Apple II Reference Manual

Autor:
Data de publicació: 30-04-2017

Apple II

Reference

Manual

January 1978

APPLE n
Reference Manual

January 1978

APPLE Computer Inc.

10260 Bandley Dr.

Cupertino, CA

95014

APPLE n Reference Manual
TABLE OF CONTENTS

GETTING STARTED WITH YOUR APPLE II

Unpacking

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

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

2. Accessories Box including the following:

a. This manual including warranty card .

b. Pair of Game Paddles

c. A.C. Power Cord

d. Cassette tape with "Breakouf'on one side and "Color Demos" on the other side.

e. Cassette recorder interface cable (miniature phone jack type)

3. If you purchased a 16K or larger system, your accessory box should also contain:

a. 16K Startrek game cassette with High Resolution Graphics Demo ("HIRES") on the flipside.

b. Applesoft Floating Point Basic Language Cassette
with an example program on the other side.

c. Applesoft reference manual

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

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

Warranty Registration Card

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

A2SJ0OMMX
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 radom text characters (typically question marks)

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

Apple Speaks Several Languages

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

Apple Integer BASIC

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

Running Your First and Second Program

Read through the next three sections that include:

1. Loading a BASIC program Tape

2. Breakout Game Tape

3. Color Demo Tape

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

Running 16K Startrek

If you have 16K Bytes or larger memory in your Apple, you will also receive a "STARTREK" game tape. Load this program just as you did the previous two, but 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 section; System Checkout, Loading a Tape and What to do when you have Loading Problems. They are discussed below.

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

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

Cassette tape recorders in the $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 1 s maximum, or to just a dim indication on the level lamp. Listen to the recorded tape after you've saved a program to ensure that the recording is "loud and clear".

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

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

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

Listening to the tape can help solve other problems as well. Flaws in the tape, excessive speed variations, and distortion can be detected this way. Saving a program several times in a row is good insurance against tape flaws. One thing to listen for is a good clean tone lasting for at least 3 1/2 seconds is needed by the computer to "set up" for proper loading. The Apple puts out this tone for anout 10 seconds when saving a program, so you normally have 6 1/2 seconds of leeway. If the playback volume is too high, you may pick up tape noise before getting to the set-up tone. Try a lower playback volume.

SYSTEM CHECKOUT

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

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

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

3. VIDEO DISPLAY INTERFACE -

a) for a video monitor - check that a cable connects the monitor to the Apple's video output port,

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

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

5. POWER ON - flip on the power switch in back of the Apple II, the "power" indicator on the keyboard will light. Also

make sure the video monitor (or TV set) is turned on.

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

1. Hit the RESET key.

An asterick, "*", should appear on the lefthand side

of the screen below the random text pattern. A flashing

white cursor will appear to the right of the asterick.

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

3. Type in the word "LOAD" on the keyboard. You should see the word in between the right facing arrow and the flashing cursor. 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 maximum.

6. Start the tape recorder in "PLAY" mode and now depress the "RETURN" key on the Apple II.

7. The cursor will disappear and Apple II will beep in a few seconds when it finds the beginning of the program. If an error message is flashed on the screen, proceed through the steps listed in the Tape Problem section of this paper.

8. A second beep will sound and the flashing cursor will reappear after the program has been successfully loaded into the computer.

9. Stop the tape recorder. You may want to rewind the program tape at this time.

10. Type in the word "RUN" and depress the "RETURN" key.

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

LOADING PROBLEMS

Occasionally, while attempting to load a BASIC program Apple II beeps and a memory full error is written on the screen. At this time you might wonder what is wrong with the computer, with the program tape, or with the cassette recorder. Stop. This is the time when you need

to take a moment and checkout the system rather than haphazardly attempting to resolve the loading problem. Thoughtful action taken here will speed in a program's entry. If you were able to successfully turn on the computer, reset it, and place it into BASIC then the Apple II is probably operating correctly. Before describing a procedure for resolving this loading problem, a discussion of what a memory full error is in order.

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

Memory Size too Small

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

suitably sized systems.

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

Cassette Recorder Inadjustment

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

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

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

Tape Problems

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

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

Dealing with the Loading Problem

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

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

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

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

4. Check the program tape by listening to it.

a) Replace it if it is defective, or

b) start it at the beginning of the program.

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

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

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

BREAKOUT GAME TAPE

PROGRAM DESCRIPTION

Breakout is a color graphics game for the Apple II computer. The object of

the game is to "knock-out* all 160 colored bricks from the playing field by hitting them with the bouncing ball. You direct the ball by hitting it with a paddle on the left side of the screen. You control the paddle with one of the Apple's Game Paddle controllers. But watch out: you can only miss the ball five times!

There are eight columns of bricks. As you penetrate through the wall the point value of the bricks increases. A perfect game is 72(1 points; after five balls have been played the computer will display your score and a rating such as "Very Good". N 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 to 15, for background, even bricks, odd bricks, paddle and ball colors. After these have been chosen the game will begin.

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 (0) of the Game I/O connector for this game.

COLOR DEMO TAPE

PROGRAM DESCRIPTION

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

REQUIREMENTS

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

11

BREAKOUT GAME
PROGRAM LISTING

PROGRAM LISTING

-•J T : _it:

??H.--1 i-j-i =.".

= : i_U= * -.'J-^-JU

?J-J-J .- = 3*i_f

?•„".: i ?-. t'l- ? : J-JT:

iiWJ l:LT: UH! :*_ i:i-?= i ?" :: .-"

U^j^-J _-•_=

-J i '-' -? »: i : i :

iif; - is ;~v= uii ? '_= uv

;--.;-. i: •*.??-•_:- iiii.;i s--j~ s uuuu-

~2]= IUL=

JK-b

I;

IrULuK*

*t-r

™ EHS.5: :-.i

i :=.? -i-Js ^~ ? Li-fi i-J-J-Jv

12

COLOR DEMO PROGRAM

LISTING

PROGRAM LISTING

0,1001 fvfvr.

rii£^

. svs i v:vi» ;_-_- £.-•_•

H= HHINI :: 4K UULuR DlHOS"' PR X N1

'-:liL= — 1 4-1 * Tz • sfif: OOii = Cf:D T — 1

"-"-"--? v ?-? - t s ?_? -i trvL"' i-t- ! i: : !T 1~ Z

•pi ; r-:if= /if .=. .™;

: Hi- s * i

ii-i isi.-"-. *

l; ntin h:L hi Ls YL In NjL Hi

K; HLXI jri= hySUK IHHMn- GO 1 8?^ P-H-fi-rliV-n/lsn: ffM nP=i?= ^0^1!

46 PRIHT ;; 4 TUNHEL" * P R 1 II I ^5 CIRC

i = uv?_?-ju ut^

= -JU = U -JL-J

r-i 5JlT
i l"fir?i

KlvjjiJ "HI! HNY i.?i rW. Nrkl UrFJlT

* 7z£i i! n : DT;.=T : T=JD!;T Hliiirii nr=lfi

Ji- *.' = = L.i:*: ? i:;iUi =?::i^:: i/=_ni?

f sir =irl ! II" 1/S Hrti-- lll

THEH GOTO 188*1: GOTO 38

TkiDUT ZUW-PU f^r^n nr-::: r-. ^r : j; ; t-.-t
inrv= ^-iii/if e/LHU ?L=:JL[/ >UU LINl

i: T. r~i . " r * ~ ! ifi T ?' ?" ?" Turn

»x: UK I ir 1 riND K^ itllrl
GOTO 188*1; GOTO 3^

T-ii? sf;f- "0- ?- T -" J x^^v. t --fi

i - I £ i ? ivi" ! -? i •-' - i i 1 /O .•• j^"-. 5 7

-i-lh GOSUB £868; GOSUB 18888

'j-n run i

isv t:i i: U ! , ! _". ! D C :1 tv; P:L:": I li

~ ": " : : ~: -. t" :" "j ;! . ~ i" T ?"• •" " -"•

h4H K-|f ;';|_~K siKlriTlC^hT?^:! z'227fj

S'J 7 J i i "- I iVJif l-Jl rU^ I ~ r J iU -J?

t FLU! nsT; HLf^l fisil UUJiUi;

viu ^^itj fjiji; ^^; L- U L U K — -J -' si = VLIN

8,39 RT 3tj: VTBB EinJ/c) HOD

"" : i TQD Oils TC T £!":*?- ": TMri;
r ; r:-i:T !.-?-.. r:": - "™ -?«.-..--.-. .-.-•!-.-.

rfurn ^^cii uu^ud id^^di uU!U

tj=39-li GObljH 2Mm GOSUB

.y-."_* iri-Lyp- r-i-iL' i =_-.-(«•?_ a« = 07 Hi

' POKE -16368 J; POP ; GOTO

13

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

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

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

NUMBERS DISPLAYED FOR EACH QUADRANT DENOTE t

* OF STARS IN THE ONES PLACE

* OF BASES IN THE TENS PLACE

* OF KLINGONS IN THE HUNDREDS PLACE
AT ANY TIME DURING THE GAME r 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
THEY ARE!

(ZERO) AND RETURN.

REGENERATE
PHASERS
GALAXY RECORD
PROBE

PROPULSION
LONG RANGE SENSORS
PHOTON TORPEDOES
COMPUTER
SHIELD ENERGY
11. LOAD PHOTON TORPEDOES
THE COMANDS ARE INVOKED BY TYPING THE NUMBER REFERING TO THEM

FOLLOWED BY A 'RETURN".

A. IF RESPONSE IS 1 THE COMPUTER

EXPECTS »W IF ONE WANTS
BETWEEN QUADRANTS AND AN
INTERNAL QUADRANT TRAVEL-
DURATION OR WARP FACTOR IS THE NUMBER OF SPACES OR
QUADRANTS THE ENTERPRISE WILL MOVE.

COURSE IS COMPASS READING IN DEGREES FOR THE DESI-
RED 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.
B. 4 FIRES PHASERS AT THE EXPENSE OF AVAILABLE ENERGY.

10. DAMAGE REPORT

WILL ASK WARP OR
TO TRAVEL IN THE

ION AND
GALAXY

IF ONE WANTS ONLY

F. 6

G.

H.

E. 5 INITIATES A SET OF QUESTIONS FOR TORPEDO FIRING.
THEY CAN BE FIRED AUTOMATICALLY IF THEY HAVE
BEEN LOCKED ON TARGET WHILE IN THE COMPUTER
MODE, OR MAY BE FIRED MANUALLY IF THE TRAGECTORY ANGLE
IS KNOWN.
8 AND 10 ALL GIVE INFORMATION ABOUT THE STATUS OF THE' SHIP
AND ITS ENVIRONMENT.
9 SETS THE SHIELD ENERGY/AVAILABLE ENERGY RATIO.
11 ASKS FOR INFORMATION ON LOADING AND UNLOADING OF

PHOTON TORPEDOES AT THE ESPENSE OF AVAILABLE ENERGY.

THE ANSWER SHOULD BE A SIGNED NUMBER. FOR EXAMPLE
+5 OR -2.
I. 7 ENTERS A COMPUTER WHICH WILL RESPOND TO THE FOLLOWING
INSTRUCTIONS:

1. COMPUTE COURSE

3. LOCK PHOTON TORPEDOES

4. LOCK COURSE 5,
6. STATUS 7,

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

2, LOCK PHASERS

COMPUTE TREJECTORY
RETURN TO COMAND MODE

COURSE OR trajectory;

270-

90

180

THIS EXPLANATION WAS WRITTEN BY ELWOOD
WOT 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 (8p-lpp%) 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 B c command places the Apple into BASIC initializing the memory pointers.

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

HI -RES demo tape.

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

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

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

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

SUMMARY OF HI-RES DEMO TAPE LOADING

1 . RESET

2. Type in C00.FFFR

3. Start tape recorder, hit RETURN

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

5. Type in "LOAD", hit RETURN

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

7. Type in "RUN", hit RETURN

8. STOP tape recorder, rewind tape.

16

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 M : " are NOT allowed.

COMMAND NAME

AUTO num

AUTO

numl,

> num2

CLR

CON

?

DEL

numl

DEL

numl,

num2

DSP

var

HIMEM

expr

GOTO expr

GR

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.

Same as above execpt increments line numbers by number num2«

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

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

Deletes line number numl.

Deletes program from line numbemuml through line number num2.

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.)

Sets highest memory location for use by BASIC at location specified by expression exp^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*.

Causes immediate jump to line number specified by expression expr.

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

LIST numl

LIST numl, num2

Lists entire program on screen.
Lists program line number numl.

Lists program line numberm^ through line number

num2.

18

LOAD expr,

LOMEM : expv

MAN

NEW

NO DSP var

NO TRACE

RUN

RUN expv

SAVE

TEXT

TRACE

Reads (Loads) a BASIC program from cassette tape.
Start tape recorder before hitting return key. Two
beeps and a " > M indicate a good load, "ERR" or "MEM"

FULL ERR" message indicates a bad tape or poor recorder performance.

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.

Clears AUTO line numbering mode to all manual line numbering after a control C* or control X*.

Clears (Scratches) current BASIC program.

Clears DSP mode for variablet?ar>.

Clears TRACE mode.

Clears variables to zero, undimensions all arrays and executes program starting at lowest statement line number.

Clears variables and executes program starting at line number specified by expression expr.

Stores (saves) a BASIC program on a cassette tape. Start tape recorder in record mode prior to hitting return key.

Sets all text mode. Screen is formated to display alpha-numeric characters on 24 lines of 40 characters each. TEXT resets scrolling window to maximum.

Sets debug mode that displays line number of each statement as it is executed.

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

BASIC Operators

Symbol Sample Statement
Prefix Operators
( ) 10 X= 4*(5 + X)

+ 20 X= 1+4*5

30 ALPHA =
-(BETA +2)

NOT 40 IF A NOT B THEN
200

Explanation

Expressions within parenthesis ( )
are always evaluated first.

Optional; +1 times following expression

Negation of following expression.

Logical Negation of following expression;
if expression is true (non-zero), 1
if expression is false (zero).

Arithmetic Operators
+ 66 Y = X+3

7J0 LET D0TS=A*B*N2

Exponentiate as in X'
shifted letter N.

NOTE

+ is

Multiplication. NOTE: Implied multi-
plication such as (2 + 3) (4) is not
allowed thus N2 in example is a variable
not N * 2.

/

80 PRINT GAMMA/S

Divide

MOD

90 X = 12 MOD 7
100 X = X M0D(Y+2)

Modulo:
first exp

+

110 P = L + G

Add

–

120 XY4 = H-D

Substract

130 HEIGHT=15

140 LET SIZE=7*5

150 A(8) = 2

155 ALPHA$ = "PLEASE"

Modulo: Remainder after division of
first expression by second expression

Assignment operator; assigns a value to
a variable. LET is optional

20

Relational and Logical Operators

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

"false" if zero.

Symbol Sample Statement

Explanation

16? IF D = E
THEN 500

Expression "equals" expression

17? IF A$(l,l)
"Y" THEN 500

or < >

180 IF ALPHA #X*Y
THEN 500

#

190 IF A$ # "NO"
THEN 500

>

200 IF A>B
THEN GO TO 50

<

210 IF A+l<B-5
THEN 100

>=

220 IF A>=B
THEN 100

<=

230 IF A+I<=B-6
THEN 200

AND

240 IF A>B AND
C<D THEN 200

OR

250 IF ALPHA OR
BETA+1 THEN 200

String variable "equals?' string variable

Expression "does not equal" expression.

String variable "does not equal" string
variable. NOTE: If strings are not
the same length, they are considered
un-equal . < > not allowed with strings

Expression "is greater than" expression

Expression "is less than" expression.

Expression "is greater than or equal to"
expression.

Expression "is less than or equal to"
expression.

Expression 1 "and" expression 2 must
both be "true" for statements to be true

If either expression 1 or expression 2
is "true", statement is "true".

21

BASIC FUNCTIONS

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

FUNCTION NAME	
ABS (expr)	300 PRINT ABS(X)
ASC (str$)	31Qf PRINT ASC("BACK M)
	320 PRINT ASC(B$) 330 PRINT ASC(B$(4,4)) 335 PRINT ASC(B$(Y))
LEN (str$)	340 PRINT LEN(B$)
PDL (expr)	350 print PDL(X)
PEEK (expr)	360 PRINT PEEK(X)
RND (expr)	370 PRINT RND(X)
	Gives absolute value of the expression expr.
	Gives decimal ASCII value of designated string variable str$. If more than one character is in designated string or sub-string, it gives decimal ASCII value of first character,
	Gives current length of designated string variable etr$i.e. 9 number of characters.
	Gives number between and 255 corres-ponding 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.
	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 FFFGf is -16
	Gives random number between and (expression expr -1) if expression expr

is positive; if minus, it gives random
number between and (expression expr +1).

Sm(exprl, 380 PRINT SCRN (XI, Yl) Gives color (number between and 15) of
expr2) screen at horizontal location designated

by expression exprl and vertical
location designated by expression expr2
Range of expression exprl is to 39. Range
of expression expr2 is to 39 if in standan
mixed colorgraphics display mode as set by
GR command or p to 47 if in all color mode
set by POKE -163J04 ,jD: POKE - 16302,0.

SGN (expr) 39/) PRINT SGN(X)

Gives siqn (not sine) of expression expr
i.e., -1 if expression expr is negative, zero
zero and +1 if expr is positive.

22

BASIC STATEMENTS

Each BASIC statement must have a
names must start with an alpha c
numeric characters up to iffl.
of the following words: AND, AT,
not begin with the letters END,
with a $ (dollar sign). Multipl
if separated by a : (colon) as 1
(including spaces) is less than
Most statements may also be used
by RUN or GOTO commands.

line number between and 32767. Variable
haracter and may be any number of alpha-
Variable names may not contain buried any

MOD, OR, STEP, or THEN. Variable names may
LET, or REM. String variables names must end
e statements may appear under the same line number
ong as the total number of characters in the line
approximately 150 characters

as commands. BASIC statements are executed

NAME

CALL expv

10 CALL-936

C0L0R=5xpr

30 C0L0R=12

DIM vavl (expvl)
stv$ (expv2)
Vav2 (expv 3)

DSP

VOV

50 DIM A(20),B(10)

60 DIM B$(30)

70 DIM C (Z)
Illeqal:

80 DIM A(30)
Leqal :

85 DIM C(1000)

Leqal :

90 DSP AX: DSP L
Illeqal:

100 DSP AX,B

102 DSP AB$

104 DSP A(5)
Legal :

105 A=A(5): DSP A

Causes execution of a machine level language subroutine at decimal memory location specified by expression expv Locations above 32767 are specified using negative numbers; i.e., location in example 10 is hexidecimal number $FC53

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

The DIM statement causes APPLE II to reserve memory for the specified variables. For number arrays APPLE reserves approximately 2 times expv bytes of memory limited by available memory. For string arrays -stv$-(expv) 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.

Sets debug mode that DSP variable vav each time it changes and the line number where the change occured.

23

NAME

END

EXAMPLE DESCRIPTION

110 END Stops program execution. Sends carriage

return and "> " BASIC prompt) to screen.

Begins FOR... NEXT loop, initializes variable var to value of expression exprl then increments it by amount in expression expr 3 each time the corresponding "NEXT"

statement is encountered, until value of expression expr 2}$ reached. If STEP exprZ is omitted, a STEP of +1 is assumed. Negative numbers are allowed.

FOR var= 110 FOR L=0 to 39

exprl J0expr2 120 FOR X=Y1 TO Y3
S7tPexpr3 130 FOR 1=39 TO 1

150 GOSUB 100 *J2

GOSUB

expr

140 GOSUB 500

GOTO expr

GR

160 GOTO 200

170 GOTO ALPHA+100

180 GR
190 GR

POKE -16302,0

HLIN exprl >
expr2Mexpr3

Note

Causes branch to BASIC subroutine starting at legal line number specified by expression expr Subroutines may be nested up to 16 levels.

Causes immediate jump to legal line number specified by expression expr.

Sets mixed standard resolution color graphics mode. Initializes COLOR = (Black) for top 40x40 of screen and sets scrolling window to lines 21 through 24 by 40 characters for four lines of text at bottom of screen. Example 190 sets all color mode (40x48 field) with no text at bottom of screen.

In standard resolution color graphics mode, this command draws a horizontal line of a predefined color (set by C0L0R=) starting at horizontal position defined by expression exprl and ending at position expr2 at vertical position defined by expression exprd .exprl ar)dexpr2 must be in the range of to 39 and exprl < = expr2 • exprZ be in the range of to 39 (or to 47 if not in mixed mode).

HLIN 0, 19 AT 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.

200 HLIN 0,39 AT 20
210 HLIN Z,Z+6 AT I

24

IF_ expression 220 IF A> B THEN
THEN statement PRINT A

230 IF X=0 THEN C=l
240 IF A#10 THEN

GOSUB 200
250 IF A$(l,l)# "Y"
THEN 100
Illegal:

260 IF L> 5 THEN 50:

ELSE 60
Legal :

270 IF L> 5 THEN 50
GO TO 60

If expression is true (non-zero) then
execute statement-, if false do not
execute statement I f statement

is an expression, then a GOTO expr
type of statement is assumed to be implied.
The "ELSE 11 in example 260 is illegal but
may be implemented as shown in example 270.

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

var2 3 str$ 290 INPUT "AMT",
DLLR
300 INPUT "Y or N? A$

IN# expr

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

LET

340 LET X=5

LIST nurnl, 350 IF X > 6 THEN
num2 LIST 50

NEXT varl 3 360 NEXT I
var2 370 NEXT J,K

Enters data into memory from I/O
device. If number input is expected,
APPLE wil output "?"; if string inout is
expected no "?" will be outputed. Multiple
numeric inputs to same statement may be

separated by a comma or a carriage return. String inputs must be separated by a carriage return only. One pair of " " may be used immediately after INPUT to output prompting text enclosed within the quotation marks to the screen.

Transfers source of data for subsequent INPUT statements to peripheral I/O slot (1-7) as specified as by expression expr. Slot 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 nvml through line number num2 to be displayed on screen.

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

NO DSP var 380 NO DSP I
NO TRACE 390 NO TRACE

Turns-off DSP debug mode for variable
Turns-off TRACE debug mode

25

PLOT expvl,

expv2 400 PLOT 15, 25
400 PLT XV, YV

POKE expvl , expr2

420 POKE 20, 40
430 POKE 7*256,
XM0D255

POP

440 POP

In standard resolution color
graphics, this command plots a small
square of a predefined color (set
by C0L0R=) at horizontal location
specified by expression expvl in
range to 39 and vertical location
specified by express i on expr 2 in range
to 39 (or to 47 if in all graphics
mode) NOTE: PLOT is upper left
and PLOT 39, 39 (or PLOT 39, 47) is
lower right corner.

Stores decimal number defined by
expression expv2 in range of
255 at decimal memory location
specified by expression expvl
Locations above 32767 are specified
by negative numbers.

"POPS" nested GOSUB return stack
address by one.

PRINT varl> vav 3 stv$

i i

REM

£ expr

RETURN

450 PRINT LI

460 PRINT LI, X2

470 PRINT "AMT= M ;DX

480 PRINT A$;B$;

490 PRINT

492 PRINT "HELLO"

494 PRINT 2+3

500 PR# 7

510 REM REMARK

520 RETURN
530 IFX= 5 THEN
RETURN

Outputs data specified by variable
vav or string variable stv$ 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.

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

PR#

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

Causes branch to statement following
last GOSUB; i.e., RETURN ends a
subroutine. Do not confuse "RETURN"
statement with Return ke^ on keyboard.

TAB expr

```
530 TAB 24
540 TAB 1+24
550 IF A#B THEN
TAB 20
```

TEXT

```
550 TEXT
560 TEXT
```

CALL-936

Moves cursor to absolute horizontal position specified by expression expr in the range of 1 to 40. Position is left to right

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 exprl 3 expr 2
AT expr 3

VTAB expr

590 VLIN 0, 39AT15
600 VLIN Z,Z+6ATY

610 VTAB 18
620 VTAB Z+2

Similar to HLIN except draws vertical line starting at exprl and ending at expr2 at horizontal position expr3.

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.

27

SPECIAL CONTROL AND EDITING CHARACTERS

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

CHARACTER
RESET key

DESCRIPTION OF ACTION

Immediately interrupts any program execution and resets computer. Also sets all text mode with scrolling window at maximum. Control is transfered to System Monitor and Apple prompts with a "*" (asterisk) and a bell. Hitting RESET key does NOT destroy existing BASIC or machine language program.

Control B

If in System Monitor (as indicated by a "*"), a control B and a carriage return will transfer control to BASIC, scratching (killing) any existing BASIC program and set HIMEM: to maximum installed user memory and LOMEM: to 2048.

Control C

If in BASIC, halts program and displays line number where stop occurred*. Program may be continued with a CON command. If in System Monitor, (as indicated by "*") control C and a carraige return will enter BASIC without killing current program.

Control G
Control H

Control J
Control V

Sounds bell (beeps speaker)

Backspaces cursor and deletes any overwritten characters from computer but not from screen. Apply supplied keyboards have special key "«-" on right side of keyboard that provides this functions without using control button

Issues line feed only

r
Compliment to H . Forward spaces cursor and copies over

written characters. Apple keyboards have "+" key on

right side which also performs this function.

Control X

Immediately deletes current line

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

28

CHARACTER

DESCRIPTION OF ACTION

Move cursor to right

Move cursor to left

Move cursor down

Move cursor up

Clear text from cursor to end of line

Clear text from cursor to end of page

Home cursor to top of page, clear text to end of page.

Table A: APPLE II COLORS AS SET BY COLOR =

Note: Colors may vary depending on TV tint (hue) setting and may also

be changed by adjusting trimmer capacitor C3 on APPLE II P.C. Board

Of = Black

1 - Magenta

2 = Dark Blue

3 = Light Purple

4 = Dark Green

5 = Grey

6 = Medium Blue

7 = Light Blue

8
9

10
11
12
13
14
15

Brown

Orange

Grey

Pink

Green

Yellow

Blue/Green

White

29

Special Controls and Features

Hex

BASIC Example

Display 1

fade Controls

C05O
C051
C052
C053
C054

10 POKE -16304,0
20 POKE -16303,0
30 POKE -16302,0
40 POKE -16301,0
50 POKE -16300,0

C055
C056
C057

60 POKE -16299,0
70 POKE -16298,0
80 POKE -16297,0

TEXT Mode Controls

0020

90 POKE 32, LI

0021

100 POKE 33, Wl

0022

110 POKE 34, Tl

0023

120 POKE 35, Bl

0024

130 CH=PEEK(36)
140 POKE 36, CH
150 TAB(CH+1)

0025

160 CV=PEEK(37)
170 POKE 37, CV
180 VTAB(CV+1)

0032

190 POKE 50,127
200 POKE 50,255

FC58

210 CALL -936

FC42

220 CALL -958

Description

Set color graphics mode

Set text mode

Clear mixed graphics

Set mixed graphics (4 lines text)

Clear display Page 2 (BASIC commands

use Page 1 only)
Set display to Page 2 (alternate)
Clear HIRES graphics mode
Set HIRES graphics mode

Set left side of scrolling window
to location specified by LI in
range of to 39.

Set window width to amount specified
by Wl. L1+W1<40. W1>0

Set window top to line specified
by Tl in range of to 23

Set window bottom to line specified
by Bl in the range of to 23. B1>T1

Read/set cusor horizontal position

in the range of to 39. If using
TAB, you must add H l" to cusor position
read value; Ex. 140 and 150 perform
identical function.

Similar to above. Read/set cusor
vertical position in the range to
23.

Set inverse flag if 127 (Ex. 190)
Set normal flag if 255(Ex. 200)

(@e) Home cusor, clear screen

(Fe) Clear from cusor to end of page

30

Hex BASIC Example

FC9C 230 CALL -868

FC66 240 CALL -922

FC70 250 CALL -912

Description
(Eg) Clear from cusor to end of line
(J C) Line feed
Scroll up text one line

Miscellaneous

C030

C000

C010

C061

C062
C063
C058
C059
C05A
C05B
C05C
C05D
C05E

C05F

360 X=PEEK(-16336)
365 POKE -16336,0

370 X=PEEK(-16384)

380 POKE -16368,0

390 X=PEEK(16287)

400 X=PEEK(-16286)
410 X=PEEK(-16285)
420 POKE -16296,0
430 POKE -16295,0
440 POKE -16294,0
450 POKE -16293,0
460 POKE -16292,0
470 POKE -16291,0
480 POKE -16290,0
490 POKE -16289,0

Toggle speaker

Read keyboard; if X>1 27 then key was pressed.

Clear keyboard strobe - always after reading keyboard.

Read PDL(0) push button switch. If X>127 then switch is "on".

Read PDL(l) push button switch.

Read PDL(2) push button switch.

Clear Game I/O AN0 output

Set Game I/O AN0 output

Clear Game I/O AN1 output

Set Game I/O AN1 output

Clear Game I/O AN2 output

Set Game I/O AN2 output

Clear Game I/O AN3 output

Set Game I/O AN3 output

APPLE II BASIC ERROR MESSAGES

*** SYNTAX ERR
*** > 32767 ERR

*** > 255 ERR

***

BAD BRANCH ERR

*** BAD RETURN ERR
*** BAD NEXT ERR

*** 16 GOSUBS ERR
*** 16 FORS ERR
*** NO END ERR
*** MEM FULL ERR

*** TOO LONG ERR

***

DIM ERR

Results from a syntactic or typing error.

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

A value restricted to the range to 255 was
outside that range.

Results from an attempt to branch to a non-
existant line number.

Results from an attempt to execute more RETURNS
than previously executed GOSUBs.

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

Results from more than 16 nested GOSUBs.

Results from more than 16 nested FOR loops.

The last statement executed was not an END.

The memory needed for the program has exceeded the memory size allotted.

Results from more than 12 nested parentheses or more than 128 characters in input line.

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
*** STRING ERR
RETYPE LINE

The number of characters assigned to a string exceeded the DIMensioned value for that string.

Results from an attempt to execute an illegal string operation.

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

E000

C000

XX

7FF

64K

56K

Monitor and BASIC Routines in ROM

Future enhancement or user supplied
PROMS

52K
48K

Peripheral I/O

XX
(HIMEM:)

» User specified RAM memory size

User Workspace

(LOMEM:)
2K

IK

Screen Memory
Internal Workspace

33

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 sequentually are the Variable Name, the Display Byte, the Next Variable Address, and the Data of the Variable. Diagramatically this is represented as:

VN

DSP

NVA

DATA ((?)

DATA(I)

DATA(N)

h n+l

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.

34

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(Q0 DATA(l) DATA(n) ST

1 hi h 2 h n+l

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 la illustrates this process diagramatically. 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 lb) 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 c command both memory pointers are initialized indicating no program statements and no variables.

To illustrate what a variable table looks like in Apple II memory suppose we want to assign the numeric variable A ($C1 is the ASCII equivalent of a with the high order bit set) the value of -1 (FF FF in hex) and then examine the memory contents. The steps in this process are outlined in example I. Variable A is defined as equal to -1 (step 1). Then for convenience another variable - B - is defined as equal to (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 ($8JBC). 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

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

READ/SAVE ROUTINE

The READ/SAVE subroutine has three parts. The first section (lines 0-10) defines variable A and transfers control to the main program. Lines 20 through 26 represents the Write data to tape routine and lines 30-38 represent the Read data from tape subroutine. Both READ and SAVE routines are executable by the BASIC "GOSUB X" (where X is 20 for write and 30 is for read) command. And as listed these routines can be directly incorporated into almost any BASIC program for read and saving a variable table. The limitation of these routines is that the whole part of a variable table is processed so it is necessary to maintain exactly the dimension statements for the variables used.

The variables used in this subroutine are defined as follows:

A = record length, must be the first variable defined

CM= the value obtained from statement a of figure 2

LM= is equal to the value of "LOMEM:"
Nominally 2048

SAVING A DATA TABLE

The first step in a hard copy routine is to place the desired data onto tape. This is accomplished by determining the length of the variable table and setting A equal to it. Next within the main program when it is time to write the data a GOSUB20 statement will execute the write to tape process. Record length, variable A, is written to tape first (line 22) followed by the desired data (line 24). When this process is completed control is returned to the main program,

READING A DATA TABLE

The second step is to read the data from tape. When it is time a GOSUB30 statement will initiate the read process. First, the record length is read in and checked to see if enough memory is available (line 32-34). If exactly the same dimension statements are used it is almost guaranteed that there will be enough memory available. After this the variable table is read in (line 34) and control is then returned to the main program (line 36). If not enough memory is available then an error is generated and control is returned to the main program (line 38)

EXAMPLE OF READ/SAVE USAGE

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

CONCLUSION

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

38

Var

Var2

t

LOMEN
$800

V^.

Var,

Unused
Memory

Pi

p 3 ••• Pn-2

/ t

CM End of 'PP beginning

Variable of

Table D

Program

Pn-1

HIMEM
Max System
Size

Variable Data

BASIC Program

Figure 1

a) PRINT PEEK(204) + PEEK(205)*256 -»? PP

b) PRINT PEEK(202) + PEEK(203)*256 -* CM

Figure 2

800 801 802 803 804 805 806 807 808 809 80A 80B 80C
CI 00 06 08 FF FF C2 00 0C 08 00 00 00

VAR DSP

NAM

L H
NVA

I

```
L H
DATA VAR DSP
NAM

-* 1



L H
NVA



I



DATA



i
```

Figure 3
$800. 80C rewritten with labelling

39

FIGURE 4b

```
READ/SAVE PROGRAM
% A=0

10 GOTO 100


20 PRINT "REWIND TAPE THEN
START TAPE RECORDER" :
INPUT "THEN HIT RETURN"
```

COMMENTS

This must be the first statement in the program. It is initially 0, 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.

```
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",
BS

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

34 IF A<0 THEN 38: P=LM+A:
IF P>HM THEN 38: CM=P:
POKE 60, LM MOD 256:
POKE 61, LM/256: POKE 62,
CM MOD 256: POKE 63, CM/256
CALL -259

36 PRINT "DATA READ IN":
RETURN

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

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

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

40

1 >A=1
>

2 >B=0
>

3 >PRINT PEEK (204) + PEEK

(205) * 256

computer responds with=
2060

5 *800.80C

Define variable A=-l , then hit RETURN

Define variable B=0, then hit RETURN

Use statement 2a to find the end of
the VARIABLE TABLE

Hit the RESET key, Apple moves into
Monitor mode.

Type in VARIABLE TABLE RANGE and HIT
the RETURN KEY.

Computer responds with:
0800- CI 00 86 08 FF FF C2 00
0808 0C 08 00 00 00

Example 1

41

Example 2

>LIST

8 fi=8

18 GOTO 188

d6 REM wait uhir iu mrt Kuuuiit

22 R=CH-LH: POKE 66,4: POKE 61

,8s POKE 62,5: POKE 63 f 8: CfiLL

-307

l4 ru!L b^iLH nliU £3be i"Ur-.L bl

frttb: PUKE b£,Ln flUI/ cOb

: POKE 63 5 Crf/?56; CRLL -38?

118 PRINT 3 26 NuHBtRS GtHERRlLL

iHL DHIfl"; PRINT H yNLN YOU fiRE R

COtW C7QD7 ~ur nr :-?!-{-»?,-]-: tl- n""

L.JUM JiiiFit lili_ ?£. , _-'JiL-*Li". ii'i £L'--"_'it

D HODE

nil KL1UKM'

ijb bhLL "ooi rKi.Nl "Os sKillHU Drt
Tfi TO WUi GOSOB 28

```
2b RETURN

-iu PF* prDfi RpTfl- CnDprsliTIMr
Oc r>Li3 Ri.fiL- fcTui! juuP.uUi tii=_

32 POKE §3,4: POKE 6!, 8; POKE

o4 if Hb sHlN oo;t"-LftTn; ir r/
HH THEN 38:CH=Pj POKE 68.LH HOD

,CH HOD 256; POKE 63,011/256

: COLL -259
36 RETURN
38 PRINT **** TOO iiCH DHTH 8RSE *#

* H : EHD
188&IHMCt) f H28>
185 FOR 1=1 T0 28:X(I)=I: HEKT

T

i

188 LH=2848:Ci?=2l8b:B=58;HH=i6383



int ^.cb; ihbLt nWJ Ktni> lilt L'fl
Tft FRGH TRPE"

i-jfj sub i-i :!= i-w.rA iy-bl sKini

B K(*;I; 5 )= s ja(I)e NEXT I

J IHrui nnu ihLn nil r,Llu?.n
M

iC-. DD?'.!7 ='G £ Q

i'-'-J j ii:J! it sfl

J -i .-. .- n. .— : : p. .». .-.

198 FOR 1=1 TO 28; PRIHT "K< re 1 1 j

195 PRIHT 'THIS IS THE EHD 8
288 EHD
```

42

A SIMPLE TONE SUBROUTINE

INTRODUCTION

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

Tone Generation

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

Toggling the speaker is a simple process, a mere PEEK - 16336 ($C030) in BASIC statement will perform this operation. This does not, however, produce tones, it only emits clicks. Generation of tones is the goal, so describing frequency and duration is needed. This is accomplished by toggling the speaker at regular intervals for a fixed period of time. Figure 1 lists a machine language routine that satisfies these requirements.
Machine Language Program

This machine language program resides in page 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 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.

44

8 8 8 S -

FF

8891-

FF

8082 —

RD

30

6885 —

o o

d C-s & £ _

r —* *LJ ^_' =_ :

D8

84

@ 8 S 8 —

Cb

81

Ca d M Q

i..* kj kj n

88

888C-

Cfi

088D-

D8

F6

088F-

fib

88

0011-

4C

8£

8014-

68

l:h LuR $C838

hh

DEY

BHE

^80H1J

L.-* L_ =_-

-Sr-Ts i

BEQ

£ 8 8 1 4

DEa

a- 1 ' 1 L_

^pjfjfi^

LDf : :

£ 6 8

T M O

?ir i~~ C-i Ci o

•*?' *-.= r _= *J L...

R

TC

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

ottoc r^ht LfUJl rm. 6 t *6l rUKL

9,1: POKE i§,?46

poke ie.pj f

£8,96: RETURN

. } K!T:i- lU*i,t rUFsL i7sUs K'f-L

FIGURE 2. BASIC "POKES"

ca riat ftjf; rmt i,us U1LI
RETURN

FIGURE 3. GOSUB

45

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 scan access these subroutines by use of , the CALL statement, and can pass information by using the POKE state-Bent. 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 $DfT{T0)

This subroutine sets High-Resolution Graphics mode with a 280 x 16? 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 3B86 (or CALL -12274)

From machine language: JSR $C0E (or JSR $Djf0E)

This subroutine clears the High-Res61ution screen without resetting the High-Resdlution Graphics mode.

PLOT Plots a point on the screen.

From BASIC: CALL 378JBT (or CALL -1158&)

From machine language: JSR $C7C (or JSR $D07C)

This subroutine plots a single point on the screen. The
X and Y coodinates of the point are passed in locations 800,
801, and 802 from BASIC, or in the A, X, and Y registers from

machine language. The Y (vertical) coordinate can be from

47

High-Rcsloution 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 (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: 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 -11S99J
From machine language: JSR $C26 (or JSR $D(f26)

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

LINE Draw a line on the screen.

48

High-Resolution Operating Routines

LINE Draws a line on the screen.

From BASIC: CALL 3786 (or CALL -US74)

Fron machine language: JSR $C95 (or JSR $t>(?9S)

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 sane manner
as for a PLOT or POSN. The color of the line is set in location
812 ($32C). After the line is drawn, the new endpoint becomes the
base endpoint for the next line drawn.

SHAPE Draws a predefined shape on the screen.
From BASIC: CALL 3805 (or CALL -11555)
From machine language: JSR $DBC (or JSB $D1BC)

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

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

SHAPE (continued)

1 will cause the shape to be drawn true sire, 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 f.r the shape. A rotation factor of 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 foom BASIC of in the A-register
fron 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:

In I i 14 !»)*!» ! £ I i *

OiOPD *' 00 ! itf>" >

I 1 4: »rr A : ~T » •• <r

Each bit^ir'DD^ptcifies 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. IThe SHAPE

SHAPE (continued)

subroutine processes the sections from right to left (least significant bit to most significant bit). IF THE REMAINING SECTIONS OF THE BYTE ARE ZERO, THEN THEY ARE IGNORED. Thus, the byte cannot end with sections of 00 (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 distin guish the vectors that

plot a point before moving:

Now "unwrap* 1 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 00 at. the end of a byte, then skip that section and go on to the next. When you have finished

SHAPE (continued)

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

52

5 V,

*.p

v e c t~ «> r 5

O^^^^itf -7 *">»-* *7^^*<*

<0
I

Z
3

1

S

c

7
8

o i <s5

i i

o o

O I

oo
o

l o

1 o

oi l

OO o

• o

boo

t_

A

C s A

o <S5

* *

1 1 1

«*<•

OO O

ti

l o o

^t*

o l

^

» O l

^-*

l o

^i

1 1 o

<r&

1 l

«•

o o o

?

5tf»cT

CooCS

*J

<- E~t|»4- t ;

05 i a

a> i i

I i ©

l I I

I

1 1

J

t-rV

o /- c

or * *A»^e. Op L^)

1 2

3F
Z <f>
64

2 O

I 5

3 (,

<8 7

F i

o o olopo o

oi r t.

O oo o -} o

O © O V -* i

o e> t o "7 t.

o o i -* 3

i o o -7> ^
o i o I -} J
O I V O -7

° I I ~7

1 o o o -^

-7

7
B

I o I o

O O -^ C

I I O 1 -y t>

I 1 I ft -7 E

Il Il "* F

. • « = • -06^ Cs vLL"?.-0yOG; rU3n-0; O:

, : ??••" a-. : i

-J -I s 3 if &«£/ wv^Kl I lt£.=1 j^'Ur.L CO

— ¥DIR» Y=— Ys- IF Y<8 IHEH Y-Vt

7£Q : ~i~ pnii'T 5£& id.^* PfWF :":« =

elB . XU)=(n(I)-?!)*9/18fV; V(I)=(

8b IfiLL iNi!:X= KHD ^4)*!^
f RHD (3)*85*85; GGSUB 2888

s bHLL ruUi

?IU II- k'HD (1888K1 IHEH 388= IF

Lfi."^„ ??-"• i

•: s_L.if : :-: : ll 1. ~:i"_^:! _??••

bO^s LULL bHHfxi; fiLhl Kl bUbUu

3888: GOTO fel8

ru-r-. .-• •- -. i -LLic. rr ?Ji/ck nn

•=?=/ js iJ*iJl if AivS UK

tl)u? OR ¥I<8 OR YDI59 JHEH
-I'-i* K — K I = I — T 1 1 bUbUt? coBH; UiLL

•_in-_ s uvj'.«= jy-j-j s yysy yi»

'His UUiUS IssSb ^UKL c'liis KHU *

'.; :jUKiv^: i"Lj: ; Lf: :";T

rijf.c nn«

??L' "i.'vH- 1 L"_ •?=_*:: : •_• _ : ? -J:

"• 1 :.- " . i i-:i -j

--»» :=i. t

i J -j -_ : _ : '_? L ! ~ L ! %.' "? : «J '_ : J V U

: J'.=u OfJi_L =_-i

r:/T , -0-_:~Q.j: e— £f?L- : : -. 0/~C T r.l

riifL? c50i ruKi

Fi|i":.-L-Jvt i l?p.s_ yyL|i'i;J t-iiLL

'&)v'j>' 1

o;?Vintoi^AU>-£: HtX! I

bUo UHLL lHiiz HIIKt HHH,Hj CBLL

:i:- : ::: :

! : v-t-i. i.i ; ^.r t. :-••_? • -•-?

i^. - ? ". . - rr - -. =

lOIV- f:~-7- i si I- \ i *; UUjUD l.uuus "LiUKfl

£m$ l^U^t bl^vsA HUD c^n; FijKt HHi

?JHHH |h PttK ~lbou4)ur8 THEN RETURN

b : Hr f Y= tHLL LiNtz HUKL b ! yy?
(279-K) HOD ?5£' POKE 881 5 K(

POKE 888,23; POKE 881 ^ 1= POKE

OC:^ i=-ii_v: rot I I TMT

l-:^^ ?• l_ij:

; r- ^|:KJ- Hrjkr Z ri)l ^^K= Kl't?!- >=il
-v J '_':-.^ «y«jn ;s-_-L- j.-_-_*a 1 -jr-.i. v v i

fA/C-J-Js fUr-.L OSCe-vJ L-llLL Lifit!

K-K^k : DIR^B: IF K/-8 HNO n : <388

L s "* ill. 1 ; L -J"-. L. , ; » i7i_j": : is !V- ! ! J!£!

54

ROD'S COLOR PATTERN

PROGRAM DESCRIPTION

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

REQUIREMENTS

4K or greater Apple II system with a color video display.

BASIC is the programming language used.

PROGRAM LISTING

V& 1-Uk y=3 iO i8

1 !-'t :; i s : i L : i ::

:Vi. DIP" i:~- ?• Au T <~.: .-.- ?.. ^ - _

• - Jr Tj '.'«' 5 s- : -'i i»~ f J r.= i

55

PROGRAM LISTING: PONG

?:t-i/Ui_ -Ji^i. i«

i.v. it i-iJJ

L.v If Hi : -J J s vis: ;:: ??'-. i"i.

uLl-;---J: ••L.i-': • : -J-J ::? « s • SL. i «

-/t;-?-t -V

-IH i.iH Uk"=K: H :i; a-I.-Y

f a-« lilLn t I U~ f=Dj ? /

H.i n-.*t~~n; []-.-:?

Uliil? C=-i

intf ITnTSi

NL^I

?v.* i =!!_!! _'«-_'??.

eJ3 ir H ifttn «D5m/=U rui {

i!M UULUk=b= VLIN P(l>,rI^j Hi

??I7I i|j:i -lir^S 1 \' r Ft i j :-Kt* '-: i J'^S)-

}iV6) IHLH VLIN PiI;tSt|.^4
RT39sP(3)=p{i)

is If
s

Ss :*ni HD-U: Tl DfSWOs'O'i TiirB

VLIN u=riy)~I Hi B* ih riJ)
{'Pf:-'"i TMfii Uj IM p="gUs4-f ^Q

III V

UULUk = »= ih H(U)>Flu) ! HEN

: _ I : i ii Li j. ! : :.?: ;_: ii : G: IT 0:' Ci V
TLi-1 U«*V/ i i:i « a ii JV/

TCf i OCr- VLili P-.** /i" Z- "=" is -j^

s-Ti, -nr^ r-.i: ;.;?

: r- X-ZiH j™ '^rr-:7; ? Hr ."i ."h,™

?v i v?"-. :~i ii,' i^sij- i L.L.: *?, iUw^ ! .

•?j i- •• •:=-?/ :_-.*". ; !?==_?? i-j-j

?;v/; ii :"-'_' iiti.n : : ~S '-. i

!hui v-i-s i- i"ii : l- •??s-n

?™;: i -:/ :+i

56

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

```
z. pfigT :- ! f?Q* Df-iT jiQt pfii-T
« fui_- *_.,_; 0. !'_»%_, -Jj'iQ. rUM.

Tiijfi-i rUf-.L -J. s iC-J« fUKL Ds«

s PfiiT 7 *'"-'= pHiT i£0. DfssT
e « >jFii. ; «-js_* fur-.E- i-^i-wi t ur-X

•/,i_-^ ; Tui-.L _UrlO-j s fUM. ii

,1: POKE i£ f £88; POKE 13,4 "

ig pf.iT- *--- iOO B DsItT iK OA* Dfi.T

iGjCto; ru?-.L _;=;). fUf-L lo'j

xttL-, ! v'lil i7.ii ! iff-.*. L-Vii'.'i

POKE 21,2: POKE 22,8: *POKE
23,96-

j^ f-Ts 3e/i{S i. ? T_"v7 ; i"_H e _00^

iv _-*<t L** VTC'/i ii_r=! ? '.-Hi.-. J.JM


C-J H- LLHDt/; i UK _>i iy ill Uu^Uu

hV PRINT Ptf? 7^" MrzT ? :


?~:lT H J- _ s -""? •"• : ~*~ .— :.-r--9-. - .; : is nr-irt ST-.: :

AQ Dt-f=rnDvD1T;UT ilDO; ~ rHMDIItrD iG7

45 B$="TH1S PROGRfiH fiLLOyS VOU TO s


POKE UOH HOD £56: POKE £4

f i ! _M:- Lt ,: J : ii E L'Pii. L ! $r-."-. i l _-i-_._-


W.GOSUB 38: GOSUB £5; PRIHT :
OSUB 38; GOSUE £5

DD S-. itnCiiD .48. f
```

?•L' -Jt •iUJ'Jl? "J i '.

_->?-'

TiT: ? LET

uOSOB 25= PRINT « GGSUB 58
uu.'Ud c-Ji rr. ins
i GGSUB £5= PRIHT

i :-.ni: i i ;-.i)ii i « '.- J U" iJ S V Jt *j1S '_? t
~ ::.:-!-!: rir^riv SETT rirT:i-i:!l fji-

i Tup 7*0 a". a pDTUT D-t: ; 7if 7i

1«

14b

C£= SCRK(X f Y):C3=15: IF C£=
15 THEN C3=5, C0LuK=C3f PLOT

Etnin «o^

IF PEEK (-16384)#1d@ THEH 15:

iP-Gst-5: piiVV _i£0£0 Q; DfivT

39: CRLL -936

QDTliT « D-t-_«|- -;"— -lee j— Af} rr-j-g

: VTRB £4; GGSUB £5: IHPUT

u r -* / ? D* = if b* I j, i /- L-

THEH 118; PRINT 5 END"; END

; UKt -I636S.8: GOTO l£b

58 B$ = ~SkETCH COLORED F IuUREh W

05 i;^-' : LUy KLbULU ! llliH bkHPHllb Hi i H
PQnoi re** opTiipM

<_*-_= r-.r-.-iu. iuEi-i'.'s -vjj'jw l'-.'s r-.i_s-.T-.il

7S k$z- : &: tE^y-Hti: Ce"eCeeD GRi^iV-O-i

75 KK=£B;T0H-£6^ GOSUB 35; RETURN

HOi rLBG THlN I3j; COLOk-C

:iE*j-j^:4- i.nLEii- -J-.t Ul i U-Jii

58

MASTERMIND PROGRAM

PROGRAM DESCRIPTION

MASTERMIND is a game of strategy that matches your wits against Apple's* The object of the game is to choose correctly which 5 colored bars have been secretly chosen by the computer. Eight different colors are possible for each bar - Red (R), Yellow (Y), Violet (V), Orange (0), 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.

59

PROGRAM LISTING: MASTERMIND

J. WW ! :P.t'L i:-JL- -uw 'is : tl. ! : fv : • I

w / si.'". -J .-' ;;":?. w ,

s AV. .- : ~-Ji A-?~ L : =ij ; ? -jfe&

8; HUH 8*39 RT V;FlHSH=h FOR

Pi - 1 I U -J I m =1 ,- - ; uU jUD 1 ub«

= NEKT H*H=1
0££ pf;p yDTT = - TO. iiki'vTS-- Dfri"

www :L-i-, 3?;ti : i ? -_• iVji-Li" * US. P.

XMijR P;[N fiRLL ~^S4 ^FT^ INV^i^F -IF)

s.-i.-. rtr-i;

jww* KLh P0KE~lb368 CLRS KBD STROBE

ww tw fit-i: w: ::„=_ _*ww w=-UE*_; -Jw?:i.i-j- »i»l/

hl iu ifiL UH11L Ur nniiLKni"!/!

uu?, iiDOLwi i j iu uULio D wULU

niriun HUllHLi Oh uULbbtbs iHlK
r ddt rirrur r:?L~~Q~!a7 c-r-.i adc "=

i- iit-.u ^iwiii ^i--^ti_i;= vw=_w:_= iw

*_• n u 1= l r ft u n s

- uUUL?"b ^K!Hi " is :U ii li

l~l iv w* ii i-.j- ? ? / :•_}•.-"?. ss-r"- i .'

) 1HLN NLK! 1; ir I z ^ THEN

j~iiwr HLXI yni i ; hLHHH = I ~FlR5H=

4MM wULUk^Id^H^w! ["UK I = i !& h'

i: ii •!-?_! is:

ti? Owe. wULUK :

I >U j; NiSf

;; ir mu IHLK

fe HrtU iKKiw Itttfi t-'K 1 h I 'liUOU

4HHH Ktr_f| S = nIS 16 - 58 INTRO

£;i "::.-: wi. M CTbsTC ": -i is _ "• : Q k-ETU CT^IIw

-wiw ai_u ji-iij iww ii'j nL?? 3L = UT
'iwCb rlLli blPll Cww HLs UUL^b

4838 REH STMTS 388-318 USER INPUT

AaA& pfis C7=? ^^^ r;-rrr n-rt-
iSiLii DCis CMDO i&&& vCil HD • TsiT

:www :-i_SE _»Ji.Ti *www WUu.w"-. LiiTi-

om m n rKlHi

u s: : : : :i:

iiil ii::i Pii. ! iw LJL-Jin :L::I

ivv wHLL ""OOO; If rtC! ™ib^ST/

16L :HLN 1ww= FUKt ~lbwbSj
Lis HP - pDTMT : ChD T-i Tn

wi w:-. t)!i!1i 3 ? Vi i~i !=_:

Q^r^T^- vUh /OU'r rrii !">.«•
u=wi-- v-.-vt- u-=i, wULUi-fi

i)= iillN I*4*c«I*4 BT 39* PRIHT

;:":•? i ^ i / a s : ti_n ! i

t i f- T~! ! _ ! _ji- nri T:! ~ r.r.r:^ s ;__-.__

ii^ iivi-n: hfe in; : Thin! " LLIILK

KEYS FOR COLOR CHflNGE"i PRIHT

HKKU= Rty^i I" UK nUVrinCL fiW Wd

Zk" i PRIHT 3 HIT RETURN TO RCC

H BVLKfiGE 5 |i Ir 1 RV>I4 iHEn
-384;' IBB 5i PRIHT £ HIT mi KEY

iw r^_;:- iiw-ii:; s •>*_» i 'J IVV

J. www 11" N~o it1t.:1 KL I UfiN s w'JLL'K"

K(H(H))*FLrlSHi HLIH N*4-£,H*
£888 IF H(IK>P<3} THEH RETORH ;

£i-«i- = Dl hT Oiii-U" V= DDTUT

5J-H T ir !!_»_'! L.1 • I! ' i> j ! * !iil!?

?•-!— r i::".:

' ; : HU >-B; L«. -i i=55 st i UKN

60

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.

Bio rhythms

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.

61

PROGRAM LISTING: BIORHYTHM

uiWt 170 = DsWl 0. £0. Dftfr™

4«I^i* HUKL t)j 16^?; HUKL b>«

e DH-T 7 00* DHVr iiOs DfliT
* i -*r-.s_ >,«*.* i v:l y f *.*«-'« ? lt-.j-

» Ok0 = Dni^L iQ i£n = Dfii/r i:

s s i. : Mi-'L t "J "_ : L'£L?s afiisL i'j £
I '-;'-. L. iTji.'vi t ! «?F>.L. i-_'ti_ :i t UF-.i.

198* POKE 19 f h POKE £8 =76=

UU 1 U oD

tt_-!. --j-Hri "j. nr-jiriL:

« 'j ; =_ ! rv.i_ issR iivL' i_-js= t yr-.i. i_- f
a DlTMDU

,H,DjYii = Y , KY(I88}*I988

7_3/H_V*: J s!Yj'ti.tja TC 11/ Li TUlU
i Ii' v -J* 1 , ii/i. .' 'L-r U !«?-.« ISii.it

Ji-U-iOiOSO. Dr?=iQk:

ft— -1 ! LsLvL: F>.i_ • Ut:l
^ HIN MtfjR^ Rl^ "'!) RC'"*-': Pi' '-:'":

L? j v .-" i ii i :"vTJ t LA S_ .- ~ S-*jy » I

POKE 34»2§; GOSUB 28: GOSUB

J=I: bk I rUKt y't.c.j. "Uf, n-

18 TO cB: COLORE: HLIH 8,31

QT vi LOT Vt Lll Til 1 QT

Mi r ; . nuA! fit SfUill ifv M!

Hi E: ?TBB 21
Vj; IF Y<18 THEH PRIHT a ~i

I rKiHi ii sii-ni

T; FRIHT

1E9 ¥TBB £3, PRINT "DfiVS LIVED s

,i:. rr-.u ?- in Os ffiinD-ii^

Jn; ruK i~i fJ .?-?? ^jLurri-.

I=lHb*(i=£HS*(I=3)s ?LIN

'- '"0 ST OOiT + Ts UTGD 0.4

SfOJ HI OO'i + i, 7JMU E-T

iilj l"UK n = ^ IU ^i^r-ln nuy d?m/

+K) HOD BV(D. GOSUB 56; FLO!

tD s KtiUKN

-?-' i -Ji': 1 - i s i Ui ? HUE/ Lv ! Jj : U:i_ L.T

;0H/£5&ti= POKE SrKK= CRL

.; : i ;y r : f : In" /;A i ;;••?% r - ?* rlrl^

"; inb 13; mH!

DDTJJT HDTDTU n S5 rnciiD 7k
TiHD LLl ihQ Cli ^fLi^! Diiin

138 PRINT i IHPUT "HHOTHER PLOi CY/H

i S-svs TUCi

itB

i /' i5=j/

Lj — -44-: J-'jS * rlrr I- -litii : ; "; ":iki "41-! - :' : i- : s

i rid '":.;. :-?-•? T • -. .• t n-Fi .!.-;.• : '•- -? i r^ -. E

H-3S*(rl)39)^fi^(fi(48)i RETURN

.05 PRINT "FORlCHST ~i; GOSUB 75

. t ii ;r-.»_= iifLSi «i~it l t_ *•_?-«?_

' Vlhb CjI |Hn ib" e KKlnl TUKLtu

ST DhTE a ;fi; H , s ;Dr f E jY: ¥THB

t=. vi/_7 t Ts-:£ s cnCUD 7S* pCTiipU

?_ s _ : r-.F-. - - a tl: _ i«* U-J^IUU !Ut t\"_ s UF-.i"

62

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

```
2 PRIHT 5 yELC8Ht TO THE DRhGOH'S rf

hLl.1

3 PRIHT- s tOU Mi iRTCB HHILE I BUI

i nn"i|T ? sn-rr nyrn i~r rri»ni rrr r
f hKini DL?= SslUs ii 5 LUhfLLtLj 1

; LL ERRSE 5

•J i Rifii iiiu !*.*_•< UKi_x ii iL.it iiib l.i_

UfiLT itt nl milt* ru ^UU bUflr 1

FT. ini id e!Uvl.;-IUO Hi I i flfo

RIGHT."

7 DpTUT 3E s 3 ?? CHD ilTT MM tHD MD
i s tit?s l. i ur-. ?_!_! it y i yr-. «j; •

HHiT ;
DDTMT «Jri' CfiD iSHiikl ?•& M^T UT T

iQ DDTkIT =TUr flBTpPT, TC Lug vHH f?U

t bKLtn »0T b
[1 PRINT 2 Tp G'M T^ TuV nnn^ HM w

li : tii:!: ; 'j -Js- i I -_• t s ts- :,• '•.•-«.':?. ?-?:! 5 Hi-

nt ^H.T -JTi-fS

IS PRIHT H BEF0RE THE DRflGOH (THE RE


Cb ^rUilt I3 f L?Lh ttLrifat iUb LHtl J)

Et ii, b?

L.i rriiFit tilt Hi'.-i ndl. JnC i-KsiauH
CBH'T GEP

cc PRINT "THROUGH IT! ) s

8S Din H$(3)

,; ihrui n*
l68 OR  C0L0R-15
```

-r,-.r. -' r.r- : T-r,f~-T =.-.-.r-.tr r — i .;.-.::

j! Ihb' d) "Kin! uHKl ^, jHhn

HOH s

i i -j mn

v t jj Hi il HLIH y,^7 Hi il HtK!

T

I3S

i u£"£

1881

I - UK. 1- i i y i ? ; ~i 1 i- i i r nt A i

1898 Q=R+D+Ltl)

1180 IF (0(3 RHD RHD (IB He) OR

£-'} INlH iii'u
III» l/R = KH& 4;
1128 GOTO IISB-WDR

f 1 * Y~^tl

iiuv ?t_iti -J-»"i L. f g^! i Hi -J-*"-:" i .-
i i'j-j ±*j S li i ?Juif

i 1 .4 D TC UHT ft TiirU 1 t iS= rs^i/ ~%i V

Tl8;Y=¥*i

ii-iK L-i T LI 0~v_0 QiV_1 ST 0~/v_i

I 146 GOTO 1835
: unt i

nu > L

1155 ¥LIH 3*Y-2 f 3*i-l nT m,

i i -JD UU i L= i & 'J-J

iiyc ir nU! u InlN iiiS;au-i^; :

llb3 HLiH ^X-c,3¥K-l HI '6*U bUI U
183;

14 PRIH 1 " BEiRRE! !!!!!;!! SOHtllHtS

CRH'T GO OVER*

):S(K)=- BBS (H(K));C=C-i
t l w • •?;= turn i «£ s t o-M; j/i. • •

8

IF Y=13 THEN 1878;H1M3>

11% 1=^1= bUiu iH3b

? jQzk rnCiiD ^5ii|5s ppTMT ; =TUl M07C T-

ALL"- 1 UvJUU vvvit fr.iin t^i- Hi-i-L. A.

EflDY"
1285 GR i C0L0R=I5

u i r%ih Hin!= iUU L-tiii v~ it-ii ILL

mMl n sHLL"

?si ? ~ zzi

ir R=i ihLR i83Ba=riU-W/S

iSoo ir i=I irttli ig7b;U=nu-ij;/

'ftl HLiii H,X> Hi o: HLiH H,

OQ OT -7Q
•j j tit -j/

1 CCS A- i ; i - Shy i-j Jt i ; LULUr -0 «

64

DRAGON MAZE cont.

1516

c$m

£828

£116

UY= RHD (13)4-1

"?ft: r-n_^. ••: :•* -(j.!!-..! -*": -':^i:U i

IrULUS— £ij ?i,in O'hsM ~c^ 0*3? ~i

fiT 39

C y — * j i C V — y V

K~ Till--. K-'lQ-itfitl tr *Aliib !t1t_rt

1568

```
Q8=K; GGSUB 7888; K=QS
IF 5X=X m SY=Y THEN 8888

II" R= ICmA"K'J InLn C&jv

ir i.-'_ £?-{-/ e-'« ?•. r-iry ":r;A3

ir ?,- rtoU U i intfi oM
IF K= BSC( S D 3 ) THEN 3588

r.j : i r. it. .-,

yft=i:uf=B

4888

FK=3*K-£iFY=3*Y-c; FOR 1=1 TO



FX=FX+DX:FY=FY+DY

hUR K=s III I  hUR L=b !U I;

D: AT Ur'i.V Livi.ii UtVT i ifs.. rf'i AD-

1; PLOT FM, Fitl; NEKT L,K:
NEXT I



- : ?  -. ¥-Yii:V: V-V iijV

ilih it £=io mv 'i-m irttN bti*Jtf
£128 GOTO 1588

£518 IF H(X+i3*(Y-i"M> HOD 18 THEN



2588 GOTO 2828




3588 DK=8:DY=1

3518 It H(X+13*<Y-1)>/18 THEN 438b



4818
4828
4838
```

. ,: .'Lu» Ii 1

GOTO 1588

.'0 ul>j<ju >j*jQv

4138 GOTO 1588
4288 bOSUB 5888
4218 C0L0R-15

fj_i-Q !!!_in iJ^'ift 1 / f -J-!, iii ui", I i

4H3&

4388
4318

4338
5B88

GOTO 1588

GOSUB 5888

C0L8R=15

HLIH3*(HU*XflT3*Y

GOTO 1588

S=S-1: FOR 1-1 TO 28: B= PEEK

^-loJ-Jb^ Ptth ??.-i&o'jb.rt rtU
(-16336)+ PEEK (-16336): HEKT

6888

6828
6838

I i RETURN

"LIT «1 •.-;?* *»

```
KKiHi "Hid «iN5'

GOSUB 5888; GOSUB 5888: G0S1

5888

PRINT s SC0RE= n ;St3

END



ih a/3a ihe.fi fodj; it* i/3i ihtH
7858

TC vYCv TUrLi 7' £3= TC t - , -"-~y TLJl«

ii iijn irtLn /ioel 1^ ij( ifitn

IF SX=I3 THEN 7858; IF KSX*
13*tSY-l)»9THEH7B18s IF
H(SX*13*(SY-i» HDD 18 THEN

DX=i:DY=8



7 SOp

7822 KX=3?SX-£; RY=3*SY-£



?"•*C¥.i :



i.ry



7823 FOR 1=1 TO 3sRX=RX*DX:RY=RY+

L- : >

7824 C0LOR=8

7825 FOR K=8 TO 1: FOR 1=8 TO h

PLOT QX+K,QY+L: NEXT L f K: COLORE

RD: FOR K=8 TO 1: FOR 1=8 TO

?Is PLOT RX+K,RY+L: HEKT ill

QK=RK;SY=RY
7po» urvT r

7835 SK=SKtDK:SY=SYtDY
```

7848 T(SX+13*(SY-1))=T(SX+13*(SY-
DH1

7845 RETURN

iuib ir ji-lo inch fidb; it" Hiftt
-13*(SY-1)»9 THEN 7868i IF
fhi^-ij^Hin-l/^ id tntn fide

7868 DX=8:DY=1: GOTO 782&
7188 lFSX=MHEH7t5B'flFT{SX+
13*(SY-1»>9 THEN 7118: IF
IKSX+13«SY-l)-i) H0D.i8 TIEH

7158

DRAGON MAZE cont

i lib i*ri-~~ il'01-ul Ul'ib tSCb

•;:r.-, :r .-:; s Tjtr-Et -r-.r.r ???? rr X. - ."is-

Hjs ir M=l IhtH imjl It KiAt
i o*j" cy _ i *• *vj Turu 7 1 £ s * t c

i v"Jv
iYV-5:

:-i~j-*j rnrnn csss. *"ii~nj) z~*iiii- ~f""IID
Q555? UU3UD- JuCuJ UUjUu -JOb^ uloUb

5688 ; GOSuB 5888; PRINT 5 THE DRR

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

67

System Monitor Commands

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

Command Format Example

Description

Examine Memory

adrs *C0F2

adrsl.adrs2

(return)

adrs2

*1024.1048

* (return)

*.4096

Examines (displays) single memory
location of (adrs)

Examines (displays) range of memory
from (adrsl) thru (adrs2)

Examines (displays) next 8 memory
locations.

Examines (displays) memory from current
location through location (adrs2)

Change Memory

adrsrdata
data data

*A256:EF 20 43

:data data
data

*:F0 A2 12

Move Memory

adrsl<adrs2.

adrs3M

*100<B010.B410M

Verify Memory

adrsl<adrs2.
adrs3V

*100<B010.B410V

Deposits data into memory starting at

location (adrs).

Deposits data into memory starting after (adrs) last used for deposits.

Copy the data now in the memory range from (adrs2) to (adrs3) into memory locations starting at (adrsl).

Verify that block of data in memory range from (adrs2) to (adrs3) exactly matches data block starting at memory location (adrsl) and displays differences if any.

68

Command Format Example

Description

Cassette I/O

adrsl.adrs2R

adrsl .adrs2W

*300.4FFR

*800.9FFW

Reads cassette data into specified memory (adrs) range. Record length must be same as memory range or an error will occur.

Writes onto cassette data from speci fied memory (adrs) range.

Display

I
N

*I

*N

Set inverse video
on white backgroun

Set normal video mode
on black background)

de. (Black characters

(White characters

Dis-assembler

adrsL

*C800L

Decodes 20 instructions starting at
memory (adrs) into 6502 assembly
nmenonic code.

Decodes next 20 instructions starting
at current memory address.

Mini -assembler
(Turn-on)

$(monitor
command)

adrs: (6502
MNEMONIC
instruction)

*F666G

$C800L

!C010:STA 23FF

Turns-on mini-assembler. Prompt

character
point).

is now a

(exclamation

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.

Assembles a mnemonic 6502 instruction into machine codes. If error, machine will refuse instruction, sound bell, and reprint line with up arrow under error.

Command Format

(space) (6502
mnemonic
instruction)

(TURN-OFF)

Example
! STA JQ1FF

I (Reset Button)

Description

Assembles instruction into next
available memory location. (Note
space between "!" and instruction)

Exits mini-assembler and returns
to system monitor.

Monitor Program Execution and Debugging

adrsG

adrsT

adrsS

(Control E)
(Control Y)

*300G

*800T

*C050S

*rC

•y 1

Runs machine level program starting
at memory (adrs).

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)

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).

Displays 6502 status registers and
opens them for modification (see Note 1)

Executes user specified machine
language subroutine starting at
memory location (3F8).

Note 1:

6502 status registers are open if they are last line displayed on screen
To change them type ":" then "data" for each register.

Example: A = 3C X = FF Y = 00 P = 32 S = F2

*: FF Changes A register only

*:FF 00 33 Changes A, X, and Y registers

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

Hexidecimal Arithmetic

datal+data2

datal-data2

*78+34

*AE-34

Performs hexidecimal sum of datal plus data2.

Performs hexidecimal difference of datal minus data2.

70

Command Format

Example

Description

Set Input/Output Ports

(X) (Control P)

•5P 1

(X) (Control K) *2K l

Sets printer output to I/O slot number (X). (see Note 2 below)

Sets keyboard input to I/O slot number (X). (see Note 2 below)

Note 2:

Only slots 1 through 7 are addressable in this mode. Address (Ex: 0P C or 0K C) resets ports to internal video display and keyboard. These commands will not work unless Apple II interfaces are plugged into specif iced I/O slot.

Multiple Commands

*100L 400G AFFT

Multiple monitor commands may be
given on same line if separated by

a "space"

1LLL

Single letter commands may be
repeated without spaces.

71

SPECIAL CONTROL AND EDITING CHARACTERS

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

CHARACTER

DESCRIPTION OF ACTION

RESET key

Control B

Immediately interrupts any program execution and resets
computer. Also sets all text mode with scrolling window

at maximum. Control is transfered to System Monitor and Apple prompts with a "*" (asterisk) and a bell. Hitting RESET key does NOT destroy existing BASIC or machine language program.

If in System Monitor (as indicated by a "*"), 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.

Control C

Control G
Control H

Control J
Control V

Control X

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 "*"), control C and a carraige return will enter BASIC without killing current program.

Sounds bell (beeps speaker)

Backspaces cursor and deletes any overwritten characters from computer but not from screen. Apply supplied keyboards have special key "<-" on right side of keyboard that provides this functions without using control button.

Issues line feed only

r
Compliment to H . Forward spaces cursor and copies over

written characters. Apple keyboards have "-?" key on

right side which also performs this function.

Immediately deletes current line.

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

SPECIAL CONTROL AND EDITING CHARACTERS
(continued)

CHARACTER DESCRIPTION OF ACTION

A F Move cursor to right

B F Move cursor to left

C r Move cursor down

Dp Move cursor up

E F Clear text from cursor to end of line

F Clear text from cursor to end of page

@ r Home cursor to top of page, clear text to end

of page.

Special Controls and Features

Hex

BASIC Example

Display Mode Controls

C05O
C051
C052
C053
C054

10 POKE -16304,0
20 POKE -16303,0
30 POKE -16302,0

40 POKE -16301,0
50 POKE -16300,0

C055
C056
C057

60 POKE -16299,0
70 POKE -16298,0
80 POKE -16297,0

TEXT Mode

Controls

0020

90 POKE 32, LI

0021

100 POKE 33, WI

0022

110 POKE 34, TI

0023

120 POKE 35,81

0024

130 CH=PEEK(36)
140 POKE 36, CH
150 TAB(CH+1)

0025

160 CV=PEEK(37)
170 POKE 37, CV
180 VTAB(CV+1)

0032

190 POKE 50,127
200 POKE 50,255

FC58

210 CALL -936

FC42

220 CALL -958

Description

Set color graphics mode

Set text mode

Clear mixed graphics

Set mixed graphics (4 lines text)

Clear display Page 2 (BASIC commands

use Page 1 only)
Set display to Page 2 (alternate)
Clear HIRES graphics mode
Set HIRES graphics mode

Set left side of scrolling window
to location specified by LI in
range of to 39.

Set window width to amount specified
byWl. L1+W1<40. W1>0

Set window top to line specified
by Tl in range of to 23

Set window bottom to line specified
by Bl in the range of to 23. B1>T1

Read/set cusor horizontal position
in the range of to 39. If using
TAB, you must add "1" to cusor position
read value; Ex. 140 and 150 perform
identical function.

Similar to above. Read/set cusor vertical position in the range to 23.

Set* inverse flag if 127 (Ex. 190)
Set normal flag if 255(Ex. 200)

(