Apple II Reference Manual

January 1978
APPLE II Reference Manual

TABLE OF CONTENTS

A. GETTING STARTED WITH YOUR
APPLE II .................... 1
1. Unpacking .................. 1
2. Warranty Registration Card .... 1
3. Check for Shipping Damage .... 2
4. Power Up .................... 2
5. APPLE II Speaks Several Languages 3
6. APPLE Integer BASIC ......... 3
7. Running Your First
   and Second Programs .......... 3
8. Running 16K Startrek .......... 3
9. Loading a Program Tape ...... 4
10. Breakout and Color Demos Tapes . 6
11. Breakout and Color
    Demos Program Listings ....... 12
12. How to Play Startrek .......... 14
13. Loading HIRES Demo Tape ..... 15

B. APPLE II INTEGER BASIC ..... 17
1. BASIC Commands ............. 18
2. BASIC Operators ............ 19
3. BASIC Functions .......... 22
4. BASIC Statements ........... 23
5. Special Control and Editing ... 28
6. Table A — Graphics Colors .. 29
7. Special Controls and Features .. 30
8. BASIC Error Messages ..... 32
9. Simplified Memory Map ..... 33
10. Data Read/Save Subroutines .. 34
11. Simple Tone Subroutines ... 43
12. High Resolution Graphics
    Subroutines and Listings ..... 46

C. APPLE II FIRMWARE .......... 67
1. System Monitor Commands ..... 68
2. Control and Editing Characters ... 72
3. Special Controls and Features .. 74
4. Annotated Monitor and
   Dis-assembly Listing .......... 76
5. Binary Floating Point Package .. 94
6. Sweet 16 Interpreter Listing .. 96
7. 6502 Op Codes ................ 100

D. APPLE II HARDWARE .......... 106
1. Getting Started with Your
   APPLE II Board ............. 107
2. APPLE II Switching Power Supply 110
3. Interfacing with the Home TV ... 112
4. Simple Serial Output .......... 114
5. Interfacing the APPLE —
   Signals, Loading, Pin
   Connections .................. 122
6. Memory —
   Options, Expansion, Map,
   Address ...................... 133
7. System Timing ................ 140
8. Schematics ................... 141
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 peripheral 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 120 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 random 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 similar 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 Integer 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 integer 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 before 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 sections; 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 $40 - $50 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 10 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 (K14) 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. Do not depress the "RETURN" key yet.

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 everything 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 160 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 720 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 0 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 (150k 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.
```
5 GOTO 15
10 IF (PDL (0):20)>6: IF (0:THEN
0:0: IF (0)=24 THEN 0:0: COLOR=
D: VLIN 0,0,5 AT 0: COLOR=0:
IF (P)=0 THEN 175: IF (P) THEN
VLIN 0,0,1 AT 0: P=0: RETURN
15 DIM A(15),B(15):A=0:1=1:
C=0:0:6: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/H? ";,B5: OR : CALL
-556: IF B5=1,196"M" THEN 40
;; FOR I=0 TO 33: COLOR=1,2#
(132): VLIN 0,39 AT 1
30 NEXT I: POKE 34,201: PRINT
: PRINT : PRINT : FOR I=0 TO
15: VTAB 21+$MOD 2: TAB I+
I+1: PRINT I+1: NEXT I: POKE
34,22: VTAB 24: PRINT "BACKGROUND":";
35 GOSUB 95:A=E: PRINT "EVEN BRICK":
;; GOSUB 95:B=E: PRINT "ODD BRICK K":GOSUB 95:C=E: PRINT "PALETTE":
;; GOSUB 95:D=E: PRINT "BALL":GOSUB 95
40 POKE 34,20: COLOR=0: 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: FOR I=
VLIN 0,39 AT I: PRINT: FOR
J=1 MOD 4 TO 39 STEP 4
45 VLIN J=1 AT I: NEXT J=1: TAB
5: PRINT "SCORE = ": PRINT
;: PRINT : POKE 34,215:8=P=
5: L=5: X=19: Y=19: L=6
50 COLOR:R: PLT X,3=3;X=19:Y=
RND (128):Y=RND (5):
2=L=L-1: IF L=1 THEN 120: TAB
6; IF L=1 THEN PRINT L=": BALLS LEFT"
55 IF L=1 THEN PRINT "LAST BALL, ";
A#: PRINT " FOR I=1 TO 100
(SYS) 10: NEXT I=1:MI=6:
60 J=Y=1: IF J=0 AND J>120 THEN
5: SYS=J=1: FOR I=1 TO 65:
SYS=J=1: FOR I=1 TO 65:
PEEK (-16336): NEXT I
65 I=8+1: IF I=6 THEN 100: GOSUB
7: COLOR=I+J=6: IF I>39
8: I+1: NEXT I=6: I=6:
80 V=V=-Y
85 PLOT X,3=3: COLOR=E: PLOT I,
K=1:X=I:Y=J: GOSUB 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 Y=ABS (Y): VLIN
K=2*K: X=2*X+1 AT I=15: M=5,1=2-2-
9: VTAB 21: TAB 13: PRINT S
105 G=PEEK (-16336): PEEK (-16336)
*: PECK (-16336): PECK (-16336)
*: PECK (-16336): PECK (-16336)
*: PECK (-16336): PECK (-16336)
*: PECK (-16336): PECK (-16336)
*
110 IF S=720 THEN 30
115 PRINT "CONGRATULATIONS, ";A#
: "YOU WIN": GOTO 165
120 PRINT "YOUR SCORE OF ";S=6:
;; GOTO 125:S<600:65
125 PRINT "TERRIBLE!": GOTO 165
130 PRINT "Lousy": GOTO 165
135 PRINT "BAD": GOTO 165
140 PRINT "FAIR": GOTO 165
145 PRINT "GOOD": GOTO 165
150 PRINT "EXCELLENT": GOTO 165
155 PRINT "NEARLY PERFECT.
165 PRINT "ANOTHER GAME ";A#: (Y/H)
;; INPUT A#: IF A#(1,1)="Y"
TH 23: TEXT = CALL -936:
V TAB 16: TAB 16: PRINT "GAME NUM-
ER": END
170 O=(PDL (0):28)/6: IF O=0 THEN
0:0: IF O=34 THEN 0:0: COLOR=
0: VLIN 0,0,5 AT 0: COLOR=0:
IF O<9 THEN 175: IF O THEN
VLIN 0,0,1 AT 0: P=0: RETURN
175 IF P=0 THEN RETURN: IF O=34
THEN VLIN 0,6,39 AT 0:6=P=0:
RETURN
180 FOR I=1 TO 65:0: PEEK (-16336)
;; NEXT I= GOTO 50
```
COLOR DEMO PROGRAM LISTING


30 TEXT = CALL -996: VTAB 4: TAB 0: PRINT "4K COLOR DEMO": PRINT "PRINT "i LINES": PRINT "2 CROS 5": PRINT "9 WAVING"

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 POL(0) CONNECTED""

60 PRINT "HIT ANY KEY FOR NEW DEMO";2:0: PRINT : PRINT "WHICH DEMO ";1: GR : IF A0 AND A11 THEN GOTO 100+1: GOTO 30

70 INPUT "WHICH DEMO WOULD YOU LIKE ";1: GR : IF A1 AND A28 THEN GOTO 100+1: GOTO 30

100 I=1+I MOD 79:J=I(I+39) MOD 179 -1-I: GOSUB 2000: GOSUB 10000: GOTO 100


300 J=I+1: J MOD 22+1: FOR I=1 TO 1295: COLOR=I MOD 4+1: PLOT (241) 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=1 TO 1 STEP -1: FOR J=1 TO 4


500 2:20: GOTO 980


640 X=I+7;L=K*K+3*K*26+70;L=32767 / L(X) POL (0) 100: POKE 0,X: POKE 1,L MOD 256: POKE 24, L/256+1: CALL 2: RETURN

700 I=3+(X)MOD 3: J=2+I*5+I*26+ 70:K=32767//X: POL (0) 100: POKE 0,1: POKE 1,K MOD 256: POKE 24,(K)255+1: CALL 2: GOSUB 10000: GOTO 700

800 X=3+8: A=X: L=20+K+4;Y=0: J=1: COLOR=6: HLINE 0,39 AT 4: COLOR=9: GOSUB 388: COLOR=12: VLINE 5,M-2 AT X


880 N=L-X: L=1-2*L+1: VLINE L1, L2 AT X: VLINE L1,L2 AT X: VLINE L1,L2 AT X=1: RETURN


1000 CALL -996


2000 COLOR= RND (16): HLINE 0,39 AT J: COLOR= RND (16): VLINE 0,39 AT J: RETURN

10000 IF PEEK (-16384)X128 THEN RETURN : POKE -16388,0: POP: GOTO 30
THIS IS A SHORT DESCRIPTION OF HOW TO PLAY STAR TREK 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 REMOTE:

- # 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 COMMANDS ARE INVOKED BY TYPING THE 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 OR WARP FACTOR IS THE NUMBER OF SPACES OR
   QUADRANTS THE ENTERPRISE WILL MOVE.
   COURSE IS COMPASS READING IN DEGREES FOR THE DESI-
   RATED DESTINATION.

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

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

   D. A 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 TRAJECTORY ANGLE
   IS KNOWN.

   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 EXPENSE OF AVAILABLE ENERGY.
   THE ANSWER SHOULD BE A SIGNED NUMBER, FOR EXAMPLE
   45 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 TRAJECTORY
   6. STATUS
   7. RETURN TO COMMAND MODE
   IN THE FIRST FIVE ONE WILL HAVE TO GIVE COORDINATES.
   COORDINATES ARE GIVEN IN MATHEMATICAL NOTATION WITH
   THE EXCEPTION THAT THE "Y" VALUE IS GIVEN FIRST.
   AN EXAMPLE WOULD BE "Y,X"

   COURSE OR TRAJECTORY:

   0
   90
   180

   270
  ----------

   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 (50-70%) and Tone (80-100%) settings.

4. Type in "C00.FFFR" on the Apple II keyboard. This is the address range of the high resolution machine language subprogram. It extends from $C000 to $FFFF. The R tells the computer to read in the data. Do not 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. Do not 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 C00.FFFR

3. Start tape recorder, hit RETURN

4. Asterick or flashing cursor reappear B^C (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

1. BASIC Commands
2. BASIC Operators
3. BASIC Functions
4. BASIC Statements
5. Special Control and Editing
6. Table A — Graphics Colors
7. Special Controls and Features
8. BASIC Error Messages
9. Simplified Memory Map
10. Data Read/Save Subroutines
11. Simple Tone Subroutines
12. High Resolution Graphics
13. Additional BASIC Program Examples
**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**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AUTO num</strong></td>
<td>Sets automatic line numbering mode. Starts at line number <strong>num</strong> and increments line numbers by 10. To exit AUTO mode, type a control X*, then type the letters &quot;MAN&quot; and press the return key.</td>
</tr>
<tr>
<td><strong>AUTO num1, num2</strong></td>
<td>Same as above except increments line numbers by number <strong>num2</strong>.</td>
</tr>
<tr>
<td><strong>CLR</strong></td>
<td>Clears current BASIC variables; undimensions arrays. Program is unchanged.</td>
</tr>
<tr>
<td><strong>CON</strong></td>
<td>Continues program execution after a stop from a control C*. Does not change variables.</td>
</tr>
<tr>
<td><strong>DEL num1</strong></td>
<td>Deletes line number <strong>num1</strong>.</td>
</tr>
<tr>
<td><strong>DEL num1, num2</strong></td>
<td>Deletes program from line number <strong>num1</strong> through line number <strong>num2</strong>.</td>
</tr>
<tr>
<td><strong>DSP var</strong></td>
<td>Sets debug mode that will display variable <strong>var</strong> everytime 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.)</td>
</tr>
<tr>
<td><strong>HIMEM: expr</strong></td>
<td>Sets highest memory location for use by BASIC at location specified by expression <strong>expr</strong> 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*.</td>
</tr>
<tr>
<td><strong>GOTO expr</strong></td>
<td>Causes immediate jump to line number specified by expression <strong>expr</strong>.</td>
</tr>
<tr>
<td><strong>GR</strong></td>
<td>Sets mixed color graphics display mode. Clears screen to black. Resets scrolling window. Displays 40x40 squares in 15 colors on top of screen and 4 lines of text at bottom.</td>
</tr>
<tr>
<td><strong>LIST</strong></td>
<td>Lists entire program on screen.</td>
</tr>
<tr>
<td><strong>LIST num1</strong></td>
<td>Lists program line number <strong>num1</strong>.</td>
</tr>
<tr>
<td><strong>LIST num1, num2</strong></td>
<td>Lists program line number <strong>num1</strong> through line number <strong>num2</strong>.</td>
</tr>
</tbody>
</table>
**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 2048 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 formatted to display alpha-numeric characters on 24 lines of 40 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 similar 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**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Sample Statement</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Prefix Operators</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( )</td>
<td>1Ø X = 4*(5 + X)</td>
<td>Expressions within parenthesis ( ) are always evaluated first.</td>
</tr>
<tr>
<td>+</td>
<td>2Ø X = 1+4*5</td>
<td>Optional; +1 times following expression.</td>
</tr>
<tr>
<td>-</td>
<td>3Ø ALPHA = -(BETA +2)</td>
<td>Negation of following expression.</td>
</tr>
<tr>
<td>NOT</td>
<td>4Ø IF A NOT B THEN 2ØØ</td>
<td>Logical Negation of following expression; Ø if expression is true (non-zero), 1 if expression is false (zero).</td>
</tr>
<tr>
<td><strong>Arithmetic Operators</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+</td>
<td>6Ø Y = X+3</td>
<td>Exponentiate as in $X^3$. NOTE: + is shifted letter N.</td>
</tr>
<tr>
<td>*</td>
<td>7Ø LET DOTS=A<em>B</em>N2</td>
<td>Multiplication. NOTE: Implied multiplication such as $(2 + 3)(4)$ is not allowed thus N2 in example is a variable not $N \cdot 2$.</td>
</tr>
<tr>
<td>/</td>
<td>8Ø PRINT GAMMA/S</td>
<td>Divide</td>
</tr>
<tr>
<td>MOD</td>
<td>9Ø X = 12 MOD 7</td>
<td>Modulo: Remainder after division of first expression by second expression.</td>
</tr>
<tr>
<td></td>
<td>10Ø X = X MOD(Y+2)</td>
<td></td>
</tr>
<tr>
<td>+</td>
<td>11Ø P = L + G</td>
<td>Add</td>
</tr>
<tr>
<td>-</td>
<td>12Ø XY4 = H-D</td>
<td>Subtract</td>
</tr>
<tr>
<td>=</td>
<td>13Ø HEIGHT=15</td>
<td>Assignment operator; assigns a value to a variable. LET is optional</td>
</tr>
<tr>
<td></td>
<td>14Ø LET SIZE=7*5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>15Ø A(8) = 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>15Ø ALPHA$ = &quot;PLEASE&quot;</td>
<td></td>
</tr>
</tbody>
</table>
Relational and Logical Operators

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

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Sample Statement</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>160 IF D = E THEN 500</td>
<td>Expression &quot;equals&quot; expression.</td>
</tr>
<tr>
<td>=</td>
<td>170 IF A$(1,1)=&quot;Y&quot; THEN 500</td>
<td>String variable &quot;equals&quot; string variable.</td>
</tr>
<tr>
<td># or &lt; &gt;</td>
<td>180 IF ALPHA #$X*Y THEN 500</td>
<td>Expression &quot;does not equal&quot; expression.</td>
</tr>
<tr>
<td>#</td>
<td>190 IF A$ # &quot;NO&quot; THEN 500</td>
<td>String variable &quot;does not equal&quot; string variable. NOTE: If strings are not the same length, they are considered un-equal. &lt; &gt; not allowed with strings.</td>
</tr>
<tr>
<td>&gt;</td>
<td>200 IF A&gt;B THEN GO TO 50</td>
<td>Expression &quot;is greater than&quot; expression.</td>
</tr>
<tr>
<td>&lt;</td>
<td>210 IF A+1&lt;B-5 THEN 100</td>
<td>Expression &quot;is less than&quot; expression.</td>
</tr>
<tr>
<td>&gt;=</td>
<td>220 IF A&gt;=B THEN 100</td>
<td>Expression &quot;is greater than or equal to&quot; expression.</td>
</tr>
<tr>
<td>&lt;=</td>
<td>230 IF A+1&lt;=B-6 THEN 200</td>
<td>Expression &quot;is less than or equal to&quot; expression.</td>
</tr>
<tr>
<td>AND</td>
<td>240 IF A&gt;B AND C&lt;D THEN 200</td>
<td>Expression 1 &quot;and&quot; expression 2 must both be &quot;true&quot; for statements to be true.</td>
</tr>
<tr>
<td>OR</td>
<td>250 IF ALPHA OR BETA+1 THEN 200</td>
<td>If either expression 1 or expression 2 is &quot;true&quot;, statement is &quot;true&quot;.</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th></th>
<th>300 PRINT ABS(X)</th>
<th>Gives absolute value of the expression expr.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABS</td>
<td>310 PRINT ASC(&quot;BACK&quot;)</td>
<td>Gives decimal ASCII value of designated string variable str$. If more than one character is in designated string or sub-string, it gives decimal ASCII value of first character.</td>
</tr>
<tr>
<td>ASC</td>
<td>320 PRINT ASC(B$)</td>
<td>Gives current length of designated string variable str$; i.e., number of characters.</td>
</tr>
<tr>
<td>LEN</td>
<td>330 PRINT ASC(B$4,4))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>335 PRINT ASC(B$(Y))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>340 PRINT LEN(B$)</td>
<td></td>
</tr>
<tr>
<td>PDL</td>
<td>350 PRINT PDL(X)</td>
<td>Gives number between 0 and 255 corresponding to paddle position on game paddle number designated by expression expr and must be legal paddle (0,1,2, or 3) or else 255 is returned.</td>
</tr>
<tr>
<td>PDL</td>
<td>360 PRINT PEEK(X)</td>
<td>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 FFF0 is -16</td>
</tr>
<tr>
<td>PEEK</td>
<td>370 PRINT RND(X)</td>
<td>Gives random number between 0 and (expression expr -1) if expression expr is positive; if minus, it gives random number between 0 and (expression expr +1).</td>
</tr>
<tr>
<td>RND</td>
<td>380 PRINT SCRN (X1,Y1)</td>
<td>Gives color (number between 0 and 15) of screen at horizontal location designated by expression expr1 and vertical location designated by expression expr2. Range of expression expr1 is 0 to 39. Range of expression expr2 is 0 to 39 if in standard mixed color graphics display mode as set by GR command or 0 to 47 if in all color mode set by POKE -16304 ,0: POKE -16302,0.</td>
</tr>
<tr>
<td>SCRN</td>
<td>390 PRINT SGN(X)</td>
<td>Gives sign (not sine) of expression expr i.e., -1 if expression expr is negative, zero and +1 if expr is positive.</td>
</tr>
<tr>
<td>SGN</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
BASIC STATEMENTS

Each BASIC statement must have a line number between 0 and 32767. Variable names must start with an alpha character and may be any number of alphanumeric characters up to 10. 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 150 characters. Most statements may also be used as commands. BASIC statements are executed by RUN or GOTO commands.

NAME

CALL expr 10 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 10 is hexadecimal number $FC53

COLOR= expr 30 COLOR=12 In standard resolution color (GR) graphics mode, this command sets screen TV color to value in expression expr in the range 0 to 15 as described in Table A. Actually expression expr may be in the range 0 to 255 without error message since it is implemented as if it were expression expr MOD 16.

DIM vari (expr1) 50 DIM A(20), B(10) The DIM statement causes APPLE II to reserve memory for the specified variables. For number arrays, APPLE reserves approximately 2 times expr1 bytes of memory limited by available memory. For string arrays - str$ (expr2) must be in the range of 1 to 255. Last defined variable may be redimensioned at any time; thus, example in line is illegal but 85 is allowed.

str$ (expr2) 60 DIM B$ (30) Legal:

var2 (expr3) 70 DIM C (2)

ILLEGAL:

80 DIM A (30)

LEGAL:

85 DIM C (1000)

DSP var Legal:

90 DSP AX; DSP L

ILLEGAL:

100 DSP AX,B

102 DSP AB$

104 DSP A (5)

LEGAL:

105 A = A (5): DSP A

Sets debug mode that DSP variable var each time it changes and the line number where the change occurred.
<table>
<thead>
<tr>
<th>NAME</th>
<th>EXAMPLE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>END</strong></td>
<td>110 END</td>
<td>Stops program execution. Sends carriage return and &quot;&gt; &quot; BASIC prompt to screen.</td>
</tr>
<tr>
<td><strong>FOR</strong></td>
<td>110 FOR L=0 TO 39</td>
<td>Begins FOR...NEXT loop, initializes variable var to value of expression expr1 then increments it by amount in expression expr2 each time the corresponding &quot;NEXT&quot; 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.</td>
</tr>
<tr>
<td></td>
<td>120 FOR X=Y1 TO Y3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>130 FOR I=39 TO 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>150 GOSUB 100 *J2</td>
<td></td>
</tr>
<tr>
<td><strong>GOSUB</strong></td>
<td>140 GOSUB 500</td>
<td>Causes branch to BASIC subroutine starting at legal line number specified by expression expr. Subroutines may be nested up to 16 levels.</td>
</tr>
<tr>
<td><strong>GOTO</strong></td>
<td>160 GOTO 200</td>
<td>Causes immediate jump to legal line number specified by expression expr.</td>
</tr>
<tr>
<td></td>
<td>170 GOTO ALPHA+100</td>
<td></td>
</tr>
<tr>
<td><strong>GR</strong></td>
<td>180 GR</td>
<td>Sets mixed standard resolution color graphics mode. Initializes COLOR = 0 (Black) for top 40x40 of screen and sets scrolling window to lines 21 through 24 by 40 characters for four lines of text at bottom of screen. Example 190 sets all color mode (40x48 field) with no text at bottom of screen.</td>
</tr>
<tr>
<td></td>
<td>190 GR: POKE -16302,0</td>
<td></td>
</tr>
<tr>
<td><strong>HLIN</strong></td>
<td>200 HLIN 0,39 AT 20</td>
<td>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 expr1 and ending at position expr2 at vertical position defined by expression expr3. expr1 and expr2 must be in the range of 0 to 39 and expr1 &lt; = expr2 . expr3 be in the range of 0 to 39 (or 0 to 47 if not in mixed mode).</td>
</tr>
<tr>
<td></td>
<td>210 HLIN 0,2 AT 1</td>
<td></td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>HLIN 0, 19 AT 0 is a horizontal line at the top of the screen extending from left corner to center of screen and HLIN 20,39 AT 39 is a horizontal line at the bottom of the screen extending from center to right corner.</td>
<td></td>
</tr>
</tbody>
</table>
**IF expression THEN statement**

220 IF A > B THEN
   PRINT A
230 IF X=0 THEN C=1
240 IF A#10 THEN
   GOSUB 200
250 IF A$(1,1)# "Y"
   THEN 100

Illegal:

260 IF L > 5 THEN 50;
   ELSE 60

Legal:

270 IF L > 5 THEN 50
   GO TO 60

**INPUT vari, var2, str$**

280 INPUT X,Y,Z(3)
290 INPUT "AMT",
   DLLR
300 INPUT "Y or N?", A$

Enters data into memory from I/O device. If number input is expected,
APPLE will output "?"; if string input is expected no "?" will be output.
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**

310 IN# 6
320 IN# Y+2
330 IN# 0

Transfers source of data for subsequent
INPUT statements to peripheral I/O slot
(1-7) as specified as by expression expr.
Slot 0 is not addressable from BASIC.
IN#0 (Example 330) is used to return data
source from peripheral I/O to keyboard
connector.

**LET**

340 LET X=5

Assignment operator. "LET" is optional

**LIST num1, num2**

350 IF X > 6 THEN
   LIST 50

Causes program from line number num1
through line number num2 to be displayed
on screen.

**NEXT vari, var2**

360 NEXT I
370 NEXT J,K

Increments corresponding "FOR" variable
and loops back to statement following
"FOR" until variable exceeds limit.

**NO DSP vari**

380 NO DSP I

Turns-off DSP debug mode for variable

**NO TRACE**

390 NO TRACE

Turns-off TRACE debug mode
PLOT, expr1, expr2  400 PLOT 15, 25
    400 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 0 to 39 and vertical location specified by expression expr2 in range 0 to 39 (or 0 to 47 if in all graphics mode) NOTE: PLOT 0 0 is upper left and PLOT 39, 39 (or PLOT 39, 47) is lower right corner.

POKE expr1, expr2  420 POKE 20, 40
    430 POKE 7*256, XMOD255  Stores decimal number defined by expression expr2 in range of 0 to 255 at decimal memory location specified by expression expr1. Locations above 32767 are specified by negative numbers.

POP  440 POP  "POPS" nested GOSUB return stack address by one.

PRINT var1, var, str$  450 PRINT L1
    460 PRINT L1, X2
    470 PRINT "AMT=";DX
    480 PRINT A$;B$;
    490 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 450) a carriage return will be generated. Commas (Ex. 460) outputs data in 5 left justified columns. Semi-colon (Ex. 470) inhibits print of any spaces. Text imbedded in " " will be printed and may appear multiple times.

PR# expr  500 PR# 7  Like IN#, transfers output to I/O slot defined by expression expr. PR# 0 is video output not I/O slot 0.

REM  510 REM REMARK  No action. All characters after REM are treated as a remark until terminated by a carriage return.

RETURN  520 RETURN
    530 IF X= 5 THEN RETURN  Causes branch to statement following last GOSUB; i.e., RETURN ends a subroutine. Do not confuse "RETURN" statement with Return key on keyboard.
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TAB expr</strong></td>
<td>Moves cursor to absolute horizontal position specified by expression <code>expr</code> in the range of 1 to 40. Position is left to right.</td>
</tr>
<tr>
<td><strong>TEXT</strong></td>
<td>Sets all text mode. Resets scrolling window to 24 lines by 40 characters. Example 560 also clears screen and homes cursor to upper left corner.</td>
</tr>
<tr>
<td><strong>TRACE</strong></td>
<td>Sets debug mode that displays each line number as it is executed.</td>
</tr>
<tr>
<td><strong>VLIN expr1, expr2, AT expr3</strong></td>
<td>Similar to HLIN except draws vertical line starting at <code>expr1</code> and ending at <code>expr2</code> at horizontal position <code>expr3</code>.</td>
</tr>
<tr>
<td><strong>VTAB expr</strong></td>
<td>Similar to TAB. Moves cursor to absolute vertical position specified by expression <code>expr</code> in the range 1 to 24. VTAB 1 is top line on screen; VTAB24 is bottom.</td>
</tr>
</tbody>
</table>
SPECIAL CONTROL AND EDITING CHARACTERS

"Control" characters are indicated by a super-scripted "C" such as G\textsuperscript{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\textsuperscript{C} and C\textsuperscript{C} must be followed by a carriage return. Screen editing characters are indicated by a sub-scripted "E" such as D\textsubscript{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\textsuperscript{C} moves to cursor to right and copies text while A\textsubscript{E} moves cursor to right but does not copy text.

<table>
<thead>
<tr>
<th>CHARACTER</th>
<th>DESCRIPTION OF ACTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESET key</td>
<td>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 &quot;*&quot; (asterisk) and a bell. Hitting RESET key does NOT destroy existing BASIC or machine language program.</td>
</tr>
<tr>
<td>Control B</td>
<td>If in System Monitor (as indicated by a &quot;*&quot;), a control B and a carriage return will transfer control to BASIC, scratching (killing) any existing BASIC program and set HIMEM: to maximum installed user memory and LOMEM: to 2048.</td>
</tr>
<tr>
<td>Control C</td>
<td>If in BASIC, halts program and displays line number where stop occurred*. Program may be continued with a CON command. If in System Monitor, (as indicated by &quot;*&quot;), control C and a carriage return will enter BASIC without killing current program.</td>
</tr>
<tr>
<td>Control G</td>
<td>Sounds bell (beeps speaker)</td>
</tr>
<tr>
<td>Control H</td>
<td>Backspaces cursor and deletes any overwritten characters from computer but not from screen. Apply supplied keyboards have special key &quot;*&quot; on right side of keyboard that provides this function.</td>
</tr>
<tr>
<td>Control J</td>
<td>Issues line feed only</td>
</tr>
<tr>
<td>Control V</td>
<td>Compliment to H\textsuperscript{C}. Forward spaces cursor and copies overwritten characters. Apple keyboards have &quot;*&quot; key on right side which also performs this function.</td>
</tr>
<tr>
<td>Control X</td>
<td>Immediately deletes current line.</td>
</tr>
</tbody>
</table>

* If BASIC program is expecting keyboard input, you will have to hit carriage return key after typing control C.
<table>
<thead>
<tr>
<th>CHARACTER</th>
<th>DESCRIPTION OF ACTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A_E )</td>
<td>Move cursor to right</td>
</tr>
<tr>
<td>( B_E )</td>
<td>Move cursor to left</td>
</tr>
<tr>
<td>( C_E )</td>
<td>Move cursor down</td>
</tr>
<tr>
<td>( D_E )</td>
<td>Move cursor up</td>
</tr>
<tr>
<td>( E_E )</td>
<td>Clear text from cursor to end of line</td>
</tr>
<tr>
<td>( F_E )</td>
<td>Clear text from cursor to end of page</td>
</tr>
<tr>
<td>( \Theta_E )</td>
<td>Home cursor to top of page, clear text to end of page.</td>
</tr>
</tbody>
</table>

Table A: APPLE II COLORS AS SET BY COLOR =

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

\[ \begin{align*}
\emptyset &= \text{Black} \\
1 &= \text{Magenta} \\
2 &= \text{Dark Blue} \\
3 &= \text{Light Purple} \\
4 &= \text{Dark Green} \\
5 &= \text{Grey} \\
6 &= \text{Medium Blue} \\
7 &= \text{Light Blue} \\
8 &= \text{Brown} \\
9 &= \text{Orange} \\
10 &= \text{Grey} \\
11 &= \text{Pink} \\
12 &= \text{Green} \\
13 &= \text{Yellow} \\
14 &= \text{Blue/Green} \\
15 &= \text{White} 
\end{align*} \]
## Special Controls and Features

### Display Mode Controls

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

### TEXT Mode Controls

<table>
<thead>
<tr>
<th>Hex</th>
<th>BASIC Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0020</td>
<td>90 POKE 32,L1</td>
<td>Set left side of scrolling window to location specified by L1 in range of 0 to 39.</td>
</tr>
<tr>
<td>0021</td>
<td>100 POKE 33,W1</td>
<td>Set window width to amount specified by W1. L1+W1&lt;40. W1&gt;0</td>
</tr>
<tr>
<td>0022</td>
<td>110 POKE 34,T1</td>
<td>Set window top to line specified by T1 in range of 0 to 23</td>
</tr>
<tr>
<td>0023</td>
<td>120 POKE 35,B1</td>
<td>Set window bottom to line specified by B1 in the range of 0 to 23. B1&gt;T1</td>
</tr>
<tr>
<td>0024</td>
<td>130 CH=PEEK(36)</td>
<td>Read/set cursor horizontal position in the range of 0 to 39. If using TAB, you must add &quot;!&quot; to cursor position read value; Ex. 140 and 150 perform identical function.</td>
</tr>
<tr>
<td></td>
<td>140 POKE 36,CH</td>
<td></td>
</tr>
<tr>
<td></td>
<td>150 TAB(CH+1)</td>
<td></td>
</tr>
<tr>
<td>0025</td>
<td>160 CV=PEEK(37)</td>
<td>Similar to above. Read/set cursor vertical position in the range 0 to 23.</td>
</tr>
<tr>
<td></td>
<td>170 POKE 37,CV</td>
<td></td>
</tr>
<tr>
<td></td>
<td>180 VTAB(CV+1)</td>
<td></td>
</tr>
<tr>
<td>0032</td>
<td>190 POKE 50,127</td>
<td>Set inverse flag if 127 (Ex. 190)</td>
</tr>
<tr>
<td></td>
<td>200 POKE 50,255</td>
<td>Set normal flag if 255 (Ex. 200)</td>
</tr>
<tr>
<td>FC58</td>
<td>210 CALL -936</td>
<td>(@E) Home cursor, clear screen</td>
</tr>
<tr>
<td>FC42</td>
<td>220 CALL -958</td>
<td>(F) Clear from cursor to end of page</td>
</tr>
<tr>
<td>Hex</td>
<td>BASIC Example</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>------------------------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td>FC9C</td>
<td>230 CALL -868</td>
<td>(E_E) Clear from cursor to end of line</td>
</tr>
<tr>
<td>FC66</td>
<td>240 CALL -922</td>
<td>(J_C) Line feed</td>
</tr>
<tr>
<td>FC70</td>
<td>250 CALL -912</td>
<td>Scroll up text one line</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Miscellaneous</strong></td>
</tr>
<tr>
<td>C030</td>
<td>360 X=PEEK(-16336)</td>
<td>Toggle speaker</td>
</tr>
<tr>
<td></td>
<td>365 POKE -16336,0</td>
<td></td>
</tr>
<tr>
<td>C000</td>
<td>370 X=PEEK(-16384)</td>
<td>Read keyboard; if X&gt;127 then key was pressed.</td>
</tr>
<tr>
<td>C010</td>
<td>380 POKE -16368,0</td>
<td>Clear keyboard strobe - always after reading keyboard.</td>
</tr>
<tr>
<td>C061</td>
<td>390 X=PEEK(16287)</td>
<td>Read PDL(0) push button switch. If X&gt;127 then switch is &quot;on&quot;.</td>
</tr>
<tr>
<td>C062</td>
<td>400 X=PEEK(-16286)</td>
<td>Read PDL(1) push button switch.</td>
</tr>
<tr>
<td>C063</td>
<td>410 X=PEEK(-16285)</td>
<td>Read PDL(2) push button switch.</td>
</tr>
<tr>
<td>C058</td>
<td>420 POKE -16296,0</td>
<td>Clear Game I/O AN0 output</td>
</tr>
<tr>
<td>C059</td>
<td>430 POKE -16295,0</td>
<td>Set Game I/O AN0 output</td>
</tr>
<tr>
<td>C05A</td>
<td>440 POKE -16294,0</td>
<td>Clear Game I/O AN1 output</td>
</tr>
<tr>
<td>C05B</td>
<td>450 POKE -16293,0</td>
<td>Set Game I/O AN1 output</td>
</tr>
<tr>
<td>C05C</td>
<td>460 POKE -16292,0</td>
<td>Clear Game I/O AN2 output</td>
</tr>
<tr>
<td>C05D</td>
<td>470 POKE -16291,0</td>
<td>Set Game I/O AN2 output</td>
</tr>
<tr>
<td>C05E</td>
<td>480 POKE -16290,0</td>
<td>Clear Game I/O AN3 output</td>
</tr>
<tr>
<td>C05F</td>
<td>490 POKE -16289,0</td>
<td>Set Game I/O AN3 output</td>
</tr>
<tr>
<td>Error Code</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>*** SYNTAX ERR</td>
<td>Results from a syntactic or typing error.</td>
<td></td>
</tr>
<tr>
<td>*** &gt; 32767 ERR</td>
<td>A value entered or calculated was less than -32767 or greater than 32767.</td>
<td></td>
</tr>
<tr>
<td>*** &gt; 255 ERR</td>
<td>A value restricted to the range 0 to 255 was outside that range.</td>
<td></td>
</tr>
<tr>
<td>*** BAD BRANCH ERR</td>
<td>Results from an attempt to branch to a non-existent line number.</td>
<td></td>
</tr>
<tr>
<td>*** BAD RETURN ERR</td>
<td>Results from an attempt to execute more RETURNS than previously executed GOSUBs.</td>
<td></td>
</tr>
<tr>
<td>*** BAD NEXT ERR</td>
<td>Results from an attempt to execute a NEXT statement for which there was not a corresponding FOR statement.</td>
<td></td>
</tr>
<tr>
<td>*** 16 GOSUBS ERR</td>
<td>Results from more than 16 nested GOSUBs.</td>
<td></td>
</tr>
<tr>
<td>*** 16 FORS ERR</td>
<td>Results from more than 16 nested FOR loops.</td>
<td></td>
</tr>
<tr>
<td>*** NO END ERR</td>
<td>The last statement executed was not an END.</td>
<td></td>
</tr>
<tr>
<td>*** MEM FULL ERR</td>
<td>The memory needed for the program has exceeded the memory size allotted.</td>
<td></td>
</tr>
<tr>
<td>*** TOO LONG ERR</td>
<td>Results from more than 12 nested parentheses or more than 128 characters in input line.</td>
<td></td>
</tr>
<tr>
<td>*** DIM ERR</td>
<td>Results from an attempt to DIMension a string array which has been previously dimensioned.</td>
<td></td>
</tr>
<tr>
<td>*** RANGE ERR</td>
<td>An array was larger than the DIMensioned value or smaller than 1 or HLIN,VLIN, PLOT, TAB, or VTAB arguments are out of range.</td>
<td></td>
</tr>
<tr>
<td>*** STR OVFL ERR</td>
<td>The number of characters assigned to a string exceeded the DIMensioned value for that string.</td>
<td></td>
</tr>
<tr>
<td>*** STRING ERR</td>
<td>Results from an attempt to execute an illegal string operation.</td>
<td></td>
</tr>
<tr>
<td>RETYPE LINE</td>
<td>Results from illegal data being typed in response to an INPUT statement. This message also requests that the illegal item be retyped.</td>
<td></td>
</tr>
</tbody>
</table>
Simplified Memory Map

FFFF  64K  Monitor and BASIC Routines in ROM

E000  56K  Future enhancement or user supplied PROMS

D000  52K  Peripheral I/O

C000  48K

XX  XX  User specified RAM memory size
    (HIMEM:)

User Workspace

7FF  2K  Screen Memory

400  1K

0  0  Internal Workspace
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:

<table>
<thead>
<tr>
<th>VN</th>
<th>DSP</th>
<th>NVA</th>
<th>DATA(0)</th>
<th>DATA(1)</th>
<th>...</th>
<th>DATA(N)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>h₁</td>
<td>h₂</td>
<td></td>
<td>hₙ+₁</td>
</tr>
</tbody>
</table>

VARIABLE NAME - up to 100 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 - 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:

<table>
<thead>
<tr>
<th>VN</th>
<th>DSP</th>
<th>NVA</th>
<th>DATA(0)</th>
<th>DATA(1),...</th>
<th>DATA(n)</th>
<th>ST</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$h_1$</td>
<td>$h_2$</td>
<td>$h_{n+1}$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 - 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 diagrammatically. Variables on the other hand are mapped into memory starting at the lowest position of RAM memory - hex $800 (2048)$ 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 $CC-CD(204-205)$ and $CA-CB(203-204)$, 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 2048 and PP is equal to 16384. These also happen to be the values of LOMEN 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 $0$ (step 2). Now that the variable table has been defined use of statement 2a indicates that CM is equal to 2060 (step 3). LOMEN has not been readjusted so it is equal to 2048. Therefore the variable table resides in memory from 2048 ($800$ hex) to 2060 ($80C$). 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 $800$ hex to $80C$ hex typing in "800.80C" 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 $C1$ 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 $800$ to $805$. We have then found
the memory range of where the variable A is mapped into memory. The reason for this will become clear in the next section.

READ/SAVE ROUTINE

The READ/SAVE subroutine has three parts. The first section (lines 0-10) defines variable A and transfers control to the main program. Lines 20 through 26 represents the Write data to tape routine and lines 30-38 represent the Read data from tape subroutine. Both READ and SAVE routines are executable by the BASIC "GOSUB X" (where X is 20 for write and 30 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 2048

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 GOSUB20 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 GOSUB30 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.
Var1 | Var2 | Var3 | Var4 | Unused | P1 | P2 | P3 | Pn-2 | Pn
-----|------|------|------|--------|----|----|----|------|----

LOMEN: $800
CM End of Variable Table
PP beginning of Program
HIMEM Max System Size

Variable Data

BASIC Program

Figure 1

a) PRINT PEEK(204) + PEEK(205)*256 → PP
b) PRINT PEEK(202) + PEEK(203)*256 → CM

Figure 2

800 801 802 803 804 805 806 807 808 809 80A 80B 80C
CI 00 06 08 FF FF C2 00 0C 08 00 00 00
L H L H L H L H
VAR DSP NVA DATA VAR DSP NVA DATA
NAM

Figure 3

$800.80C rewritten with labelling

39
READ/SAVE PROGRAM

\[ A = \emptyset \]

This must be the first statement in the program. It is initially \( \emptyset \), but if data is to be saved, it will equal the length of the database.

10 GOTO 100

This statement moves command to the main program.

20 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 60,4:
   POKE 61,8: POKE 62,5:
   POKE 63,8: CALL -307

Writing data table to tape

24 POKE 60,LM MOD 256:
   POKE 61, LM/256:
   POKE 62, CM MOD 256:
   POKE 63, CM/256:
   CALL -307

26 PRINT "DATA TABLE SAVED":
   RETURN

Returning control to main program.

30 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 60,4: POKE 61,8:
   POKE 62,5: POKE 63,8:
   CALL -259

34 IF A<\( \emptyset \) 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

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  \texttt{>A=1} \\
   > \texttt{Define variable }A=-1\texttt{, then hit RETURN}

2  \texttt{>B=\emptyset} \\
   > \texttt{Define variable }B=\emptyset\texttt{, then hit RETURN}

3  \texttt{>PRINT PEEK (204) + PEEK (205) * 256} \\
   \texttt{computer responds with= 2060} \texttt{Use statement 2a to find the end of the VARIABLE TABLE}

4  > \\
   * \texttt{Hit the RESET key, Apple moves into Monitor mode.}

5  *\texttt{800.80C} \texttt{Type in VARIABLE TABLE RANGE and HIT the RETURN KEY.}

Computer responds with:
\texttt{800- C1 00 86 08 FF FF C2 00} \\
\texttt{8008 0C 08 00 00 00} \\

Example 1
Example 2

XLIST

0 R=0
10 GOTO 100
20 REM WRITE DATA TO TAPE ROUTINE
22 A=CHR,L=POKE 66,4: POKE 61
30 : POKE 62,5: POKE 63,8: CALL
37 : -367
24 POKE 60,LN MOD 256: POKE 61
40 ,LM/256: POKE 62,CH MOD 256
50 : POKE 63,CM/256: CALL -367

26 REM
30 REM READ DATA SUBROUTINE
32 POKE 60,4: POKE 61,2: POKE
34 62,5: POKE 63,8: CALL -259
36 IF A<>0 THEN 38: P=LM+R: IF P
38 THEN 38: CM=P: POKE 60,LM MOD
40 256: POKE 61,LM/256: POKE 62
45 ,CM MOD 256: POKE 63,CM/256
50 : CALL -259
38 RETURN
38 PRINT *** TOO MUCH DATA BASE **
50: END
100 DIM A(1),X(20)
105 FOR I=1 TO 20;X(I)=I: NEXT
60 I
100 LN=2048:CM=2106:R=58:HN=16383
110 PRINT "20 NUMBERS GENERATED"
120 PRINT "NOW WE ARE GOING TO SAVE
130 THE DATA": PRINT "WHEN YOU ARE R
140 EADY START THE RECORDER IN RECUR
150 D MODE": INPUT "AND HIT RETURN"
160 ,A$
170 CALL -996: PRINT "NOW WRITING DA
180 TA TO TAPE": GOSUB 20
190 PRINT "NOW THE DATA IS SAVED"
200 PRINT "NOW WE ARE GOING TO CLEAR
210 THE X(20) TABLE AND READ THE DA
220 TA FROM TAPE"
230 FOR I=1 TO 20;X(I)=0: PRINT
240 "X(";I");=";X(I);: NEXT I
250 PRINT "NOW START TAPE RECORDER"
260 : INPUT "AND THEN HIT RETURN"
270 ,A$
280 PRINT "A ",A
290 GOSUB 30
300 PRINT "ALL THE DATA READ IN"
310 FOR I=1 TO 20: PRINT "X(";I");=
320 ";X(I);: NEXT I
330 PRINT "THIS IS THE END"
340 END

42
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 ($C030$) 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 0 of memory from $02$ (2) to $14$ (20). $00$ (00) is used to store the relative period (P) between toggling of the speaker and $01$ (01) is used as the memory location for the value of relative duration (D). Both P and D can range in value from $00$ (0) 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 $02$ and then a
delay in time equal to the value in $00$ occurs. This process is repeated until the tone has lasted a relative period of time equal to the duration (value in $01$) 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 32000 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 20 locations of page 0.

After the GOSUB 32000 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 0 and 255 a GOSUB 25 statement is used to initiate the generation of a tone. The values of P and D are placed into $00$ and $01$ 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  ??
0001- FF  ??
0002- AD 30 C0  LDA $C030
0005- 88  DEY
0006- D0 04  BNE $000C
0008- C6 01  DEC $01
000A- F0 00  BEO $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
3,192: POKE 5,136: POKE 6,200
9,71: POKE 7,4: POKE 8,138: POKE
9,1: POKE 10,240
32005 POKE 11,8: POKE 12,202: POKE
13,208: POKE 14,246: POKE 15
166: POKE 16,8: POKE 17,76
9: POKE 18,2: POKE 19,8: POKE
20,96: RETURN

FIGURE 2. BASIC "POKES"

25 POKE 8,F: POKE 1,0: 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 C00 to FFF (hex). Equivalent addresses for the ROM subroutines will be in italic type face.
High-Resolution Operating Subroutines

INIT  Initializes High-Resolution Graphics mode.

From BASIC: CALL 3072 (or CALL -12288)

From machine language: JSR $C00 (or JSR $D00)

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 3086 (or CALL -12274)

From machine language: JSR $C0E (or JSR $D0E)

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

PLOT  Plots a point on the screen.

From BASIC: CALL 3780 (or CALL -11580)

From machine language: JSR $C7C (or JSR $D7C)

This subroutine plots a single point on the screen. The X and Y coordinates 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
High-Resolution Operating Subroutines

PLOT (continued)
(top of screen) to 159 (bottom of screen) and is passed in
location 802 or the A-register; but the X (horizontal) coordinate
can range from 0 (left side of screen) to 279 (right side of screen)
and must be split between locations 800 (X MOD 256) and 801
(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: 0 is BLACK,
85 ($55) is GREEN, 170 ($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 $D026)

   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 conjunction 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.
High-Resolution Operating Routines

**LINE**  Draws a line on the screen.

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

*From machine language: JSR $C95 (or JSR $D995)*

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.

**SHAPE**  Draws a predefined shape on the screen.

*From BASIC: CALL 3805 (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 scaling factor and the rotation factor. The scaling factor determines the relative size of the shape. A scaling factor of
SHAPE (continued)

I 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 $806$ 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 0 will cause the shape to be drawn right-side up, where a rotation factor if 16 will draw the shape rotated $90^0$ clockwise, etc. The rotation factor is passed in location $807$ 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 00 byte, it is finished.

The three sections are arranged in a byte like this:

```
0 0 D D P P D D
Section 3  Section 2  Section 1
```

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 $\emptyset\emptyset$ (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 $90^\circ$ 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 $\emptyset\emptyset$ 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 hexadecimal 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: \[ \begin{array}{c c c}
\emptyset & \emptyset & \emptyset \\
0 & 0 & 0 \\
1 & 0 & 0 \\
2 & 0 & 0 \\
3 & 0 & 0 \\
4 & 0 & 0 \\
5 & 0 & 0 \\
6 & 0 & 0 \\
7 & 0 & 0 \\
8 & 0 & 0 \\
9 & 0 & 0 \\
\end{array} \]

Figure 1.

This vector cannot be a plot vector or a move up (↑).

Hex-decimal Codes

<table>
<thead>
<tr>
<th>( B )</th>
<th>( A )</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0</td>
</tr>
<tr>
<td>0 0 0 1</td>
<td>0 0 0 1</td>
<td>1</td>
</tr>
<tr>
<td>0 0 1 0</td>
<td>0 0 1 0</td>
<td>2</td>
</tr>
<tr>
<td>0 1 0 0</td>
<td>0 1 0 0</td>
<td>3</td>
</tr>
<tr>
<td>0 1 0 1</td>
<td>0 1 0 1</td>
<td>4</td>
</tr>
<tr>
<td>0 1 1 0</td>
<td>0 1 1 0</td>
<td>5</td>
</tr>
<tr>
<td>0 1 1 1</td>
<td>0 1 1 1</td>
<td>6</td>
</tr>
<tr>
<td>1 0 0 0</td>
<td>1 0 0 0</td>
<td>7</td>
</tr>
<tr>
<td>1 0 1 0</td>
<td>1 0 1 0</td>
<td>8</td>
</tr>
<tr>
<td>1 1 0 0</td>
<td>1 1 0 0</td>
<td>9</td>
</tr>
</tbody>
</table>

Figure 2.

\( \emptyset \) denotes end of vector table.
REM HIRES DEMO-BASIC LISTING

I LIST
1 INIT=3672; CLEAR=3866; POSN=5761
 :PLOT=3780; LINE=3706; SHAPE=3065; FND=5667; SINBTL=5040
5 D I N (10), Y(10)

10 TEXT: CALL 336; VTB 4: TRB: TRB
:PRINT "*** 16K APPLE II ***"
:PRINT "*** HIGH RESOLUTION GRAPHICS DEMO ***": PRINT
15 PRINT "1 RANDOM LINE DRAW AT BASIC SPEED"
:PRINT "2 RANDOM SHAPE PROJECTED INTO CORNER"
20 PRINT "3 CHRIS' HUNDFOLLY"
:PRINT "4 RANDOM SHAPE SPIRALING INTO POINT": PRINT "5 SPIROGRAPH"
25 PRINT "6 HI-RES DONUT": PRINT "7 RANDOM WAVE FORM": PRINT "8 SURF OF TWO SINE WAVES"
30 PRINT "PRINT ANY KEY FOR NEW DEMO": PRINT "TYPE 'CONTROL C'
:RETURN BUTTON THEN "EXIT AND RETURN BUTTON TO STOP"

50 PRINT: INPUT " WHICH DEMO DO YOU WANT "#:XI
90 IF XI(1)=OR XI(8)=THEN 10: CALL INIT: GOTO 100;XI
100 CALL INIT:XI=0;Y=0: GOSUB 2000
:POKE 812,255: CALL PLOT
(300) THEN POKE 23,;PEEK (23)+ RND (3)+1 MOD 465: GOSUB 2000: GOTO 110
200 GOSUB 1000: X= RND (2)+279;Y= RND (2)+159: CALL PLOT: FOR
I=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
530 IF RND (566)/C THEN POKE 29
, RND (4)+55; Y=Y+YDIR; B: IF
Y=0 AND Y<160 THEN 518; T0 R=1
:YDIR:=Y; Y:=1; IF X<0 THEN Y:=Y+
310: GOSUB 2000: GOTO 518
600 POKE -16392; B: POKE 768,5; POKE 759,8: POKE 800,149: POKE 801
,8: POKE 802,0: POKE 804,0: POKE 805,3: POKE 812,255: CALL PLOT
610 FOR R=0 TO 4160: POKE 807,R MOD
64: POKE 808,2+6* NOT (R MOD 65): CALL SHAPE: NEXT R: GOSUB
3000: GOTO 610
700 J= RND (10); RND (10); X= RND
(3)+ RND (31); RND (60); L= RND (5)+8: PRINT "FREQ=": "
;j+l): FREQUENCY=":K
710 GOSUB 4000: GOSUB 3000: GOTO
720
730 INPUT " REL FREQ #:",J: INPUT
" REL FREQ #:",K: INPUT "MODE (0
=50Lb, 1=0PINT=)":L
810 GOSUB 4000: GOSUB 3000: GOTO
800
800 CALL CLEAR: POKE 812, RND
(3)+RND (3)+1 MOD 465: GOSUB
2000: PEEK (23)+RND (2): FOR I=1 TO RX(I)+1: RND
(160);X(I)= RND (160); NEXT
I
1100 X=X(I)+Y= Y(I): GOSUB 2000: RETURN
2000 POKE 800,X MOD 256; POKE 801,
X+255; POKE 802,Y RETURN
3000 IF PEAK (-16394)/X2 THEN RETURN
POKE -16368; B: POP : GOTO
10
4000 CALL INIT: POKE 812,255;R=0
;B=0: FOR I=1 TO 279;R=R+J,
MOD 256;B=(B+K) MOD 255
;Y= POKE (SINBTL*A) PEAK (SINBTL+ B)+5/16
4010 POKE 800,I MOD 256; POKE 801,
I+255; POKE 804,1: CALL LINE-
64 NOT I OR L); NEXT I: RETURN
210 X(I)=X(I)-X(I)*9/10+X(I)Y(I)=(
Y(I)-Y)+9/10+Y: NEXT I: GOSUB
3000: GOTO 200
300 CALL INIT:X= RND (24)+10+K
:Y= RND (14)+10+28: POKE 812
, RND (3)+55: GOSUB 2000
:CALL PLOT
310 IF RND (1000)>I THEN 300: IF
NOT RND (200) THEN POKE 29
, RND (4)+55
320 X=+X; RND (3)+1+55;Y=Y+ (-
RND (3)+1):IF X(1) OR
X(1)+279 OR Y(1) OR Y(1)+159 THEN
LINE: GOSUB 3000: GOTO 310
400 GOSUB 1000: POKE 812, RND
(3)+55: CALL PLOT
410 FOR J=1 TO 25: FOR I=1 TO R:
POKE 800,X(I) MOD 256: POKE
801,Y(I): CALL LINE
420 X=(X(I)-30+(Y(I)-30)/3)+9/10
+(8*Y(I)+(Y(I)-8-(X(I)-10)
+y(1)):X(I)=X; NEXT I,
J: GOSUB 3000: GOTO 400
500 CALL INIT: POKE 800,8: CALL
PLOT:XI=0:Y=0;XDIR=I:YDIR=I:1
5:1=3:2=8
510 POKE 800,8: POKE 801,8: POKE
802,Y: CALL LINE: POKE 800,
(279-X) MOD 256: POKE 801,:X
24: POKE 802,159: CALL LINE:
POKE 800,234: POKE 801,1: POKE
802,159: CALL LINE
515 IF RND (500) THEN 520;R=1: RND
(13)+2+ RND (11)+C=-R+ RND
(7)
520 POKE 800,X MOD 256: POKE 801,
:Y+255: POKE 802,8: CALL LINE:
=X+X++8: IF X(0) RND AND X(256)
THEN 530;XDIR=XRDIR: X=X: IF
X(0) THEN X=X+558
54
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
125 COLOR=W/(3*(I+J)+1)*12
130 PLOT I,K: PLOT K,I: PLOT 40 -I,40-K
135 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
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 CY0) OR (VYO= A85 (Y) AND X=0) THEN V=4: RND (9)
140 IF X=0 THEN VYO= A85 (Y)
145 R=39-R: B=39-B: C=-C
150 IF PEEK (-16286)>>127 AND S#5 THEN S=5
155 IF PEEK (-16287)>>127 AND S#0 THEN S=5
160 GOTO 65
165 COLOR=8: PLOT X,C,Y
170 COLOR=15: PLOT X,Y+Y,(Y+Y)-1 AND Y+Y(40)
175 FOR T=1 TO 75: M=PEEK (-16336)
180 IF X=CO THEN GOSUB 235: NEXT T
185 PRINT 3: PRINT S
190 COLOR=0: PLOT X,C,Y
195 IF S=15 OR S=0 THEN 260
200 COLOR=0: PLOT X,Y+Y,(Y+Y)-1 AND Y+Y(40)
205 FOR T=1 TO 75: IF T MOD 500 THEN 210: IF PEEK (-16286) >>127 AND S#5 THEN S=5
210 IF PEEK (-16287)>>127 AND S#0 THEN S=5
215 GOSUB 235: NEXT T
220 IF H AND CY0 THEN 136
225 PP=PP(CY0)
230 GOTO 65

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 A through P 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 3,40: POKE 4,192: POKE 5,163: POKE 6,0
   : POKE 7,32: POKE 8,160: POKE 9,252: POKE 10,165: POKE 11
   :1, POKE 12,230: POKE 13,4
10 POKE 14,196: POKE 15,24: POKE 16,248: POKE 17,5: POKE 18,
   :190: POKE 19,1: POKE 20,76: POKE 21,2: POKE 22,8: POKE
   23,96
15 DIM B$(40): TEXT : CALL -936
   : GOTO 90
20 CALL -936: GOTO 90
25 R=L:LEN(B$): FOR Z=1 TO R: GOSUB
65: PRINT B$(Z,2);: NEXT Z:
   : GOSUB 70: RETURN
30 B$= "***************
   ***************": 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=28:TON=28: GOSUB 85: RETURN

65 KK=19:TON=19: GOSUB 85: RETURN

70 KK=28:TON=58: GOSUB 85:KK=30
   : TON=98: GOSUB 85: RETURN
75 KK=28:TON=20: GOSUB 85: RETURN
80 KK=8:TON=258: GOSUB 85:KK=9
   : TON=250: GOSUB 85: RETURN
85 POKE 1,TON MOD 256: POKE 24
   : ,TON/256+1: POKE 0,1:CALL 2:RETURN
90 GOSUB 30: GOSUB 25: PRINT :
   : TAB 5: GOSUB 35: GOSUB 25
   : PRINT: GOSUB 20: 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$= "abcdefhijklmnop": CALL
   : -936
115 FOR Z=0 TO 15: COLOR=Z: PRINT
   : Z=2+4,39: VTAB 21: GOSUB 75
   : : TAB Z=2+5: PRINT B$(Z+1,2+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= POL (1)+39/255*X= POL (0
   :)+39/255: VTAB 24: TAB 1: PRINT
   : "CURSOR POSITION: X=":X;" Y=
   : "Y;":
130 IF PEEK (-16384))=127 THEN 145
   : IF 81=X AND Y=8 THEN 125
   : COLOR=C2: PLOT X,Y: IF
   : NOT FLAG THEN 135: COLOR=C1:
   : PLOT X,Y

135 C2= SCRNX,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))=165 THEN 155
   : FLAG=0: POKE -16368,0: POKE
   34,20: COLOR=0: MLIN 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

58
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
402 (APPLE COMPUTER)
10 DIM X(3),C(3),D(3),R(3),W(3),X(3)
8):X(I)=2*X(I-2)-12*X(I-3)+1:X(I)
4)=13*X(I-2)+3*X(I-6)+9*X(I-7)+15
11:X(I)=5*X(I)*"GORYWONX"

80 TEXT : CALL -936; PRINT *

WELCOME;
ME TO THE GAME OF MASTERMIND!

YOUR OBJECT IS TO GUESS 5 COLOR
5 (WHICH)
90 PRINT "I WILL MAKE UP) IN THE MINIMUM NUMBER OF GUESSES. THE RE
2 ARE EIGHT DIFFERENT COLORS TO CHOOSE FROM."
40 PRINT *

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

*: CALL -304; TAB 7; PRINT "HIT ANY KEY TO BEGIN PLAY"

100 CALL -300; IF Peek (-16384)
(132 THEN 100: POKE -16388
0: CE : PRINT FOR I=1 TO
8: W(I)= RND (8)+1; COLOR=W
1): HLN=1+4-2;I AT 39: PRINT "xX(I),1); NEXT I
110 TRY=0: PRINT "LETTER
1: ARROW KEYS FOR ADVANCE AND BACK:
CK": PRINT "HIT RETURN TO ACC
EPT GUESS #";

200 Y=TRY+2 MOD 36+1;TRY=TRY+1:
TAB 32: PRINT TRY;": COLOR=
0: HLN=6;39 AT Y:FLASH=1: FOR
N=1 TO 5:R(N)=N: GOSUB 1000
1: NEXT N=1
300 FOR WAIT=1 TO 19: KEY= PEEK
(-16384): IF KEY(132 THEN 310
: POKE -16388,0:FLASH=1: FOR
I=1 TO 3: IF KEY(N) = S$N(N(I))
) THEN NEXT I: IF I=9 THEN
310:R(N)=1:KEY=149
310:GOSUB 1000: IF KEY=141 THEN
400: IF KEY=136 AND M=1 OR
KEY=149 AND M=6 THEN M=M+KEY/
5-28: NEXT WAIT;FLASH=1-FLASH:
GOTO 300
400 COLOR=15:M=0: FOR I=1 TO 5:
8(I)=C(I)+1:J=1: GOSUB 2000: NEXT
1: IF N=5 THEN 500: COLOR=5
: FOR J=1 TO 5: FOR I=1 TO
5: GOSUB 2000: NEXT 1,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>4 THEN
PRINT "POOR"; PRINT "); CALL
-304: TAB 5: PRINT "HIT ANY KEY
TO PLAY AGAIN"; GOTO 100
1000 IF N=6 THEN RETURN : COLOR=
X(RN(N) )=FLASH: HLN=N=4-2,N=4 AT Y: RETURN
2000 IF R(I) =X(D(I)) THEN RETURN :
H=H+1: PLOT 21+H,Y: PRINT "xX(I)=8@D(I)+5: RETURN
BIORHYTHM PROGRAM

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 6,8 : POKE 7,32; POKE 8,168; POKE 9,252; POKE 10,165; POKE 11,1; POKE 12,208; POKE 13,4 10 POKE 14,196; POKE 15,24; POKE 16,248; POKE 17,5; POKE 18,198; POKE 19,1; POKE 20,24; POKE 21,2; POKE 22,0; POKE 23,36 15 GOTO 85 20 TT=3: GOSUB 30: RETURN 25 PRINT "**************************
****************************
*****************************

30 KK=8:TN=580: GOSUB 45: RETURN
35 KK=8:TN=258: GOSUB 45: RETURN
45 POKE 1,178 MOD 256: POKE 24 ,178+1: POKE 0,KK: CALL 2: RETURN
50 A=19-(P#B(1)/100)*X(P#100<
C(1))+X(P#100+C(1))/100*B(1)
34*(1+w(P#188-C(1))/100+b(1))
188-34*(C(1))/100+b(1));
65 KK=7:TN=18: GOSUB 70: RETURN
70 POKE 1,178 MOD 256: POKE 24 ,
178+1: POKE 0,KK: CALL 2:
RETURN
75 GOSUB 60: INPUT "DATE (M,D,Y) " 
,M,D,Y;Y=T+(Y100)/10000
59+824A/4-A/400MM+31-M+3-M/
7-M/5-3*(M<2):0: IF N<8 THEN 
N=N/21525: RETURN
85 DIM H(10),B(3),C(3),B(3),B(1)=348+B(2)=86+B(3)=242+C(1)=575+C(2)=700+C(3)=85+B(1)=231+B(2)=28
90 BY(3)=33: TEXT ; CALL -936: 
POKE 34,29: GOSUB 28: GOSUB
25: GOSUB 20: PRINT : TAB 10 : 
PRINT "APPLE II BIORHYTHM (4K) 
: TAB 15: PRINT
95 GOSUB 25: TAB 5: PRINT "COPYRIGHT 
T 1977 APPLE COMPUTER INC."
: POKE 34,24: VTAB 24
100 GOSUB 60: INPUT "NAME 
",N#
: VTAB 22: PRINT H#: VTAB 24 : 
PRINT "BIRTH 
: VTAB 22: TAB 21: PRINT "BIRTH 
DATE 
:VTAB 24: N=H: CALL -868
105 PRINT "FORECAST 
: GOSUB 75
: N=N-H1: IF NO THEN N=N*2152
: VTAB 23: TAB 10: PRINT "FORECAST 
ST RATE 
: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 
" PE ": VTAB 24
120 VTAB 23: PRINT "DAYS LIVED 
: VTAB 23: FOR I=1 TO 3: COLOR=1+
1+6*(I-2)+8*(I-3): VLIN 
0,39 AT 33+I: VTAB 24
125 FOR X=0 TO 31: P=(X MOD BV(1)
+X) MOD BV(1): 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 maze is 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.
PROGRAM LISTING: DRAGON MAZE

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 DUMP INTO THEM."
6 PRINT "TO MOVE, YOU HIT 'R' FOR RIGHT, 'L' FOR LEFT, 'U' FOR UP, AND 'D' FOR DOWN. DO NOT HIT RETURN."
7 PRINT "THE OBJECT IS FOR YOU (THE GREEN DOT)"
8 PRINT "TO GET TO THE DOOR ON THE RIGHT SIDE."
9 PRINT "BEFORE THE DRAGON (THE RED DOT) EATS YOU."
10 PRINT "YOU."
11 PRINT "BEWARE!!!!!! SOMETIMES THE DRAGON GETS REAL MAD, AND CLIMBS OVER A WALL."
12 PRINT "BUT MOST OF THE TIME, HE CAN'T GO OVER."
13 PRINT "AND HAS TO GO AROUND."
14 PRINT "HINT: YOU CAN OFTEN TELL WHERE A WALL IS, EVEN BEFORE YOU CAN SEE IT, BY"
15 PRINT "THE FACT THAT THE DRAGON CAN'T GET"
16 PRINT "THROUGH IT!"
17 PRINT "TYPE 'GO' TO BEGIN."
18 PRINT "INPUT R#
19 PRINT "GR : COLOR=15"
20 PRINT "COLOR=#"
21 PRINT "CALL 936: PRINT "DRAGON MAZE"
22 PRINT "PRINT "GARY J. SHAN"
23 PRINT "NON."
24 PRINT "FOR I=0 TO 39 STEP 3: VLIN 0,39 AT I: HLIN 0,39 AT I: NEXT I"
25 PRINT "I"
26 PRINT "COLOR=#"
27 PRINT "S=1000"
28 PRINT "1000 DIM M(169),T(169)"
29 PRINT "1001 FOR I=1 TO 163: T(I)=0: NEXT I"
30 PRINT "1010 FOR I=1 TO 163: M(I)=I: NEXT I"
31 PRINT "1020 X=RND(13)+1: Y=RND(13)+1: D=169"
32 PRINT "1030 IF C=1 THEN 1200"
33 PRINT "1040 R=0: D=0: L=0: U=0: I(X)=X+13*(Y-1)
34 PRINT "M(K)=ABS(M(K))+C-1"
35 PRINT "1050 IF X=13 THEN 1660: R=M(K+1)"
36 PRINT "0"
37 PRINT "1060 IF Y=13 THEN 1670: D=M(K+13)"
38 PRINT "0"
39 PRINT "1070 IF X=1 THEN 1680: L=M(K-1)"
40 PRINT "0"
41 PRINT "1080 IF Y=1 THEN 1690: U=M(K-13)"
42 PRINT "0"
43 PRINT "1090 Q=R+D+L+U"
44 PRINT "1100 IF (Q<3 AND RND(18)<2) OR Q=0 THEN 1170"
45 PRINT "1110 OR= RND(4)"
46 PRINT "1120 GOTO 1130+10*OR"
47 PRINT "1130 IF NOT R THEN 1110: M(K)=M(K)+1: X=X+1"
48 PRINT "1135 VLIN 3*X-2,3*Y-1 AT 3*(X-1)"
49 PRINT "1140 GOTO 1835"
50 PRINT "1145 YLIN 3*X-2,3*Y-1 AT 3*(Y-1)"
51 PRINT "1150 GOTO 1835"
52 PRINT "1155 VLIN 3*X-2,3*Y-1 AT 3*X"
53 PRINT "1160 GOTO 1835"
54 PRINT "1165 HLIN 3*X-2,3*Y-1 AT 3*Y: GOTO 1835"
55 PRINT "1170 X= RND(13)+1: Y= RND(13)+1"
56 PRINT "1180 IF M(X+13*(Y-1))=0 THEN 1170"
57 PRINT "1190 C=C+1: GOTO 1835"
58 PRINT "1200 GOSUB 5088: PRINT "THE MAZE IS READY."
59 PRINT "1205 GR : COLOR=15"
60 PRINT "1210 VLIN 0,39 AT O: VLIN 0,39 AT 39: HLIN 0,39 AT O: HLIN 0,39 AT 39"
61 PRINT "1220 X=1: R= RND(13)+1: COLOR=0: PLOT 3*X-2,3*Y-2"
DRAGON MAZE cont.

1225 HX=3*HX-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 GQ=3*GQ-2;GQ=GQ+2
1270 RD=1
1500 K= PEEK (-16364): IF K<128 THEN
  1500
1510 PEEK -16368,0
1515 Q=K: GOSUB 70000;K=0
1516 IF SX=K AND SY=0 THEN 2000
1520 IF K= ASC("R") THEN 2000
1530 IF K= ASC("L") THEN 2500
1540 IF K= ASC("U") THEN 2000
1550 IF K= ASC("D") THEN 2500
1560 GOSUB 5000: GOTO 1500
2000 DX=1:DY=0
2010 IF MX=13*(Y-1)) MOD 10 THEN
  4000
2020 FX=3*FX-2;FY=3*FY-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:
  2060 PLOT HX+K,HY+L: NEXT L,K: COLOR=2
  2060: FOR K=0 TO 1: FOR L=0 TO 1:
1: PLOT FX+K,FY+L: NEXT L,K:
  HX=FX;HY=FY
2110 NEXT I
2115 X=X+10;Y=Y+1
2116 IF X=13 AND Y=WY THEN 6000
2120 GOTO 1500
2300 DX=-1:DY=0
2510 IF MX=13*(Y-1)-1) MOD 10 THEN
  4100
2520 GOTO 2020
3000 DX=0: DY=-1
3010 IF MX=13*(Y-2))//10 THEN 4200
3020 GOTO 2020
3030 DX=0: DY=1
3050 IF MX=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 VLIN 3*(X-1),3*X AT 3*(Y-1)
4230 GOTO 1500
4300 GOSUB 5000
4310 COLOR=15
4320 VLIN 3*(X-1),3*X AT 3*Y
4330 GOTO 1500
5000 S=5:1: FOR I=1 TO 20: R= PEEK (-16336)+ PEEK (-16336)+ PEEK (-16336)+ PEEK (-16336): NEXT 1: RETURN
6000 PRINT "YOU WIN!"
6010 GOSUB 5000: GOSUB 5000: GOSUB 5000
6020 PRINT "SCORE=D;S+3"
6030 END

7000 IF X=SK THEN 7005: IF Y=SY THEN
7050
7001 IF X=SK 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
7100 DX=1:DY=0
7200 COLOR=8
7222 RX=3*RX+2;RY=3*SY-2
7223 FOR I=1 TO 3: RX=RX+1:RY=RY+DY
7224 COLOR=0
7225 FOR K=0 TO 1: FOR L=0 TO 1:
  7225 PLOT RX+K,RY+L: NEXT L,K: COLOR=
  RO: FOR K=0 TO 1: FOR L=0 TO
  1: PLOT RX+K,RY+L: NEXT L,K:
  RX=RX+DY:RY
7300 NEXT I
7305 SX=SK+DX;SY=SY+DY
7400 T(SX)=13*(SY-1))=T(SX)=13*(SY-1))//1
7405 RETURN
7500 IF SY=13 THEN 7100: IF T(SX)=
13*(SY-1))//9 THEN 7050: IF
M(SX+13*(SY-1))//10 THEN 7100
7600 DX=0: DY=1: GOTO 7000
7100 IF SX=1 THEN 7150: IF T(SX)=
13*(SY-1))//9 THEN 7110: IF
M(SX+13*(SY-1)) MOD 10 THEN
7150
DRAGON MAZE cont.

7110 DX=-1:DY=0: GOTO 7020
7150 IF SY=1 THEN 7065: IF T(SX+
13*(SY-1))<9 THEN 7160: IF
M(SX+13*(SY-1)-13)/10 THEN
7095
7160 DX=0:DY=-1: GOTO 7020
8000 GOSUB 5000: GOSUB 5000: GOSUB
5000: GOSUB 5000: PRINT "THE DRAG
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 (control C). NOTE: "adrs" is a four digit hexadecimal number and "data" is a two digit hexadecimal number. Remember to press "return" button at the end of each line.

<table>
<thead>
<tr>
<th>Command Format</th>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Examine Memory</td>
<td></td>
<td></td>
</tr>
<tr>
<td>adrs</td>
<td>*C0F2</td>
<td>Examines (displays) single memory location of (adrs)</td>
</tr>
<tr>
<td>adrs1.adrs2</td>
<td>*1024.1048</td>
<td>Examines (displays) range of memory from (adrs1) thru (adrs2)</td>
</tr>
<tr>
<td>(return)</td>
<td>* (return)</td>
<td>Examines (displays) next 8 memory locations.</td>
</tr>
<tr>
<td>.adrs2</td>
<td>*.4096</td>
<td>Examines (displays) memory from current location through location (adrs2)</td>
</tr>
<tr>
<td>Change Memory</td>
<td></td>
<td></td>
</tr>
<tr>
<td>adrs:data</td>
<td>*A256:EF 20 43</td>
<td>Deposits data into memory starting at location (adrs).</td>
</tr>
<tr>
<td>data data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>:data data</td>
<td>*:F0 A2 12</td>
<td>Deposits data into memory starting after (adrs) last used for deposits.</td>
</tr>
<tr>
<td>Move Memory</td>
<td></td>
<td></td>
</tr>
<tr>
<td>adrs1&lt;adrs2.</td>
<td>*100&lt;B010.B410M</td>
<td>Copy the data now in the memory range from (adrs2) to (adrs3) into memory</td>
</tr>
<tr>
<td>adrs3M</td>
<td></td>
<td>locations starting at (adrs1).</td>
</tr>
<tr>
<td>Verify Memory</td>
<td></td>
<td></td>
</tr>
<tr>
<td>adrs1&lt;adrs2.</td>
<td>*100&lt;B010.B410V</td>
<td>Verify that block of data in memory range from (adrs2) to (adrs3) exactly</td>
</tr>
<tr>
<td>adrs3V</td>
<td></td>
<td>matches data block starting at memory location (adrs1) and displays</td>
</tr>
<tr>
<td></td>
<td></td>
<td>differences if any.</td>
</tr>
<tr>
<td>Command Format</td>
<td>Example</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>Cassette I/O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>adrs1.adrs2R</td>
<td>*300.4FFR</td>
<td>Reads cassette data into specified memory (adrs) range. Record length must be same as memory range or an error will occur.</td>
</tr>
<tr>
<td>adrs1.adrs2W</td>
<td>*800.9FFW</td>
<td>Writes onto cassette data from specified memory (adrs) range.</td>
</tr>
<tr>
<td>Display</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>*I</td>
<td>Set inverse video mode. (Black characters on white background)</td>
</tr>
<tr>
<td>N</td>
<td>*N</td>
<td>Set normal video mode. (White characters on black background)</td>
</tr>
<tr>
<td>Dis-assembler</td>
<td></td>
<td></td>
</tr>
<tr>
<td>adrsL</td>
<td>*C800L</td>
<td>Decodes 20 instructions starting at memory (adrs) into 6502 assembly mnemonic code.</td>
</tr>
<tr>
<td>L</td>
<td>*L</td>
<td>Decodes next 20 instructions starting at current memory address.</td>
</tr>
<tr>
<td>Mini-assembler</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Turn-on)</td>
<td>*F666G</td>
<td>Turns-on mini-assembler. Prompt character is now a &quot;.&quot; (exclamation point).</td>
</tr>
<tr>
<td>$(monitor command)</td>
<td>!$C800L</td>
<td>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.</td>
</tr>
<tr>
<td>adrs:(6502 MNEMONIC instruction)</td>
<td>!C910:STA 23FF</td>
<td>Assembles a mnemonic 6502 instruction into machine codes. If error, machine will refuse instruction, sound bell, and reprint line with up arrow under error.</td>
</tr>
<tr>
<td>Command Format</td>
<td>Example</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>(space) ((6502 \text{ mnemonic instruction}))</td>
<td><code>! STA 01FF</code></td>
<td>Assembles instruction into next available memory location. (Note space between <code>!</code> and instruction)</td>
</tr>
<tr>
<td>(TURN-OFF)</td>
<td><code>(Reset Button)</code></td>
<td>Exits mini-assembler and returns to system monitor.</td>
</tr>
</tbody>
</table>

### Monitor Program Execution and Debugging

- **adrsG**
  - `*300G`
  - Runs machine level program starting at memory (adrs).
- **adrsT**
  - `*300T`
  - Traces a program starting at memory location (adrs) and continues trace until hitting a breakpoint. Break occurs on instruction `00` (BRK), and returns control to system monitor. Opens 6502 status registers (see note 1).
- **adrsS**
  - `*C050S`
  - Single steps through program beginning at memory location (adrs). Type a letter `S` for each additional step that you want displayed. Opens 6502 status registers (see Note 1).
- **(Control E)**
  - `*EC`
  - Displays 6502 status registers and opens them for modification (see Note 1).
- **(Control Y)**
  - `*YC`
  - Executes user specified machine language subroutine starting at memory location (3F8).

**Note 1:**

6502 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 = 00  P = 32  S = F2

*FF  Changes A register only
*FF 00 33  Changes A, X, and Y registers

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

### Hexadecimal Arithmetic

- **data1+data2**
  - `*78+34`
  - Performs hexadecimal sum of data1 plus data2.
- **data1-data2**
  - `*AE-34`
  - Performs hexadecimal difference of data1 minus data2.
## Command Format

<table>
<thead>
<tr>
<th>Command Format</th>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set Input/Output Ports</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(X) (Control P)</td>
<td>*5pC</td>
<td>Sets printer output to I/O slot number (X). (see Note 2 below)</td>
</tr>
<tr>
<td>(X) (Control K)</td>
<td>*2kC</td>
<td>Sets keyboard input to I/O slot number (X). (see Note 2 below)</td>
</tr>
</tbody>
</table>

**Note 2:**
Only slots 1 through 7 are addressable in this mode. Address Ø (Ex: ØpC or ÖkC) resets ports to internal video display and keyboard. These commands will not work unless Apple II interfaces are plugged into specified I/O slot.

## Multiple Commands

<table>
<thead>
<tr>
<th>Command Format</th>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>*1ggl 4ggg AFFT</td>
<td>Multiple monitor commands may be given on same line if separated by a &quot;space&quot;.</td>
</tr>
<tr>
<td></td>
<td>*LLLLL</td>
<td>Single letter commands may be repeated without spaces.</td>
</tr>
</tbody>
</table>
**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 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.

<table>
<thead>
<tr>
<th>CHARACTER</th>
<th>DESCRIPTION OF ACTION</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RESET key</strong></td>
<td>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 &quot;*&quot; (asterisk) and a bell. Hitting RESET key does NOT destroy existing BASIC or machine language program.</td>
</tr>
<tr>
<td><strong>Control B</strong></td>
<td>If in System Monitor (as indicated by a &quot;*&quot;), a control B and a carriage return will transfer control to BASIC, scratching (killing) any existing BASIC program and set HIMEM: to maximum installed user memory and LOMEM: to 2048.</td>
</tr>
<tr>
<td><strong>Control C</strong></td>
<td>If in BASIC, halts program and displays line number where stop occurred*. Program may be continued with a CON command. If in System Monitor, (as indicated by &quot;*&quot;), control C and a carriage return will enter BASIC without killing current program.</td>
</tr>
<tr>
<td><strong>Control G</strong></td>
<td>Sounds bell (beeps speaker)</td>
</tr>
<tr>
<td><strong>Control H</strong></td>
<td>Backspaces cursor and deletes any overwritten characters from computer but not from screen. Apply supplied keyboards have special key &quot;&lt;&quot; on right side of keyboard that provides this functions without using control button.</td>
</tr>
<tr>
<td><strong>Control J</strong></td>
<td>Issues line feed only</td>
</tr>
<tr>
<td><strong>Control V</strong></td>
<td>Compliment to $h^C$. Forward spaces cursor and copies overwritten characters. Apple keyboards have &quot;&lt;&quot; key on right side which also performs this function.</td>
</tr>
<tr>
<td><strong>Control X</strong></td>
<td>Immediately deletes current line.</td>
</tr>
</tbody>
</table>

* If BASIC program is expecting keyboard input, you will have to hit carriage return key after typing control C.
<table>
<thead>
<tr>
<th>CHARACTER</th>
<th>DESCRIPTION OF ACTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>A_E</td>
<td>Move cursor to right</td>
</tr>
<tr>
<td>B_E</td>
<td>Move cursor to left</td>
</tr>
<tr>
<td>C_E</td>
<td>Move cursor down</td>
</tr>
<tr>
<td>D_E</td>
<td>Move cursor up</td>
</tr>
<tr>
<td>E_E</td>
<td>Clear text from cursor to end of line</td>
</tr>
<tr>
<td>F_E</td>
<td>Clear text from cursor to end of page</td>
</tr>
<tr>
<td>G_E</td>
<td>Home cursor to top of page, clear text to end of page.</td>
</tr>
</tbody>
</table>
## Special Controls and Features

### Display Mode Controls

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

### TEXT Mode Controls

<table>
<thead>
<tr>
<th>Hex</th>
<th>BASIC Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0020</td>
<td>90 POKE 32,L1</td>
<td>Set left side of scrolling window to location specified by L1 in range of 0 to 39.</td>
</tr>
<tr>
<td>0021</td>
<td>100 POKE 33,W1</td>
<td>Set window width to amount specified by W1. W1+W1&lt;40. W1&gt;0</td>
</tr>
<tr>
<td>0022</td>
<td>110 POKE 34,T1</td>
<td>Set window top to line specified by T1 in range of 0 to 23</td>
</tr>
<tr>
<td>0023</td>
<td>120 POKE 35,B1</td>
<td>Set window bottom to line specified by B1 in the range of 0 to 23. B1&gt;T1</td>
</tr>
<tr>
<td>0024</td>
<td>130 CH=PEEK(36)</td>
<td>Read/set cursor horizontal position in the range of 0 to 39. If using TAB, you must add &quot;1&quot; to cursor position; Ex. 140 and 150 perform identical function.</td>
</tr>
<tr>
<td></td>
<td>140 POKE 36,CH</td>
<td></td>
</tr>
<tr>
<td></td>
<td>150 TAB(CH+1)</td>
<td></td>
</tr>
<tr>
<td>0025</td>
<td>160 CV=PEEK(37)</td>
<td>Similar to above. Read/set cursor vertical position in the range 0 to 23.</td>
</tr>
<tr>
<td></td>
<td>170 POKE 37,CV</td>
<td></td>
</tr>
<tr>
<td></td>
<td>180 VTAB(CV+1)</td>
<td></td>
</tr>
<tr>
<td>0032</td>
<td>190 POKE 50,127</td>
<td>Set inverse flag if 127 (Ex. 190)</td>
</tr>
<tr>
<td></td>
<td>200 POKE 50,255</td>
<td>Set normal flag if 255 (Ex. 200)</td>
</tr>
<tr>
<td>FC58</td>
<td>210 CALL -936</td>
<td>(@F) Home cursor, clear screen</td>
</tr>
<tr>
<td>FC42</td>
<td>220 CALL -958</td>
<td>(F) Clear from cursor to end of page</td>
</tr>
<tr>
<td>Hex</td>
<td>BASIC Example</td>
<td>Description</td>
</tr>
<tr>
<td>-----</td>
<td>---------------</td>
<td>-------------</td>
</tr>
<tr>
<td>FC9C</td>
<td>230 CALL -868</td>
<td>(E) Clear from cursor to end of line</td>
</tr>
<tr>
<td>FC66</td>
<td>240 CALL -922</td>
<td>(J) Line feed</td>
</tr>
<tr>
<td>FC70</td>
<td>250 CALL -912</td>
<td>Scroll up text one line</td>
</tr>
</tbody>
</table>

**Miscellaneous**

<table>
<thead>
<tr>
<th>Hex</th>
<th>BASIC Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C030</td>
<td>360 X=PEEK(-16336) 365 POKE -16336,0</td>
<td>Toggle speaker</td>
</tr>
<tr>
<td>C000</td>
<td>370 X=PEEK(-16384)</td>
<td>Read keyboard; if X&gt;127 then key was pressed.</td>
</tr>
<tr>
<td>C010</td>
<td>380 POKE -16368,0</td>
<td>Clear keyboard strobe - always after reading keyboard.</td>
</tr>
<tr>
<td>C061</td>
<td>390 X=PEEK(16287)</td>
<td>Read PDL(0) push button switch. If X&gt;127 then switch is &quot;on&quot;.</td>
</tr>
<tr>
<td>C062</td>
<td>400 X=PEEK(-16286)</td>
<td>Read PDL(1) push button switch.</td>
</tr>
<tr>
<td>C063</td>
<td>410 X=PEEK(-16285)</td>
<td>Read PDL(2) push button switch.</td>
</tr>
<tr>
<td>C058</td>
<td>420 POKE -16296,0</td>
<td>Clear Game I/O AN0 output</td>
</tr>
<tr>
<td>C059</td>
<td>430 POKE -16295,0</td>
<td>Set Game I/O AN0 output</td>
</tr>
<tr>
<td>C05A</td>
<td>440 POKE -16294,0</td>
<td>Clear Game I/O AN1 output</td>
</tr>
<tr>
<td>C05B</td>
<td>450 POKE -16293,0</td>
<td>Set Game I/O AN1 output</td>
</tr>
<tr>
<td>C05C</td>
<td>460 POKE -16292,0</td>
<td>Clear Game I/O AN2 output</td>
</tr>
<tr>
<td>C05D</td>
<td>470 POKE -16291,0</td>
<td>Set Game I/O AN2 output</td>
</tr>
<tr>
<td>C05E</td>
<td>480 POKE -16290,0</td>
<td>Clear Game I/O AN3 output</td>
</tr>
<tr>
<td>C05F</td>
<td>490 POKE -16289,0</td>
<td>Set Game I/O AN3 output</td>
</tr>
<tr>
<td>LABEL</td>
<td>EP2</td>
<td>$</td>
</tr>
<tr>
<td>---------</td>
<td>-----</td>
<td>---</td>
</tr>
<tr>
<td>LOC0</td>
<td>EP2</td>
<td>00</td>
</tr>
<tr>
<td>LOC1</td>
<td>EP2</td>
<td>01</td>
</tr>
<tr>
<td>WNDLFT</td>
<td>EP2</td>
<td>20</td>
</tr>
<tr>
<td>WNDWDT</td>
<td>EP2</td>
<td>21</td>
</tr>
<tr>
<td>WNDTOP</td>
<td>EP2</td>
<td>22</td>
</tr>
<tr>
<td>WNDTLM</td>
<td>EP2</td>
<td>23</td>
</tr>
<tr>
<td>CII</td>
<td>EP2</td>
<td>24</td>
</tr>
<tr>
<td>CV</td>
<td>EP2</td>
<td>25</td>
</tr>
<tr>
<td>GRASL</td>
<td>EP2</td>
<td>26</td>
</tr>
<tr>
<td>GRASH</td>
<td>EP2</td>
<td>27</td>
</tr>
<tr>
<td>BASL</td>
<td>EP2</td>
<td>28</td>
</tr>
<tr>
<td>BASH</td>
<td>EP2</td>
<td>29</td>
</tr>
<tr>
<td>BAS2L</td>
<td>EP2</td>
<td>2A</td>
</tr>
<tr>
<td>BAS2H</td>
<td>EP2</td>
<td>2B</td>
</tr>
<tr>
<td>H2</td>
<td>EP2</td>
<td>2C</td>
</tr>
<tr>
<td>LNNEM</td>
<td>EP2</td>
<td>2C</td>
</tr>
<tr>
<td>RTNL</td>
<td>EP2</td>
<td>2C</td>
</tr>
<tr>
<td>V2</td>
<td>EP2</td>
<td>2D</td>
</tr>
<tr>
<td>RNNEM</td>
<td>EP2</td>
<td>2D</td>
</tr>
<tr>
<td>RTNH</td>
<td>EP2</td>
<td>2D</td>
</tr>
<tr>
<td>MASK</td>
<td>EP2</td>
<td>2F</td>
</tr>
<tr>
<td>CHKSUM</td>
<td>EP2</td>
<td>2F</td>
</tr>
<tr>
<td>FORMAT</td>
<td>EP2</td>
<td>2E</td>
</tr>
<tr>
<td>LASTIN</td>
<td>EP2</td>
<td>2F</td>
</tr>
<tr>
<td>LENGTH</td>
<td>EP2</td>
<td>2F</td>
</tr>
<tr>
<td>SIGN</td>
<td>EP2</td>
<td>2F</td>
</tr>
<tr>
<td>COLOR</td>
<td>EP2</td>
<td>30</td>
</tr>
<tr>
<td>MODE</td>
<td>EP2</td>
<td>31</td>
</tr>
<tr>
<td>INVFIG</td>
<td>EP2</td>
<td>32</td>
</tr>
<tr>
<td>PROMPT</td>
<td>EP2</td>
<td>33</td>
</tr>
<tr>
<td>YSAV</td>
<td>EP2</td>
<td>34</td>
</tr>
<tr>
<td>YSAVI</td>
<td>EP2</td>
<td>35</td>
</tr>
<tr>
<td>CSWL</td>
<td>EP2</td>
<td>36</td>
</tr>
<tr>
<td>CSWH</td>
<td>EP2</td>
<td>37</td>
</tr>
<tr>
<td>KSWL</td>
<td>EP2</td>
<td>38</td>
</tr>
<tr>
<td>KSWH</td>
<td>EP2</td>
<td>39</td>
</tr>
<tr>
<td>PCL</td>
<td>EP2</td>
<td>3A</td>
</tr>
<tr>
<td>PCH</td>
<td>EP2</td>
<td>3B</td>
</tr>
<tr>
<td>XOT</td>
<td>EP2</td>
<td>3C</td>
</tr>
<tr>
<td>A1L</td>
<td>EP2</td>
<td>3C</td>
</tr>
<tr>
<td>A1H</td>
<td>EP2</td>
<td>3D</td>
</tr>
<tr>
<td>A2L</td>
<td>EP2</td>
<td>3E</td>
</tr>
<tr>
<td>A2H</td>
<td>EP2</td>
<td>3F</td>
</tr>
<tr>
<td>A3L</td>
<td>EP2</td>
<td>40</td>
</tr>
<tr>
<td>A3H</td>
<td>EP2</td>
<td>41</td>
</tr>
<tr>
<td>A4L</td>
<td>EP2</td>
<td>42</td>
</tr>
<tr>
<td>A4H</td>
<td>EP2</td>
<td>43</td>
</tr>
<tr>
<td>A5L</td>
<td>EP2</td>
<td>44</td>
</tr>
<tr>
<td>A5H</td>
<td>EP2</td>
<td>45</td>
</tr>
</tbody>
</table>

---

TITLE  "APPLE II SYSTEM MONITOR"
ACC EQU $45
VSEG EQU $46
YRPG EQU $47
STATUS EQU $48
SPTW EQU $49
RNDL EQU $4A
FNDH EQU $4B
ACL EQU $50
ACH EQU $51
XINXL EQU $52
XTWMP EQU $53
AUXL EQU $54
AUXH EQU $55
PICK EQU $59
IR EQU $200
USADR EQU $30FB
DIC EQU $39FB
IFOLJC EQU $33FF
ICADR EQU $C000
X0 EQU $C000
XTSTIF EQU $C010
TAPEOUT EQU $C020
SKPR EQU $C030
TXICLR EQU $C050
TYSPT EQU $C051
MIXCLR EQU $C052
MIXSET EQU $C053
LOWSCR EQU $C054
HISCR EQU $C055
LOPES EQU $C056
HIRES EQU $C057
TAPEIN EQU $C060
PADMLO EQU $C064
PTRIG EQU $C070
BASIC EQU $F000
BASIC2 EQU $F003

F000: 4A PLOT LSR A T-COORD/2
F001: 08 PHP SAVE LSR IN CARRY
F002: 20 47 FB JSR GBASCALC CALC BASE AOP IN GBASEL H
F005: 28 PLP RESTORE LSR FROM CARRY
F006: A9 0F LDA #$0F MASK 00F IF EVEN
F008: 90 02 BCC RTMASK
F00A: 69 E0 ADC #$E0 MASK $F0 IF ODD
F00C: 85 2E RTMASK STA MASK
F00E: B1 26 PLOT1 LDA (GBASL),Y DATA
F010: 45 30 EOR COLOR XOR COLOR
F012: 25 2E AND MASK AND MASK
F014: 51 26 EOR (GBASL),Y XOR DATA
F016: 91 26 STA (GBASL),Y TO DATA
F018: 60 RTS
F019: 20 00 F8 JSR HLINE JSR PLOT PLOT SQUARE
F01C: C4 2C HLINE1 CPY H2 DONE?
F01E: 80 11 BCS RTS1 YES, RETURN
F020: C8 INY NO, INCR INDEX (X-COORD)
F021: 20 0E F8 JSR PLOT1 PLOT NEXT SQUARE
F024: 90 F6 BCC HLINE1 ALWAYS TAKEN
F026: 69 01 VLINE2 ADC #$01 NEXT Y-COORD
F028: 48 VLINE PHA SAVE ON STACK
F029: 20 00 F8 JSR PLOT PLOT SQUARE
F02C: PLA
F02D: C5 2D CMP V2 DONE?
F02F: 90 F5 BCC VLINE2 NO,LOOP.
F031: 60 RTS1 RTS
F032: A0 2F CLRSCR LDY #$2F MAX Y, FULL SCR CLR
F034: 00 02 PNE CLRSC2 ALWAYS TAKEN
F036: A0 27 CLRTOP LDY #$27 MAX Y, TOP SCR CLR
F038: 84 2D CLRSC2 STY V2 STORIF AS BOTTOM COORD
F03A: A0 27 LDY #$27 RIGHT MOST X-COORD (COLUMN)
F03C: A9 00 CLRSC3 LDA #$0 TO COP FOR VLINE CALLS
F03E: 85 30 STA COLOR CLEAR COLOR (BLACK)
F040: 20 28 F6 JSR VLINE JMP VLINE
F043: 88 DEY NEXT LFT MOST X-COORD
F044: 10 F6 AXL CLRSC3 LOOP UNTIL DONE.
F046: 60 RTS
F047: 45 GBASCALC PHA FOR INPUT NDMECH
F048: 44 LSR A
F049: 29 03 AND #$03
F04B: 09 04 OPA #$04 GENERATE GBASH=00001FC
F04D: 85 27 STA GOASH AND GBASH=00000000
F04F: 68 PLA
F050: 29 18 AND #$18
F052: 90 02 SCC GBASCALC
F054: 69 7F ADC #$F
F056: 85 26 GBASCALC STA GBASL
F8E7: 90 F2  BCC PRNRBL RECOVER MNEMONIC INDEX
F8E9: 68  PLA
F8EA: A8  TAY
F8EB: B9 C0 F9  LDA MNEML,Y
F8EE: 85 2C  STA LNMEM
F8F0: B9 00 FA  LDA MNEM,Y
F8F3: 85 2D  STA RMNEM
F8F5: A9 00 PRML1
F8F7: A0 05 LDY #$05
F8F9: 06 2D PRML2
F8FB: 26 2C  ROL LNMEM
F8FD: 2A  ROL A
F8FE: 88  DEY
F8FF: D0 F8  BNE PRML2
F901: 69 BF  ADC #$BF
F903: 20 ED FD  JSR COUT
F906: CA  DEX
F907: D0 EC  BNE PRML1
F909: 20 48 F9  JSR PRBLNK
F90C: A4 2F  LDY LENGTH
F90E: A2 06  LDX #$06
F910: ED 03 CPX #$03
F912: P0 1C  BEO PRML5
F914: 06 2E  ASL RNMEM
F916: 90 0B  BCC PRADR3
F918: BD 83 F9  LDA CHAR-1,X
F91B: 20 ED FD  JSR COUT
F91E: BD B9 F9  LDA CHAR-2,X
F921: F0 03  BEO PRADR3
F923: 20 ED FD  JSR COUT
F926: CA  DEX PRADR3
F927: D0 E7  RNE PRADR1
F929: 60  RTS
F92A: 88  PRADR4  DEY
F92B: 30 E7  BMI PRADR2
F92D: 20 CA FD  JSR PRRTYE
F930: A5 2E PRADR5  LDA FORMAT
F932: C9 E6  CMP #$E6
F934: B1 3A  LDA (PCL),Y
F936: 90 F2  BCC PRADR4
F938: 20 56 F9 RELADR
F93B: AA  JCJ PRADJ3
F93C: E8  INX
F93D: D0 01  RNE PRRTYX
F93F: C8  INY
F940: 98 PRRTYX TYA
F941: 20 DA FD PRRTAX  JSR PRRTYE
F944: 8A  PHRTX
F945: 4C DA FD  JMP PRRTYE
F948: A2 03 PRBLNK  LDY #$03
F94A: A9 A0 PRBL2
F94C: 20 ED FD PRBL3
F94F: CA  DEX
F950: D0 F8  BNE PRBL2
F952: 60  RTS
F953: 38 PCADJ  SEC
F954: A5 2F PCADJ2
F956: A4 3B PCADJ3  LDA LENGTH
F958: AA  TAX
F959: 10 01  BPL PCADJ4
F95B: 88  DEY
F95C: 65 3A PCADJ4
F95E: 90 01  BCC RTS2
F960: C8  INY
F961: 60 RTS2 RTS

* FMT1 BYTES: XXXXXXX0 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
F967: 80 04 90
F96A: 03 22
F96C: 54 33 0D
F96F: 80 04
F971: 90 04 20
F974: 54 33
F976: 0D 80 04
F979: 90 04
F97B: 20 54 3B
F97E: 0D 80
F980: 04 50 00
F983: 22 44
F985: 33 0D C8
F988: 44 00

DPF $04,$20,$54,$30,$0D
DPF $54,$33,$0D,$80,$04
DPF $90,$04,$20,$54,$33
DPF $0D,$80,$54,$90,$04
DPF $20,$54,$33,$0D,$80
DPF $04,$90,$04,$20,$54
DPF $33,$0D,$C8,$44,$00

79
F98A: 11 22 44  
F98B: 33 0D  
F98F: C8 44 A9  
F992: 01 22  
F994: 44 33 0D  
F997: 80 04  
F999: 90 01 22  
F99C: 44 33  
F99E: 0D 60 04  
F9A1: 90  
F9A2: 26 31 87  
F9A5: 9A  
F9A6: 00  
F9A7: 21  
F9A8: 81  
F9A9: 82  
F9AA: 00  
F9AB: 00  
F9AC: 59  
F9AD: 4D  
F9AE: 91  
F9AF: 92  
F9B0: 8E  
F9B1: 4A  
F9B2: 85  
F9B3: 9D  
F9B4: AC A9 AC  
A3 A8 A4  
F9B9: 9A 00  
F9BD: A4 A4  
F9CE: 1B A1 9D  
F9C9: 8A 1D 23  
F9CC: 9D 88 1D  
F9CF: A1 00 29  
F9D2: 19 AE 69  
F9D5: A8 19 23  
F9D8: 24 53 18  
F9DB: 23 24 53  
F9DE: 19 A1  
F9E0: 00 1A 5B  
F9E3: 58 A5 69  
F9E5: 24 24  
F9EB: AE AE A8  
F9EF: AD 29 00  
F9F3: 15 9C 6D  
F9F9: 9C A5 69  
F9FE: 29 53  
F9FB: 84 13 34  
F9FF: 11 A5 69  
F9FE: 23 A0  
FA00: D8 62 5A  
FA03: 48 26 62  
FA06: 94 88 54  
FA09: 44 C8 54  
FA0C: 68 44 E6  
FA0F: 94 00 B4  
FA12: 08 84 74  
FA15: B4 28 6E  
FA18: 74 F4 CC  
FA1B: 4A 72 F2  
FA1C: 4A 8A  
FA20: 00 AA A2  
FA23: A2 74 74  
FA26: 74 72  
FA28: 44 68 82  
FA2B: 32 B2 00  
FA2E: 22 00  
FA30: 1A 1A 26  
FA33: 26 72 72  
FA36: 88 C8  
FA38: C4 CA 26  
FA3B: 48 44 44  
FA3E: A2 C8  
DFB $11, $22, $44, $33, $0D  
DFB $C8, $44, $A9, $01, $22  
DFB $44, $33, $0D, $80, $04  
DFB $90, $01, $22, $44, $33  
DFB $0D, $80, $04, $90  
DFB $26, $31, $07, $9A 3XXY01 INSTR'S  
DFB $00, ERR  
DFB $21, IMM  
DFB $81, Z-PAGE  
DFB $82, ABS  
DFB $00, IMPLIED  
DFB $00, ACCU MULATOR  
DFB $59, (ZPAG,X)  
DFB $4D, (ZPAG), Y  
DFB $91, ZPAG, X  
DFB $92, ABS, X  
DFB $86, ABS, Y  
DFB $4A, (ABS)  
DFB $85, ZPAG, Y  
DFB $9D, RELATIVE  
DFB (X=INDEX)
DEF  SFF,SPF,SFF
JSR  INSTDSP  DISASSEMBLE ONE INST
AT (PCL,H)
STA  RTNL  ADJUST TO USER
PLA  STACK.SAVE
STA  RTNH
RTN.ADR.
LDA  XQINIT
LDA  XINIT-1,X INIT QEX AREA
STA  QXT,X
DEX
BNE  XQINIT
LDA  (PCL,X)  USER QCODE BYTE
BEO  XBRK  SPECIAL IF BREAK
LDA  XJMP
MCP  #520  HANDLE JSR,PTS,JMP,
MCP  #560  JMP ( ), RTI SPECIAL
BEQ  XRTS
MCP  #54C
SEC  XJMP
MCP  #56C
SEC  XJMPAT
MCP  #540
SEC  XRTI
AND  #51F
SBR  #514
MCP  #504  COPY USRF INST TO XEO AREA
BEQ  XQ2  WITH TRAILING NOPS
LDA  (PCL),Y  CHANGE REL BRANCH
STA  QXTNZ,Y  DISP TO 4 POR
DEX:  88  JMP TO BRANCH OR
BPL  XQ1  NBRANCH FROM XEO.
JSR  RESTORE  RESTORE USER REG CONTENTS.
JSR  USER OP FROM RAM
JSR  (RETURN TO NBRANCH)
PLA  68  **IRQ HANDLER
asl A
asl A
asl A
BMI  BREAK  TEST FOR BREAK
JMP  (IROLOC)  USER ROUTINE VECTOR IN RAM
JSR  SAV1  SAVE REG'S ON BREAK
INCL  A  INCLUDING PC
STA  PCL
JSR  REGDSP1  AND REG'S
XRTI  CLR
PLA  SIMULATE RTI BY expect
STA  STATUS  STATUS FROM STACK, THEN RTS
PLA  RTS SIMULATION
STA  PCL  EXTRACT PC FROM STACK
AND  UPDATE PC BY 1 (LEN=0)
STA  PCH  PRINT USER PC.
JSR  RGDSP1  AND REG'S
JMP  MON  GO TO MONITOR
XRTI  CLC
STA  PCH
STA  STATUS  STATUS FROM STACK, THEN RTS
STA  PCL  EXTRACT PC FROM STACK
PLA  AND UPDATE PC BY 1 (LEN=0)
PCADJ3
PCADJ2
JSR  RGDSP1  UPDATE PC AND PUSH
TAX  ONTO STACK FOR
JSR  SICULATE
TAX
PHA
PHA
PHA
PLA  #502
CLC  (JMP) SIMULATE.
JSR  (JMP) SIMULATE.
JSR  RGDSP1  UPDATE PC AND PUSH
TAX  ONTO STACK FOR
JSR  SICULATE
TAX
PHA
PHA
PHA
PLA  #502
CLC  (JMP) SIMULATE.
JSR  (JMP) SIMULATE.
JSR  RGDSP1  UPDATE PC AND PUSH
TAX  ONTO STACK FOR
JSR  SICULATE
TAX
PHA
PHA
PHA
PLA  #502
CLC  (JMP) SIMULATE.
JSR  (JMP) SIMULATE.
JSR  RGDSP1  UPDATE PC AND PUSH
TAX  ONTO STACK FOR
JSR  SICULATE
TAX
PHA
PHA
PHA
PLA  #502
CLC  (JMP) SIMULATE.
JSR  (JMP) SIMULATE.
JSR  RGDSP1  UPDATE PC AND PUSH
TAX  ONTO STACK FOR
JSR  SICULATE
TAX
PHA
PHA
PHA
PLA  #502
CLC  (JMP) SIMULATE.
JSR  (JMP) SIMULATE.
JSR  RGDSP1  UPDATE PC AND PUSH
TAX  ONTO STACK FOR
JSR  SICULATE
TAX
PHA
PHA
PHA
PLA  #502
CLC  (JMP) SIMULATE.
JSR  (JMP) SIMULATE.
JSR  RGDSP1  UPDATE PC AND PUSH
TAX  ONTO STACK FOR
JSR  SICULATE
TAX
PHA
PHA
PHA
PLA  #502
CLC  (JMP) SIMULATE.
JSR  (JMP) SIMULATE.
JSR  RGDSP1  UPDATE PC AND PUSH
TAX  ONTO STACK FOR
JSR  SICULATE
TAX
PHA
PHA
PHA
PLA  #502
CLC  (JMP) SIMULATE.
JSR  (JMP) SIMULATE.
JSR  RGDSP1  UPDATE PC AND PUSH
TAX  ONTO STACK FOR
JSR  SICULATE
TAX
PHA
PHA
PHA
PLA  #502
CLC  (JMP) SIMULATE.
JSR  (JMP) SIMULATE.
JSR  RGDSP1  UPDATE PC AND PUSH
TAX  ONTO STACK FOR
JSR  SICULATE
TAX
PHA
PHA
PHA
PLA  #502
CLC  (JMP) SIMULATE.
JSR  (JMP) SIMULATE.
JSR  RGDSP1  UPDATE PC AND PUSH
TAX  ONTO STACK FOR
JSR  SICULATE
TAX
PHA
PHA
PHA
PLA  #502
CLC  (JMP) SIMULATE.
JSR  (JMP) SIMULATE.
JSR  RGDSP1  UPDATE PC AND PUSH
TAX  ONTO STACK FOR
JSR  SICULATE
TAX
PHA
PHA
PHA
PLA  #502
CLC  (JMP) SIMULATE.
JSR  (JMP) SIMULATE.
FAE0: A9 00  LDA #ACC/256
FAE0: B5 41  STA A9H
FAE2: A2 FB  LDX #$FB
FAE4: A9 A0  RDP,1  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 #$9D
FAF1: 20 ED FD  JSR COUT
FAF4: B5 4A  LDA ACC+5,X
FAF6: 20 DA FD  JSR PBYTE
FAF9: E8  INX
FAFA: 30 E8  BMI RDP,1
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 PRRNC  JSR SAVE NORMAL RETURN AFTER
FB0C: 38  SEC XEO USER OF
FB0F: B0 9E  PCS PCINC3 GO UPDATE PC
FB11: EA  INITBL NOP
FB12: EA  NOP DUMMY FILL FOR
FB13: 4C 00 PB  JMP PRRNC XEO AREA
FB16: 4C FD FA  JMP BRANCH
FB19: C1  RTBL DFS $C1
FB1A: D8  DFS $D8
FB1B: D9  DFS $D9
FB1C: D0  DFS $D0
FB1D: D3  DFS $D3
FB1E: AD 70 CO PREAD LDA FTP1G TRIGGER PADDLES
FB21: A0 00  LDY #$00 INIT COUNT
FB22: EA  NOP COMPENSATE FOR 1ST COUNT
FB24: EA  NOP
FB25: BD 64 CO PREAD2 LDA PADDLO,X COUNTER 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 CO  LDA LORES LINES
FB36: AD 54 CO  LDA LOWSCR INIT VIDEO MODE
FB39: AD 51 CO SETTXT LDA TXYSTRT SET FOR TEXT MODE
FB3C: A9 00  LDA #$00 FULL SCREEN WINDOW
FB3E: F0 0B  REO SETNN
FB40: AD 50 CO SETGR LDA TXYCLR SET FOR GRAPHICS MODE
FB43: AD 53 CO LDA MIXSET LOWER 4 LINES AS
FB46: 20 36 F8  JSR CLPTOP TEXT WINDOW
FB49: A9 14  LDA #$14
FB4B: 85 22  SETND STA WNDTOP SET FOR 40 COL WINDOW
FB4D: A9 00  LDA #$00 TOP IN A-REG,
FB4F: 85 20  STA WNDLFT BTM AT LINE 24
FB51: A9 28  LDA #$28
FB53: 85 21  STA WNDSTM
FB55: A9 18  LDA #$16
FB57: 85 23  STA WNDTM VTAB TO ROW 23
FB59: A9 17  LDA #$17
FB5B: 85 25  TABV STA CV VTAB TO ROW IN A-REG
FB5D: 4C 22 FC  JMP VTAB
FB60: 20 A4 FE MILPM JSP #$01 ASS VAL OF AC AUX
FB63: A0 10  MUL LDY #$10 INDEX FOR 16 BITS
FB65: A5 50  MUL2 LDA ACI ACK * AUX + XTND
FB67: 4A  LSR A TO AC, XTND
FB68: 90 0C  BCC MUL4 IF NO CARRY,
FB6A: 18  CLC NO PARTIAL PPUD.
FB6B: A2 FE  LDX #$FE
FB6D: B5 54  MUL3 LDA XTNDL+2,X ADD MPLCND (AUX)
FB6F: 75 56  AOC AUXL+2,X TO PARTIAL PROD
FB71: 95 54  STA XTNDL+2,X (XTND).
FB73: E4  INX
FB74: D0 F7  ABF "MUL3
FB76: A2 03  MUL4 LDX #$03
FB7E: 76  DFB #$76
FB79: 50  CPE #$50
FB7A: CA  DEY
FB7B: 10 FE  BPL MUL5
FB7D: 88  DEY
FB7E: D0 E5  BNE MUL2
FB80: 60  PHS
PC14: 80 0D  ACS RTS4 IF TOP LINE THEN RETURN
PC20: C6 25  DEC CV  DFCR CURSOR V-INDEX
PC22: A5 25  LDA CV  GET CURSOR V-INDEX
PC24: 20 C1 FB  VTABLE JSR PASCAL  GENERATE BASE ADDR
PC27: 65 20  LDA WHDLFT ADD WINDOW LEFT INDEX
PC29: 85 28  STA BASL TO BASL
PC30: 60 RTS4  PTS
PC2C: 49 C0  ESC1 EOR #$C0  ESC?
PC2E: F0 28  FPO HOME  IF NO DO HOME AND CLEAR
PC30: 69 FD  ADC #$FD ESC-A OP B CHECK
PC34: A5 00  LDA #SPS & ADVANCE A, ADVANCE
PC34: F0 DA  BCS BS  R, BACKSPACE
PC36: 69 FD  ADC #$FD ESC-C OP D CHECK
PC38: 90 2C  SBC LF  C, DOWN
PC3A: F0 DE  BCS UP  D, GO UP
PC3C: 69 FD  ADC #$FD ESC-E OP F CHECK
PC3E: 90 5C  SBC CREL0L  E, CLEAR TO END OF LINE
PC40: D0 E9  BNE RTS4  NOT F, RETURN
PC42: A4 24  CLREOP  LDY CH  CURSOR H TO Y INDEX
PC44: A5 25  LDA CV  CURSOR V TO A-REGISTER
PC46: 48 0A  CLEOP1  PLA  SAVE CURRENT LINE ON STK
PC47: 20 24 FC  JSR VTABLE CALC BASE ADDRESS
PC4A: 20 9E FC  JSR CLEOL2 CLEAR TO FOL, SET CARRY
PC4D: A0 00  LDA #$00 CLEAR FROM H INDEX=0 FOR FST
PC4F: 0A 48  LDA INCURRENT CURRENT LINE
PC50: 69 00  ADC #$00 (CARRY IS SET)
PC52: C5 23  CMP $D50F DONE TO BOTTOM OF INDEX?
PC54: 90 F0  SCS CLEOP1 NC, KEEP CLEAPING LINES
PC56: B0 CA  PCS VTABLE YES, PASS TO CURRENT LINE
PC58: A5 22  HME LDA #WINDUP INIT CURSOR V
PC5A: 85 25  STA CV AND H-INDICES
PC5C: A0 00  LDA #$00
PC5E: 84 24  NTY CH  THEN CLEAR TO END OF PAGE
PC60: F0 E4  SEO CLEOP1
PC62: A9 00  CR LDA #$00 CURSOR TO LEFT OF INDEX
PC64: 85 24  STA CH (SET CURSOR H=0)
PC66: B6 25  LFS INC CV INCR CURSOR V(DOWN 1 LINE)
PC68: A5 25  LDA CV
PC6A: C5 23  CMP ($D50F) OFF SCREEN?
PC6C: 90 B6  SCS VTAB? NO, SET BASE ADDR
PC6E: C6 25  SCS CV DEC CURSOR V(BACK TO BOTTOM LINE)
PC70: A5 22  SCROLL  LDA WINDUP START AT TOP OF SCRL WIND
PC72: 48 0A  PHA
PC73: 20 24 FC  JSR VTABZ GENERATE BASE ADDRESS
PC76: A5 28  SCRL1 LDA BASL COPY BASL,H
PC78: 85 2A  STA BAS2L TO BAS2L,H
PC7A: A5 29  LDA BAS4
PC7C: 85 28  STA BAS2H
PC7E: A4 21  LDY #WIDTH INIT Y TO RIGHTMOST INDEX
PC80: 88  DEY OF SCROLLING WINDOW
PC81: 68 0A  PLA
PC82: 69 01  ADC #$01 INC PR LINE NUMBER
PC84: C5 23  CMP $D50F DONE?
PC86: B0 0D  SCS SCRL3 YES, FINISH
PC88: 48 0A  PHA
PC89: 20 24 FC  JSR VTABZ FORM BASL,H (BASE ADDR)
PC8C: B1 28  SCRL2 LDA (BASL),Y MOVE A CHR UP ON LINE
PC8E: 91 2A  STA (BAS2L),Y
PC90: 88  DEY NEXT CHAR OF LINE
PC91: 10 F9  BPL SCRL2
PC93: 30 E1  RAI SCRL1 NEXT LINE
PC95: A0 00  SCRL3 LDA $5000 CLEAR BOTTOM LINE
PC97: 20 9E FC  JSR CLEOL2 GET BASE ADDR FOR BOTTOM LINE
PC99: B0 86  ACS VTAB CARRY IS SET
PC9C: A4 24  CLREOL  LDY CH CURSOR H INDEX
PC9E: A9 A0  CLEOL1 LDA #$A0
PCAE: 91 28  CLEOL2 STA (BASL),Y STORE BLANKS FROM 'HERE'
PCAF: C8 06  INX TO END OF LINES (WIDTH)
PCAC: C4 21  CPY #WIDTH
PCAD: 95 09  SCS CLEOL2
PCAE: 60 00  RTS
PCAF: 38 06  WAIT SEC
PCAG: 48 06  WAIT2 PHA
PCAA: E9 01  WAIT3 SCS #$01
PCAC: D0 FC  PNS WAIT3 1.020 USEC
PCAE: 60 00  PLA
PCAF: E9 01  SRC #$01
PCB0: D0 F6  BNE WAIT2
PCB3: 60 00  RTN
PCB4: E6 42  NXTA4 INC A4L INC# 2-BYTE A4
PCB6: D0 02  PNE NXTA1 AND A1
PCB8: F6 43  INC A4H
PCBA: A5 3C  NXTA1 LDA AIL INC# 2-BYTE A1.
PCBC: C5 3E  CMP AIL
PCBD: A5 3D  LDA AIL AND COMPARE TO A2
SEC A2h (CARRY SET IF >=)
INC \1L
BMI \734
INC A1h
RTS: 60
RTE: 40
LDY \$4A P WRITE A*256 'LONG 1'
JSP ZEPFDY HALF CYCLES
JMP HEADP (650 USEC EACH)
ADC \$FF
BCC HEADR THEN A 'SHORT 0'
PCD: 20 D8 FC WRITBIT JSR ZEPFDY WRITE TWO HALF CYCLES
PCD9: CB INY OR 250 USEC ('0')
PCD: 1C INY OR 500 USEC ('0')
PCD8: 08 ZEPFDY DEY
PCDC: 00 FD BNE ZEPFDY
PCDE: 00 05 BCC WRTAPE Y IS COUNT FOR
PCD0: A0 32 LDY \$52 TIMING LOOP
PCD1: 88 ONEDLY DEY
PCD2: 90 05 PNE ONEDLY
PCD5: AC 20 CO WRTAPE LDY TAPEOUT
PCD8: A0 2C LDY \$2C
PCDA: CA DEX
PCDB: 40 RTS
PCDC: A2 08 RDBYTE LDY \$08 8 BITS TO READ
PCDD: 48 RDBYTE2 PHA READ TWO TRANSITIONS
PCDE: 20 FA FC JSP ROBIT (FIND EDGE)
PCDF: 68 PLA
PCDG: 2A PHL A NEXT JOT
PCDH: A0 3A LDY \$3A COUNT FOR SAMPLES
PCDI: D0 F5 BNE ROBYTE2
PCDJ: 60 RTS
PCDK: 2A ROBYTE2 JSR ROBIT'
PCDL: 88 ROBIT DEY DECR Y UNTIL
PCDM: AD 60 CO LDA TAPFIN TAPE TRANSITION
PCDN: 45 2F EOR LASTIN
PCDO: 10 F8 PPL ROBIT
PCDP: 45 2F EOR LASTIN
PCDQ: 85 2F STA LASTIN
PCDR: C0 80 CPY \$80 SET CARRY ON Y-REG.
PCDS: 60 PTS
PDDC: A4 2A LDA CH
PDDD: H1 2B LDA (PASL),Y SET SCREEN TO FLASH
PDE: 48 PHA
PDDF: 29 3F ANP \$3F
PDDG: 09 40 ORA \$40
PDDH: 91 2B STA (PASL),Y
PDDI: 68 PLA
PDDJ: 6C 3B 00 JMP (K:0L) GO TO USER KEY-IN
PDDK: E6 4E KEYIN INC \#ML
PDDL: D0 B6 BNE KEYIN2 INCR AND NUMBER
PDDF: E6 4F INC \#NH
PDDG: 2A 00 CO KEYIN2 BIT ROB KEY DOWN?
PDDH: 0F 10 BPL KEYIN LOOP
PDDI: 91 2B STA (PASL),Y REPLACE FLASHING SCREEN
PDDJ: AD 00 CO LDA \$20 SET KEYCODE
PDDK: 2C 10 CO BIT RDSTP CLR KEY STPDB
PDDL: 60 PTS
PDDM: 20 0C FD ESC JSP PDKFY GET DPKFY
PDDN: 20 2C FC ESC JMP ROCH1
PDDO: 20 0C FD PDKCHAR JSP ROCH1 HANDLE ESC FUNC.
PDDP: 20 0C FD PDKKEY JSP ROCH1 READ KEY
PDDQ: C9 93 CMP \$93 
CPL \$93 
PCDD: F0 03 
PLY \#ESC 
PCDD: 48 RTS
PCDE: A5 32 NOTCR LDA INVFLG
PCDF: 48 PHA
PCDG: 85 32 STA INVFLG ECHO USER LINE
PCDH: BD 00 02 LDA INX, X NON INVERSE
PCDI: 20 ED FD JSR COUT
PCDJ: 6A 80 PLA
PCDK: 85 32 STA INVFLG
PCDL: BD 00 02 LDA INX
PCDM: C9 45 CMP \$56 CHECK FOR EDIT KEYS
PCDN: 80 1D BEO BCKSPC BS, CTRL-X
PCDO: C9 98 CMP \$98
PCDP: 00 0A BEO CANCEL
PCDQ: E0 F8 CPX \$FF MARGIN?
PCDR: 90 03 BCC NOTCR1
PCDS: 20 3A FF JSR \#ELL YES, SOUND PELL
PCDT: E8 NOTCR1 INX ADVANCE INPUT INDEX
PCDU: D0 13 BNE NXTCHAR
PCDV: A5 DC CANCEL LDA \$DC BACKSLASH AFTER CANCELLED LINE
PCDW: 20 ED FD JSR COUT
FD67: 20 8E FD GETLNZ JSR CROUT OUTPUT CR
FD6A: A5 35 GETLN LDA PROMPT
FD6C: 20 ED FD JSR COUT OUTPUT PROMPT CHAR
FD6F: A2 01 LDX #S01 INIT INPUT INDEX
FD71: 8A ECKSPC TXA WILL #ACKSPACE TO 0
FD72: FD F3 SEQ GETLN?
FD74: CA DEX
FD75: 20 35 FD NXTCLAF JSR PCHAP
FD78: C9 95 CMP #PICK USE SCREEN CHAR
FD7A: D0 02 BNE COUT FOR CTRL-U
FD7C: B1 28 LDA (SACL),Y
FD7E: C9 E0 CAPTST CMP #$F0
FD80: 90 02 BCC ADDINP CONVERT TO CAPS
FD82: 29 DF AND #$DF
FD84: 90 00 02 ADDINP STA IX,X ADD TO INPUT BUF
FD87: C9 BD CMP #$BD
FD89: D0 B2 BNE NOTCR
FD8B: 20 9C FC JSR CLPLEOL CLR TO FOL IF CR
FD8E: A9 8D CROUT LDA #$8D
FD90: D0 5B BNE COUT
FD92: A4 3D LOY ALH PRINT CP,AL IN HEX
FD94: A6 3C LDY ALH
FD96: 20 6E FD PHYX2 JSR CROUT
FD99: 20 4F F9 JSR PHYTX
FD9C: A0 00 LDY #$00
FD9E: A9 AD LDA #$AD PRINT ','
FDAD: 4C ED FD JNP COUT
FDA3: A5 3C XAMB LDA ALL
FDA5: 09 07 CRA #$07 SFT TO FINISH AT
FDA7: 85 3E STA A2L MOD P=7
FDA9: A5 3D LDA ALH
FDAH: 85 3F STA A2H
FDAD: A5 3C MODCHK LDA A2L
FDAF: 29 07 AND #$07
FD81: D0 03 BNE DATOUT
FD83: 20 92 FD XAN JSR PA1
FD86: A9 A0 DATAOUT LDA #$A0
FD88: 20 FD FD JSR COUT OUTPUT BLANK
FD8B: B1 3C LDA (ALL),Y
FD9D: 20 DA FD JSR PHYTX OUTPUT BYTE IN HEX
FDCC: 20 BA FC JSR NXTA1
FD3: 90 E8 BCC MODCHK CHECK IF TIME TO,
FD5: 60 RTS4C RTS PPMT ADOP
FD7: 4A XAMPX LSR A DETERMINE IF MOD
FD77: 90 EA BCC XAM "MODE IS XAM"
FD99: 4A LSR A ADD, OR SUB
FDCA: 4A LSR A
FDCC: A5 3F LDA A2L
FDDC: 90 02 BCC ADD
FDCF: 49 FF EQR #$FF SUB: FORM 2'S COMPLEMENT
FDDE: 65 3C ADD ADC AIL
FDD4: 48 PHA
FDD7: A9 B0 LDA #$9D
FDD9: 20 ED FD JSR COUT PRINT '>', THEN RESULT
FDDA: 68 PLA
FDDB: 48 PRBYTE PHA PRINT BYTE AS 2 HEX
FDDB: 4A LSR A DIGITS, DESTROYS A-REG
FDDC: 4A LSR A
FDDE: 4A LSR A
FDFF: 20 E5 FD JSR PHEX2
FD2: 68 PLA
FDE3: 29 0F PHEX AND #$DF PRINT HEX DIG IN A-REG
FDE5: 09 B0 PHEX2 ORA #$9D LSR'S
FDE7: C9 BA CMP #$BA
FDE9: 90 01 BCC COUT
FDEB: 69 06 ADC #$06
FDED: 6C 36 00 COUT JMP (CSVX) VECTOR TO USER OUTPUT ROUTINE
FDF0: C9 A0 COU1 CMP #$A0
FDF2: 90 02 BCC Cout2 DON'T OUTPUT CTRL'S INVERSE
FDF4: 25 32 AND INVLG MASK WITH INVERSE FLAG
FDF6: 84 35 Cout2 STY YSAV1 SAV Y-REG
FDF8: 4B PHA SAV A-REG
FDF9: 20 FD FB JSR VIDOUT OUTPUT A-REG AS ASCII
FDFF: 68 PLA RESTORE A-REG
FDFF: A4 35 LDY YSAV1 AND Y-REG
FDFF: 60 RTS THEN RETURN
FE00: C6 34 BL1 DEC YSAV
FE02: F0 9F BEO XAMB
FE04: CA BLANK DEX BLANK TO MON
FE05: D0 16 BNE SETMD2 AFTER BLANK
FE07: C9 BA CMP #$BA DATA STORE MODE?
FE09: D0 B8 BNE XAMPm NO, XAM, ADD OR SUB
FE0B: 85 31 STA MODE KEEP IN STORE MODE
FE0D: A5 3E LDA A2L
FEB6: 20 75 FE GO JSR A1PC AND TO PC IF SPEC'D
FEB9: 20 3F FF JSR RESTORE RESTORE META PEGS
FEBC: 6C 3A 0D JSR P (PCL) GO TO USER PRT SURR
FEBF: 4C 07 FA REG JSR REGSP TO REG DISPLAY
FEC2: C6 34 TPACF DEC YSAV
FEC4: 20 75 FE STEP JSR A1PC AND TO PC IF SPEC'D
FEC7: 4C 43 FA JSR STEP TAKE ONE STEP
FECA: 4C F6 03 USF JSR USRADDR TO USR SURR AT USRADDR
FECD: A9 40 WRITE LDA $40
FECE: 20 C9 FC JSR HEADER WRITE 10-6FC HEADER
FED2: A0 27 LDY #$27
FED4: A2 00 WR1 LDX #$00
FED6: 41 3C EOP (ALL,X)
FED9: 48 PLA
FEDE: A1 3C LDA (ALL,X)
FEF2: 20 8A FC JSR WRPTC
FEF5: 6C LDA (ALL,X)
FEF8: B0 4D FC JSR WRPTC
FEF3: D0 FA BNE WRPTC
FEF5: 60 RTS
FEF6: 20 00 FC CRMON JSR BL1 HANDLE CR AS PLANK
FEF9: 68 PLA AND PTT TO MON
FEFB: D0 6C BNE WMON
FEFC: 20 FA FC READ JSR R02B1T FIND TAPEIN EDGE
FEFD: A9 16 LDA #$16
FEF0: 20 C9 FC JSR HEADER DWAY 3.5 SECONDS
FEF5: 85 2E STA CHKSUM INIT CHKSUM=$FF
FEF7: 20 FA FC JSR R02B1T FIND TAPEIN EDGE
FEF9: A0 24 R02 LDX #$24 LOOK FOR SYNC PIT
FEOF: 20 FD FC JSR R02B1T (SHUNT 0)
FEOF: B0 F9 6C DD LOOPUNTILFOUND
FEOF: 14 A0 18 RD LDA #$18 INDXX PEG 0/1 TEST
FEOF: 18 40 FC R03 JSR ROPYTC READ A BYTE
FEOF: 81 3C STA (ALL,X) STORE AT (A1)
FEB1: 45 2E EOR CHKSUM
FEBD: 85 2E STA CHKSUM UPDATE RUNNING CHKSUM
FEBF: 20 3A FC JSR XXTA1 INCR A1, COMPARE TO A2
FEC2: A0 35 LDX #$35 COMPENSATE 0/1 INDX
FEC4: 90 F0 R03 LDA R03 LOOP UNTIL DONE
FEF6: 20 EC FC JSR R02YTC READ CHKSUM BYTE
FEF9: C5 2E CMP CHKSUM
FEF1: 00 B0 R00 LDX #$00 GOOD, SOUND BELL AND RETURN
FEF2: A9 C5 PREPR LDA #$05
FEF2: 20 ED FD JSR COUT PRINT "ERR", THEN BELL
FEE3: A9 D2 LDA #$02
FEE4: 20 ED FD JSR COUT
FEE7: 20 ED FD JSR COUT
FEEA: A9 87 BELL LDA #$87 OUTPUT BELL AND RETURN
FEEC: 4C ED FD JSR COUT
FFE3: A5 4B RESTORE LDA STATUS RESTORE 6502 PEG CONTENTS
FFE4: 48 PLA USED BY DEBUG SOFTWARE
FFE4: A5 45 LDA ACC
FFE6: A6 46 REST2 LDX XPEG
FFE6: A4 47 LDY YPEG
FFE8: 28 PLP
FF49: 60 RTS
FF4A: 85 45 SAVE STA ACC SAVE 6502 REG CONTENTS
FF4C: B6 46 SAVE STX XPEG
FF4E: 84 47 STY YPEG
FF50: 08 PHP
FF51: 68 PLA
FF52: B5 48 STA STATUS
FF54: DA TSX
FF55: B6 49 ST2 SPNT
FF57: DB CLR
FF58: 60 PPS
FF59: 20 84 FE RSET JSR SLEDGE SPY SCREEN MODE
FF5C: 20 2F FB JSR INIT AND INIT KBD/SCREEN
FF5F: 20 93 FE JSR SETVID AS I/O DEV'S
FF62: 20 89 FE JSR SETKMD
FF65: DA MON CLD MUST SET HEX MODE!
FF66: 20 3A FF JSR BELL
FF69: A9 AA MONZ LDA #$AA "*" PROMPT FOR MON
FF6B: 85 33 STA PROMPT
FF6D: 20 67 FD JSR CEILNZ READ A LINE
FF70: 20 C7 FF JSR $920B CLEAR MON MODE, SCAN IDX
FF73: 20 A7 FF JSR $920F GETITEM, GET ITEM, NON-HEX
FF76: 84 34 STY $FFA4 CHAR IN A-REG
FF78: A0 17 LDY #$17 X-DEC=0 IF NO HEX INPUT
FF7A: 88 CHRSRC DEY
FF7B: 30 8E RTI MON NOT FOUND, GO TO MON
FF7D: D9 CC FF CALL BARTLIB, FIND CHRNO CHAR IN TEL
FF80: D0 8E CALL CHRSRC
FF82: 20 BE FF JSR $9209 FOUND, CALL CORRESPONDING
FF85: A4 34 LDY $FFA4 SUFFIX
FF87: 4C 73 FF JMP $9214
FF8A: A2 83 DIC LDX $5B3
FF8C: 0A ASL A
FF8D: 0A ASL A GOT HEX DIC, SHIFT INTO A2
FF8E: 0A ASL A
FF90: 0A ASL A
FF91: 26 3E ROL A2L
FF93: 26 3F POL A2H
FF95: CA DEX LEAVE X$=PE IF DIG
FF96: 10 F8 SBL $9212
FF98: A5 31 LDA MODE
FF9A: D0 0B PNE $9216 IF MODE IS ZERO
FF9C: B5 3F LDA A2H,X THEN COPY A2 TO
FF9E: 95 3D STA A1H,X 1L AND A3
FFA0: 95 41 STA A3H,X
FFA2: E8 RXTXS INX
FFA3: F0 F3 BCF $9215
FFA5: D0 06 CMP $9210
FFA7: 02 00 GETYN LDX $5C0 CLEAR A2
FFA8: 86 3E SXA A2L
FFA9: 86 3F STH A2H
FFAB: B9 00 02 VSXCHP LDA INY GET CHAR
FFBD: C8 IXY
FFBE: 49 B2 LDX $58U
FFB3: C9 0A CMP $50A
FFB5: 90 D3 BCC DIC IF HEX DIG, THEN
FFB7: 69 86 ADC $5A8
FFB9: C9 FA CMP $5FA
FFBA: BC 00 0C DCS IOG
FFBC: 60 BIS
FFBE: A9 FE TOSUB LDA $C0/256 XINF-HIGH-ORDER
FFCC: 4B 48 PHA SUPP ADR ON SFK
FFCD: B9 E3 FF LDA $901F,X FCSH IN ORDER
FFCF: 4B 48 PHA SUPP ADR ON SFK
FFD1: A9 31 LDA MODE
FFD4: A8 00 2AHDE LDX *50C CLP MODE, OLD REG
FFD9: d4 31 STY MODE TO A-REG
FFDB: 60 05 RTS GO TO SUPR VIA PTS
FFDC: C4 CHRTPL DEX $28C "CTRL-C"
FFDD: A2 DPY $282 "CTRL-Y"
FFDF: EE DEX $289 "CTRL-E"
FFE0: 0D DEX $286 "CTRL-D"
FFE2: 68 DEX $285 "CTRL-A"
FFE1: 66 DEX $284 "CTRL-B"
FFE8: A6 DEX $280 "CTRL-`"
FFED: 44 DEX $27F "CTRL+"
FFF0: 06 DEX $27E "CTRL-`
FFF4: 85 DEX $27D "CTRL-`
FFF7: A7 DEX $27B "CTRL-`
FFF9: C6 DEX $27A "CTRL-`
FFF2: 99 DEX $279 "CTRL-`
FFF3: B2 SUPTPL DEX #BASECONT-1
FFFE: C8 DEX #USER-1
FFF5: 8E DEX #RESET-1
FFF6: C1 DEX #TRACE-1
FFFE: 35 DEX #FVFY-1
0000 1F DEX #LINK-1
0089
FFF0: 83   DFB #SETNORM-1
FFF1: 7F   DFB #SETINV-1
FFF2: 5D   DFB #LIST-1
FFF3: CC   DFB #WRITE-1
FFF4: 59   DFB #GO-1
FFF5: FC   DFB #READ-1
FFF6: 17   DFB #SETMODE-1
FFF7: 17   DFB #SPTMODE-1
FFF8: F5   DFB #CPMON-1
FFF9: 03   DFB #BLANK-1
FFFA: F0   DFB #NMI  NMI VECTOR
FFFF: 03   DFB #NMI/256
FFFC: 59   DFB #RESET  RESET VECTOR
FFFD: FF   DFB #RESET/256
FFFE: 86   DFB #IRC  IRO VECTOR
FFFF: FA   DFB #IRC/256

XTNC      EQU $3C
**APPLE-II**
**MINI-ASSEMBLER**

**COPYRIGHT 1977 PY**
**APPLE COMPUTER INC.**

**ALL RIGHTS RESERVED**

**S. NOZIAK**

**A. GAUM**

***************************************************************************
**TITLE "APPLE-II MINI-ASSEMBLER"**

**FGBAAT** EQU $2F
**FGBAT** EQU $2F
**MBI** EQU $31
**PRIM** EQU $33
**YSAV** EQU $34
**L** EQU $35
**PCL** EQU $37
**PCH** EQU $39
**AII** EQU $37
**AI** EQU $36
**AQ** EQU $35
**A1** EQU $34
**A2** EQU $33
**A3** EQU $32
**A** EQU $31
**A1** EQU $30
**A2** EQU $29
**A3** EQU $28
**A** EQU $27
**A1** EQU $26
**A2** EQU $25
**A3** EQU $24
**A** EQU $23
**A1** EQU $22
**A2** EQU $21
**A3** EQU $20
**A** EQU $19
**A1** EQU $18
**A2** EQU $17
**A3** EQU $16
**A** EQU $15
**A1** EQU $14
**A2** EQU $13
**A3** EQU $12
**A** EQU $11
**A1** EQU $10
**A2** EQU $9
**A3** EQU $8
**A** EQU $7
**A1** EQU $6
**A2** EQU $5
**A3** EQU $4
**A** EQU $3
**A1** EQU $2
**A2** EQU $1
**A3** EQU $0
**A** EQU $0

**P500: 49 1 REL SBC $501** IS P/F COMPATIBLE
**P502: 2A 4 A LSR A** WITH RELATIVE CODE?
**P503: D0 14 DEX** NO.
**P505: A4 3F DEX** DIRECTIONS
**P507: A6 3F LDF AZ1** DIRECTION OF DEC:
**P509: D0 01 BNE REL2**
**P50B: 88 DEY**
**P50C: CA REL2 DEX**
**P50D: 8A TXA**
**P50E: 18 CLC**
**P50F: E5 3A SBC PCL**
**P511: 85 3E STA AZ1** FORM ADDR-PC-2
**P513: 10 01 BPL REL3**
**P515: C8 INY**
**P516: 98 REL3 TYA**

91
F517: E5 3B  SBC  PCH
F519: D0 6B  ERR3  BNE  ERR  ERROR IF >1-BYTE BRANCH
F51B: A6 2F  FINDOP  LDY  LENGTH
F51D: B9 3D 00  FNOP2  LDA  A1H,Y  MOVE INST TO (PC)
F520: 91 3A  STA  (PCL),Y
F522: 88  DEY
F523: 10 F8  GPL  FNOP2
F525: 20 1A PC  JSR  CURSUP
F528: 20 1A PC  JSR  CURSUP  RESTORE CURSOR
F52B: 20 D0 F8  JSR  INSTDSP  TYPE FORMATTED LINE
F52F: 20 53 F9  JSR  PCDIJ  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  PAKEMON2  DEY  CHECK NEXT DELIM
F545: 30 4B  BMI  PRESET2  ERR IF UNRECOGNIZED DELIM
F547: D9 CC FF  CMP  CHRTBL,Y  COMPARE WITH DELIM TABLE
F54A: D0 F8  BNE  PAKEMON2  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  #$00
F554: C6 34  DEC  YSAV
F556: 20 00 FE  JSR  BL1  HANDLE CR OUTSIDE MONITOR
F559: 4C 95 F5  JMP  NXTLINE
F55C: A5 3B  TRYNEXT  LDA  AIH,D  TRY TRIAL OPCODE
F55E: 20 BE F8  JSR  INSDS2  GET PTR+LENGTH FOR OPCODE
F561: AA  TAX
F562: BD 00 FA  LDA  MNEM,R,X  GET LOWER MNEMONIC BYTE
F565: C5 42  CMP  A4L  MATCH?
F567: D0 13  BNE  NEXTOP  NO, TRY NEXT OPCODE
F569: BD C6 F9  LDA  MNEM,L,X  GET UPPER MNEMONIC BYTE
F56C: C5 43  CMP  A4H  MATCH?
F56E: D0 DC  BNE  NEXTOP  NO, TRY NEXT OPCODE.
F570: A5 44  LDA  PFT  NEXT
F572: A4 2E  LDY  FORMAT  GET TRIAL FORMAT
F574: C0 9D  CPY  #$9D  TRIAL FORMAT RELATIVE?
F576: F0 88  BEQ  REL  YES.
F578: C5 2E  NREL  CMP  FORMAT  SAME FORMAT?
F57A: F0 9F  BNE  FINDOP  YES.
F57C: C6 3D  NEXTOP  DEC  AIH  NO, TRY NEXT OPCODE
F57E: D0 DC  BNE  TRYNEXT
F580: E6 44  INC  PFT  NO MORE, TRY WITH LEN+2
F582: C6 35  DEC  L  WAS L=2 ALREADY?
F584: F0 06  BEQ  TRYNEXT  NO.
F586: AA 34  ERR  LDX  YSAV  YES, UNRECOGNIZED INST.
F588: 98 82  ERR2  TYA
F589: AA  TAX
F58A: 20 4A F9  JSR  PRBL2  PRINT " UNDER LAST READ
F58D: A9 DE  LDA  #$DE  CHAR TO INDICATE ERROR
F58F: 20 ED FD  JSR  COUNT POSITION.
F592: 20 3A FF  RESET2  JSR  BELL
F595: A9 A1  NXTLINE  LDA  #$A1  "1"
F597: B5 33  STA  PROMPT  INITIALIZE PROMPT
F599: 20 67 FD  JSR  GETLN2  GET LINE.
F59C: 20 C7 FF  JSR  2MODE  INIT SCREEN STUFF
F59F: AD 00 02  LDA  IN  GET CHAR
F5A2: C9 A0  CMP  #$A0  ASCII BLANK?
F5A4: F0 11  *  HEI  SPACE  YES
F5A6: C0  INY
F5A7: C9 A4  CMP  #$A4  ASCII ' $ ' IN COL 1?
F5A9: F0 92  BEO  FAKEMON  YES, SIMULATE MONITOR
F5AB: B6 66  DEY  NO, RACKUP 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 PRECEEDING COLON.
F5B6: 20 78 FE  JSR  A1PCLP  MOVE ADR TO PCL, PCH.
F5B9: A9 03  SPACE  LDA  #$3  COUNT OF CHAR IN MNEMONIC
F5BB: 85 3D  STA  AIH
F5BD: 20 34 F6  NXTMN  LSR  GETNWP  GET FIRST MNEM CHAR.
F5C0: 0A  NXM  ASL A
F5C1: E9 BE  SBC  #$BE  SUBTRACT OFFSET
F5C3: C9 C2  CMP  #$C2  LEGAL CHAR?
F5C5: 90 C1  BCC  ERR2  NO.
F5C7: 0A 0A  ASL A  COMPRESS-LEFT JUSTIFY
F5C8: 0A  ASL A
F5C9: A2 04  LDY  #$4
F5CB: 0A  NXM2  ASL A  DO 5 TRIPLE WORD SHIFTS
APPENDIX II FLOATING POINT ROUTINES

COPYRIGHT 1977 BY APPLE COMPUTER INC.

ALL RIGHTS RESERVED

S. GOZNIK

TITLE "FLOATING POINT ROUTINES"

SIGN EEP SF3
X2 EEP SF4
X2 EP5 SF5
X1 EPZ SF8
M1 EPZ SF9
E EPZ SF10

GLEVEL EMU S3P5
ORG SF425

ADD CLC CLEAR CARRY.

LDX #61 INDEX FOR 3-BYTE ADD.

LDA #1.X
ADC #2,X AND A BYTE OF MANT2 TO MANT1.

STA #1.X

DPY INDEX TO NEXT MORE SIGNIF. BYTE.

BLP ADD1 LOOP UNTIL DONE.

RTS RETURN

ASL SIGN CLEAR LSB OF SIGN.

JSR AB3WAP ASS VAL OF X1, THEN SWAP WITH M2

BIT #1 MANT1 NEGATIVE?

BSL AB3WAP1 NO, SWAP WITH MANT2 AND RETURN.

JSR FC00PL YES, COMPLEMENT IT.

INC SIGN 1C0 SIGN, COMPLEMENTING LSB.

AB3WAP1 SEC SET CARRY FOR RETURN TO MUL/DIV.

LDX #$4 INDEX FOR 4-BYTE SWAP.

Swap CLY #1,X SWAP A BYTE OF EXP/MANT1 WITH

STA #1,X EXP/MANT2 AND LEAVE A COPY OF

STA #1-X MANT1 IN X (3 BYTES). F+3 USED

CA DFY ADVANCE INDEX TO NEXT BYTE.

DEP S3P1 LOOP UNTIL DONE.

RETURN

LDA #6E INIT EXP1 TO 14.

STA #1 THEN NORMALIZE TO FLOAT.

LDA #5C0 UPPER TWO BITS UNUSED?

JSR FC00PL CML MANT1,CLEAPS CARRY UNLESS 0

JSR AEGNSKP RIGHT SHIFT MANT1 OR SWAP WITH

JSR ADD ADJIGNED MANTISSAS.

JSR ADDED NO OVERFLOW, NORMALIZE RESULT.

JSR AEGNSKP O/V: SHIFT X1 RIGHT, CARRY INTO SIGN
F47B: 90 C4 ALONG XEI BCC SWAP SWAP IF CARRY CLEAR,
* FUSE SHIFT RIGHT ARITH.
F47D: A5 F9 RTAP LDA X1 SIGN OF MANT1 INTO CARRY FOR
F47E: 0A 2A N1 A RIGHT ARITH SHIFT.
F480: E6 F8 RTLOC INC X1 INC X1 TO ADJUST FOR RIGHT SHIFT
F482: F0 75 REC OVER EXP1 OUT OF RANGE.
F484: A2 FA RTLOG1 LDX #5FE INDEX FOR 6-BYTE RIGHT SHIFT.
F486: 76 FF WOP1 ROR L+3,X
F488: E9 INX NEXT BYTE OF SHIFT.
F489: D0 F9 BNE #01 LOOP UNTIL DONE.
F48B: 60 RTS RETURN.
F48C: 20 32 F4 FMUL JSR #01 ABS VAL OF MANT1, MANT2.
F48E: 55 F8 ADC X1 ADC EXP1 TO EX2 W2 FOR PRODUCT EXP
F491: 20 E2 F4 JSR #02 CHECK PROP. EXP AND PREP. FOR MUL
F494: 1E CLC CLEAR CARRY FOR FIRST BIT.
F495: B0 84 F4 MUL JSR RTLOG1 #1 AND RIGHT (PROD AND MLEIP). IF CARRY CLEAR, SKIP PARTIAL PROD
F49A: 20 25 F4 JSR #03 ADD MULICAND TO PRODUCT.
F49D: 88 FMUL2 DEY NEXT MULT ITERATION.
F49E: 10 F5 RPL NULL LOOP UNTIL DONE.
F4A0: 46 F3 NDFND LSR SIGN TEST SIGN LS.
F4A1: 90 FA NPFX #100 SET CARRY FOR SUBTRACT.
F4A4: 38 FCOMPL SEC SET CARRY FOR SUBTRACT.
F4A5: A2 03 LDL #53 INDEX FOR 3-BYTE SUBTRACT.
F4A7: A9 00 C0PL1 LDA #50 CLEAR A.
F4A8: 95 F8 SBC X1,X SUBTRACT BYTE OF EXP.
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 SEQ ADDENDO NORMALIZE (OR SHIFT RT IF OVL). IF CARRY CLEAR, SKIP PARTIAL PROD
F4B5: F5 F6 FDIV JSR #01 32 32 JSR #01 32 32 TAKE ABS VAL OF MANT1, MANT2.
F4B7: 20 E2 F4 JSR #02 SAVE AS QUOTIENT EXP.
F4BA: 38 DIV1 SEC SET CARRY FOR SUBTRACT.
F4BB: A2 02 LDL #52 INDEX FOR 3-BYTE SUBTRACTION.
F4BD: 85 F5 DIV2 LDA #2,X
F4BF: F5 PC SBC E,X SUBTRACT A BYTE OF E FROM MANT2.
F4C1: 4B PHA SAVE ON STACK.
F4C2: CA DEX NEXT MOST SIGNIFICANT BYTE.
F4C3: A1 08 RPL DIV2 LOOP UNTIL DONE.
F4C5: A2 4D LDL #5FD INDEX FOR 3-BYTE CONDITIONAL MOVE
F4C7: 68 DIV3 PLA FULL BYTE OF DIFFERENCE OFF STACK
F4CB: 90 02 BCC DIV4 IF M2X THEN DON'T RESTORE M2.
F4CA: 95 F8 STA #25,X
F4CC: E8 DIV4 INX NEXT LESS SIGNIFICANT BYTE.
F4CD: D0 F8 BNE DIV3 LOOP UNTIL DONE.
F4CF: 26 FB ROL M1+2 ROLL QUOTIENT LEFT, CARRY INTO LS.
F4D1: 26 FA ROL M1+1
F4D3: 26 F9 ROL M1
F4D5: 06 F7 ALE #2+2
F4D7: 26 F6 ROL #2+1 SHIFT DIVIDEND LEFT.
F4D9: 26 F5 ROL #2
F4DB: B0 1C ACS OVFL OVL IS DUE TO UNNORM DIVISOR
F4DC: 88 DEY NEXT DIVIDE ITERATION.
F4DE: D0 DA BNE DIV1 LOOP UNTIL DONE 23 ITERATIONS.
F4FE: 86 FA MD2 STX #1+2
F4F4: 86 FA STX #1+1
F4F6: 86 F9 STX #1
F4FB: B0 00 ACS OVCHE IF CAC, SET CARRY, CHECK FOR OVL.
F4FA: 30 04 #61 MD3 IF NGE THEN 32 UNDERFLOW.
F4FC: 68 PLA POP ONE RETURN LEVEL.
F4ED: 68 PLA
F4EE: 20 B2 HCC NORKY CLEAR X1 AND RETURN.
F4F0: 49 80 MD3 EOR #180 COMPLEMENT SIGN BIT OF EXPONENT.
F4F2: 85 F6 STA X1 STORE IT.
F4F4: A0 17 LOY #17 COUNT 24 MUL/23 DIV ITERATIONS.
F4F6: 60 PRTS RETURN.
F4F7: 10 F7 OVCHE #03 IF POSITIVE EXP THEN NO OVL.
F4F9: 4C F5 03 OVFL JSR OVLQCY
F4FA: C0 03 OVC CY
F500: 20 7D F4 FIX1 JSR STAR
F504: A5 F8 FIX LDA X1
F506: 10 13 RPL UNDFL
F504: C9 F1 CIP #580
F506: 0D F5 EDV FIX1
F508: 24 F9 BIT #1
F50A: 10 0A RPL FIXPTS
F50C: A5 F8 LDA #1+2
F50E: F0 06 SE0 FIXRTS
F510: 86 F9 INC #1+1
F512: D0 02 BNE FIXRTS
F514: 86 F9 INC #1
F516: 60 FIXRTS RTS
F517: A9 00 UNDFL LDA #50
F519: 85 00 STA #1
F51B: 85 FA STA #1+1
F51D: 60 RTS
********************************************************************
*       APPLE-II PSEUDO  
*       MACHINE INTERPRETER    
*       COPYRIGHT 1977 
*       APPLE COMPUTER INC    
*       ALL RIGHTS RESERVED   
*       S. WOZNIAK             
********************************************************************

TITLE "SWEET16 INTERPRETER"

ROL EPZ $0
ROH EPZ $1
R14d CPZ $10
R15L EPZ $1E
R15H EPZ $1F
SA6PAG EQU $F7
SAVE EQU SFF4A
RESTORE EQU SFF3F
JRG $6F9

F689:  20 4A FF   S.W16  J3H  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 AND PEPS
F692:  20 98 F6   SW16D  JSP SW16C  INTERPRET AND EXECUTE
F695:  4C 92 F6  JMP SW16S ONE SWEET16 INSTR.
F698:  E6 1E SW16C INC R15L
F69A:  D0 02  RGE SW16D INC P SWEET16 PC FOR FETCH
F69C:  E8 1F INC R15H
F69E:  A9 F7 SW16D LDA $816PAG
F6A0:  48 PLA PUSH ON STACK FOR PTS
F6A1:  A0 00 LDX $0
F6A3:  B1 1E LDA (R15L),Y FETCH INSTR
F6A5:  29 0F AND $0F MASK REG SPECIFICATION
F6A7:  0A AYL A DOUBLE FOR 2-BYTE REGISTERS
F6A8:  AA TAX TO X-REG FOR INDEXING
F6A9:  4A LSR A
F6AC:  51 1E EOR (R15L),Y NOT HAVE OPCODE
F6AC:  E0 0B BEQ TOPP IF ZERO THEN NON-REG OP
F6AC:  86 1D STX R14H INDICATE 'PRIOR RESULT REG' INDEX
F6BD:  4A LSR A
F6B1:  4A LSR A OPCODE*2 TO LSR'S
F6B2:  4A LSR A
F6B3:  A5 PAY TO Y-REG FOR INDEXING
F6B4:  B9 F1 F6 LDA CPSH-2,Y LOW-ORDER ADDR BYT
F6B7:  48 PHA ONTO STACK
F6BA:  60 PTS ONTO REG-OP ROUTINE
F6BA:  E6 1E TOPP INC R15L
F6BD:  D0 02 RGE PW2 INC PC
F6BF:  E6 1F INC $15H
F6BF:  E0 F6 TGAS2 LDA BPSH,X LOW-ORDER ADDR 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 RC, INC.
F6C6:  60 PHS OCTO NON-REG OP ROUTINE
F6C7:  68 RTWZ PLA TOP RETURN ADDRESS
F6C9:  20 3F PF JSR RESTORE RESTORE 6502 REG CONTENTS
F6CC:  6C 1E 00 JMP (R15L) RETURN TO 6502 CODE VIA PC
F6CF:  F1 1F SETZ LDA (R15L),Y HIGH-ORDER BYTE OF CONSTANT

96
F601: 95 01 STA ROL,X
F603: 8E DEY
F604: 01 1E LDA ($15L),Y LOW-ORDER BYTE OF CONSTANT
F606: 95 00 STA ROL,X
F608: 98 TYA Y-REG CONTAINS 1
F609: 8E SBC
F61A: 65 1E ADC $15L ADD 2 TO PC
F61C: 85 1E STA $15L
F61E: 90 02 BCC SET2
F61E: 66 1F INC $15H
F622: 60 RTS
F623: 02 SPI1 RTS
F624: 04 OPENL DFP SET-1 (1X)
F626: 90 DFP EN-1 (0)
F627: DF DFP ET-1 (1)
F628: 9E DFP ET-1 (3X)
F629: 25 DFP LAT-1 (4X)
F62A: AF DFP $C-1 (3)
F62B: 16 DFP STP-1 (5X)
F62C: B2 DFP BAR-1 (4)
F62D: 47 DFP LDDAT-1 (6X)
F62E: B9 DFP H8-1 (5)
F62F: 51 DFP STDAT-1 (7X)
F630: C0 DFP RZ-1 (6)
F631: 2F DFP POP-1 (8X)
F632: C9 DFP AM-1-1 (7)
F633: 5E DFP STP-1 (9K)
F634: D2 DFP PM-1-1 (8)
F635: 85 DFP ADD-1 (AX)
F636: DD DFP BM-1-1 (9)
F637: 6E DFP SUS-1 (BX)
F638: 05 DFP R-1 (A)
F639: 33 DFP MPCP-1 (CKX)
F63A: 88 DFP PS-1 (E)
F63B: 70 DFP CPR-1 (DK)
F63C: 93 DFP BS-1 (C)
F63D: 1E DFP INR-1 (FX)
F63E: E7 DFP NUL-1 (D)
F63F: 65 DFP DCR-1 (FX)
F700: E7 DFP NUL-1 (E)
F702: E7 DFP NUL-1 (F)
F703: 10 CA SET BPL SET2 ALWAYS TAKEN
F705: B5 00 LD LDA ROL,X
F705: B5 00 LD LDA ROL,X
F707: 85 00 STA ROL
F709: B5 01 LDA ROL,X MOVE RX TO RO
F70B: 85 01 STA ROL
F70D: 60 RTS
F70E: A5 00 ST LDA ROL
F710: 95 00 STA ROL,X MOVE RO TO RX
F712: A5 01 LDA ROL
F714: 95 01 STA ROL,X
F716: 60 RTS
F717: A5 00 STAT LDA ROL
F719: 81 00 STAT2 STA (ROL,X) STORE BYTE INDIRECT
F71A: 00 00 LDY #$0
F71D: 84 1D STAT3 STY $14H INDICATE RO IS RESULT REG
F71F: F6 00 INR INC ROL,X
F721: D0 02 BNE INR2 INCR RX
F723: F6 01 INC ROL,X
F725: 60 INR2 RTS
F726: A1 00 LDAT LDA (ROL,X) LOAD INDIRECT (RX)
F728: B5 00 STA ROL IO RO
F729: A0 00 LDY #$0
F72C: 84 01 STY ROL ZER0 HIGH-ORDER PO BYTE
F72E: F0 ED BE0 STAT3 ALWAYS TAKEN
F730: A0 00 POP LDY #$0 HIGH ORDER BYTE = 0
F732: F0 06 BE0 POP2 ALWAYS TAKEN
F734: 20 66 F7 POPD JSR DCF DECR RX
F737: A1 00 LDA (ROL,X) POP HIGH-ORDER BYTE RX
F739: A8 TAY SAVE IN Y-REG
F73A: 20 66 F7 POP2 JSR DCF DECR RX
F73D: A1 00 LDA (ROL,X) LOW-ORDER BYTE
F73F: B5 00 STA ROL TO RO
F741: 84 01 ROL
F743: A0 00 POP3 LDY #$0 INDICATE RO AS LAST RSLT REG
F745: 84 1D STY $14H
F747: 60 RTS
F748: 20 26 F7 LDDAT JSR LDA LOW-ORDER BYTE TO RO, INCR PX
F748: A1 00 LDA (ROL,X) HIGH-ORDER BYTE TO RO
F74D: 85 01 STA ROL
F74F: 4C 1F F7 JSR INR INCR RX
F752: 20 17 F7 STDAT JSR STAT STORED INDIRECT LOW-ORDER
F755: A5 01 LDA R0H BYTE AND INCP RX. THEN
F757: 81 00 STA (ROL,X) STORE HIGH-ORDER BYTE.
F759: 4C 1F P7 JMP INR INCR RX AND RETURN
F75C: 20 66 F7 STPAT JSR OCF DECP RX
F75F: A5 00 LDA ROL
F761: E1 00 STA (POL,X) STORE LOW BYTE @RX
F763: 4C 43 P7 JMP POLP INDICATE PO AS LAST Rslt REG
F766: B5 00 DCP LDA ROL,X
F768: D0 02 BNE DCRC2 DECP RX
F76A: D6 01 DEC ROLH
F76C: D6 00 DCR2 DEC ROL,X
F76E: 60 RTS
F76F: A0 00 SUI LDY #$0 RESULT TO RO
F771: 38 CPR SEC NOTE Y-REG = 13*2 FOR CPR
F772: A5 00 LDA ROL
F774: F5 00 SBC ROL,X
F776: 99 00 00 STA ROL,Y RO-RX TO RY
F779: A5 01 LDA ROH
F77B: F5 01 SBC ROH,X
F77D: 99 00 00 SUB2 STA ROH,Y
F780: 9E TAY 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 ROL
F788: 75 00 ADC ROL,X
F78A: B5 00 STA ROL RO+RX TO RO
F78C: A5 01 LDA ROH
F78E: 75 01 ADC ROH,X
F790: A0 00 LDY #$0 RO FOR RESULT
F792: F0 E9 BEO SUB2 FINISH ADD
F794: A5 1E BS LDA R15L NOTE X-REG IS 12*21
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: 80 0E BNC ECS BNC2 NO CARRY TEST
F7A1: B1 1F BR1 LDA (R15),Y DISPLACEMENT BYTE
F7A3: 10 01 SPL BR2
F7A5: 88 DEY
F7A6: 65 1E BR2 ADC R15L ADD TO PC
F7A8: 85 1E STA R15L
F7AA: 98 TAY
F7AB: 65 1F ADC R15H
F7AD: 85 1F STA R15H
F7AF: 60 BNC2 RTS
F7B0: 80 EC SC RCS SR
F7B2: 60 RTS
F7B3: 0A 3P 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 8B SPL EPI BRANCH IF SO
F7B9: 60 RTS
F7BA: 0A 3P ASL A DOUBLE RESULT-REG INDEX
F7BB: AA TAX
F7BC: B5 01 LDA R0H,X TEST FOR MINUS
F7BD: 30 E1 BMI MRI
F7C0: 60 RTS
F7C1: 0A 82 ASL A DOUBLE RESULT-REG INDEX
F7C2: AA TAX
F7C3: B5 00 LDA ROL,X TEST FOR ZERO
F7C5: 15 01 OPA ROL,X (AND BYTES)
F7C7: F0 0A DEO #F1 BRANCH IF SO
F7C9: 60 RTS
F7CA: 0A 3P ASL A DOUBLE RESULT-REG INDEX
F7CB: AA TAX
F7CC: B5 00 LDA ROL,X TEST FOR NONZERO
F7CE: 15 01 ORA ROH,X (AND BYTES)
F7D0: D0 CF BNE BR1 BRANCH IF SO
F7D2: 60 RTS
F7D3: 0A ASL A DOUBLE RESULT-REG INDEX
F7D4: AA TAX
F7D5: B5 00 LDA ROL,X CHECK BOTH BYTES
F7D7: 35 01 AND ROH,X FOR $FF (MINUS 1)
F7D9: 49 FF EOR #$FF
F7DA: P0 C4 BEQ #F1 BRANCH IF SO
F7DD: 60 RTS
F7DE: 0A ASL A DOUBLE RESULT-REG INDEX
F7DF: AA TAX
F7E0: B5 00 LDA ROL,X CHECK BOTH BYTES FOR NO $FF
F7E2: 35 01 AND ROH,X
F7E4: 49 FF EOR #$FF
F7E6: D0 09 BNE #F1 BRANCH IF NOT MINUS 1
F7E8: 60 RTS
F7E9: A2 18 RS LDX #$18 12*2 FOR R12 AS STK POINTER
F7EB: 20 66 F7  JSR  DCR  DCR  STACK  POINTER
F7EE: A1 00  LDA  (R0L,X)  POP  HIGH  RETURN  ADDR  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

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

A: Accumulator
X, Y: Index Registers
M: Memory
B: Borrow
P: Processor Status Register
S: Stack Pointer
V: Change
_: No Change
+ Add
- Subtract
\$: Logical Exclusive Or
\& Transfer From Stack
\# Transfer To Stack
\( \rightarrow \) Transfer To
\( \leftarrow \) Transfer To
^ 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

\[ \begin{array}{cccccccc}
  & C & 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 & 0 \\
\end{array} \]

FIGURE 2. ROTATE ONE BIT LEFT (MEMORY OR ACCUMULATOR)

\[ \begin{array}{cccccccc}
  & M \text{ OR } A & 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 & C \\
\end{array} \]

FIGURE 3. 

\[ \begin{array}{cccccccc}
  & C & 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 & \\
\end{array} \]

NOTE 1: BIT — TEST BITS

Bit 6 and 7 are transferred to the status register. If the result of A \& M is zero then Z=1, otherwise Z=0.

PROGRAMMING MODEL

\[ \begin{array}{cccccccc}
  & 7 & 0 & & & & & \\
  & A & & & & & & \\
  & 7 & 0 & & & & & \\
  & Y & & & & & & \\
  & 7 & 0 & & & & & \\
  & X & & & & & & \\
  & 15 & 7 & 0 & & & & \\
  & PCH & PCL & & & & & \\
  & 7 & 0 & & & & & \\
  & 01 & S & & & & & \\
\end{array} \]

ACCUMULATOR
INDEX REGISTER Y
INDEX REGISTER X
PROGRAM COUNTER
STACK POINTER

\[ \begin{array}{cccccccc}
  & 7 & 0 & & & & & \\
  & \text{PROCESSOR STATUS REGISTER, "P"} & & & & & & \\
  & \text{CARRY} & & & & & & & & & & & & & & & \\
  & \text{ZERO} & & & & & & & & & & & & & & & \\
  & \text{INTERRUPT DISABLE} & & & & & & & & & & & & & & & \\
  & \text{DECIMAL MODE} & & & & & & & & & & & & & & & \\
  & \text{BREAK COMMAND} & & & & & & & & & & & & & & & \\
  & \text{OVERFLOW} & & & & & & & & & & & & & & & \\
  & \text{NEGATIVE} & & & & & & & & & & & & & & & \\
\end{array} \]

101
## INSTRUCTION CODES

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Operation</th>
<th>Addressing Mode</th>
<th>Assembly Language Form</th>
<th>HEX DP Code</th>
<th>No. Bytes</th>
<th>&quot;P&quot; Status Reg. N Z C I D V</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC</td>
<td>Add memory to accumulator with carry</td>
<td>A · M · C → A · C</td>
<td>Immediate</td>
<td>ADC ≤ Oper</td>
<td>69</td>
<td>2</td>
<td>V V V V</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Zero Page</td>
<td>ADC Oper X</td>
<td>65</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Zero Page, X</td>
<td>ADC Oper Y</td>
<td>75</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Absolute</td>
<td>ADC Oper X</td>
<td>60</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Absolute, X</td>
<td>ADC Oper X</td>
<td>70</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Absolute, Y</td>
<td>ADC Oper Y</td>
<td>79</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(indirect), X</td>
<td>ADC Oper Y</td>
<td>61</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(indirect), Y</td>
<td>ADC Oper Y</td>
<td>71</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>AND</td>
<td>&quot;AND&quot; memory with accumulator</td>
<td>A · A · M → A</td>
<td>Immediate</td>
<td>AND ≤ Oper</td>
<td>29</td>
<td>2</td>
<td>V V V V</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Zero Page</td>
<td>AND Oper</td>
<td>25</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Zero Page, X</td>
<td>AND Oper X</td>
<td>35</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Absolute</td>
<td>AND Oper</td>
<td>20</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Absolute, X</td>
<td>AND Oper X</td>
<td>30</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Absolute, Y</td>
<td>AND Oper</td>
<td>39</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(indirect), X</td>
<td>AND Oper X</td>
<td>21</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(indirect), Y</td>
<td>AND Oper</td>
<td>31</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>ASL</td>
<td>Shift left one bit (Memory or Accumulator)</td>
<td>(See Figure 1)</td>
<td>Accumulator</td>
<td>ASL A</td>
<td>0A</td>
<td>1</td>
<td>V V</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Zero Page</td>
<td>ASL Oper</td>
<td>06</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Zero Page, X</td>
<td>ASL Oper X</td>
<td>16</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Absolute</td>
<td>ASL Oper</td>
<td>0E</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Absolute, X</td>
<td>ASL Oper X</td>
<td>1E</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>BCC</td>
<td>Branch on carry clear</td>
<td>Branch on C=0</td>
<td>Relative</td>
<td>BCC Oper</td>
<td>90</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>BCS</td>
<td>Branch on carry set</td>
<td>Branch on C=1</td>
<td>Relative</td>
<td>BCS Oper</td>
<td>90</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>BEQ</td>
<td>Branch on result zero</td>
<td>Branch on Z=1</td>
<td>Relative</td>
<td>BEQ Oper</td>
<td>90</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>BIT</td>
<td>Test bits in memory with accumulator</td>
<td>A · A · M · M2 → N, M1 → Y</td>
<td>Zero Page</td>
<td>BIT ≤ Oper</td>
<td>24</td>
<td>2</td>
<td>M2 V V V V, M1</td>
</tr>
<tr>
<td>BMI</td>
<td>Branch on result minus</td>
<td>Branch on N=1</td>
<td>Relative</td>
<td>BMI Oper</td>
<td>30</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>BNE</td>
<td>Branch on result not zero</td>
<td>Branch on Z=0</td>
<td>Relative</td>
<td>BNE Oper</td>
<td>30</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>BPL</td>
<td>Branch on result plus</td>
<td>Branch on N=0</td>
<td>Relative</td>
<td>BPL Oper</td>
<td>30</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>BRK</td>
<td>Force Break</td>
<td>Forced Interrupt PC=2 + P #</td>
<td>Implied</td>
<td>BRK*</td>
<td>00</td>
<td>1</td>
<td>1 1 1 1</td>
</tr>
<tr>
<td>BVC</td>
<td>Branch on overflow clear</td>
<td>Branch on V=0</td>
<td>Relative</td>
<td>BVC Oper</td>
<td>50</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

### Notes
- "N" is set if negative, otherwise 0.
- "Z" is set if zero, otherwise 1.
- "C" is set if carry, otherwise 1.
- "I" is the interrupt enable flag.
- "D" is the decimal mode flag.
- "V" is set if overflow, otherwise 1.

### Other
- A BRK command is stored as BRK*.
- BRK* is stored in the program counter.
- BRK* is used for forced interrupts.
- BRK* is used for program flow control.
- BRK* is used for program flow control.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Operation</th>
<th>Addressing Mode</th>
<th>Assembly Language Form</th>
<th>HEX DP Code</th>
<th>No. Bytes</th>
<th>&quot;P&quot; Status Reg. N Z C I D V</th>
</tr>
</thead>
<tbody>
<tr>
<td>BVS</td>
<td>Branch on overflow set</td>
<td>Branch on V=1</td>
<td>Relative</td>
<td>BVS Oper</td>
<td>70</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>CLC</td>
<td>Clear carry flag</td>
<td>D → C</td>
<td>Implied</td>
<td>CLC</td>
<td>18</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>CLD</td>
<td>Clear decimal mode</td>
<td>D → D</td>
<td>Implied</td>
<td>CLD</td>
<td>88</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>CLI</td>
<td>Clear interrupt</td>
<td>D → I</td>
<td>Implied</td>
<td>CLI</td>
<td>58</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>CLV</td>
<td>Clear overflow flag</td>
<td>D → V</td>
<td>Implied</td>
<td>CLV</td>
<td>88</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>CMP</td>
<td>Compare memory and accumulator</td>
<td>A → M</td>
<td>Immediate</td>
<td>CMP ≤ Oper</td>
<td>9</td>
<td>2</td>
<td>V V V V</td>
</tr>
<tr>
<td>CPX</td>
<td>Compare memory and index X</td>
<td>X → M</td>
<td>Immediate</td>
<td>CPX ≤ Oper</td>
<td>9</td>
<td>2</td>
<td>V V</td>
</tr>
<tr>
<td>CPY</td>
<td>Compare memory and index Y</td>
<td>Y → M</td>
<td>Immediate</td>
<td>CPY ≤ Oper</td>
<td>9</td>
<td>2</td>
<td>V V</td>
</tr>
<tr>
<td>DEC</td>
<td>Decrement memory by one</td>
<td>M → M</td>
<td>Immediate</td>
<td>DEC Oper</td>
<td>9</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>DEX</td>
<td>Decrement index X by one</td>
<td>X → X</td>
<td>Implied</td>
<td>DEX</td>
<td>9</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>DEY</td>
<td>Decrement index Y by one</td>
<td>Y → Y</td>
<td>Implied</td>
<td>DEY</td>
<td>88</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

### Notes
- "C" is set if carry, otherwise 1.
- "Z" is set if zero, otherwise 1.
- "I" is the interrupt enable flag.
- "D" is the decimal mode flag.
- "V" is set if overflow, otherwise 1.

### Other
- A BRK command is stored as BRK*.
- BRK* is stored in the program counter.
- BRK* is used for forced interrupts.
- BRK* is used for program flow control.
- BRK* is used for program flow control.

---

*P* is the program status register.
## INSTRUCTION CODES

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>EOR</td>
<td>&quot;Exclusive-Or&quot; memory with accumulator</td>
<td>A VM → A</td>
<td>Immediate</td>
<td>EOR #Op</td>
<td>49</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Zero Page</td>
<td>EOR Op</td>
<td>45</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zero Page,X</td>
<td>EOR Op,X</td>
<td>55</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Absolute,X</td>
<td>EOR Op,X</td>
<td>40</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Absolute,Y</td>
<td>EOR Op,Y</td>
<td>50</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Indirect,X)</td>
<td>EOR (Op,X)</td>
<td>59</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Indirect,Y)</td>
<td>EOR (Op,Y)</td>
<td>41</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>51</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INC</td>
<td>Increment memory by one</td>
<td>M + 1 → M</td>
<td>Zero Page</td>
<td>INC #Op</td>
<td>E6</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Zero Page,X</td>
<td>INC Op,X</td>
<td>F6</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Absolute,X</td>
<td>INC Op,X</td>
<td>EE</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Absolute,Y</td>
<td>INC Op,Y</td>
<td>FE</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INX</td>
<td>Increment index X by one</td>
<td>X + 1 → X</td>
<td>Implied</td>
<td>INX</td>
<td>E8</td>
<td>1</td>
</tr>
<tr>
<td>INY</td>
<td>Increment index Y by one</td>
<td>Y + 1 → Y</td>
<td>Implied</td>
<td>INY</td>
<td>C8</td>
<td>1</td>
</tr>
<tr>
<td>JMP</td>
<td>Jump to new location</td>
<td>(PC+1) → PCL</td>
<td>Absolute</td>
<td>JMP #Op</td>
<td>4C</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>(PC+2) → PCH</td>
<td>JMP (Op)</td>
<td>6C</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JSR</td>
<td>Jump to new location saving return address</td>
<td>PC-2 → PC</td>
<td>Absolute</td>
<td>JSR #Op</td>
<td>20</td>
<td>3</td>
</tr>
<tr>
<td>LDA</td>
<td>Load accumulator with memory</td>
<td>M → A</td>
<td>Immediate</td>
<td>LDA #Op</td>
<td>A9</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Zero Page</td>
<td>LDA Op</td>
<td>A5</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zero Page,X</td>
<td>LDA Op,X</td>
<td>B5</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Absolute,X</td>
<td>LDA Op,X</td>
<td>A0</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Absolute,Y</td>
<td>LDA Op,Y</td>
<td>B0</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Indirect,X)</td>
<td>LDA (Op,X)</td>
<td>A1</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Indirect,Y)</td>
<td>LDA (Op,Y)</td>
<td>B1</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDX</td>
<td>Load index X with memory</td>
<td>M → X</td>
<td>Immediate</td>
<td>LDX #Op</td>
<td>A2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Zero Page</td>
<td>LDX Op</td>
<td>A6</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zero Page,Y</td>
<td>LDX Op,Y</td>
<td>B6</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Absolute,X</td>
<td>LDX Op,X</td>
<td>AE</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Absolute,Y</td>
<td>LDX Op,Y</td>
<td>BE</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDY</td>
<td>Load index Y with memory</td>
<td>M → Y</td>
<td>Immediate</td>
<td>LDY #Op</td>
<td>A0</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Zero Page</td>
<td>LDY Op</td>
<td>A4</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zero Page,X</td>
<td>LDY Op,X</td>
<td>B4</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Absolute,X</td>
<td>LDY Op,X</td>
<td>AC</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Absolute,Y</td>
<td>LDY Op,Y</td>
<td>BC</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LSR</td>
<td>Shift right one bit (memory or accumulator)</td>
<td>(See Figure 1)</td>
<td>Accumulator</td>
<td>LSR A</td>
<td>4A</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Zero Page</td>
<td>LSR Op</td>
<td>46</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zero Page,X</td>
<td>LSR Op,X</td>
<td>56</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Absolute,X</td>
<td>LSR Op,X</td>
<td>4E</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Absolute,Y</td>
<td>LSR Op,Y</td>
<td>5E</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOP</td>
<td>No operation</td>
<td>No Operation</td>
<td>Implied</td>
<td>NOP</td>
<td>EA</td>
<td>1</td>
</tr>
<tr>
<td>ORA</td>
<td>&quot;OR&quot; memory with accumulator</td>
<td>A VM → A</td>
<td>Immediate</td>
<td>ORA #Op</td>
<td>09</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Zero Page</td>
<td>ORA Op</td>
<td>05</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zero Page,X</td>
<td>ORA Op,X</td>
<td>15</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Absolute,X</td>
<td>ORA Op,X</td>
<td>0D</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Absolute,Y</td>
<td>ORA Op,Y</td>
<td>1D</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Indirect,X)</td>
<td>ORA (Op,X)</td>
<td>01</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Indirect,Y)</td>
<td>ORA (Op,Y)</td>
<td>11</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PHA</td>
<td>Push accumulator on stack</td>
<td>A →</td>
<td>Implied</td>
<td>PHA</td>
<td>48</td>
<td>1</td>
</tr>
<tr>
<td>PHP</td>
<td>Push processor status on stack</td>
<td>P →</td>
<td>Implied</td>
<td>PHP</td>
<td>08</td>
<td>1</td>
</tr>
<tr>
<td>PLA</td>
<td>Pull accumulator from stack</td>
<td>A</td>
<td>Implied</td>
<td>PLA</td>
<td>68</td>
<td>1</td>
</tr>
<tr>
<td>PLP</td>
<td>Pull processor status from stack</td>
<td>P</td>
<td>Implied</td>
<td>PLP</td>
<td>28</td>
<td>1</td>
</tr>
<tr>
<td>ROL</td>
<td>Rotate one bit left (memory or accumulator)</td>
<td>(See Figure 2)</td>
<td>Accumulator</td>
<td>ROL A</td>
<td>2A</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Zero Page</td>
<td>ROL Op</td>
<td>26</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zero Page,X</td>
<td>ROL Op,X</td>
<td>36</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Absolute,X</td>
<td>ROL Op,X</td>
<td>2E</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Absolute,Y</td>
<td>ROL Op,Y</td>
<td>3E</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROR</td>
<td>Rotate one bit right (memory or accumulator)</td>
<td>(See Figure 3)</td>
<td>Accumulator</td>
<td>ROR A</td>
<td>6A</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Zero Page</td>
<td>ROR Op</td>
<td>66</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zero Page,X</td>
<td>ROR Op,X</td>
<td>76</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Absolute,X</td>
<td>ROR Op,X</td>
<td>6F</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Absolute,Y</td>
<td>ROR Op,Y</td>
<td>7E</td>
<td>3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

103
## INSTRUCTION CODES

<table>
<thead>
<tr>
<th>Name Description</th>
<th>Operation</th>
<th>Addressing Mode</th>
<th>Assembly Language Form</th>
<th>HEX DP Code</th>
<th>No. Bytes</th>
<th>&quot;P&quot; Status Reg. N Z C I D V</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTI Return from interrupt</td>
<td>P → PC</td>
<td>Implied</td>
<td>RTI</td>
<td>40</td>
<td>1</td>
<td>From Stack</td>
</tr>
<tr>
<td>RTS Return from subroutine</td>
<td>PC, PC → PC</td>
<td>Implied</td>
<td>RTS</td>
<td>1</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>SBC Subtract memory from accumulator with borrow</td>
<td>A : M : C → A</td>
<td>Immediate Zero Page Zero Page, X Absolute Absolute, X Absolute, Y (indirect,X) (indirect,Y)</td>
<td>SBC #Op SBC Op SBC Op,X SBC Op, Y SBC (Op),X SBC (Op),Y</td>
<td>E9 2 E5 2 F5 2 ED 3 FD 3 F9 2 E1 2 F1 2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>SEC Set carry flag</td>
<td>1 → C</td>
<td>Implied</td>
<td>SEC</td>
<td>38</td>
<td>1</td>
<td>---</td>
</tr>
<tr>
<td>SED Set decimal mode</td>
<td>1 → D</td>
<td>Implied</td>
<td>SED</td>
<td>F8</td>
<td>1</td>
<td>---</td>
</tr>
<tr>
<td>SEI Set interrupt disable status</td>
<td>1 → I</td>
<td>Implied</td>
<td>SEI</td>
<td>78</td>
<td>1</td>
<td>---</td>
</tr>
<tr>
<td>STA Store accumulator in memory</td>
<td>A → M</td>
<td>Zero Page Zero Page, X Absolute Absolute, X Absolute, Y (indirect,X) (indirect,Y)</td>
<td>STA Op STA Op,X STA Op, Y STA (Op),X STA (Op),Y</td>
<td>85 2 95 2 8D 3 9D 3 81 2 91 2</td>
<td>2</td>
<td>---</td>
</tr>
<tr>
<td>STX Store index X in memory</td>
<td>X → M</td>
<td>Zero Page Zero Page, Y Absolute</td>
<td>STX Op STX Op, Y STX Op</td>
<td>86 2 96 2 86 3</td>
<td>2</td>
<td>---</td>
</tr>
<tr>
<td>STY Store index Y in memory</td>
<td>Y → M</td>
<td>Zero Page Zero Page, X Absolute</td>
<td>STY Op STY Op, X STY Op</td>
<td>94 2 94 2 8C 3</td>
<td>2</td>
<td>---</td>
</tr>
<tr>
<td>TAX Transfer accumulator to index X</td>
<td>A → X</td>
<td>Implied</td>
<td>TAX</td>
<td>AA</td>
<td>1</td>
<td>---</td>
</tr>
<tr>
<td>TAY Transfer accumulator to index Y</td>
<td>A → Y</td>
<td>Implied</td>
<td>TAY</td>
<td>A8</td>
<td>1</td>
<td>---</td>
</tr>
<tr>
<td>TSX Transfer stack pointer to index X</td>
<td>S → X</td>
<td>Implied</td>
<td>TSX</td>
<td>BA</td>
<td>1</td>
<td>---</td>
</tr>
<tr>
<td>Code</td>
<td>Instruction</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>-------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>BRK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>ORA - (Indirect, X)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>03</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>04</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>05</td>
<td>ORA - Zero Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>06</td>
<td>ASL - Zero Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>07</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>08</td>
<td>PHP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>09</td>
<td>ORA - Immediate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0A</td>
<td>ASL - Accumulator</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0B</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0C</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0D</td>
<td>ORA - Absolute</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0E</td>
<td>ASL - Absolute</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0F</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>BPL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>ORA - (Indirect, Y)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>ORA - Zero Page, X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>ASL - Zero Page, X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>CLC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>ORA - Absolute, Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1A</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1B</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1C</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1D</td>
<td>ORA - Absolute, X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1E</td>
<td>ASL - Absolute, X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1F</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>JSR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>AND - (Indirect, X)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>BIT - Zero Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>AND - Zero Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>ROL - Zero Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>PLP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>AND - Immediate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2A</td>
<td>ROL - Accumulator</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2B</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2C</td>
<td>BIT - Absolute</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2D</td>
<td>AND - Absolute</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2E</td>
<td>ROL - Absolute</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2F</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>BMI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>AND - (Indirect, Y)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>AND - Zero Page, X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>ROL - Zero Page, X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>SEC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>AND - Absolute, Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3A</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3B</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3C</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3D</td>
<td>AND - Absolute, X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3E</td>
<td>ROL - Absolute, X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3F</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>RTI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>EOR - (Indirect, X)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>EOR - Zero Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>LSR - Zero Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>47</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>PHA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>EOR - Immediate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4A</td>
<td>LSR - Accumulator</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4B</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4C</td>
<td>JMP - Absolute</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4D</td>
<td>EOR - Absolute</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4E</td>
<td>LSR - Absolute</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4F</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>BVC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>51</td>
<td>EOR (Indirect, Y)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>52</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>53</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>54</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>55</td>
<td>EOR - Zero Page, X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>56</td>
<td>LSR - Zero Page, X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>57</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>58</td>
<td>CLI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>59</td>
<td>EOR - Absolute, Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5A</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5B</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5C</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5D</td>
<td>EOR - Absolute, X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5E</td>
<td>LSR - Absolute, X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5F</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>RTS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>61</td>
<td>ADC - (Indirect, X)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>62</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>63</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>65</td>
<td>ADC - Zero Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>66</td>
<td>ROR - Zero Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>67</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>68</td>
<td>PLA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>69</td>
<td>ADC - Immediate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6A</td>
<td>ROR - Accumulator</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6B</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6C</td>
<td>JMP - Indirect</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6D</td>
<td>ADC - Absolute</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6E</td>
<td>ROR - Absolute</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6F</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>BVS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>71</td>
<td>ADC - (Indirect), Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>72</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>73</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>74</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>75</td>
<td>ADC - Zero Page, X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>76</td>
<td>ROR - Zero Page, X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>77</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>78</td>
<td>SEI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>79</td>
<td>ADC - Absolute, Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7A</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7B</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7C</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7D</td>
<td>ADC - Absolute, X NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7E</td>
<td>ROR - Absolute, X NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7F</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>81</td>
<td>STA - (Indirect, X)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>82</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>83</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>84</td>
<td>STY - Zero Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>85</td>
<td>STA - Zero Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>86</td>
<td>STX - Zero Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>87</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>88</td>
<td>DEX</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>89</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8A</td>
<td>TXA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8B</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8C</td>
<td>STY - Absolute</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8D</td>
<td>STA - Absolute</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8E</td>
<td>STX - Absolute</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8F</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>BCC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>91</td>
<td>STA - (Indirect, Y)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>92</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>93</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>94</td>
<td>STY - Zero Page, X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>95</td>
<td>STA - Zero Page, X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>96</td>
<td>STX - Zero Page, Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>97</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>98</td>
<td>TYA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>99</td>
<td>STA - Absolute, Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9A</td>
<td>TXS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9B</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9C</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9D</td>
<td>STA - Absolute, X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9E</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9F</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A0</td>
<td>LDY - Immediate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A1</td>
<td>LDA - (Indirect, X)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A2</td>
<td>LDX - Immediate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A3</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A4</td>
<td>LDY - Zero Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A5</td>
<td>LDA - Zero Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A6</td>
<td>LDX - Zero Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A7</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A8</td>
<td>TAY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A9</td>
<td>LDA - Immediate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AA</td>
<td>TAX</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AB</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AC</td>
<td>LDY - Absolute</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AD</td>
<td>Absolute</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AE</td>
<td>LDX - Absolute</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AF</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B0</td>
<td>BCS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B1</td>
<td>LDA - (Indirect, Y)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B2</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B3</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B4</td>
<td>LDY - Zero Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B5</td>
<td>LDA - Zero Page, X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B6</td>
<td>LDS - Zero Page, Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B7</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B8</td>
<td>CLV</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B9</td>
<td>LDA - Absolute, Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BA</td>
<td>TXS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BB</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BC</td>
<td>LDY - Absolute, X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BD</td>
<td>LDA - Absolute, X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BE</td>
<td>LDS - Absolute, Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BF</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C0</td>
<td>CPY - Immediate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C1</td>
<td>CMP - (Indirect, X)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C3</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4</td>
<td>CPY - Zero Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C5</td>
<td>CMP - Zero Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C6</td>
<td>DEC - Zero Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C7</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C8</td>
<td>INY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C9</td>
<td>CMP - Immediate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CA</td>
<td>DEX</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CB</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CC</td>
<td>CPY - Absolute</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>CMP - Relative</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CE</td>
<td>DEC - Absolute</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CF</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D0</td>
<td>BNE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D1</td>
<td>CMP - (Indirect, Y)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D2</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D4</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D5</td>
<td>CMP - Zero Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D6</td>
<td>DEC - Zero Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D7</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D8</td>
<td>CLD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D9</td>
<td>CMP - Absolute, Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DA</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DB</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DC</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DD</td>
<td>CMP - Absolute</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DE</td>
<td>DEC - Absolute, X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DF</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E0</td>
<td>CPX - Immediate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E1</td>
<td>SBC - (Indirect, X)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E2</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E3</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E4</td>
<td>CPX - Zero Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E5</td>
<td>SBC - Zero Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E6</td>
<td>INC - Zero Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E7</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E8</td>
<td>INX</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E9</td>
<td>SBC - Immediate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EA</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EB</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EC</td>
<td>CPY - Absolute</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ED</td>
<td>SBC - Absolute</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EE</td>
<td>INC - Absolute</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EF</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F0</td>
<td>BED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F1</td>
<td>SBC - (Indirect, Y)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F2</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F3</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F4</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F5</td>
<td>SBC - Zero Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F6</td>
<td>INC - Zero Page, X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F7</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F8</td>
<td>SED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F9</td>
<td>SBC - Absolute, Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FA</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FB</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FC</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FD</td>
<td>SBC - Absolute, X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FE</td>
<td>INC - Absolute, X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FF</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
APPLE II HARDWARE

1. Getting Started with Your APPLE II Board
2. APPLE II Switching Power Supply
3. Interfacing with the Home TV
4. Simple Serial Output
5. Interfacing the APPLE — Signals, Loading, Pin Connections
6. Memory — Options, Expansion, Map, Address
7. System Timing
8. Schematics
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-verte"™. 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 - 350mA.
   b. For 8K, 20K or 32K - 550mA.
   c. For 12K, 24K, 36K or 48K - 850mA.

2. +5 Volts at 1.6 amps

3. -5 Volts at 10mA.

4. OPTIONAL: If -12 Volts is required by your keyboard. (If using an APPLE II supplied keyboard, you will need -12V at 50mA.)
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:

   1. 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 minimize resistance. With cable assembly disconnected from APPLE II motherboard, 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 cabling.
POWER UP

1. Turn power on. If power supplies overload, immediately turn off and recheck power cable wiring. Verify operating supply voltages are within +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.
THE APPLE II SWITCHING POWER SUPPLY

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 wire 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 temperatures. An over-voltage crowbar shutdown system and an auxiliary control feedback loop are provided to ensure that even very unlikely failure modes will not cause damage to the APPLE 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., 197V to 132V.

Under no circumstances, should more than 140 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 \( \frac{1}{8} \) that of the +5V. The -5V supply load is \( \frac{1}{10} \) that of the +5V. The -12V supply load is \( \frac{1}{10} \) that of the +5V.

The supply voltages are +5.0 ± 0.15 volts, +11.8 ± 0.5 volts, -12.0 ± 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 250 mA
+ 5V - use no more than 500 mA
- 5V - use no more than 200 mA
- 12V - use no more than 200 mA

The power supply is allowed to run indefinitely 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 40 to 50 db below the signal level to reduce worms to an acceptable level. When it is remembered that only 2 to 6 mV (across 300Ω) is presented to the VHF input of the TV set, then stray RFI getting into the TV must be less than 50μV 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 be enclosed 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 co-ax 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 A2M010X. The M&R "Sup-r-Mod" is supplied with a co-ax 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 #03-09-2151 or #03-09-2153. Recommended terminals are Molex #02-09-2136. 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 $3E9. 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 8Q column printer as the primary output device.
2. PR#0 will inactivate the printer transferring control back to the Video monitor as the primary output device.

3. In Monitor mode $370G 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 "370.03E9W 370.03E9W".
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. BC (control B) into BASIC
2. Turn the teletype (printer on)
3. Type in the following
   10 CALL 880
   15 PRINT "ABCD...XYZ01123456789"
   20 PR#0
   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 10 activates the teletype machine routine and all "PRINT" statements following it will be printed to the teletype until a PR#0 statement is encountered. Then the text in line 15 will appear on the teletype's output. Line 20 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-33 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.
FIGURE 1  ASR-33

FIGURE 2  RS-232
TELETYPE DRIVER Routines

3:42 P.M., 11/18/1977

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26

***********************

**meldwidth EQU $21 ; FOR APPLE-II

ch EQU $24 ; cursor horiz.

Csvl EQU $36 ; char. out switch

ysave EQU $778 ; column count loc.

colcnt EQU $7f8

mark EQU $c058

space EQU $c059

wait EQU $fca8

org $370

****************************

***WARNING: OPERAND OVERFLOW IN LINE 27

0370: A9 82

0372: 85 36

0374: A9 03

0376: 85 37

0378: A9 48

037a: 85 21

037c: a5 24

037e: 8d f8 07 34

0381: 60 35

0382: 48 36

0383: 48 37

0384: ad f8 07 38

0386: c5 24 39

0389: 68 40

038a: b0 03 41

038c: 48 42

038d: a9 a0 43

038f: 2c c0 03 44

0392: f0 03 45

0394: ee f8 07 46

0397: 20 c1 03 47

039a: 68 48

039b: 48 49

039c: 90 e6 50

039e: 49 0d 51

03a0: 0a 52

03a1: d0 0d 53

**TTINIT: LDA #ttout

sta csvl

lda #ttout/256 ; high byte

sta csvl+1

lda #$72 ; set window width

sta wndwidth ; to number columns on tv

lda ch

sta colcnt ; where we are now.

rts

Ttout: pha ; save twice

tstph: pha ; on stack.

tstph: lda colcnt ; check for a tab.

cmp ch

pla ; restore output char.

bcc testctrl ; if c set, no tab

lda #$ao ; print a space.

testctrl: bit rtsi ; trick to determine

beq prntit ; if control char.

inc colcnt ; if not, add one to col

jsr dochar ; print the char on tty

pltn ; restore char

pla ; and put back on stack

phln ; and print.

tstabh: bcc ttout2 ; do more spaces for tab

eor #$0d ; check for car ret.

asl a ; elim parity

bnh finish ; if not cr, done.

**FIGURE 3a
TELETYPE DRIVER ROUTINES

3:42 P.M., 11/18/1977

03A3:  B0 8B 07 54
03A6:  A9 8A 55
03A8:  20 C1 03 56
03AB:  A9 58 57
03AD:  20 A8 FC 58
03B0:  AD F8 07 59
03B3:  F0 08 60
03B5:  E5 21 61
03B7:  E9 F7 62
03B9:  90 04 63
03BA:  69 1F 64
03BD:  85 24 65
03BF:  68 66
03C0:  60 67
03C1:  8C 78 07 69
03C4:  08 70
03C7:  A0 03 71
03C8:  18 72
03C9:  80 05 73
03CA:  AD 59 CO 74
03CB:  90 03 75
03DD:  AD 58 CO 76
03D3:  A9 D7 77
03D4:  48 78
03D5:  A9 20 80
03D6:  4A 81
03D7:  90 FD 82
03DA:  68 83
03DC:  E9 01 84
03DE:  D0 F5 85
03E0:  68 86
03E1:  6A 87
03E2:  88 88
03E3:  D0 E3 89
03E5:  AC 78 07 90
03E8:  28 91
03E9:  60 92

03A3: STA COLCNT ; CLEAR COLUMN COUNT
03A6: LDA #$8A ; NOW DO LINE FEED
03A8: JSR DOCHAR
03AB: LDA #$53
03AD: JSR WAIT ; 200MSEC DELAY FOR LINE
03B0: FINISH: LDA COLCNT ; CHECK IF IN MARGIN
03B3: BEQ SETCH ; FOR CR, RESET CH
03B5: SBC NWDTH ; IF SO, CARRY SET.
03B7: SBC #$F7
03B9: BCC RETURN
03BA: ADC #$1F ; ADJUST CH
03BD: SETCH: STA CH ; RETURN TO CALLER ROUTINE:
03BF: RETURN: PLA
03C0: RTS
03C1: 8C 78 07 69
03C4: 08 70
03C7: A0 03 71
03C8: 18 72
03C9: 80 05 73
03CA: AD 59 CO 74
03CB: 90 03 75
03DD: AD 58 CO 76
03D3: A9 D7 77
03D4: 48 78
03D5: A9 20 80
03D6: 4A 81
03D7: 90 FD 82
03DA: 68 83
03DC: E9 01 84
03DE: D0 F5 85
03E0: 68 86
03E1: 6A 87
03E2: 88 88
03E3: D0 E3 89
03E5: AC 78 07 90
03E8: 28 91
03E9: 60 92

03A3: STA COLCNT ; CLEAR COLUMN COUNT
03A6: LDA #$8A ; NOW DO LINE FEED
03A8: JSR DOCHAR
03AB: LDA #$53
03AD: JSR WAIT ; 200MSEC DELAY FOR LINE
03B0: FINISH: LDA COLCNT ; CHECK IF IN MARGIN
03B3: BEQ SETCH ; FOR CR, RESET CH
03B5: SBC NWDTH ; IF SO, CARRY SET.
03B7: SBC #$F7
03B9: BCC RETURN
03BA: ADC #$1F ; ADJUST CH
03BD: SETCH: STA CH ; RETURN TO CALLER ROUTINE:
03BF: RETURN: PLA
03C0: RTS
03C1: 8C 78 07 69
03C4: 08 70
03C7: A0 03 71
03C8: 18 72
03C9: 80 05 73
03CA: AD 59 CO 74
03CB: 90 03 75
03DD: AD 58 CO 76
03D3: A9 D7 77
03D4: 48 78
03D5: A9 20 80
03D6: 4A 81
03D7: 90 FD 82
03DA: 68 83
03DC: E9 01 84
03DE: D0 F5 85
03E0: 68 86
03E1: 6A 87
03E2: 88 88
03E3: D0 E3 89
03E5: AC 78 07 90
03E8: 28 91
03E9: 60 92

* HERE IS THE TELETYPE PRINT A CHARACTER ROUTINE:

03C1: 8C 78 07 69
03C4: 08 70
03C7: A0 03 71
03C8: 18 72
03C9: 80 05 73
03CA: AD 59 CO 74
03CB: 90 03 75
03DD: AD 58 CO 76
03D3: A9 D7 77
03D4: 48 78
03D5: A9 20 80
03D6: 4A 81
03D7: 90 FD 82
03DA: 68 83
03DC: E9 01 84
03DE: D0 F5 85
03E0: 68 86
03E1: 6A 87
03E2: 88 88
03E3: D0 E3 89
03E5: AC 78 07 90
03E8: 28 91
03E9: 60 92

03C1: DOCHAR: STY YSAVE ; SAVE STATUS.
03C4: PHP
03C7: LDY #$03 ; 11 BITS (1 START, 9 MARK
03C8: CLC
03C9: ;BEGIN WITH SPACE (STE
03CA: BCS MARKOUT ; SEND A SPACE
03CB: LDA SPACE
03DD: BCC TTOUT4
03D3: MARKOUT: LDA MARK ;SEND A MARK
03D4: TTOUT4: LDA #$D7 ;DELAY 9.091 MSEC FOR
03D5: PHA ; 110 BAUD
03D6: DLY1: LDA #$20
03D7: DLY2: LSR A
03DA: BCC DLY2
03DC: PLA
03DE: SBC #$01
03E0: BNE DLY1
03E1: PLA
03E2: ROR A ; NEXT BIT (STOP BITS)
03E3: DEY LOOP II BITS.
03E5: BNE TTOUT3
03E8: LDY YSAVE ;RESTORE Y-REG.
03E9: PLP ;RESTORE STATUS
03EA: RTS ;RETURN

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

FIGURE 3b
<table>
<thead>
<tr>
<th>Instruction</th>
<th>Address</th>
<th>Hex 1</th>
<th>Hex 2</th>
<th>Hex 3</th>
<th>Hex 4</th>
<th>Hex 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>CH</td>
<td>0024</td>
<td>0033</td>
<td>0039</td>
<td>0065</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COLCNT</td>
<td>07F8</td>
<td>0034</td>
<td>0038</td>
<td>0046</td>
<td>0054</td>
<td>0059</td>
</tr>
<tr>
<td>CSWL</td>
<td>0036</td>
<td>0028</td>
<td>0030</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DLY1</td>
<td>03D5</td>
<td>0085</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DLY2</td>
<td>03D8</td>
<td>0082</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOCHAR</td>
<td>03C1</td>
<td>0047</td>
<td>0056</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FINISH</td>
<td>03B0</td>
<td>0053</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MARK</td>
<td>C058</td>
<td>0077</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MARKOUT</td>
<td>03D0</td>
<td>0074</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRNTIT</td>
<td>0397</td>
<td>0045</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RETURN</td>
<td>03BF</td>
<td>0063</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTS1</td>
<td>C059</td>
<td>0044</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SETCH</td>
<td>03CD</td>
<td>0060</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPACE</td>
<td>03E0</td>
<td>0075</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TESTCTRL</td>
<td>03E8</td>
<td>0041</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TTINIT</td>
<td>03F0</td>
<td>0044</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOUT</td>
<td>03F2</td>
<td>0027</td>
<td>0029</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOUT2</td>
<td>03F4</td>
<td>0050</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOUT3</td>
<td>03F8</td>
<td>0089</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOUT4</td>
<td>03FA</td>
<td>0076</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WAIT</td>
<td>FC08</td>
<td>0058</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WNDWDTH</td>
<td>0021</td>
<td>0032</td>
<td>0061</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>YSAVE</td>
<td>0778</td>
<td>0069</td>
<td>0090</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ILE:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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

1. CONNECTOR LOCATION DIAGRAM
2. CASSETTE DATA JACKS (2 EACH)
3. GAME I/O CONNECTOR
4. KEYBOARD CONNECTOR
5. PERIPHERAL CONNECTORS (8 EACH)
6. POWER CONNECTOR
7. SPEAKER CONNECTOR
8. VIDEO OUTPUT JACK
9. AUXILIARY VIDEO OUTPUT CONNECTOR
Figure 18 Connector Location Detail
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 1.

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, \( Z_{OUT} = 100 \) Ohms. Located at K13 as illustrated 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

<table>
<thead>
<tr>
<th>TOP VIEW</th>
<th>(Front Edge of PC Board)</th>
</tr>
</thead>
<tbody>
<tr>
<td>+5V 1</td>
<td>16 N.C.</td>
</tr>
<tr>
<td>SW0 2</td>
<td>15 AN0</td>
</tr>
<tr>
<td>SW1 3</td>
<td>14 AN1</td>
</tr>
<tr>
<td>SW2 4</td>
<td>13 AN2</td>
</tr>
<tr>
<td>C040 STB 5</td>
<td>12 AN3</td>
</tr>
<tr>
<td>PDL0 6</td>
<td>11 PDL3</td>
</tr>
<tr>
<td>PDL2 7</td>
<td>10 PDL1</td>
</tr>
<tr>
<td>GND 8</td>
<td>9 N.C.</td>
</tr>
</tbody>
</table>

LOCATION J14
SIGNAL DESCRIPTIONS FOR GAME I/O

ANG-AN3: 8 addresses (C058-C05F) 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.

CQ4Q STB: A utility strobe output. Will go low during 62 of a read or write cycle to addresses CQ4Q-CQ4F. This is a 74LSxx series TTL output.

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

NC: No connection.

PDLQ-PDL3: Paddle control inputs. Requires a 0-150K ohm variable resistance and +5V for each paddle. Internal 100 ohm resistors are provided in series with external pot to prevent excess current if pot goes completely to zero ohms.

SWQ-SW2: Switch inputs. Testable by reading from addresses C061-C063 (or C069-C06B). 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

<table>
<thead>
<tr>
<th>TOP VIEW</th>
<th>(Front Edge of PC Board)</th>
</tr>
</thead>
<tbody>
<tr>
<td>+5V</td>
<td>1</td>
</tr>
<tr>
<td>STROBE</td>
<td>2</td>
</tr>
<tr>
<td>RESET</td>
<td>3</td>
</tr>
<tr>
<td>N.C.</td>
<td>4</td>
</tr>
<tr>
<td>B6</td>
<td>5</td>
</tr>
<tr>
<td>B5</td>
<td>6</td>
</tr>
<tr>
<td>B7</td>
<td>7</td>
</tr>
<tr>
<td>GND</td>
<td>8</td>
</tr>
<tr>
<td>16</td>
<td>N.C.</td>
</tr>
<tr>
<td>15</td>
<td>-12V</td>
</tr>
<tr>
<td>14</td>
<td>N.C.</td>
</tr>
<tr>
<td>13</td>
<td>B2</td>
</tr>
<tr>
<td>12</td>
<td>B1</td>
</tr>
<tr>
<td>11</td>
<td>B4</td>
</tr>
<tr>
<td>10</td>
<td>B3</td>
</tr>
<tr>
<td>9</td>
<td>N.C.</td>
</tr>
</tbody>
</table>

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 100mA.

-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 #2H25C0-111 (or equivalent) 50 pin card edge connectors with pins on .10" centers. Location and pin outs are illustrated in Figures 1 and 4.

SIGNAL DESCRIPTION FOR PERIPHERAL I/O

A0-A15: 16 bit system address bus. Addresses are set up by the 6502 within 300ns after the beginning of A1. These lines will drive up to a total of 16 standard TTL loads.

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 A2 (500ns). Each will drive 4 standard TTL loads.

D0-D7: 8 bit system data bus. During a write cycle data is set up by the 6502 less than 300ns after the beginning of D2. During a read cycle the 6502 expects data to be ready no less than 100ns before the end of D2. 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 D000 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 (500nS). This line will drive 4 standard TTL loads.

I/O STROBE: Pin 20 on all peripheral connectors will go low during Ø₂ of a read or write to any address C800-CFFF. This line will drive a total of 4 standard TTL loads.

IRQ: Interrupt request line to the 6502. 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 6502. This line has a 3K Ohm pullup to +5V and should be driven with an open collector output. It is active low.

Q₃: A 1MHz (nonsymmetrical) general purpose timing signal. Will drive up to a total of 16 standard TTL loads.

RDY: "Ready" line to the 6502. 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/W: READ/ WRITE line from 6502. 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.

$\Phi_0$: Microprocessor phase 0 clock. Will drive up to a total of 16 standard TTL loads.

$\Phi_1$: Phase 1 clock, complement of $\Phi_0$. 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 # 9-35028-1,6 pin connector. See location and pin out in Figures 1 and 5.

PIN DESCRIPTION
GND: (2 pins) system circuit ground. 0 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)

TOP VIEW
(Back Edge of PC Board)

PINOUT

GND 26  25 +5V
DMA IN 27  24 DMA OUT
INT IN 28  23 INT OUT
NMI 29  22 DMA
IRQ 30  21 RDY
RES 31  20 I/O STROBE
INH 32  19 N.C.
-12V 33  18 R/W
-5V 34  17 A15
N.C. 35  16 A14
7M 36  15 A13
Q3 37  14 A12
+1 38  13 A11
USER 1 39  12 A10
+0 40  11 A9
DEVICE SELECT 41  10 A8
D7 42  9 A7
D6 43  8 A6
D5 44  7 A5
D4 45  6 A4
D3 46  5 A3
D2 47  4 A2
D1 48  3 A1
D0 49  2 A0
+12V 50  1 I/O SELECT

(Toward Front Edge of PC Board)
LOCATIONS J2 TO J12

Figure 5  POWER CONNECTOR

PINOUT

(Toward Right Side of PC Board)

(Blue/White Wire) -12V
(Orange Wire) +5V
(Black Wire) GND

-5V (Blue Wire)
+12V (Orange/White Wire)
GND (Black Wire)

LOCATION K1
SPEAKER CONNECTOR

This is a MOLEX KK 1ØØ series connector with two .25" square pins on .1Ø" 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

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 Ø 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 .10" 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 2.0 Volts into 470 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

PINOUT

- +12V
- -5V
- VIDEO
- GND

Back Edge of PC Board

Right Edge of PC Board

LOCATION J14B

132
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 MUST 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 SELECT SOCKETS

TOP VIEW

<table>
<thead>
<tr>
<th>PINOUT</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0000-0FFF) 4K &quot;0&quot; BLOCK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>16K &quot;0&quot; BLOCK (0000-3FFF)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(0000-1FFF) 4K &quot;1&quot; BLOCK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>16K &quot;1&quot; BLOCK (0000-7FFF)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(0000-2FFF) 4K &quot;2&quot; BLOCK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>16K &quot;2&quot; BLOCK (0000-BFFF)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(0000-3FFF) 4K &quot;3&quot; BLOCK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>16K &quot;3&quot; BLOCK (0000-EFFF)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1000-4FFF) 4K &quot;4&quot; BLOCK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>16K &quot;4&quot; BLOCK (1000-7FFF)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1000-5FFF) 4K &quot;5&quot; BLOCK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>16K &quot;5&quot; BLOCK (1000-BFFF)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1000-6FFF) 4K &quot;6&quot; BLOCK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>16K &quot;6&quot; BLOCK (1000-CFFF)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1000-7FFF) 4K &quot;7&quot; BLOCK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>16K &quot;7&quot; BLOCK (1000-FFF)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

LOCATIONS D1, E1, F1
MEMORY

TABLE OF CONTENTS

1. INTRODUCTION
2. INSTALLING YOUR OWN RAM
3. MEMORY SELECT SOCKETS
4. MEMORY MAP BY 4K BLOCKS
5. DETAILED MAP OF ASSIGNED ADDRESSES

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, 20K 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 6502 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, located 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-C10 on the board), and you want them to occupy the first 16K of addresses starting at 00000, jumper pin 14 to pin 10 on all three MEMORY SELECT sockets (thereby assigning row "C" to the 00000-3FFF 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 40000 and 50000 respectively, jumper pin 13 to pin 5 (thereby assigning row "D" to the 40000-4FFF range of memory), and jumper pin 12 to pin 6 (thereby assigning row "E" to the 50000-5FFF 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 00000-5FFF.

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

<table>
<thead>
<tr>
<th>HEX</th>
<th>Address(es)</th>
<th>Used By</th>
<th>Used For</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAGE ZERO</td>
<td>0000-001F</td>
<td>UTILITY</td>
<td>register area for &quot;sweet 16&quot; 16 bit firmware processor.</td>
</tr>
<tr>
<td>0020-004D</td>
<td>MONITOR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>004E-004F</td>
<td>MONITOR</td>
<td></td>
<td>holds a 16 bit number that is randomized with each key entry.</td>
</tr>
<tr>
<td>0050-0055</td>
<td>UTILITY</td>
<td></td>
<td>integer multiply and divide work space.</td>
</tr>
<tr>
<td>0055-00FF</td>
<td>BASIC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00FF-00FF</td>
<td>UTILITY</td>
<td></td>
<td>floating point work space.</td>
</tr>
<tr>
<td>PAGE ONE</td>
<td>00100-01FF</td>
<td>6502</td>
<td>subroutine return stack.</td>
</tr>
<tr>
<td>PAGE TWO</td>
<td>0200-02FF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAGE THREE</td>
<td>03F8</td>
<td>MONITOR</td>
<td>Y (control Y) will cause a JSR to this location.</td>
</tr>
<tr>
<td>03FB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>03FE-03FF</td>
<td></td>
<td></td>
<td>NMI's are vectored to this location.</td>
</tr>
<tr>
<td>0400-07FF</td>
<td>DISPLAY</td>
<td>text or color graphics primary page.</td>
<td></td>
</tr>
<tr>
<td>0800-0BFF</td>
<td>DISPLAY</td>
<td>text or color graphics secondary page.</td>
<td></td>
</tr>
</tbody>
</table>

Memory Map Pages Ø to BFF

136
<table>
<thead>
<tr>
<th>HEX ADDRESS</th>
<th>ASSIGNED FUNCTION</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>C00X</td>
<td>Keyboard input.</td>
<td>Keyboard strobe appears in bit 7. ASCII data from keyboard appears in the 7 lower bits.</td>
</tr>
<tr>
<td>C01X</td>
<td>Clear keyboard strobe.</td>
<td></td>
</tr>
<tr>
<td>C02X</td>
<td>Toggle cassette output.</td>
<td></td>
</tr>
<tr>
<td>C03X</td>
<td>Toggle speaker output.</td>
<td></td>
</tr>
<tr>
<td>C04X</td>
<td>&quot;C040 STB&quot;</td>
<td>Output strobe to Game I/O connector.</td>
</tr>
<tr>
<td>C050</td>
<td>Set graphics mode</td>
<td></td>
</tr>
<tr>
<td>C051</td>
<td>&quot; text &quot;</td>
<td></td>
</tr>
<tr>
<td>C052</td>
<td>Set bottom 4 lines graphics</td>
<td></td>
</tr>
<tr>
<td>C053</td>
<td>&quot; &quot; &quot; &quot; text</td>
<td></td>
</tr>
<tr>
<td>C054</td>
<td>Display primary page</td>
<td></td>
</tr>
<tr>
<td>C055</td>
<td>&quot; secondary page</td>
<td></td>
</tr>
<tr>
<td>C056</td>
<td>Set high res. graphics</td>
<td></td>
</tr>
<tr>
<td>C057</td>
<td>&quot; color &quot;</td>
<td></td>
</tr>
<tr>
<td>C058</td>
<td>Clear &quot;AN0&quot;</td>
<td>Annunciator 0 output to Game I/O connector.</td>
</tr>
<tr>
<td>C059</td>
<td>Set &quot;</td>
<td></td>
</tr>
<tr>
<td>C05A</td>
<td>Clear &quot;AN1&quot;</td>
<td>Annunciator 1 output to Game I/O connector.</td>
</tr>
<tr>
<td>C05B</td>
<td>Set &quot;</td>
<td></td>
</tr>
<tr>
<td>C05C</td>
<td>Clear &quot;AN2&quot;</td>
<td>Annunciator 2 output to Game I/O connector.</td>
</tr>
<tr>
<td>C05D</td>
<td>Set &quot;</td>
<td></td>
</tr>
<tr>
<td>C05E</td>
<td>Clear &quot;AN3&quot;</td>
<td>Annunciator 3 output to Game I/O connector.</td>
</tr>
<tr>
<td>C05F</td>
<td>Set &quot;</td>
<td></td>
</tr>
<tr>
<td>HEX ADDRESS</td>
<td>ASSIGNED FUNCTION</td>
<td>COMMENTS</td>
</tr>
<tr>
<td>-------------</td>
<td>------------------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>C060/8</td>
<td>Cassette input</td>
<td>State of &quot;Cassette Data In&quot; appears in bit 7.</td>
</tr>
<tr>
<td>C061/9</td>
<td>&quot;SW1&quot;</td>
<td>State of Switch 1 (\land) Game I/O connector appears in bit 7.</td>
</tr>
<tr>
<td>C062/A</td>
<td>&quot;SW2&quot;</td>
<td>State of Switch 2 input on Game I/O connector appears in bit 7.</td>
</tr>
<tr>
<td>C063/B</td>
<td>&quot;SW3&quot;</td>
<td>State of Switch 3 input on Game I/O connector appears in bit 7.</td>
</tr>
<tr>
<td>C064/C</td>
<td>Paddle 0 timer output</td>
<td>State of timer output for Paddle 0 appears in bit 7.</td>
</tr>
<tr>
<td>C065/D</td>
<td>&quot; 1 &quot;</td>
<td>State of timer output for Paddle 1 appears in bit 7.</td>
</tr>
<tr>
<td>C066/E</td>
<td>&quot; 2 &quot;</td>
<td>State of timer output for Paddle 2 appears in bit 7.</td>
</tr>
<tr>
<td>C067/F</td>
<td>&quot; 3 &quot;</td>
<td>State of timer output for Paddle 3 appears in bit 7.</td>
</tr>
<tr>
<td>C07X</td>
<td>&quot;PDL STB&quot;</td>
<td>Triggers paddle timers during (\varphi_2).</td>
</tr>
<tr>
<td>C08X</td>
<td>DEVICE SELECT 0</td>
<td>Pin 41 on the selected Peripheral Connector goes low during (\varphi_2).</td>
</tr>
<tr>
<td>C09X</td>
<td>&quot; 1 &quot;</td>
<td></td>
</tr>
<tr>
<td>C0AX</td>
<td>&quot; 2 &quot;</td>
<td></td>
</tr>
<tr>
<td>C0BX</td>
<td>&quot; 3 &quot;</td>
<td></td>
</tr>
<tr>
<td>C0CX</td>
<td>&quot; 4 &quot;</td>
<td></td>
</tr>
<tr>
<td>C0DX</td>
<td>&quot; 5 &quot;</td>
<td></td>
</tr>
<tr>
<td>C0EX</td>
<td>&quot; 6 &quot;</td>
<td></td>
</tr>
<tr>
<td>C0FX</td>
<td>&quot; 7 &quot;</td>
<td></td>
</tr>
<tr>
<td>C10X</td>
<td>&quot; 8 &quot;</td>
<td></td>
</tr>
<tr>
<td>C11X</td>
<td>&quot; 9 &quot;</td>
<td></td>
</tr>
<tr>
<td>C12X</td>
<td>&quot; A</td>
<td></td>
</tr>
<tr>
<td>HEX ADDRESS</td>
<td>ASSIGNED FUNCTION</td>
<td>COMMENTS</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------------</td>
<td>----------</td>
</tr>
<tr>
<td>C13X</td>
<td>DEVICE SELECT</td>
<td>B</td>
</tr>
<tr>
<td>C14X</td>
<td></td>
<td>C</td>
</tr>
<tr>
<td>C15X</td>
<td></td>
<td>D</td>
</tr>
<tr>
<td>C16X</td>
<td></td>
<td>E</td>
</tr>
<tr>
<td>C17X</td>
<td></td>
<td>F</td>
</tr>
<tr>
<td>C1XX</td>
<td>I/O SELECT</td>
<td>1</td>
</tr>
<tr>
<td>C2XX</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>C3XX</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>C4XX</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>C5XX</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>C6XX</td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>C7XX</td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>C8XX</td>
<td></td>
<td>8, I/O STROBE</td>
</tr>
<tr>
<td>C9XX</td>
<td></td>
<td>9,</td>
</tr>
<tr>
<td>CAXX</td>
<td></td>
<td>A,</td>
</tr>
<tr>
<td>CBXX</td>
<td></td>
<td>B,</td>
</tr>
<tr>
<td>CCXX</td>
<td></td>
<td>C,</td>
</tr>
<tr>
<td>CDXX</td>
<td></td>
<td>D,</td>
</tr>
<tr>
<td>CEXX</td>
<td></td>
<td>E,</td>
</tr>
<tr>
<td>CFXX</td>
<td></td>
<td>F,</td>
</tr>
<tr>
<td>D000-D7FF</td>
<td>ROM socket D0</td>
<td></td>
</tr>
<tr>
<td>D800-DFFF</td>
<td></td>
<td>D8</td>
</tr>
<tr>
<td>E000-E7FF</td>
<td></td>
<td>E0</td>
</tr>
<tr>
<td>E800-EFFF</td>
<td></td>
<td>E8</td>
</tr>
<tr>
<td>F000-F7FF</td>
<td></td>
<td>F0</td>
</tr>
<tr>
<td>F800-FFFF</td>
<td></td>
<td>F8</td>
</tr>
</tbody>
</table>

NOTES:
1. Peripheral Connector 0 does not get this signal.
2. I/O SELECT 1 uses the same addresses as DEVICE SELECT 8-F.

Pin 1 on the selected Peripheral Connector goes low during $\phi_2$.

Expansion connectors.

Spare.

BASIC.

1K of BASIC, 1K of utility.

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.

φ0: Phase 0 clock to microprocessor, 1.023 MHz nominal.

φ1: Microprocessor phase 1 clock, complement of φ0, 1.023 MHz nominal.

φ2: Same as φ0. Included here because the 6502 hardware and programming manuals use the designation φ2 instead of φ0.

φ3: 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 φ1, and is stable about 300ns after the start of φ1.

DATA WRITE: During a write cycle, data from the microprocessor appears on the data bus during φ2, and is stable about 300ns after the start of φ2.

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 φ2.

SYSTEM TIMING DIAGRAM

[Diagram showing timing relationships]
FIGURE S-1  APPLE II SYSTEM DIAGRAM
FIGURE S-2 MPU AND SYSTEM BUS
FIGURE S-4   SYNC COUNTER
FIGURE S-6 4K/16K RAM SELECT
FIGURE S-7 RAM ADDRESS MUX

"D" SOURCES ARE FROM SYNC COUNT

TO RAM ADDRESS LINES

SCREEN ADDRESS FROM SYNC COUNT

*SEE FIG. S-8 FOR OTHER HALF OF C12
FIGURE S-8 4K TO 48K RAM MEMORY WITH DATA LATCH
FIGURE S-9  PERIPHERAL I/O CONNECTOR PINOUT AND CONTROL LOGIC