APPLE II
Reference Manual
January 1978

APPLE Computer Inc.
10260 Bandley Dr.
Cupertino, CA
95014
# TABLE OF CONTENTS

## A. GETTING STARTED WITH YOUR APPLE II

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-assembler 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

## 13. Additional BASIC Program Examples .......... 55

a. Rod's Color Pattern (4K) .......................... 55
b. Pong (4K) ........................................... 56
c. Color Sketch (4K) ................................... 57
d. Mastermind (8K) .................................... 59
e. Biorhythm (4K) ...................................... 61
f. Dragon Maze (4K) .................................. 63
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 Star TREK 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 similiar to the use of the shift key in that it indicates a different function to the Apple. Control key functions are not displayed on your TV screen but the Apple still gets the message. Now depress and release the "RETURN" key to tell Apple that you have finished typing a line on the keyboard. A right facing arrow (">") called a caret will now appear as the prompt character to indicate that Apple is now in its 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 Starrek

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 about 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 asterisk, "*", 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 asterisk.

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 straightforward: 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 BC (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 preceding 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 color graphics 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(Ø)) can be used to control the audio and visual displays. Ideas from this program can be incorporated into other programs with a little modification.

REQUIREMENTS

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

5 GOTO 15
10 G<C FOL (R)-20)/6; IF Q<0 THEN Q=0: IF Q=34 THEN Q=34: COLOR=0:
05 VLIN 0,0+5 AT 0: COLOR=0:
IF P=0 THEN 175: IF Q THEN
V LIN 0,0-1 AT 0: P=0: RETURN
15 DIM HF(15),BF(10)+H-1+B-13:
C+30=6;C-15: TEXT: CALL -
935: 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 * 'N* , 'B* GR: CALL -
-363: IF BK(1,14)*M THEN 40
 ; ; FOR I=0 TO 35: COLOR=0+24
(1(32): VLIN 0,39 AT 1
30 NEXT I: POKE 34,28: PRINT
: PRINT : FOR I=0 TO 15:
V TAB 21+I AND 2: TAB 1+
I1: PRINT I1: NEXT I:
NEXT I: NEXT I: NEXT I:
34,22: V TAB 24: PRINT : PRINT
"BACKGROUND":
35 GOSUB 95+B+E: PRINT " EVEN BRICK")
 : GOSUB 95:B+E: PRINT " G00 BRIC
K*": GOSUB 95:B+E: PRINT "PAVOL
": GOSUB 95:B+E: PRINT " BALL")
 : GOSUB 95
30 POKE 34,28: COLOR=0: FOR I=
0 TO 35: V LIN 0,39 AT 1: NEXT
1: FOR I=20 TO 34 STEP 2: TAB
I 1: PRINT 1/2+2: COLOR=0:
V LIN 0,39 AT 1: COLOR=C: FOR
J=1 MOD 4 TO 39 STEP 4
45 V LIN J,J+1 AT 1: NEXT J,1: TAB
5: PRINT *SCORE = 0*: PRINT
: PRINT : POKE 34,21:5#B: P=
5: L=5#X: Y=19#L=6
50 COLOR=0: PLOT X,Y;3:5;3:5;5-
AND 128#V=1#V=5# AND 5-
2L=E-1: IF L=1 THEN 120: TAB
6: IF L=1 THEN PRINT L* "BALLS L
E":
55 IF L=1 THEN PRINT "LAST BALL, "
:*A#: PRINT : FOR I=1 TO 100
: GOSUB 10: NEXT I: M=100
60 IF=I=2 6X=I+: FOR I=1 TO 6X=
PEEK (-16336): NEXT I
65 I*W=I*: IF I then 180: GOSUB
170: COLOR=R-K=J/3: IF J<39
THEN 75: IF SCRN(1,K)>R THEN
85: IF I then 180: H=F-W+1: Y=
N(5)+1# =X (K-P)+2-5=M+1
70 Z=PEEK (-16336)-PEEK (-16336)
: PEEK (-16336)-PEEK (-16336)
: PEEK (-16336)-PEEK (-16336)
: GOTO 85
75 FOR I=1 TO 6X=PEEK (-16336)
: NEXT I: K=M=9
80 V=W
85 PLOT X,Y;3:COLOR=E: PLOT I,
K+Y+1;Y+1: GOTO 60
90 PRINT "NO VIVES. RESTART":
95 INPUT " COLOR (0 TO 15),E:"
 IF E=0 OR EXIS THEN 96: RETURN
100 IF M THEN V=ABS(V): V L IN
K/242,K/242+1 AT 1=5+51/2-
9: VTAB 21: TAB 15: PRINT 5
105 Q= PEEK (-16336)-PEEK (-16336)
: PEEK (-16336)-PEEK (-16336)
: PEEK (-16336)-PEEK (-16336)
: PEEK (-16336)-PEEK (-16336)
: PEEK (-16336)-PEEK (-16336)

110 IF S<20 THEN 90
115 PRINT "CONGRATULATIONS, " ,A#
: " YOU WOKE": GOTO 165
120 PRINT " YOUR SCORE OF " ,S*/ IS "
: GOTO 125:(S<100)=5
125 PRINT " TERRIBLE": GOTO 165
130 PRINT " LOUSY": GOTO 165
135 PRINT " Poor": GOTO 165
140 PRINT " Fair": GOTO 165
145 PRINT " Good": GOTO 165
150 PRINT " Very Good": GOTO 165
155 PRINT " Excellent": GOTO 165
160 PRINT " NEARLY PERFECT":
165 PRINT " ANOTHER GAME " ,A#: (Y/H)
: " INPUT A#: IF A>(1,1)=Y
: THEN 25: TEXT: CALL -363:
: V TAB 10: TAB 10: PRINT " GAME OV
ER": END
170 G=(C POL (R)-20)/6: IF Q<0 THEN
0=0: IF Q=34 THEN Q=34: COLOR=0:
: V LIN 0,0+5 AT 0: COLOR=R:
: IF P=0 THEN 175: IF Q THEN
V LIN 0,0-1 AT 0: P=0: RETURN
175 IF P=0 THEN RETURN : IF Q<34
: THEN V LIN 0,0+5 AT 0: P=0:
RETURN
180 FOR I=1 TO 6X=6: PEEK (-16336)
: NEXT I: GOTO 59

12

"DTCA2DOC-469-012.PICT" 311 KB 2001-06-26 dpi: 600h x 600v pix: 2613h x 4055v

Source: David T Craig
COLOR DEMO PROGRAM LISTING

10 DIM (C4); POKE 2,173: POKE 3,46: POKE 4,192: POKE 5,165:
7: POKE 6,8: POKE 7,32: POKE 8,168: POKE 9,252: POKE 10,
165: POKE 11,1: POKE 12,268

20 POKE 13,4: POKE 14,198: POKE 15,24: POKE 16,240: POKE 17,5: POKE 18,198: POKE 19,1:
POKE 20,76: POKE 21,2: POKE 22,8: POKE 23,96

30 TEXT : CALL -93: VТАB 4: TAB 8: PRINT "4K COLOR DEMO": PRINT 5: PRINT "LINES": PRINT "_CROS
5: PRINT "3 WEAVING": 40 PRINT "4 TUNNEL": PRINT "5 CIRC
E": PRINT "SPiral "": PRINT 7 TONES "": PRINT "_SPRING"

50 PRINT "9 HYPERBOLA": PRINT 10 PRINT "COLOR BARS": PRINT : PRINT **NEEDS POLL (0) CONNECTED**
60 PRINT "HIT ANY KEY FOR NEW DEMO": Z=0: PRINT : INPUT "WHICH DEMO"
B, J: GR : IF 1=1 AND 1=1 THEN GOTO 100
70 INPUT "WHICH DEMO WOULD YOU LIKE"
*, I: GR : IF I AND 1=0 THEN GOTO 100: GOTO 38

100 I=1+I MOD 79: J=I+1)39)X79

100: GOSUB 2000: GOSUB 10000 : GOTO 100
200 I=1+I MOD 39: J=1: GOSUB 2000:

300 J=I+1: I=I MOD 22+1: FOR I=1
TO 1295: COLOR=1 MOD 37: (3+J) MOD 35: NEXT
1: GOSUB 10000: GOTO 300
400 FOR I=1 TO 4:C(I)=RND (16)
7: NEXT I

410 FOR I=5 TO 1 STEP -1: (I+1)
=C(I): NEXT I: (I)=RND (16)
5: FOR I=1 TO 5: FOR J=1 TO 4
4
420 COLOR=C(J)+1=J+1+I+I: X=39-
L: HLIN K, L AT R: VLIN L, K AT L:
1: HLIN K, L AT L: VLIN L, K AT X:
K: NEXT J, I: GOSUB 10000: GOTO 410

500 Z=29: GOTO 900

600 COLOR=RND (16): FOR I=8 TO
18 STEP 2: J=9+I: HLIN I, J AT 1:
1: GOSUB 640: VLIN L, J AT J:
GOSUB 640

610 VLIN I, J AT J: GOSUB 640:
VLIN I+2, J AT I+2: GOSUB 640
700 J=I+1 MOD 15: FOR Y=0 TO 39
800 K=I+1 MOD 32: COLOR=RND (16)
900 X=J+1 MOD 3: VТАB 24(I+1) MOD 2
1000 PRINT J: IF I=0 THEN
GOTO 610: GOSUB 10000: GOTO 900

1000 IF PEEK (-16384)X128 THEN RETURN
: POKE -16368,0: POP : GOTO 30
APPLE II STARTREK VERSION

This is a short description of how to play StarTrek on the Apple computer.

The universe is made up of 44 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 (% are moves to that quadrant, and destroy.

Numbers displayed for each quadrant denote:
1. # of stars in the ones place
2. # of bases in the tens place
3. # 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 'O' (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 referring to them followed by a "return".
   A. If response is 1 the computer will ask warp or ion and expects 'W' if one wants to travel in the galaxy between quadrants and an 'I' if one wants only internal quadrant travel.
   Duration of warp factor is the number of spaces or quadrants the enterprise will move.
   Course is compass reading in degrees for the desired 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, 7, 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 +5 or -2.

I. 7 enters a computer which will respond to the following instructions:
   1. Compute course
   2. Lock phasers
   3. Lock photon torpedoes
   4. Lock course
   5. Compute 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=2'

Course or trajectory:
0

270-----------------------90

180

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

14
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 $C00 to $FFF. 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 B0 (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

**AUTO num**

Sets automatic line numbering mode. Starts at line number num and increments line numbers by 10. To exit AUTO mode, type a control X*, then type the letters "MAN" and press the return key.

**AUTO num1, num2**

Same as above except increments line numbers by number num2.

**CLR**

Clears current BASIC variables; undimensions arrays. Program is unchanged.

**CON**

Continues program execution after a stop from a control C*. Does not change variables.

**DEL num1**

Deletes line number num1.

**DEL num1, num2**

Deletes program from line number num1 through line number num2.

**DSP var**

Sets debug mode that will display variable var every-time that it is changed along with the line number that caused the change. (NOTE: RUN command clears DSP mode so that DSP command is effective only if program is continued by a CON or GOTO command.)

**HIMEM: expr**

Sets highest memory location for use by BASIC at location specified by expression expr in decimal. HIMEM: may not be increased without destroying program. HIMEM: is automatically set at maximum RAM memory when BASIC is entered by a control B*.

**GOTO expr**

Causes immediate jump to line number specified by expression expr.

**GR**

Sets mixed color graphics display mode. Clears screen to black. Resets scrolling window. Displays 40x40 squares in 15 colors on top of screen and 4 lines of text at bottom.

**LIST**

Lists entire program on screen.

**LIST num1**

Lists program line number num1.

**LIST num1, num2**

Lists program line number num1 through line number num2.
LOAD expr.
Reads (Loads) a BASIC program from cassette tape.
Start tape recorder before hitting return key. Two
beeps and a ">" indicate a good load. "ERR" or "MEM"
FULL ERR" message indicates a bad tape or poor recorder
performance.

LOMEM: expr
Similar to HIMEM: except sets lowest memory location
available to BASIC. Automatically set at 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>Prefix Operators</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 2Ø0</td>
<td>Logical Negation of following expression; Ø if expression is true (non-zero), 1 if expression is false (zero).</td>
</tr>
</tbody>
</table>

### Arithmetic Operators

<table>
<thead>
<tr>
<th>Operation</th>
<th>Sample Statement</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<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 $N \times 2$.</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>155 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</td>
<td>Expression &quot;equals&quot; expression.</td>
</tr>
<tr>
<td></td>
<td>THEN 500</td>
<td></td>
</tr>
<tr>
<td>=</td>
<td>170 IF A$(1,1)=&quot;Y&quot; THEN 500</td>
<td>String variable &quot;equals&quot; string variable.</td>
</tr>
<tr>
<td># or &lt;&gt;</td>
<td>180 IF ALPHA #*Y THEN 500</td>
<td>Expression &quot;does not equal&quot; expression.</td>
</tr>
<tr>
<td>#</td>
<td>190 IF A$ # &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 500</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>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABS(expr)</td>
<td>300 PRINT ABS(X) Gives absolute value of the expression expr.</td>
</tr>
<tr>
<td>ASC(str$)</td>
<td>310' PRINT ASC(&quot;BACK&quot;) Gives decimal ASCII value of designated string variable str$. If more than one character is in designated string or sub-string, it gives decimal ASCII value of first character.</td>
</tr>
<tr>
<td>LEN(str$)</td>
<td>340 PRINT LEN(B$) Gives current length of designated string variable str$; i.e., number of characters.</td>
</tr>
<tr>
<td>PDL(expr)</td>
<td>350 PRINT PDL(X) 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>PEEK(expr)</td>
<td>360 PRINT PEEK(X) Gives the decimal value of number stored of decimal memory location specified by expression expr. For MEMORY locations above 32676, use negative number; i.e., HEX location FFF0 is -16</td>
</tr>
<tr>
<td>RND(expr)</td>
<td>370 PRINT RND(X) 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>SCRN(expr1, expr2)</td>
<td>380 PRINT SCRN (X1,Y1) 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 colorgraphics display mode as set by GR command or B to 47 if in all color mode set by POKE -16384,B: POKE -16382,O.</td>
</tr>
<tr>
<td>SGN(expr)</td>
<td>390 PRINT SGN(X) Gives sign (not sine) of expression expr i.e., -1 if expression expr is negative, zero if zero and +1 if expr is positive.</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 alpha-numeric characters up to 10. Variable names may not contain burried 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 var1 (expr1)

50 DIM A(10),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)

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 occured.

Source: David T Craig
NAME | EXAMPLE | DESCRIPTION
--- | --- | ---
END | 11Ø END | Stops program execution. Sends carriage return and "> " BASIC prompt) to screen.
FOR var= | 11Ø FOR L=Ø to 39 | Begins FOR...NEXT loop, initializes variable var to value of expression expr1
expr2 TO expr3 | 12Ø FOR X=Y1 TO Y3 | then increments it by amount in expression expr3 each time the corresponding "NEXT" statement is encountered, until value of expression expr2 is reached. If STEP expr3 is omitted, a STEP of +1 is assumed. Negative numbers are allowed.
STEP expr3 | 13Ø FOR I=39 TO 1 | 15Ø GOSUB 1ØØ *J2
GOSUB expr | 14Ø GOSUB 5ØØ | Causes branch to BASIC subroutine starting at legal line number specified by expression expr. Subroutines may be nested up to 16 levels.
GOTO expr | 16Ø GOTO 2ØØ | Causes immediate jump to legal line number specified by expression expr.
 | 17Ø GOTO ALPHA+1ØØ
GR | 18Ø GR | Sets mixed standard resolution color graphics mode. Initializes COLOR = Ø (Black) for top 4Ø×4Ø of screen and sets scrolling window to lines 21 through 24 by 4Ø characters for four lines of text at bottom of screen. Example 19Ø sets all color mode (4Ø×48 field) with no text at bottom of screen.
HLIN expr1, | 20Ø HLIN Ø,39 AT 2Ø | 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 Ø to 39 and expr1 < = expr2. expr3 must be in the range of Ø to 39 (or Ø to 47 if not in mixed mode).
expr2AT expr3 | 21Ø HLIN Z,Z+6 AT I

Note: HLIN Ø, 19 AT Ø is a horizontal line at the top of the screen extending from left corner to center of screen and HLIN 20,39 AT 39 is a horizontal line at the bottom of the screen extending from center to right corner.
**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

**LEGAL**

260 IF $L > 5$ THEN 50:
    ELSE 60

270 IF $L > 5$ THEN 50
    GO TO 60

**INPUT var1, var2, str$**

280 INPUT $X,Y,Z(3)$

290 INPUT "AMT",
    DLLR

300 INPUT "Y or N?", A$

**IN# expr**

310 IN# 6

320 IN# Y+2

330 IN# 0

**LET**

340 LET $X = 5$

**LIST num1, num2**

350 IF $X > 6$ THEN
    LIST 50

**NEXT var1, var2**

360 NEXT I

370 NEXT J,K

**NO DSP var**

380 NO DSP I

**NO TRACE**

390 NO TRACE

If expression is true (non-zero) then execute statement; if false do not execute statement. If statement is an expression, then a GOTO expr type of statement is assumed to be implied. The "ELSE" in example 260 is illegal but may be implemented as shown in example 270.

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

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.

Assignment operator. "LET" is optional

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

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

Turns-off DSP debug mode for variable

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>
</table>
| **TAB expr** | 530 TAB 24  
540 TAB I+24  
550 IF A#B THEN  
TAB 20 | Moves cursor to absolute horizontal position specified by expression expr in the range 1 to 40. Position is left to right. |
| **TEXT** | 550 TEXT  
560 TEXT: CALL-936 | 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. |
| **TRACE** | 570 TRACE  
580 IFN > 32000  
THEN TRACE | Sets debug mode that displays each line number as it is executed. |
| **VLIN expr1, expr2** | 590 VLIN 0, 39AT15  
600 VLIN Z,Z+6ATY | Similar to HLIN except draws vertical line starting at expr1 and ending at expr2 at horizontal position expr3. |
| **VTAB expr** | 610 VTAB 18  
620 VTAB Z+2 | Similar to TAB. Moves cursor to absolute vertical position specified by expression expr in the range 1 to 24. VTAB 1 is top line on screen; VTAB24 is bottom. |
SPECIAL CONTROL AND EDITING CHARACTERS

"Control" characters are indicated by a super-scripted "C" such as G^C. They are obtained by holding down the CTRL key while typing the 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_E 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>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 functions without using control button.</td>
</tr>
<tr>
<td>Control J</td>
<td>Issues line feed only</td>
</tr>
<tr>
<td>Control V</td>
<td>Compliment to H^C. Forward spaces cursor and copies over written 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&lt;sub&gt;E&lt;/sub&gt;</td>
<td>Move cursor to right</td>
</tr>
<tr>
<td>B&lt;sub&gt;E&lt;/sub&gt;</td>
<td>Move cursor to left</td>
</tr>
<tr>
<td>C&lt;sub&gt;E&lt;/sub&gt;</td>
<td>Move cursor down</td>
</tr>
<tr>
<td>D&lt;sub&gt;E&lt;/sub&gt;</td>
<td>Move cursor up</td>
</tr>
<tr>
<td>E&lt;sub&gt;E&lt;/sub&gt;</td>
<td>Clear text from cursor to end of line</td>
</tr>
<tr>
<td>F&lt;sub&gt;E&lt;/sub&gt;</td>
<td>Clear text from cursor to end of page</td>
</tr>
<tr>
<td>@&lt;sub&gt;E&lt;/sub&gt;</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.

<table>
<thead>
<tr>
<th>Color</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 = Black</td>
<td>8 = Brown</td>
</tr>
<tr>
<td>1 = Magenta</td>
<td>9 = Orange</td>
</tr>
<tr>
<td>2 = Dark Blue</td>
<td>10 = Grey</td>
</tr>
<tr>
<td>3 = Light Purple</td>
<td>11 = Pink</td>
</tr>
<tr>
<td>4 = Dark Green</td>
<td>12 = Green</td>
</tr>
<tr>
<td>5 = Grey</td>
<td>13 = Yellow</td>
</tr>
<tr>
<td>6 = Medium Blue</td>
<td>14 = Blue/Green</td>
</tr>
<tr>
<td>7 = Light Blue</td>
<td>15 = White</td>
</tr>
</tbody>
</table>

29
## Special Controls and Features

<table>
<thead>
<tr>
<th>Hex</th>
<th>BASIC Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Display Mode Controls</td>
</tr>
<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>
<tr>
<td></td>
<td></td>
<td>TEXT Mode Controls</td>
</tr>
<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;1&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>(0E) Home cursor, clear screen</td>
</tr>
<tr>
<td>FC42</td>
<td>220 CALL -958</td>
<td>(FE) Clear from cursor to end of page</td>
</tr>
</tbody>
</table>

Source: David T Craig
<table>
<thead>
<tr>
<th>Hex</th>
<th>BASIC Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FC9C</td>
<td>23Ø CALL -868</td>
<td>(E₉) Clear from cursor to end of line</td>
</tr>
<tr>
<td>FC66</td>
<td>24Ø CALL -922</td>
<td>(J₉) Line feed</td>
</tr>
<tr>
<td>FC70</td>
<td>25Ø CALL -912</td>
<td>Scroll up text one line</td>
</tr>
</tbody>
</table>

**Miscellaneous**

| C030 | 36Ø X=PEEK(-16336) 365 POKE -16336,Ø | Toggle speaker |
| C000 | 37Ø X=PEEK(-16384) | Read keyboard; if X>127 then key was pressed. |
| C010 | 38Ø POKE -16368,Ø | Clear keyboard strobe - always after reading keyboard. |
| C061 | 39Ø X=PEEK(16287) | Read PDL(Ø) push button switch. If X>127 then switch is "on". |
| C062 | 40Ø X=PEEK(-16286) | Read PDL(1) push button switch. |
| C063 | 41Ø X=PEEK(-16285) | Read PDL(2) push button switch. |
| C05B | 42Ø POKE -16296,Ø | Clear Game I/O ANØ output |
| C059 | 43Ø POKE -16295,Ø | Set Game I/O ANØ output |
| C05A | 44Ø POKE -16294,Ø | Clear Game I/O AN1 output |
| C05B | 45Ø POKE -16293,Ø | Set Game I/O AN1 output |
| C05C | 46Ø POKE -16292,Ø | Clear Game I/O AN2 output |
| C05D | 47Ø POKE -16291,Ø | Set Game I/O AN2 output |
| C05E | 48Ø POKE -16290,Ø | Clear Game I/O AN3 output |
| C05F | 49Ø POKE -16289,Ø | Set Game I/O AN3 output |

Source: David T Craig
APPLE II BASIC ERROR MESSAGES

*** SYNTAX ERR  Results from a syntactic or typing error.

*** > 32767 ERR  A value entered or calculated was less than
                 -32767 or greater than 32767.

*** > 255 ERR    A value restricted to the range 0 to 255 was
                 outside that range.

*** BAD BRANCH ERR  Results from an attempt to branch to a non-
                    existant line number.

*** BAD RETURN ERR  Results from an attempt to execute more RETURNs
                    than previously executed GOSUBs.

*** BAD NEXT ERR    Results from an attempt to execute a NEXT state-
                    ment for which there was not a corresponding
                    FOR statement.

*** 16 GOSUBS ERR  Results from more than 16 nested GOSUBs.

*** 16 FORS ERR    Results from more than 16 nested FOR loops.

*** NO END ERR     The last statement executed was not an END.

*** MEM FULL ERR   The memory needed for the program has exceeded
                    the memory size allotted.

*** TOO LONG ERR   Results from more than 12 nested parentheses or
                    more than 128 characters in input line.

*** DIM ERR        Results from an attempt to DIMension a string
                    array which has been previously dimensioned.

*** RANGE ERR      An array was larger than the DIMensioned
                    value or smaller than 1 or HLIN, VLIN,
                    PLOT, TAB, or VTAB arguments are out of
                    range.

*** STR OVFL ERR   The number of characters assigned to a string
                    exceeded the DIMensioned value for that string.

*** STRING ERR     Results from an attempt to execute an illegal
                    string operation.

RETYPE LINE        Results from illegal data being typed in response
                    to an INPUT statement. This message also requests
                    that the illegal item be retyped.
Simplified Memory Map

FFFF --------- 64K  Monitor and BASIC Routines in ROM

E0000 -------- 56K  Future enhancement or user supplied PROMS

D0000 -------- 52K  Peripheral I/O

C0000 -------- 48K

XX ----------- XX ---- User specified RAM memory size
(HIMEM:)

User Workspace

7FF ----------- 2K  Screen Memory

400 --------- 1K

0 ----------- g  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 01 when DSP set in BASIC initiates a process that displays this variable with the line number every time it is changed within a program.

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

DATA - hexadecimal equivalent of numeric information, represented in pairs of bytes, low order byte first.
String variables are formatted a bit differently than numeric ones. These variables have one extra attribute - a string terminator which designates the end of a string. A string variable is formatted as follows:

```
VN  DSP  NVA  DATA(0)  DATA(1)....  DATA(n)  ST
```

- **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 B 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 ($CI$ 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 CI 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

36
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 straightforward process with the Read/Save subroutine listed in figure 4. This routine will increase the flexibility of the Apple II by providing a permanent record of the data generated within a program. This program can be reprocessed. The Read/Save routines are a valuable addition to any data processing program.
Figure 1

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

Figure 2

$800.80C rewritten with labelling
READ/SAVE PROGRAM

Ø  A=Ø

10  GOTO 100

20  PRINT "REWIND TAPE THEN
    START TAPE RECORDER";
    INPUT "THEN HIT RETURN",
    B$

22  A=CM-LM: POKE 60,4:
    POKE 61,8: POKE 62,5:
    POKE 63,8: CALL -307

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

26  PRINT "DATA TABLE SAVED";
    RETURN

30  PRINT "REWIND THE TAPE
    THEN START TAPE RECORDER";
    INPUT "AND HIT RETURN",
    B$

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

34  IF A<Ø THEN 38: P=LM+A:
    IF P>HM THEN 38: CM=P:
    POKE 60, LM MOD 256:
    POKE 61, LM/256: POKE 62,
    CM MOD 256: POKE 63, CM/256:
    CALL -259

36  PRINT "DATA READ IN";
    RETURN

38  PRINT "***TOO MUCH DATA
    BASE***": RETURN

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

Comments:

This must be the first statement in the program. It is initially Ø, but if data is to be saved, it will equal the length of the data base.

This statement moves command to the main program.

Lines 20-26 are the write data to tape subroutine.

Writing data table to tape

Returning control to main program.

Lines 30-38 are the READ data from tape subroutine.

Checking the record length (A) for memory requirements if everything is satisfactory the data is READ in.

Returning control to main program.
1 >A=1
   >
2 >B=Ø
   >
3 >PRINT PEEK (204) + PEEK (205) * 256
   computer responds with=2060
4 >
   *
5 *800.BOC

Computer responds with:
Ø800- C1 ØØ 86 Ø8 FF FF C2 ØØ
Ø808 ØC Ø8 ØØ ØØ ØØ

Example 1
Example 2

```
;LIST
0 R=0
10 GOTO 100
20 REM WRITE DATA TO TAPE ROUTINE
   POKE 64,LM MOD 256: POKE 61 ,LM/256: POKE 62,CN MOD 256
   ; POKE 63,CN/256: CALL -307
26 RETURN
30 REM READ DATA SUBROUTINE
   POKE 60,4: POKE 61,8: POKE 62,5: POKE 63,9: CALL -259
34 IF A<>0 THEN 38:P=LM+1: IF P>
   HM THEN 38:CN=P: POKE 60,LM MOD 256: POKE 61,LM/256: POKE 62
   ,CN MOD 256: POKE 63,CN/256: CALL -259
36 RETURN
38 PRINT "*** TOO MUCH DATA BASE ***
   *": END
100 DIM A(1),X(25)
185 FOR I=1 TO 25: X(I)=I: NEXT I
188 LM=2948: CN=2106: R=50: HM=16383
110 PRINT "20 NUMBERS GENERATED"
120 PRINT "NOW WE ARE GOING TO SAVE
   THE DATA": PRINT "WHEN YOU ARE R
   EADY START THE RECORDER IN RECOR
   D MODE": INPUT "AND HIT RETURN"
   ,A#
130 CALL -336: PRINT "NOW WRITING DA
   TA TO TAPE": GOSUB 20
135 PRINT "NOW THE DATA IS SAVED"
140 PRINT "NOW WE ARE GOING TO CLEAR
   THE X(20) TABLE AND READ THE DA
   TA FROM TAPE"
150 FOR I=1 TO 25:X(I)=0: PRINT
   ";(I;I)=;X(I): NEXT I
160 PRINT "NOW START TAPE RECORDER"
   ;: INPUT "AND THEN HIT RETURN"
   ,A#
165 PRINT "A ",,A
170 GOSUB 30
180 PRINT "ALL THE DATA READ IN"
190 FOR I=1 TO 25: PRINT "X(I)+;I;
   *=;X(I); NEXT I
195 PRINT "THIS IS THE END"
200 END
```
A SIMPLE TONE SUBROUTINE

INTRODUCTION

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

Tone Generation

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

Toggling the speaker is a simple process, a mere PEEK - 16336 ($C030$) 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.
FIGURE 1. Machine Language Program
adapted from a program by P. Lutas.

32000 POKE 2,173: POKE 3,48: POKE
4,192: POKE 5,136: POKE 6,208
: POKE 7,4: POKE 8,195: POKE
9,1: POKE 10,240
32005 POKE 11,8: POKE 12,286: POKE
13,286: POKE 14,246: POKE 15
,166: POKE 16,8: POKE 17,76
: POKE 18,2: POKE 19,8: POKE
20,96: RETURN

FIGURE 2. BASIC "POKES"

25 POKE 8,P: POKE 1,D: CALL 2:
RETURN

FIGURE 3. GOSUB
High-Resolution Operating Subroutines

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

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

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

In the following subroutine descriptions, all addresses given will be in decimal. The hexadecimal substitutes will be preceded by a dollar sign ($). All entry points given are for the cassette tape subroutines, which load into addresses 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 -115991)
   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 $D95)

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 3895 (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
High-Resolution Operating Subroutines

SHAPE (continued)

1 will cause the shape to be drawn true size, while a scaling factor of 2 will draw the shape double size, etc. The scaling factor is passed in location 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° 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:

```
<table>
<thead>
<tr>
<th>7 6 5 4 3 2 1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>D D P D D P O O D</td>
</tr>
</tbody>
</table>
```

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
High-Resolution Operating Subroutines

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° 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

51
High-Resolution Operating Subroutines

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:

```
Φ
1
2
3
4
5
6
7
8
9
```

Figure 1.

```
C  B  A
0 1 0
1 1 0
1 0 1
0 0 0
```

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

Hexadecimal Codes

```
0 0 0 0 → 0
0 0 0 1 → 1
0 0 1 0 → 2
0 0 1 1 → 3
0 1 0 0 → 4
0 1 0 1 → 5
0 1 1 0 → 6
0 1 1 1 → 7
1 0 0 0 → 8
1 0 0 1 → 9
1 0 1 0 → A
1 0 1 1 → B
1 1 0 0 → C
1 1 0 1 → D
1 1 1 0 → E
1 1 1 1 → F
```

Figure 2.
REN HIERES DEMO-BASIC LISTING

LIST

1 INIT=3672: CLEAR=3866: P0=3761:
   :PLTS=3785:LINES=3783:SHAPE=
   3865: FND=3664:5INTBL=3840
5 DIM X(18),Y(18)

10 TEXT : CALL -396 : VTab 4 : TAB
10 : PRINT "*** 16K APPLE II ***"
   :PRINT " *** HIGH RESOLUTION G
   GRAPHICS DEMOS *** : PRINT"
15 PRINT "* 1 RANDOM LINE DRAW AT BAS
IC SPEED* : PRINT "2 RANDOM SHAPE
PROJECTED INTO CORNER"
28 PRINT "3 CHRS' MAG FULLY"
   :PRINT 4 RANDOM SHAPE SPIRALING
   INTO POINT* : PRINT "5 SPIROGRAP
H"
25 PRINT " 6 HI-RES DONUT* : PRINT
" 7 RANDOM WAVE FORM* : PRINT
" 8 SUM OF TWO SIN WAVE"
30 PRINT : PRINT "HIT ANY KEY FOR NW
EW DEMO* : PRINT "TYPE 'CONTROL C'
" ; RETURN BUTTON THEN TYPE 'T
EXIT AND RETURN BUTTON TO STOP"

50 PRINT : INPUT "WHICH DEMO # DO Y
OU WANT?",X1
90 IF X1=1 OR X1=8 THEN 10: CALL
   INIT: GOTO 1000+X1
100 CALL INIT: X=48:Y=X : GOSUB 2000
   :POKE 812,255: CALL PLOT
110 X= RND (2000):Y= RND (1600): GOSUB
   2000: CALL LINE: IF NOT RND
   3000 THEN POKE 23,X: PEEK (23)+
   RND (3)+11 MOD 4+85 : GOSUB 3000:
   GOTO 110
200 GOSUB 1000: RND (2)+279:Y= RND
   (2)+159: CALL PLOT: FOR
   J=1 TO 30: FOR I=1 TO R: POKE
   800,X(I) MOD 256: POKE 801,
   X(I)+255: POKE 802,Y(I): CALL
   LINE
530 IF RND (5000):X THEN POKE 28,
   X(I)+160: Y(I)+160: IF
   Y(I)-Y(I)+160 THEN 510: Y(I)+=
   Y(I)+160: IF Y(I) THEN Y(I)+
   310: GOSUB 3000: GOTO 510
600 POKE 16382,0: POKE 768,5: POKE
   769,0: POKE 800,140: POKE 801,
   .0: POKE 802,0: POKE 804,0:
   POKE 805,3: POKE 812,255: CALL
   PLOT
610 FOR R=0 TO 4160: POKE 887,R MOD
   64: POKE 886,2+6: NOT (R MOD
   65): CALL SHAPE: NEXT R: GOSUB
   3000: GOTO 610
700 J= RND (16)+ RND (16) : X=RND
   (3)+1 : RND (5)+L= RND (9)+R: PRINT "FREQ#1 ='
   :F;FREQUENCY=1
   710 GOSUB 4000: GOSUB 3000: GOTO
   700
800 INPUT "REL FREQ #1=",J: INPUT
   "REL FREQ @2=",K: INPUT "MODE (0
   =SOLID, 1=POINTS)" =L
810 GOSUB 4000: GOSUB 3000: GOTO
   800
1000 CALL CLEAR: POKE 812, RND (3)+
   85+85:R= RND (3)+2+ RND
   (2): FOR I=1 TO R: X(I)= RND
   (1600):Y(I)= RND (160): NEXT
   I
   1010 X=X(I):Y=Y(I): GOSUB 2000: RETURN
   2000 POKE 800,X MOD 256: POKE 801,
   ,X,255: POKE 802,Y,255:
2000: CALL CLEAR: POKE 812, RND (3)+
   85+85:R= RND (3)+2+ RND
   (2): FOR I=1 TO R: X(I)= RND
   (1600):Y(I)= RND (160): NEXT
   I
   3000 IF PEAK (16384) THEN 1200
   THEN RETURN
   3000: POKE -15368,8: POP : GOTO
   10
4000 CALL INIT : POKE 812,55:A=0
   :B=0: FOR I=1 TO 279: A=A+:J
   MOD 256:B=(B+A) MOD 256:Y=
   (PEEK (SINTBL+I)+PEEK (SINTBL+
   B))%16
4010 POKE 800,1 MOD 256: POKE 801,
   ,1255: POKE 802,J: CALL LINE:
   64< NOT I OR L>: NEXT I: RETURN
210 X(I)=X(I)-Y(I)+180:Y(I)=Y(I)+180:
   NEXT I,J: GOSUB 3000: GOTO 200
300 CALL INIT : X=RND (24)+10+20
   :Y=RND (4)+10+20: POKE 812,
   RND (3)+85+85: GOSUB 2000
   :CALL PLOT
310 IF RND (1000):X THEN 300: IF
   NOT RND (286) THEN POKE 28,
   RND (4)+05
320 X=X+: RND (3)-1)*25:Y=Y+
   (RND (3)-1)*15: IF X<0 OR
   X>127 OR Y<0 OR Y>159 THEN
   320
330 X=X+1:Y=Y+1: GOSUB 2000: CALL
   LINE : GOSUB 3000: GOTO 310
400 GOSUB 1000: POKE 812, RND (3)+
   85+85: CALL PLOT
410 FOR J=1 TO 23: FOR I=1 TO R:
   POKE 800,X(I) MOD 255: POKE
   801,X,255: POKE 802,Y,255:
   CALL LINE: CALL LINE
   420 Y=(X(I)-80)*(Y(I)-80): MOD 10
   +80:Y(I)=(Y(I)-50-(X(I)-50)
   )/80/9000:XI=X(I): NEXT 1,
   J: GOSUB 3000: GOTO 400
500 CALL INIT: POKE 800,X: CALL
   PLOT:XI=0:Y=0:XI1=0:Y1=1:
   A=5:9:3:0:8
510 POKE 800,A: POKE 801,A: POKE
   802,A: CALL LINE: POKE 800,
   (279-X) MOD 256: POKE 801,X
   24: POKE 802,159: CALL LINE:
   POKE 800,23: POKE 801,1: POKE
   802,159: CALL LINE
515 IF RND (500) THEN 520:A=1+1 RND
   (13)+B+1: RND (3)+C+4+ RND
   (7)
520 POKE 800,X MOD 256: POKE 801,
   ,X,255: POKE 802,A: CALL LINE:
   X=A+10+10: IF X<0 AND X>250
   THEN 530:X=+10: X1=X: IF
   X<0 THEN X=X+550
54

Source: David T Craig
ROD'S COLOR PATTERN

PROGRAM DESCRIPTION

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

REQUIREMENTS

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

PROGRAM LISTING

100 GR
105 FOR W=3 TO 50
110 FOR I=1 TO 19
115 FOR J=0 TO 19
120 K=I+J
130 COLOR=J*2-(I*3)+W/12
135 PLOT I,K; PLOT K,I; PLOT 40
   -I,40-K
136 PLOT 40-K,40-I; PLOT K,40-I;
   PLOT 40-I,K; PLOT I,40-K; PLOT
   40+K,J
140 NEXT J,I
145 NEXT W; GOTO 105
PROGRAM LISTING: PONG

5 REM PONG BY MENDALL BITTER
10 REM 7/7/77
15 REM PADELLE SWITCHES CONTROL
    PADELLE SIZE AFTER A MISS
    OR DURING A HIT
20 GR
25 DIM P(3): DIM HP4(10)
30 R=38686:G=1:G=1
35 COLOR=13: HLIN 1,38 AT 0: HLIN 1,38 AT 39
40 CALL -936: VTAB 23: INPUT "HAND"
    ALL OR PONG? * ,HP4*
45 INPUT "PADELLE SIZE (1-6) ",
    PS: IF PS<1 OR PS>6 THEN 45
        S=5:PS=1
50 CALL -936
55 IF HP4(1)>HP4(3) THEN 205
60 H=1: COLOR=13: HLIN 8,0,39 AT
    39: GOTO 285
65 FOR Y=0 TO B STEP C
70 Y=4+Y: IF Y>1 AND Y<38 THEN
    60: IF Y<1 THEN Y=1: IF Y>38
    THEN Y=38
75 Y=Y: FOR T=1 TO 5:x=PEEK
    (-1636): NEXT T
80 IF X=C OR X=9+C THEN 85: COLOR=0:
    PLOT X,C,Y: COLOR=15: PLOT
7,Y
85 TY=T: IF X MOD 2=0 THEN GOSUB
    235: NEXT X
90 GOSUB 235
95 IF SCREEN(x,Y+Y*C+40 AND Y+
    V)-1)=0 THEN 165
100 FOR T=1 TO 10:M=PEEK (-1636)
    ) NEXT T
105 IF H AND C#0 THEN 136
110 PP=P(X-30)
115 IF Y=PP THEN V=3: IF Y>PP+1
    THEN V=2: IF Y=PP+2 THEN V=1
120 IF Y=PP+3 THEN V=-1: IF Y=PP+
    4 THEN V=2: IF Y=PP+5 THEN
    V=3
125 IF S=0 THEN V=3- RND (7)
130 COLOR=0: PLOT X,C,Y
135 IF H AND C#0 OR (VYO= ABS
    (V) AND X=0) THEN V=-4- RND
    (9)
140 IF X=0 THEN VYO= ABS (V)
145 X=39-RX=39-RX=0: C
150 IF PEEK (-16286)<=127 AND S#5
    THEN S=S+1
155 IF PEEK (-16287)<=127 AND S#5
    THEN S=S-1
160 GOTO 65
165 COLOR=8: PLOT X,C,Y
170 COLOR=15: PLOT X,Y+4*(Y+3)/
    1 AND Y+4(48)
175 FOR T=1 TO 75:M=PEEK (-1636)
    + PEER (-1636): PEER (-1636)
    ): NEXT T
180 IF X=0 THEN SR=SR+1: IF X=39
    THEN SL=SL+1
185 VTAB 23: TAB 7; PRINT SL: TAB
33: PRINT SR
260 PRINT **: END
265 END
190 COLOR=8: PLOT X,C,Y
195 IF SL=15 OR SR=15 THEN 260
200 COLOR=0: PLOT X,Y+4*(Y+4)/1
    AND Y+4(48)
205 FOR T=1 TO 75: IF T MOD 5=0
    THEN 218: IF PEEK (-16285)
    >127 AND S#5 THEN S=S+1: IF
    PEEK (-16287)<=127 AND S#5
    THEN S=S-1
210 GOSUB 235: NEXT T
215 Y=F(P(X): IF X=0 THEN YY=F(C1
    )
220 IF H THEN YY= RND (37)+1
225 V=1- RND (3)
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 5,165: POKE 6,0
10 POKE 14,193: POKE 15,24: POKE 16,240: POKE 17,56: POKE 18,180: POKE 19,1:POKE 20,76:
:POKE 21,2: POKE 22,0:POKE 23,36
15 DIM BN(48): TEXT : CALL -936
: GOTO 98
20 CALL -936: GOTO 98
30 B$=" pensions in" RETURN
35 B$=" Color Sketch": RETURN
40 B$=" Copyright Apple Computer 1978": RETURN
45 B$=" This program allows you to * : RETURN
50 B$=" Sketch colored figures in" RETURN
55 B$=" Low resolution graphics with *RETURN
60 XX=23:TON=24: GOSUB 85: RETURN
65 XX=18:TON=18: GOSUB 85: RETURN
70 XX=30:TON=50: GOSUB 85:XX=30:
: TON=50: GOSUB 85: RETURN
75 XX=28:TON=28: GOSUB 85: RETURN
80 XX=8:TON=256: GOSUB 85:XX=9:
: TON=256: GOSUB 85: RETURN
85 POKE 1, TON MOD 256: POKE 24
,TON/256+1: POKE 0,XX: CALL 2: RETURN
90 GOSUB 38: GOSUB 25: PRINT :
: TBR 13: GOSUB 35: GOSUB 25
: PRINT : GOSUB 30: GOSUB 25
: PRINT : TAB 5: GOSUB 40: GOSUB 25:
: PRINT : GOSUB 30: GOSUB 25
: PRINT 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$:
105 GR
110 B$="ABCDFGHJKLMNQP": CALL
-936
115 FOR Z=0 TO 15: COLOR=Z: PLOT
241+4,39: VTAB 21: GOSUB 75
120 VTAB 22:B$="Type a letter to change color": GOSUB 25: PRINT
: B$="Type space bar to stop plot
*: GOSUB 25: PRINT
125 Y= POL (1)X32/255;X= POL (0)
: 39/255: VTAB 24: TAB 1: PRINT
:*CSOR POSITION: X=;Y=; Y=
:*Y": "
130 IF PEEK (-16384))27 THEN 145
: IF XI=X AND YI=Y THEN 125
: COLOR=C: PLOT XI,YI: IF
NOT FLAG THEN 135: COLOR=C:
: PLOT X,Y
135 C2= SCREEN(X,Y);C2=15: IF C2=
15 THEN C3=5: COLOR=C1: PLOT
X,Y;XI=K;YI=I
140 GOTO 125
145 IF PEEK (-16384)160 THEN 155
:FLAG=0: POKE -16368,0: POKE
34,26: COLOR=0: HLINE 0,39 AT
39: CALL -936
150 PRINT :B$="CONTINUE OR STOP"
: VTAB 24: GOSUB 25: INPUT
:* (C/5) *,B$: IF B$11)="C"
THEN 118: PRINT "END": END
155 FLAG=1;C= PEEK (-16384)193
: POKE -16368,0: GOTO 125

58

Source: David T Craig
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

8 REM GAME OF MASTERMIND 8-25-77
WOZ (APPLE COMPUTER)
10 DIM K(3),C(8),D(5),X(8),X(8)
20 X(1)=2:X(2)=12:X(3)=X(1)
40: X(5)=3:X(6)=9:X(7)=15
50: X(8)=5: X(9)="BGRYOWK*
60 TEXT : CALL -936: PRINT 

WELCOME

TO THE GAME OF MASTERMIND!

YOUR OBJECT IS TO GUESS 5 COLORS

5 WHICH

30 PRINT * I WILL MAKE UP IN THE NEXT MINIMUM NUMBER OF GUESSES. THERE

E ARE EIGHT DIFFERENT COLORS TO CHOOSE FROM.*

40 PRINT *

FEWER THAN 7 GUESSES--EXCELLENT; PRINT "7 TO 9 GUESSES--GOOD"; PRINT "10 TO 14 GUESSES--AVERAGE"

50 PRINT "MORE THAN 14 GUESSES--POOR"

*: CALL -384: TAB 7: PRINT

"HIT ANY KEY TO BEGIN PLAY"

100 CALL -300: IF PEEK (-16384)
100: GOSUB 1000: IF KEY=149

"1 TO 5: COLOR=I; I=I: GOSUB 2000: NEXT I: IF I=5 THEN 500: RETURN"

300 REM CALL -394 SETS INVERSE VID

310 REM CALL -388 SETS NORMAL VID

320 REM PEEK (-16384) IS KB (ASCII)

330 REM POKE-16388 CLRS KB STROBE

340 REM CALL-936 CLEARS SCREEN AND

350 REM CURSOR TO TOP LEFT.

350 REM IN 310, KEY=5-28=-1 OR +1

360 REM STMS 10-30 INTRO

370 REM STMS 100-110 NEW SETUP

380 REM STMS 200 NEW GUESSES

390 REM STMS 300-310 USER INPUT

400 REM STMS 400 GUESS EVAL

410 REM STMS 500-510 WIN

420 REM STMS 600-700 COLOR LINE

430 REM SUBR 2000 MATCH TEST

END
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 6,0;
: POKE 7,32; POKE 8,168; POKE 9,252; POKE 10,165; POKE 11,1;
: POKE 12,288; POKE 13,4; POKE 14,198; POKE 15,24; POKE 16,248; POKE 17,5; POKE 18,198; POKE 19,1; POKE 20,76; POKE 21,2; POKE 22,0; POKE 23,96;
15 GOTO 85;
28 TT=3: GOSUB 38: RETURN;
25 PRINT "**********: RETURN;
30 KK=8:TH=588: GOSUB 45: RETURN;
35 KK=8:TH=258: GOSUB 45: RETURN;
40 KK=8:TH=258: GOSUB 45:KK=9;
:TH=258; GOSUB 45: RETURN;
45 POKE 1,TH MOD 256: POKE 24,
:TH=256+1: POKE 0,0,KK: CALL 2;
: RETURN;
50 A=0: B=0: C=0: D=0: X=0: Y=0: Z=0;
55 X=0: Y=0: Z=0: 
:RETURN;
60 J=1: G= POKE 34,23: FOR X=10 TO 28: COLOR=3: HLINE 0,31
: AT X: NEXT X: HLINE 1,3 AT
: 2: VTAB 21;
115 FOR Y=1 TO 31: STEP 3: PRINT Y;
: IF Y > 10 THEN PRINT " ";
: PRINT " ": NEXT Y: PRINT " P E M " : VTAB 24;
120 VTAB 23: PRINT "DAYS LIVED "
: IN: FOR I=1 TO 3: COLOR=1+I
: 0,39 AT 33+I: VTAB 24;
125 FOR X=0 TO 31: P=45 MOD BV(1)
: I;
130 PRINT : INPUT "ANOTHER PLOT (Y/N ) ",B3; IF B3(1,1)="Y" THEN
: 90: END;

62

Source: David T Craig
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 BUMP INTO THEM."
6 PRINT "TO MOVE, YOU HIT 'R' FOR RIGHT.,"
7 PRINT "'L' FOR LEFT, 'U' FOR UP, AND 'D' FOR DOWN. DO NOT HIT RETURN!"
8 PRINT "THE OBJECT IS FOR YOU (THE GREEN DOT)."
9 PRINT "GET TO THE DOOR ON THE RIGHT SIDE."
10 PRINT "BEFORE THE DRAGON (THE RED DOT) EATS YOU."
11 PRINT "Beware!!!!!!!!!!! Sometimes the dragon gets real mad, and climbs over a wall."
12 PRINT "Most of the time, he can't go over."
13 PRINT "AND HAS TO GO AROUND."
14 PRINT "PRINT: YOU CAN OFTEN TELL WHERE A WALL."
15 PRINT "STOP, EVEN BEFORE YOU CAN SEE IT, BY."
16 PRINT "THE FACT THAT THE DRAGON CAN'T GET."
17 PRINT "THROUGH IT."
18 PRINT "IN A MISTY." 
19 PRINT "GO TO BEGIN."
20 PRINT "TYPE 'GO' TO BEGIN."
21 PRINT "INPUT A":
22 PRINT "COLOR=15"
23 PRINT "CALL -936: PRINT "DRAGON MAZE"
24 PRINT "TAB (25): PRINT "GARY J. SHAN"
25 PRINT "Now:" 
26 PRINT "FOR i=0 TO 39 STEP 3: VLINE 0,39 AT i: HLINE 0,39 AT i: NEXT i"
27 PRINT "128 COLOR=0"
28 PRINT "DIM X(169),Y(169)"
29 PRINT "FOR i=1 TO 169:(i-1)=i: NEXT i"
30 PRINT "165: FOR i=1 TO 169:Y(i)=i: NEXT i"
31 PRINT "1830 X= RND (13)+1; Y= RND (13)+1; C=169"
32 PRINT "1835 IF C=1 THEN 1280"
33 PRINT "1840 R=8: D=8: B=8: U=8: X=X+13*Y-Y"
34 PRINT ":MK=0: ABS (MK):C=C-1"
35 PRINT "1850 IF X=13 THEN 1868: R=MK+1"
36 PRINT "0"
37 PRINT "1868 IF Y=13 THEN 1870: D=MK+1"
38 PRINT "0"
39 PRINT "1870 IF X=1 THEN 1870: L=MK-1"
40 PRINT "0"
41 PRINT "1880 IF Y=1 THEN 1890: U=MK-1"
42 PRINT "0"
43 PRINT "1080 G=G+1: D=0: U=0: X=X+13*Y-Y"
44 PRINT "1100 IF I<9 THEN 1170"
45 PRINT "1100 OR= RND (4)"
46 PRINT "1120 GOTO 1130: OR=0"
47 PRINT "1130 IF NOT OR THEN 1110: MK=MK+1"
48 PRINT "1135 VLINE 3*Y-2,3*X-1 AT 3*X-1"
49 PRINT "1136 GOTO 1085"
50 PRINT "1140 IF NOT D THEN 1110: MK=MK+1"
51 PRINT "1145 HLINE 3*X-2,3*Y-1 AT 3*Y-1"
52 PRINT "1146 GOTO 1085"
53 PRINT "1150 IF NOT L THEN 1110: MK=MK+1"
54 PRINT "1155 VLINE 3*Y-2,3*X-1 AT 3*X-1"
55 PRINT "1156 GOTO 1085"
56 PRINT "1160 IF NOT 0 THEN 1110: MK=MK+1=
57 PRINT "1165 HLINE 3*X-2,3*Y-1 AT 3*Y-1"
58 PRINT "1170 X= RND (13)+1: Y= RND (13)+1"
59 PRINT "1180 IF MK+13*Y-Y=10 THEN 1170"
60 PRINT "1190 C=C+1: GOTO 1085"
61 PRINT "1200 GOSUB 5000: PRINT "THE MAZE IS READY"
62 PRINT "COLOR=15"
63 PRINT "1220 X=1: Y= RND (13)+1: X=X+13:COLOR=0:
64 PRINT "PLOT 3*X-2,3*Y-2"
DRAGON MAZE cont.

1225 RX=3*RY-2; NY=3*Y-2
1230 Wy= RND (13)+1
1240 COLOR=0: VLIN 3*NY-2,3*NY-1
   AT 39
1250 SX=13*RY-WY
1260 DX=3*SX-2; WY=3*SY-2
1270 RD=1
1500 K= PEEK (-16384); IF K<>128 THEN
   1500
1510 POKE -16368, 0
1515 00=K: GOSUB 7000; X=00
1516 IF SX<X AND SY=Y THEN 8000
1520 IF K= ASC("R") THEN 2000
1530 IF K= ASC("L") THEN 2500
1540 IF K= ASC("U") THEN 3000
1550 IF K= ASC("D") THEN 3500
1560 GOSUB 5000; GOTO 1500
2000 DX=1; DY=0
2010 IF NX+13*(Y-1) MOD 10 THEN
   4000
2020 FX=3*XY-2; FY=3*Y-2; FOR I=1 TO 3
2030 FX=FX+DX; FY=FY+DY
2600 FOR K=0 TO 1; FOR L=0 TO 1:
   PLOT RX+K,RY+L: NEXT L,X: COLOR=8:
   FOR K=0 TO 1: FOR L=0 TO 1:
   I: PLOT RX+K,RY+L: NEXT L,X:
   RX=FX+K; FY=FY+L:
2110 NEXT I
2115 RX=RX+DX; XY=Y+DY
2116 IF X=13 AND Y=6000 THEN 6000
2120 GOTO 1500
2500 DX=1; DY=0
2510 IF MX+13*(Y-1) MOD 10 THEN
   4000
2520 GOTO 2020
3000 DX=0; DY=-1
3100 IF MX+13*(Y-2)/10 THEN 4200
3620 GOTO 2020
3500 DX=0; DY=1
3510 IF MX+13*(Y-1)/10 THEN 4300
3520 GOTO 2020
4000 GOSUB 5000
4010 COLOR=5
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)
4300 GOTO 1500
4300 GOSUB 5000
4310 COLOR=15
4320 HLIN 3*X-1,3*X AT 3*Y
4330 GOTO 1500
5000 S=1: FOR I=1 TO 28; A: PEEK
   (-16336)+ PEEK (-16336)+ PEEK
   (-16336)+ PEEK (-16336)+ NEXT
   1: RETURN
6000 PRINT "YOU WIN!"
6100 GOSUB 5000; GOSUB 5000: GOSUB
   5000
6200 PRINT "SCORE=":S+3
6300 END
7000 IF X>SY THEN 7085: IF Y>SY THEN
   7050
7001 IF X<SY THEN 7100: IF Y<SY THEN
   7150
7005 IF SX<13 THEN 7050: IF TX<SY+
   13*(SY-1)/10 THEN 7100: IF
   MX<13*(SY-1)/10 THEN 7050
7010 DX=1; DY=0
7020 COLOR=0
7022 RX=3*SY-2; RY=3*SY-2
7023 FOR I=1 TO 3: RX=RX+DX; RY=RY+
   DY
7024 COLOR=0
7025 FOR K=0 TO 1: FOR L=0 TO 1:
   PLOT RX+K,RY+L: NEXT L,X: COLOR=
   8: FOR K=0 TO 1: FOR L=0 TO 1:
   I: PLOT RX+K,RY+L: NEXT L,X:
   RX=RX+DY; RY=
7030 NEXT 1
7035 SY=SY+DX; SX=SY+DY
7040 T(SX+13*(SY-1))=T(SX+13*(SY-
   1)+1)
7045 RETURN
7050 IF SY<13 THEN 7100: IF T(SX+
   13*(SY-1))=T(SX+13*(SY-1)) THEN
7060 DX=0; DY=1: GOTO 7020
7100 IF SX<1 THEN 7150: IF T(SX+
   13*(SY-1))=T(SX+13*(SY-1)) MOD 10 THEN
7150

Source: David T Craig
DRAGON MAZE cont.

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

>
APPLE II FIRMWARE

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

Apple II contains a powerful machine level monitor for use by the advanced programmer. To enter the monitor either press the RESET button on the keyboard or \texttt{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\textsuperscript{0} (control C). \textbf{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>data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Move Memory</td>
<td></td>
<td></td>
</tr>
<tr>
<td>adrs1&lt;adrs2.</td>
<td>*100&lt;010B410M</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;010B410V</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><strong>Cassette I/O</strong></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><strong>Display</strong></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><strong>Dis-assembler</strong></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><strong>Mini-assembler</strong></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>:C010: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>
</tbody>
</table>
### Command Format

<table>
<thead>
<tr>
<th>Command Format</th>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(space) (6502 mnemonic instruction)</td>
<td>! STA 01FF</td>
<td>Assembles instruction into next available memory location. (Note space between &quot;!&quot; and instruction)</td>
</tr>
<tr>
<td>(TURN-OFF)</td>
<td>! (Reset Button)</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 | *800T | 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 (3F0). |

**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

**Set Input/Output Ports**

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

**Note 2:**

Only slots 1 through 7 are addressable in this mode. Address \( \emptyset \) (Ex: \( \emptyset pC \) or \( \emptyset kC \)) 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</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*1GG1 4GGG AFFT</td>
<td>Multiple monitor commands may be given on same line if separated by a &quot;space&quot;.</td>
</tr>
<tr>
<td>*LLLL</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 GC. They are obtained by holding down the CTRL key while typing the specified letter. Control characters are NOT displayed on the TV screen. BC and CC must be followed by a carriage return. Screen editing characters are indicated by a sub-scripted "E" such as DE. 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, UC moves to cursor to right and copies text while AE 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 without using control button.</td>
</tr>
<tr>
<td>Control J</td>
<td>Issues line feed only</td>
</tr>
<tr>
<td>Control V</td>
<td>Compliment to HC. Forward spaces cursor and copies over written 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.
### SPECIAL CONTROL AND EDITING CHARACTERS

(continued)

<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>@_E</td>
<td>Home cursor to top of page, clear text to end of page</td>
</tr>
</tbody>
</table>
## Special Controls and Features

### Hex BASIC Example Description

#### 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 32L1</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;1&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_E) 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_F) 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>
</tbody>
</table>

**Miscellaneous**

| C030 | 360 X=PEEK(-16336) 365 POKE -16336,0 | Toggle speaker                                   |
| C000 | 370 X=PEEK(-16384) | Read keyboard; if X>127 then key was pressed.   |
| C010 | 380 POKE -16368,0  | Clear keyboard strobe - always after reading keyboard. |
| C061 | 390 X=PEEK(16287)  | Read PDL(0) push button switch. If X>127 then switch is "on". |
| C062 | 400 X=PEEK(-16286) | Read PDL(1) push button switch.                 |
| C063 | 410 X=PEEK(-16285) | Read PDL(2) push button switch.                 |
| C058 | 420 POKE -16296,0  | Clear Game I/O AN0 output                       |
| C059 | 430 POKE -16295,0  | Set Game I/O AN0 output                         |
| C05A | 440 POKE -16294,0  | Clear Game I/O AN1 output                       |
| C05B | 450 POKE -16293,0  | Set Game I/O AN1 output                         |
| C05C | 460 POKE -16292,0  | Clear Game I/O AN2 output                       |
| C05D | 470 POKE -16291,0  | Set Game I/O AN2 output                         |
| C05E | 480 POKE -16290,0  | Clear Game I/O AN3 output                       |
| C05F | 490 POKE -16289,0  | Set Game I/O AN3 output                         |
| LOC0 | EPZ | $00 |
| LOC1 | EPZ | $01 |
| WNDLFT | EPZ | $20 |
| WNDPTH | EPZ | $21 |
| WNDTOP | EPZ | $22 |
| WNDBTM | EPZ | $23 |
| CH | EPZ | $24 |
| CV | EPZ | $25 |
| GRASL | EPZ | $26 |
| GRASH | EPZ | $27 |
| DASL | EPZ | $28 |
| BASL | EPZ | $29 |
| BAS2L | EPZ | $2A |
| BAS2H | EPZ | $2B |
| H2 | EPZ | $2C |
| LNNEM | EPZ | $2C |
| RTNL | EPZ | $2C |
| V2 | EPZ | $2D |
| RNNEM | EPZ | $2D |
| RTNF | EPZ | $2D |
| MASK | EPZ | $2F |
| CHKSUM | EPZ | $2F |
| FORMAT | EPZ | $2F |
| LASTIN | EPZ | $2F |
| LENGTH | EPZ | $2F |
| SIGN | EPZ | $2F |
| COLOR | EPZ | $30 |
| MODE | EPZ | $31 |
| INVFLG | EPZ | $32 |
| PROMPT | EPZ | $33 |
| YSAV | EPZ | $34 |
| YSAV1 | EPZ | $35 |
| CSWL | EPZ | $36 |
| CSWH | EPZ | $37 |
| NSWL | EPZ | $38 |
| NSWH | EPZ | $39 |
| PCL | EPZ | $3A |
| PCH | EPZ | $3B |
| XOT | EPZ | $3C |
| A1L | EPZ | $3C |
| A1H | EPZ | $3D |
| A2L | EPZ | $3E |
| A2H | EPZ | $3F |
| A3L | EPZ | $40 |
| A3H | EPZ | $41 |
| A4L | EPZ | $42 |
| A4H | EPZ | $43 |
| A5L | EPZ | $44 |
| A5H | EPZ | $45 |
ACC  EP2  $45
XREG  EP2  $46
YREG  EP2  $47
STATUS  EP2  $48
SPHT  EP2  $49
HNL  EP  $4A
PNHN  EP2  $4B
ACL  EP2  $50
ACH  EP2  $51
XIROL  EP2  $52
XIRPL  EP2  $53
AUXL  EP2  $54
AUXN  EP2  $55
PICL  EP2  $56
IN  EQU $0200
USHADIR EQU $03F8
IHI EQU $03FB
IFOLOC EQU $03FF
ICONR EQU $C000
KAP EQU $C000
KNSRTHIR EQU $C010
TAPFOUT EQU $C020
SPCR EQU $C030
TXCLR EQU $C050
TXTSET EQU $C051
MIXCLR EQU $C052
MIXSET EQU $C053
LONSCR EQU $C054
HISCR EQU $C055
LORES EQU $C056
HIRES EQU $C057
TAPPIN EQU $C060
PADINS EQU $C064
PPFPRG EQU $C070
BASIC EQU $5000
BASIC2 EQU $5003
ORG $8800  FOR START ADDRESS
P800: 4A  PLOT  LSR A  X-COORD \\
F801: 0B  PHP  SAVE LSB IN CARRY
P802: 20 47 F6  JSR GRASCALCALC BASE ADDR IN GBASL,H
P805: 28  PLP  RESTORE LSB FROM CARRY
P806: A9 0F  LDA #$80  MASK $80 IF EVEN
P808: 90 02  BCC RPFMARK
P80A: 69 E0  ADC $80  MASK $80 IF ODD
P80C: 85 22  RPFMARK  STA $80
P80E: 81 26  PLOT1  LDA (GBASL),Y DATA
P810: 45 30  EOR COLOR XOR COLOR
P812: 25 26  AND MASK  AND MASK
P814: 51 26  EOR (GBASL),X XOR DATA
P816: 91 26  STA (GBASL),X TO DATA
P818: 60  RTS
P819: 20 00 F8  HLINE  JSR PLOT  PLOT SQUARE
P81C: C4 2C  HLINE1  CPY H2  DONE?
P81E: 80 11  BCS RTS1  YES, RETURN
P820: C8  INY  NO, INCX INDEX (X-COORD)
P821: 20 00 F6  JSR PLOT1. PLOT NEXT SQUARE
P824: 90 F6  SCC HLINE1 ALWAYS TAKEN
P826: 69 01  VLINE2  ADC #$01  NEXT Y-COORD
P828: 48  VLINE  PHA  SAVE IN STACK
P829: 20 00 F8  JSR PLOT  PLOT SQUARE
P82C: 68  PLA
P82D: C5 2D  CMP V2  DONE?
P82F: 90 F5  RCL VLINE2 NO, LOOP.
P831: 60  RTS1  RTS
P832: A0 2F  CLRSCF  LDY #$2F  MAX Y, FULL SCRNL CLR
P834: 00 02  PME CLRSC2 ALWAYS TAKEN
P836: A0 27  CLRTOP  LDY #$27  MAX Y, TOP SCRNL CLR
P838: 84 2D  CLRSC2  STY V2 STORE AS LOWER COORD  
* FOR VLINE CALLS
P83A: A0 27  LDY #$27  RIGHTMOST X-COORD (COLUMN)
P83C: 9F 00  CLRSC3  LDA #$50  TOP COORD FOR VLINE CALLS
P83E: 85 30  STA COLOUR CLR54 COLOR (BLACK)
P840: 20 28 F6  JSR VLINE TRN VLINE
P843: 88  DEY NEXT LFFTMOST Y-COORD
P844: 10 F6  NFL CH4C3 LOOP UNTIL DONE.
P846: 60  RTS
P847: 48  GRASCALCALC PHA  FOR INPUT MODEFE3
P848: 4A  INY A
P849: 29 03 AND #$03
P84A: 09 04 GPA #$04 GENERATE GRASS=A0C001FG
P84D: 05 27 STA GBASH
P84F: 69 PLA
P850: 29 18 AND #$18
P852: 90 02 SCL CACALC
P854: 69 7F ADC #$7F
P856: 85 26 GSCALCALC STA GBASH

77

Page 0082 of 0156

78

"DTCA2DOC-469-078.PICT" 406 KB 2001-06-26 dpi: 800h x 800v pix: 2608h x 5379v

80

Source: David T Craig
| FA40: | FF FF FF | OEF SPP,SPF,SPF |
| FA43: | 20 00 00 | STEP | JSR INST358P | DISASSEMBLE ONE INST |
| FA46: | 6B | PLA | AT (PCL,R) |
| FA47: | 85 22 | STA | RTNL | ADJUST TO USER |
| FA49: | 68 | PLA | STACK, SAVE |
| FA4A: | 85 2D | STA | RTNH | RTM ADDR. |
| FA4C: | 8D 2B | LDX | $50P |
| FA4E: | BD 10 00 | XININIT | LDA | INIT3L1,X INIT XEO AREA |
| FA51: | 95 3C | STA | XOT,X |
| FA53: | CA | DEX |
| FA54: | D0 F8 | XININIT |
| FA56: | A1 3A | LDA | (PCL,X) | USER Opcode Byte |
| FA58: | F0 42 | BEQ | XNRK | SPECIAL IF BREAK |
| FA59: | A4 2F | LDP | LENGTH | LEN FROM DISASSEMBLY |
| FA5C: | 99 20 | CMP | $20 |
| FA5E: | F0 59 | BEQ | XJSR | HANDLE JSR, PTS, JMP, |
| FA60: | 99 60 | CMP | $560 | JMP ( ), PTI SPECIAL |
| FA62: | F0 45 | BEQ | XRTS |
| FA64: | C9 4C | CMP | $56C |
| FA68: | C9 6C | CMP | $56C |
| FA6A: | F0 59 | BEQ | XJMPAT |
| FA6C: | C9 | CMP | $40 |
| FA6F: | 21 3A | LDA | (PCL,Y) | CHANGE REL BRANCH |
| FA70: | 99 00 | STA | XOTNH,Y | DISP TO 4 FOR |
| FA71: | 88 | JMP | TO BRANCH OR |
| FA75: | 10 F8 | BPL | XQ1 | NBRAANCH FROM EQX. |
| FA78: | 20 3F | JSR | RESTORE | RESTORE USER REG CONTENTS. |
| FA7B: | 4C 3C | JMP | XQTH2 | XEO USER OP FROM RAM |
| FA7C: | 85 45 | IRQ | STA | ACC (RETURN TO NBRANCH) |
| FA80: | 6B | PLA |
| FA82: | 48 | PLA |
| FA84: | 0A | ASL | **IRQ HANDLER |
| FA88: | 0A | ASL | A |
| FA8C: | 0A | ASL | A |
| FA8C: | 30 03 | BMI | BREAK | TEST FOR BREAK |
| FA8F: | 6C FE 03 | JMP | (IORLOC) | USER ROUTINE VECTOR IN RAM |
| FA92: | 28 | PNP | BREAK | |
| FA93: | 20 4C | JSR | SAV1 | SAVE REG'S ON BREAK |
| FA95: | 6B | PLA | INCLUDING PC |
| FA97: | 85 3A | STA | PCL |
| FA98: | 68 | PLA |
| FA9A: | 85 3B | STA | PCH |
| FA9C: | 20 82 | JSR | IPSDS1 | PRINT USER PC. |
| FA9F: | 20 0A | JSR | RGDS1 | AND REG'S |
| FAAC: | 4C 65 | JMP | MON | GO TO MONITOR |
| FAA5: | 18 | XRTI | CLC |
| FAA6: | 6B | PLA | SIMULATE RTI BY EXPECTING |
| FAA7: | 85 4B | STA | STATUS | STATUS FROM STACK, THEN RTS |
| FAA9: | 6B | XRTS | PLA | RTS SIMULATION |
| FAAB: | 85 3A | STA | PCL | EXTRACT PC FROM STACK |
| FAAC: | 68 | PLA | AND UPDATE PC BY 1 (LEN-0) |
| FAAD: | 85 3B | STA | PCH |
| FAAF: | A5 2F | JSR | PCAD2J3 |
| FAB1: | 20 56 | JSR | PCAD2J3 |
| FABC: | 84 3B | STY | PCH |
| FAB6: | 18 | CLC |
| FABC: | 55 14 | BCC | NEWPCL |
| FAB9: | 18 | XJSR | CLC |
| FABA: | 20 54 | JSR | PCAD2J3 | (UPDATE PC AND PUSH |
| FABD: | 9A | TAX | onto stack for |
| FABE: | 9B | TYA | JSR SIMULATE |
| FABF: | 4B | PHA |
| FAC0: | 5A | TAX |
| FAC1: | 4B | PHA |
| FAC2: | A0 02 | LDY | $92 |
| FAC4: | 18 | XJMP | CLC |
| FAC5: | B1 3A | JSR | XJMPAT | LDA | (PCL),Y | LOAD PC FOR JMP, |
| FAC7: | AA | TAY | (JMP) SIMULATE. |
| FAC8: | 88 | DEY |
| FAC9: | B1 3A | LDA | (PCL),Y |
| FACB: | 86 3B | STX | PCH |
| FACD: | 85 3A | NEWPCL | STA | PCL |
| FACF: | B0 03 | BCS | XJMP |
| FADD: | A5 2D | RTNJMP | LDA | RTNH |
| FADE: | 4B | PHA |
| FAF4: | A5 2C | LDA | RTNL |
| FAF6: | 4B | PHA |
| FADD: | 20 BE FD | JSR | CRTOUT | DISPLAY USER REG |
| FADD: | A9 45 | RGDS1 | LDA | $ACC | CONTENTS WITH |
| FADD: | 85 40 | STA | A3L | LABELS |

81
PC18: 80 0B BCS RTS4 IF TOP LINE THEN RETURN
PC20: C6 25 DEC CV DEC R CURSOR V-INDEX
PC22: A5 25 VTAB LDA CV GET CURSOR V-INDEX
PC23: 20 FB VTAB2 JSR PASCAL CLEAR PASTE BASE ADDR
PC27: 65 20 ADC WHLEFT AND WINDOW LEFT INDEX
PC29: 85 28 STA BASL TO BASL
PC31: 49 C0 ESC1 EOP $50 ESC?
PC32: F0 28 RPO HOME IF ON, DO HOME AND CLEAR
PC33: 6F 9D ADC #$FD ESC-A OR B CHECK
PC34: 9C 0D BCC ADVANCE A, ADVANCE
PC36: 69 FD ADC #$FD ESC-C OR D CHECK
PC38: 90 2C SEC LF SEC DOWN
PC3A: 00 DE BEQ UP D, GO UP
PC3C: 69 FD ADC #$FD ESC-E OR F CHECK
PC3E: 90 2C SEC CLREOL F, CLEAR TO END OF LINE
PC40: 00 69 BNE $54 NOT F, RETURN
PC42: A4 24 CLIOP LDY CH CURSOR H TO Y INDEX
PC44: A5 25 LDY CV CURSOR V TO A-REGISTER
PC46: 48 CLIOP1 PHA SAVE CURRENT LINE ON STK
PC47: 20 24 JSR VTAB2 CALC BASE ADDRESS
PC49: 20 9E JSR CLIOLX CLEAR TO END, SET CARRY
PC80: 00 60 LDY #$00 CLEAR P""M"M H INDEX=0 FOR POFST
PC84: 68 PLA INCREMENT CURRENT LINE
PC85: 69 00 ADC #$00 (CARRY IS SET)
PC86: C5 23 CMP $D000 DONE TO BOTTOM OF WINDOW?
PC88: 90 F0 BCC CLIOP1 NC, KEEP CLEARING LINES
PC8E: 80 CA BCS VTAB YES, TAB TO CURRENT LINE
PC91: A5 22 LDA #00 INIT CURSOR V
PC93: A5 25 STA CV AND H-INDICES
PC95: A0 00 LDY #$00 THEN CLEAR TO END OF PAGE
PC96: 84 24 STY CH
PC98: 0D 04 BNE CLIOP1
PC9A: A9 00 LDA #$00 CURSOR TO LEFT OF INDY
PC9C: 85 24 STA CH,CH (PET CURSOR H=0)
PC9E: 86 25 INC CV INCR CURSOR V(DOWN 1 LINE)
PCA0: A5 25 LDA CV
PCB2: 86 25 CMP $D000 OFF SCREEN?
PCB4: 90 86 BCC CLREOL4 MG, SET BASE ADDR
PCB6: C6 25 SEC CLREOL5 DEC CURSOR V(BACK TO BOTTOM LINE)
PCB8: A5 22 SCROLL LDA #0000 START AT TOP OF SCR2 WIN3
PCB9: 48 PHA
PCB9: 20 24 JSR VTAB3 GENERATE BASE ADDRESS
PCBA: 85 28 LDA BASL COPY BASL,H
PCBC: 85 2A STA BASL TO BASL,H
PCBD: 85 29 LDA BASH
PCBE: 85 2B STA BAS2H
PCBF: A4 21 LDY $D000 INIT Y TO RIGHTMOST INDEX
PCC0: 81 8F DEY OF SCROLLING WINDOW
PCC1: 09 8F PLA
PCC2: 69 01 ADC #$01 INCX LINE NUMBER
PCC4: C5 23 CMP $D000 DONE?
PCC6: 80 0D BCS SCRL3 YES, FINISH
PCC8: 48 PHA
PCC9: 20 24 JSR VTAB3 FORM BASL,H (BASE ADDR)
PCCA: B1 85 LDA (BASL),Y MOVE A CHR UP ON LINE
PCCD: 91 2A STA (BAS2L),Y
PCCE: 88 88 DEY NEXT CHAP OF LINE
PCCF: 10 09 BPL SCRL2
PCD1: 30 81 BMI SCRL1 NEXT LINE
PCD3: A0 00 BCC SCRL3 CLEAR BOTTOM LINE
PCD6: 20 9E JSR CLIOLX GET PASTE ADDR FOR BOTTOM LINE
PCD8: 80 86 SEC VTAB CAPS IS SET
PCD9: A4 24 CLREOL LDA CH CURSOR H INDEX
PCDB: A9 80 CLREOL LDA $AB STORE BLANKS FROM 'HERE'
PCC0: 81 28 STA (BASL),Y STORE BLANKS FROM 'HERE'
PCC2: C0 8F INY TO END OF LINE (WD/WDTH)
PCC5: 90 F9 BCS CLREOL2
PCC7: 60 PTS
PCC8: 38 87 07 09 WAIT SCC
PCC9: 48 87 07 09 WAIT2 PHA
PCDA: E9 01 07 03 07 00 07 01 WAIT3 SPC $50
PCDC: 80 FC 07 03 07 03 07 03 PLY WAIT3 1.0204 USEC...
PCDE: 68 87 07 03 07 03 07 03 PLA
PCF0: E9 01 07 03 07 03 07 03 SPC $50
PCF1: 00 F0 07 03 07 03 07 03 BNE WAIT2
PCF3: 60 07 03 07 03 07 03 PTS
PCF4: 86 42 INCPA4 INC A4 INC 2-FYPE A4
PCF6: 00 02 INC NXATA1 AND A1
PCF8: 86 43 INC A4 INC 2-FYPE A1
PCFA: A5 3C LDA A4 INC NXATA1 AND A1
PCFC: C5 3E CMP A4L INC 2-FYPE A1
PCFE: A5 3D LDA A4L AND COMPARE TO A2

84
FCC0: E5 3F SBC A2h  (CARRY SET IF >=)
FCC2: E6 3C INC A1h
FCC4: D0 02 RNE RTSB
FCC6: E6 3D INC A1h
FCCB: 60 RTS RTS
FCCL: A0 48 HEAD LDR #40p WRITE A#56 'LONG 1'
FCCK: 20 DB FC JSR ZERONE FULL CYCLES
FCCH: D9 0F SNE HEAD (650 USEC EACH)
FCDO: 69 FF ADC #5F FF
FCD2: E0 05 MPS MODA THEN A 'SHOFT 0'
FCD3: A0 21 (400 USEC)
FCD6: 20 DB FC WRIT JSR ZERONE WRITE TWO FULL CYCLES
FCD9: C8 INY OF 250 USEC ('0')
FCD4: C8 INY OR 500 USEC ('0')
FCDB: 88 ZERONE DEY
FCDG: D0 FD BNE ZERONE
FCDE: 00 05 ACC WRAP Y IS COUNTER FOR
FCED: A0 32 LDR #512 TIMING LOOP
FCE2: 88 ONELOL DEY
FCE3: D0 FD ADD Y ENE ONELOL
FCE5: AC 20 CO WRAP LDR TAKEOUT
FCE8: A0 2C LDR #2C
FCE9: CA DEX
FCEB: 60 RTS
FCEC: A2 68 RDRY16 LDR #50 8 BITS TO READ
FCEE: 4B RDRY16 PHS READ TWO TRANSITIONS
FCFe: 20 FA FC JSR RDRY16 (FIND EDGE)
FCE2: 68 PLA
FCE4: 2A ROL A NEXT BIT
FCE4: AD 3A LDR #3A COUNT FOR SAMPLES
FCE6: CA DEX
FCE7: D0 05 BNE RDRY16
FCE9: 60 RTS
FCFA: 20 FD FC WED2 LDR RDIY IT
FCFD: 88 RDIY IT DEY DEC Y UNTIL
FCFE: AD 60 CO LDA TAPPIN TAPP TRANSITION
FDF1: 45 2F EOR LASTIN
FDF2: 10 08 ANL RDIY IT
FDF5: 45 2F EOR LASTIN
FDF7: 08 2F STA LASTIN
FDF9: 00 80 CPY #255 SET CARRY ON Y-REG.
FDD6: 60 RTS
FDD7: A4 24 SD6 KEY LDR CN
FDD9: B1 28 LDA (PRL),Y SET SCREEN TO FLASH
FDEL: 47 PLA
FD1: 29 3F AND #3Fh 3DO 51h '41h
FD1: 09 40 OPA #30h
FD15: 91 28 STA (PRL),Y
FD17: 68 PLA
FD18: 6C 3B 00 Jmp (KSEL) GO TO USER KEY-IN
FD1B: E6 46 KEYIN INC #61h
FD1D: D0 02 SBF KEYIN2 INCR AND NUMBER
FD1F: E6 4F INC EON
FD21: 2C 00 04 KEYIN2 BIT NDB KEY DONT?
FD24: 10 05 EYL KEY1 LOOP
FD28: 2C 28 00 STA (PRL),Y REPLACE FLASHING SCREEN
FD28: B0 00 CO LDA #255 TDF KEYCODE
FD2B: 2C 10 00 FIT KPOODPP CLR KEY STORE
FD2E: 60 RTS
FD30: 20 DC FD ESC JSR PUKSY GET KPOODDP
FD32: 20 2C FC JSR E5 81h HANDLE ESC FUNC.
FD35: 20 DC FD POCHAP JSR PUKSY READ KEY
FD38: C9 9B CMP #95h ESC?
FD3A: F0 03 CMP #80h ESC YES, DON'T RETURN
FD3C: 60 RTS
FD3D: A5 32 NOTCR LDA INVPLG
FD3F: 48 PLA
FD40: A9 02 LDA #5Fh
FD42: 05 32 STA INVPLG (ECHO USER LINE
FD44: BD 00 02 LDA IN, X NAME IF NOT INVRS.
FD47: 20 ED FD JSR COUT
FD4A: 88 PLA
FD4B: 85 32 STA INVPLG
FD4D: BD 00 02 LDA IN, X NAME IF NOT INVRS.
FD50: C9 88 CMP #68h CHECK FOR EDIT KEYS
FD52: F0 1D EEO BCSKPC 88, CTRL-X.
FD54: C9 98 CMP #89h
FD56: F0 0A EEO CANCEL
FD58: CPX #18h MARGIN?
FD5A: 00 03 BCC NOTCR
FD5C: 20 3A FF JSR BELL YES, SOUND BELL.
FD5F: 8B NOTCR INX ADVANCE INPUT INDEX
FD60: D0 13 RNE NXTCHR
FD62: A9 DC CANCEL LDA #55h BACKSPACE AFTER CANCELLED LINE
FD64: 20 ED FD JSR COUT

85
FE0F: 91 40  STA (A3L),Y  STOPE AS LOW BYTE AS (A3)
FE11: 8E 40  INC A3L
FE13: D0 02  BNE RTS5  INCR A3, RETURN
FE15: B6 41  INC A3H
FE17: 60  RTS
FE18: A4 34  SFTMODE  LDY YSAV  SAVE CONVERTED '+', '+',
FE1A: B9 FF 01  LDA IN-1,Y  '-', '-' AS MODE.
FE1B: B5 31  STAMOD2  STA '00DP
FE1F: 60  RTS
FE20: A2 01  LT  LDX #601
FE22: B5 3E  LT2  LDA A2L,X  COPY A2 (2 BYTES) TO
FE24: 95 42  STA A4L,X  A4 AND A5
FE26: 95 44  STA A5L,X
FE28: CA  DEX
FE29: 10 F7  BPL LT2
FE2B: 60  RTS
FE2C: 81 3C  MOVE  LDA (A1L),Y  MOVE (A1 TO A2) TO
FE2E: 91 42  STA (A4L),Y  (A4)
FE30: 20 B4 FC  JSR NXTA4
FE33: 90 F7  BCC MOVE
FE35: 60  RTS
FE36: B1 3C  VFY  LDA (A1L),Y  VERIFY (A1 TO A2) WITH
FE38: D1 42  CMF (A4L),Y  (A4)
FE3A: F0 3C  PEO VFYOK
FE3C: 20 92 FD  JSR PHA1
FE3F: B1 3C  LDA (A1L),Y
FE41: 20 DA FD  JSR PRYYTE
FE44: A9 A0  LDA #5A6
FE46: 20 ED FD  JSR COUT
FE49: A9 A8  LDA #5AB
FE4B: 20 ED FD  JSR COUT
FE4E: B1 42  LDA (A4L),Y
FE50: 20 DA FD  JSR PRYYTE
FE53: A9 A9  LDA #5A9
FE55: 20 ED FD  JSR COUT
FE58: 20 B4 FC  VFYOK  JSR NXTA4
FE5B: 90 09  BCC VFY
FE5D: 60  RTS
FE5E: 20 75 FE  LIST  JSR A1PC  VEE A1 (2 BYTES) TO
FE61: A9 14  STA #61A  PC IF SPEC'D AND
FE63: 48  LIST2  PHA DISSEMBLE 20 INSTRS
FE64: 20 D0 F8  JSR LHSTDSP
FE67: 20 53 F9  JSP PCREL ADJUST PC EACH INSTR
FE6A: 85 3A  STA PCL
FE6C: 84 38  STY PCH
FE6E: 68  PLA
FE6F: 38  SEC
FE70: E9 01  SBC #501  NEXT OF 20 INSTRS
FE72: D0 EF  BNE LIST2
FE74: 60  RTS
FE75: 8A  A1PC  TXA IF USEw SPEC'D ADDR
FE76: F0 07  PEC A1PCSTS COPY FROM A1 TO PC
FE78: 85 3C  A1PCLP  LDA (A3L),X
FE7A: 95 3A  STA PCL,X
FE7C: CA  DEX
FE7D: 10 F9  FPL A1PCLP
FE7F: 60  RTS
FE80: A0 3F  A1PCRTS  RTS
FE81: D0 08  SETINV  LDY #3P  SET FOR INVERSE VIO
FE82: DD 02  NBE SETINVL VIA COUT
FE84: D0 FF  SETNORM  LDX #5FF  SET FOR NORMAL VIO
FE86: 84 32  SETINVL  STY INY
FE88: 60  RTS
FE89: A9 00  SETKBD  LDA #500 SIMULATE PORT #0 INPUT
FE8B: B5 3E  IMPORT  STA A2L SPECIFIED (KEYIN ROUTINE)
FE8D: A2 38  INPT  LDX #55L
FE8F: 0A 1B  LDI #keyw
FE91: D0 08  BNE TOPK
FE93: A9 00  SETVID  LDA #500 SIMULATE PORT #0 OUTPUT
FE95: 85 3E  OUTPORT  STA A2L SPECIFIED (COUT1 ROUTINE)
FE97: A2 36  OUTPORT  LDX #5CWL
FE99: A0 FF  LDY #TOP1
FE9B: A5 3E  IOPRT  LDA A2L SET PAM IN/OUT VECTORS
FE9D: 29 0F  AND #5DF
FE9F: F0 06  BNE TOPR1
FEA1: 09 C0  GBA #IADAP/256
FEA3: A0 00  LDY #500
FEA5: F0 02  BNE TOPR2
FEA7: A9 FD  IOPRT1 LDA #COUT1/256
FEA9: 00 00  LDX LOC0,X
FEAB: 95 01  STA LOC1,X
FEAD: 60  RTS
FEAE: EA  NOP
FEAF: EA  NOP
FEB0: C0 00 ED  KBASE  NBP BASIC TO BASIC WITH SCRATCH
FEB3: 4C 03  ED  BASCOMP  NBP BASIC2 TO BASIC WITH SCRATCH

Source: David T Craig
FE86: 20 75 FE GO JSR A1PC ADD TO PC IF SPEC'D
FE89: 20 3F FF JSP RESTORE RESTORE META REGS
FEBC: 6C 3A 00 JRP (FLL) GO TO USER FROM
FEBD: 97 07 FA RGP1 JMP REDRISP TO REG DISPLAY
FEC2: C6 34 TSPCE DEC YSAV
FED2: 20 75 FE STEPF JSR A1PC ADD TO PC IF SPEC'D
FED7: 4C 63 FA RGP1 JMP RESTORE RESTORE TEMP
FECA: 4F 03 USP JSR USRHR AT USRBsr
FEOC: A9 40 WRITE LDA #$40
FEFD: 20 29 PC JSR HEADR WRITE 10-SEC HEADER
FED2: A9 27 LDY #$27
FED4: A2 00 W1 LDX #$00
FED6: 41 3C EOR (A1, X)
FED8: 48 PLA
FEDA: A1 3C LDA (A1, X)
FED8: 20 ED FE JSR #=YIFE
FEED: 20 3A FC JSR #YXAT
FEEL: A0 1D LDY #$1D
FEEM: 68 PLA
FEF0: 00 90 LCC $=1
FEF1: A0 22 LDY #$22
FEF3: 20 ED FE JSR WROYTE
FEF5: F0 40 BCC FLL
FEFD: A2 10 WRBYTE LDY #$10
FEFE: 0A WRYT2 ASL A
FEF0: 20 D6 FC JSR #RBIT
FEF3: D0 FA JNE #R2T2
FEF5: 60 RTS
FEF6: 20 00 FC CRMON JSR #L1 HANDLE CR AS BLANK
FEF9: 68 PLA THEN POP STACK
FEFA: 68 PLA AND RTN TO MON
FEFB: D0 6C JSR #MON
FEFC: 20 FA FC READ JSR #READ1 FIND TAPEPIN EDGE
FEFD: A9 16 LDA #$16
FEF0: 20 C9 FC JSR HEADR D=FAV 3.5 SECONDS
FEF5: 85 2E STA CHSUM INIT CHSUM=5FF
FEF6: 20 3A FC JSR #READ1 FIND TAPEPIN EDGE
FEF7: A0 4D RD2 LDY #$24 LOOK FOR SYNC BIT
FEF8: 20 FD FC JSR #RBIT (SHORT 0)
FEF9: B0 #F LOOP UNTIL FOUND
FEFA: 20 FC FC JSR #READ1 SKIP SECOND SYNC H-CYCLE
FF14: A0 3B LDY #$3B INDEX FOR 0/1 TEST
FF16: 20 EC FC #R3 JSR #RDYFE READ A BYTE
FF19: 01 40 STA (A1, X) SQUARE AT (A1)
FF1B: 45 2E STA CHSUM
FF1C: 85 2E STA CHSUM UPDATE RUNNING CHSUM
FF1D: 20 42 FC JSR #XAT1 INCR A1, COMPARE TO A2
FF20: A0 3D LDY #$3D COMPARE CHSUM TO CHSUM
FF24: 90 F0 ECC RD1 LOOP UNTIL DONE
FF26: 20 EC FC JSR #RDYTE READ CHSUM 1 BYTE
FF29: C5 2E CMP CHSUM
FF2C: 00 0D JSR #S RELL GOOD, SOUND BELL AND RETURN
FF2D: A9 C5 JSR #MPPERR LDA #$C5 PRINT "ERP", THEN BELL
FF2F: A9 01 JSR #CONT LDA #$D1 PRINT "ERP", THEN BELL
FF32: A9 02 JSR #D2 LDA #$D2 PRINT "ERP", THEN BELL
FF34: 20 ED FC JSR #CONT PRINT "ERP", THEN BELL
FF37: 20 ED FC JSR #CONT PRINT "ERP", THEN BELL
FF3A: A9 87 JSR #SEL LDA #$E7 OUTPUT BELL AND RETURN
FF3C: 4C ED FC JSR #SEL LDA #$E7 OUTPUT BELL AND RETURN
FF3F: A5 4B JSR #RESTORE LDA STATUS RESTORE 6502 REG CONTENTS
FF41: 48 PLA
FF42: A5 45 LDA ACC
FF44: A6 46 #REST1 LEX KREG
FF46: A4 47 LDX #REG
FF48: 28 PLP
FF49: 60 RTS
FF4A: 85 45 STA ACC SAVE 6502 #R CONTENTS
FF4C: 86 46 STA ACC SAVE 6502 #R CONTENTS
FF4E: 84 47 STY #REG
FF50: 08 PHP
FF51: 68 PLA
FF52: 85 4B STA STATUS
FF54: 0A TXA
FF55: 86 49 STA SXIT
FF57: 08 CLD
FF58: 60 RTS
FF59: 20 45 PE RST JSR #SETUPA SET SCREEN MODE
FF5C: 20 2F FB JSR #INIT AND INIT KB/SCREEN
FF5D: 20 93 FE JSR #INIT SETVID A S1/O DEV'S
FF62: 20 69 FE JSR #SETUPB
FF65: 08 MON CLD MUST SET H6 MODE1
FF66: 20 3A FF JSR #S RELL
FF69: A9 AA MONZ LDA #$AA ** PROMPT FOR MON
FF6B: 85 33 STA PROMPT
FF6D: 20 67 FD JSR #GETLINE READ A LINE

88

"DTCA2DOC-469-088.PICT" 415 KB 2001-06-26 dpi: 800 h x 800 v pix: 2531 h x 5391 v
**APPLE-II MINI-ASSEMBLER**

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

**FORMAT** EP2 52F
**LENGTH** EP2 52F
**MODE** EP2 531
**PREFIX** EP2 533
**SVAW** EP2 534
**L** EP2 535
**PCL** EP2 53A
**PCH** EP2 53B
**ISA** EP2 53C
**ISB** EP2 53F
**ISA** EP2 542
**ISB** EP2 543
**PMT** EP2 544
**IN** EQU $200
**INSDS2** EQU $F662
**INSTDF** EQU $F600
**PRL2** EQU $F94A
**PCL** EQU $F953
**CHAR1** EQU $F964
**CHAR2** EQU $F99A
**MLNG** EQU $F9C0
**MAEXP** EQU $FA0D
**CURRDFU** EQU $FCA1
**GETRAM** EQU $F067
**COUT** EQU $F060
**AL** EQU $F000
**AL** EQU $FE76
**FELL** EQU $FE3A
**GETNUM** EQU $FFA7
**TÜS** EQU $FFP7
**Z-ODE** EQU $FPC7
**CPF13** EQU $FPPC
**QRC** EQU $F500

```assembly
F500:   84 81   REL   SBC  #R1  IS NOT COMPATIBLE
F502:   4A   LSP  A  WITH RELATIVE MODE?  NO.
F503:   D0 14   CALL  $F93
F505:   A4 3F   LNY  A2:   LFX  A2L  DOUBLE DECREMENT
F507:   A6 3F   BNE  REL2
F509:   D0 01   DEF
F50B:   88   DEF
F50C:   CA   REL2   DEX
F50D:   8A   TAX
F50E:   18   CLC
F50F:   E5 3A   SBC  PCL  FORM ADDR-PC-2
F511:   B5 3E   STA  A2L
F513:   10 01   BPL  REL3
F515:   C8   INX
F516:   98   REL3   TYA
```

91

F517: E5 3B SBC PCH
F519: D0 6B ERR3 BNE ERR ERROR IF >1-BYTE BRANCH
F51B: A4 2F FINDOP LOY LENGTH
F51D: B9 3D 00 FNXOP2 LOA A1,Y MOVE INST TO (PC)
F520: 91 3A STA (PCL),Y
F522: 88 DEY
F523: 10 FB BPL PXNOP2
F525: 20 1A FC JSR CURSUP
F528: 20 1A FC JSR CURSUP RESTORE CURSOR
F52B: 20 DB 0F JSR INSTDSP TYPE FORMATTED LINE
F52E: 20 53 F9 JSR PCADJ UPDATE PC
F531: 84 3B STY PCH
F533: 85 3A STA PCL
F535: 4C 95 F5 JMP NXTLINE GET NEXT LINE
F538: B0 8E FF FAKEMON JSR TOSUB GO TO DELIM HANDLER
F53B: A4 34 LDY Y3AV RESTORE Y-INDEX
F53D: 20 A7 FF FAKEMON JSR GETNUM READ PARAM
F540: 8A 34 STY Y3AV SAVE Y-INDEX
F542: A0 17 LDY #$17 INIT DELIMITER INDEX
F544: 88 FAKEMON2 DEY CHECK NEXT DELIM
F545: 30 4B BNE RESET2 ERR IF UNRECOGNIZED DELIM
F547: D9 CC FF CMP CHRTBL,Y COMPARE WITH DELIM TABLE
F54A: D0 F8 BNE FAKEMON2 NO MATCH
F54C: C0 15 CPY #$15 MATCH, IS IT CR?
F54E: D0 E0 BNE FAKEMON3 NO, HANDLE IT IN MONITOR
F550: A5 31 LDA MODE
F552: A0 00 LDY #$0 HANDLE CR OUTSIDE MONITOR
F555: C4 34 SEC Y3AV
F556: 20 00 FE JSR BL1 HANDLE CR OUTSIDE MONITOR
F559: 4C 95 F5 JMP NXTLINE GET TRIAL OPCODE
F55C: A5 3D TRYNEXT LDA A1H GET TRIAL OPCODE
F55E: B0 8E FF JSR INSDB2 GET FMT+LENGTH FOR OPCODE
F561: AA TAX
F562: BD 00 FA LDA MNEMBX GET LOWER MNEMONIC BYTE
F565: C5 42 CMP A4L MATCH?
F567: D0 13 BNE NEXTOP NO, TRY NEXT OPCODE
F569: BD C0 F9 LDA MNEMBX GET UPPER MNEMONIC BYTE
F56C: C5 43 CMP A4H MATCH?
F56E: BD 0C BNE NEXTOP NO, TRY NEXT OPCODE.
F570: A5 44 LDA FMT
F572: A4 2E LDY FORMAT GET TRIAL FORMAT
F574: C0 9D CPY #$9D TRIAL FORMAT RELATIVE?
F576: F0 88 BSE REL YES.
F578: C5 2E NREL CMP FORMAT SAME FORMAT?
F57A: F0 9F BEO FINDOP YES.
F57C: C6 3D NEXTOP DEC A1H NO, TRY NEXT OPCODE
F57E: D0 DC BNE TRYNEXT
F580: E6 44 INC FMT NO MORE, TRY WITH LEN=2
F582: C6 35 DEC L WAS L=2 ALREADY?
F584: F0 D6 BEO TRYNEXT NO.
F586: A4 34 ERR LDY Y3AV YES, UNRECOGNIZED INST.
F588: 9B ERR2 TYA
F589: AA TAX
F58A: 20 4A F9 JSR PRBL2 PRINT "UNDER LAST READ
F58D: 09 4E LDA $DE CHAR TO INDICATE ERROR
F58F: 20 ED FD JSR COUT POSITION.
F592: 20 3A FF RESET2 JSR BELL
F595: A9 01 NXTLINE LDA #$A1 '1'
F597: B5 33 STA PROMPT INITIALIZE PROMPT
F599: 20 67 FD JSR GETLM2 GET LINE.
F59C: 20 C7 FF JSR SMODE INIT SCREEN STUFF
F59F: AD 00 02 LDA IN GET CHAR
F5A2: C9 AO CMP #$A0 ASCII BLANK?
F5A4: F0 13 * RLO SPACE YES
F5A6: C8 INY
F5A7: C9 A4 CMP #$A4 ASCII 'S' IN COL 17
F5A9: 89 3A REO PAREM YES, SIMULATE MONITOR
F5AB: 88 DEY NO, BACKUP A CHAR
F5AC: 20 A7 FF JSR GETNUM GET A NUMBER
F5AF: C9 33 CMP #$93 '1' TERMINATOR?
F5B1: D0 05 ERR4 BJR #$F2 NO, ERR.
F5B3: 8A TXA
F5B4: F0 02 BEO ERR2 NO ADR PRECEDING COLON.
F5B6: 20 78 FE JSR A1CPLF MOVE ADR TO PCL, PCH.
F5BA: AD 03 SPACE LDA #$3 COUNT OF CHAR IN MNEMONIC
F5BD: B5 3D STA A1H
F5BF: 20 34 F6 NXTMN JSR GETNSP GET FIRST MNEMONIC CHAR.
F5C0: 0A NXTM ASL A
F5C1: E9 BE BSC #$BE SUBTRACT OFFSET
F5C3: C5 C2 CMP #$C2 LEGAL CHAR?
F5C5: 90 C1 BCC ERR2 NO.
F5C7: 0A ASL A COMPRESS-LEFT JUSTIFY
F5CB: 0A ASL A
F5C9: A2 04 LDX #$4 DO 5 TRIPLE WORD SHIFTS

92

"DTCA2DOC-469-092.PICT" 470 KB 2001-06-26 dpi: 800h x 800v pix: 2487h x 5412v
Source: David T Craig
TITLE "FLOATING POINT ROUTINES"
SIGN EPZ $F3
X2 EPZ $F4
M2 EPZ $F5
X1 EPZ $F6
M1 EPZ $F7
E EPZ $F8
OVL0C EQU $3F5
OPG $F425

ADD CLC CLEAR CARRY.
LDX #12 IMUX FOR 3-BYTE ADD.
LDA #1,X AND A BYTE OF MANT2 TO MANT1.
ADC M2,X INDEX TO NEXT SIGNIFICANT BYTE.
STA #1,X
DPL ADD1 LOOP UNTIL DONE.
RTS RETURN

MUL ASL SIGN CLEAR LSB OF SIGN.
JSR ARS/AP ARS VAL OF M1, THEN SNAP WITH M2
BIT #1 MANT1 NEGATIVE?
LASHA P1 NO, SNAP WITH MANT2 AND RETURN.
JSR FCCMP YES, COMPLEMENT IT.
INC SIGN LASC SIGN, COMPLEMENTING LSB.
SEC SET CARRY FOR RETURN TO H/DIV.

SNAP LDX #$F4 INDEX FOR 4-BYTE SNAP.
STY #1,X
LDA X1-1,X SWAP A BYTE OF EXP/MANT1 WITH
LDY X2-1,X EXP/MANT2 AND LEAVE A COPY OF
MANT1 IN E (3 BYTES). IF+3 USED
STX X1-1,X

DFX ADVANCE INDEX TO NEXT BYTE.

SNAP1 LDX #$F4 LOOP UNTIL DONE.

RTS RETURN

LDA #$FEE INIT EXP1 TO 14,
STA X1 THEN NORMALIZE TO FLOAT.

LDA #1,F5 HIGH-GORDER MANIT BYTE.
CMP #$C0 UPPER TWO BITS UNEQUAL?
RTS PLS1 YES, RETURN WITH "MANT1 NORMALIZED
DEC X1 DECREMENT EXP1.

ASL M+2 SHIFT MANT1 (3 BYTES) LEFT.

ROL M1

LDA X1 EXP1 ZERO?
BNE NORM1 NO, CONTINUE NORMALIZATION.

RTS RETURN.

CMPL MANT1, CLEARS CARRY UNLESS 0
PCOMPL

JSR ALGNSWP RIGHT SHIFT MANT1 OR SWAP WITH
LDA X2

CMP X' COMPARE EXP1 WITH EXP2.

BNE SMPSGN IF , SNAP ADDENDS OR ALIGN MANTS.

JSR ADD AOE ALIGNED MANTISSAS.

BVC NORM NO OVERFLOW, NORMALIZE RESULT.

BVS RLOG OV: SHIFT $1 RIGHT, CARRY INTO SIGN.
***************
* APPLE-II PSEUDO *
* MACHINE INTERPRETER *
* COPYRIGHT 1977 *
* APPLE COMPUTER INC *
* ALL RIGHTS RESERVED *
* S. WOZNIAK *
* ***************

TITLE: "SWEET16 INTERPRETER"

```
R0L EPI $0
R0H EPI $1
R14H EPI $10
R15L EPI $1E
R15H EPI $1F
S16PA G EQU $F7
SAVE EQU SFF#A
RESTORE EQU SFF3F
ORG $669

F689: 20 4A FF S:16 JSR SAVE PRESERVE 6502 REG CONTENTS
F68C: 68 PLA
F68D: 85 1E STA R15L INIT SWEET16 PC
F68E: 68 PLA FROM RETURN
F690: 85 1F STA R15H ADDRESS
F691: 20 98 F6 $16B JSP S16C INTERPRET AND EXECUTE
F692: 4C 92 F6 JMP S16F ONE SWEET16 INSTR.
F693: 66 1E S16C INC R15L
F694: D0 02 BNE S16D INC SWEET16 PC FOR FETCH
F695: E6 1F INC R15H
F696: AA 0F S16D LDA S16PA PUSH ON STACK FOR PTS
F697: 4B PHA
F698: A0 00 LDD $0
F699: B1 1E LDA (R15L),Y FETCH INSTR
F69A: 29 0F LDA (R15L),Y, F MASK REG SPECIFICATION
F69B: 0A ASL A DOUBLE FOR 2-BYTE REGISTERS
F69C: AA TAK TO X-REG FOR INDEXING
F69D: 4A LSR A
F69E: 51 1E EOR (R15L),Y NOW HAVE OPCODE
F69F: 00 BF BEO TWR IF ZERO THEN NON-REG OP
F6A0: 86 1D STX R14H INDICATE 'PRIOR RESULT REG'
F6A1: 4A LSR A
F6A2: 4A LSR A OPCODE*2 TO LSP'S
F6A3: 4A LSR A
F6A4: 84 F1 F6 LDA OPTB-2,Y LOW-ORDER ADDR BYTE
F6A5: 48 PHA ONTO STACK
F6A6: 60 PTS GOTO REG-OP ROUTINE
F6A7: E6 EE TCHP INC R15L
F6A8: D0 02 BNE R0H*2 INCPC
F6A9: E6 1F INC R15H
F6AA: 4D F4 F6 TCAK2 LDA SFRAL,X LOW-ORDER ADP BYTE
F6AB: 48 PHA ONTO STACK FOR NON-RPG OP
F6AC: A5 1D LDA R14H 'PRIOR RESULT REG' INDEX
F6AD: 4A LSR A PREPARE CARRY FOR RC, BNC.
F6AE: 60 PTS GOTO NON-REG OP ROUTINE
F6AF: 68 RTNZ PLA JP RETURN ADDRESS
F6B0: 68 PLA
F6B1: 20 3F FF JSR RESTORE RESTORE 6502 REG CONTENTS
F6B2: 6C 1E 00 JMP (R15L) RETURN TO 6502 CODE VIA PC
F6B3: 81 1F SETZ LDA (R15L),Y HIGH-ORDER BYTE OF CONSTANT
```

96

Source: David T Craig
```
F601:  95 01 STA RH,X
F603:  8B DEY
F604:  81 EE LDA (D15L),Y LOW-ORDER BYTE OF CONSTANT
F606:  90 00 STA RL1,X
F608:  90 00 TYA Y-REG CONTAINS 1
F60A:  38 SEC
F60C:  65 EE ADC $15L ADD 2 TO PC
F60E:  9E 00 STA $15L
F610:  90 02 BCC SFT2
F612:  E6 01 INC $15H
F614:  02 SFT2 RTS
F616:  F9 PTRN DPH SFT-1 (1X)
F618:  04 DPH LD-1 (2X)
F61A:  90 DEY
F61C:  0D DPP ST-1 (0)
F61E:  9E DB RNC-1 (2)
F620:  25 DB JMPAT-1 (4X)
F622:  A6 DB TPC-1 (2X)
F624:  16 DB SFTAT-1 (5X)
F626:  82 DB SP-1 (4)
F628:  47 DB LDDAT-1 (6X)
F62A:  89 DB $NH-1 (5)
F62C:  51 DB SFTAD-1 (7X)
F62E:  C0 DB RP-1 (6)
F630:  2F DB POPL-1 (6X)
F632:  C9 DB ADD-1 (AX)
F634:  5B DB SRM-1 (7X)
F636:  6E DB SU3-1 (RX)
F638:  05 DB AD-1 (AX)
F63A:  33 DB SQM-1 (9)
F63C:  8B DB POPQ-1 (CX)
F63E:  EB DB PS-1 (8)
F640:  70 DB CPR-1 (D)
F642:  33 DB BS-1 (C)
F644:  1E DB INR-1 (F)
F646:  67 DB NUL-1 (0)
F648:  65 DB DCR-1 (FX)
F64A:  67 DB NUL-1 (E)
F64C:  67 DB NUL-1 (UNUSE)
F64E:  DD DB NUL-1 (0X)
F703:  10 CA SET BPL SETS ALWAYS TAKEN
F705:  B5 00 LDA ROL,X
F707:  00 LD RX TO RO
F709:  B5 01 STA ROL
F70B:  B5 01 STA ROL
F70D:  60 RTS
F70F:  A5 00 ST LDA ROL
F711:  95 00 STA ROL,X MOVE RX TO RX
F713:  A5 01 STA ROL
F715:  95 01 STA ROL,X
F717:  60 RTS
F719:  A5 00 STAT LDA ROL
F71B:  81 00 STAT2 (ROL,X) STORE BYTE INDIRECT
F71D:  AA 00 LDA $50
F71F:  4D 10 STAT3 STY $14H INDICATE RO IS RESULT REG
F721:  F6 00 INC ROL,X
F723:  00 02 BNE INR2 INC R RX
F725:  F6 01 INC ROL,X
F727:  60 RTS
F729:  A1 00 LOAD LDA (ROL,X) LOAD INDIRECT (RX)
F72B:  00 STA ROL 10 RO
F72D:  00 LDY $50
F72F:  84 01 STY ROL ZERO HIGH-ORDER PO BYTE
F731:  F0 ED BEO STAD3 ALWAYS TAKEN
F733:  A0 00 POP LDY $50 HIGH ORDER BYTE = 0
F735:  F0 06 BEO POP2 ALWAYS TAKEN
F737:  20 66 F7 POPD JSR DCR DCR RX
F739:  A0 00 POP LDA (ROL,X) POP HIGH-ORDER BYTE #RX
F73B:  48 A0 TAY SAVE $M-REG
F73D:  20 66 F7 POP2 JSR DCP DCR RX
F73F:  A0 00 LDA (ROL,X) LOW-ORDER BYTE
F741:  85 00 STA ROL TO RX
F743:  84 01 STY ROL
F745:  00 POP3 LDY $50 INDICATE RO AS LAST RSLT REG
F747:  60 RTS
F749:  20 26 F7 LDDAT JSR LDDAT LOW-ORDER BYTE TO RX, INC R RX
F74B:  A1 00 LDA (ROL,X) HIGH-ORDER BYTE TO RX
F74D:  85 01 STA ROL
F74F:  4C 1F F7 JMP INR INCR RX
F751:  20 17 F7 STAD JSR STAT STOPE INDIRECT LOW-ORDER
```

Source: David T Craig
F755: A5 01  LDA  R0H  BYTE AND INCR RX, THEN
F757: B1 00  STA  (#L,RX)  STORE HIGH-ORDER BYTE.
F759: 4C 1F  F7  JMP  #M  INCX #2 AND RETURN
F75C: 20 66  F7  JSR  HCF  DECX RX
F75F: A5 00  LDA  R0L  ROU
F761: 01 00  STA  (#L,RX)  STORE 0D LOW BYTE RX
F763: 4C 43  F7  JMP  #PO3  INCREASE PO AS LAST RS1 REG
F766: B5 05  DCY  STA  (#L,RX)  LDA  ROO,RX
F76B: D0 02  DCR  STA  (#L,RX)  LDA  ROY
F76A: D6 01  DEC  R0H,RX
F76C: D6 00  DCR  #R0L,RX
F76E: 60  RTS
F770: A0 00  SBC  #00  LDX  $F0  DFAULT TO RO
F771: 1B  CPI  SBC  #00  NOTE: Y-REG = 13*2 FOR CPR
F772: A5 00  LDA  ROH  ROU
F774: F5 00  SBC  R0L,RX  RO-RX TO FY
F776: 99 00  00  STA  ROL,Y  RO-RX TO FY
F77A: A5 01  LDA  ROH
F77B: F5 01  SBC  ROH,RX
F77D: 99 01  00  SBC  ROL,Y
F780: 98  TAC  LAST RESULT REG*2
F781: 69 00  ADC  #00  CARRY TO LS1B
F783: 85 1D  STA  R14H
F785: 60  RTS
F786: A5 00  ADD  R0H  ROU
F788: 75 00  ADC  R0L,RX  RO+RX TO RO
F78A: 85 00  STA  ROL,RX  RO+RX TO RO
F78C: A5 01  LDA  ROH
F78E: 75 01  ADC  ROL,RX  RO+RX TO RO
F790: A0 00  LDI  #00  RO FOR RESULT
F792: F0 E9  BEO  #S2  FINISH ADD
F794: A5 1E  BS  LDA  R15L  NOTE X-REG IS 12*21
F796: 20 19  F7  JSR  STAT2  PUSH LOW PC BYTE VIA R12
F798: A5 1F  LDA  R15H
F79B: 20 19  F7  JSR  STAT2  PUSH HIGH-ORDER PC BYTE
F79F: 18  BR  SBC  BCS  NO CARRY TEST
F7A1: 81 1E  BPL  R15H  DISPLACEMENT BYTE
F7A3: 10 01  BPL  R12
F7A5: 6E  DEY
F7A6: 65 1E  BR2  ADD TO PC
F7A8: 85 1E  STA  R15L
F7AA: 90  TYA
F7AB: 65 1F  ADC  R15H
F7AD: 85 1F  STA  R15H
F7AF: 60  RTS
F7B0: 80  EC  SBC  SBC
F7B2: 60  RTS
F7B3: 0A  AP  ASL  @  LOUOLE RESULT-REG INDEX
F7B4: AA  TAX  TO X-REG FOR INDEXING
F7B5: 85 01  LDA  #ROH,X  TEST FOR PLUS
F7B7: 10 88  BPL  #61  PRANCH IF SO
F7B9: 60  RTS
F7BA: 0A  ASL  A  DOUBLE RESULT-REG INDEX
F7BB: AA  TAX  TO X-REG FOR MINUS
F7BC: B5 01  LDA  #ROD,X  TEST FOR MINUS
F7BE: 30 E1  BMI  #16
F7CF: 60  RTS
F7C1: 0A  B7  ASL  A  DOUBLE RESULT-REG INDEX
F7C4: AA  TAX
F7C5: D5 00  ICA  R0L,X  TEST FOR ZERO
F7C7: 15 01  ORA  ROH,X  (BOTH PYFES)
F7C9: D0 08  EOR  #V1  PRANCH IF SO
F7CA: 60  RTS
F7CC: B5 00  LDA  R0L,X  TEST FOR #ZERO
F7CE: 15 01  ORA  ROH,X  (BOTH BYTES)
F7DF: D0 CF  BNE  #BR1  BRANCH IF SO
F7D2: 60  RTS
F7D3: 0A  BM1  ASL  A  DOUBLE RESULT-REG INDEX
F7D4: AA  TAX
F7D5: B5 00  LDA  R0L,X  CHECK BOTH BYTES
F7D7: D5 01  AND  ROH,X  FOR #FF (MINUS 1)
F7D9: 4F FF  EOR  #$FF
F7DB: F0 C4  EOR  ROH  BRANCH IF SO
F7DD: 60  RTS
F7DE: 0A  BM1  ASL  A  DOUBLE RESULT-REG INDEX
F7DF: AA  TAX
F7E0: B5 00  LDA  R0L,X
F7E2: 35 01  AND  ROH,X  CHECK BOTH BYTES FOR NO #FF
F7E4: 49 FF  EOR  #$FF
F7E6: DD 09  BNE  #V1  BRANCH IF NOT MINUS 1
F7E8: 60  RTS
F7EB: A2 1B  RS  LDX  #F1A  12*2 FOR R12 AS STK POINTER

Source: David T Craig
F7EB: 20 66 F7  JSR DCR  DECR STACK POINTER
F7EE: A1 00  LDA (R1L,X) POP HIGH RETURNADR TO PC
F7F0: 85 1F  STA R1SP
F7F2: 20 66 F7  JSP DCR  SAME FOR LOW-ORDER BYTE
F7F5: A1 00  LDA (R1L,X)
F7F7: 85 1E  STA R1SL
F7F9: 60  RTS
F7FA: 4C C7 F6 RTN  JMP RTN2
### 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>CLV</td>
<td>Clear Overflow Flag</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
C  Borrow
P  Processor Status Register
S  Stack Pointer
+  Add
·  Logical AND
-  Subtract
\(\lor\)  Logical Exclusive Or
\(\lnot\)  Transfer From Stack
\(\rightarrow\)  Transfer To Stack
\(\leftarrow\)  Transfer To
\(\leftarrow\)  Transfer To
\(\lor\)  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

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

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

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

FIGURE 3.

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

NOTE 1: BIT — TEST BITS

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

PROGRAMMING MODEL

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

ACCUMULATOR
INDEX REGISTER Y
INDEX REGISTER X
PROGRAM COUNTER
STACK POINTER

PROCESSOR STATUS REGISTER, "P"

\[
\begin{array}{cccccccc}
&  &  &  &  &  &  & 1 \\
N & V & B & D & I & Z & C \\
\end{array}
\]

CARRY
ZERO
INTERRUPT DISABLE
DECIMAL MODE
BREAK COMMAND
OVERFLOW
NEGATIVE

101
### INSTRUCTION CODES

<table>
<thead>
<tr>
<th>Name Description</th>
<th>Operation</th>
<th>Addressing Mode</th>
<th>Assembly Language Form</th>
<th>HEX OP Code</th>
<th>No. Bytes</th>
<th>&quot;F&quot; Status Reg. N Z C D V</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC</td>
<td>A M+ C → A C</td>
<td>Immediate</td>
<td>ADC: Oper</td>
<td>69</td>
<td>2</td>
<td>(\checkmark)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Zero Page</td>
<td>ADC: Oper</td>
<td>65</td>
<td>2</td>
<td>(\checkmark)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Absolute</td>
<td>ADC: Oper</td>
<td>63</td>
<td>3</td>
<td>(\checkmark)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Absolute X</td>
<td>ADC: Oper</td>
<td>70</td>
<td>3</td>
<td>(\checkmark)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Absolute Y</td>
<td>ADC: Oper</td>
<td>64</td>
<td>3</td>
<td>(\checkmark)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(indirect X)</td>
<td>ADC: Oper</td>
<td>61</td>
<td>2</td>
<td>(\checkmark)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(indirect Y)</td>
<td>ADC: Oper</td>
<td>71</td>
<td>2</td>
<td>(\checkmark)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name Description</th>
<th>Operation</th>
<th>Addressing Mode</th>
<th>Assembly Language Form</th>
<th>HEX OP Code</th>
<th>No. Bytes</th>
<th>&quot;F&quot; Status Reg. N Z C D V</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND</td>
<td>A M → A</td>
<td>Immediate</td>
<td>AND: Oper</td>
<td>29</td>
<td>2</td>
<td>(\checkmark)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Zero Page</td>
<td>AND: Oper</td>
<td>25</td>
<td>2</td>
<td>(\checkmark)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Absolute</td>
<td>AND: Oper</td>
<td>20</td>
<td>3</td>
<td>(\checkmark)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Absolute X</td>
<td>AND: Oper</td>
<td>30</td>
<td>3</td>
<td>(\checkmark)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Absolute Y</td>
<td>AND: Oper</td>
<td>36</td>
<td>3</td>
<td>(\checkmark)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(indirect X)</td>
<td>AND: Oper</td>
<td>21</td>
<td>2</td>
<td>(\checkmark)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(indirect Y)</td>
<td>AND: Oper</td>
<td>31</td>
<td>2</td>
<td>(\checkmark)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name Description</th>
<th>Operation</th>
<th>Addressing Mode</th>
<th>Assembly Language Form</th>
<th>HEX OP Code</th>
<th>No. Bytes</th>
<th>&quot;F&quot; Status Reg. N Z C D V</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCC</td>
<td>Branch on carry clear</td>
<td>Branch on C=0</td>
<td>BCC: Oper</td>
<td>90</td>
<td>2</td>
<td>(\checkmark)</td>
</tr>
<tr>
<td>BCS</td>
<td>Branch on carry set</td>
<td>Branch on C=1</td>
<td>BCS: Oper</td>
<td>80</td>
<td>2</td>
<td>(\checkmark)</td>
</tr>
<tr>
<td>BEQ</td>
<td>Branch on result zero</td>
<td>Branch on Z=0</td>
<td>BEQ: Oper</td>
<td>90</td>
<td>2</td>
<td>(\checkmark)</td>
</tr>
<tr>
<td>BIT</td>
<td>Test bits in memory with accumulator</td>
<td>A M X N → Y, NM → X</td>
<td>BIT: Oper</td>
<td>24</td>
<td>2</td>
<td>(\checkmark)</td>
</tr>
<tr>
<td>BMI</td>
<td>Branch on result minus</td>
<td>Branch on N=1</td>
<td>BMI: Oper</td>
<td>30</td>
<td>2</td>
<td>(\checkmark)</td>
</tr>
<tr>
<td>BNE</td>
<td>Branch on result not zero</td>
<td>Branch on Z=0</td>
<td>BNE: Oper</td>
<td>80</td>
<td>2</td>
<td>(\checkmark)</td>
</tr>
<tr>
<td>BPL</td>
<td>Branch on result plus</td>
<td>Branch on N=0</td>
<td>BPL: Oper</td>
<td>10</td>
<td>2</td>
<td>(\checkmark)</td>
</tr>
<tr>
<td>BRK</td>
<td>Force Break</td>
<td>Forced interrupt</td>
<td>BRK: Oper</td>
<td>00</td>
<td>1</td>
<td>(\checkmark)</td>
</tr>
<tr>
<td>BVC</td>
<td>Branch on overflow clear</td>
<td>Branch on V=0</td>
<td>BVC: Oper</td>
<td>50</td>
<td>2</td>
<td>(\checkmark)</td>
</tr>
<tr>
<td>BVS</td>
<td>Branch on overflow set</td>
<td>Branch on V=1</td>
<td>BVS: Oper</td>
<td>70</td>
<td>2</td>
<td>(\checkmark)</td>
</tr>
<tr>
<td>CMP</td>
<td>Compare memory and accumulator</td>
<td>A M</td>
<td>CMP: Oper</td>
<td>0</td>
<td>(\checkmark)</td>
<td></td>
</tr>
<tr>
<td>CPX</td>
<td>Compare memory and index X</td>
<td>X M</td>
<td>CPX: Oper</td>
<td>0</td>
<td>(\checkmark)</td>
<td></td>
</tr>
<tr>
<td>CPY</td>
<td>Compare memory and index Y</td>
<td>Y M</td>
<td>CPY: Oper</td>
<td>0</td>
<td>(\checkmark)</td>
<td></td>
</tr>
<tr>
<td>DEC</td>
<td>Decrement memory by one</td>
<td>M 1 M</td>
<td>DEC: Oper</td>
<td>0</td>
<td>(\checkmark)</td>
<td></td>
</tr>
<tr>
<td>DEX</td>
<td>Decrement index X by one</td>
<td>X 1 X</td>
<td>DEX: Oper</td>
<td>0</td>
<td>(\checkmark)</td>
<td></td>
</tr>
<tr>
<td>DEY</td>
<td>Decrement index Y by one</td>
<td>Y 1 Y</td>
<td>DEY: Oper</td>
<td>0</td>
<td>(\checkmark)</td>
<td></td>
</tr>
<tr>
<td>CLC</td>
<td>Clear carry flag</td>
<td>0 → C</td>
<td>CLC: Oper</td>
<td>0</td>
<td>(\checkmark)</td>
<td></td>
</tr>
<tr>
<td>CLI</td>
<td>Clear decimal mode</td>
<td>0 → D</td>
<td>CLI: Oper</td>
<td>0</td>
<td>(\checkmark)</td>
<td></td>
</tr>
<tr>
<td>CLV</td>
<td>Clear overflow flag</td>
<td>0 → V</td>
<td>CLV: Oper</td>
<td>0</td>
<td>(\checkmark)</td>
<td></td>
</tr>
<tr>
<td>CLD</td>
<td>Clear decimal mode</td>
<td>0 → D</td>
<td>CLD: Oper</td>
<td>0</td>
<td>(\checkmark)</td>
<td></td>
</tr>
</tbody>
</table>
# 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>Mnemonic</th>
<th>Bit Status Reg.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>EDR</strong></td>
<td>A V M → A</td>
<td>Immediate</td>
<td>EDR Oper</td>
<td>49 2</td>
<td></td>
<td>Z C I D Y</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Zero Page</td>
<td>EDR Oper</td>
<td>45 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Zero Page,X</td>
<td>EDR Oper,X</td>
<td>55 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Absolute</td>
<td>EDR Oper</td>
<td>4D 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Absolute,X</td>
<td>EDR Oper,X</td>
<td>5D 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Absolute,Y</td>
<td>EDR Oper,Y</td>
<td>59 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Indirect.X)</td>
<td>EDR Oper,X</td>
<td>41 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Indirect.Y)</td>
<td>EDR Oper,Y</td>
<td>51 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>INC</strong></td>
<td>M + 1 → M</td>
<td>Zero Page</td>
<td>INC Oper</td>
<td>6E 2</td>
<td></td>
<td>Z C I D Y</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Zero Page,X</td>
<td>INC Oper,X</td>
<td>6F 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Absolute</td>
<td>INC Oper</td>
<td>EE 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Absolute,X</td>
<td>INC Oper,X</td>
<td>FE 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>INX</strong></td>
<td>X + 1 → X</td>
<td>Implied</td>
<td>INX</td>
<td>E8 1</td>
<td></td>
<td>Z C I D Y</td>
</tr>
<tr>
<td><strong>INY</strong></td>
<td>Y + 1 → Y</td>
<td>Implied</td>
<td>INY</td>
<td>C8 1</td>
<td></td>
<td>Z C I D Y</td>
</tr>
<tr>
<td><strong>JMP</strong></td>
<td>(PC+1) → PCL</td>
<td>Absolute</td>
<td>JMP Oper</td>
<td>4C 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(PC-1) → PCL</td>
<td>Absolute</td>
<td>JMP Oper</td>
<td>6C 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>JSR</strong></td>
<td>(PC+1) → PCL</td>
<td>Absolute</td>
<td>JSR Oper</td>
<td>20 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(PC-1) → PCL</td>
<td>Absolute</td>
<td>JSR Oper</td>
<td>3F 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>LDA</strong></td>
<td>M → A</td>
<td>Immediate</td>
<td>LDA Oper</td>
<td>A9 2</td>
<td></td>
<td>Z C I D Y</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Zero Page</td>
<td>LDA Oper</td>
<td>A5 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Zero Page,X</td>
<td>LDA Oper,X</td>
<td>B5 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Absolute</td>
<td>LDA Oper</td>
<td>A0 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Absolute,X</td>
<td>LDA Oper,X</td>
<td>BD 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Absolute,Y</td>
<td>LDA Oper,Y</td>
<td>B9 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Indirect.X)</td>
<td>LDA Oper,X</td>
<td>A1 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Indirect.Y)</td>
<td>LDA Oper,Y</td>
<td>B1 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>LDX</strong></td>
<td>M → X</td>
<td>Immediate</td>
<td>LDX Oper</td>
<td>A2 2</td>
<td></td>
<td>Z C I D Y</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Zero Page</td>
<td>LDX Oper</td>
<td>A6 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Zero Page,X</td>
<td>LDX Oper,X</td>
<td>B6 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Absolute</td>
<td>LDX Oper</td>
<td>AE 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Absolute,X</td>
<td>LDX Oper,Y</td>
<td>BE 3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Instruction Details**

- **LSR**
  - Function: Shift right one bit (memory or accumulator)
  - Source: LSR Oper
  - Destination: LSR Oper

- **NOP**
  - Function: No operation
  - Source: NOP

- **ORA**
  - Function: OR memory with accumulator
  - Source: ORA Oper

- **PHA**
  - Function: Push accumulator on stack
  - Source: PHA

- **PLA**
  - Function: Pull accumulator from stack
  - Source: PLA

- **ROL**
  - Function: Rotate one bit left (memory or accumulator)
  - Source: ROL Oper

- **ROR**
  - Function: Rotate one bit right (memory or accumulator)
  - Source: ROR Oper
## INSTRUCTION CODES

<table>
<thead>
<tr>
<th>Name Description</th>
<th>Operation</th>
<th>Addressing Mode</th>
<th>Assembly Language Form</th>
<th>HEX OP Code</th>
<th>No. Bytes</th>
<th>&quot;F&quot; Status Reg.</th>
<th>Z C I D V</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTI Return from interrupt</td>
<td>P+PCi</td>
<td>Implied</td>
<td>RTI</td>
<td>40</td>
<td>1</td>
<td>From Stack</td>
<td></td>
</tr>
<tr>
<td>RTS Return from subroutine</td>
<td>PC+, PC+1 → PC</td>
<td>Implied</td>
<td>RTS</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SBC Subtract memory from accumulator with borrow</td>
<td>A - M - C → A</td>
<td>Immediate</td>
<td>SBC #Oper</td>
<td>E9</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Zero Page</td>
<td>SBC Oper</td>
<td>E5</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Zero Page,X</td>
<td>SBC Oper,X</td>
<td>F5</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Absolute</td>
<td>SBC Oper</td>
<td>E0</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Absolute,X</td>
<td>SBC Oper,X</td>
<td>F0</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Absolute,Y</td>
<td>SBC Oper,Y</td>
<td>F9</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Indirect,X)</td>
<td>SBC (Oper,X)</td>
<td>E1</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Indirect,Y)</td>
<td>SBC (Oper,Y)</td>
<td>F1</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEC Set carry flag</td>
<td>1 → C</td>
<td>Implied</td>
<td>SEC</td>
<td>3B</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SED Set decimal mode</td>
<td>1 → D</td>
<td>Implied</td>
<td>SED</td>
<td>F6</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEI Set interrupt disable status</td>
<td>1 → I</td>
<td>Implied</td>
<td>SEI</td>
<td>7B</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STA Store accumulator in memory</td>
<td>A → M</td>
<td>Zero Page</td>
<td>STA Oper</td>
<td>85</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Zero Page,X</td>
<td>STA Oper,X</td>
<td>95</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Absolute</td>
<td>STA Oper</td>
<td>80</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Absolute,X</td>
<td>STA Oper,X</td>
<td>90</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Absolute,Y</td>
<td>STA Oper,Y</td>
<td>99</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Indirect,X)</td>
<td>STA (Oper,X)</td>
<td>81</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Indirect,Y)</td>
<td>STA (Oper,Y)</td>
<td>91</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STX Store index X in memory</td>
<td>X → M</td>
<td>Zero Page</td>
<td>STX Oper</td>
<td>86</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Zero Page,Y</td>
<td>STX Oper,Y</td>
<td>96</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Absolute</td>
<td>STX Oper</td>
<td>98</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STY Store index Y in memory</td>
<td>Y → M</td>
<td>Zero Page</td>
<td>STY Oper</td>
<td>84</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Zero Page,X</td>
<td>STY Oper,X</td>
<td>94</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Absolute</td>
<td>STY Oper</td>
<td>8C</td>
<td>3</td>
<td></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>
<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>
<td></td>
</tr>
<tr>
<td>TSX Transfer stack pointer to index X</td>
<td>S → X</td>
<td>Implied</td>
<td>TSX</td>
<td>B8</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TXA Transfer index X to accumulator</td>
<td>X → A</td>
<td>Implied</td>
<td>TXA</td>
<td>8A</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TXS Transfer index X to stack pointer</td>
<td>X → S</td>
<td>Implied</td>
<td>TXS</td>
<td>9A</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TYA Transfer index Y to accumulator</td>
<td>Y → A</td>
<td>Implied</td>
<td>TYA</td>
<td>9B</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code</td>
<td>Description</td>
<td></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>
<td></td>
</tr>
<tr>
<td>00</td>
<td>BRK</td>
<td></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>
<td></td>
</tr>
<tr>
<td>02</td>
<td>NOP</td>
<td></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>
<td></td>
</tr>
<tr>
<td>04</td>
<td>NOP</td>
<td></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>
<td></td>
</tr>
<tr>
<td>06</td>
<td>ASL — Zero Page</td>
<td></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>
<td></td>
</tr>
<tr>
<td>08</td>
<td>PIP</td>
<td></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>
<td></td>
</tr>
<tr>
<td>0A</td>
<td>ASL — Accumulator</td>
<td></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>
<td></td>
</tr>
<tr>
<td>0C</td>
<td>NOP</td>
<td></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>
<td></td>
</tr>
<tr>
<td>0E</td>
<td>ASL — Absolute</td>
<td></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>
<td></td>
</tr>
<tr>
<td>10</td>
<td>BPL</td>
<td></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>
<td></td>
</tr>
<tr>
<td>12</td>
<td>NOP</td>
<td></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>
<td></td>
</tr>
<tr>
<td>14</td>
<td>NOP</td>
<td></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>
<td></td>
</tr>
<tr>
<td>16</td>
<td>ASL — Zero Page, X</td>
<td></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>
<td></td>
</tr>
<tr>
<td>18</td>
<td>CLC</td>
<td></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>
<td></td>
</tr>
<tr>
<td>1A</td>
<td>NOP</td>
<td></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>
<td></td>
</tr>
<tr>
<td>1C</td>
<td>NOP</td>
<td></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>
<td></td>
</tr>
<tr>
<td>1E</td>
<td>ASL — Absolute, X</td>
<td></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>
<td></td>
</tr>
<tr>
<td>20</td>
<td>JSR</td>
<td></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>
<td></td>
</tr>
<tr>
<td>22</td>
<td>NOP</td>
<td></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>
<td></td>
</tr>
<tr>
<td>24</td>
<td>BIT — Zero Page</td>
<td></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>
<td></td>
</tr>
<tr>
<td>26</td>
<td>ROL — Zero Page</td>
<td></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>
<td></td>
</tr>
<tr>
<td>28</td>
<td>PLP</td>
<td></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>
<td></td>
</tr>
<tr>
<td>2A</td>
<td>ROL — Accumulator</td>
<td></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>
<td></td>
</tr>
<tr>
<td>2C</td>
<td>BIT — Absolute</td>
<td></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>
<td></td>
</tr>
<tr>
<td>2E</td>
<td>ROL — Absolute</td>
<td></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>
<td></td>
</tr>
<tr>
<td>30</td>
<td>NOP</td>
<td></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>
<td></td>
</tr>
<tr>
<td>32</td>
<td>NOP</td>
<td></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>
<td></td>
</tr>
<tr>
<td>34</td>
<td>NOP</td>
<td></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>
<td></td>
</tr>
<tr>
<td>36</td>
<td>ROL — Zero Page, X</td>
<td></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>
<td></td>
</tr>
<tr>
<td>38</td>
<td>SEC</td>
<td></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>
<td></td>
</tr>
<tr>
<td>3A</td>
<td>NOP</td>
<td></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>
<td></td>
</tr>
<tr>
<td>3C</td>
<td>NOP</td>
<td></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>
<td></td>
</tr>
<tr>
<td>3E</td>
<td>ROL — Absolute, X</td>
<td></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>
<td></td>
</tr>
<tr>
<td>40</td>
<td>RTI</td>
<td></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>
<td></td>
</tr>
<tr>
<td>42</td>
<td>NOP</td>
<td></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>
<td></td>
</tr>
<tr>
<td>44</td>
<td>NOP</td>
<td></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>
<td></td>
</tr>
<tr>
<td>46</td>
<td>LSR — Zero Page</td>
<td></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>
<td></td>
</tr>
<tr>
<td>48</td>
<td>PHA</td>
<td></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>
<td></td>
</tr>
<tr>
<td>4A</td>
<td>LSR — Accumulator</td>
<td></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>
<td></td>
</tr>
<tr>
<td>4C</td>
<td>JMP — Absolute</td>
<td></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>
<td></td>
</tr>
<tr>
<td>4E</td>
<td>LSR — Absolute</td>
<td></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>
<td></td>
</tr>
<tr>
<td>50</td>
<td>BVC</td>
<td></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>
<td></td>
</tr>
<tr>
<td>52</td>
<td>NOP</td>
<td></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>
<td></td>
</tr>
<tr>
<td>54</td>
<td>NOP</td>
<td></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>
<td></td>
</tr>
<tr>
<td>56</td>
<td>LSR — Zero Page, X</td>
<td></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>
<td></td>
</tr>
<tr>
<td>58</td>
<td>CLI</td>
<td></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>
<td></td>
</tr>
<tr>
<td>5A</td>
<td>NOP</td>
<td></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>
<td></td>
</tr>
<tr>
<td>5C</td>
<td>NOP</td>
<td></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>
<td></td>
</tr>
<tr>
<td>5E</td>
<td>LSR — Absolute, X</td>
<td></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>
<td></td>
</tr>
<tr>
<td>60</td>
<td>RTS</td>
<td></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>
<td></td>
</tr>
<tr>
<td>62</td>
<td>NOP</td>
<td></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>
<td></td>
</tr>
<tr>
<td>64</td>
<td>NOP</td>
<td></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>
<td></td>
</tr>
<tr>
<td>66</td>
<td>ROR — Zero Page</td>
<td></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>
<td></td>
</tr>
<tr>
<td>68</td>
<td>PLA</td>
<td></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>
<td></td>
</tr>
<tr>
<td>6A</td>
<td>ROR — Accumulator</td>
<td></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>
<td></td>
</tr>
<tr>
<td>6C</td>
<td>JMP — Indirect</td>
<td></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>
<td></td>
</tr>
<tr>
<td>6E</td>
<td>ROR — Absolute</td>
<td></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>
<td></td>
</tr>
<tr>
<td>70</td>
<td>BVS</td>
<td></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>
<td></td>
</tr>
<tr>
<td>72</td>
<td>NOP</td>
<td></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>
<td></td>
</tr>
<tr>
<td>74</td>
<td>NOP</td>
<td></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>
<td></td>
</tr>
<tr>
<td>76</td>
<td>ROR — Zero Page</td>
<td></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>
<td></td>
</tr>
<tr>
<td>78</td>
<td>SEI</td>
<td></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>
<td></td>
</tr>
<tr>
<td>7A</td>
<td>NOP</td>
<td></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>
<td></td>
</tr>
<tr>
<td>7C</td>
<td>NOP</td>
<td></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>
<td></td>
</tr>
<tr>
<td>7E</td>
<td>ROR — Absolute, X NOP</td>
<td></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>
<td></td>
</tr>
<tr>
<td>80</td>
<td>NOP</td>
<td></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>
<td></td>
</tr>
<tr>
<td>82</td>
<td>NOP</td>
<td></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>
<td></td>
</tr>
<tr>
<td>84</td>
<td>STY — Zero Page</td>
<td></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>
<td></td>
</tr>
<tr>
<td>86</td>
<td>STX — Zero Page</td>
<td></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>
<td></td>
</tr>
<tr>
<td>88</td>
<td>DEY</td>
<td></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>
<td></td>
</tr>
<tr>
<td>8A</td>
<td>TXA</td>
<td></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>
<td></td>
</tr>
<tr>
<td>8C</td>
<td>STY — Absolute</td>
<td></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>
<td></td>
</tr>
<tr>
<td>8E</td>
<td>STX — Absolute</td>
<td></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>
<td></td>
</tr>
<tr>
<td>90</td>
<td>BCC</td>
<td></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>
<td></td>
</tr>
<tr>
<td>92</td>
<td>NOP</td>
<td></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>
<td></td>
</tr>
<tr>
<td>94</td>
<td>STY — Zero Page, X</td>
<td></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>
<td></td>
</tr>
<tr>
<td>96</td>
<td>STX — Zero Page, Y</td>
<td></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>
<td></td>
</tr>
<tr>
<td>98</td>
<td>TYA</td>
<td></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>
<td></td>
</tr>
<tr>
<td>AA</td>
<td>TXS</td>
<td></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>
<td></td>
</tr>
<tr>
<td>AC</td>
<td>CMP — Immediate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AD</td>
<td>STA — Absolute, X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AE</td>
<td>CMP — Zero Page</td>
<td></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>
<td></td>
</tr>
<tr>
<td>B0</td>
<td>BCS</td>
<td></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>
<td></td>
</tr>
<tr>
<td>B2</td>
<td>NOP</td>
<td></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>
<td></td>
</tr>
<tr>
<td>B4</td>
<td>LDY — Zero Page, X</td>
<td></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>
<td></td>
</tr>
<tr>
<td>B6</td>
<td>LDX — Zero Page, Y</td>
<td></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>
<td></td>
</tr>
<tr>
<td>B8</td>
<td>CLV</td>
<td></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>
<td></td>
</tr>
<tr>
<td>BA</td>
<td>TSX</td>
<td></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>
<td></td>
</tr>
<tr>
<td>BC</td>
<td>LDY — Absolute, X</td>
<td></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>
<td></td>
</tr>
<tr>
<td>BE</td>
<td>LDX — Absolute, Y</td>
<td></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>
<td></td>
</tr>
<tr>
<td>C0</td>
<td>CPY — Immediate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C1</td>
<td>CMP — (Indirect, XI)</td>
<td></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>
<td></td>
</tr>
<tr>
<td>C3</td>
<td>NOP</td>
<td></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>
<td></td>
</tr>
<tr>
<td>C5</td>
<td>CMP — Zero Page</td>
<td></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>
<td></td>
</tr>
<tr>
<td>C7</td>
<td>NOP</td>
<td></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>
<td></td>
</tr>
<tr>
<td>C9</td>
<td>CMP — Immediate</td>
<td></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>
<td></td>
</tr>
<tr>
<td>CB</td>
<td>NOP</td>
<td></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>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>CMP — Absolute</td>
<td></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>
<td></td>
</tr>
<tr>
<td>CF</td>
<td>NOP</td>
<td></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>
<td></td>
</tr>
<tr>
<td>D1</td>
<td>CMP — (Indirect, Y)</td>
<td></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>
<td></td>
</tr>
<tr>
<td>D3</td>
<td>NOP</td>
<td></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>
<td></td>
</tr>
<tr>
<td>D5</td>
<td>CMP — Zero Page, X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D6</td>
<td>DEC — Zero Page, X</td>
<td></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>
<td></td>
</tr>
<tr>
<td>D8</td>
<td>CLD</td>
<td></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>
<td></td>
</tr>
<tr>
<td>DA</td>
<td>NOP</td>
<td></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>
<td></td>
</tr>
<tr>
<td>DC</td>
<td>NOP</td>
<td></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>
<td></td>
</tr>
<tr>
<td>DE</td>
<td>DEC — Absolute</td>
<td></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>
<td></td>
</tr>
<tr>
<td>E0</td>
<td>CPX — Immediate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E1</td>
<td>SBC — (Indirect, XI)</td>
<td></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>
<td></td>
</tr>
<tr>
<td>E3</td>
<td>NOP</td>
<td></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>
<td></td>
</tr>
<tr>
<td>E5</td>
<td>SBC — Zero Page</td>
<td></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>
<td></td>
</tr>
<tr>
<td>E7</td>
<td>NOP</td>
<td></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>
<td></td>
</tr>
<tr>
<td>E9</td>
<td>SBC — Immediate</td>
<td></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>
<td></td>
</tr>
<tr>
<td>EB</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EC</td>
<td>CPX — Absolute</td>
<td></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>
<td></td>
</tr>
<tr>
<td>EE</td>
<td>INC — Absolute</td>
<td></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>
<td></td>
</tr>
<tr>
<td>F0</td>
<td>BEO</td>
<td></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>
<td></td>
</tr>
<tr>
<td>F2</td>
<td>NOP</td>
<td></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>
<td></td>
</tr>
<tr>
<td>F4</td>
<td>NOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F5</td>
<td>SBC — Zero Page, X</td>
<td></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>
<td></td>
</tr>
<tr>
<td>F7</td>
<td>NOP</td>
<td></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>
<td></td>
</tr>
<tr>
<td>F9</td>
<td>SBC — Absolute, Y</td>
<td></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>
<td></td>
</tr>
<tr>
<td>FB</td>
<td>NOP</td>
<td></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>
<td></td>
</tr>
<tr>
<td>FD</td>
<td>SBC — Absolute, X</td>
<td></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>
<td></td>
</tr>
<tr>
<td>FF</td>
<td>NOP</td>
<td></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-verter"\textsuperscript{TM} Higher channel (7-13) modulators generally provide better system performance than lower channel modulators (2-6).

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

1. +12 Volts with the following current capacity:
   a. For 4K or 16K systems - 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 mother board, turn on power supplies and verify voltages on connector pins. Improper supply connections such as reverse polarity can severely damage your APPLE II.

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

3. Plug in speaker cable.

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

5. Connect video cable.

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

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

8. With power supplies turned off, plug in power connector to mother board then recheck all 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 slightly. 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 auxilliary 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., $107\text{V}$ to $132\text{V}$.

Under no circumstances, should more than $140\text{VAC}$ 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 $\frac{1}{2}$ 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 $+5.0 \pm 0.15$ volts, $+11.8 \pm 0.5$ volts, $-12.0 \pm 1$V, $-5.2 \pm 0.5$ 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 50dB 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 coax cable connecting the modulator to the TV set repeatedly through a Ferrite toroid core. Apple Computer supplies these cores in a kit, along with a 4 circuit connector/cable assembly to match the auxiliary video connector found on the Apple II board. This kit has order number A2M010X. The M&R "Sup-r-Mod" is supplied with a coax cable and toroids.

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

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

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

The ASR - 33 Teletype

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

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

b) Above Terminal Strip X is a connector socket identified as "2". Pin 8 is the input line + or high; Pin 7 is the input line - or low. This connector mates with a Molex receptacle model 1375 #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 5-11 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 80 column printer as the primary output device.
2. PR#Ø will inactivate the printer transferring control back to the Video monitor as the primary output device.

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

Saving the Machine Language Program

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

The way it is saved is as follows:
1. Insert a blank program cassette into the tape recorder and rewind it.

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

3. Type in "37Ø.Ø3E9W 37Ø.Ø3E9W".

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

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

The Program

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

1. BC (control B) into BASIC

   10 CALL 88Ø 2ØØ
   15 PRINT "ABCD...XYZØ1123456789"
   20 PR#Ø
   25 END

2. Turn the teletype (printer on)

3. Type in the following

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

TITLE 'TELETYPET DRIVER ROUTINES'

* * * **************
* TTYDRIVER: *
* TELETYPET OUTPUT *
* ROUTINE FOR 72 *
* COLUMN PRINT WITH *
* BASIC LIST *
* *
* COPYRIGHT 1977 BY: *
* APPLE COMPUTER INC. *
* 11/18/77 *
* *
* R. WIGGINSTON *
* S. WOZNIAK *
* *
* **************

VNDWIDTH EQU $21 ; FOR APPLE-II
CH EQU $24 ; CURSOR HORIZ.
CSWL EQU $36 ; CHAR. OUT SWITCH
YSAVE EQU $778
COLCNT EQU $7F8 ; COLUMN COUNT LOC.
MARK EQU $C058
SPACE EQU $C059
WAIT EQU $FCA8
ORG $370

***WARNING: OPERAND OVERFLOW IN LINE 27

0370: A9 82 27 TTINIT: LDA #TTOUT
0372: 85 36 28 STA CSWL ; POINT TO TTY ROUTINES
0374: A9 03 29 LDA #TTOUT/256 ; HIGH BYTE
0376: 85 37 30 -STA CSWL+1
0378: A9 48 31 LDA #72 ; SET WINDOW WIDTH
037A: 85 21 32 -STA VNDWIDTH ; TO NUMBER COLUMNS ONLY
037C: A5 24 33 -LDA CH
037E: 8D F8 07 34 STA COLCNT ; WHERE WE ARE NOW.
0381: 60 35 RTS
0383: 48 36 TTOUT: PHA ; SAVE TWICE
0385: 48 37 PHA ; ON STACK.
0387: AD F8 07 38 TTOUT2: LDA COLCNT ; CHECK FOR A TAB.
0389: C5 24 39 CMP CH
038B: 68 40 PLA ; RESTORE OUTPUT CHAR.
038D: B0 03 41 & & BCS TESTCTRL ; IF C SET, NO TAB
038F: 48 42 PHA
0391: A9 A0 43 LDA #$A0 ; PRINT A SPACE.
0393: 2C C0 03 44 TESTCTRL: BIT RTS1 ; TRICK TO DETERMINE
0395: F0 03 45 BEQ PRTNTIT ; IF CONTROL CHAR.
0397: EE F8 07 46 INC COLCNT ; IF NOT, ADD ONE TO CH
0399: 20 C1 03 47 PRTNTIT: JSR DOLCHAR ; PRINT THE CHAR ON TTY
039A: 68 48 PLA ; RESTORE CHAR
039B: 48 49 PHA ; AND PUT BACK ON STACK
039D: 90 E6 50 BCC TTOUT2 ; DO MORE SPACES FOR TA
039F: 49 DD 51 EOR #$D0 ; CHECK FOR CAR RET.
03A1: 0A 52 ASL A ; SELIM PARITY
03A3: DD 0D 53 BNE FINISH ; IF NOT CR, DONE.

FIGURE 3a

119

Source: David T Craig
TELETYPING DRIVER ROUTINES

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

03A3: 8D F8 07 54 STA COLCNT CLEAR COLUMN COUNT
03A6: A9 8A 55 LDA $8A putting in a header
03A8: 20 C1 03 56 JSR DOCHAR
03AB: A9 58 57 LDA $58
03AD: 20 A8 FC 58 JSR WAIT 200msec delay for line
03B0: AD F8 07 59 FINISH: LDA COLCNT CHECK IF IN MARGIN
03B3: F0 08 60 BEQ SETCH FOR CR, RESET CH
03B5: E5 21 61 SBC VNDNTH & 8 IF SO, CARRY SET.
03B7: E9 F7 62 SBC $F7
03BA: 90 04 63 BCC RETURN
03B3: 69 1F 64 ADC $1F
03BD: 85 24 65 SETCH: STA CH
03BF: 68 66 RETURN: PLA
03C0: 60 67 RTS1: RTS
03C1: 8C 78 07 69 DOCHAR: STY YSAVE
03C4: 08 70 PHP SAVE STATUS.
03C5: A0 03 71 LDY $03 11 bits (1 start, 9 info)
03C7: 18 72 CLC BEGIN WITH SPACE (STX).
03C8: 4B 73 TTOUT3: PHA SAVE A REG AND SET FOR
03C9: B0 05 74 BCS MARKOUT MARKOUT.
03CB: AD 59 C0 75 LDA SPACE SEND A SPACE.
03CE: 90 03 76 BCC TTOUT4 SEND A MARK.
03D0: AD 58 C0 77 MARKOUT: LDA MARK SEND A MARK.
03D3: A9 D7 78 TTOUT4: LDA #$D7 DELAY 9.071 msec for
03D5: 48 79 DLYI: PHA 110 baud.
03D6: A9 20 80 LDA #$20
03D8: 4A 81 DLY2: LSR A
03DA: 90 FD 82 BCC DLY2
03DB: 68 83 PLA
03DC: 29 01 84 SBC $01
03DE: DO F5 85 BNE DLY1
03E0: 68 86 PLA
03E1: 6A 87 BOR A NEXT BIT (STOP BITS)
03E2: 88 88 DEY LOOP 11 BITS.
03E3: DO E3 89 BNE TTOUT3
03E5: AC 78 07 90 LDY YSAVE RESTORE Y-REG.
03E8: 28 91 PLP RESTORE STATUS
03E9: 60 92 RTS RETURN

******SUCCESSFUL ASSEMBLY: NO ERRORS

FIGURE 3b

120

Source: David T Craig
<table>
<thead>
<tr>
<th>CROSS-REFERENCE</th>
<th>TELETYPE DRIVER ROUTINES</th>
</tr>
</thead>
<tbody>
<tr>
<td>CH</td>
<td>0024 0033 0039 0065</td>
</tr>
<tr>
<td>COLCNT</td>
<td>07F8 0034 0038 0046 0054 0059</td>
</tr>
<tr>
<td>CSVL</td>
<td>0036 0028 0030</td>
</tr>
<tr>
<td>DLYI</td>
<td>03D5 0085</td>
</tr>
<tr>
<td>DLY2</td>
<td>03D8 0082</td>
</tr>
<tr>
<td>DOCHAR</td>
<td>03C1 0047 0056</td>
</tr>
<tr>
<td>FINISH</td>
<td>03B0 0053</td>
</tr>
<tr>
<td>MARK</td>
<td>C058 0077</td>
</tr>
<tr>
<td>MARKOUT</td>
<td>03D0 0074</td>
</tr>
<tr>
<td>PRINTIT</td>
<td>0397 0045</td>
</tr>
<tr>
<td>RETURN</td>
<td>03BF 0063</td>
</tr>
<tr>
<td>RTSI</td>
<td>03C0 0044</td>
</tr>
<tr>
<td>SETCH</td>
<td>03BD 0060</td>
</tr>
<tr>
<td>SPACE</td>
<td>C059 0075</td>
</tr>
<tr>
<td>TESTCTRL</td>
<td>038F 0041</td>
</tr>
<tr>
<td>TTINIT</td>
<td>0370</td>
</tr>
<tr>
<td>TTOUT</td>
<td>0382 0027 0029</td>
</tr>
<tr>
<td>TTOUT2</td>
<td>0384 0050</td>
</tr>
<tr>
<td>TTOUT3</td>
<td>03C8 0089</td>
</tr>
<tr>
<td>TTOUT4</td>
<td>03D3 0076</td>
</tr>
<tr>
<td>WAIT</td>
<td>FCA8 0058</td>
</tr>
<tr>
<td>WNDWIDTH</td>
<td>0021 0032 0061</td>
</tr>
<tr>
<td>YSAVE</td>
<td>0778 0069 0090</td>
</tr>
<tr>
<td>ILE!</td>
<td></td>
</tr>
</tbody>
</table>

**FIGURE 3c**

121
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
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. \( V_{IN} = 1 V_{pp} \) (nominal), \( Z_{IN} = 12\,\text{K Ohms} \). 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. \( V_{OUT} = 25 \, \text{mV} \) into 100 Ohms, \( Z_{OUT} = 100 \, \text{Ohms} \). 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

| TOP VIEW |
|---|---|
| **+5V** | 1 | 16 | N.C. |
| SW0  | 2 | 15 | AN0 |
| SW1  | 3 | 14 | AN1 |
| SW2  | 4 | 13 | AN2 |
| C040STB | 5 | 12 | AN3 |
| PDL0 | 6 | 11 | PDL3 |
| PDL2 | 7 | 10 | PDL1 |
| GND  | 8 | 9 | N.C. |

LOCATION J14

Source: David T Craig
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.

CO40 STBL:  A utility strobe output. Will go low during $t_2$ of a read or write cycle to addresses CO40-C04F. This is a 74LSxx series TTL output.

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

NC:  No connection.

PDL0-PDL3:  Paddle control inputs. Requires a $\Omega$-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.

SW0-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

+5V  16 N.C.
STROBE  15 -12V
RESET  14 N.C.
N.C.  13 B2
B6  12 B1
B5  11 B4
B7  10 B3
GND  9 N.C.

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. 0 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 #2HW25C0-111 (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 A2. During a read cycle the 6502 expects data to be ready no less than 100ns before the end of A2. These lines will drive up to a total of 8 standard 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.

Ø0: Microprocessor phase Ø clock. Will drive up to a total of 16 standard TTL loads.

Ø1: Phase 1 clock, complement of Ø0. Will drive up to a total of 16 standard TTL loads.

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

+12V: Positive 12-Volt supply.

+5V: Positive 5-Volt supply

-5V: Negative 5-Volt supply.

-12V: Negative 12-Volt supply.

POWER CONNECTOR

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

PIN DESCRIPTION

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

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

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

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

-12V: Negative 5-Volt line from power supply.
Figure 4  
PERIPHERAL CONNECTORS  
( Eight of Each )

PINOUT  
( Back Edge of PC Board )

GND  26  +5V  25  DMA IN  27  DMA OUT  24  INT IN  28  INT OUT  23  NMI  29  DMA  22  IRQ  30  RDY  21  RES  31  I/O STROBE  20  INH  32  N.C.  19  -12V  33  R/W  18  7M  36  A15  17  Q3  37  A14  16  #1  38  A13  15  USER 1  39  A12  14  40  A11  13  41  A10  12  42  A9  11  43  A8  10  44  A7  9  45  A6  8  46  A5  7  47  A4  6  48  A3  5  49  A2  4  50  A1  3  51  A0  2  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 )

TOP VIEW

( BLUE/WHITE WIRE ) -12V
( ORANGE WIRE ) +5V
( BLACK WIRE ) GND

-5V ( BLUE WIRE )
+12V ( ORANGE/WHITE WIRE )
GND ( BLACK WIRE )

LOCATION K1

130
**SPEAKER CONNECTOR**

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

**SIGNAL DESCRIPTION FOR SPEAKER**

+5V: System +5 Volts

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

Figure 6

---

**VIDEO OUTPUT JACK**

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

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

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

See locations illustrated in Figure 1.
AUXILIARY VIDEO OUTPUT CONNECTOR

This is a MOLEX KK 100 series connector with four .25" square pins on .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

<table>
<thead>
<tr>
<th>PINOUT</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0000-0FFF) 4K</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
</tr>
<tr>
<td>(1000-1FFF) 4K</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(2000-2FFF) 4K</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(3000-3FFF) 4K</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(4000-4FFF) 4K</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(5000-5FFF) 4K</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(6000-6FFF) 4K</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></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 0000, jumper pin 14 to pin 10 on all three MEMORY SELECT sockets (thereby assigning row "C" to the 0000-3FFF 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 4000 and 5000 respectively, jumper pin 13 to pin 5 (thereby assigning row "D" to the 4000-4FFF range of memory), and jumper pin 12 to pin 6 (thereby assigning row "E" to the 5000-5FFF 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 0000-5FFF.

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>Address (Hex)</th>
<th>Used By</th>
<th>Used For</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>Utility</td>
<td>register area for &quot;sweet 16&quot; 16 bit firmware processor.</td>
<td></td>
</tr>
<tr>
<td>0020-004D</td>
<td>Monitor</td>
<td>holds a 16 bit number that is randomized with each key entry.</td>
<td></td>
</tr>
<tr>
<td>004E-004F</td>
<td>Monitor</td>
<td>integer multiply and divide work space.</td>
<td></td>
</tr>
<tr>
<td>0050-0055</td>
<td>Utility</td>
<td>floating point work space.</td>
<td></td>
</tr>
<tr>
<td>0055-00FF</td>
<td>Basic</td>
<td>subroutine return stack.</td>
<td></td>
</tr>
<tr>
<td>0070-00FF</td>
<td>Utility</td>
<td>character input buffer.</td>
<td></td>
</tr>
<tr>
<td>0100-01FF</td>
<td></td>
<td>Y, (control Y) will cause a JSR to this location.</td>
<td></td>
</tr>
<tr>
<td>0100-01FF</td>
<td></td>
<td>NMI's are vectored to this location.</td>
<td></td>
</tr>
<tr>
<td>0100-01FF</td>
<td></td>
<td>IRQ's are vectored to the address pointed to by these locations.</td>
<td></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>BASIC initializes LOHEN to location 0800.</td>
</tr>
</tbody>
</table>
# I/O and ROM Address Detail

<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></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; 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></td>
<td></td>
<td>State of Switch 1 \ input on Game I/O connector appears in bit 7.</td>
</tr>
<tr>
<td>C061/9</td>
<td>&quot;SW1&quot;</td>
<td>State of Switch 2 input on Game I/O connector appears in bit 7.</td>
</tr>
<tr>
<td>C062/A</td>
<td>&quot;SW2&quot;</td>
<td>State of Switch 3 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 timer output for Paddle 0 appears in bit 7.</td>
</tr>
<tr>
<td>C064/C</td>
<td>Paddle 0 timer output</td>
<td>State of timer output for Paddle 1 appears in bit 7.</td>
</tr>
<tr>
<td>C065/D</td>
<td>&quot; 1 &quot;</td>
<td>State of timer output for Paddle 2 appears in bit 7.</td>
</tr>
<tr>
<td>C066/E</td>
<td>&quot; 2 &quot;</td>
<td>State of timer output for Paddle 3 appears in bit 7.</td>
</tr>
<tr>
<td>C067/F</td>
<td>&quot; 3 &quot;</td>
<td>Triggers paddle timers during $\phi_2$.</td>
</tr>
<tr>
<td>C07X</td>
<td>&quot;PDL STB&quot;</td>
<td></td>
</tr>
<tr>
<td>C08X</td>
<td>DEVICE SELECT 0</td>
<td>Pin 41 on the selected Peripheral Connector goes low during $\phi_2$.</td>
</tr>
<tr>
<td>C09X</td>
<td>&quot; 1 &quot;</td>
<td>Expansion connectors.</td>
</tr>
<tr>
<td>C0A9X</td>
<td>&quot; 2 &quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>C0BX</td>
<td>&quot; 3 &quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>C0CX</td>
<td>&quot; 4 &quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>C0DX</td>
<td>&quot; 5 &quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>C0EX</td>
<td>&quot; 6 &quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>C0FX</td>
<td>&quot; 7 &quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>C10X</td>
<td>&quot; 8 &quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>C11X</td>
<td>&quot; 9 &quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>C12X</td>
<td>&quot; A &quot;</td>
<td>&quot;</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-FFFFF</td>
<td></td>
<td>F8</td>
</tr>
</tbody>
</table>

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

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
FIGURE S-1  APPLE II SYSTEM DIAGRAM

141
FIGURE S-2 MPU AND SYSTEM BUS
FIGURE S-4  SYNC COUNTER
FIGURE S-7 RAM ADDRESS MUX

Source: David T Craig
Figure S-8 4K to 48K RAM Memory with Data Latch
FIGURE S-9  PERIPHERAL I/O CONNECTOR PINOUT AND CONTROL LOGIC

149