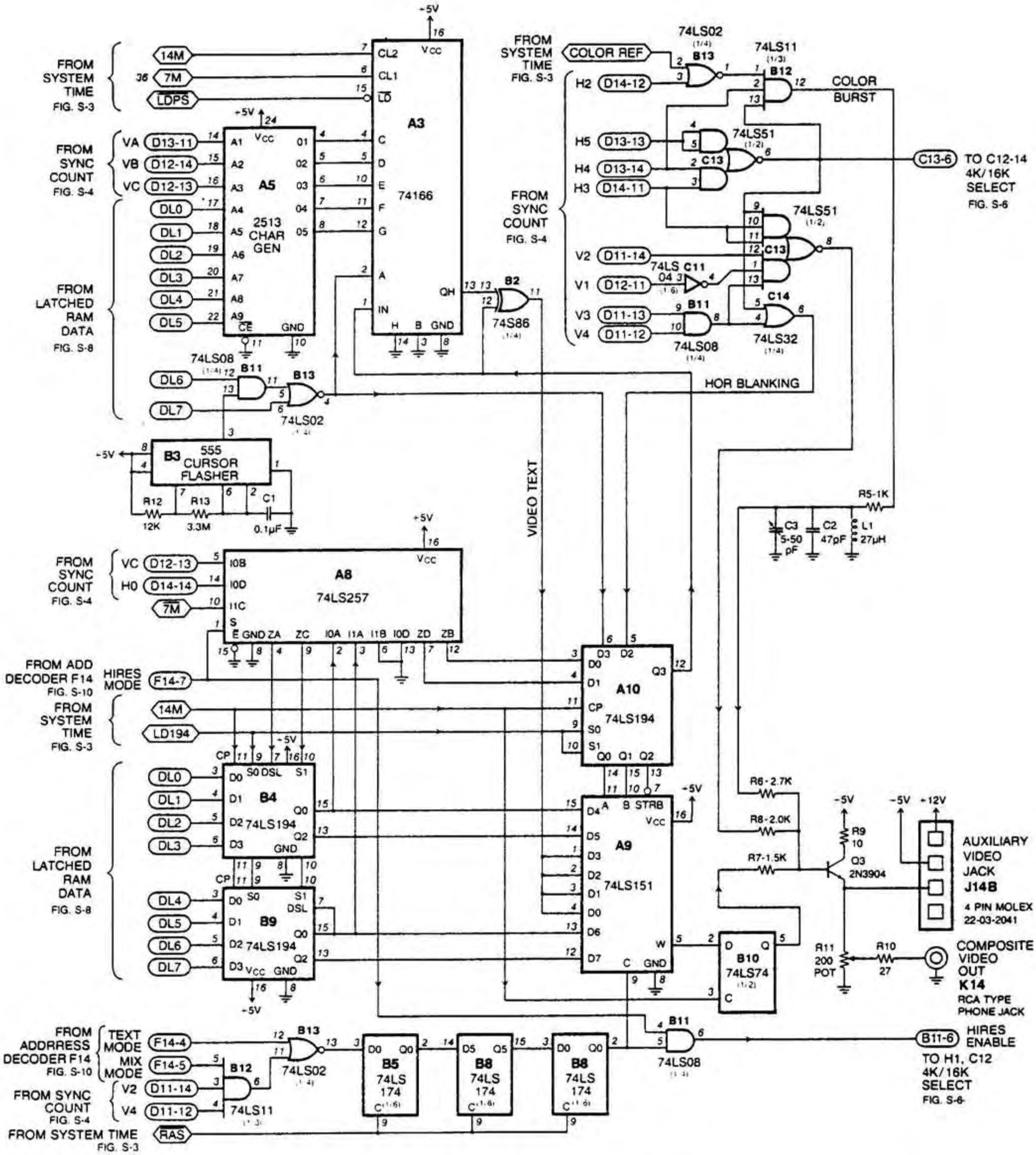


FIGURE S-11 VIDEO GENERATOR

apple computer inc.

10260 BRANDLEY DRIVE
CUPERTINO, CALIFORNIA 95014 U.S.A.
TELEPHONE (408) 996-1010

apple computer inc.

10260 BRANDLEY DRIVE
CUPERTINO, CALIFORNIA 95014 U.S.A.
TELEPHONE (408) 996-1010