## Apple II Redbook Digital Edition









The Apple logo is a trademark of Apple Computer, Inc., registered in the U.S. and other countries.

## **THINKING OF BUYING** FOR YOUR APPLE II PC?

Elevate your Apple II Experience with ReActiveMicro



## info@reactivemicro.com 1 (800) REACTIVE (732-2848) reactivemicro.com https://m.facebook.com/reactivemicrousa

**NEW PRODUCTS - NEW INNOVATIONS - SHOP ONLINE** 

## Leading Experts in Retro Computing

SPECIALIZING IN THE APPLE II FAMILY OF PERSONAL COMPUTERS

# Apple I Reference Manual

January 1978

leference Manual

## APPLE II Reference Manual

January 1978

Apple Part No. 030-0004-00

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

### APPLE II Reference Manual TABLE OF CONTENTS

A.	G	ETTING STARTED WITH YOUR	1
	1		י 1
	י. כ	Warranty Begistration Card	י 1
	2. 2	Chock for Shipping Damage	ו ס
	J.	Power Up	2
	4. 5		2
	ວ. ເ	APPLE In Speaks Several Languages .	2
	0. 7	Pupping Your First	0
	1.	and Second Programs	3
	8.	Running 16K Startrek	3
	9.	Loading a Program Tape	4
1	0.	Breakout and Color Demos Tapes	6
1	1.	Breakout and Color Demos Program Listings	12
1	12.	How to Play Startrek	14
1	13.	Loading HIRES Demo Tape	15
В.	AF	PPLE II INTEGER BASIC	17
	1.	BASIC Commands	18
	2.	BASIC Operators	19
	3.	BASIC Functions	22
	4.	BASIC Statements	23
	5.	Special Control and Editing	28
	6.	Table A – Graphics Colors	29
	7.	Special Controls and Features	30
	8.	BASIC Error Messages	32
	9.	Simplified Memory Map	33
1	10.	Data Read/Save Subroutines	34
1	11.	Simple Tone Subroutines	43
-	12.	High Resolution Graphics Subroutines and Listings	46

	13	3. Additional BASIC Program	
			55
		a. Rod's Color Pattern (4K)	55
		b. Pong (4K)	56
		c. Color Sketch (4K)	57
		d. Mastermind (8K)	69
		e. Biorhythm (4K)	61
		f. Dragon Maze (4K)	
C.	A	PPLE II FIRMWARE	67
	1.	System Monitor Commands	68
	2.	Control and Editing Characters	72
	3.	Special Controls and Features	74
	4.	Annotated Monitor and	
		Dis-assembler Listing	76
	5.	Binary Floating Point Package	94
	6.	Sweet 16 Interpreter Listing	96
	7.	6502 Op Codes	100
D.	AF	PPLE II HARDWARE	106
	1.	Getting Started with Your APPLE II Board	107
	2.	APPLE II Switching Power Supply.	110
	3.	Interfacing with the Home TV	112
	4.	Simple Serial Output	114
	5.	Interfacing the APPLE –	
		Signals, Loading, Pin	
		Connections	122
	6.	Memory –	
		Options, Expansion, Map, Address	133
	7	System Timina	1/10
	γ. Ω	Schomatics	1/1
	о.	Junemanues	141

#### Unpacking

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

- 1. Apple II system including mother printed circuit board with specified amount of RAM memory and 8K of ROM memory, switching power supply, keyboard, and case assembly.
- 2. Accessories Box including the following:
  - a. This manual including warranty card.
  - b. Pair of Game Paddles
  - c. A.C. Power Cord
  - d. Cassette tape with "Breakout"on one side and "Color Demos" on the other side.
  - e. Cassette recorder interface cable (miniature phone jack type)
- 3. If you purchased a 16K or larger system, your accessory box should also contain:
  - a. 16K Startrek game cassette with High Resolution Graphics Demo ("HIRES") on the flipside.
  - b. Applesoft Floating Point Basic Language Cassette with an example program on the other side.
  - c. Applesoft reference manual
- In addition other items such as a vinyl carrying case or hobby board peripherial may have been included if specifically ordered as "extras".

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

#### Warranty Registration Card

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

#### A2SØØMMX

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

#### A2SØØØ8X

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 <u>before</u> you "RUN" it, type in "HIMEM: 16384" to set exactly where in memory this program is to run.

#### 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'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.

4

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.

5

- 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:

- Hit the RESET key. An asterick, "\*", should appear on the lefthand side of the screen below the random text pattern. A flashing white cursor will appear to the right of the asterick.
- 2. Hold down the CTRL key, depress and release the B key, then depress the "RETURN" key and release the "CTRL" key. A right facing arrow should appear on the lefthand side of the screen with a flashing cursor next to it. If it doesn't, repeat steps 1 and 2.
- 3. Type in the word "LOAD" on the keyboard. You should see the word in between the right facing arrow and the flashing cursor. Do not depress the "RETURN" key yet.
- 4. Insert the program cassette into the tape recorder and rewind it.
- 5. If not already set, adjust the Volume control to 50-70% maximum. If present, adjust the Tone control to 80-100% maximum.

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

The steps in loading a program have been completed and if 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.

8

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  $B_{\rm C}$  (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, orb) 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.

#### PROGRAM DESCRIPTION

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

There are eight columns of bricks. As you penetrate through the wall the point value of the bricks increases. A perfect game is 720 points; after five balls have been played the computer will display your score and a rating such as "Very Good". "Terrible!", etc. After ten hits of the ball, its speed with double, making the game more difficult. If you break through to the back wall, the ball will rebound back and forth, racking up points.

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

#### REQUIREMENTS

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

#### PLAYING BREAKOUT

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

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

NOTE: A game paddle (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.

#### PROGRAM LISTING

5 GOTO 15

- 10 Q=( PDL (0)-20)/6: IF Q(0 THEN Q=0: IF Q>=34 THEN Q=34: COLOR= D: VLIN Q.Q+5 AT 0: COLOR=A: IF P>Q THEN 175: IF Q THEN VLIN 0.Q-1 AT 0:P=Q:RETURN
- 15 DIM A\$(15),B\$(10):A=1:B=13: C=9:D=6:E=15: TEXT : CALL -936: VTAB 4: TAB 10: PRINT \*\*\*\* BREAKOUT \*\*\*\*:PRINT
- 20 PRINT OBJECT IS TO DESTROY ALL BRICKS": PRINT : INPUT "HI. WHAT'S YOUR NAME? ".A\$
- 25 PRINT "STANDARD COLORS ":A\$ :: INPUT "Y/N? ".B\$: GR: CALL -936: IF B\$(1.1)#"N" THEN 40 : FOR I=0 TO 39: COLOR=I/2\* (I(32): VLIN 0.39 AT I
- 30 NEXT I: POKE 34.20: PRINT : PRINT : PRINT : FOR I=0 TO 15: YTAB 21+I MOD 2: TAB I+ I+1: PRINT I:: NEXT I: POKE 34.22: YTAB 24: PRINT : PRINT "BACKGROUND":
- 35 GOSUB 95: A=E: PRINT "EVEN BRICK" 80 V=-V ::GOSUB 95:B=E: PRINT "ODD BRIC 85 PLOT X,Y/3: COLOR=E: PLOT I, K":: GOSUB 95:C=E: PRINT "PADDLE ":: GOSUB 95:D=E: PRINT "BALL" 90 PRINT "INVALID. REENTER": ::GOSUB 95
- 40 POKE 34.20: COLOR=A: FOR I= 0 TO 39: VLIN 0.39 AT I: NEXT I: FOR I=20 TO 34 STEP 2: TAB I+1: PRINT I/2-9;: COLOR=B: VLIN 0,39 AT I: COLOR=C: FOR J=I MOD 4 TO 39 STEP 4

- 45 VLIN J,J+1 AT I: NEXT J,I: TAB 5: PRINT "SCORE=0":PRINT
  - : PRINT : POKE 34.21:5=0:P= S:L=S:X=19:Y=19:L=6
- 50 COLOR=A: PLOT X,Y/3:X=19:Y= RND (120):V=−1:W= RND (5)-2:L=L-1: IF L<1 THEN 120: TAB 6: IF L>1 THEN PRINT L; BALLS L EFT
- 55 IF L=1 THEN PRINT "LAST BALL, " 110 IF 5<720 THEN 80 :A\$: PRINT : FOR I=1 TO 100 : GOSUB 10: NEXT I:M=1:N=0
- 60 J=Y+W: IF J>=0 AND J<120 THEN 65:⋓=-⋓:J=Y: FOR I-1 TO 6:K= PEEK (-16336): NEXT I
- 65 I=X+V: IF I<0 THEN 180: GOSUB 170: COLOR=A:K=J/3: IF I>39 130 PRINT "LOUSY.": GOTO 165 THEN 75: IF SCRN(I.K)=A THEN 85: IF I THEN 100:N=N+1:V=( H>5)+1:W=(K-P)+2-5:M=1
- 70 Z= PEEK (-16336)-PEEK (-16336 )+ PEEK (-16336)- PEEK (-16336 )+ PEEK (-16336)- PEEK (-16336 )+ PEEK (-16336): GOTO 85
  - 75 FOR I=1 TO 6:M= PEEK (-16336 ): NEXT I:I=X:M=0

    - K:X=I:Y=J: GOTO 60

    - IF E<0 OR E>15 THEN 90: RETURN
- 100 IF M THEN V= ABS (V): VLIN K/2\*2.K/2\*2+1 AT I:S=S+I/2-9: VTAB 21: TAB 13: PRINT S 105 Q= PEEK (-16336)- PEEK (-16336 )+ PEEK (-16336)- PEEK (-16336 )+ PEEK (-16336)- PEEK (-16336) )+ PEEK (-16336)- PEEK (-16336 )+ PEEK (-16336)- PEEK (-16336 ) 115 PRINT "CONGRATULATONS. ":A\$ :" YOU WIN!": GOTO 165 120 PRINT "YOUR SCORE OF ";S;" IS " :: GOTO 125+(S/100)\*5 125 PRINT "TERRIBLE!": GOTO 165 135 PRINT "POOR.": GOTO 165 140 PRINT "FAIR.": GOTO 165 145 PRINT "GOOD.": GOTO 165 150 PRINT "VERY GOOD.": GOTO 165 155 PRINT "EXCELLENT.": GOTO 165 160 PRINT "NEARLY PERFECT." 165 PRINT "ANOTHER GAME ":A\$:" (Y/N) ":: INPUT A\$: IF A\$(1.1)="Y" THEN 25: TEXT : CALL -936: VTAB 10: TAB 10: PRINT "GAME OV ER": END 95 INPUT " COLOR (0 TO 15)".E: 170 Q=( PDL (0)-20)/6: IF Q(0 THEN Q=0: IF Q>=34 THEN Q=34: COLOR=
  - D: VLIN Q.Q+5 AT 0: COLOR=A: IF P>Q THEN 175: IF Q THEN VLIN 0,Q-1 AT 0:P=Q: RETURN
  - 175 IF P=Q THEN RETURN : IF Q#34 THEN VLIN Q+6.39 AT 0:P=Q: RETURN 180 FOR I=1 TO 80:0= PEEK (-16336 >: HEXT I: GOTO 50

#### PROGRAM LISTING

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

- 20 POKE 13,4: POKE 14,198: POKE 15,24: POKE 16,240: POKE 17 ,5: POKE 18,198: POKE 19,1: POKE 20,76: POKE 21,2: POKE 22,0: POKE 23,96
- 30 TEXT : CALL -936: VTAB 4: TAB 8: PRINT "4K COLOR DEMOS": PRINT : PRINT "1 LINES": PRINT "2 CROS S": PRINT "3 WAVING"

40 PRINT "4 TUNNEL": PRINT "5 CIRCL E": PRINT "6 SPIRAL \*\*\*": PRINT "7 TONES \*\* ": PRINT "8 SPRING"

- 50 PRINT "9 HYPERBOLA": PRINT "10 COLOR BARS": PRINT : PRINT "\*\* NEEDS PDL(0) CONNECTED" : PRINT 60 PRINT "HIT ANY KEY FOR NEW DEMO"
- :Z=0: PRINT : INPUT "WHICH DEMO # ",I: GR : IF I>0 AND I<11 THEN GOTO 100+I: GOTO 30 70 INPUT "WHICH DEMO WOULD YOU LIKE ",I: GR : IF I AND I<20 THEN GOTO 100\*I: GOTO 30
- 100 I=1+I MOD 79: J=I+(I)39)\*(79 -I-I): GOSUB 2000: GOSUB 10000 : GOTO 100
- 200 I=1+I MOD 39:J=1: GOSUB 2000 :J=30-I: GOSUB 2000: GOSUB 10000: GOTO 200
- 300 J=J+1:J=J MOD 22+1: FOR I=1 TO 1295: COLOR=I MOD J+7: PLOT (2\*I) MOD 37.(3\*I) MOD 35: NEXT I: GOSUB 10000: GOTO 300 400 FOR I=1 TO 4:C(I)=RND (16) : NEXT I 410 FOR I=3 TO 1 STEP -1:C(I+1) =C(I): MEXT I:C(1)= RMD (16 ): FOR I=1 TO 5: FOR J=1 TO 4 420 COLOR=C(J):L=J\*5+14+I:K=39-L: HLIN K,L AT K: YLIN K,L AT L: HLIN K.L AT L: VLIN K.L AT K: NEXT K.I: GOSUB 10000: GOTO 410 500 Z=20: GOTO 900 600 COLOR= RND (16): FOR I=0 TO 18 STEP 2:J=39-I: HLIN I,J AT I: GOSUB 640: VLIN I.J AT J: GOSUB 640 610 HLIN I+2. J AT J: GOSUB 640 VLIN I+2. J AT I+2: GOSUB 640 : NEXT I 620 COLOR= RND (16): FOR I=18 TO 0 STEP -2:J=39-I: VLIN I+2. J AT I+2: GOSUB 640: HLIN I+ 2.J AT J: GOSUB 640 630 VLIN I.J AT J: GOSUB 640: HLIN I.J AT I: GOSUB 640: NEXT I: GOSUB 10000: GOTO 600 640 K=I+7:L=K\*K\*5\*K\*26+70:L=32767 /L\*( PDL (0)/10): POKE 0,K: POKE 1.L MOD 256: POKE 24. L/256+1: CALL 2: RETURN
- 700 I= RND (30)+3:J=I+I\*5+I\*26+ 70:K=32767/J\*( PDL (0)/10): POKE 0,I: POKE 1,K MOD 256 : POKE 24,(K)255)+1: CALL 2 : GOSUB 10000: GOTO 700 800 X=3:A=1000:P=A:L=20:W=4:Y=0 :J=1: COLOR=6: HLIN 0,39 AT 4: COLOR=9: GOSUB 880: COLOR= 12: VLIN 5,M-2 AT X 810 N=2\*A-P-A/W: COLOR=0: GOSUB 880: VLIN 5,39 AT X:X=X+1: IF
  - 880: VLIN 5,39 AT X:X=X+1: IF X<39 THEN 820:X=3: VLIN 5,39 AT 1: VLIN 5,39 AT 2 820 P=A:A=N:Y=A/100: COLOR=12: GOSUB
  - 880: COLOR=9: VLIN 5,M-2 AT X: COLOR=15: PLOT X-2,M: FOR I=0 TO J: NEXT I: GOSUB 10000 : GOTO 810
  - 880 M=L-Y:L1=M-1:L2=M+1: VLIN L1, L2 AT X-1: VLIN L1,L2 AT X: VLIN L1,L2 AT X+1: RETURN
  - 900 I=1+I MOD 15: FOR Y=0 TO 39 : FOR X=0 TO 39: COLOR=I+( ABS (20-X)-Z)\*( ABS (20-Y)-Z)/25 : PLOT X,Y: NEXT X,Y: GOSUB 10000: GOTO 900
- 1000 CALL -936
- 1010 J=1+K MOD 32: COLOR=J/2: VLIN 0,39 AT 3+J: VTAB 21+(J/2) MOD 2: TAB 3+J: IF J MOD 2 THEN PRINT J/2;: GOSUB 10000: GOTO 1010
- 2000 COLOR= RND (16); HLIN 0,39 AT J: COLOR= RND (16): VLIN 0, 39 AT J: RETURN
- 10000 IF PEEK (-16384)×128 THEN RETURN : POKE -16368,0: POP : GOTO 30

APPLE II STARTREK VERSION -.-.-. -.-.-. THIS IS A SHORT DESCRIPTION OF HOW TO PLAY STARTREK ON THE APPLE COMPUTER. THE UNIVERSE IS MADE UP OF 64 RUADRANTS IN AN 8 BY 8 MATRIX. THE RUADRANT IN WHICH YOU 'THE ENTERPRISE ' ARE, IS IN WHITE, AND A BLOW UP OF THAT RUADRANT IS FOUND IN THE LOWER LEFT YOUR SPACE SHIP STATUS IS FOUND IN A TABLE TO CORNER. THE RIGHT SIDE OF THE RUADRANT BLOW UP. THIS IS A SEARCH AND DESTROY MISSION. THE OBJECT IS TO LONG-RANGE SENSE FOR INFORMATION AS TO WHERE KLINGONS (K) ARE, MOVE TO THAT RUADRANT, AND DESTROY. NUMBERS DISPLAYED FOR EACH QUADRANT DENOTE: # OF STARS IN THE ONES PLACE # OF STARS IN THE ONES PLACE # OF BASES IN THE TENS PLACE # OF KLINGONS IN THE HUNDREDS PLACE AT ANY TIME DURING THE GAME, FOR INSTANCE BEFORE ONE TOTALLY RUNS OUT OF ENERGY, OR NEEDS TO REGENERATE ALL SYSTEMS, ONE MOVES TO A QUADRANT WHICH INCLUDES A BASE, IONS NEXT TO THAT BASE (B) AT WHICH TIME THE BASE SELF-DESTRUCTS AND THE ENTERPRISE (E) HAS ALL SYSTEMS "GO" AGAIN. TO PLAY: 1. THE COMMANDS CAN BE OBTAINED BY TYPING A "O" (ZERO) AND RETURN. THEY ARE: 1. PROPULSION 3. LONG RANGE SENSORS 5. PHOTON TORPEDOES 2. REGENERATE 4. PHASERS 6. GALAXY RECORD 7. COMPUTER 8. PROBE 9. SHIELD ENERGY 10.DAMAGE REPORT 11.LOAD PHOTON TORPEDDES 2. THE COMANDS ARE INVOKED BY TYPING THE NUMBER REFERING TO THEM FOLLOWED BY A "RETURN". FOLLOWED BY A "RETURN". A. IF RESPONSE IS 1 THE COMPUTER WILL ASK WARP OR ION AND EXPECTS "W" IF ONE WANTS TO TRAVEL IN THE GALAXY BETWEEN QUADRANTS AND AN "I" IF ONE WANTS ONLY INTERNAL QUADRANT TRAVEL. DURATION OF WARP FACTOR IS THE NUMBER OF SPACES OR QUADRANTS THE ENTERPRISE WILL MOVE. COURSE IS COMPASS READING IN DEGREES FOR THE DESI-RED DESTINATION. B. A 2 REGENERATES THE ENERGY AT THE EXPENSE OF TIME. C. A 3 GIVES THE CONTENTS OF THE IMMEDIATE. ADJACENT QUADRANTS. THE GALAXY IS WRAP-AROUND IN ALL DIRECTIONS. D. 4 FIRES PHASERS AT THE EXPENSE OF AVAILABLE ENERGY. E. 5 INITIATES A SET OF QUESTIONS FOR TORPEDO FIRING. THEY CAN BE FIRED AUTOMATICALLY IF THEY HAVE BEEN LOCKED ON TARGET WHILE IN THE COMPUTER MODE, OR MAY BE FIRED MANUALLY IF THE TRAGECTORY ANGLE IS KNOWN. F. 6, 8 AND 10 ALL GIVE INFORMATION ABOUT THE STATUS OF THE SHIP AND ITS ENVIRONMENT. G. 9 SETS THE SHIELD ENERGY/AVAILABLE ENERGY RATIO.
 H. 11 ASKS FOR INFORMATION ON LOADING AND UNLOADING OF PHOTON TORPEDOES AT THE ESPENSE OF AVAILABLE ENERGY. THE ANSWER SHOULD BE A SIGNED NUMBER. FOR EXAMPLE +5 OR -2. I. 7 ENTERS A COMPUTER WHICH WILL RESPOND TO THE FOLLOWING INSTRUCTIONS: 1. COMPUTE COURSE 2. LOCK PHASERS 
 3. LOCK PHOTON TORPEDOES

 4. LOCK COURSE

 5. COMPUTE TREJECTORY

 6. STATUS

 7. RETURN TO COMAND MODE
 IN THE FIRST FIVE ONE WILL HAVE TO GIVE COORDINATES. COORDINATES ARE GIVEN IN MATHMATICAL NOTATION WITH THE EXCEPTION THAT THE 'Y' VALUE IS GIVEN FIRST. AN EXAMPLE WOULD BE 'Y'X' COURSE OR TRAJECTORY: 0 270-----90 180

#### PROCEDURE

- Power up system turn the AC power switch in the back of the Apple II on. You should see a random matrix of question marks and other text characters. If you don't, consult the operator's manual for system checkout procedures.
- 2. Hit the RESET key. On the left hand side of the screen you should see an asterisk and a flashing cursor next to it below the text matrix.
- 3. Insert the HI-RES demo tape into the cassette and rewind it. Check Volume (50-70%) and Tone (80-100%) settings.
- 4. Type in "CØØ.FFFR" on the Apple II keyboard. This is the address range of the high resolution machine language subprogram. It extends from \$CØØ to \$FFF. The R tells the computer to read in the data. <u>Do not</u> depress the "RETURN" key yet.
- 5. Start the tape recorder in playback mode and depress the "RETURN" key. The flashing cursor disappears.
- 6. A beep will sound after the program has been read in. STOP the tape recorder. Do not rewind the program tape yet.
- 7. Hold down the "CTRL" key, depress and release the B key, then depress the "RETURN" key and release the "CTRL" key. You should see a right facing arrow and a flashing cursor. The Bc command places the Apple into BASIC initializing the memory pointers.
- 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.

- A second beep will sound, and the right facing arrow will reappear with the flashing cursor. STOP the tape recorder. Rewind the tape.
- 11. Type in "HIMEM:8192" and hit the "RETURN" key. This sets up memory for high resolution graphics.
- 12. Type in "RUN" and hit the "RETURN" key. The screen should clear and momentarily a HI-RES demo menu table should appear. The loading sequence is now completed.

SUMMARY OF HI-RES DEMO TAPE LOADING

- 1. RESET
- 2. Type in CØØ.FFFR
- 3. Start tape recorder, hit RETURN
- 4. Asterick or flashing cursor reappear Bc (CTRL B) into BASIC
- 5. Type in "LOAD", hit RETURN
- 6. BASIC prompt (7) and flashing cursor reappear. Type in "HIMEN:8192", hit RETURN
- 7. Type in "RUN", hit RETURN
- 8. STOP tape recorder, rewind tape.

## APPLE II INTEGER BASIC

- 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. Simpfilied Memory Map
- 10. Data Read Save Subroutines
- 11. Simple Tone Subroutires
- 12. High Resolution Graphics
- 13. Additional BASIC Program Examples

#### BASIC COMMANDS

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

#### COMMAND NAME

<u>AUTO</u> num	Sets automatic line numbering mode. Starts at line number <i>num</i> and increments line numbers by 10. To exit AUTO mode, type a control X*, then type the letters "MAN" and press the return key.
<u>AUTO</u> num1, num2	Same as above execpt increments line numbers by number num2.
CLR	Clears current BASIC variables; undimensions arrays. Program is unchanged.
<u>CON</u>	Continues program execution after a stop from a control C*. Does not change variables.
<u>DEL</u> numl,	Deletes line number num1.
<u>DEL</u> num1, num2	Deletes program from line number <i>numl</i> through line number <i>num2</i> .
<u>DSP</u> var	Sets debug mode that will display variable <i>var</i> every time that it is changed along with the line number that caused the change. (NOTE: RUN command clears DSP mode so that DSP command is effective only if program is continued by a CON or GOTO command.)
HIMEM expr	Sets highest memory location for use by BASIC at location specified by expression <i>expr</i> in <u>decimal</u> . HIMEM: may not be increased without destroying program. HIMEM: is automatically set at maximum RAM memory when BASIC is entered by a control B*.
<u>GOTO</u> expr	Causes immediate jump to line number specified by expression <i>expr</i> .
<u>GR</u>	Sets mixed color graphics display mode. Clears screen to black. Resets scrolling window. Displays 40x40 squares in 15 colors on top of screen and 4 lines of text at bottom.
LIST	Lists entire program on screen.
LIST numl	lists program line number num1
LIST num1, num2	Lists program line number num1 through line number num2.

- LOADexpr.Reads (Loads) a BASIC program from cassette tape.<br/>Start tape recorder before hitting return key. Two<br/>beeps and a " > " indicate a good load. "ERR" or "MEM"<br/>FULL ERR" message indicates a bad tape or poor recorder<br/>performance.LOMEM:exprSimilar to HIMEM: except sets lowest memory location<br/>available to BASIC. Automatically set at 2048 when
- available to BASIC. Automatically set at 2048 when BASIC is entered with a control B\*. Moving LOMEM: destroys current variable values.
- MAN Clears AUTO line numbering mode to all manual line numbering after a control C\* or control X\*.
- <u>NEW</u> Clears (Scratches) current BASIC program.

<u>NO DSP</u> *var* Clears DSP mode for variable var.

- <u>NO TRACE</u> Clears TRACE mode.
- <u>RUN</u> Clears variables to zero, undimensions all arrays and executes program starting at lowest statement line number.
- <u>RUN</u> expr Clears variables and executes program starting at line number specified by expression expr.
- SAVE Stores (saves) a BASIC program on a cassette tape. Start tape recorder in record mode prior to hitting return key.
- <u>TEXT</u> 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.
- <u>TRACE</u> 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<sup>C</sup>.

#### BASIC Operators

<u>Symbol</u>	Sample Statement	Explanation
Prefix (	)perators	
( )	10 X = 4*(5 + X)	Expressions within parenthesis ( ) are always evaluated first.
+	2Ø X= 1+4*5	Optional; +1 times following expression.
-	3Ø ALPHA = -(BETA +2)	Negation of following expression.
ΝΟΤ	40 IF A NOT B THEN 200	Logical Negation of following expression; Ø if expression is true (non-zero), l if expression is false (zero).
Arithme	etic Operators	
Ţ	60 Y = X 3	Exponentiate as in X $^3$ . NOTE: $\ \uparrow$ is shifted letter N.
*	7Ø LET DOTS=A*B*N2	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 MOD(Y+2)	Modulo: Remainder after division of first expression by second expression.
+	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"	Assignment operator; assigns a value to a variable. LET is optional

#### Relational and Logical Operators

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

Symbol	Sample Statement	Explanation
=	160 IF D = E THEN 500	Expression "equals" expression.
=	17Ø IF A\$(1,1)= "Y" THEN 500	String variable "equal'string variable.
# or < >	18Ø IF ALPHA #X*Y THEN 500	Expression "does not equal" expression.
#	19Ø IF A\$ # "NO" THEN 5ØØ	String variable "does not equal" string variable. NOTE: If strings are not the same length, they are considered un-equal. < > not allowed with strings.
>	200 IF A>B THEN GO TO 50	Expression "is greater than" expression.
<	210 IF A+1 <b-5 THEN 100</b-5 	Expression "is less than" expression.
>=	22Ø IF A>=B THEN 100	Expression "is greater than or equal to" expression.
<=	23Ø IF A+1<=B-6 THEN 200	Expression "is less than or equal to" expression.
AND	24Ø IF A>B AND C <d 200<="" td="" then=""><td>Expression 1 "and" expression 2 must both be "true" for statements to be true.</td></d>	Expression 1 "and" expression 2 must both be "true" for statements to be true.
OR	25Ø IF ALPHA OR BETA+1 THEN 200	If either expression 1 or expression 2 is "true", statement is "true".

#### BASIC FUNCTIONS

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

ABS	(expr)	300	PRINT	ABS(X)	Gives absolute value of the expression $expr$ .
ASC	(str\$)	310 320 330 335	PRINT PRINT PRINT PRINT	ASC("BACK") ASC(B\$) ASC(B\$(4,4)) ASC(B\$(Y))	Gives decimal ASCII value of designated string variable str. If more than one character is in designated string or sub-string, it gives decimal ASCII value of first character
LEN	(str\$)	340	PRINT	LEN(B\$)	Gives current length of designated string variable <i>str\$</i> ;i.e., number of characters.
PDL	(expr)	350	PRINT	PDL(X)	Gives number between Ø and 255 corresponding ponding to paddle position on game paddle number designated by expression expr and must be legal paddle (Ø,1,2,or 3) or else 255 is returned.
PEEK	(expr)	360	PRINT	PEEK(X)	Gives the decimal value of number stored of decimal memory location specified by expression <i>expr</i> . For MEMORY locations above 32676, use negative number; i.e., HEX location FFFØ is -16
RND	(expr)	370	PRINT	RND(X)	Gives random number between V and (expression expr -1) if expression expr is positive; if minus, it gives random number between Ø and (expression expr +1).
SCRN exp	l(exprl, m2)	380	PRINT	SCRN (X1,Y1)	OGives color (number between Ø and 15) of screen at horizontal location designated by expression <i>exprl</i> and vertical location designated by expression <i>expr2</i> Range of expression <i>exprl</i> is Ø to 39. Range of expression expr2 is Ø to 39 if in standar mixed colorgraphics display mode as set by GR command or Ø to 47 if in all color mode set by POKE -16304 ,Ø: POKE - 16302,Ø'.
SGN	(expr)	39Ø	PRINT	SGN(X)	Gives sign (not sine) of expression <i>expr</i> i.e., -1 if expression <i>expr</i> is negative,zero zero and +1 if <i>expr</i> is positive.

#### BASIC STATEMENTS

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

NAME

CALL expr	10 CALL-936	Causes execution of a machine level language subroutine at <u>decimal</u> memory location specified by expression <i>expr</i> Locations above 32767 are specified using negative numbers; i.e., location in example 10 is hexidecimal number \$FC53
<u>COLOR</u> =expr	3Ø COLOR=12	In standard resolution color (GR) graphics mode, this command sets screen TV color to value in expression <i>expr</i> in the range Ø to 15 as described in Table A. Actually expression <i>expr</i> may be in the range Ø to 255 without error message since it is implemented as if it were expression <i>expr</i> MOD 16.
DIM varl (expr1) str\$ (expr2) var2 (expr3)	50 DIM A(20),B(10) 60 DIM B\$(30) 70 DIM C (2) Illegal: 80 DIM A(30) Legal: 85 DIM C(1000)	The DIM statement causes APPLE II to reserve memory for the specified variables. For number arrays APPLE reserves approximately 2 times $expr$ bytes of memory limited by available memory. For string arrays $-str$ ; $-(expr)$ must be in the range of 1 to 255. Last defined variable may b'e redimensioned at any time; thus, example in line is illegal but 85 is allowed.
<u>DSP</u> var	Legal: 90 DSP AX: DSP L Illegal: 100 DSP AX,B 102 DSP AB\$ 104 DSP A(5) Legal: 105 A=A(5): DSP A	Sets debug mode that DSP variable <i>var</i> each time it changes and the line number where the change occured.

NAME	EXAMPLE	DESCRIPTION
END	110 END	Stops program execution. Sends carriage return and "> " BASIC prompt) to screen.
<u>FOR</u> var= exp'21 T0expr2 STEPexpr3	110 FOR L=0 to 39 120 FOR X=Y1 TO Y3 130 FOR 1=39 TO 1 150 GOSUB 100 *J2	Begins FORNEXT loop, initializes variable var to value of expression <i>exprl</i> then increments it by amount in expression <i>expr3</i> each time the corresponding "NEXT" statement is encountered, until value of expression <i>expr8</i> is reached. If STEP <i>expr3</i> is omitted, a STEP of +1 is assumed. Negative numbers are allowed.
<u>GOSUB</u> expr	140 GOSUB 500	Causes branch to BASIC subroutine starting at legal line number specified by expression <i>expr</i> Subroutines may be nested up to 16 levels.
<u>GOTO</u> expr	16Ø GOTO 20Ø 17Ø GOTO ALPHA+1ØØ	Causes immediate jump to legal line number specified by expression <i>expr</i> .
<u>GR</u>	180 GR 190 GR: POKE -16302,0	Sets mixed standard resolution color graphics mode. Initializes COLOR = Ø (Black) for top 4Øx4Ø of screen and sets scrolling window to lines 21 through 24 by 4Ø characters for four lines of text at bottom of screen. Example 19Ø sets all color mode (4Øx48 field) with no text at bottom of screen.
HLIN expr1, expr2ATexpr3	200 HLIN 0,39 AT 20 210 HLIN Z,Z+6 AT I	In standard resolution color graphics mode, this command draws a horizontal line of a predefined color (set by COLOR=) starting at horizontal position defined by expression exprl and ending at position expr2 at vertical position defined by expression expr3.expr1 and $expr3$ must be in the range of Ø to 39 and $expr1 \le expr3$ . $expr3$ be in the range of Ø to 39 (or Ø to 47 if not in mixed mode).
Note:	HLIN Ø, 19 AT Ø is a horiz	zontal line at the top of the screen

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

<u>IF</u> expression <u>THEN</u> statemen	22Ø t 23Ø 24Ø 25Ø Illegal 26Ø	IF A> B THEN PRINT A IF X=Ø THEN C=1 IF A#1Ø THEN GOSUB 2ØØ IF A\$(1,1)# "Y" THEN 1ØØ : IF L> 5 THEN 5Ø: ELSE 6Ø	If expression is true (non-zero) then execute statement; if false do not execute statement. If statement is an expression, then a GOTO expr type of statement is assumed to be implied. The "ELSE" in example 260 is illegal but may be implemented as shown in example 270.
INDIIT warl	Legal: 27Ø	IF L> 5 THEN 5Ø GO TO 6Ø	
var2, str\$	28Ø 29Ø 3ØØ	INPUT X,Y,Z(3) INPUT "AMT", )LLR INPUT "Y or N?", A\$	Enters data into memory from I/O device. If number input is expected, APPLE wil output "?"; if string input is expected no "?" will be outputed. Multiple numeric inputs to same statement may be separated by a comma or a carriage return. String inputs must be separated by a carriage return only. One pair of " " may be used immediately after INPUT to output prompting text enclosed within the quotation marks to the screen.
<u>IN#</u> expr	31Ø 32Ø 33Ø	IN# 6 IN# Y+2 IN# 0	Transfers source of data for subsequent INPUT statements to peripheral I/O slot (1-7) as specified as by expression <i>expr.</i> Slot Ø is not addressable from BASIC. IN#Ø (Example 33Ø) is used to return data source from peripherial I/O to keyboard connector.
<u>LET</u>	34Ø	LET X=5	Assignment operator. "LET" is optional
LIST num1, num2	35Ø L	IF X > 6 THEN IST 50	Causes program from line number <i>numl</i> through line number num2 to be displayed on screen.
NEXT varl, var2	36Ø 37Ø	NEXT I NEXT J,K	Increments corresponding "FOR" variable and loops back to statement following "FOR" until variable exceeds limit.
<u>NO DSP</u> var	38Ø	NO DSP I	Turns-off DSP debug mode for variable
NO TRACE	39Ø	NO TRACE	Turns-off TRACE debug mode

PLOT expr1, expr2	400 PLOT 15, 25 400 PLT XV,YV	In standard resolution color graphics, this command plots a small square of a predefined color (set by COLOR=) at horizontal location specified by expression <i>expr1</i> in range Ø to 39 and vertical location specified by expression <i>expr2</i> in range Ø to 39 (or Ø to 47 if in all graphics mode) NOTE: PLOT Ø Ø is upper left and PLOT 39, 39 (or PLOT 39, 47) is lower right corner.
POKE expr1, expr2	420 POKE 20, 40 430 POKE 7*256, XMOD255	Stores <u>decimal</u> number defined by expression <i>expr2</i> in range of Ø 255 at <u>decimal</u> memory location specified by expression <i>expr1</i> Locations above 32767 are specified by negative numbers.
POP	44Ø POP	"POPS" nested GOSUB return stack address by one.
<u>PRINT</u> var1, var, str\$	450 PRINT L1 460 PRINT Li, X2 470 PRINT "AMT=";DX 480 PRINT A\$;B\$; 490 PRINT 492 PRINT "HELLO"	Outputs data specified by variable var or string variable str\$ starting at current cursor location. If there is not trailing "," or ";" (Ex 450) a carriage return will be generated.
	494 PRINT 2+3	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.
<u>PR#</u> expr	500 PR# 7	Like IN#, transfers output to I/O slot defined by expression <i>expr</i> PR# Ø is video output not I/O slot Ø.
REM	510 REM REMARK	No action. All characters after REM are treated as a remark until terminated by a carriage return.
<u>RETURN</u>	520 RETURN 530 IFX= 5 THEN RETURN	Causes branch to statement following last GOSUB; i.e., RETURN ends a subroutine. Do not confuse "RETURN" <u>statement</u> with Return <u>key</u> on keyboard.

TAB expr	53Ø TAB 24 54Ø TAB 1+24 55Ø IF A#B THEN TAB 2Ø	Moves cursor to absolute horizontal position specified by expression <i>expr</i> in the range of 1 to 40. Position is left to right
<u>TEXT</u>	550 TEXT 560 TEXT: CALL-936	Sets all text mode. Resets scrolling window to 24 lines by 4Ø characters. Example 56Ø also clears screen and homes cursor to upper left corner
TRACE	570 TRACE 580 IFN >32000 THEN TRACE	Sets debug mode that displays each line number as it is executed.
VLIN exprl, expr2 AT expr3	590 VLIN Ø, 39AT15 600 VLIN Z,Z+6ATY	Similar to HLIN except draws vertical line starting at <i>exprl</i> and ending at <i>expr2</i> at horizontal position <i>expr3</i> .
VTAB expr	61Ø VTAB 18 62Ø VTAB Z+2	Similar to TAB. Moves cursor to absolute vertical position specified by expression expr in the range 1 to 24. VTAB 1 is top line on screen; VTAB24 is bottom.

#### SPECIAL CONTROL AND EDITING CHARACTERS

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

CHARACTE	ER	DESCRIPTION OF ACTION
RESET ke	ey	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	В	If in System Monitor (as indicated by a "*"), a control B and a carriage return will transfer control to BASIC, <u>scratching (killing) any existing BASIC program</u> and set HIMEM: to maximum installed user memory and LOMEM: to 2048.
Control	C	If in BASIC, halts program and displays line number where stop occurred*. Program may be continued with a CON command. If in <u>System</u> Monitor, (as indicated by "*"), control C and a carraige return will enter BASIC <u>without</u> killing current program.
Control	G	Sounds bell (beeps speaker)
Control	Η	Backspaces cursor and deletes any overwritten characters from computer but not from screen. Apply supplied keyboards have special key "←" on right side of keyboard that provides this functions without using control button.
Control	J	Issues line feed only
Control	V	Compliment to H <sup>C</sup> . Forward spaces cursor and copies over written characters. Apple keyboards have "→" key on right side which also performs this function.
Control	X	Immediately deletes current line.
	*	If BASIC program is expecting keyboard input, you will have to hit carriage return key after typing control C.
# CHARACTER

# DESCRIPTION OF ACTION

A <sub>E</sub>	Move cursor to right
B <sub>E</sub>	Move cursor to left
C <sub>E</sub>	Move cursor down
D <sub>E</sub>	Move cursor up
EE	Clear text from cursor to end of line
F <sub>E</sub>	Clear text from cursor to end of page
@ <sub>E</sub>	Home cursor to top of page, clear text to end of page.

Table A: APPLE II COLORS AS SET BY COLOR =

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

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

<u>Hex</u>	BASIC Example	Description
Display	Mode Controls	
C05Ø C051 C052 C053 C054	10 POKE -16304,0 20 POKE -16303,0 30 POKE -16302,0 40 POKE -16301,0 50 POKE -16300,0	Set color graphics mode Set text mode Clear mixed graphics Set mixed graphics (4 lines text) Clear display Page. 2 (BASIC commands
C055 C056 C057	6Ø POKE -16299,Ø 7Ø POKE -16298,Ø 8Ø POKE -16297,Ø	Set display to Page 2 (alternate) Clear HIRES graphics mode Set HIRES graphics mode
<u>TEXT Mo</u>	<u>de Controls</u>	
0020	90 POKE 32,L1	Set left side of scrolling window to location specified by Ll in range of Ø to 39.
ØØ21	100 POKE 33,W1	Set window width to amount specified by WI. Ll+W1<4Ø. W1≻Ø
ØØ22	110 POKE 34,T1	Set window top to line specified by Tl in range of Ø to 23
ØØ23	12Ø POKE 35,B1	Set window bottom to line specified by Bl in the range of Ø to 23. B1>T1
ØØ24	130 CH=PEEK(36) 140 POKE 36,CH 150 TAB(CH+1)	Read/set cusor horizontal position in the range of Ø to 39. If using TAB, you must add "1" to cusor positior read value; Ex. 140 and 150 perform identical function.
ØØ25	160 CV=PEEK (37) 170 POKE 37,CV 180 VTAB(CV+1)	Similar to above. Read/set cusor vertical position in the range Ø to 23.
ØØ32	190 POKE 50,127 200 POKE 50,255	Set inverse flag if 127 (Ex. 190) Set normal flag if 255(Ex. 200)
FC58	21Ø CALL -936	(@ <sub>E</sub> ) Home cusor, clear screen
FC42	220 CALL -958	(F <sub>F</sub> ) Clear from cusor to end of page

<u>Hex</u>	<u>BASIC Example</u>	Description
FC9C	23Ø CALL -868	( $E_E$ ) Clear from cusor to end of line
FC66	24Ø CALL -922	(J <sup>C</sup> ) Line feed
FC7Ø	250 CALL -912	Scroll up text one line

# <u>Miscellaneous</u>

CØ3Ø	36Ø X=PEEK(-16336) 365 POKE -16336,Ø	Toggle speaker
CØØØ	37Ø X=PEEK(-16384)	Read keyboard; if X>127 then key was pressed.
CØ1Ø	380 POKE -16368,0	Clear keyboard strobe – always after reading keyboard.
CØ61	39Ø X=PEEK(16287)	Read PDL(Ø) push button switch. If X>127 then switch is "on".
CØ62	400 X=PEEK(-16286)	Read PDL(1) push button switch.
CØ63	410 X=PEEK(-16285)	Read PDL(2) push button switch.
CØ58	420 POKE -16296,0	Clear Game I/O ANØ output
CØ59	430 POKE -16295,0	Set Game I/O ANØ output
CØ5A	440 POKE -16294,0	Clear Game I/O AN1 output
CØ5B	450 POKE -16293,0	Set Game I/O AN1 output
CØ5C	460 POKE -16292,0	Clear Game I/O AN2 output
CØ5D	470 POKE -16291,0	Set Game I/O AN2 output
CØ5E	480 POKE -16290,0	Clear Game I/O AN3 output
CØ5F	490 POKE -16289,0	Set Game I/O AN3 output

# APPLE II BASIC ERROR MESSAGES

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



### 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:

ΥN	DSP	NVA	DATA(0)	DATA(1)	, DATA(N)
1			h 1	h <sub>2</sub>	h <sub>n</sub> +1

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

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

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

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

VN	DSP	NVA	DATA(Ø)	DATA(1)	DATA(n)	ST
1			hı	h2	h <sub>n+l</sub>	
		VARIABLE represent valents v	NAME - up to ted in memory with the high	100 character as ASCII equi order bit set	S -	
		DSP (DIS DSP set that dis line num within a	PLAY) BYTE - s in BASIC, init plays this van ber every time program.	set to Øl when tiates a proce riable with th e it is change	s s e d	
		NVA (NEX bytes (f high orde location	T VARIABLE ADD irst low orden er) indicating of the next v	DRESS) – two r, the second g the memory variable.		
		DATA - AS order bit	SCII equivaler t set.	nts with high		
		STRING TI order bit END of s	ERMINATOR (ST) t set characte tring.	) – none high er indicating		

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 1b) and continue until all the variables have been mapped into memory or until they collide with the program statements. In the event of the latter case a memory full error will be generated

\*Top of RAM memory is a function of the amount of memory. 16384 will be the value of "HIMEM:" for a 16K system. The computer keeps track of the amount of memory used for the variable table and program statements. By placing the end memory location of each into \$CC-CD(204-205) and \$CA-CB(203-204), respectively. These are the BASIC memory program pointers and their values can be found by using the statements in Figure 2. CM defined in Figure 1 as the location of the end of the variable tape is equal to the number resulting from statement a of Figure 2. PP, the program pointer, is equal to the value resulting from statement 2b. These statements(Figure 2) can then be used on any Apple II computer to find the limits of the program and variable table.

### FINDING THE VARIABLE TABLE FROM BASIC

First, power up the Apple II, reset it, and use the CTRL B (control B) command to place the system into BASIC initializing the memory pointers. Using the statements from Figure 2 it is found that for a 16K Apple II CM is equal to 2048 and PP is equal to 16384. These also happen to be the values of OMEN and HIMEN: But this is expected because upon using the Bc command both memory pointers are initialized indicating no program statements and no variables.

To illustrate what a variable table looks like in Apple II memory suppose we want to assign the numeric variable A (C1 is the ASCII equivalent of a with the high order bit set) the value of -1 (FF FF in hex) and then examine the memory contents. The steps in this process are outlined in example I. Variable A is defined as equal to -1 (step 1). Then for convenience another variable - B is defined as equal to 0 (step 2). Now that the variable table has been defined use of statement 2a indicates that CM is equal to 2060 (step 3). LOMEN has not been readjusted so it is equal to 2060 (88C). Depressing the "RESET" key places the Apple II into the monitor mode (step 4).

We are now ready to examine the memory contents of the variable table. Since the variable table resides from \$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 Cl is equal to the variable name and is the memory equivalent of "A" and that FF FF is the equivalent of -1. From this, since the variable name is at the beginning of the table and the data is at the end, the variable table representation of A extends from \$800 to \$805. We have then found the memory range of where the variable A is mapped into memory. The reason forthis will become clear in the next section.

# READ/SAVE ROUTINE

The READ/SAVE subroutine has three parts. The first section (lines  $\emptyset$ -1 $\emptyset$ ) defines variable A and transfers control to the main program. Lines 2 $\emptyset$  through 26 represents the Write data to tape routine and lines 3 $\emptyset$ -38 represent the Read data from tape subroutine. Both READ and SAVE routines are executable by the BASIC "GOSUB X" (where X is 2 $\emptyset$  for write and 3 $\emptyset$  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 GOSUB2Ø statement will execute the write to tape process. Record length, variable A, is written to tape first (line 22) followed by the desired data (line 24). When this process is completed control is returned to the main program.

# READING A DATA TABLE

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

### EXAMPLE OF READ/SAVE USAGE

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

### CONCLUSION

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





a) PRINT PEEK(2Ø4) + PEEK(2Ø5)\*256 → PP
b) PRINT PEEK(2Ø2) + PEEK(2Ø3)\*256 → CM

Figure 2

8ØØ C1	8Ø1 ØØ	8Ø2 Ø6	8Ø3 Ø8	8Ø4 FF	8Ø5 FF	8Ø6 C2	8Ø7 ØØ	8Ø8 ØC	8Ø9 Ø8	8ØA ØØ	8ØB ØØ	8ØC ØØ
		Ĺ	΄ Η	L	Н			Ĺ	H			
VAR	DSP	N	VA	DA	TA	VAR	DSP	N	VA	DA	ТА	
NAM		•	1			NAM		Y				
						1				<u> </u>		_1
NAM			í	<b>→</b>		NAM 		Ľ		<b>→</b>		

Figure 3 \$800.80C rewritten with labelling

FIGURE 4b

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

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

1	>A=1 >	Define variable A=-1, then hit RETURN
2	>B=Ø >	Define variable $B=\emptyset$ , then hit RETURN
3	>PRINT PEEK (204) + PEEK (205) * 256	Use statement 2a to find the end of the VARIABLE TABLE
	computer responds with= 2060	
4	> *	Hit the RESET key, Apple moves into Monitor mode.
5	*800.80C	Type in VARIABLE TABLE RANGE and HIT the RETURN KEY.
Com	puter responds with:	
Ø8Ø	Ø- C1 ØØ 86 Ø8 FF FF C2 ØØ	
Ø8Ø	8 ØC Ø8 ØØ ØØ ØØ	

Example 1

Example 2

0 A=0 10 GOTO 100 20 REM WRITE DATA TO TAPE ROUTINE 22 A=CM-LM: POKE 60,4: POKE 61 ,8: POKE 62,5: POKE 63,8: CALL -307 24 POKE 60,LM MOD 256: POKE 61 .LM/256: POKE 62.CM MOD 256 : POKE 63, CM/256: CALL -307 26 RETURN 30 REM READ DATA SUBROUTINE 32 POKE 60,4: POKE 61,8: POKE 62,5: POKE 63,8: CALL -259 34 IF A(0 THEN 38:P=LM+A: IF P> HM THEN 38: CM=P: POKE 60.LM MOD 256: POKE 61.LM/256: POKE 62 ,CM MOD 256: POKE 63,CM/256 : CALL - 259 36 RETURN 38 PRINT \*\*\*\* TOO MUCH DATA BASE \*\* \*":END 100 DIM A\$(1),X(20) 105 FOR I=1 TO 20:X(I)=I: NEXT Ι 108 LM=2048:CM=2106:A=58:HM=16383

XLIST

110 PRINT "20 NUMBERS GENERATED" 120 PRINT "NOW WE ARE GOING TO SAVE THE DATA": PRINT "WHEN YOU ARE R EADY START THE RECORDER IN RECOR D MORE": INPUT "AND HIT RETURN" ,8\$ 130 CALL -936: PRINT "NOW WRITING DA TA TO TAPE": GOSUB 20 135 PRINT "NOW THE DATA IS SAVE" 140 PRINT "NOW WE ARE GOING TO CLEAR THE X(20) TABLE AND READ THE DA TA FROM TAPE" 150 FOR I=1 TO 20:X(I): NEXT I \*X(\*;I;\*)= \*;X(I): HEXT I 160 PRINT "NOW START TAPE RECORDER" : INPUT "AND THEN HIT RETURN" ,8\$ 165 PRINT "A ",A 170 GOSUB 30 180 PRINT "ALL THE DATA READ IN" 190 FOR I-1 TO 20: PRINT \*X(\*:I: •)=";X(I): NEXT I 195 PRINT "THIS IS THE END" 200 END

# INTRODUCTION

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

### Tone Generation

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

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

### Machine Language Program

This machine language program resides in page  $\emptyset$  of memory from  $\$\emptyset 2$  (2) to \$14 (2 $\emptyset$ ).  $\$\emptyset\emptyset$  ( $\emptyset\emptyset$ ) is used to store the relative period (P) between toggling of the speaker and  $\$\emptyset 1$  ( $\emptyset$ 1) is used as the memory location for the value of relative duration ( $\emptyset$ ). Both P and D can range in value from  $\$\emptyset\emptyset$  ( $\emptyset$ ) 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  $\$\emptyset 2$  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 Ø and 255 a GOSUB 25 statement is used to initiate the generation of a tone. The values of P and D are placed into \$00 and \$01 and the CALL2 command activates the machine language program that toggles the speaker. After the tone has ended control is returned to the main program.

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

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

## CONCLUSION

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

0000-	FF			???	
0001-	FF			255	
0002-	AD	30	CØ	LDA	\$C030
0005-	88			DEY	
0006-	DØ	04		BNE	\$000C
0008-	C6	01		DEC	\$01
000A-	FØ	08		BEQ	\$0014
000C-	CA			DEX	
000D-	DØ	F6		BNE	\$0005
000F-	A6	00		LDX	\$00
0011-	4C	92	00	JMP	\$0002
0014-	60			RTS	

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

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

### FIGURE 2. BASIC "POKES"

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

FIGURE 3. GOSUB

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

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

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

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

INIT Initiates High-Resolution Graphics mode. From BASIC: CALL 3072 (or CALL -12288) From machine language: JSR \$C00 (or JSR \$D000)

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

<u>CLEA</u>R Clears the screen. From BASIC: CALL 3886 (or CALL -12274) From machine language: JSR SCOE (or JSR \$L000E)

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

<u>PLO</u>T Plots a point on the screen. From BASIC: CALL 3780 (or CALL -21589) From machine language: JSR \$C7C (or JSR \$L107C)

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

(top of screen) to 159 (bottom of screen) and is passed in location 802 or the A-register; but the X (horizonțal) coordinate can range from \$\$ (left side of screen) to 279 (right side of screen) and must be split between locations 8\$\$\$ (X MOD 256) and 8\$\$\$ (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, 17\$\$ (\$AA) is VIOLET, and 255 (\$FF) is WHITE.

<u>POSN</u> Positions a point on the screen. From BASIC: CALL 3761 (or CALL -11599] From machine language: JSR \$C26 (or JSR \$D\$/26)

This subroutine does all calculations for a PLOT, but does not plot a point (it leaves the screen unchanged). This is useful when used in conjumction with LINE or SHAPE (described later). To use this subroutine, set up the X and Y coordinates just the : same as for PLOT. The color in location 812 (\$326) is ignored.

LINE Draw a line on the screen.

# High-Resolution Operating Routines

LINE Draws a line on the screen.

From BASIC: CALL 3786 (or CALL -11574) From machine language: JSR \$C95 (or JSR \$DØ95)

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

SHAPE Draws a predefined shape on the screen.
From BASIC: CALL 38\$5 (or CALL -11555)
From machine language: JSR \$DBC (or JSR \$DIBC)

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

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

# SHAPE (continued)

X

6

1 will cause the shape to be drawn true size, while a scaling factor of 2 will draw the shape double size, etc. The scaling factor is passed in location 806 from BASIC or \$32F from machine language. The rotation factor specifies one of 64 possible angles of rotation for the shape. A rotation factor of  $\beta$  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 form BASIC of in the A-register from machine language.

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

The three sections are arranged in a byte like this: T 00= 00 Move 5 6 n Ŀ D 4 ۰. Section 2. 5 c. 2410-Sector 3 Each bit pair DD specifies a direction to move, and the two bits P specify whether or not to plot a point before moving. Notice that the last section (most significant bits) does not have a P field, so it can only be a move without plotting. The SHAPE

SHAPE (continued)

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

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

Suppose we want to draw a shape like this:

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

only 90° angles on the turns.



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

っ			-7	44	
5				Ŷ	
5		4		2	
77		5			Ē
オ	Z	4	K-	K-	[]]
	12.1			_	

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

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

# SHAPE (continued)

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

>REM HIRES DEMO-BASIC LISTING

#### XLIST

- 1 INIT=3072:CLEAR=3086:POSN=3761 :PLOT=3780:LINE=3786:SHAPE= 3805:FIND=3667:SINTBL=3840
- 5 DIM X(10),Y(10)
- 10 TEXT : CALL -936: VTAB 4: TAB 10: PRINT \*\*\*\* 16 APPLE II \*\*\*\* : PRINT \* \*\*\* HIGH RESOLUTION G RAPHICS DEMOS \*\*\*\*: PRINT
- 15 PRINT "1 RANDOM LINE DRAW AT BAS IC SPEED":PRINT "2 RANDOM SHAPE PROJECTED INTO CORNER"
- 20 PRINT "3 CHRIS" MAD FOLLY": PRINT "4 RANDOM SHAPE SPIRALING INTO POINT": PRINT "5 SPIROGRAP H"
- 25 PRINT "6 HI-RES DONUT": PRINT "7 RANDON WAVE FORM": PRINT "8 SUM OF TWO SINE WAVES"
- 30 PRINT : PRINT "HIT ANY KEY FOR N EW DEMO": PRINT "TYPE "CONTROL C " ; RETURN BUTTON THEN TYPE "T EXT AND RETURN BUTTON TO STOP"
- 50 PRINT : INPUT "WHICH DENO # DO Y OU WANT ",X1
- 90 IF X1<1 OR X1>8 THEN 10: CALL INIT: GOTO 100\*X1
- 100 CALL INIT:X=4-:Y=X: GOSUB 2000 : POKE 812,255: CALL PLOT
- 110 X= RND (200):Y=RND (160): GOSUB 2000: CRLL LINE: IF NOT RND (300) THEN POKE 23,(PEEK ( 28)+ RND (3)+1) MOD 4\*85: GOSUB 3000: GOTO 110
- 200 GOSUB 1000:X= RND (2)=279:Y= RND (2)\*159: CALL PLOT: FOR J=1 TO 30: FOR I=1 TO R: POKE 800,X(I) MOD 256: POKE 801, X(I))255: POKE 802,Y(I): CALL LINE

- 530 IF RND (500)<C THEN POKE 28 , RND (4)\*85:Y=Y+YDIR\*8: IF Y>0 AND Y<160 THEN 510:YDIR= -YDIR:Y=-Y: IF Y<0 THEN Y=Y+ 318: GOSUB 3000: GOTO 510 600 POKE -16302,0: POKE 768,5: POKE
- 769,0: POKE 800,140: POKE 801 ,0: POKE 802,0: POKE 804,0: POKE 805:3: POKE 812,255: CALL POSN
- 610 FOR R=0 TO 4160: POKE 807,R MOD 64: POKE 806,2+6\* NOT (R MOD 65): CALL SHAPE: NEXT R: GOSUB 3000: GOTO 610
- 700 J= RHD (10)+ RHD (10):K= RHD (33)+ RHD (31)+ RHD (60):L= RHD (9)/8: PRINT \*FREQ#1 \* ;J;\* FREQ#2= \*;K
- 710 GOSUB 4000: GOSUB 3000: GOTO 700
- 800 INPUT "REL FREQ #1=",J: INPUT "REL FREQ #2=",K: INPUT "MODE (0 =SOLID, 1=POINTS)",L
- 810 GOSUB 4000: GOSUB 3000: GOTO 800
- 1000 CALL CLEAR: POKE 812, RND( 3)\*85+85:R= RND (3)+2+ RND (2): FOR I=1 TO R:X(I)= RND (160):Y(I)= RND (160): NEXT I
- 1010 X=X(1):Y=Y(1): GOSUB 2000: RETURN 2000 POKE 800,X MOD 256: POKE 801 ,X)255: POKE 802,Y: RETURN
- 3000 IF PEEK (-16384)(128 THEN RETURN : POKE -16386,0: POP : GOTO 10 4000 CALL INIT: POKE 812.255:A=0
- :B=0: FOR 1=0 TO 279:A=(A+J) HOD 256:B=(B+K) MOD 256:Y= ( PEEK (SINTBL+A)+ PEEK (SINTBL+ B))\*5/16
- 4010 POKE 800,I MOD 256: POKE 801 ,I>255: POKE 802,Y: CALL LINE 6\*(NOT I OR L): NEXT I: RETURN

- 210 X(I)=(X(I)-X)\*9/10+X:Y(I)=( Y(I)-Y)\*9/10+Y: NEXT I,J: GOSUB 3000: GOTO 200
- 300 CALL INIT:X= RND (24)\*10+20 :Y= RND (14)\*10+20: POKE 812 , RND (3)\*85+85: GOSUB 2000 ; CALL PLOT
- 310 IF RND (1000)<1 THEN 300: IF NOT RND (200) THEN POKE 28, RND (4)\*85
- 320 X1=X+( RHD (3)-1)\*25:Y1=Y+( RHD (3)-1)\*15: IF X1<0 OR X1>279 OR Y1<0 OR Y1>159 THEN 320
- 330 X=X1:Y=Y1: GOSUB 2000: CALL LINE: GOSUB 3000: GOTO 310
- 400 GOSUB 1000: POKE 812, RND( 3)\*85+85: CALL PLOT
- 410 FOR J=1 TO 25: FOR I=1 TO R: POKE 800,X(I) MOD 255: POKE 801,X)255: POKE 802,Y(I): CALL LINE
- 420 X=(X(I)-80+(Y(I)-80)/8)\*9/10 +80:Y(I)=(Y(I)-88-(X(I)-80) /8)\*9/10+80:X(I)=X: NEXT I, J: GOSUB 3000: GOTO 400
- 500 CALL INIT: POKE 800,0: CALL PLOT:X=0:Y=0:XDIR=1:YDIR=1: A=5:B=3:C=8
- 510 POKE 800,0: POKE 801,0: POKE 802,Y: CALL LINE: POKE 800, (279-X) MOD 256: POKE 801,X( 24: POKE 802,159: CALL LINE: POKE 800,23: POKE 801,1: POKE 802,159-Y: CALL LINE
- 515 IF RND (500) THEN 520:A=1+ RND (13):B=2+ RND (8):C=4+ RND (7)
- 520 POKE 800,X MOD 256: POKE 801 ,X>255: POKE 802,0: CALL LINE: X=X+XDIR\*A: IF X>0 AND X<288 THEN 530:XDIR=-XDIR:X=-X: OF X<0 THEN X=X+558

### ROD'S COLOR PATTERN

### PROGRAM DESCRIPTION

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

# REQUIREMENTS

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

PROGRAM LISTING

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

7/7/77 10 REM 15 REM PADDLE SWITCHES CONTROL PADDLE SIZE AFTER A MISS OR DURING A HIT 20 GR 25 DIM P(3): DIM HP\$(10) 30 = 38:8=1:0=-1 35 COLOR=13: HLIN 1.38 AT 0: HLIN 1.38 AT 39 40 CALL -936: VTAB 23: INPUT \*HANDB ALL OR PONG ? ", HP\$ 45 INPUT "PADDLE SIZE (1-6) ", PS: IF PS(1 OR PS)6 THEN 45 :S=PS-1 50 CALL -936 55 IF HP\$(1)#"H" THEN 205 60 H=1: COLOR=13: VLIN 0.39 AT 39: GOTO 205 65 FOR X=A TO B STEP C 70 Y=YY+Y: IF Y>1 AND Y<38 THEN 80 IF Y<1 THEN Y=1: IF Y>38 THEN Y=38 75 V=-V: FOR T=1 TO 5:M= PEEK (-15336): NEXT T 80 IF X=C OR X=39+C THEN 85: COLOR= 0: PLOT X-C,YY: COLOR=15: PLOT X.Y 85 YY=Y: IF X MOD 2=0 THEN GOSUB 235: NEXT X 90 GOSUB 235 95 IF SCRN(X.Y+V\*(Y+V(40 AND Y+ Y>-1>>=0 THEN 165 100 FOR T=1 TO 10:M= PEEK (-16336 ): NEXT T 105 IF H AND C>0 THEN 130 110 PP=P(X/38) 115 IF Y=PP THEN V=3: IF Y=PP+1 THEN V=2: IF Y=PP+2 THEN V= 1

5 REM PONG BY WENDELL BITTER 120 IF Y=PP+3 THEN Y=-1: IF Y=PP+ 235 IF H THEN 245:P(1)=(( PDL ( 4 THEN V=-2: IF Y=PP+5 THEN V=-3 125 IF S=0 THEN V=3- RND (7) 130 COLOR=0: PLOT X-C.Y 135 IF (H AND C>0) OR (VY0= ABS (Y) AND X=0) THEN V=4- RND (9) 140 IF X=0 THEN VY0= ABS (V) 145 A=39-A:B=30-B:C=-C 150 IF PEEK (-16286))127 AND S# 245 P(0)=(( PDL (0)-24)\*20)/115 5 THEN S=S+1 155 IF PEEK (-16287)>127 AND S# 0 THEN S=S-1 160 GOTO 65 165 COLOR=0: PLOT X-C.Y 170 COLOR=15: PLOT X.Y+V\*(Y+V)-1 AND Y+V(40) 175 FOR T=1 TO 75:M= PEEK (-16336 )+ PEEK (-16336)- PEEK (-16336 255 COLOR=0: IF P(0))P(2) THEN ): NFXT T 180 IF X=0 THEN SR=SR+1: IF X=39 THEN SL=SL+1 185 VTAB 23: TAB 7: PRINT SL;: TAB 260 PRINT ••: END 33: PRINT SR 190 COLOR=0: PLOT X-C, Y 195 IF SL=15 OR SR=15 THEN 260 200 COLOR=0: PLOT X.Y+V\*(Y+V)-1 AND Y+VY(40) 205 FOR T=1 R0 75: IF T MOD 5\*0 THEN 210: IF PEEK (-16286) >127 AND S#5 THEN S=S+1: IF PEEK (-16287)>127 AND S#0 THEN S=S-1 210 GOSUB 235: NEXT T 215 YY=P(0): IF X=0 THEN YY=P(1 ) 220 IF H THEN YY= RND (37)+1 225 V=1- RHD (3) 230 GOTO 65

1)-24)\*20)/115: IF (1)=P(3 ) THEN 245: IF P(1)(0 THEN P(1)=0: IF P(1)+S>39 THEN P( 1)=39-5 240 COLOR=6: VLIN P(1).P(1)+5 AT 39: COLOR=0: IF P(1))P(3) THEN VLIN 0.P(1)-1 AT 39: IF P(1

XP(3) THEN VLIN P(1) AT 39:P(3)=P(1) : IF P(0)(0 THEN P(0)=0: IF P(0)=P(2) THEN RETURN : IF P(0)+5>39 THEN P(0)=39-5 250 COLOR=6: VLIN P(0).P(0)+5 AT 0: COLOR=0: IF P(0)>P(2) THEN VLIN 0.P(0)-1 AT 0: IF P(0)

- (P(2) THEN VLIN P(0)+5+1.39 AT Ø
- VLIN 0.P(0)-1 AT 0: IF P(0) (P(2) THEN VLIN P(0)+S+1.39 AT 0:P(2)=P(0): RETURN
- 265 END

## **PROGRAM DESCRIPTION**

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

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

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

### PROGRAM LISTING: COLOR SKETCH

5 POKE 2.173: POKE 3.48: POKE 4,192: POKE 5,165: POKE 6,8 : POKE 7,32: POKE 8,168: POKE 9.252: POKE 10.165: POKE 11 ,1: POKE 12,208: POKE 13,4 10 POKE 14,198: POKE 15,23: POKE 16.248: POKE 17.5: POKE 18. 188: POKE 19.2: POKE 28.76: POKE 21,2: POKE 22,0: POKE 23,96 15 DIM B\$(40): TEXT: CALL -936 : GOTO 90 20 CALL -936: GOTO 98 65: PRINT B\$(Z.Z):: NEXT Z: GOSUB 70: RETURN 30 8\$="\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 35 B\$="COLOR SKETCH": RETURN 40 B\$=COPYRIGHT APPLE COMPUTER 197 7": RETURN 45 B\$="THIS PROGRAM ALLLOWS YOU TO " : RETURN 50 B\$="SKETCH COLORED FIGURES IN" 120 YTAB 22-B\$="TYPE A LETTER TO CH : RETURN 55 B\$="LOW RESOLUTION GRAPHICS WITH PADDLES": RETURN 60 KK=20:TON=28: GOSUB 85: RETURN 65 KK=10: TON=10: GOSUB 85: RETURN 70 KK=30:TON=50: GOSUB 85:KK=30 :TON=98: GOSUB 85: RETURN 75 KK=20: TON=20: GOSUB 85: RETURN 80 KK=0:TON=250: GOSUB 85:KK=9 :TON=250: GOSUB 85: RETURN

85 POKE 1.TOM MOD 256: POKE 24 ,TON/256+1: POKE 0,KK: CALL 2: RETURN 90 GOSUB 30: GOSUB 25: PRINT : TAB 10: GOSUB 35: GOSUB 25 : PRINT : GOSUB 30: GOSUB 25 : PRINT : TAB 5: GOSUB 40: GOSUB 25: PRINT : GOSUB 30: GOSUB 25 95 PRINT: GOSUB 70: GOSUB 45: GOSUB 25: PRINT : GOSUB 50 : GOSUB 25: PRINT : GOSUB 55 : GOSUB 25: PRINT 25 A=LEN(B\$): FOR Z-1 TO A: GOSUB 100 PRINT : PRINT : GOSUB 70: INPUT 155 FLAG=1: C=PEEK (-16384)-193 •WHEN READY HIT RETURN•,B\$ 105 GR 110 B\$="ABCDEFGHIJKLMHOP": CALL -936 115 FOR Z=8 TO 15: COLOR=Z: PLOT 2\*2+4,39: YTAB 21: GOSUB 75 : TAB Z\*2+5: PRINT 2\$(Z+I).Z+ 2):: GOSUB 75: NEXT 2: TAB 1 ANGE COLOR. ": GOSUB 25: PRINT :B\$="TYPE SPACE BAR TO STOP PLOT .": GOSUB 25: PRINT 125 Y=PDL (1)\*38/255:X= PDL (8 )\*39/255: YTAB 24: TAB1: PRINT "CURSOR POSITION: X=";X;" Y=" ;Y;' ';: 130 IF PEEK (-16384))127 THRN 145 : IF X1=X AND Y1=Y THEN 126 : COLOR=C2: PLOY X1.Y1: IF NOT FLAG THEN 135: COLOR=C: PLOT X.Y

15 THEN C3=5: COLOR=C3: PLOT X,Y: X1=X:Y1=Y 140 GOTO 125 145 IF PEEK (-16384)\*160 THEN 155 :FLAG=0: POKE -16368,0: POKE 34,28: COLOR=0: HLIN 0,39 AT 39: CALL -936 150 PRINT :B\$="CONTINUE OR STOP" : VTAB 24: GOSUB 25: INPUT " (C/5) ".B\$: IF B\$(1.1)="C" THEN 110: PRINT "END" :END

135 C2=SCRN(X.Y): C3=15: IF C2=

: POKE -16368.0: GOTO 125

### MASTERMIND PROGRAM

### PROGRAM DESCRIPTION

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

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

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

0 REM GAME OF MASTERMIND 8-25-77 WOZ (APPLE COMPUTER) 10 DIM A(6),C(8),D(5),X(8),X\$( 8):X(1)=2:X(2)=12:X(3)=1:X( 4)=13:X(5)=3:X(6)=9:X(7)=15

:X(8)=5:X\$="BRGYVOWX" 20 TEXT : CALL -936: PRINT •

#### WELCO

ME TO THE GAME OF MASTERMIND!

YOUR OBJECT US TO GUESS 5 COLOR S (WHICH" 30 PRINT "I WILL MAKE UP) IN THE MI NIMUM NUMBER OF GUESSES. THER

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

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

100 CALL -380: IF PEEK (-16385) (132 THEN 100: POKE -16368. 0: GR : PRINT : FOR I=1 TO 8:C(I)= RND (8)+1: COLOR=X( I): HLIN I\*4-2.I\*4 AT 39: PRINT • •:X\$(I.I):: NEXT I 110 TRY=0: PRINT : PRINT • LETTER KEYS FOR COLOR CHANGE": PRINT ARROW KEYS FOR ADVANCE AND BA CK": PRINT " HIT RETURN TO ACC EPT GUESS #";

200 Y=TRY\*2 MOD 36+1:TRY=TRY+1: 3000 REM CALL -384 SETS INVERSE VID TAB 32: PRINT TRY:: COLOR= 0: HLIN 0.39 AT Y:FLASH=1: FOR N=1 TO 5:A(N)=8: GOSUB 1000 : NEXT N:N=1 300 FOR WAIT=1 TO 10:KEY= PEEK (-16384): IF KEY(132 THEN 310 : POKE -16386.0:FLASH=1: FOR I=1 TO 8: IF KEY(> ASC(X\$(I) ) THEN NEXT I: IF I=9 THEN 310:A(N)=I:KEY=149 310 GOSUB 1000: IF KEY=141 THEN 400: IF KEY=136 AND N>1 OR KEY=149 AND N(6 THEN N=N+KEY/ 5-28: NEXT WAIT:FLASH=1-FLASH: 4050 REM STMTS 500-5100 WIN GOTO 300 E ARE EIGHT DIFFERENT COLORS TO 400 COLOR=15:N=0: FOR I=1 TO 5: D(I)=C(I):J=I: GOSUB 2000: NEXT I: IF N=5 THEN 500: COLOR=5 : FOR J=1 TO 5: FOR I=1 TO 5: GOSUB 2000: NEXT I.J: GOTO 200 500 PRINT : PRINT \* YOU GOT IN . ;TRY;" TRIES (";: IF TRY(7 THEN PRINT "EXCELLENT":: IF TRY> 6 AND TRY <10 THEN PRINT "GOOD" ; 510 IF TRY>9 AND TRY<15 THEN PRINT "AVERAGE";: IF TRY>14 THEN PRINT "POOR":: PRINT ")": CALL -384: TAB 5: PRINT "HIT THE KEY TO PLAY AGAIN": GOTO 100 1000 IF N=6 THEN RETURN : COLOR= X(A(N))\*FLASH: HLIN N\*4-2.N\* 4 AT Y: RETURN 2000 IF A(I)()D(J) THEN RETURN : M=M+1: PLOT 21+M+M,Y: PRINT ""::A(I)=0:D(J)=9: RETURN

3010 REM CALL -380 SETS NORMAL VID 3020 REM PEEK(-16384) IS KBD (ASCII) (IF > 127 THEN STROBE SET) 3030 REM POKE-16386 CLRS KBD STROBE 3040 REM CALL-936 CLEARS SCREEN AND TABS CURSOR TO UPPER LEFT. 3050 REM IN 310. KEY/5-28= -1 OR +1 (ARROW KEY=136 OR 149 ASCII) 4000 REM STMTS 10-50 INTRO 4010 REM STMTS 100-110 NEW SETUP 4020 REM STMT 200 NEW GUESS 4030 REM STMTS 300-310 USER INPUT 4040 REM STMT 400 GUESS EVAL 4060 REM SUBR 1000 COLOR LINE 4070 REM SUBR 2000 MATCH TEST

### PROGRAM DESCRIPTION

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

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

# Physical

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

### Emotional

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

### Mental

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

### Biorhythms

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

### REQUIREMENTS

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

- 5 POKE 2.173: POKE 3.48: POKE 4.192: POKE 5.165: POKE 6.8 : POKE 7.32: POKE 8.168: POKE 9.252: POKE 10.165: POKE 11 ,1: POKE 12,208: POKE 13.4 10 POKE 14,198, POKE 15,24: POKE 16,240: POKE 17,5: POKE 18, 198: POKE 19.1: POKE 20.76: POKE 21.2: POKE 22.8: POKE 23,96 15 GOTO 85 20 TT=3: GOSUB 30: RETURN 25 PRINT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 30 KK=8: TON=500: GOSUB 45: RETURN 35 KK=8: TON=250: GOSUB 45: RETURN 40 KK=8: TON=250: GOSUB 45: KK=0 :TON=250: GOSUB 45: RETURN 45 POKE 1.TON MOD 256: POKE 23 ,TON/256+1: POKE 0,KK: CALL 2: RETURN 50 A=(19-(P\*B(I)/100))\*(P\*100( C(I))+(P\*100)C(I))\*(P\*100(= 3\*C(I))\*((P\*100-C(I))/100\*B( I)/100) 55 A=A+(P\*100>3\*C(I))\*(38-((P\* 100-3\*C(I))/100\*B(I)/100)): A=39\*(A>39)+A\*(A(40): RETURN
- 60 KK=8:TN=500: GOSUB 70:KK=9: TM=250: GOSUB 70: RETURN 65 KK=7:TM=10: GOSUB 70: RETURN

70 POKE 1.TM MOD 256: POKE 24. 110 J=1: GR : POKE 34.23: FOR X= TM/256+1: POKE 0.KK: CALL 2 : RETURN 75 GOSUB 60: INPUT "DATE (M,D,Y,) " .M.D.Y:Y=Y+(Y(100)\*1900 80 R=Y-(M(3):N=Y MOD 58\*365-Y/ 58\*82+A/4-A/400+M\*31-M/12-M/ 7-M/5-3\*(M)2)+D: IF N(0 THEN N=N+21252: RETURN 85 DIM M\$(10),B\$(3),B(3),C(3), BY(3):B(1)=348:B(2)=286:B(3 )=242:C(1)=575:C(2)=700:C(3 )=825:BY(1)=23:BV(2)=28 90 BV(3)=33: TEXT : CALL -936: POKE 34.20: GOSUB 20: GOSUB 25: GOSUB 20: PRINT : TAB 10 : PRINT • APPLE II BIORHYTHM (4K) ": TAB 15: PRINT 95 GOSUB 25: TAB 5: PRINT \*COPYRIGH T 1977 APPLE COMPUTER INC. :POKE 34,24: VTAB 24 100 GOSUB 60: INPUT "NAME ".N\$: VTAB 22: PRINT N\$: VTAB 24 : PRINT \*BIRTH \*:: GOSUB 75 : VTAB 22: TAB 21: PRINT \*BIRTH DATE ":M: ". ":D: ". ":Y: VTAB 24:N1=N: CALL -868 105 PRINT "FORECASE ";: GOSUB 75 :N=N-N1: IF N(0 THEN N=N+21252 : VTAB 23: TAB 18: PRINT "FORECA ST DATE "; M; ", "; D; ", "; Y: VTAB 24: CALL -868

- 18 TO 20: COLOR=3: HLIN 0.31 AT X: NEXT X: HLIN 1.3 AT 3: HLIN 1,3 AT 37: VLIN 2,4 AT 2: YTAB 21 115 FOR Y=1 TO 31 STEP 3: PRINT
  - Y;: IF Y<10 THEN PRINT •; : PRINT " ":: NEXT Y: PRINT • P E M•: VTAB 24
- 120 YTAB 23: PRINT \*DAYS LIVED \* :N: FOR I=1 TO 3: COLOR=1\*( I=1)+6\*(I=2)+8\*(I=3): VLIN 0,39 AT 33+I+I: VTAB 24
- 125 FOR X=0 TO 31:P=(N MOD BV(I) +X) MOD BY(I): GOSUB 50: PLOT X.A: GOSUB 65: NEXT X: NEXT Ι
- 130 PRINT : INPUT "ANOTHER PLOT (Y/N) ) ".B\$: IF B\$(1.1)="Y" THEN 90: END

 $\rangle$ 

### PROGRAM DESCRIPTION

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

# REQUIREMENTS

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

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

1 TEXT : CALL -936

- 2 PRINT "WELCOME TO THE DRAGON'S M AZE!"
- 3 PRINT "YOU MAY WATCH WHILE I BUI LD A MAZE."
- 4 PRINT "BUT WHEN IT'S COMPLETE, I "LL ERASE" 5 PRINT "THE PICTURE. THEN YOU'LL
- ONLY SEE THE WALLS AS YOU BUMP I NTO THEM.
- 6 PRINT "TO MOVE, YOU HIT "R" FOR RIGHT,"
- 7 PRINT "'L' FOR LEFT, 'U' FOR UP, AND"
- 8 PRINT "'D' FOR DOWN. DO NOT HIT RETURN!"
- 9 PRINT
- 10 PRINT "THE OBJECT IS FOR YOU (TH E GREEN DOT"
- 11 PRINT "TO GET TO THE DOOR ON THE RIGHT SIDE"
- 12 PRINT "BEFORE THE DRAGON (THE RE D DOT) EATS"
- 13 PRINT "YOU."
- 14 PRINT "BEWARE!!!!!!!! SOMETIMES THE DRAGON"
- 15 PRINT "GETS REAL MAD, AND CLIMBS OVER A WALL."
- 16 PRINT "BUT MOST OF THE TIME, HE CAN'T GO OVER"
- 17 PRINT "AND HAS TO GO AROUND."

18 PRINT

19 PRINT "(HINT, YOU CAN OFTEN TELL WHERE A WALL"

20 PRINT \*IS. EVEN BEFORE YOU CAN S EE IT. BY" 21 PRINT THE FACE THAT THE DRAGON CAN 'T GET 22 PRINT "THROUGH IT!)" 23 PRINT 89 DIM A\$(3) 90 PRINT "TYPE 'GO' TO BEGIN" :: INPUT A\$ 100 GR : COLOR=15 105 CALL -936: PRINT "DRAGON MAZE" ;: TAB (25): PRINT "GARY J. SHAN NON . 110 FOR I=0 TO 39 STEP 3: YLIN 0.39 AT I: HLIN 0.39 AT I: NEXT I 120 COLOR=0 130 S=1000 1000 DIM M(169),T(I)=0: NEXT 1001 FOR I=1 TO 169:T(I)=0: NEXT I 1010 FOR I=1 TO 169:M(I)=11: HEXT Ī 1030 X= RND (13)+1:Y= RND (13)+1 :C=169 1035 IF C=1 THEN 120 1040 R=0:D=0:L=0:U=0:K=X+13\*(Y-1 ):M(K)= ABS (M(K)):C=C−1 1050 IF X=13 THEN 1060:R=M(K+1)> 0 1060 IF Y=13 THEN 1070:D=M(K+13) ١Ø 1070 IF X=1 THEN 1030:L=M(K-1))0 1080 IF Y=1 THEN 1090:U=M(K-13)> Й

1090 Q=R+D+L+U 1100 IF (Q<3 AND RND (10)<2) OR Q=0 THEN 1178 1110 DR= RND (4) 1120 GOTO 1130+10\*DR 1130 IF NOT R THEN 1110:M(K)=M(K) +1:X=X+1 1135 VLIN 3\*Y-2.3\*Y-1 AT 3\*(X-1) 1136 GOTO 1035 1140 IF NOT D THEN 1110:M(K)=M(K) +10:Y=Y+1 1145 HLIN 3\*X-2.3\*X-1 AT 3\*(Y-1) 1146 GOTO 1035 1150 IF NOT L THEN 1110:M(K-1)=M( K-1)-1:X=X-1 1155 YLIN 3\*Y-2.3\*Y-1 AT 3\*X 1156 GOTO 1035 1160 IF NOT U THEN 1110: M(K-13)= ₩(K-13)-10:Y=Y-1 1165 HLIN 3\*X-2.3\*X-1 AT 3\*Y: GOTO 1035 1170 X= RND (13)+1:Y= RND (13)+1 1180 IF M(X+13\*(Y-1)))0 THEN 1170 1190 C=C+1: GOTO 1035 1200 GOSUB 5000: PRINT "THE MAZE IS R EADY \* 1205 GR : COLOR=15 1210 VLIN 0,39 AT 0: VLIN 0,39 AT 39: HLIN 0,39 AT 0: HLIN 0, 39 AT 39 1220 X=1:Y= RND (13)+1: COLOR=8: PLOT 3\*X-2,3\*Y-2
1225 HX=3\*X-2:HY=3\*Y-2 2520 GOTO 2020 1230 WY= RHD (13)+1 1240 COLOR=0: VLIN 3\*WY-2,3\*WY-1 3010 IF M(X+13\*(Y-2))/10 THEN 4200 AT 39 1250 SX=13:SY=WY 1260 QX=3\*SX-2:QY=3\*SY-2 1270 RD=1 1500 K= PEEK (-16384): IF K(128 THEN 1500 1510 POKE -16368.0 1515 QQ=K: GOSUB 7000:K=QQ 1516 IF SX=X AND SY=Y THEN 8000 1520 IF K= ASC( "R") THEN 2000 1530 IF K= ASC("L") THEN 2500 1540 IF K= ASC( "U") THEN 3000 1550 IF K= ASC( \*D\*) THEN 3500 1560 GOSUB 5000: GOTO 1500 2000 DX=1:DY=0 2010 IF M(X+13\*(Y-1)) MOD 10 THEN 4000 2020 FX=3\*X-2:FY=3\*Y-2: FOR I=1 TO 4220 HLIN 3\*(X-1),3\*X AT 3\*(Y-1) 3 2030 FX=FX+DX:FY=FY+DY 2040 COLOR=0 2060 FOR K=0 TO 1: FOR L=0 TO 1: 4310 COLOR=15 PLOT HX+K,HY+L: NEXT L,K: COLOR= 4320 VLIN 3\*(X-1),3\*X AT 3\*Y 8: FOR K=0 TO 1: FOR L=0 TO 1: PLOT FX+K.FY+L: NEXT L.K: HX=FX:HY=FY 2110 NEXT I 2115 X=X+DX:Y=Y+DY 2116 IF X=13 AND Y=WY THEN 6000 2120 GOTO 1500 2500 DX=-1:DY=0 2510 IF M(X+13\*(Y-1)-1 MOD 10 THEN 6020 PRINT "SCORE=";5+3 4100

3000 DX=0:DY=-1 3020 GOTO 2020 3500 DX=0:DY=1 3510 IF M(X+13(Y-1))/10 THEN 4300 3520 GOTO 2020 4000 GOSUB 5000 4010 COLOR=15 4020 VLIN 3\*(Y-1),3\*Y AT 3\*X 4030 GOTO 1500 4100 GOSUB 5000 4110 COLOR=15 4120 VLIN 3\*(Y-1).3\*Y AT 3\*(X-1) 4130 GOTO 1500 4200 GOSUB 5000 4210 COLOR=15 4230 GOTO 1500 4300 GOSUB 5000 4330 GOTO 1500 5000 S=S-1: FOR I=1 TO 20:R= PEEK 13\*(SY-1)))9 THEN 7110: IF (-16336)+ PEEK (-16336)+ PEEK (-16336)+ PEEK (-16336): NEXT I: RETURN 6000 PRINT "YOU WIN!" 6010 GOSUB 5000: GOSUB 5000: GOSUB 5000

6030 END

7000 IF X>SX THEN 7005: IF Y>SY THEN 7050 7001 IF X<SX THEN 7100: IF Y<SY THEN 7150 7005 IF SX=13 THEN 7050: IF Y(SX+ 13\*(SY-1))>9 THEN 7010: IF M(SX+13(SY-1)) MOD 10 THRN 7050 7010 DX=1:DY=0 7020 COLOR=0 7022 RX=3\*5X-2:RY=3\*5Y-2 7023 FOR I=1 TO 3:RX=RX+DX:RY=RY+ DY 7024 COLOR=0 7025 FOR K=0 TO 1: FOR L=0 TO 1: PLOT QX+K.QY+L: NEXT L.K: COLOR= RD: FOR K=0 TO 1: FOR L=0 TO 1: PLOT RX+K.RY+L: NEXT L.K: QX=RX:QY=RY 7030 NEXT I 7035 SX=SX+DX:SY=SY+DY 7040 T(SX+13 THEN 7100: IF T(SX+ 13\*(SY-1))>9 THEN 7060: IF M(SX+13\*(SY-1))/10 THEN 7100 7060 DX=0:DY=1: GOTO 7020 7100 IF SX=1 THEN 7150: IF T(SX+

M(SX+13\*(SY-1)-1) MOD 10 THEN 7150

#### DRAGON MAZE cont.

7110 DX=-1:DY=0: GOTO 7020 7150 IF SY=1 THEN 7005: IF T(SX+ 13\*(SY-1)))9 THEN 7160: IF M(SX+13\*(SY-1)-13)/10 THEN 7005 7160 DX=0:DY=-1: GOTO 7020 8000 GOSUB 5000: GOSUB 5000: GOSUB 5000: GOSUB 5000: PRINT "THE DRA GON GOT YOU!" 89999 END

 $\rangle$ 

# APPLE II FIRMWARE

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

#### System Monitor Commands

Apple II contains a powerful machine level monitor for use by the advanced programmer. To enter the monitor either press RESET button on keyboard or CALL-151 (Hex FF65) from Basic. Apple II will respond with an "\*" (asterisk) prompt character on the TV display. This action will not kill current BASIC program which may be re-entered by a C<sup>C</sup> (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.

<u>Command</u> Format	Example	Description
Examine Memory		
adrs	*CØF2	Examines (displays) single memory location of (adrs)
adrsl.adrs2	*1024.1048	Examines (displays) range of memory from (adrsl) thru (adrs2)
(return)	*(return)	Examines (displays) next 8 memory locations.
.adrs2	*.4096	Examines (displays) memory from current location through location (adrs2)
Change Memory		
adrs:data data data	*A256:EF 2Ø 43	Deposits data into memory starting at location (adrs).
:data data data	*:FØ A2 12	Deposits data into memory starting after (adrs) last used for deposits.
Move Memory		
adrsl≺adrs2. adrs3M	*100 <b010.b410m< td=""><td>Copy the data now in the memory range from (adrs2) to (adrs3) into memory locations starting at (adrsl).</td></b010.b410m<>	Copy the data now in the memory range from (adrs2) to (adrs3) into memory locations starting at (adrsl).
Verify Memory		
adsr1≺adrs2 adrs3V	*100 <b010.b410v< td=""><td>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.</td></b010.b410v<>	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.

<u>Command Format</u>	Example	Description
<u>Cassette I/O</u>		
adrsl.adrs2R	*300.4FFR	Reads cassette data into specified memory (adrs) range. Record length must be same as memory range or an error will occur.
adrsl.adrs2W	*800.9FFW	Writes onto cassette data from speci- fied memory (adrs) range.
Display		
Ι	*I	Set inverse video mode. (Black characters on white background)
Μ	*N	Set normal video mode. (White characters on black background)
<u>Dis-assembler</u>		
adrsL	*C800L	Decodes 20 instructions starting at memory (adrs) into 6502 assembly nmenonic code.
L	*L	Decodes next 2Ø instructions starting at current memory address.
Mini-assembler		
(Turn-on)	*F666G	Turns-on mini-assembler. Prompt character is now a "!" (exclamation point).
\$(monitor command)	!\$C8ØØL	Executes any monitor command from mini- assembler then returns control to mini- assembler. Note that many monitor commands change current memory address reference so that it is good practice to retype desired address reference upon return to mini-assembler.
adrs:(65Ø2 MNEMONIC instruction)	!CØ10:STA 23FF	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	Example	Description
(space) (65Ø2 mnemonic instruction)	! STA Ø1FF	Assembles instruction into next available memory location. (Note space between "f" and instruction)
(TURN-OFF)	! (Reset Button)	Exits mini-assembler and returns to system monitor.

#### Monitor Program Execution and Debuging

adrsG	*300G	Runs machine level program starting at memory (adrs).
adrs⊤	*800T	Traces a program starting at memory location (adrs) and continues trace until hitting a breakpoint. Break occurs on instruction ØØ (BRK), and returns control to system monitor. Opens 6502 status registers (see note 1)
asrdS	*CØ5ØS	Single steps through program beginning at memory location (adrs). Type a letter S for each additional step that you want displayed. Opens 6502 status registers (see Note 1).
(Control E)	*EC	Displays 6502 status registers and opens them for modification (see Note 1)
(Control Y)	*ү <sup>С</sup>	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 = ØØ P = 32 S = F2 \*: FF Changes A register only \*: FF ØØ 33 Changes A, X, and Y registers

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

Hexidecimal Ari	thmetic	
datal+data2	*78+34	Performs hexidecimal sum of datal plus data2.
datal-data2	*AE-34	Performs hexidecimal difference of datal minus data2.

Comr	nand Format	Example	Description
Set	Input/Output	Ports	
(X)	(Control P)	*5P <sup>C</sup>	Sets printer output to I/O slot
(X)	(Control K)	*2K <sup>C</sup>	number (X). (see Note 2 belo Sets keyboard input to I/O s number (X). (see Note 2 belo

Note 2:

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

Multiple Commands

*100L 400G AFFT	Multiple monitor commands may be given on same line if separated by a "space".
*LLLL	Single letter commands may be repeated without spaces.

#### SPECIAL CONTROL AND EDITING CHARACTERS

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

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

# SPECIAL CONTROL AND EDITING CHARACTERS

### (continued)

CHARACTER DESCRIPTION	01	ACIION
-----------------------	----	--------

- A<sub>F</sub> Move cursor to right
- B<sub>E</sub> Move cursor to left
- C<sub>F</sub> Move cursor down
- D<sub>F</sub> Move cursor up
- E<sub>F</sub> Clear text from cursor to end of line
- F<sub>E</sub> Clear text from cursor to end of page

<u>Hex</u>	<u>BASIC Example</u>	Description
Display	Mode Controls	
C05Ø C051 C052 C053 C054	10 POKE -16304,0 20 POKE -16303,0 30 POKE -16302,0 40 POKE -16301,0 50 POKE -16300,0	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)
C055 C056 C057	70 POKE -16293,0 70 POKE -16298,0 80 POKE -16297,0	Clear HIRES graphics mode Set HIRES graphics mode
TEXT Mo	de Controls	
0020	90 POKE 32,L1	Set left side of scrolling window to location specified by Ll in range of Ø to 39.
0021	100 POKE 33,W1	Set window width to amount specified by Wl. Ll+Wl<4Ø. Wl>Ø
ØØ22	110 POKE 34,T1	Set window top to line specified by Tl in range of Ø to 23
ØØ23	120 POKE 35,B1	Set window bottom to line specified by Bl in the range of Ø to 23. B1>T1
0024	130 CH=PEEK(36) 140 POKE 36,CH 150 TAB(CH+1)	Read/set cusor horizontal position in the range of Ø to 39. If using TAB, you must add "1" to cusor position read value; Ex. 14Ø and 15Ø perform identical function.
ØØ25	160 CV=PEEK(37) 170 POKE 37,CV 180 VTAB(CV+1)	Similar to above. Read/set cusor vertical position in the range Ø to 23.
ØØ32	190 POKE 50,127 200 POKE 50,255	Set inverse flag if 127 (Ex. 190) Set normal flag if 255(Ex. 200)
FC58	210 CALL -936	(@ <sub>E</sub> ) Home cusor, clear screen
FC42	22Ø CALL -958	(F <sub>E</sub> ) Clear from cusor to end of page

<u>Hex</u>	BASIC Example	Description
FC9C	23Ø CALL -868	(E <sub>E</sub> ) Clear from cusor to end of line
FC66	24Ø CALL -922	(J <sup>C</sup> ) Line feed
FC7Ø	250 CALL -912	Scroll up text one line

# <u>Miscellaneous</u>

CØ3Ø	36Ø X=PEEK(-16336) 365 POKE -16336,Ø	Toggle speaker
CØØØ	37Ø X=PEEK(-16384	Read keyboard; if X>127 then key was pressed.
CØ1Ø	380 POKE -16368,0	Clear keyboard strobe – always after reading keyboard.
CØ61	39Ø X=PEEK(16287)	Read PDL(Ø) push button switch. If X>127 then switch is "on".
CØ62	400 X=PEEK(-16286)	Read PDL(1) push button switch.
CØ63	410 X=PEEK(-16285	Read PDL(2) push button switch.
CØ58	420 POKE -16296,0	Clear Game I/O ANØ output
CØ59	430 POKE -16295,0	Set Game I/O ANØ output
CØ5A	440 POKE -16294,0	Clear Game I/O ANl output
CØ5B	450 POKE -16293,0	Set Game I/O AN1 output
CØ5C	460 POKE -16292,0	Clear Game I/O AN2 output
CØ5D	470 POKE -16291,0	Set Game I/O AN2 output
CØ5E	480 POKE -16290,0	Clear Game I/O AN3 output
CØ5F	490 POKE -16289,0	Set Game I/O AN3 output

******	*****	*****	* * * * * *	* * *			
*				*			
*	APPLE	II		*			
* SY	STEM MO	ONITO	R	*			
*				*			
* COP	YRIGHT	1977	BY	*			
* APPL	E COMPU	JTER,	INC.	*			
*				*			
* ALL	RIGHTS	RESE	RVED	*			
*				*			
*	S. WOZI	IIAK		*			
*	A. BAU	JM		*			
*				*			
******	*****	*****	* * * * * *	* * *			
	TITLE			"APPLE	II	SYSTEM	MONITOR"
LOCO	EPZ	\$00					
LOC1	EPZ	\$01					
WNDLFT	EPZ	\$20					
WNDWDTH	EPZ	\$21					
WNDTOP	EPZ	\$22					
WNDBTM	EPZ	\$23					
СН	EPZ	\$24					
CV	EPZ	\$25					
GBASL	EPZ	\$26					
GBASH	EPZ	\$27					
BASL	EPZ	\$28					
BASH	EPZ	\$29					
BAS2L	EPZ	\$2A					
BAS2H	EPZ	\$2B					
H2	EPZ	\$2C					
LMNEM	EPZ	\$2C					
RTNL	EPZ	\$2C					
V2	EPZ	\$2D					
RMNEM	EPZ	Ş∠D					
RTNH	EPZ	ŞZD ¢2₽					
MASK	EPZ	ŞZE ÇQT					
CHKSUM	EPZ	\$2E \$2E					
FORMAT	EPZ	Ş∠E ¢2⊡					
LASTIN	EPZ	92F 625					
LENGTH	EP2 ED7	92F 62F					
SIGN	EPZ	92F 620					
MODE	EF4 ED7	\$30 \$31					
TNVETC	EF4 ED7	630 704					
	EF 2 E D 7	\$32 \$32					
VGAV	EF2 FD7	\$37					
VGAVI	FD7	\$35					
CSWL	EF2 FD7	\$32					
CSWH	EPZ	\$37					
KCMI	FD7	¢38					
KSWH	EPZ	\$39					
PCT.	EPZ	\$32					
РСН	EPZ	\$3B					
XOT	EP7	\$3C					
Alt	EPZ	\$30					
AlH	EPZ	\$3D					
A2L	EP7	\$3E					
A2H	EP7	\$3F					
A3L	EPZ	\$40					
АЗН	EPZ	\$41					
A4L	EPZ	\$42					
A4H	EPZ	\$43					
A5L	EPZ	\$44					
A5H	EPZ	\$45					

				ACC	EOU	\$45	
				XREG	EOU	\$46	
				YREG	EQU	\$47	
				STATUS	EQU	\$48	
				SPNT	EQU	\$49	
				RNDL	EQU	\$4E	
				RNDH	EQU	\$4F	
				ACL	EQU	\$50 ¢51	
				ACH	EQU	\$5⊥ ¢52	
				XTNDL	EQU	\$53	
				AUXL	EOU	\$54	
				AUXH	EQU	\$55	
				PICK	EQU	\$95	
				IN	EQU	\$0200	
				USRADR	EQU	\$03F8	
				TROLOC	EQU	\$03FB \$03FF	
				TOADR	EQU	\$C000	
				KBD	EQU	\$C000	
				KBDSTRB	EQU	\$C010	
				TAPEOUT	EQU	\$C020	
				SPKR	EQU	\$C030	
				TATCLR	EQU	\$C050 \$C051	
				MIXCLR	EQU	\$C052	
				MIXSET	EOU	\$C053	
				LOWSCR	EQU	\$C054	
				HISCR	EQU	\$C055	
				LORES	EQU	\$C056	
				HIRES	EQU	\$C057	
				PADDL0	EQU	\$C064	
				PTRIG	EQU	\$C070	
				BASIC	EQU	\$E000	
				BASIC2	EQU	\$E003	
<b>F</b> 000.	4 7				ORG	ŞF800	ROM START ADDRESS
F801.	4A 08			PLOI	DHD	A	SAVE LSB IN CARRY
F802:	20	47	F8		JSR	GBASCALC	CALC BASE ADR IN GBASL,H
F805:	28				PLP		RESTORE LSB FROM CARRY
F806:	A9	0 F			LDA	#\$0F	MASK \$0F IF EVEN
F808:	90	02			BCC	RTMASK	NAGE (EQ IE ODD
F80A:	69 85	E0 2 F		ртмаск	ADC STTA	#ŞEU Mask	MASK \$FO IF ODD
F80E:	Bl	26		PLOT1	LDA	(GBASL),Y	DATA
F810:	45	30			EOR	COLOR	EOR COLOR
F812:	25	2E			AND	MASK	AND MASK
F814:	51	26			EOR	(GBASL),Y	XOR DATA
F816:	91 60	26			STA	(GBASL),Y	TO DATA
F819:	20	0.0	F8	HLINE	JSR	PLOT	PLOT SOUARE
F81C:	C4	2C		HLINEL	CPY	H2	DONE?
F81E:	в0	11			BCS	RTSl	YES, RETURN
F820:	C8	~ -			INY		NO, INCR INDEX (X-COORD)
F821:	20	0E	F8		JSR	PLOTI	PLOT NEXT SQUARE
F824:	90 69	r 6 01		VI.INEZ	ADC	HLINEI #\$01	NEXT V-COORD
F828:	48	01		VLINE	PHA	ПООТ	SAVE ON STACK
F829:	20	00	F8		JSR	PLOT	PLOT SQUARE
F82C:	68				PLA		
F82D:	C5	2D			CMP	V2	DONE?
F82F:	90 60	гэ		RTC1	BCC BTS	VLINEZ	NO, LOOP.
F832:	A0	2F		CLRSCR	LDY	#\$2F	MAX Y, FULL SCRN CLR
F834:	D0	02			BNE	CLRSC2	ALWAYS TAKEN
F836:	A0	27		CLRTOP	LDY	#\$27	MAX Y, TOP SCREEN CLR
F838:	84	2D		CLRSC2	STY	V2	STORE AS BOTTOM COORD
F837.	<b>A</b> O	27		*	FOR	VLINE CALLS	
F83C:	A9	00		CLRSC3	LDA	#\$27 #\$00	TOP COORD FOR VLINE CALLS
F83E:	85	30			STA	COLOR	CLEAR COLOR (BLACK)
F840:	20	28	F8		JSR	VLINE	DRAW VLINE
F843:	88				DEY	at D a a a	NEXT LEFTMOST X-COORD
F844:	Ε0 Τ0	т. е			вњс рђГ	CLKSC3	LOOP UNTIL DONE.
F847:	48			GBASCALC	PHA		FOR INPUT 000DEFGH
F848:	4A				LSR	A	
F849:	29	03			AND	#\$03	
F84B:	09	04			ORA	#\$04	GENERATE GBASH=000001FG
F84D:	85 60	27			STA DLA	GBASH	AND CRASL-HDEDEOOO
F850:	29	18			AND	#\$18	THE GRAD-HDEDEOOO
F852:	90	02			BCC	GBCALC	
F854:	69	7F			ADC	#\$7F	
F856:	85	26		GBCALC	STA	GBASL	

F858:	0A				ASL	A	
F859:	0A 05	26			ASL	A CBAST	
F85C:	85	26			STA	GBASL	
F85E:	60	20			RTS	021102	
F85F:	A5	30		NXTCOL	LDA	COLOR	INCREMENT COLOR BY 3
F861:	18				CLC		
F862:	69	03		00000	ADC	#\$03	
F864:	29	30		SETCOL	AND	#ŞUF COLOP	SETS COLOR=1/*A MOD 16
F868:	0 A	50			ASL	A	BOTH HALF BYTES OF COLOR EQUAL
F869:	0A				ASL	A	
F86A:	0A				ASL	А	
F86B:	0A				ASL	A	
F86C:	05	30			ORA	COLOR	
F86E:	85	30			STA	COLOR	
F8/U:	60 47			CODN	RTS	7	DEAD CODEEN V COODD /2
F872:	4A 08			SCRN	DHD	A	SAVE LSB (CARRY)
F873:	20	47	F8		JSR	GBASCALC	CALC BASE ADDRESS
F876:	Bl	26			LDA	(GBASL),Y	Y GET BYTE
F878:	28				PLP	. ,.	RESTORE LSB FROM CARRY
F879:	90	04		SCRN2	BCC	RTMSKZ	IF EVEN, USE LO H
F87B:	4A				LSR	A	
F87C:	4A				LSR	A	ANTER NIGH WALL DURE DOWN
F87D: F87F•	4A 1 A				LSR	A	SHIFT HIGH HALF BYTE DOWN
F87F:	29	0 F		RTMSK7	AND	4 #\$0F	MASK 4BTTS
F881:	60	01		RIIIDIG	RTS	1 4 0 1	
F882:	A6	3A		INSDS1	LDX	PCL	PRINT PCL, H
F884:	A4	3B			LDY	PCH	
F886:	20	96	FD		JSR	PRYX2	
F889:	20	48	F9		JSR	PRBLNK	FOLLOWED BY A BLANK
F88C:	AL	3A		TNADAO	LDA	(PCL,X)	GET OP CODE
FOOE:	А8 4 л			INSDSZ	TAI	7	
F890:	90	09			BCC	A TEVEN	EVEN/ODD TEST
F892:	6A	0,5			ROR	10,010	BIT 1 TEST
F893:	в0	10			BCS	ERR	XXXXXX11 INVALID OP
F895:	C9	A2			CMP	#\$A2	
F897:	F0	0C			BEQ	ERR	OPCODE \$89 INVALID
F899:	29	87			AND	#\$87	MASK BITS
F89B:	4A			IEVEN	LSR	A	LSB INTO CARRY FOR L/R TEST
F89C:	AA BD	62	FО			<u> ምለጥ</u> ገ	<b>ΔΕΨ ΕΟΡΜΔΨ ΙΝΟΕΥ ΒΥΨΕ</b>
F8A0:	20	79	F8		JSR	SCRN2	R/L HEYTE ON CARRY
F8A3:	D0	04	10		BNE	GETFMT	
F8A5:	A0	80		ERR	LDY	#\$80	SUBSTITUTE \$80 FOR INVALID OPS
F8A7:	A9	00			LDA	#\$00	SET PRINT FORMAT INDEX TO 0
F870.	7 7			GETEMT	mλv		
LORD.	AA	-	-	0211111	IAA	-	
F8AA:	BD	A6	F9	0211111	LDA	FMT2,X	INDEX INTO PRINT FORMAT TABLE
F8AA: F8AD:	BD 85	A6 2E	F9		LDA STA	FMT2,X FORMAT	INDEX INTO PRINT FORMAT TABLE SAVE FOR ADR FIELD FORMATTING
F8AA: F8AD: F8AF:	BD 85 29	A6 2E 03	F9	*	LDA STA AND	FMT2,X FORMAT #\$03 P=1 BYTE	INDEX INTO PRINT FORMAT TABLE SAVE FOR ADR FIELD FORMATTING MASK FOR 2BIT LENGTH
F8AA: F8AD: F8AF: F8B1:	BD 85 29 85	A6 2E 03 2F	F9	*	LDA STA AND (STA	FMT2,X FORMAT #\$03 P=1 BYTE, LENGTH	INDEX INTO PRINT FORMAT TABLE SAVE FOR ADR FIELD FORMATTING MASK FOR 2BIT LENGTH , 1=2 BYTE, 2=3 BYTE)
F8AA: F8AD: F8AF: F8B1: F8B3:	AA BD 85 29 85 98	A6 2E 03 2F	F9	*	LDA STA AND (STA TYA	FMT2,X FORMAT #\$03 P=1 BYTE, LENGTH	INDEX INTO PRINT FORMAT TABLE SAVE FOR ADR FIELD FORMATTING MASK FOR 2BIT LENGTH , 1=2 BYTE, 2=3 BYTE) OPCODE
F8AA: F8AD: F8AF: F8B1: F8B3: F8B4:	BD 85 29 85 98 29	A6 2E 03 2F 8F	F9	*	LDA STA AND (STA TYA AND	FMT2,X FORMAT #\$03 P=1 BYTE, LENGTH #\$8F	INDEX INTO PRINT FORMAT TABLE SAVE FOR ADR FIELD FORMATTING MASK FOR 2BIT LENGTH , 1=2 BYTE, 2=3 BYTE) OPCODE MASK FOR 1XXX1010 TEST
F8AA: F8AD: F8AF: F8B1: F8B3: F8B4: F8B6:	AA BD 85 29 85 98 29 AA	A6 2E 03 2F 8F	F9	*	LDA STA AND (STA TYA AND TAX	FMT2,X FORMAT #\$03 P=1 BYTE, LENGTH #\$8F	INDEX INTO PRINT FORMAT TABLE SAVE FOR ADR FIELD FORMATTING MASK FOR 2BIT LENGTH , 1=2 BYTE, 2=3 BYTE) OPCODE MASK FOR 1XXX1010 TEST SAVE IT
F8AA: F8AD: F8AF: F8B1: F8B3: F8B4: F8B6: F8B6: F8B7:	AA BD 85 29 85 98 29 AA 98 29	A6 2E 03 2F 8F	F9	*	LDA STA AND STA TYA AND TAX TYA LDY	FMT2,X FORMAT #\$03 P=1 BYTE, LENGTH #\$8F	INDEX INTO PRINT FORMAT TABLE SAVE FOR ADR FIELD FORMATTING MASK FOR 2BIT LENGTH , 1=2 BYTE, 2=3 BYTE) OPCODE MASK FOR 1XXX1010 TEST SAVE IT OPCODE TO A AGAIN
F8AA: F8AD: F8AF: F8B1: F8B3: F8B4: F8B6: F8B6: F8B7: F8B8: F8B8:	AA BD 85 29 85 98 29 AA 98 A0 E0	A6 2E 03 2F 8F 03	F9	*	LDA STA AND ( STA TYA AND TAX TYA LDY CBY	<pre>FMT2,X FORMAT #\$03 P=1 BYTE, LENGTH #\$8F #\$03 #\$903</pre>	INDEX INTO PRINT FORMAT TABLE SAVE FOR ADR FIELD FORMATTING MASK FOR 2BIT LENGTH , 1=2 BYTE, 2=3 BYTE) OPCODE MASK FOR 1XXX1010 TEST SAVE IT OPCODE TO A AGAIN
F8AA: F8AD: F8AF: F8B1: F8B3: F8B3: F8B3: F8B4: F8B6: F8B6: F8B8: F8BA: F8BA:	AA BD 85 29 85 98 29 AA 98 A0 E0 F0	A6 2E 03 2F 8F 03 8A 0B	F9	*	LDA STA AND ( STA TYA AND TAX TYA LDY CPX BEO	<pre>FMT2,X FORMAT #\$03 P=1 BYTE, LENGTH #\$8F #\$03 #\$8A MNNDX3</pre>	INDEX INTO PRINT FORMAT TABLE SAVE FOR ADR FIELD FORMATTING MASK FOR 2BIT LENGTH , 1=2 BYTE, 2=3 BYTE) OPCODE MASK FOR 1XXX1010 TEST SAVE IT OPCODE TO A AGAIN
F8AA: F8AD: F8AF: F8B1: F8B3: F8B4: F8B6: F8B6: F8B7: F8B8: F8BA: F8BA: F8BE:	AA BD 85 29 85 98 29 AA 98 A0 E0 F0 4A	A6 2E 03 2F 8F 03 8A 0B	F9	*	LDA STA AND STA TYA AND TAX TYA LDY CPX BEQ LSR	<pre>FMT2,X FORMAT #\$03 P=1 BYTE, LENGTH #\$8F #\$03 #\$8A MNNDX3 A</pre>	INDEX INTO PRINT FORMAT TABLE SAVE FOR ADR FIELD FORMATTING MASK FOR 2BIT LENGTH , 1=2 BYTE, 2=3 BYTE) OPCODE MASK FOR 1XXX1010 TEST SAVE IT OPCODE TO A AGAIN
F8AA: F8AA: F8AF: F8AF: F8B1: F8B3: F8B4: F8B4: F8B6: F8B8: F8B8: F8B5: F8B5:	AA BD 85 29 85 98 29 AA 98 A0 E0 F0 4A 90	A6 2E 03 2F 8F 03 8A 0B 08	F9	* MNNDX1	LDA STA AND ( STA TYA AND TAX TYA LDY CPX BEQ LSR BCC	<pre>FMT2,X FORMAT #\$03 P=1 BYTE, LENGTH #\$8F #\$03 #\$8A MNNDX3 A MNNDX3</pre>	INDEX INTO PRINT FORMAT TABLE SAVE FOR ADR FIELD FORMATTING MASK FOR 2BIT LENGTH , 1=2 BYTE, 2=3 BYTE) OPCODE MASK FOR 1XXX1010 TEST SAVE IT OPCODE TO A AGAIN FORM INDEX INTO MNEMONIC TABLE
F8AA: F8AA: F8AA: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B5: F8B5: F8B5: F8B5: F8B5: F8B5: F8B5: F8B5:	AA BD 85 29 85 98 29 AA 98 A0 E0 F0 4A 90 4A	A6 2E 03 2F 8F 03 8A 0B 08	F9	* MNNDX1	LDA STA AND STA TYA AND TAX TYA LDY CPX BEQ LSR BCC LSR	FMT2,X FORMAT #\$03 P=1 BYTE, LENGTH #\$8F #\$03 #\$8A MNNDX3 A MNNDX3 A	INDEX INTO PRINT FORMAT TABLE SAVE FOR ADR FIELD FORMATTING MASK FOR 2BIT LENGTH , 1=2 BYTE, 2=3 BYTE) OPCODE MASK FOR 1XXX1010 TEST SAVE IT OPCODE TO A AGAIN FORM INDEX INTO MNEMONIC TABLE
F8AA: F8AA: F8AA: F8AF: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B5: F8B5: F8B5: F8B5: F8B5: F8C2:	<ul> <li>AA</li> <li>BD</li> <li>85</li> <li>29</li> <li>85</li> <li>98</li> <li>29</li> <li>AA</li> <li>98</li> <li>A0</li> <li>E0</li> <li>F0</li> <li>4A</li> <li>90</li> <li>4A</li> <li>4A</li> </ul>	A6 2E 03 2F 8F 03 8A 0B 08	F9	* MNNDX1 MNNDX2	LDA STA AND STA TYA AND TAX TYA LDY CPX BEQ LSR LSR LSR	FMT2,X FORMAT #\$03 P=1 BYTE, LENGTH #\$8F #\$03 #\$8A MNNDX3 A MNNDX3 A A	INDEX INTO PRINT FORMAT TABLE SAVE FOR ADR FIELD FORMATTING MASK FOR 2BIT LENGTH , 1=2 BYTE, 2=3 BYTE) OPCODE MASK FOR 1XXX1010 TEST SAVE IT OPCODE TO A AGAIN FORM INDEX INTO MNEMONIC TABLE 1) 1XXX1010=>00101XXX
F8AA: F8AA: F8AA: F8AF: F8BA: F8BA: F8BA: F8BA: F8BA: F8BA: F8BA: F8BA: F8BF: F8C1: F8C2: F8C3:	BD 85 29 85 29 85 29 85 29 85 29 82 98 80 E0 F0 4A 90 4A 4A 90	A6 2E 03 2F 8F 03 8A 0B 08	F9	* MNNDX1 MNNDX2	LDA STA AND STA TYA AND TAX TYA LDY CPX BEQ LSR BCC LSR LSR ORA	FMT2, X FORMAT #\$03 P=1 BYTE, LENGTH #\$8F #\$03 #\$8A MNNDX3 A MNNDX3 A A #\$20	INDEX INTO PRINT FORMAT TABLE SAVE FOR ADR FIELD FORMATTING MASK FOR 2BIT LENGTH , 1=2 BYTE, 2=3 BYTE) OPCODE MASK FOR 1XXX1010 TEST SAVE IT OPCODE TO A AGAIN FORM INDEX INTO MNEMONIC TABLE 1) 1XXX1010=>00101XXX 2) XXXYYY01=>00111XXX
F8AA: F8AA: F8AA: F8AF: F8BA: F8BA: F8BA: F8BA: F8BA: F8BA: F8BA: F8BA: F8BA: F8BF: F8C1: F8C2: F8C3: F8C5:	AA BD 85 29 85 29 AA 98 A0 E0 F0 4A 90 4A 4A 09 88	A6 2E 03 2F 8F 03 8A 0B 08 08	F9	* MNNDX1 MNNDX2	LDA STA AND STA AND TYA AND TAX TYA LDY CPX BEQ LSR BCC LSR LSR ORA DEY DEY DYZ	<pre>FMT2, X FORMAT #\$03 P=1 BYTE, LENGTH #\$8F #\$03 #\$8A MNNDX3 A MNNDX3 A A #\$20 WWWDW2</pre>	INDEX INTO PRINT FORMAT TABLE SAVE FOR ADR FIELD FORMATTING MASK FOR 2BIT LENGTH , 1=2 BYTE, 2=3 BYTE) OPCODE MASK FOR 1XXX1010 TEST SAVE IT OPCODE TO A AGAIN FORM INDEX INTO MNEMONIC TABLE 1) 1XXX1010=>00101XXX 2) XXXYYY10=>00111XXX 3) XXXYYY10=>00110XXX
F8AA: F8AA: F8AA: F8AA: F8BA: F8BA: F8BA: F8BA: F8BA: F8BA: F8BA: F8BA: F8BA: F8BA: F8BA: F8BA: F8BA: F8BA: F8C2: F8C3: F8C5: F8C6: F8C6:	AA BD 85 29 85 98 29 AA 98 AO F0 4A 90 88 00 C 8	A6 2E 03 2F 8F 03 8A 0B 08 08 20 FA	F9	* MNNDX1 MNNDX2	LDA STA AND ( STA TYA AND TAX TYA LDY CPX BEQ LSR BCC LSR LSR ORA DEY BNE TNY	FMT2, X FORMAT #\$03 P=1 BYTE, LENGTH #\$8F #\$03 #\$8A MNNDX3 A MNNDX3 A A #\$20 MNNDX2	<pre>INDEX INTO PRINT FORMAT TABLE SAVE FOR ADR FIELD FORMATTING MASK FOR 2BIT LENGTH , 1=2 BYTE, 2=3 BYTE) OPCODE MASK FOR 1XXX1010 TEST SAVE IT OPCODE TO A AGAIN FORM INDEX INTO MNEMONIC TABLE 1) 1XXX1010=&gt;00101XXX 2) XXXYY10=&gt;00111XXX 3) XXXYY10=&gt;00110XXX 4) XXXYY10=&gt;00110XXX 5) XXXYY00=&gt;0000XXX</pre>
F8AA: F8AD: F8AF: F8B3: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B5: F8B5: F8B5: F8C2: F8C2: F8C3: F8C5: F8C6: F8C9:	AA BD 85 29 85 98 29 AA 98 A0 F0 4A 90 88 00 88 88	A6 2E 03 2F 8F 03 8A 0B 08 20 FA	F9	* MNNDX1 MNNDX2	LDA STA AND STA AND TYA AND TAX TYA LDY CPX BEQ LSR BCC LSR DEY BNE INY DEY	<pre>FMT2, X FORMAT #\$03 P=1 BYTE, LENGTH #\$8F #\$03 #\$8A MNNDX3 A MNNDX3 A #\$20 MNNDX2</pre>	INDEX INTO PRINT FORMAT TABLE SAVE FOR ADR FIELD FORMATTING MASK FOR 2BIT LENGTH , 1=2 BYTE, 2=3 BYTE) OPCODE MASK FOR 1XXX1010 TEST SAVE IT OPCODE TO A AGAIN FORM INDEX INTO MNEMONIC TABLE 1) 1XXX1010=>00101XXX 2) XXXYYY01=>00111XXX 3) XXXYYY10=>00110XXX 4) XXXYY100=>00100XXX 5) XXXXX000=>000XXXXX
<pre>F8AA: F8AD: F8AF: F8B3: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B5: F8B4: F8B5: F8C1: F8C2: F8C3: F8C4: F8C4: F8C4: F8C4: F8C4: F8C4: F8C4: F8C4: F8C4: F8C4: F8C4: F8C4: F8C4: F8C4: F8C4: F8C4: F8C4: F8B4: F8C4: F</pre>	<ul> <li>AA</li> <li>BD</li> <li>85</li> <li>29</li> <li>85</li> <li>29</li> <li>AA</li> <li>98</li> <li>A0</li> <li>E0</li> <li>F0</li> <li>4A</li> <li>90</li> <li>4A</li> <li>90</li> <li>4A</li> <li>90</li> <li>4A</li> <li>09</li> <li>88</li> <li>D0</li> <li>C8</li> <li>80</li> </ul>	A6 2E 03 2F 8F 03 8A 0B 08 20 FA F2	F9	* MNNDX1 MNNDX2 MNNDX3	LDA STA AND STA AND TYA AND TAX TYA LDY CPX BEQ LSR BCC LSR DEY BNE INY DEY BNE	<pre>FMT2,X FORMAT #\$03 P=1 BYTE, LENGTH #\$8F #\$03 #\$8A MNNDX3 A MNNDX3 A #\$20 MNNDX2 MNNDX1</pre>	<pre>INDEX INTO PRINT FORMAT TABLE SAVE FOR ADR FIELD FORMATTING MASK FOR 2BIT LENGTH , 1=2 BYTE, 2=3 BYTE) OPCODE MASK FOR 1XXX1010 TEST SAVE IT OPCODE TO A AGAIN FORM INDEX INTO MNEMONIC TABLE 1) 1XXX1010=&gt;00101XXX 2) XXXYYY01=&gt;00111XXX 3) XXXYYY10=&gt;00110XXX 4) XXXYY100=&gt;00100XXX 5) XXXXX000=&gt;000XXXXX</pre>
F8AA: F8AD: F8AF: F8B1: F8B4: F8B4: F8B6: F8B7: F8B7: F8B8: F8B7: F8B7: F8B7: F8B7: F8B7: F8C1: F8C2: F8C3: F8C5: F8C6: F8C8: F8C4: F8C4: F8C4: F8C4: F8C4: F8C4: F8C4: F8C4: F8C4: F8C4: F8C4: F8C4: F8B4: F8C4:	<ul> <li>BD</li> <li>85</li> <li>98</li> <li>29</li> <li>85</li> <li>98</li> <li>29</li> <li>AA</li> <li>98</li> <li>A0</li> <li>E0</li> <li>F0</li> <li>4A</li> <li>90</li> <li>4A</li> <li>90</li> <li>4A</li> <li>90</li> <li>4A</li> <li>09</li> <li>88</li> <li>D0</li> <li>60</li> </ul>	A6 2E 03 2F 8F 03 8A 0B 08 20 FA F2	F9	* MNNDX1 MNNDX2 MNNDX3	LDA STA AND STA AND TYA AND TAX TYA LDY CPX BEQ LSR BCC LSR DEY BNE INY DEY BNE RTS	<pre>FMT2,X FORMAT #\$03 P=1 BYTE, LENGTH #\$8F #\$03 #\$8A MNNDX3 A MNNDX3 A #\$20 MNNDX2 MNNDX1</pre>	<pre>INDEX INTO PRINT FORMAT TABLE SAVE FOR ADR FIELD FORMATTING MASK FOR 2BIT LENGTH , 1=2 BYTE, 2=3 BYTE) OPCODE MASK FOR 1XXX1010 TEST SAVE IT OPCODE TO A AGAIN FORM INDEX INTO MNEMONIC TABLE 1) 1XXX1010=&gt;00101XXX 2) XXXYY10=&gt;00111XXX 3) XXXYY10=&gt;00110XXX 4) XXXYY10=&gt;00100XXX 5) XXXXX000=&gt;0000XXXXX</pre>
F8AA: F8AD: F8AF: F8AF: F8BA: F8BA: F8BA: F8BA: F8BA: F8BA: F8BA: F8BA: F8BA: F8BA: F8BA: F8BA: F8BA: F8BA: F8BA: F8CA: F8BA: F8CA:	BD 85 29 85 29 85 29 85 29 85 29 85 29 85 29 85 29 85 29 85 29 85 29 85 29 85 29 85 29 85 29 85 29 85 85 29 85 85 29 85 85 29 85 85 29 85 85 29 85 85 29 85 85 85 29 85 85 85 85 85 85 85 85 85 85 85 85 85	A6 2E 03 2F 8F 03 8A 0B 08 20 FA F2 FF	F9 FF	* MNNDX1 MNNDX2 MNNDX3	LDA STA AND STA AND TYA AND TAX TYA LDY CPX BEQ LSR BECC LSR BECC LSR BECC LSR BECC LSR BECC LSR BECC LSR BECC LSR BECC STA STA AND TAX TYA AND TAX TYA AND TAX TYA AND TAX TYA AND TAX TYA AND TAX TYA AND TAX TYA AND TAX TYA AND TAX TYA AND TAX TYA AND TAX TYA AND TAX TYA AND TAX TYA AND TAX TYA AND TAX TYA AND TAX TYA AND TAX TYA AND TAX TYA AND TAX TYA CPX BEC DISR BEC DISR DISR DISR DISR DISR DISR DISR DISR	<pre>FMT2,X FORMAT #\$03 P=1 BYTE, LENGTH #\$8F #\$03 #\$8A MNNDX3 A MNNDX3 A #\$20 MNNDX2 MNNDX1 \$FF,\$FF,\$</pre>	<pre>INDEX INTO PRINT FORMAT TABLE SAVE FOR ADR FIELD FORMATTING MASK FOR 2BIT LENGTH , 1=2 BYTE, 2=3 BYTE) OPCODE MASK FOR 1XXX1010 TEST SAVE IT OPCODE TO A AGAIN FORM INDEX INTO MNEMONIC TABLE 1) 1XXX1010=&gt;00101XXX 2) XXXYY10=&gt;00111XXX 3) XXXYY10=&gt;00110XXX 4) XXXYY100=&gt;00100XXX 5) XXXXX000=&gt;000XXXXX</pre>
F8AA:         F8AF:         F8AF:         F8B1:         F8B7:         F8B6:         F8B7:         F8B8:         F8B7:         F8B7:         F8B7:         F8B7:         F8B7:         F8B7:         F8B7:         F8C1:         F8C1:         F8C2:         F8C6:         F8C8:         F8C9:         F8C2:         F8C2:         F8C2:         F8C4:         F8C5:         F8C6:         F8C7:         F8C0:	BD 85 29 85 29 85 29 85 29 85 29 85 29 85 29 85 29 85 29 85 29 85 29 85 29 85 29 85 29 80 60 88 80 60 87 82 80 82 80 80 80 80 80 80 80 80 80 80 80 80 80	A6 2E 03 2F 8F 03 8A 0B 08 20 FA FA F2 FF	F9 FF F8	* MNNDX1 MNNDX2 MNNDX3 INSTDSP	LDA STA AND STA TYA AND TAX TYA LDY CPX BEQ LSR BEQ LSR BCC LSR BEQ LSR BCC LSR BEQ LSR BEQ LSR BEQ LSR BEQ LSR BEQ LSR BEQ LSR BNE STA AND TAX TYA BEQ LSR BCC LSR BNE JSR JSR JSR JSR JSR JSR JSR JSR JSR JSR	<pre>FMT2,X FORMAT #\$03 P=1 BYTE, LENGTH #\$8F #\$03 #\$8A MNNDX3 A A #\$20 MNNDX2 MNNDX1 \$FF,\$FF,\$ INSDS1</pre>	<pre>INDEX INTO PRINT FORMAT TABLE SAVE FOR ADR FIELD FORMATTING MASK FOR 2BIT LENGTH , 1=2 BYTE, 2=3 BYTE) OPCODE MASK FOR 1XXX1010 TEST SAVE IT OPCODE TO A AGAIN FORM INDEX INTO MNEMONIC TABLE 1) 1XXX1010=&gt;00101XXX 2) XXXYY10=&gt;00111XXX 3) XXXYY10=&gt;00110XXX 4) XXXYY100=&gt;00100XXX 5) XXXXX000=&gt;000XXXXX</pre>
F8AA:         F8AA:         F8AF:         F8BF:         F8B4:         F8B6:         F8B7:         F8B8:         F8C1:         F8C2:         F8C6:         F8C8:         F8C9:         F8C2:         F8C2:         F8C4:         F8C5:         F8C6:         F8C7:         F8C8:         F8C9:         F8C1:         F8C0:         F8D0:         F8D0:         F8D1:	AA BD 85 98 29 AA 29 AA 29 AA 29 AA 29 AA 29 AA 29 AA 29 AA 29 AA 29 AA 29 AA 29 AA 20 20 AA 20 20 AA 20 A 20 AA 20 AA 20 AA 20 AA 20 AA 20 AA 20 AA 20 AA 20 AA 20 AA 20 A 20 AA 20 AA 20 AA 20 AA 20 AA 20 AA 20 AA 20 AA 20 AA 20 AA 20 AA 20 AA 20 AA 20 A 2 A 2	A6 2E 03 2F 8F 03 8A 0B 08 08 20 FA F2 FF	F9 FF F8	* MNNDX1 MNNDX2 MNNDX3 INSTDSP	LDA STA AND STA AND TYA AND TAX TYA LDY CPX BEQ LSR BCC LSR BCC LSR BCC LSR BCC LSR BNE INY DEY BNE TNY DFB JSR PHA	<pre>FMT2,X FORMAT #\$03 P=1 BYTE, LENGTH #\$8F #\$03 #\$8A MNNDX3 A A #\$20 MNNDX2 MNNDX1 \$FF,\$FF,\$ INSDS1</pre>	INDEX INTO PRINT FORMAT TABLE SAVE FOR ADR FIELD FORMATTING MASK FOR 2BIT LENGTH , 1=2 BYTE, 2=3 BYTE) OPCODE MASK FOR 1XXX1010 TEST SAVE IT OPCODE TO A AGAIN FORM INDEX INTO MNEMONIC TABLE 1) 1XXX1010=>00101XXX 2) XXXYY10=>00111XXX 3) XXXYY10=>00110XXX 4) XXXYY10=>00110XXX 5) XXXXX000=>000XXXXX 5) XXXXX000=>000XXXXX
F8AA:         F8AA:         F8AF:         F8BF:         F8B4:         F8B6:         F8B7:         F8B8:         F8B8:         F8B8:         F8B6:         F8B7:         F8B8:         F8B7:         F8B8:         F8B7:         F820:         F8C6:         F8C7:         F8C8:         F8C9:         F8C4:         F8C0:         F8D0:         F8D3:         F8D4:         F8D4:	AA BD 85 98 29 AA 29 AA 29 AA 29 AA 29 AA 29 AA 29 AA 29 AA 29 AA 29 AA 29 AA 29 AA 29 AA 20 4A 20 4A 20 85 20 AA 20 4A 20 20 4A 20 20 4A 20 20 4A 20 20 4A 20 20 20 4A 20 20 20 4A 20 20 20 4A 20 20 20 20 20 20 20 20 20 20 20 20 20	A6 2E 03 2F 8F 03 8A 0B 08 08 20 FA F2 FF 82 3A	F9 FF F8	* MNNDX1 MNNDX2 MNNDX3 INSTDSP PRNTOP	LDA STA AND STA AND TYA AND TAX TYA LDY CPX BEQ LSR BCC LSR BCC LSR BCC LSR BNE INY DEY BNE INY DEY BNE INY DFB JSR PHA LDA	<pre>FMT2,X FORMAT #\$03 P=1 BYTE, LENGTH #\$8F #\$03 #\$8A MNNDX3 A A #\$20 MNNDX2 MNNDX1 \$FF,\$FF,\$ INSDS1 (PCL),Y</pre>	INDEX INTO PRINT FORMAT TABLE SAVE FOR ADR FIELD FORMATTING MASK FOR 2BIT LENGTH , 1=2 BYTE, 2=3 BYTE) OPCODE MASK FOR 1XXX1010 TEST SAVE IT OPCODE TO A AGAIN FORM INDEX INTO MNEMONIC TABLE 1) 1XXX1010=>00101XXX 2) XXXYY10=>00110XXX 4) XXXYY10=>00110XXX 4) XXXYY10=>00100XXX 5) XXXXX000=>000XXXXX
F8AA:         F8AA:         F8AA:         F8BA:         F8B4:         F8B4:         F8B4:         F8B4:         F8B4:         F8B5:         F8B6:         F8B6:         F8B7:         F8B6:         F8B7:         F8B6:         F8B7:         F8C1:         F8C2:         F8C3:         F8C4:         F8C5:         F8C6:         F8C0:         F8D0:         F8D4:         F8D4:         F8D4:	AA BD B5 29 85 85 98 29 85 98 20 80 80 80 80 60 60 60 60 60 88 00 60 67 52 98 80 80 80 80 80 80 80 80 80 80 80 80 80	A6 2E 03 2F 8F 03 8A 0B 08 08 20 FA F2 FF 82 3A 00	F9 FF F8 FD	* MNNDX1 MNNDX2 MNNDX3 INSTDSP PRNTOP	LDA STA AND STA AND TYA AND TAX TYA LDY CPX BEQ LSR BCC LSR BCC LSR BCC LSR BNE INY DEY BNE INY DEY BNE JSR PHA LDA JSR LDY	<pre>FMT2,X FORMAT #\$03 P=1 BYTE, LENGTH #\$8F #\$03 #\$8A MNNDX3 A A #\$20 MNNDX2 MNNDX2 MNNDX1 \$FF,\$FF,\$ INSDS1 (PCL),Y PRBYTE #\$01</pre>	INDEX INTO PRINT FORMAT TABLE SAVE FOR ADR FIELD FORMATTING MASK FOR 2BIT LENGTH , 1=2 BYTE, 2=3 BYTE) OPCODE MASK FOR 1XXX1010 TEST SAVE IT OPCODE TO A AGAIN FORM INDEX INTO MNEMONIC TABLE 1) 1XXX1010=>00101XXX 2) XXXYY10=>00110XXX 3) XXXYY10=>00110XXX 4) XXXYY10=>00100XXX 5) XXXXX000=>000XXXXX 5) XXXXX000=>000XXXXX
F8AA: F8AD: F8AF: F8B4: F8B4: F8B4: F8B4: F8B6: F8B7: F8B7: F8B7: F8B7: F8B7: F8B7: F8B7: F8B7: F8B7: F8C7:	AA BD BS 29 85 98 29 85 98 29 80 60 40 40 80 40 80 60 60 60 60 60 60 60 60 60 60 60 82 90 80 80 80 80 80 80 80 80 80 80 80 80 80	A6 2E 03 2F 8F 03 8A 0B 08 08 20 FA F2 F7 82 3A 04 04	F9 FF F8 FD F9	* MNNDX1 MNNDX2 MNNDX3 INSTDSP PRNTOP PRNTBL	LDA STA AND STA TYA AND TAX TYA LDY CPX BEQ LSR BCC LSR LSR ORA DEY BNE INY DEY BNE INY DEY BNE INY DEY BNE LDA SR PHA LDA JSR	<pre>FMT2,X FORMAT #\$03 P=1 BYTE, LENGTH #\$8F #\$03 #\$8A MNNDX3 A A #\$20 MNNDX2 MNNDX2 MNNDX1 \$FF,\$FF,\$ INSDS1 (PCL),Y PRBYTE #\$01 PRB12</pre>	INDEX INTO PRINT FORMAT TABLE SAVE FOR ADR FIELD FORMATTING MASK FOR 2BIT LENGTH , 1=2 BYTE, 2=3 BYTE) OPCODE MASK FOR 1XXX1010 TEST SAVE IT OPCODE TO A AGAIN FORM INDEX INTO MNEMONIC TABLE 1) 1XXX1010=>00101XXX 2) XXXYY10=>00110XXX 3) XXXYY10=>00110XXX 4) XXXYY10=>00100XXX 5) XXXXX000=>000XXXXX SFF GEN FMT, LEN BYTES SAVE MNEMONIC TABLE INDEX PRINT 2 BLANKS
F8AA: F8AD: F8AF: F8B4: F8B4: F8B4: F8B6: F8B7: F8B7: F8B7: F8B7: F8B7: F8B7: F8B7: F8B7: F8B7: F8B7: F8B7: F8C7:	AA BB5 29 85 929 A0 E0 40 40 80 60 F0 40 80 60 F0 48 D0 60 F7 48 B20 24 82 20 20 82 80 80 80 80 80 80 80 80 80 80 80 80 80	A6 2E 03 2F 8F 03 8A 0B 08 08 20 FA F2 FF 23A 01 4A 2F	F9 FF F8 FD F9	* MNNDX1 MNNDX2 MNNDX3 INSTDSP PRNTOP PRNTBL	LDA STA AND STA AND TYA AND TAX TYA LDY CPX BEQ LSR BCC LSR LSR CPX BNE INY DEY BNE INY DEY BNE INY DFB JSR PHA LDA JSR CPY	<pre>FMT2,X FORMAT #\$03 P=1 BYTE, LENGTH #\$8F #\$03 #\$8A MNNDX3 A A #\$20 MNNDX2 MNNDX2 MNNDX1 \$FF,\$FF,\$ INSDS1 (PCL),Y PRBYTE #\$01 PRBL2 LENGTH</pre>	INDEX INTO PRINT FORMAT TABLE SAVE FOR ADR FIELD FORMATTING MASK FOR 2BIT LENGTH , 1=2 BYTE, 2=3 BYTE) OPCODE MASK FOR 1XXX1010 TEST SAVE IT OPCODE TO A AGAIN FORM INDEX INTO MNEMONIC TABLE 1) 1XXX1010=>00101XXX 2) XXXYY10=>00110XXX 4) XXXYY10=>00110XXX 4) XXXYY10=>00100XXX 5) XXXXX000=>000XXXXX SFF GEN FMT, LEN BYTES SAVE MNEMONIC TABLE INDEX PRINT 2 BLANKS PRINT 1NST (13 BYTES)
F8AA:         F8AF:         F8B1:         F8B7:         F8C1:         F8C2:         F8C3:         F8C4:         F8C5:         F8C6:         F8C7:         F8C8:         F8C9:         F8C01:         F8D02:         F8D1:         F8D2:         F8D4:         F8D5:         F8D6:         F8D8:         F8D8:         F8D8:         F8D8:         F8D8:         F8D8:         F8D8:         F8E0:	AA BB5 29 85 929 A98 A98 A98 A98 A98 A98 A98 A98 A98 A9	A6 2E 03 2F 8F 03 8A 0B 08 20 FA F2 FF 82 3A 01 4A 2F	F9 FF F8 FD F9	* MNNDX1 MNNDX2 MNNDX3 INSTDSP PRNTOP PRNTBL	LDA STA AND STA AND TYA AND TAX TYA LDY CPX BEQ LSR BCC CPX BEQ LSR BCC LSR SCC LSR DFY SNE FB DFB JSR DFB JSR LDA JSR LDX JSR CPY INY	<pre>FMT2,X FORMAT #\$03 P=1 BYTE, LENGTH #\$8F #\$03 #\$8A MNNDX3 A A #\$20 MNNDX2 MNNDX2 MNNDX1 \$FF,\$FF,\$ INSDS1 (PCL),Y PRBYTE #\$01 PRBL2 LENGTH</pre>	INDEX INTO PRINT FORMAT TABLE SAVE FOR ADR FIELD FORMATTING MASK FOR 2BIT LENGTH , 1=2 BYTE, 2=3 BYTE) OPCODE MASK FOR 1XXX1010 TEST SAVE IT OPCODE TO A AGAIN FORM INDEX INTO MNEMONIC TABLE 1) 1XXX1010=>00101XXX 2) XXXYYY01=>00111XXX 3) XXXYYY10=>00110XXX 4) XXXYY10=>00100XXX 5) XXXXX000=>0000XXXXX 5) XXXXX000=>0000XXXXX SFF GEN FMT, LEN BYTES SAVE MNEMONIC TABLE INDEX PRINT 2 BLANKS PRINT 1NST (13 BYTES) IN A 12 CHR FIELD
F8AA:         F8AD:         F8AF:         F8B1:         F8B4:         F8B7:         F8B8:         F8B8:         F8B8:         F8B8:         F8B8:         F8B6:         F8B7:         F8B8:         F8B7:         F8B8:         F8B7:         F8C1:         F8C2:         F8C3:         F8C4:         F8C5:         F8C6:         F8C7:         F8C8:         F8C9:         F8D1:         F8D2:         F8D2:         F8D2:         F8D2:         F8D2:         F8D2:         F8D2:         F8E1:	AA BD B5 29 85 29 29 29 29 29 29 29 20 40 20 20 20 20 20 20 20 20 20 20 20 20 20	A6 2E 03 2F 8F 03 8A 0B 03 80 8 08 20 FA F2 F7 82 3A 01 4A 2F F1	F9 FF F8 FD F9	* MNNDX1 MNNDX2 MNNDX3 INSTDSP PRNTOP PRNTBL	LDA STA AND STA AND TYA AND TAX TYA LDY CPX BEQ LSR BCC CPX BEQ LSR BCC LSR DFY BNE RTS DFB JSR LDA JSR LDA JSR CPY BNC	<pre>FMT2,X FORMAT #\$03 P=1 BYTE, LENGTH #\$8F #\$03 #\$8A MNNDX3 A #\$20 MNNDX3 A A #\$20 MNNDX2 MNNDX1 \$FF,\$FF,\$ INSDS1 (PCL),Y PRBYTE #\$01 PRBL2 LENGTH PRNTOP</pre>	INDEX INTO PRINT FORMAT TABLE SAVE FOR ADR FIELD FORMATTING MASK FOR 2BIT LENGTH , 1=2 BYTE, 2=3 BYTE) OPCODE MASK FOR 1XXX1010 TEST SAVE IT OPCODE TO A AGAIN FORM INDEX INTO MNEMONIC TABLE 1) 1XXX1010=>00101XXX 2) XXXYY10=>00110XXX 4) XXXYY10=>00110XXX 4) XXXYY10=>00100XXX 5) XXXXX000=>0000XXXXX 5) XXXXX000=>0000XXXXX FFF GEN FMT, LEN BYTES SAVE MNEMONIC TABLE INDEX PRINT 2 BLANKS PRINT 1NST (13 BYTES) IN A 12 CHR FIELD
F8AA:         F8AD:         F8AF:         F8B1:         F8B7:         F8B8:         F8B8:         F8B8:         F8B8:         F8B8:         F8B8:         F8B8:         F8B7:         F8B8:         F8B8:         F8B8:         F8B7:         F8C1:         F8C2:         F8C3:         F8C4:         F8C5:         F8C6:         F8C7:         F8C8:         F8C9:         F8D1:         F8D2:         F8D2:         F8D2:         F8D2:         F8D2:         F8E0:         F8E1:         F8E2:	AA BD B529 85529 AA 980 A00 4A 900 4A 900 4A 900 4A 900 4A 800 60 FF0 481 20 20 420 420 420 420 420 420 420 420 4	A6 2E 03 2F 8F 03 8F 03 80 08 20 FA F2 F7 82 3A 01 4A 2F F1 03	F9 FF F8 FD F9	* MNNDX1 MNNDX2 MNNDX3 INSTDSP PRNTOP PRNTBL	LDA STA AND STA AND TYA AND TAX TYA LDY CPX BEQ LSR BCC LSR BCC LSR BNE LSR DFB JSR LDA JSR LDA JSR LDA JSR LDA SCC LDA SCC LDA SCC LDA SCA CPY EX SCA CPY SCA SCA CPY SCA CPY SCA CPY SCA CPY SCA SCA CPY SCA SCA SCA SCA SCA SCA SCA SCA SCA SCA	<pre>FMT2,X FORMAT #\$03 P=1 BYTE, LENGTH #\$8F #\$03 #\$8A MNNDX3 A #\$20 MNNDX3 A A #\$20 MNNDX1 \$FF,\$FF,\$ INSDS1 (PCL),Y PRBYTE #\$01 PRBL2 LENGTH PRNTOP #\$03</pre>	INDEX INTO PRINT FORMAT TABLE SAVE FOR ADR FIELD FORMATTING MASK FOR 2BIT LENGTH , 1=2 BYTE, 2=3 BYTE) OPCODE MASK FOR 1XXX1010 TEST SAVE IT OPCODE TO A AGAIN FORM INDEX INTO MNEMONIC TABLE 1) 1XXX1010=>00101XXX 2) XXXYYY01=>00111XXX 3) XXXYY10=>00110XXX 4) XXXYY10=>00100XXX 5) XXXX000=>0000XXXXX 5) XXXX000=>0000XXXXX SFF GEN FMT, LEN BYTES SAVE MNEMONIC TABLE INDEX PRINT 2 BLANKS PRINT 1NST (13 BYTES) IN A 12 CHR FIELD CHAR COUNT FOR MNEMONIC PRINT

FRE/•	90	F2			BCC	DRNTRT.	
FOFO.	60				DUC	IRRIDE	DECOVED MNEMONIC INDEX
FSE9:	00				РЬА		RECOVER MNEMONIC INDEX
F8EA:	A8				TAY		
F8EB:	В9	C0	F9		LDA	MNEML,Y	
<b>F8FF</b>	85	20			STA	TMNEM	FETCH 3_CHAR MNEMONIC
FOEL.	50	20			DIA		PETCH J-CHAR MNEMONIC
F.8F.0:	В9	00	F.A		LDA	MNEMR,Y	(PACKED IN 2-BYTES)
F8F3:	85	2D			STA	RMNEM	
F8F5:	Α9	00		PRMN1	LDA	#\$00	
F0F7.	70	05			TDV	#¢05	
r or / :	AU	05			гот	#\$05	
F8F9:	06	2D		PRMN2	ASL	RMNEM	SHIFT 5 BITS OF
F8FB:	26	2C			ROL	LMNEM	CHARACTER INTO A
FOFD.	27				DOT	21111211	(CIENDC CADDY)
FOFD:	ZA				ROL		(CLEARS CARRI)
F8FE:	88				DEY		
F8FF:	D0	F8			BNE	PRMN2	
F901.	69	BF			ADC	#¢BF	ADD "2" OFFSFT
1 901.	0 9	DI			ADC	# Y DI	ADD : OFFSET
F903:	20	ΕD	FD		JSR	COUT	OUTPUT A CHAR OF MNEM
F906:	CA				DEX		
F007.	D0	FC			DNF	DDMNI	
T 0 0 0 .	20		-		JONE	DDDINK	
F909:	20	48	F.9		JSR	PRBLNK	OUTPUT 3 BLANKS
F90C:	Α4	2F			LDY	LENGTH	
F90E:	Δ2	06			T.DX	#\$06	CNT FOR 6 FORMAT BITS
F010.	ΠO	0.2		ומתגמת	CDY	#¢02	Chi fon o fondili bilb
F910:	EU	03		PRADRI	CPX	#\$03	
F912:	F0	1C			BEQ	PRADR5	IF X=3 THEN ADDR.
F914:	06	2E		PRADR2	ASL	FORMAT	
F016.	00	05			PCC	201702	
1910.	90	015			всс	FRADRS	
F918:	BD	в3	F9		LDA	CHAR1-1,	X
F91B:	20	ΕD	FD		JSR	COUT	
E01E.		<u>р</u> 0	то Т			CUAD2 1	v
F91E:	BD	В9	F 9		LDA	CHARZ-1,	λ
F921:	F0	03			BEQ	PRADR3	
F923:	20	ΕD	FD		JSR	COUT	
E026.	<u> </u>			2008002	DEV		
F920:	CA			PRADRS	DEX	_	
F927:	D0	Ε7			BNE	PRADRl	
F929:	60				RTS		
E027.	00				DEV		
F9ZA:	88	_		PRADR4	DEI		
F92B:	30	Ε7			BMI	PRADR2	
F92D:	20	DA	FD		JSR	PRBYTE	
E020.	76	2			TDA	ТОРМАЛ	
r930:	AS	26		PRADRO	LDA	FURMAI	
F932:	C9	E8			CMP	#\$E8	HANDLE REL ADR MODE
F934:	Bl	3A			LDA	(PCL),Y	SPECIAL (PRINT TARGET,
F036.	00	E 2			PCC		
1930.	90	r z			всс	FRADR4	NOI OFFSEI)
F938:	20	56	F9	RELADR	JSR	PCADJ3	
F93B:	AA				TAX		PCL, PCH+OFFSET+1 TO A, Y
F03C.	τo				TNV		
F93C:	E8				INX		
F93C: F93D:	E8 D0	01			INX BNE	PRNTYX	+1 TO Y,X
F93C: F93D: F93F:	E8 D0 C8	01			INX BNE INY	PRNTYX	+1 TO Y,X
F93C: F93D: F93F:	E8 D0 C8	01		DDNWVV	INX BNE INY	PRNTYX	+1 TO Y,X
F93C: F93D: F93F: F940:	E8 D0 C8 98	01		PRNTYX	INX BNE INY TYA	PRNTYX	+1 TO Y,X
F93C: F93D: F93F: F940: F941:	E8 D0 C8 98 20	01 DA	FD	PRNTYX PRNTAX	INX BNE INY TYA JSR	PRNTYX PRBYTE	+l to y,x Output target adr
F93C: F93D: F93F: F940: F941: F944:	E8 D0 C8 98 20 8A	01 DA	FD	PRNTYX PRNTAX PRNTX	INX BNE INY TYA JSR TXA	PRNTYX PRBYTE	+1 TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN
F93C: F93D: F93F: F940: F941: F944: F945:	E8 D0 C8 98 20 8A 4C	01 DA	FD	PRNTYX PRNTAX PRNTX	INX BNE INY TYA JSR TXA IMP	PRNTYX PRBYTE	+1 TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN
F93C: F93D: F93F: F940: F941: F944: F945:	E8 D0 C8 98 20 8A 4C	01 DA DA	FD FD	PRNTYX PRNTAX PRNTX	INX BNE INY TYA JSR TXA JMP	PRNTYX PRBYTE PRBYTE	+1 TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN
F93C: F93D: F93F: F940: F941: F944: F945: F948:	E8 D0 C8 98 20 8A 4C A2	01 DA DA 03	FD FD	PRNTYX PRNTAX PRNTX PRBLNK	INX BNE INY TYA JSR TXA JMP LDX	PRNTYX PRBYTE PRBYTE #\$03	+1 TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT
F93C: F93D: F93F: F940: F941: F944: F945: F948: F948: F94A:	E8 D0 C8 98 20 8A 4C A2 A9	01 DA DA 03 A0	FD FD	PRNTYX PRNTAX PRNTX PRBLNK PRBL2	INX BNE INY TYA JSR TXA JMP LDX LDA	PRNTYX PRBYTE PRBYTE #\$03 #\$A0	+1 TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE
F93C: F93D: F93F: F940: F941: F944: F945: F948: F948: F94A:	E8 D0 C8 98 20 8A 4C A2 A9 20	01 DA DA 03 A0 FD	FD FD	PRNTYX PRNTAX PRNTX PRBLNK PRBL2 PPBL 3	INX BNE INY TYA JSR TXA JMP LDX LDA ISP	PRNTYX PRBYTE PRBYTE #\$03 #\$A0 COUT	+1 TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK
F93C: F93D: F93F: F940: F941: F944: F945: F948: F948: F94A: F94C:	E8 D0 C8 98 20 8A 4C A2 A9 20	01 DA DA 03 A0 ED	FD FD FD	PRNTYX PRNTAX PRNTX PRBLNK PRBL2 PRBL3	INX BNE INY TYA JSR TXA JMP LDX LDA JSR	PRNTYX PRBYTE PRBYTE #\$03 #\$A0 COUT	<pre>+1 TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK</pre>
F93C: F93D: F93F: F940: F941: F944: F944: F945: F948: F948: F94A: F94C: F94F:	E8 D0 C8 98 20 8A 4C A2 A9 20 CA	01 DA DA 03 A0 ED	FD FD FD	PRNTYX PRNTAX PRNTX PRBLNK PRBL2 PRBL3	INX BNE INY TYA JSR TXA JMP LDX LDA JSR DEX	PRNTYX PRBYTE #\$03 #\$A0 COUT	+l to y,x OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK
F93C: F93D: F93F: F940: F941: F944: F944: F948: F948: F94A: F94A: F94A: F94C: F94F: F950:	E8 D0 C8 98 20 8A 4C A2 A9 20 CA D0	01 DA DA 03 A0 ED F8	FD FD FD	PRNTYX PRNTAX PRNTX PRBLNK PRBL2 PRBL3	INX BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE	PRNTYX PRBYTE #\$03 #\$A0 COUT PRBL2	+1 TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK LOOP UNTIL COUNT=0
F93C: F93D: F93F: F940: F941: F944: F945: F948: F948: F94A: F94C: F94F: F950: F952:	E8 D0 C8 98 20 8A 4C A2 20 CA D0 60	01 DA DA 03 A0 ED F8	FD FD FD	PRNTYX PRNTAX PRNTX PRBLNK PRBL2 PRBL3	INX BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE BNE BTS	PRNTYX PRBYTE PRBYTE #\$03 #\$A0 COUT PRBL2	+1 TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK LOOP UNTIL COUNT=0
F93C: F93D: F93F: F940: F941: F944: F945: F948: F948: F948: F947: F947: F947: F950: F952:	E8 D0 C8 98 20 8A 4C A2 20 CA D0 60	01 DA DA 03 A0 ED F8	FD FD FD	PRNTYX PRNTAX PRNTX PRBLNK PRBL2 PRBL3	INX BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE RTS STS	PRNTYX PRBYTE #\$03 #\$A0 COUT PRBL2	+1 TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK LOOP UNTIL COUNT=0
F93C: F93D: F940: F941: F944: F945: F948: F948: F948: F94A: F944: F947: F946: F946: F950: F952: F953:	E8 D0 C8 98 20 8A 4C A2 A9 20 CA D0 60 38	01 DA DA 03 A0 ED F8	FD FD FD	PRNTYX PRNTAX PRNTX PRBLNK PRBL2 PRBL3 PCADJ	INX BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE RTS SEC	PRNTYX PRBYTE #\$03 #\$A0 COUT PRBL2	+1 TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK LOOP UNTIL COUNT=0 0=1-BYTE,1=2-BYTE
F93C: F93D: F93F: F940: F941: F944: F945: F945: F948: F94A: F94A: F947: F950: F952: F953: F954:	E8 D0 C8 98 20 8A 4C A2 20 CA D0 60 38 A5	01 DA DA 03 A0 ED F8 2F	FD FD FD	PRNTYX PRNTAX PRNTX PRBLNK PRBL2 PRBL3 PCADJ PCADJ2	INX BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE RTS SEC LDA	PRNTYX PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH	<pre>+1 TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK LOOP UNTIL COUNT=0 0=1-BYTE,1=2-BYTE 2=3-BYTE</pre>
F93C: F93D: F93F: F944: F944: F944: F948: F948: F948: F948: F948: F948: F950: F950: F952: F953: F953:	E8 D0 C8 98 20 8A 4C A2 20 CA D0 60 38 A5 A4	01 DA DA 03 A0 ED F8 2F 3B	FD FD FD	PRNTYX PRNTAX PRNTX PRBLNK PRBL2 PRBL3 PCADJ PCADJ PCADJ2 PCADJ3	INX BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE RTS SEC LDA LDY	PRNTYX PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCH	<pre>+1 TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK LOOP UNTIL COUNT=0 0=1-BYTE,1=2-BYTE 2=3-BYTE</pre>
F93C: F93F: F94F: F944: F944: F944: F948: F948: F948: F948: F948: F948: F948: F959: F952: F953: F956: F956:	E8 D0 C8 98 20 8A 4C A2 20 CA D0 60 38 A5 A4	01 DA 03 A0 ED F8 2F 3B	FD FD	PRNTYX PRNTX PRNTX PRBLNK PRBL2 PRBL3 PCADJ PCADJ2 PCADJ3	INX BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE RTS SEC LDA LDY	PRNTYX PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCH	<pre>+1 TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK LOOP UNTIL COUNT=0 0=1-BYTE,1=2-BYTE 2=3-BYTE TEST DISPLACEMENT CICK</pre>
F93C: F93F: F947: F944: F944: F944: F945: F948: F947: F947: F947: F950: F952: F952: F954: F956: F958:	E8 D0 C8 98 20 8A 4C A9 20 CA D0 60 38 A5 A4 AA	01 DA DA 03 A0 ED F8 2F 3B	FD FD	PRNTYX PRNTAX PRNTX PRBL2 PRBL3 PCADJ PCADJ2 PCADJ3	INX BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE RTS SEC LDA LDY TAX	PRNTYX PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCH	<pre>+1 TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK LOOP UNTIL COUNT=0 0=1-BYTE,1=2-BYTE 2=3-BYTE TEST DISPLACEMENT SIGN</pre>
F93C: F93F: F94F: F944: F944: F948: F948: F948: F948: F948: F948: F948: F950: F952: F952: F953: F9558: F959: F959:	E8 D0 C8 98 20 8A 4C A2 20 CA D0 60 38 A5 A4 AA 10	01 DA 03 A0 ED F8 2F 3B 01	FD FD	PRNTYX PRNTAX PRNTX PRBLNK PRBL2 PRBL3 PCADJ PCADJ2 PCADJ2 PCADJ3	INX BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE RTS SEC LDA LDY TAX BPL	PRNTYX PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCH PCADJ4	<pre>+l TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK LOOP UNTIL COUNT=0 0=l-BYTE,l=2-BYTE TEST DISPLACEMENT SIGN (FOR REL BRANCH)</pre>
F93C1: F93F1: F940: F940: F944: F944: F944: F945: F947: F947: F950: F9552: F9554: F958: F958: F958: F958:	E8 D0 C8 98 20 8A 4C A9 20 CA D0 60 38 A5 A4 AA 10 88	01 DA DA 03 A0 ED F8 2F 3B 01	FD FD	PRNTYX PRNTAX PRNTX PRBLNK PRBL2 PRBL3 PCADJ PCADJ2 PCADJ2 PCADJ3	INX BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE SEC LDA LDY TAX BPL DEY	PRNTYX PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCH PCADJ4	<pre>+1 TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK LOOP UNTIL COUNT=0 0=1-BYTE,1=2-BYTE 2=3-BYTE TEST DISPLACEMENT SIGN  (FOR REL BRANCH) EXTEND NEG BY DEC PCH</pre>
F93C1: F93F: F94F: F944: F944: F945: F944: F945: F947: F947: F950: F952: F9552: F9554: F9558: F9558: F9558: F9558:	E8 D0 C8 98 20 8A 4C A9 20 CA D0 60 38 A5 A4 A0 88 65	01 DA 03 A0 ED F8 2F 3B 01	FD FD	PRNTYX PRNTAX PRNTX PRBLNK PRBL2 PRBL3 PCADJ PCADJ2 PCADJ2 PCADJ3	INX BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE RTS SEC LDA LDY TAX BPL DEY	PRNTYX PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCH PCADJ4	<pre>+l to y,x OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK LOOP UNTIL COUNT=0 0=1-BYTE,1=2-BYTE 2=3-BYTE TEST DISPLACEMENT SIGN (FOR REL BRANCH) EXTEND NEG BY DEC PCH</pre>
F93C: F93F: F94F: F944: F944: F944: F948: F948: F948: F948: F948: F948: F950: F950: F950: F950: F9558: F958: F958: F958: F958: F958:	E8 D0 C8 98 20 8A 4C A9 20 CA 20 CA 20 CA 00 60 38 5 A4 A0 88 60	01 DA 03 A0 ED F8 2F 3B 01 3A	FD FD	PRNTYX PRNTAX PRNTX PRBLNK PRBL2 PRBL3 PCADJ PCADJ2 PCADJ2 PCADJ4	INX BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE RTS SEC LDA LDY TAX BPL DEY ADC	PRNTYX PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCH PCADJ4 PCL	<pre>+l TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK LOOP UNTIL COUNT=0 0=1-BYTE,1=2-BYTE 2=3-BYTE TEST DISPLACEMENT SIGN (FOR REL BRANCH) EXTEND NEG BY DEC PCH</pre>
F93C1: F93F1: F9441: F9441: F9441: F9448: F9452: F9447: F9447: F9522: F9524: F9554: F9554: F9558: F9558: F9558: F9558: F9558: F9558: F9558: F9558:	E8 D0 C8 98 20 8A 4C 20 CA D0 60 38 A2 D0 60 38 A4 10 88 65 90	01 DA 03 A0 ED F8 2F 3B 01 3A 01	FD FD FD	PRNTYX PRNTAX PRNTX PRBL2 PRBL3 PCADJ PCADJ2 PCADJ3 PCADJ4	INX BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE RTS SEC LDA LDY TAX BPL LDA LDY TAX BPL CC	PRNTYX PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCH PCADJ4 PCL RTS2	<pre>+1 TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK LOOP UNTIL COUNT=0 0=1-BYTE,1=2-BYTE TEST DISPLACEMENT SIGN  (FOR REL BRANCH) EXTEND NEG BY DEC PCH PCL+LENGTH(OR DISPL)+1 TO A</pre>
F93C: F93F: F941: F944: F944: F945: F948: F948: F948: F948: F948: F950: F952: F952: F9554: F959: F958:	E8 D0 C8 20 8A 4C A9 20 A2 D0 60 8A A2 D0 60 88 A4 A0 88 5 90 C8	01 DA 03 A0 ED F8 2F 3B 01 3A 01	FD FD FD	PRNTYX PRNTAX PRNTX PRBLNK PRBL2 PRBL3 PCADJ PCADJ2 PCADJ2 PCADJ3 PCADJ4	INX BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE RTS SEC LDA LDY TAX BPL DEY ADC BCC INY	PRNTYX PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCH PCADJ4 PCL RTS2	<pre>+1 TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK LOOP UNTIL COUNT=0 0=1-BYTE,1=2-BYTE 2=3-BYTE TEST DISPLACEMENT SIGN (FOR REL BRANCH) EXTEND NEG BY DEC PCH PCL+LENGTH(OR DISPL)+1 TO A CARRY INTO Y (PCH)</pre>
F93C1: F93F: F940: F940: F944: F944: F944: F945: F945: F947: F947: F955: F9554: F958	E8 D0 C8 98 20 8A 4C A9 20 CA D0 60 38 A4 A0 88 5 90 8 60 8 8 5 90 8 60 8 8 8 65 90 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8	01 DA 03 A0 ED F8 2F 3B 01 3A 01	FD FD FD	PRNTYX PRNTAX PRNTX PRBLNK PRBL2 PRBL3 PCADJ PCADJ2 PCADJ2 PCADJ3 PCADJ4	INX BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE LDA LDY TAX BPL DEY ADC BCC INY	PRNTYX PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCH PCADJ4 PCL RTS2	<pre>+1 TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK LOOP UNTIL COUNT=0 0=1-BYTE,1=2-BYTE 2=3-BYTE TEST DISPLACEMENT SIGN  (FOR REL BRANCH) EXTEND NEG BY DEC PCH PCL+LENGTH(OR DISPL)+1 TO A CARRY INTO Y (PCH)</pre>
F93C: F93F: F944: F944: F944: F944: F944: F944: F948: F944: F944: F947: F950: F952: F952: F9554: F958:	E8 D0 C8 98 20 8A 4C 20 A9 20 A2 20 CA D0 60 38 A5 A4 10 88 65 90 C8 60	01 DA DA 03 A0 ED F8 2F 3B 01 3A 01	FD FD FD	PRNTYX PRNTAX PRNTX PRBL2 PRBL3 PCADJ PCADJ2 PCADJ2 PCADJ3 PCADJ4 RTS2	INX BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE RTS SEC LDA LDY TAX BPL DEY ADC BCC INY RTS	PRNTYX PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCH PCADJ4 PCL RTS2	<pre>+l TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK LOOP UNTIL COUNT=0 0=1-BYTE,1=2-BYTE 2=3-BYTE TEST DISPLACEMENT SIGN (FOR REL BRANCH) EXTEND NEG BY DEC PCH PCL+LENGTH(OR DISPL)+1 TO A CARRY INTO Y (PCH)</pre>
F93C: F93D: F93F: F944: F944: F944: F948: F948: F948: F948: F948: F948: F950: F950: F950: F9552: F95	E8 D0 C8 20 82 A2 20 CA 20 CA 20 CA 20 CA 20 CA 20 CA 20 CA 20 CA 20 CA 20 CA 20 CA 20 CA 20 CA 20 CA 20 CA 20 CA 20 CA 20 CA 20 CA 20 A 20	01 DA 03 A0 ED F8 2F 3B 01 3A 01	FD FD FD	PRNTYX PRNTAX PRNTX PRBLNK PRBL2 PRBL3 PCADJ PCADJ2 PCADJ2 PCADJ3 PCADJ4 RTS2	INX BNE INY TYA JJSR TXA JMP LDX LDA JSR DEX BNE RTS SEC LDA LDY TAX BPL DEY ADC INY RTS FMT1	PRNTYX PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCH PCADJ4 PCL RTS2 BYTES:	<pre>+1 TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK LOOP UNTIL COUNT=0 0=1-BYTE,1=2-BYTE 2=3-BYTE TEST DISPLACEMENT SIGN (FOR REL BRANCH) EXTEND NEG BY DEC PCH PCL+LENGTH(OR DISPL)+1 TO A CARRY INTO Y (PCH) XXXXXY0 INSTRS</pre>
F93C: F93G: F940: F944: F944: F944: F945: F948: F948: F946: F946: F950: F952: F956: F958: F959: F959: F959: F959: F956:	E8 D0 C8 20 82 A2 20 CA 20 20 20 60 38 A2 20 20 60 38 A4 20 88 65 90 C8 60	01 DA 03 A0 ED F8 2F 3B 01 3A 01	FD FD FD	PRNTYX PRNTAX PRNTX PRBL2 PRBL3 PCADJ PCADJ2 PCADJ2 PCADJ3 PCADJ4 RTS2 *	INX BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE LDA LDX LDA LDY TAX BPL DEY ADC BCC INY RTS FMT1 IF Y	PRNTYX PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCH PCADJ4 PCL RTS2 BYTES:	<pre>+1 TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK LOOP UNTIL COUNT=0 0=1-BYTE,1=2-BYTE TEST DISPLACEMENT SIGN (FOR REL BRANCH) EXTEND NEG BY DEC PCH PCL+LENGTH(OR DISPL)+1 TO A CARRY INTO Y (PCH) XXXXXY0 INSTRS THEN LEFT HALF BYTE</pre>
F93C: F93F: F94F: F944: F944: F944: F944: F948: F948: F948: F948: F950: F950: F950: F9552: F9552: F9558: F9581: F9	E8 D0 C8 20 A2 CA D0 60 385 AA A20 CA D0 60 385 AA A10 865 90 C8 60	01 DA 03 A0 ED F8 2F 3B 01 3A 01	FD FD FD	PRNTYX PRNTAX PRNTX PRBLNK PRBL2 PRBL3 PCADJ PCADJ2 PCADJ2 PCADJ3 PCADJ4 RTS2 *	INX BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE RTS SEC LDA LDY TAX BPL DEY ADC BCC INY RTS FMT1 IF Y	PRNTYX PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCH PCADJ4 PCL RTS2 BYTES: =0 =1	<pre>+l TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK LOOP UNTIL COUNT=0 0=1-BYTE,1=2-BYTE TEST DISPLACEMENT SIGN (FOR REL BRANCH) EXTEND NEG BY DEC PCH PCL+LENGTH(OR DISPL)+1 TO A CARRY INTO Y (PCH) XXXXXY0 INSTRS THEN LEFT HALF BYTE THEN DIGHT WALE DYME</pre>
F93C: F93D: F93F: F944: F944: F944: F948: F948: F948: F948: F948: F948: F950: F950: F950: F950: F955: F955: F955: F955: F955: F955: F955: F956: F961:	E8 D0 28 40 20 20 20 20 20 20 20 20 20 20 20 20 20	01 DA DA 03 A0 ED F8 2F 3B 01 3A 01	FD FD FD	PRNTYX PRNTAX PRNTX PRBLNK PRBL2 PRBL3 PCADJ PCADJ2 PCADJ2 PCADJ3 PCADJ4 RTS2 *	INX BNE INY TYA JJRR TXA JMP LDX LDA JSR DEX BNE RTS SEC LDA LDY TAX BPL DEY ADC INY RTS FMT1 IF Y IF Y	PRNTYX PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCH PCADJ4 PCL RTS2 BYTES: =0 =1	<pre>+1 TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK LOOP UNTIL COUNT=0 0=1-BYTE,1=2-BYTE 2=3-BYTE TEST DISPLACEMENT SIGN (FOR REL BRANCH) EXTEND NEG BY DEC PCH PCL+LENGTH(OR DISPL)+1 TO A CARRY INTO Y (PCH) XXXXXY0 INSTRS THEN LEFT HALF BYTE THEN RIGHT HALF BYTE THEN RIGHT HALF BYTE</pre>
F93C: F93G: F940: F944: F944: F944: F945: F948: F947: F947: F950: F952: F952: F952: F958: F959: F959: F959: F959: F959: F956:	E8 D0 C8 920 20 20 20 20 20 20 20 20 20 20 20 20 2	01 DA 03 A0 ED F8 2F 3B 01 3A 01	FD FD FD	PRNTYX PRNTAX PRNTX PRBL2 PRBL3 PCADJ PCADJ2 PCADJ3 PCADJ4 RTS2 * *	INX BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE CDA LDA LDA LDA LDA LDA LDA LDA LDA LDA L	PRNTYX PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCH PCADJ4 PCL RTS2 BYTES: =0 =1	<pre>+l to y,x OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK LOOP UNTIL COUNT=0 0=1-BYTE,1=2-BYTE 0=1-BYTE,1=2-BYTE TEST DISPLACEMENT SIGN (FOR REL BRANCH) EXTEND NEG BY DEC PCH PCL+LENGTH(OR DISPL)+1 TO A CARRY INTO Y (PCH) XXXXXY0 INSTRS THEN LEFT HALF BYTE THEN RIGHT HALF BYTE (X=INDEX)</pre>
F93C: F93F: F94F: F944: F944: F944: F948: F948: F948: F948: F948: F948: F950: F950: F950: F9552: F9558: F959: F958	E8 D0 C8 820 8A 4C 20 20 20 20 20 20 20 20 20 20 20 20 20	01 DA DA 03 A0 ED F8 2F 3B 01 3A 01 20	FD FD FD	PRNTYX PRNTAX PRNTX PRBLNK PRBL2 PRBL3 PCADJ PCADJ2 PCADJ2 PCADJ3 PCADJ4 RTS2 * *	INX BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE RTS SEC LDA LDY TAX BPL DEY ADC BCC INY RTS FMT1 IF Y: IF Y:	PRNTYX PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCH PCADJ4 PCL RTS2 BYTES: =0 =1	<pre>+l to y,x OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK LOOP UNTIL COUNT=0 0=1-BYTE,1=2-BYTE TEST DISPLACEMENT SIGN (FOR REL BRANCH) EXTEND NEG BY DEC PCH PCL+LENGTH(OR DISPL)+1 TO A CARRY INTO Y (PCH) XXXXXY0 INSTRS THEN LEFT HALF BYTE THEN RIGHT HALF BYTE (X=INDEX)</pre>
F93C1 F93C1 F93C1 F94C1 F94C1 F9441 F9441 F9441 F945 F945 F945 F945 F9502 F950	E8 D08 208 A22 D060 38 A24 20 D060 38 A4A10 88 60 030 30 04 00 30	01 DA DA 03 A0 ED F8 2F 3B 01 3A 01	FD FD FD	PRNTYX PRNTAX PRNTX PRBL2 PRBL2 PRBL3 PCADJ2 PCADJ2 PCADJ3 PCADJ4 RTS2 * * *	INX BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE CDA LDA LDA LDA LDA LDA LDA LDA LDA LDA TAX BPL DEY ADC BCC INY RTS FMT1 IF Y:	PRNTYX PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCADJ4 PCL RTS2 BYTES: =0 =1	<pre>+1 TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK LOOP UNTIL COUNT=0 0=1-BYTE,1=2-BYTE 2=3-BYTE TEST DISPLACEMENT SIGN  (FOR REL BRANCH) EXTEND NEG BY DEC PCH PCL+LENGTH(OR DISPL)+1 TO A CARRY INTO Y (PCH)</pre>
F93C: F93F: F94F: F944: F944: F944: F944: F944: F944: F947: F950: F950: F952: F9552: F958:	E8 D0 C8 820 8A 4C 20 8A 4C 20 8A 4C 20 60 38 A5 A4 A0 0 C8 60 C8 60 C8 60 C8 20 C8 20 C8 20 C8 20 C8 20 C8 20 C8 20 C8 20 C8 20 20 C8 C8 C8 C8 C8 C8 C8 C8 C8 C8 C8 C8 C8	01 DA 03 A0 ED F8 2F 3B 01 3A 01 20 00	FD FD FD	PRNTYX PRNTAX PRNTX PRBLNK PRBL2 PRBL3 PCADJ2 PCADJ2 PCADJ3 PCADJ4 RTS2 * * * *	INX BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE RTS SEC LDA LDY TAX BPL DEY ADC BCC INY RTS FMT1 IF Y: IF Y: DFB	PRNTYX PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCH PCADJ4 PCL RTS2 BYTES: =0 =1 \$04,\$20,	<pre>+l TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK LOOP UNTIL COUNT=0 0=1-BYTE,1=2-BYTE 2=3-BYTE TEST DISPLACEMENT SIGN (FOR REL BRANCH) EXTEND NEG BY DEC PCH PCL+LENGTH(OR DISPL)+1 TO A CARRY INTO Y (PCH) XXXXXY0 INSTRS THEN LEFT HALF BYTE THEN RIGHT HALF BYTE (X=INDEX)</pre>
F93C1 F93C1 F93C1 F94C1 F94C1 F9441 F9441 F9441 F9442 F9450 F947 F9502 F95521 F95521 F95521 F95581 F95581 F95581 F95581 F95581 F95601 F9601 F9601 F9601 F9601 F9601 F9601 F9601 F9601 F9601 F9601 F9601 F9601 F9601 F9601 F9502 F950	E8 D08 208 A29 200 A29 200 CA D00 600 385 AA4 200 CA 000 385 AA4 108865 908 60 04 30080 80	01 DA 03 A0 ED F8 2F 3B 01 3A 01 20 0D	FD FD FD	PRNTYX PRNTAX PRNTX PRBLNK PRBL2 PRBL3 PCADJ2 PCADJ2 PCADJ3 PCADJ4 RTS2 * * * * *	INX BNE INY JSR TYA JSR TXA JMP LDX LDA JSR DEX BNE RTS SEC LDA LDY TAX BPL DEY ADC BCC INY RTS FMT1 IF Y: IF Y:	PRNTYX PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCADJ4 PCL RTS2 BYTES: =0 =1 \$04,\$20,	<pre>+1 TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK LOOP UNTIL COUNT=0 0=1-BYTE,1=2-BYTE 2=3-BYTE TEST DISPLACEMENT SIGN (FOR REL BRANCH) EXTEND NEG BY DEC PCH PCL+LENGTH(OR DISPL)+1 TO A CARRY INTO Y (PCH) XXXXXY0 INSTRS THEN LEFT HALF BYTE THEN RIGHT HALF BYTE (X=INDEX) \$54,\$30,\$0D</pre>
F93C: F93G: F93F: F944: F944: F944: F944: F944: F945: F944: F945: F950: F950: F9552: F9552: F9552: F9552: F9552: F9551: F9561: F960: F961: F962: F962: F966: F964: F964: F964: F966:	E8 D08 20 84 20 20 20 20 20 20 20 20 20 20 20 20 20	01 DA 03 04 ED F8 2F 3B 01 3A 01 3A 01 20 004 22	FD FD FD	PRNTYX PRNTAX PRNTX PRBL2 PRBL3 PCADJ PCADJ2 PCADJ3 PCADJ4 RTS2 * * * * *	INX BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE CDA LDA LDA LDA LDA LDA LDA LDA LDA LDA L	PRNTYX PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCADJ4 PCL RTS2 BYTES: =0 =1 \$04,\$20, \$80,\$04.	<pre>+1 TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK LOOP UNTIL COUNT=0 0=1-BYTE,1=2-BYTE TEST DISPLACEMENT SIGN (FOR REL BRANCH) EXTEND NEG BY DEC PCH PCL+LENGTH(OR DISPL)+1 TO A CARRY INTO Y (PCH) XXXXXY0 INSTRS THEN LEFT HALF BYTE THEN RIGHT HALF BYTE (X=INDEX) \$54,\$30,\$0D \$90,\$03,\$22</pre>
F93C1: F93C1: F93F: F944: F944: F944: F944: F944: F944: F944: F944: F944: F950: F950: F950: F950: F950: F950: F950: F950: F950: F960: F960: F960: F960: F960: F960: F960: F960: F960: F960: F960: F960: F960: F960: F960: F960: F960: F960: F960: F950	E8 D08 20 84 20 20 20 20 20 20 20 20 20 20 20 20 20	01 DA 03 A0 ED F8 2F 3B 01 3A 01 20 0D 04 233	FD FD FD 54 90	PRNTYX PRNTAX PRNTX PRBLNK PRBL2 PRBL3 PCADJ2 PCADJ2 PCADJ3 PCADJ4 RTS2 * * * *	INX BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE RTS SEC LDA LDY TAX BPL DEY ADC BCC INY RTS FMT1 IF Y: IF Y: DFB	PRNTYX PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCADJ4 PCL RTS2 BYTES: =0 =1 \$04,\$20,\$ \$80,\$04,\$	<pre>+l TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK LOOP UNTIL COUNT=0 0=1-BYTE,1=2-BYTE TEST DISPLACEMENT SIGN (FOR REL BRANCH) EXTEND NEG BY DEC PCH PCL+LENGTH(OR DISPL)+1 TO A CARRY INTO Y (PCH) XXXXXY0 INSTRS THEN LEFT HALF BYTE THEN RIGHT HALF BYTE (X=INDEX) \$54,\$30,\$0D \$90,\$03,\$22</pre>
F93C1: F93C1: F93C1: F947: F947: F947: F944: F944: F944: F944: F944: F947: F950: F95	E80C9820A4220C00638544A292CA06086590C860 43003454	01 DA DA 03 A0 ED F8 2F 3B 01 3A 01 20 00 04 22 33	FD FD FD 54 90	PRNTYX PRNTAX PRNTX PRBLNK PRBL2 PRBL3 PCADJ2 PCADJ2 PCADJ3 PCADJ4 RTS2 * * * *	INX BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE CDA LDA LDA LDA LDA LDA LDA LDA LDA TAX BPL DEY ADC BCC INY RTS FMT1 IF Y: IF Y: DFB	PRNTYX PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCADJ4 PCL RTS2 BYTES: =0 =1 \$04,\$20, \$80,\$04,	<pre>+1 TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK LOOP UNTIL COUNT=0 0=1-BYTE,1=2-BYTE 2=3-BYTE TEST DISPLACEMENT SIGN (FOR REL BRANCH) EXTEND NEG BY DEC PCH PCL+LENGTH(OR DISPL)+1 TO A CARRY INTO Y (PCH) XXXXXY0 INSTRS THEN LEFT HALF BYTE (X=INDEX) \$54,\$30,\$0D \$90,\$03,\$22 \$CD \$80,\$04</pre>
F93C: F93F: F944: F944: F944: F944: F944: F944: F944: F944: F944: F944: F950: F952: F952: F952: F952: F952: F955: F955: F955: F956: F966:	E8 D08 20 84 C298 20 84 C20 D0 60 84 20 CA D0 60 83 85 20 80 20 80 20 80 20 80 20 80 20 80 80 80 80 80 80 80 80 80 80 80 80 80	01 DA 03 A0 ED F8 2F 3B 01 3A 01 3A 01 20 00 4 22 33 04	FD FD 54 90	PRNTYX PRNTAX PRNTX PRBLNK PRBL2 PRBL3 PCADJ PCADJ2 PCADJ2 PCADJ4 RTS2 * * * * *	INX BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE RTS SEC LDA LDY TAX BPL LDA LDY TAX BPL ADC BCC INY RTS FMT1 IF Y: IF Y: DFB DFB	PRNTYX PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCADJ4 PCL RTS2 BYTES: =0 \$04,\$20, \$80,\$04, \$54,\$33,	<pre>+l TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK LOOP UNTIL COUNT=0 0=1-BYTE,1=2-BYTE TEST DISPLACEMENT SIGN (FOR REL BRANCH) EXTEND NEG BY DEC PCH PCL+LENGTH(OR DISPL)+1 TO A CARRY INTO Y (PCH) XXXXXY0 INSTRS THEN LEFT HALF BYTE THEN RIGHT HALF BYTE (X=INDEX) \$54,\$30,\$0D \$90,\$03,\$22 \$0D,\$80,\$04</pre>
F93C1: F93F: F93F: F944:: F944:: F944:: F944:: F944:: F944:: F944:: F955: F955: F955: F955: F955: F955: F955: F955: F955: F955: F956: F966	E8 D08 20 20 20 20 20 20 20 20 20 20 20 20 20	01 DA 03 ED F8 2F 3B 01 3A 01 20 00 04 222 30 4 04	FD FD 54 90 20	PRNTYX PRNTAX PRNTX PRBLNK PRBL2 PRBL3 PCADJ PCADJ2 PCADJ3 PCADJ4 RTS2 * * * *	INX BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE RTS SEC LDA LDY TAX BPL DEY ADC BCC INY RTS FMT1 IF Y: IF Y: DFB DFB	PRNTYX PRBYTE #\$03 #\$40 COUT PRBL2 LENGTH PCADJ4 PCL RTS2 BYTES: =0 =1 \$04,\$20, \$80,\$04, \$54,\$33,	<pre>+l TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK LOOP UNTIL COUNT=0 0=1-BYTE,1=2-BYTE 2=3-BYTE TEST DISPLACEMENT SIGN (FOR REL BRANCH) EXTEND NEG BY DEC PCH PCL+LENGTH(OR DISPL)+1 TO A CARRY INTO Y (PCH) XXXXXXY0 INSTRS THEN LEFT HALF BYTE (X=INDEX) \$54,\$30,\$0D \$90,\$03,\$22 \$0D,\$80,\$04</pre>
F93C1: F93C1: F93F1: F940: F940: F944: F944: F944: F944: F944: F944: F946: F952: F952: F952: F952: F952: F9558: F9558: F956: F9561: F9661: F9661: F9661: F9661: F9661: F9661: F974:	E8008842920A0060842000000000000000000000000000000000	01 DA DA 03 A0 ED F8 2F 3B 01 3A 01 20 00 04 22 33 04 04 33	FD FD 54 90 20	PRNTYX PRNTAX PRNTX PRBLNK PRBL2 PRBL3 PCADJ2 PCADJ2 PCADJ3 PCADJ4 RTS2 * * * * *	INX BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE CDA LDA LDA LDA LDA LDA LDA LDA LDA LDA L	PRNTYX PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCADJ4 PCL RTS2 BYTES: =0 =1 \$04,\$20, \$80,\$04, \$54,\$33, \$90,\$04	<pre>+1 TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK LOOP UNTIL COUNT=0 0=1-BYTE,1=2-BYTE 2=3-BYTE TEST DISPLACEMENT SIGN (FOR REL BRANCH) EXTEND NEG BY DEC PCH PCL+LENGTH(OR DISPL)+1 TO A CARRY INTO Y (PCH) XXXXXY0 INSTRS THEN LEFT HALF BYTE THEN RIGHT HALF BYTE (X=INDEX) \$54,\$30,\$0D \$90,\$03,\$22 \$0D,\$80,\$04 \$20,\$54,\$33</pre>
F93C:         F93F:         F93F:         F944:         F955:         F955:         F966:         F971:         F971:	E8         D08         28         C98         200         38         200         38         200         38         200         38         38         38         38         38         38         38         38         38         38         38         38         39         38         39         30	01 DA 03 A0 ED F8 2F 3B 01 3A 01 3A 01 20 00 4 22 33 04 04 33 04	FD FD 54 90 20	PRNTYX PRNTAX PRNTX PRBLNK PRBL2 PRBL3 PCADJ2 PCADJ2 PCADJ3 PCADJ4 RTS2 * * * *	INX BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE RTS SEC LDA LDY TAX BPL LDY TAX BPL ADC BCC INY RTS FMT1 IF Y: IF Y: DFB DFB DFB	PRNTYX PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCADJ4 PCL RTS2 BYTES: =1 \$04,\$20, \$80,\$04, \$80,\$04, \$90,\$04,	<pre>+l TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK LOOP UNTIL COUNT=0 0=1-BYTE,1=2-BYTE TEST DISPLACEMENT SIGN (FOR REL BRANCH) EXTEND NEG BY DEC PCH PCL+LENGTH(OR DISPL)+1 TO A CARRY INTO Y (PCH) XXXXXY0 INSTRS THEN RIGHT HALF BYTE THEN RIGHT HALF BYTE (X=INDEX) \$54,\$30,\$0D \$90,\$03,\$22 \$0D,\$80,\$04 \$20,\$54,\$33</pre>
F93C1: F93C1: F93C1: F94C1: F94C1: F94C1: F94C1: F94C1: F94C1: F95C2: F9	E8         D08         20         84         20 <td>01 DA DA 03 A0 ED F8 2F 3B 01 3A 01 20 0D 04 22 33 04 23 3 04 33 80</td> <td>FD FD FD 54 90 0D 20</td> <td>PRNTYX PRNTAX PRNTX PRBLNK PRBL2 PRBL3 PCADJ2 PCADJ2 PCADJ3 PCADJ4 RTS2 * * * * *</td> <td>INX BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE RTS SEC LDA LDY TAX BPL DEY ADC BCC INY RTS FMT1 IF Y: IF Y: DFB DFB</td> <td>PRNTYX PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCADJ4 PCL RTS2 BYTES: =0 =1 \$04,\$20, \$80,\$04, \$54,\$33, \$90,\$04,</td> <td><pre>+1 TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK LOOP UNTIL COUNT=0 0=1-BYTE,1=2-BYTE 2=3-BYTE TEST DISPLACEMENT SIGN (FOR REL BRANCH) EXTEND NEG BY DEC PCH PCL+LENGTH(OR DISPL)+1 TO A CARRY INTO Y (PCH) XXXXXY0 INSTRS THEN LEFT HALF BYTE (X=INDEX) \$54,\$30,\$0D \$90,\$03,\$22 \$0D,\$80,\$04 \$20,\$54,\$33 </pre></td>	01 DA DA 03 A0 ED F8 2F 3B 01 3A 01 20 0D 04 22 33 04 23 3 04 33 80	FD FD FD 54 90 0D 20	PRNTYX PRNTAX PRNTX PRBLNK PRBL2 PRBL3 PCADJ2 PCADJ2 PCADJ3 PCADJ4 RTS2 * * * * *	INX BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE RTS SEC LDA LDY TAX BPL DEY ADC BCC INY RTS FMT1 IF Y: IF Y: DFB DFB	PRNTYX PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCADJ4 PCL RTS2 BYTES: =0 =1 \$04,\$20, \$80,\$04, \$54,\$33, \$90,\$04,	<pre>+1 TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK LOOP UNTIL COUNT=0 0=1-BYTE,1=2-BYTE 2=3-BYTE TEST DISPLACEMENT SIGN (FOR REL BRANCH) EXTEND NEG BY DEC PCH PCL+LENGTH(OR DISPL)+1 TO A CARRY INTO Y (PCH) XXXXXY0 INSTRS THEN LEFT HALF BYTE (X=INDEX) \$54,\$30,\$0D \$90,\$03,\$22 \$0D,\$80,\$04 \$20,\$54,\$33 </pre>
F93C1:         F93F93F1:         F944:         F955:         F955:         F966:         F970:         F970:	E8 D08 20 84 20 20 20 20 20 20 20 20 20 20 20 20 20	01 DA 03 A0 ED F8 2F 3B 01 3A 01 3A 01 20 004 22 33 04 04 33 04 04 33 04	FD FD 54 90 20 04	PRNTYX PRNTAX PRNTX PRBL2 PRBL3 PCADJ PCADJ2 PCADJ3 PCADJ4 RTS2 * * * * *	INX BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE RTS SEC LDA LDY TAX BPL LDA LDY TAX BPL CO INY RTS FMT1 IF Y: IF Y: DFB DFB DFB DFB	PRNTYX PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCADJ4 PCL RTS2 BYTES: =0 =1 \$04,\$20, \$80,\$04, \$54,\$33, \$90,\$04, \$00,\$80,	<pre>+1 TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK LOOP UNTIL COUNT=0 0=1-BYTE,1=2-BYTE TEST DISPLACEMENT SIGN (FOR REL BRANCH) EXTEND NEG BY DEC PCH PCL+LENGTH(OR DISPL)+1 TO A CARRY INTO Y (PCH) XXXXXY0 INSTRS THEN LEFT HALF BYTE THEN RIGHT HALF BYTE (X=INDEX) \$54,\$30,\$00 \$90,\$03,\$22 \$0D,\$80,\$04 \$20,\$54,\$33 \$04,\$90,\$04</pre>
F93C1:         F93J7:         F941:         F944:         F955:         F955:         F960:         F960:         F960:         F966:         F966:         F966:         F966:         F966:         F966:         F966:         F966:         F976:         F976:         F9778:	E8 D08 28 A2 20 A2 D06 38 A2 A2 006 38 54 20 20 20 20 20 20 20 20 20 20 20 20 20	01 DA 03 ED F8 2F 3B 01 3A 01 3A 01 20 00 04 223 304 04 33 80 04 54	FD FD 54 90 0D 20 04 3B	PRNTYX PRNTAX PRNTX PRBLNK PRBL2 PRBL3 PCADJ2 PCADJ2 PCADJ3 PCADJ4 RTS2 * * * *	INX BNE INY JSR TYA JMP LDX LDA JSR DEX BNE RTS SEC LDA LDY TAX BPL DEY ADC BCC IDA LDY TAX BPL DFF BCC INY RTS FMT1 IF Y: IF Y: DFB DFB DFB	PRNTYX PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCADJ4 PCL RTS2 BYTES: =0 =1 \$04,\$20, \$80,\$04, \$54,\$33, \$90,\$04, \$0D,\$80,	<pre>+l TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK LOOP UNTIL COUNT=0 0=1-BYTE,1=2-BYTE TEST DISPLACEMENT SIGN (FOR REL BRANCH) EXTEND NEG BY DEC PCH PCL+LENGTH(OR DISPL)+1 TO A CARRY INTO Y (PCH) XXXXXY0 INSTRS THEN LEFT HALF BYTE THEN RIGHT HALF BYTE (X=INDEX) \$54,\$30,\$0D \$90,\$03,\$22 \$0D,\$80,\$04 \$20,\$54,\$33 \$04,\$90,\$04</pre>
F93C1:         F93G1:         F93G1:         F940:         F944:         F952:         F9551:         F9552:         F9551:         F9551:         F9561:         F9661:         F9661:         F9662:         F9661:         F9661:         F9661:         F9661:         F9761:         F9761:         F9779:         F9779:         F9779:         F9779:         F9779:	E8         D08         28         C92         84         20         638         42         200         638         442         200         638         442         200         638         442         200         638         440         865         020         0300         540         900         200         200         200         200         201         202         203         204         205         206         400         207         208         209         200         201         202         203         204         205         206         207         208         209         2000         201         202         203         204<	01 DA DA 03 A0 ED F8 2F 3B 01 3A 01 20 004 22 33 04 04 23 3 04 52 04	FD FD FD 54 90 0D 20 04 3B	PRNTYX PRNTAX PRNTX PRBLNK PRBL2 PRBL3 PCADJ2 PCADJ2 PCADJ3 PCADJ4 RTS2 * * * *	INX BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE CLDA LDY TAX BPL DEY ADC BCC INY RTS FMT1 IF Y: IF Y: DFB DFB DFB DFB	PRNTYX PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCADJ4 PCL RTS2 BYTES: =0 =1 \$04,\$20, \$80,\$04, \$54,\$33, \$90,\$04, \$00,\$80, \$20,\$54	<pre>+1 TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK LOOP UNTIL COUNT=0 0=1-BYTE,1=2-BYTE 2=3-BYTE TEST DISPLACEMENT SIGN (FOR REL BRANCH) EXTEND NEG BY DEC PCH PCL+LENGTH(OR DISPL)+1 TO A CARRY INTO Y (PCH) XXXXXY0 INSTRS THEN LEFT HALF BYTE (X=INDEX) \$54,\$30,\$0D \$90,\$03,\$22 \$0D,\$80,\$04 \$20,\$54,\$33 \$04,\$90,\$04 \$38,\$0D \$80</pre>
F93C1:         F93F93F1         F944:         F952:         F955:         F955:         F966:         F966:         F966:         F966:         F966:         F966:         F966:         F976:         F9771:         F977:	E8         D08         208         A20         D06         38         422         CD0         608         A400         608         A400         608         040         540         040         540         020         020	01 DA 03 00 ED F8 2F 3B 01 3A 01 3A 01 20 00D 04 233 04 33 04 33 04 54 80	FD FD FD 54 90 0D 20 04 38	PRNTYX PRNTAX PRNTX PRBLNK PRBL2 PRBL3 PCADJ PCADJ2 PCADJ2 PCADJ4 RTS2 * * * *	INX BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE RTS SEC LDA LDY TAX BPL LDA LDY TAX BPL CPA BPL TAX BPL TAX BPL DFB DFB DFB DFB DFB DFB	PRNTYX PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCADJ4 PCL RTS2 BYTES: =0 \$04,\$20, \$80,\$04, \$90,\$04, \$90,\$04, \$0D,\$80, \$20,\$54,	<pre>+l TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK LOOP UNTIL COUNT=0 0=1-BYTE,1=2-BYTE TEST DISPLACEMENT SIGN (FOR REL BRANCH) EXTEND NEG BY DEC PCH PCL+LENGTH(OR DISPL)+1 TO A CARRY INTO Y (PCH) XXXXXY0 INSTRS THEN LEFT HALF BYTE THEN RIGHT HALF BYTE (X=INDEX) \$54,\$30,\$00 \$90,\$03,\$22 \$0D,\$80,\$04 \$20,\$54,\$33 \$04,\$90,\$04 \$3B,\$0D,\$80</pre>
F93CD:         F93GD:         F93F         F9340:         F9441:         F9442:         F9443:         F9444:         F9442:         F9442:         F9442:         F9442:         F9442:         F9552:         F9552:         F9558:         F9558:         F9561:         F9661:         F9662:         F9665:         F9667:         F9667:         F96761:         F9761:         F9662:         F9661:         F9662:         F9764:         F9774:         F9779:         F9779:         F9778:         F9780:	E8         D08         208         A20         D00         383         A4         A00         383         A4         A00         383         A4         A00         383         44         400         383         44         400         383         400         380 <td>01 DA DA 03 A0 ED F8 2F 3B 01 3A 01 3A 01 20 0D 04 22 33 04 54 04 33 80 04 590</td> <td>FD FD FD 54 90 0D 20 04 3B 00</td> <td>PRNTYX PRNTAX PRNTX PRBL2 PRBL2 PRBL3 PCADJ2 PCADJ2 PCADJ4 RTS2 * * * * *</td> <td>INX BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE RTS SEC LDA LDY TAX BPL DEY ADC BCC INY RTS FMT1 IF Y: IF Y: DFB DFB DFB DFB DFB</td> <td>PRNTYX PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCADJ4 PCL RTS2 BYTES: =0 =1 \$04,\$20, \$80,\$04, \$54,\$33, \$90,\$04, \$0D,\$80, \$20,\$54,\$</td> <td><pre>+1 TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK LOOP UNTIL COUNT=0 0=1-BYTE,1=2-BYTE 2=3-BYTE TEST DISPLACEMENT SIGN (FOR REL BRANCH) EXTEND NEG BY DEC PCH PCL+LENGTH(OR DISPL)+1 TO A CARRY INTO Y (PCH) XXXXXY0 INSTRS THEN LEFT HALF BYTE THEN RIGHT HALF BYTE (X=INDEX) \$54,\$30,\$0D \$90,\$03,\$22 \$0D,\$80,\$04 \$20,\$54,\$33 \$04,\$90,\$04 \$3B,\$0D,\$80</pre></td>	01 DA DA 03 A0 ED F8 2F 3B 01 3A 01 3A 01 20 0D 04 22 33 04 54 04 33 80 04 590	FD FD FD 54 90 0D 20 04 3B 00	PRNTYX PRNTAX PRNTX PRBL2 PRBL2 PRBL3 PCADJ2 PCADJ2 PCADJ4 RTS2 * * * * *	INX BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE RTS SEC LDA LDY TAX BPL DEY ADC BCC INY RTS FMT1 IF Y: IF Y: DFB DFB DFB DFB DFB	PRNTYX PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCADJ4 PCL RTS2 BYTES: =0 =1 \$04,\$20, \$80,\$04, \$54,\$33, \$90,\$04, \$0D,\$80, \$20,\$54,\$	<pre>+1 TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK LOOP UNTIL COUNT=0 0=1-BYTE,1=2-BYTE 2=3-BYTE TEST DISPLACEMENT SIGN (FOR REL BRANCH) EXTEND NEG BY DEC PCH PCL+LENGTH(OR DISPL)+1 TO A CARRY INTO Y (PCH) XXXXXY0 INSTRS THEN LEFT HALF BYTE THEN RIGHT HALF BYTE (X=INDEX) \$54,\$30,\$0D \$90,\$03,\$22 \$0D,\$80,\$04 \$20,\$54,\$33 \$04,\$90,\$04 \$3B,\$0D,\$80</pre>
F93C1:         F93G1:         F93G1:         F94G1:         F94G1:         F94G1:         F94G1:         F94G1:         F94G1:         F94G1:         F94G1:         F94G1:         F95G2:         F95G1:         F95G1:         F96G1:         F96G2:         F96G1:         F96G2:         F96G2:         F96G3:         F96G4:         F9761:         F9761:         F9767:         F9767:         F9771:         F9778:         F9778:         F9780:         F9800:	E8         D08         28         C92         84         200         63         85         92         84         200         865         0300         500         400         900         200         210 <tr td=""></tr>	01 DA DA 30 ED F8 2F 3B 01 3A 01 20 004 222 33 04 04 23 304 04 24 80 04 44	FD FD 54 90 01 20 04 3B 00	PRNTYX PRNTAX PRNTX PRBLNK PRBL2 PRBL3 PCADJ2 PCADJ2 PCADJ4 RTS2 * * * * *	INX BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE CDA LDA LDA LDA LDA LDA LDA LDA LDA TAX BPL DEY ADC BCC INY RTS FMT1 IF Y: DFB DFB DFB DFB DFB DFB	PRNTYX PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCADJ4 PCL RTS2 BYTES: =0 =1 \$04,\$20, \$80,\$04, \$54,\$33, \$90,\$04, \$20,\$54, \$20,\$54, \$04,\$90.	<pre>+1 TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK LOOP UNTIL COUNT=0 0=1-BYTE,1=2-BYTE 2=3-BYTE TEST DISPLACEMENT SIGN (FOR REL BRANCH) EXTEND NEG BY DEC PCH PCL+LENGTH(OR DISPL)+1 TO A CARRY INTO Y (PCH) XXXXXY0 INSTRS THEN LEFT HALF BYTE THEN RIGHT HALF BYTE (X=INDEX) \$54,\$30,\$0D \$90,\$03,\$22 \$0D,\$80,\$04 \$20,\$54,\$33 \$04,\$90,\$04 \$3B,\$0D,\$80 \$00,\$22,\$44</pre>
F93C1:         F93JF:         F944:         F955:         F955:         F966:         F966:         F966:         F966:         F966:         F976:         F9774:         F9778:         F9780:         F9850:         F9850:         F9850:         F970:         F970:         F970:         F970:         F970:         F980:         F980:         F980:	E8         D08         28         C98         200         300	01 DA 03 00 ED F8 2F 3B 01 3A 01 3A 01 20 00D 42 233 04 33 04 33 04 54 80 90 40 00	FD FD FD 54 90 0D 20 04 3B 00 C8	PRNTYX PRNTAX PRNTX PRBLNK PRBL2 PRBL3 PCADJ PCADJ2 PCADJ2 PCADJ4 RTS2 * * * * *	INX BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE RTS SEC LDA LDY TAX BPL LDY TAX BPL CY ADC BCC INY RTS FMT1 IF Y: IF Y: DFB DFB DFB DFB DFB DFB	PRNTYX PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCADJ4 PCL RTS2 BYTES: =0 \$04,\$20, \$80,\$04, \$90,\$04, \$90,\$04, \$90,\$04, \$00,\$80, \$00,\$80, \$00,\$54, \$04,\$90,	<pre>+l TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK LOOP UNTIL COUNT=0 0=1-BYTE,1=2-BYTE TEST DISPLACEMENT SIGN (FOR REL BRANCH) EXTEND NEG BY DEC PCH PCL+LENGTH(OR DISPL)+1 TO A CARRY INTO Y (PCH) XXXXXY0 INSTRS THEN LEFT HALF BYTE THEN RIGHT HALF BYTE (X=INDEX) \$54,\$30,\$0D \$90,\$03,\$22 \$0D,\$80,\$04 \$20,\$54,\$33 \$04,\$90,\$04 \$3B,\$0D,\$80 \$00,\$22,\$44</pre>
F93CL:         F93GP:         F940::         F944::         F955::         F955::         F9558::         F9560::         F9661::         F9662::         F9664::         F9761::         F9761::         F9761::         F9762::         F9761::         F9761::         F9762::         F9779::         F9783::         F9833::         F9835:	E8         D08       28         L08       20         A20       00         638       34         AA0       865         030       54         900       20         223       4	01 DA DA 03 A0 ED F8 2F 3B 01 3A 01 20 00 04 22 33 04 54 09 044 540 90 042 00 00 00 00 00 00 00 00 00 00 00 00 00	FD FD FD 54 90 0D 20 04 3B 00 C8	PRNTYX PRNTAX PRNTX PRBLNK PRBL2 PRBL3 PCADJ2 PCADJ2 PCADJ4 RTS2 * * * * *	INX BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE CLDA LDY TAX BPL DEY ADC BCC INY TAX BPL DEY ADC BCC INY TAX BPL DEY ADC BCC INY TAX BPL DEY ADC BCC INY TAX BPL DEY ADC BCC DFB DFB DFB DFB DFB	PRNTYX PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCADJ4 PCL RTS2 BYTES: =0 =1 \$04,\$20, \$80,\$04, \$54,\$33, \$90,\$04, \$20,\$54, \$20,\$54, \$20,\$54, \$20,\$54, \$20,\$54, \$20,\$54, \$20,\$54, \$20,\$54, \$20,\$54, \$20,\$54, \$20,\$54, \$20,\$54,\$20, \$20,\$54,\$20,\$20,\$20,\$20,\$20,\$20,\$20,\$20,\$20,\$20	<pre>+1 TO Y,X OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK LOOP UNTIL COUNT=0 0=1-BYTE,1=2-BYTE 2=3-BYTE TEST DISPLACEMENT SIGN (FOR REL BRANCH) EXTEND NEG BY DEC PCH PCL+LENGTH(OR DISPL)+1 TO A CARRY INTO Y (PCH) XXXXXY0 INSTRS THEN LEFT HALF BYTE (X=INDEX) \$54,\$30,\$0D \$90,\$03,\$22 \$0D,\$80,\$04 \$20,\$54,\$33 \$04,\$90,\$04 \$3B,\$0D,\$80 \$00,\$22,\$44 \$C8,\$44,\$00</pre>

		~ ~							
F98A:	11	22 0 D	44		DFB	\$11 \$22	¢11 ¢33	\$0D	
F98F:	C8	44	A9		DFB	ŞII, ŞZZ,	,44 <b>,</b> 927,	ŞUD	
F992:	01	22			DFB	\$C8,\$44,	\$A9,\$01,	\$22	
F994:	44	33	0 D						
F997:	80	04	~ ~		DFB	\$44,\$33,	\$0D,\$80,	\$04	
F999:	90 11	33 01	22		חדם	\$90 \$01	¢ 7 7 ¢ 1 1	\$33	
F99E:	0D	80	04		DID	, , , , , , , , , , , , , , , , , , ,	922 <b>,</b> 944,	<b>733</b>	
F9Al:	90				DFB	\$0D,\$80,	\$04,\$90		
F9A2:	26	31	87						
F9A5:	9A				DFB	\$26,\$31,	\$87,\$9A	\$ZZXXXY01	INSTR'S
F9A6:	21			FMT2	DFB	\$00 \$21	ERR		
F9A8:	81				DFB	\$81	Z-PAGE		
F9A9:	82				DFB	\$82	ABS		
F9AA:	00				DFB	\$00	IMPLIED	)	
F9AB:	00				DFB	\$00	ACCUMUL	ATOR	
F9AC: F9AD:	29 4Π				DFB	\$39 \$4D	(ZPAG, A	v	
F9AE:	91				DFB	\$91	ZPAG,X	-	
F9AF:	92				DFB	\$92	ABS,X		
F9B0:	86				DFB	\$86	ABS,Y		
F9BL:	4A 05				DFB	\$4A ¢9E	(ABS)		
F9B3:	00 00				DFB	\$9D	RELATIV	Έ.	
F9B4:	AC .	A9	AC		212	+ 5 2		-	
F9B7:	A3 .	A8	A4						
		~ ~		CHARl	ASC	",),#(\$"			
F9BA:	D9	00	D8	CHAD 2	חשת	¢D0 ¢00	¢ D Q ¢ A /	¢ 7 4 ¢ 0 0	
F9BD:	A4 .	A4	00	*CHAR2	DFB "V" 0	\$D9,\$00, "X\$\$" 0	ŞD8,ŞA4,	\$A4,\$00	
				*	MNEMI	, AÇÇ ,O L	IS OF F	ORM:	
				*	(A) 2	XXXXX000			
				*	(B) 2	XXXYY100			
				*	(C)	1XXX1010			
				*	(D) /	XXXYYYYO]			
				*	(=)	(X=INDEX)			
F9C0:	lC	8A	lC						
F9C3:	23	5D	8B	MNEML	DFB	\$1C,\$8A,	\$1C,\$23,	\$5D <b>,</b> \$	
F9C6:	TR .	AL 1D	9D 23		DFB	\$18 \$A1	۵۹۵ מפא	\$1D \$23	
F9CC:	9D	8B	1D		DID	φid, φiii,	<i>, , , , , , , , , , , , , , , , , , , </i>	φ1D <b>,</b> φ23	
F9CF:	Al	00	29		DFB	\$9D,\$8B,	\$1D,\$A1,	\$00 <b>,</b> \$29	
F9D2:	19	AE	69						
F9D5:	A8	19	23		DFB	\$19,\$AE,	\$69,\$A8,	\$19,\$23	
F9D8:	23	24	1D 53		DFB	\$24.\$53.	\$1B.\$23.	\$24.\$53	
F9DE:	19	Al			DFB	\$19,\$A1	(A) F	ORMAT ABOV	Е
F9E0:	00	lA	5B						
F9E3:	5B	A5	69		DFB	\$00,\$1A,	\$5B,\$5B,	\$A5,\$69	
F9E6: F9E8:	24 AE	24 AE	<b>2</b> 8		DFB	\$24 <b>,</b> \$24	(B) F	ORMAT	
F9EB:	AD	29	00		DFB	\$AE,\$AE,	\$A8,\$AD,	\$29,\$00	
F9EE:	7C	00			DFB	\$7C,\$00	(C) F	ORMAT	
F9F0:	15	9C	6D						
F9F3:	9C.	A5	69		DFB	\$15,\$9C,	\$6D,\$9C,	\$A5,\$69	
F9F8:	84	13	34		Drb	\$29 <b>,</b> \$33	(D) F	ORMAI	
F9FB:	11 .	A5	69		DFB	\$84,\$13,	\$34,\$11,	\$A5,\$69	
F9FE:	23	A0			DFB	\$23,\$A0	(E) F	ORMAT	
FA00:	D8	62	5A			450 460		***	
FA03:	48	26	62 54	MNEMR	DFB	\$D8,\$62,	\$5A,\$48,	\$26,\$62	
FA00:	94 44	00 C8	54		DFB	\$94.\$88.	\$54.\$44.	\$C8,\$54	
FA0C:	68	44	E8		212	+5 1 / + 0 0 / 1	+• - / + /	+00/+01	
FAOF:	94	00	в4		DFB	\$68,\$44,	\$E8 <b>,</b> \$94,	\$00 <b>,</b> \$В4	
FA12:	08	84	74		DED	¢00 ¢04	674 654	\$20 \$CT	
FA19:	В4 74	∠8 F4	оE CC		DEB	<b>₽U8,</b> \$84,	ş/4 <b>,</b> ŞB4,	<b>২∠૪,</b> ३6Ε	
FAlB:	4A	72	F2		DFB	\$74,\$F4,	\$CC,\$4A,	\$72,\$F2	
FAlE:	A4	8A			DFB	\$A4,\$8A	(A) F	ORMAT	
FA20:	00	AA	A2		DE5	600 6	630 6-0	674 67°	
FA23:	A2 71	/4 72	14		DFB DFB	\$UU,\$AA, \$74 \$72	, AZ,ŞA2	\$/4 <b>,</b> \$/4 ΌΡΜΔΨ	
FA28:	, 4 44	, 2 68	в2		DID	Ÿ/± <b>;</b> Ÿ/∠	(D) F	OWINT 1	
FA2B:	32	В2	00		DFB	\$44,\$68,	\$B2 <b>,</b> \$32,	\$B2,\$00	
FA2E:	22	00	<b>.</b> .		DFB	\$22,\$00	(C) F	ORMAT	
FA30:	1A 26		26		מקת	לא לא	606 60C	670 670	
гАЗЗ: FA36•	20 88	72 C8	12		DFR	\$18,\$18, \$88.\$C8	,07ל,07ל שוטיי	ኇ/ረ <b>,</b>	
FA38:	C4	CA	26		210	+001900		~	
FA3B:	48	44	44		DFB	\$C4,\$CA,	\$26 <b>,</b> \$48,	\$44,\$44	
E72E.	72	00			DFB	672 600	(12) 1		

FA40:	FF	FF	FF		DFB	SFF.SFF.SFF	e.
FA43:	20	D0	F8	STEP	JSR	INSTDSP	DISASSEMBLE ONE INST
FA46:	68				PLA		AT (PCL H)
FD47.	85	20			STIA	RTNI.	ADJUST TO USER
FD/Q.	68	20			DLA	RIND	STACK SAVE
	00	20			CUN	DUNI	
FA4A.	70	20			JIA		KIN ADK.
FA4C:	AZ	10	ΠD	VOTNTE			
FA4E:	BD	10	гB	XQINIT	LDA	INITBL-1,X	INIT XEQ AREA
FA51:	95	3C			STA	XQT,X	
FA53:	CA				DEX		
FA54:	D0	F8			BNE	XQINIT	
FA56:	Al	3A			LDA	(PCL,X)	USER OPCODE BYTE
FA58:	F0	42			BEO	XBRK	SPECIAL IF BREAK
FA5A.	Δ4	2 F				LENGTH	LEN FROM DISASSEMBLY
FASC.	C 9	20			CMD	#\$20	
FAJC.		2 U E O			DEO	π920 VICD	HANDLE TOD DEC THD
FASE:	FU	29			BEQ	XJSR I A C A	HANDLE JSR, RTS, JMP,
FA60:	C9	60			СМР	#\$60	JMP (), RTI SPECIAL
FA62:	F0	45			BEQ	XRTS	
FA64:	C9	4C			CMP	#\$4C	
FA66:	F0	5C			BEQ	XJMP	
FA68:	C9	6C			CMP	#\$6C	
FA6A:	F0	59			BEO	XJMPAT	
FA6C:	C9	40			CMP	#\$40	
FA6E.	FO	35			BEO	XBUT	
EN70.	20	15				#¢1₽	
FA70.	10	14			TOD	πφ±r #¢1/	
FA/2:	49	14			EOR	# \$ 14 # \$ 0.4	
FA/4:	C9	04			СМР	#\$04	COPY USER INST TO XEQ AREA
FA76:	F0	02			BEQ	XQ2	WITH TRAILING NOPS
FA78:	Bl	3A		XQl	LDA	(PCL),Y	CHANGE REL BRANCH
FA7A:	99	3C	00	XQ2	STA	XQT,Y	DISP TO 4 FOR
FA7D:	88			~	DEY	~ ·	JMP TO BRANCH OR
FA7E:	10	F8			BPT.	X01	NBRANCH FROM XEO.
E700.	20	35	$\nabla \nabla$		TCD	DECHUDE	DECTODE LIGED DEC CONTENTS
FAOU.	20	20	- F F		JAK	NOT	RESIDRE USER REG CONTENTS.
FA83:	40	30	00		JMP	XQT	XEQ USER OF FROM RAM
FA86:	85	45		IRQ	STA	ACC	(RETURN TO NBRANCH)
FA88:	68				PLA		
FA89:	48				PHA		**IRQ HANDLER
FA8A:	0A				ASL	A	
FA8B:	0A				ASL	A	
FA8C:	0A				ASL	А	
FA8D:	30	03			BMT	BREAK	TEST FOR BREAK
FASE.	60	ਸੂਸ	03		TMD	(TROLOC)	USED DOUTTINE VECTOD IN DAM
FA01.	20	ГĽ	05	DDEAV		(INQLOC)	OBER ROOTINE VECTOR IN RAM
FA92:	20	40		DREAK	TOP	0.117]	ANTE DEGLA ON DDENK
FA93:	20	4C	E.E.		JSR	SAVI	SAVE REG'S ON BREAK
FA96:	68				PLA		INCLUDING PC
FA97:	85	3A			STA	PCL	
FA99:	68				PLA		
FA9A:	85	3B			STA	PCH	
FA9C:	20	82	F8	XBRK	JSR	INSDS1	PRINT USER PC.
FA9F:	2.0	DA	FA		JSR	RGDSP1	AND REG'S
F222.	4 C	65	ਜ਼ਾਜ		TMP	MON	CO TO MONITOR
E772.	10	05		VDWT	CIC	non	GO TO MONITON
FAAJ.	60			XKII			CIMILANE DUI DV EVDECUINC
FAA0:	00	4.0			PLA		SIMULAIE RII BI EXPECTING
FAA/:	85	48			STA	STATUS	STATUS FROM STACK, THEN RTS
FAA9:	68			XRTS	PLA		RTS SIMULATION
FAAA:	85	3A			STA	PCL	EXTRACT PC FROM STACK
FAAC:	68				PLA		AND UPDATE PC BY 1 (LEN=0)
FAAD:	85	3B		PCINC2	STA	PCH	
FAAF:	A5	2F		PCINC3	LDA	LENGTH	UPDATE PC BY LEN
FAB1:	20	56	F9		TSR	DONDTO	
FAR4 .	0 /				001	PCADJ3	
FAR6.	84	3B			STY	PCADJ 3 PCH	
FAR7.	84 18	3B			STY	PCADJ 3 PCH	
FAD/.	84 18 90	3B			STY CLC BCC	PCADJ 3 PCH	
<b>ENDO</b> .	18 90	3B 14		עדפט	STY CLC BCC	PCADJ3 PCH NEWPCL	
FAB9:	84 18 90 18	3B 14		XJSR	STY CLC BCC CLC	PCADJ3 PCH NEWPCL	
FAB9: FABA:	18 90 18 20	3B 14 54	F9	XJSR	STY CLC BCC CLC JSR	PCADJ3 PCH NEWPCL PCADJ2	UPDATE PC AND PUSH
FAB9: FABA: FABD:	84 18 90 18 20 AA	3B 14 54	F9	XJSR	STY CLC BCC CLC JSR TAX	PCADJ3 PCH NEWPCL PCADJ2	UPDATE PC AND PUSH ONTO STACH FOR
FAB9: FABA: FABD: FABE:	18 90 18 20 AA 98	3B 14 54	F9	XJSR	STY CLC BCC CLC JSR TAX TYA	PCADJ3 PCH NEWPCL PCADJ2	UPDATE PC AND PUSH ONTO STACH FOR JSR SIMULATE
FAB9: FABA: FABD: FABE: FABF:	84 18 90 18 20 AA 98 48	3B 14 54	F9	XJSR	STY CLC BCC CLC JSR TAX TYA PHA	PCADJ3 PCH NEWPCL PCADJ2	UPDATE PC AND PUSH ONTO STACH FOR JSR SIMULATE
FAB9: FABA: FABD: FABE: FABF: FAC0:	84 18 90 18 20 AA 98 48 8A	3B 14 54	F9	XJSR	STY CLC BCC CLC JSR TAX TYA PHA TXA	PCADJ3 PCH NEWPCL PCADJ2	UPDATE PC AND PUSH ONTO STACH FOR JSR SIMULATE
FAB9: FABA: FABD: FABE: FABF: FAC0: FAC1:	84 18 90 18 20 AA 98 48 8A 48	3B 14 54	F9	XJSR	STY CLC BCC CLC JSR TAX TYA PHA TXA PHA	PCADJ3 PCH NEWPCL PCADJ2	UPDATE PC AND PUSH ONTO STACH FOR JSR SIMULATE
FAB9: FABA: FABD: FABE: FABF: FAC0: FAC1: FAC2:	84 18 90 18 20 AA 98 48 8A 48 8A 48 A0	3B 14 54	F9	XJSR	STY CLC BCC CLC JSR TAX TYA PHA TXA PHA LDY	PCADJ3 PCH NEWPCL PCADJ2 #\$02	UPDATE PC AND PUSH ONTO STACH FOR JSR SIMULATE
FAB9: FABA: FABD: FABE: FABF: FAC0: FAC1: FAC2: FAC4.	84 18 90 18 20 AA 98 48 8A 48 8A 48 A0 18	3B 14 54 02	F9	XJSR	STY CLC BCC CLC JSR TAX TYA PHA TXA PHA LDY CLC	PCADJ3 PCH NEWPCL PCADJ2 #\$02	UPDATE PC AND PUSH ONTO STACH FOR JSR SIMULATE
FAB9: FABA: FABD: FABE: FABF: FAC0: FAC1: FAC2: FAC4: FAC5:	84 18 90 18 20 AA 98 48 8A 48 A0 18 B1	3B 14 54 02 3A	F9	XJSR XJMP XIMP2T	STY CLC BCC CLC JSR TAX TYA PHA TXA PHA LDY CLC LDA	PCADJ3 PCH NEWPCL PCADJ2 #\$02	UPDATE PC AND PUSH ONTO STACH FOR JSR SIMULATE
FAB9: FABA: FABD: FABE: FABF: FAC0: FAC1: FAC2: FAC4: FAC5:	84 18 90 18 20 AA 98 48 8A 48 8A 48 A0 18 B1	3B 14 54 02 3A	F9	XJSR XJMP XJMPAT	STY CLC BCC CLC JSR TAX TYA PHA TXA PHA LDY CLC LDA	PCADJ3 PCH NEWPCL PCADJ2 #\$02 (PCL),Y	UPDATE PC AND PUSH ONTO STACH FOR JSR SIMULATE
FAB9: FABA: FABD: FABE: FAC0: FAC1: FAC2: FAC4: FAC5: FAC7:	84 18 90 18 20 AA 98 48 8A 48 A0 18 B1 AA	3B 14 54 02 3A	F9	XJSR XJMP XJMPAT	STY STY CLC BCC CLC JSR TAX TYA PHA TXA PHA LDY CLC LDA TAX SEV	<pre>PCADJ3 PCH NEWPCL PCADJ2 #\$02 (PCL),Y</pre>	UPDATE PC AND PUSH ONTO STACH FOR JSR SIMULATE
FAB9: FABA: FABD: FABE: FAC0: FAC1: FAC2: FAC4: FAC5: FAC7: FAC8:	84 18 90 18 20 AA 98 48 8A 48 8A 48 A0 18 B1 AA 88	3B 14 54 02 3A	F9	XJSR XJMP XJMPAT	CLC BCC CLC JSR TAX TYA PHA TXA PHA LDY CLC LDA TAX DEY	<pre>PCADJ3 PCH NEWPCL PCADJ2 #\$02 (PCL),Y</pre>	UPDATE PC AND PUSH ONTO STACH FOR JSR SIMULATE LOAD PC FOR JMP, (JMP) SIMULATE.
FAB9: FABA: FABD: FABE: FAC0: FAC1: FAC2: FAC2: FAC5: FAC5: FAC7: FAC8: FAC9:	84 18 90 18 20 AA 98 48 84 84 80 18 81 AA 881	<ul> <li>3B</li> <li>14</li> <li>54</li> <li>02</li> <li>3A</li> <li>3A</li> </ul>	F9	XJSR XJMP XJMPAT	CLC BCC CLC JSR TAX TYA PHA TXA PHA LDY CLC LDA TAX DEY LDA	<pre>PCADJ3 PCH NEWPCL PCADJ2 #\$02 (PCL),Y (PCL),Y</pre>	UPDATE PC AND PUSH ONTO STACH FOR JSR SIMULATE LOAD PC FOR JMP, (JMP) SIMULATE.
FAB9: FABA: FABD: FABF: FAC0: FAC1: FAC2: FAC4: FAC5: FAC7: FAC8: FAC9: FAC8:	84 18 90 18 20 AA 98 88 48 80 18 81 81 86	3B 14 54 02 3A 3A 3B	F9	XJSR XJMP XJMPAT	STY CLC BCC CLC JSR TAX TYA PHA TXA PHA LDY CLC LDA TAX DEY LDA STX	PCADJ3 PCH NEWPCL PCADJ2 #\$02 (PCL),Y PCH	UPDATE PC AND PUSH ONTO STACH FOR JSR SIMULATE LOAD PC FOR JMP, (JMP) SIMULATE.
FAB9: FABA: FABD: FABF: FAC1: FAC1: FAC2: FAC4: FAC5: FAC5: FAC7: FAC8: FAC9: FAC9: FAC9:	84 18 90 18 20 AA 98 48 8A 48 8A 88 B1 86 85	3B 14 54 02 3A 3B 3A	F9	XJSR XJMP XJMPAT NEWPCL	STY CLC BCC CLC JSR TAX TYA PHA TXA PHA LDY CLC LDA TAX DEY LDA STX STA	<pre>PCADJ3 PCH NEWPCL PCADJ2 #\$02 (PCL),Y (PCL),Y PCH PCL</pre>	UPDATE PC AND PUSH ONTO STACH FOR JSR SIMULATE LOAD PC FOR JMP, (JMP) SIMULATE.
FAB9: FABA: FABD: FABF: FAC0: FAC1: FAC2: FAC2: FAC4: FAC5: FAC7: FAC8: FAC9: FAC9: FAC9: FAC5: FAC7:	84 18 90 18 20 AA 98 48 48 48 48 48 80 18 81 86 85 80	3B 14 54 02 3A 3B 3A 53 F3	F9	XJSR XJMP XJMPAT NEWPCL	CLC BCC CLC JSR TAX TYA PHA TXA PHA LDY CLC LDA TAX DEY LDA STX STA BCS	<pre>PCADJ3 PCH NEWPCL PCADJ2 #\$02 (PCL),Y (PCL),Y PCL XJMP</pre>	UPDATE PC AND PUSH ONTO STACH FOR JSR SIMULATE LOAD PC FOR JMP, (JMP) SIMULATE.
FAB9: FABA: FABD: FABC: FAC0: FAC1: FAC2: FAC2: FAC4: FAC5: FAC7: FAC8: FAC9: FAC9: FACB: FAC1: FAC1: FAC1: FAC1: FAC1: FAC2: FAC2: FAC2: FAC2: FAC3: FAC4: FAC5:	84 18 90 18 20 AA 98 48 48 48 48 80 18 81 88 81 86 85 80 A5	3B 14 54 02 3A 3B 3A 53 2D	F9	XJSR XJMP XJMPAT NEWPCL RTNJMP	CLC BCC CLC JSR TAX TYA PHA TXA PHA LDY CLC LDA TAX DEY LDA STX STA BCS LDA	<pre>PCADJ3 PCH NEWPCL PCADJ2 #\$02 (PCL),Y (PCL),Y PCH PCL XJMP RTNH</pre>	UPDATE PC AND PUSH ONTO STACH FOR JSR SIMULATE LOAD PC FOR JMP, (JMP) SIMULATE.
FAB9: FABA: FABD: FABE: FAC0: FAC0: FAC1: FAC2: FAC4: FAC5: FAC7: FAC8: FAC9: FAC8: FAC5: FAC9:	84 18 90 18 20 AA 98 48 80 88 80 88 80 85 80 A5 48	<ul> <li>3B</li> <li>14</li> <li>54</li> <li>02</li> <li>3A</li> <li>3B</li> <li>3A</li> <li>3B</li> <li>3A</li> <li>F3</li> <li>2D</li> </ul>	F9	XJSR XJMP XJMPAT NEWPCL RTNJMP	STY CLC BCC CLC JSR TAX TYA PHA TXA PHA LDY CLC LDA TAX DEY LDA STX STA BCS LDA PHA	PCADJ3 PCH NEWPCL PCADJ2 #\$02 (PCL),Y (PCL),Y PCH PCL XJMP RTNH	UPDATE PC AND PUSH ONTO STACH FOR JSR SIMULATE LOAD PC FOR JMP, (JMP) SIMULATE.
FAB9: FABA: FABD: FABC1: FAC0: FAC1: FAC2: FAC2: FAC4: FAC5: FAC7: FAC8: FAC9: FAC9: FAC9: FAC9: FAC9: FAC1: FAC1: FAC3: FAC4: FAC4: FAC5:	84 18 90 18 20 A8 98 48 48 48 48 48 81 85 85 85 85 85 85 85 85 85 85	<ul> <li>3B</li> <li>14</li> <li>54</li> <li>02</li> <li>3A</li> <li>3A</li> <li>3B</li> <li>3A</li> <li>3A</li> <li>2D</li> <li>2C</li> </ul>	F9	XJSR XJMP XJMPAT NEWPCL RTNJMP	CLC BCC CLC JSR TAX TYA PHA TXA PHA LDY CLC LDA TAX DEY LDA STX STA BCS LDA PHA LDA	<pre>PCADJ3 PCH NEWPCL PCADJ2 #\$02 (PCL),Y (PCL),Y PCL XJMP RTNH RTNI.</pre>	UPDATE PC AND PUSH ONTO STACH FOR JSR SIMULATE LOAD PC FOR JMP, (JMP) SIMULATE.
FAB9: FABA: FABD: FABF: FAC0: FAC1: FAC2: FAC2: FAC2: FAC5: FAC7: FAC8: FAC9:	84 18 90 18 20 A20 48 48 48 48 48 48 80 18 81 86 85 80 A5 48 48 48 48 48 48 48 48 48 48	<ul> <li>3B</li> <li>14</li> <li>54</li> <li>02</li> <li>3A</li> <li>3A</li> <li>3A</li> <li>3A</li> <li>3A</li> <li>2D</li> <li>2C</li> </ul>	F9	XJSR XJMP XJMPAT NEWPCL RTNJMP	CLC BCC CLC JSR TAX TYA PHA TXA PHA LDY CLC LDA TAX DEY LDA STX STA BCS LDA PHA LDA PHA	PCADJ3 PCH NEWPCL PCADJ2 #\$02 (PCL),Y (PCL),Y PCL XJMP RTNH RTNL	UPDATE PC AND PUSH ONTO STACH FOR JSR SIMULATE LOAD PC FOR JMP, (JMP) SIMULATE.
FAB9: FABA: FABD: FABC: FAC0: FAC1: FAC2: FAC2: FAC4: FAC5: FAC7: FAC8: FAC9: FAC9: FAC1: FAC1: FAC1: FAC3: FAC4: FAC4: FAC5: FAC5: FAC6: FAC7: FAC7: FAC7: FAC6:	84 18 90 18 20 A 98 48 48 48 48 48 48 48 80 48 85 86 85 85 85 85 85 85 85 85 85 85	3B 14 54 02 3A 3B 3A 53 2D 2C	F9	XJSR XJMP XJMPAT NEWPCL RTNJMP	CLC BCC CLC JSR TAX PHA TXA PHA LDY CLC LDA TAX DEY LDA STX STA BCS LDA PHA LDA PHA LDA STSP	PCADJ3 PCH NEWPCL PCADJ2 #\$02 (PCL),Y (PCL),Y PCH PCL XJMP RTNH RTNL CPOUT	UPDATE PC AND PUSH ONTO STACH FOR JSR SIMULATE LOAD PC FOR JMP, (JMP) SIMULATE.
FAB9: FABA: FABD: FABC: FAC0: FAC1: FAC2: FAC4: FAC4: FAC5: FAC7: FAC8: FAC9: FAC8: FAC9: FAC9: FAC9: FAC1: FAC1: FAC1: FAC2: FAC2: FAC3: FAC4: FAC4: FAC4: FAC4: FAC5: FAC4: FAC4: FAC5: FAC4: FAC5: FAC4: FAC4: FAC5: FAC4: FAC4: FAC4: FAC5: FAC4: FAC4: FAC4: FAC4: FAC5: FAC4:	84 18 918 20 AA 98 48 48 48 48 48 48 81 85 48 48 48 48 81 85 48 48 48 48 48 48 48 48 48 48	3B 14 54 02 3A 3B 3A 53 2D 2C 8E	F9	XJSR XJMP XJMPAT NEWPCL RTNJMP	STY CLC BCC JSR TAX TYA PHA TXA PHA LDY CLC LDA TAX DEY LDA STX STA BCS LDA PHA LDA PHA LDA PHA LDA PHA LDA	PCADJ3 PCH NEWPCL PCADJ2 #\$02 (PCL),Y (PCL),Y PCH PCL XJMP RTNH RTNL CROUT	UPDATE PC AND PUSH ONTO STACH FOR JSR SIMULATE LOAD PC FOR JMP, (JMP) SIMULATE.
FAB9: FABA: FABA: FABC0: FAC1: FAC2: FAC2: FAC2: FAC4: FAC5: FAC7: FAC8: FAC9: FAC8: FAC9: FAC9: FAC9: FAC9: FAC9: FAC1: FAC1: FAC3: FAC4: FAC5:	84 18 90 20 20 20 20 20 20 20 20 20 2	3B 14 54 02 3A 3B 3A 3B 3A 53 2D 2C 8E 45	F9	XJSR XJMP XJMPAT NEWPCL RTNJMP REGDSP RGDSP1	STY CLC BCC CLC JSR TAX TYA PHA TXA PHA LDY CLC LDA TAX DEY LDA STX STA BCS LDA PHA LDA PHA LDA PHA	<pre>PCADJ3 PCH NEWPCL PCADJ2 #\$02 (PCL),Y (PCL),Y (PCL),Y PCL XJMP RTNH RTNL CROUT #ACC PCA PCL PCL PCL PCL PCL PCL PCL PCL PCL PCL</pre>	UPDATE PC AND PUSH ONTO STACH FOR JSR SIMULATE LOAD PC FOR JMP, (JMP) SIMULATE. DISPLAY USER REG CONTENTS WITH
FAB9: FABA: FABA: FABA: FABF: FAC7: FAC1: FAC2: FAC2: FAC4: FAC5: FAC7: FAC8: FAC9: FAC9: FAC9: FAC9: FAC1:	84 18 90 20 20 20 20 20 20 20 20 20 20 20 20 20	<ul> <li>3B</li> <li>14</li> <li>54</li> <li>02</li> <li>3A</li> <li>3B</li> <li>3A</li> <li>3A</li> <li>3B</li> <li>3A</li> <li>3A</li> <li>3B</li> <li>3A</li> &lt;</ul>	F9	XJSR XJMP XJMPAT NEWPCL RTNJMP REGDSP RGDSP1	STY CLC BCC CLC JSR TAX TYA PHA TXA PHA TXA PHA LDY CLC LDA TAX DEY LDA STX STA BCS LDA PHA JSR LDA STA	<pre>PCADJ3 PCH NEWPCL PCADJ2 #\$02 (PCL),Y (PCL),Y (PCL),Y PCL XJMP RTNH RTNL CROUT #ACC A3L CT</pre>	UPDATE PC AND PUSH ONTO STACH FOR JSR SIMULATE LOAD PC FOR JMP, (JMP) SIMULATE. DISPLAY USER REG CONTENTS WITH LABELS

FADE:	A9	00			LDA	#ACC/256	
FAE0:	85	41			STA	АЗН	
FAE2:	A2	FB			LDX	#\$FB	
FAE4:	A9	A0	ΠD	RDSPI	LDA	#\$A0	
FAE0: FAE0:	20 80	5D 15	FD FA		JSK	COUT PTRL_SFR Y	
FAEC:	2.0	ED	FD		JSR	COUT	
FAEF:	A9	BD			LDA	#\$BD	
FAF1:	20	ED	FD		JSR	COUT	
FAF4:	В5	4A			LDA	ACC+5,X	
FAF6:	20	DA	FD		JSR	PRBYTE	
FAF9:	E8				INX		
FAFA:	30	E8			BMI	RDSPl	
FAFC:	60				RTS		
FAFD:	18	0.1		BRANCH	CLC	400	BRANCH TAKEN,
FAFE:	AU Dl	0T			LDY	#\$01 (DOI) V	ADD LEN+2 TO PC
FBUU:	DT BT	JA EC	ΨO		LDA	(PCL),I	
FB02:	20	32	гэ		STA	PCADJ 5	
FB07:	98	511			ΤΥΑ	101	
FB08:	38				SEC		
FB09:	в0	A2			BCS	PCINC2	
FB0B:	20	4A	FF	NBRNCH	JSR	SAVE	NORMAL RETURN AFTER
FB0E:	38				SEC		XEQ USER OF
FB0F:	в0	9E			BCS	PCINC3	GO UPDATE PC
FBll:	ΕA			INITBL	NOP		
FB12:	ΕA				NOP		DUMMY FILL FOR
FB13:	4C	0B	FB		JMP	NBRNCH	XEQ AREA
FB16:	4C	FD	FA		JMP	BRANCH	
FB19:	C1			RTBL	DFB	\$C1	
FBLA:	D8				DFB	ŞD8	
FBIB:	D9				DFB	\$D9	
FBIC:	DO				DFB	\$D0	
FBID:	D3	70	<b>a</b> 0		DFB	ŞD3 DEDIC	MDICCED DADDIEC
FBLE:	AD	/0	CU	PREAD		PTRIG #¢00	TRIGGER PADDLES
FDZI:	HU EV	00			NOD	#\$00	COMDENSATE FOR 1ST COUNT
FB21.	ΕA				NOP		COMPENSATE FOR 151 COUNT
FB25.	BD	64	CO	DREAD2		Χ Δ.ΤΠΠΑΥ	COUNT Y-REG EVERY
FB28.	10	04	CU	FREADZ	BDA BDT.	RTS2D	12 USEC
FB2A:	C 8	04			TNY	KI52D	12 0510
FB2B:	D0	F8			BNE	PREAD2	EXIT AT 255 MAX
FB2D:	88				DEY		
	~ ~				221		
FBZE:	60			RTSZD	RTS		
FB2E: FB2F:	60 A9	00		RTS2D INIT	RTS LDA	#\$00	CLR STATUS FOR DEBUG
FB2E: FB2F: FB31:	60 A9 85	00 48		RTS2D INIT	RTS LDA STA	#\$00 STATUS	CLR STATUS FOR DEBUG SOFTWARE
FB2E: FB2F: FB31: FB33:	60 A9 85 AD	00 48 56	C0	INIT	RTS LDA STA LDA	#\$00 STATUS LORES	CLR STATUS FOR DEBUG SOFTWARE
FB2E: FB2F: FB31: FB33: FB36:	60 A9 85 AD AD	00 48 56 54	C0 C0	RTS2D INIT	RTS LDA STA LDA LDA	#\$00 STATUS LORES LOWSCR	CLR STATUS FOR DEBUG SOFTWARE INIT VIDEO MODE
FB2E: FB2F: FB31: FB33: FB36: FB39:	60 A9 85 AD AD AD	00 48 56 54 51	C0 C0 C0	RTS2D INIT SETTXT	RTS LDA STA LDA LDA LDA	#\$00 STATUS LORES LOWSCR TXTSET	CLR STATUS FOR DEBUG SOFTWARE INIT VIDEO MODE SET FOR TEXT MODE
FB2E: FB2F: FB31: FB33: FB36: FB39: FB3C:	60 A9 85 AD AD AD A9	00 48 56 54 51 00	C0 C0 C0	RTS2D INIT SETTXT	RTS LDA STA LDA LDA LDA LDA	#\$00 STATUS LORES LOWSCR TXTSET #\$00	CLR STATUS FOR DEBUG SOFTWARE INIT VIDEO MODE SET FOR TEXT MODE FULL SCREEN WINDOW
FB2E: FB2F: FB31: FB33: FB36: FB39: FB3C: FB3E:	60 A9 85 AD AD AD AD F0	00 48 56 54 51 00 0B	C0 C0 C0	RTS2D INIT SETTXT	RTS LDA STA LDA LDA LDA LDA BEQ	#\$00 STATUS LORES LOWSCR TXTSET #\$00 SETWND	CLR STATUS FOR DEBUG SOFTWARE INIT VIDEO MODE SET FOR TEXT MODE FULL SCREEN WINDOW
FB2E: FB2F: FB31: FB33: FB36: FB39: FB3C: FB3E: FB40:	60 A9 85 AD AD AD A9 F0 AD	00 48 56 54 51 00 0B 50	C0 C0 C0	RTS2D INIT SETTXT SETGR	RTS LDA STA LDA LDA LDA BEQ LDA	#\$00 STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR	CLR STATUS FOR DEBUG SOFTWARE INIT VIDEO MODE SET FOR TEXT MODE FULL SCREEN WINDOW SET FOR GRAPHICS MODE
FB2E: FB2F: FB31: FB33: FB36: FB39: FB3C: FB3E: FB40: FB43:	60 A9 85 AD AD AD AD F0 AD AD	00 48 56 51 00 0B 50 53	C0 C0 C0 C0	RTS2D INIT SETTXT SETGR	RTS LDA STA LDA LDA LDA BEQ LDA LDA	#\$00 STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET	CLR STATUS FOR DEBUG SOFTWARE INIT VIDEO MODE SET FOR TEXT MODE FULL SCREEN WINDOW SET FOR GRAPHICS MODE LOWER 4 LINES AS
FB2E: FB2F: FB31: FB33: FB36: FB39: FB3C: FB3E: FB40: FB43: FB46:	60 A9 85 AD AD AD AD F0 AD 20	00 48 56 54 51 00 08 50 53 36	C0 C0 C0 C0 F8	RTS2D INIT SETTXT SETGR	RTS LDA STA LDA LDA LDA BEQ LDA LDA JSR	#\$00 STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP	CLR STATUS FOR DEBUG SOFTWARE INIT VIDEO MODE SET FOR TEXT MODE FULL SCREEN WINDOW SET FOR GRAPHICS MODE LOWER 4 LINES AS TEXT WINDOW
FB2E: FB2F: FB31: FB36: FB36: FB3C: FB3C: FB40: FB43: FB46: FB49:	60 A9 85 AD AD AD AD F0 AD 20 A9	00 48 56 51 00 50 50 50 36 14	C0 C0 C0 C0 F8	RTS2D INIT SETTXT SETGR	RTS LDA STA LDA LDA LDA LDA LDA LDA JSR LDA	#\$00 STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14	CLR STATUS FOR DEBUG SOFTWARE INIT VIDEO MODE SET FOR TEXT MODE FULL SCREEN WINDOW SET FOR GRAPHICS MODE LOWER 4 LINES AS TEXT WINDOW
FB2E: FB2F: FB31: FB36: FB36: FB3C: FB3E: FB40: FB43: FB46: FB49: FB4B:	60 A9 85 AD AD AD AD AD 20 AD 20 A9 85	00 48 56 51 00 53 36 14 22	C0 C0 C0 C0 F8	RTS2D INIT SETTXT SETGR SETWND	RTS LDA STA LDA LDA LDA BEQ LDA LDA JSR LDA STA	#\$00 STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP	CLR STATUS FOR DEBUG SOFTWARE INIT VIDEO MODE SET FOR TEXT MODE FULL SCREEN WINDOW SET FOR GRAPHICS MODE LOWER 4 LINES AS TEXT WINDOW SET FOR 40 COL WINDOW
FB2E: FB2F: FB31: FB36: FB36: FB39: FB36: FB39: FB40: FB40: FB46: FB49: FB4B: FB4D: FB42F:	60 A9 85 AD AD AD AD AD 20 AD 20 A9 85 A9	00 48 56 54 51 00 50 53 36 14 22 00 20	C0 C0 C0 C0 F8	RTS2D INIT SETTXT SETGR SETWND	RTS LDA STA LDA LDA LDA LDA BEQ LDA LDA STA LDA STA	#\$00 STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$14	CLR STATUS FOR DEBUG SOFTWARE INIT VIDEO MODE SET FOR TEXT MODE FULL SCREEN WINDOW SET FOR GRAPHICS MODE LOWER 4 LINES AS TEXT WINDOW SET FOR 40 COL WINDOW TOP IN A-REG,
FB2E: FB2F: FB31: FB36: FB39: FB36: FB39: FB36: FB40: FB40: FB44: FB44: FB4B: FB4D: FB4F:	60 A9 85 AD AD AD AD AD AD AD AD AD 85 A9 85 85	00 48 56 51 00 50 53 36 14 22 00 20 28	C0 C0 C0 C0 F8	RTS2D INIT SETTXT SETGR SETWND	RTS LDA STA LDA LDA LDA LDA LDA LDA JSR LDA STA LDA	<pre>#\$00 STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$14 WNDTOP #\$28</pre>	CLR STATUS FOR DEBUG SOFTWARE INIT VIDEO MODE SET FOR TEXT MODE FULL SCREEN WINDOW SET FOR GRAPHICS MODE LOWER 4 LINES AS TEXT WINDOW SET FOR 40 COL WINDOW TOP IN A-REG, BTTM AT LINE 24
FB2E: FB2F: FB31: FB36: FB39: FB36: FB39: FB30: FB30: FB40: FB43: FB440: FB450: FB50: FB50: FB50: FB450: FB	60 A9 85 AD AD AD AD AD 20 AD 20 85 A9 85 85 85	00 48 56 54 50 00 53 36 14 22 00 20 28 21	C0 C0 C0 C0 F8	RTS2D INIT SETTXT SETGR SETWND	RTS LDA STA LDA LDA LDA LDA LDA LDA STA LDA STA LDA STA	<pre>#\$00 STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$00 WNDLFT #\$28 WNDWDTH</pre>	CLR STATUS FOR DEBUG SOFTWARE INIT VIDEO MODE SET FOR TEXT MODE FULL SCREEN WINDOW SET FOR GRAPHICS MODE LOWER 4 LINES AS TEXT WINDOW SET FOR 40 COL WINDOW TOP IN A-REG, BTTM AT LINE 24
FB2E: FB2E: FB31: FB33: FB36: FB39: FB39: FB39: FB39: FB39: FB40: FB443: FB449: FB459:	60 A9 85 AD AD AD AD AD 20 A9 85 A9 85 A9 85	00 48 56 54 51 00 53 36 14 22 00 20 28 21	C0 C0 C0 C0 F8	RTS2D INIT SETTXT SETGR SETWND	RTS LDA STA LDA LDA LDA LDA LDA LDA STA LDA STA LDA STA LDA	<pre>#\$00 STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$00 WNDLFT #\$28 WNDWDTH #\$18</pre>	CLR STATUS FOR DEBUG SOFTWARE INIT VIDEO MODE SET FOR TEXT MODE FULL SCREEN WINDOW SET FOR GRAPHICS MODE LOWER 4 LINES AS TEXT WINDOW SET FOR 40 COL WINDOW TOP IN A-REG, BTTM AT LINE 24
FB2E: FB2E: FB31: FB33: FB36: FB39: FB39: FB39: FB39: FB40: FB449: FB449: FB449: FB449: FB449: FB447: FB51: FB551: FB557:	60 A9 85 AD AD AD AD AD AD AD AD 85 A9 85 A9 85 85	00 48 56 54 51 00 53 36 14 22 00 28 21 18 23	C0 C0 C0 C0 F8	RTS2D INIT SETTXT SETGR SETWND	RTS LDA STA LDA LDA LDA LDA LDA LDA STA LDA STA LDA STA LDA STA	<pre>#\$00 STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$14 WNDLFT #\$28 WNDWDTH #\$18 WNDBTM</pre>	CLR STATUS FOR DEBUG SOFTWARE INIT VIDEO MODE SET FOR TEXT MODE FULL SCREEN WINDOW SET FOR GRAPHICS MODE LOWER 4 LINES AS TEXT WINDOW SET FOR 40 COL WINDOW TOP IN A-REG, BTTM AT LINE 24
FB2E: FB2E: FB33: FB36: FB39: FB39: FB39: FB39: FB39: FB40: FB40: FB49: FB49: FB49: FB49: FB49: FB49: FB51: FB55: FB55: FB55: FB55:	60 A9 85 AD AD AD AD AD AD AD AD AD AD AD 85 A9 85 A9 85 A9 85 A9	00 48 56 54 51 00 50 53 36 14 22 00 28 21 18 23 17	C0 C0 C0 C0 F8	RTS2D INIT SETTXT SETGR SETWND	RTS LDA STA LDA LDA LDA LDA LDA LDA LDA STA LDA STA LDA STA LDA LDA	<pre>#\$00 STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$00 WNDLFT #\$28 WNDWDTH #\$18 WNDBTM #\$17</pre>	CLR STATUS FOR DEBUG SOFTWARE INIT VIDEO MODE SET FOR TEXT MODE FULL SCREEN WINDOW SET FOR GRAPHICS MODE LOWER 4 LINES AS TEXT WINDOW SET FOR 40 COL WINDOW TOP IN A-REG, BTTM AT LINE 24 VTAB TO ROW 23
FB2E: FB2E: FB33: FB36: FB39: FB39: FB39: FB39: FB32: FB40: FB49: FB49: FB48: FB48: FB48: FB48: FB48: FB51: FB55: FB55: FB55: FB55: FB55:	60 A9 85 AD AD AD AD AD AD AD AD AD AD AD 85 A9 85 A9 85 A9 85 85 85	00 48 56 51 00 50 50 36 14 22 00 28 21 18 23 17 25	C0 C0 C0 F8	RTS2D INIT SETTXT SETGR SETWND	RTS LDA STA LDA LDA LDA LDA LDA LDA LDA STA LDA STA LDA STA LDA STA	<pre>#\$00 STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDDTOP #\$00 WNDLFT #\$28 WNDWDTH #\$18 WNDBTM #\$17 CV</pre>	CLR STATUS FOR DEBUG SOFTWARE INIT VIDEO MODE SET FOR TEXT MODE FULL SCREEN WINDOW SET FOR GRAPHICS MODE LOWER 4 LINES AS TEXT WINDOW SET FOR 40 COL WINDOW TOP IN A-REG, BTTM AT LINE 24 VTAB TO ROW 23 VTABS TO ROW IN A-REG
FB2E: FB2F: FB33: FB36: FB39: FB39: FB37: FB40: FB40: FB440: FB440: FB440: FB440: FB440: FB51: FB551: FB551: FB550: FB550:	60 A9 85 AD AD AD AD AD AD AD AD AD AD AD AD 85 A9 85 A9 85 A9 85 40 85 40 85 A9 85 A9 85 A0 85 AD AD AD AD AD AD AD AD AD AD AD AD AD	00 48 56 51 00 50 50 20 20 20 21 18 23 17 25 22	C0 C0 C0 F8 FC	RTS2D INIT SETTXT SETGR SETWND	RTS LDA STA LDA LDA LDA LDA LDA LDA LDA STA LDA STA LDA STA LDA STA STA JMP	<pre>#\$00 STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$14 WNDTOP #\$28 WNDULFT #\$28 WNDWDTH #\$18 WNDBTM #\$17 CV VTAB</pre>	CLR STATUS FOR DEBUG SOFTWARE INIT VIDEO MODE SET FOR TEXT MODE FULL SCREEN WINDOW SET FOR GRAPHICS MODE LOWER 4 LINES AS TEXT WINDOW SET FOR 40 COL WINDOW TOP IN A-REG, BTTM AT LINE 24 VTAB TO ROW 23 VTABS TO ROW IN A-REG
FB2E: FB2E: FB33: FB36: FB39: FB36: FB39: FB30: FB30: FB40: FB40: FB440: FB440: FB440: FB440: FB440: FB551: FB551: FB550: FB550: FB500:	60 A9 85 AD AD AD AD AD AD AD AD AD AD AD AD AD	000 488 566 544 5100 0B 503 366 200 200 202 218 233 17 255 222 A4	C0 C0 C0 F8 F7 F7	RTS2D INIT SETTXT SETGR SETWND TABV MULPM	RTS LDA STA LDA LDA LDA LDA LDA LDA LDA LDA STA LDA STA LDA STA LDA STA LDA STA JMP JSR	<pre>#\$00 STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$14 WNDTOP #\$28 WNDULFT #\$28 WNDWDTH #\$18 WNDBTM #\$17 CV VTAB MD1</pre>	CLR STATUS FOR DEBUG SOFTWARE INIT VIDEO MODE SET FOR TEXT MODE FULL SCREEN WINDOW SET FOR GRAPHICS MODE LOWER 4 LINES AS TEXT WINDOW SET FOR 40 COL WINDOW TOP IN A-REG, BTTM AT LINE 24 VTAB TO ROW 23 VTABS TO ROW IN A-REG ABS VAL OF AC AUX
FB2E: FB2E: FB31: FB33: FB36: FB37: FB36: FB37: FB46: FB49: FB49: FB49: FB49: FB49: FB49: FB49: FB55: FB57: FB59: FB59: FB59: FB50:	60 A9 85 AD AD AD AD AD AD AD AD AD AD AD AD 85 A9 85 A9 85 A9 85 A9 85 A9 85 A0 85 AD AD AD AD AD AD AD AD AD AD AD AD AD	000 488 566 544 5100 0B 503 366 142 200 200 200 202 218 233 17 255 222 A410	C0 C0 C0 F8 FC FB	RTS2D INIT SETTXT SETGR SETWND TABV MULPM MUL	RTS LDA STA LDA LDA LDA LDA LDA LDA LDA STA LDA STA LDA STA LDA STA LDA STA LDA LDA LDA LDA LDA LDA LDA LDA LDA LD	<pre>#\$00 STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$14 WNDLFT #\$28 WNDWDTH #\$18 WNDBTM #\$17 CV VTAB MD1 #\$10</pre>	CLR STATUS FOR DEBUG SOFTWARE INIT VIDEO MODE SET FOR TEXT MODE FULL SCREEN WINDOW SET FOR GRAPHICS MODE LOWER 4 LINES AS TEXT WINDOW SET FOR 40 COL WINDOW TOP IN A-REG, BTTM AT LINE 24 VTAB TO ROW 23 VTABS TO ROW IN A-REG ABS VAL OF AC AUX INDEX FOR 16 BITS
FB2E: FB2E: FB2E: FB33: FB36: FB39: FB39: FB39: FB39: FB39: FB40: FB49: FB49: FB49: FB48: FB48: FB55: FB57: FB55: FB55: FB55: FB55: FB50:	60 A9 85 AD AD AD AD AD AD AD AD AD AD AD AD 85 A9 85 A9 85 A9 85 A9 85 A0 85 A0 85 AD AD AD AD AD AD AD AD AD AD AD AD AD	000 488 56 54 500 00B 500 500 200 200 200 200 200 200 200 201 183 217 252 244 100 500	C0 C0 C0 F8 FC FB	RTS2D INIT SETTXT SETGR SETWND TABV MULPM MUL MUL2	RTS LDA STA LDA LDA LDA LDA LDA LDA LDA STA LDA STA LDA STA STA STA LDA LDA LDA LDA LDA LDA LDA LDA LDA LD	<pre>#\$00 STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$14 WNDUTOP #\$10 WNDLFT #\$18 WNDWDTH #\$17 CV VTAB MD1 #\$10 ACL</pre>	CLR STATUS FOR DEBUG SOFTWARE INIT VIDEO MODE SET FOR TEXT MODE FULL SCREEN WINDOW SET FOR GRAPHICS MODE LOWER 4 LINES AS TEXT WINDOW SET FOR 40 COL WINDOW TOP IN A-REG, BTTM AT LINE 24 VTAB TO ROW 23 VTABS TO ROW IN A-REG ABS VAL OF AC AUX INDEX FOR 16 BITS ACX * AUX + XTND
FB2E: FB2E: FB2E: FB33: FB36: FB39: FB39: FB39: FB39: FB39: FB40: FB49: FB49: FB40: FB44: FB44: FB51: FB55: FB55: FB55: FB55: FB55: FB50: FB60: FB60: FB65: FB65: FB65: FB67:	60 A9 85 AD AD AD AD AD AD AD AD AD AD AD AD AD	000 488 56 54 500 00B 500 500 200 200 200 200 200 200 200 200	C0 C0 C0 F8 FC FB	RTS2D INIT SETTXT SETGR SETWND TABV MULPM MUL MUL2	RTS LDA STA LDA LDA LDA LDA LDA LDA LDA LDA LDA LD	#\$00 STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$14 WNDTOP #\$14 WNDLFT #\$28 WNDWDTH #\$18 WNDBTM #\$17 CV VTAB MD1 #\$10 ACL A	CLR STATUS FOR DEBUG SOFTWARE INIT VIDEO MODE SET FOR TEXT MODE FULL SCREEN WINDOW SET FOR GRAPHICS MODE LOWER 4 LINES AS TEXT WINDOW SET FOR 40 COL WINDOW TOP IN A-REG, BTTM AT LINE 24 VTAB TO ROW 23 VTABS TO ROW IN A-REG ABS VAL OF AC AUX INDEX FOR 16 BITS ACX * AUX + XTND TO AC, XTND
FB2E: FB2E: FB31: FB33: FB36: FB39: FB39: FB39: FB39: FB40: FB49: FB49: FB49: FB49: FB49: FB49: FB49: FB49: FB51: FB55: FB55: FB55: FB55: FB55: FB50: FB60: FB66: FB60: FB60: FB50: FB50: FB50: FB50: FB50: FB50: FB50: FB50: FB40: FB50: FB60:	600 A99 8DD ADD ADD ADD ADD ADD ADD ADD ADD ADD	000 486 54 51 000 53 366 14 220 200 228 21 182 23 17 252 24 10 50 0C	C0 C0 C0 F8 FC FB	RTS2D INIT SETTXT SETGR SETWND TABV MULPM MUL MUL2	RTS LDA STA LDA LDA LDA LDA LDA LDA LDA LDA LDA LD	<pre>#\$00 STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$14 WNDLFT #\$28 WNDWDTH #\$18 WNDBTM #\$17 CV VTAB MD1 #\$10 ACL A MUL4</pre>	CLR STATUS FOR DEBUG SOFTWARE INIT VIDEO MODE SET FOR TEXT MODE FULL SCREEN WINDOW SET FOR GRAPHICS MODE LOWER 4 LINES AS TEXT WINDOW SET FOR 40 COL WINDOW TOP IN A-REG, BTTM AT LINE 24 VTAB TO ROW 23 VTABS TO ROW IN A-REG ABS VAL OF AC AUX INDEX FOR 16 BITS ACX * AUX + XTND TO AC, XTND IF NO CARRY,
FB2E: FB2E: FB36: FB36: FB39: FB36: FB39: FB36: FB39: FB32: FB40: FB40: FB40: FB40: FB40: FB55: FB55: FB55: FB55: FB55: FB55: FB55: FB56: FB66:	60 A9 85 AD AD AD AD AD AD AD AD AD AD AD AD AD	000 486 54 51 000 53 366 14 220 020 2821 1823 177 255 224 A40 50 0C	C0 C0 C0 F8 FC FB	RTS2D INIT SETTXT SETGR SETWND TABV MULPM MUL MUL2	RTS LDA STA LDA LDA LDA LDA LDA LDA LDA LDA LDA LD	<pre>#\$00 \$STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$00 WNDLFT #\$28 WNDWDTH #\$18 WNDBTM #\$17 CV VTAB MD1 #\$10 ACL A MUL4 "************************************</pre>	CLR STATUS FOR DEBUG SOFTWARE INIT VIDEO MODE SET FOR TEXT MODE FULL SCREEN WINDOW SET FOR GRAPHICS MODE LOWER 4 LINES AS TEXT WINDOW SET FOR 40 COL WINDOW TOP IN A-REG, BTTM AT LINE 24 VTAB TO ROW 23 VTABS TO ROW IN A-REG ABS VAL OF AC AUX INDEX FOR 16 BITS ACX * AUX + XTND TO AC, XTND IF NO CARRY, NO PARTIAL PROD.
FB2E: FB2E: FB32: FB33: FB36: FB39: FB39: FB32: FB32: FB40: FB40: FB48: FB48: FB48: FB55: FB55: FB55: FB55: FB55: FB55: FB55: FB55: FB56: FB65: FB65: FB66: FB68: FB68: FB68: FB68:	600 A95 ADD ADD ADD ADD ADD ADD ADD ADD ADD AD	000 48 56 54 51 000 200 200 200 200 200 200 200 200 20	C0 C0 C0 F8 FC FB	RTS2D INIT SETTXT SETGR SETWND TABV MULPM MUL MUL2	RTS LDA STA LDA LDA LDA LDA LDA LDA LDA JSR LDA STA LDA STA LDA STA LDA STA LDA LDA LDA LDA LDA LDA LDA LDA LDA LD	<pre>#\$00 STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$14 WNDTOP #\$28 WNDWDTH #\$18 WNDBTM #\$17 CV VTAB MD1 #\$10 ACL A MUL4 #\$FE</pre>	CLR STATUS FOR DEBUG SOFTWARE INIT VIDEO MODE SET FOR TEXT MODE FULL SCREEN WINDOW SET FOR GRAPHICS MODE LOWER 4 LINES AS TEXT WINDOW SET FOR 40 COL WINDOW TOP IN A-REG, BTTM AT LINE 24 VTAB TO ROW 23 VTABS TO ROW IN A-REG ABS VAL OF AC AUX INDEX FOR 16 BITS ACX * AUX + XTND TO AC, XTND IF NO CARRY, NO PARTIAL PROD.
FB2E: FB2E: FB31: FB33: FB36: FB37: FB36: FB40: FB49: FB49: FB49: FB49: FB49: FB49: FB49: FB551: FB551: FB551: FB552: FB550: FB563: FB652: FB663: FB664: FB675: FB6	600 A95 ADD ADD ADD ADD ADD ADD ADD ADD ADD AD	000485654510000533642200028821183217522244100500CFE45	C0 C0 C0 F8 FC FB	RTS2D INIT SETTXT SETGR SETWND TABV MULPM MUL MUL2 MUL3	RTS LDA STA LDA LDA LDA LDA LDA LDA LDA LDA STA LDA STA LDA STA LDA STA LDA LDA LDA LDA LDA LDA LDA LDA LDA LD	<pre>#\$00 STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDDTOP #\$00 WNDLFT #\$28 WNDWDTH #\$18 WNDBTM #\$17 CV VTAB MD1 #\$10 ACL A MUL4 #\$FE XTNDL+2,X</pre>	CLR STATUS FOR DEBUG SOFTWARE INIT VIDEO MODE SET FOR TEXT MODE FULL SCREEN WINDOW SET FOR GRAPHICS MODE LOWER 4 LINES AS TEXT WINDOW SET FOR 40 COL WINDOW TOP IN A-REG, BTTM AT LINE 24 VTAB TO ROW 23 VTABS TO ROW IN A-REG ABS VAL OF AC AUX INDEX FOR 16 BITS ACX * AUX + XTND TO AC, XTND IF NO CARRY, NO PARTIAL PROD.
FB2E: FB2E: FB2E: FB31: FB33: FB36: FB39: FB39: FB39: FB39: FB40: FB49: FB49: FB49: FB49: FB49: FB49: FB49: FB55: FB57: FB55: FB57: FB59: FB50: FB50: FB63: FB66: FB60: FB66:	600 A95 ADD ADD ADD ADD ADD ADD ADD ADD ADD AD	008 48 56 51 000 53 36 14 22 000 208 21 82 31 7 52 22 41 0 50 0 54 50 50 50 50 50 50 50 50 50 50 50 50 50	C0 C0 C0 F8 FC FB	RTS2D INIT SETTXT SETGR SETWND TABV MULPM MULPM MUL2 MUL3	RTS LDA STA LDA LDA LDA LDA LDA LDA LDA LDA STA LDA STA LDA STA LDA STA LDA LDA LDA LDA LDA LDA LDA LDA LDA LD	<pre>#\$00 STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDDTOP #\$00 WNDLFT #\$28 WNDWDTH #\$17 CV VTAB MD1 #\$10 ACL A MUL4 #\$FE XTNDL+2,X AUXL+2,X</pre>	CLR STATUS FOR DEBUG SOFTWARE INIT VIDEO MODE SET FOR TEXT MODE FULL SCREEN WINDOW SET FOR GRAPHICS MODE LOWER 4 LINES AS TEXT WINDOW SET FOR 40 COL WINDOW TOP IN A-REG, BTTM AT LINE 24 VTAB TO ROW 23 VTABS TO ROW IN A-REG ABS VAL OF AC AUX INDEX FOR 16 BITS ACX * AUX + XTND TO AC, XTND IF NO CARRY, NO PARTIAL PROD.
FB2E: FB2E: FB2E: FB33: FB36: FB39: FB39: FB39: FB39: FB39: FB40: FB49: FB49: FB49: FB49: FB49: FB49: FB49: FB49: FB55: FB57: FB55: FB57: FB55: FB57: FB50: FB56: FB66: FB66: FB66: FB66: FB66: FB66: FB66: FB67: FB66: FB66: FB66: FB66: FB66: FB67: FB66: FB60: FB66:	600 A95 ADD ADD ADD ADD ADD ADD ADD ADD ADD AD	00 456 54 50 00 50 53 36 14 220 20 228 21 18 33 17 252 24 00 50 50 50 53 6 14 220 20 28 21 18 33 17 25 22 40 50 50 50 50 50 50 50 50 50 50 50 50 50	C0 C0 C0 F8 FC FB	RTS2D INIT SETTXT SETGR SETWND TABV MULPM MUL2 MUL2	RTS LDA STA LDA LDA LDA LDA LDA LDA LDA LDA STA LDA STA STA STA LDA LDA LDA LDA LDA LDA LDA LDA LDA LD	<pre>#\$00 STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$14 WNDUTP #\$28 WNDWDTH #\$18 WNDWDTH #\$17 CV VTAB MD1 #\$10 ACL A MUL4 #\$FE XTNDL+2,X XTNDL+2,X</pre>	CLR STATUS FOR DEBUG SOFTWARE INIT VIDEO MODE SET FOR TEXT MODE FULL SCREEN WINDOW SET FOR GRAPHICS MODE LOWER 4 LINES AS TEXT WINDOW SET FOR 40 COL WINDOW TOP IN A-REG, BTTM AT LINE 24 VTAB TO ROW 23 VTABS TO ROW IN A-REG ABS VAL OF AC AUX INDEX FOR 16 BITS ACX * AUX + XTND TO AC, XTND IF NO CARRY, NO PARTIAL PROD. ADD MPLCND (AUX) TO PARTIAL PROD (XTND)
FB2E: FB2E: FB2E: FB33: FB36: FB39: FB39: FB39: FB39: FB39: FB40: FB49: FB49: FB40: FB40: FB40: FB40: FB51: FB55: FB55: FB55: FB55: FB55: FB560: FB66: FB66: FB66: FB66: FB67: FB77:	600 A95 ADD AD0 FDD AD0 859 859 859 859 850 A05 A05 A05 A05 A05 A05 A05 A05 A05 A	00 48 56 51 00 50 53 36 14 220 228 21 83 217 25 224 105 0 50 56 4 56 4 56 4 56 57 57 57 57 57 57 57 57 57 57 57 57 57	C0 C0 C0 F8 FC FB	RTS2D INIT SETTXT SETGR SETWND TABV MULPM MUL2 MUL2	RTS LDA STA LDA LDA LDA LDA LDA LDA LDA LDA LDA LD	<pre>#\$00 STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$14 WNDUTH #\$18 WNDBTM #\$17 CV VTAB MD1 #\$10 ACL A MUL4 #\$FE XTNDL+2,X AUXL+2,X XTNDL+2,X</pre>	CLR STATUS FOR DEBUG SOFTWARE INIT VIDEO MODE SET FOR TEXT MODE FULL SCREEN WINDOW SET FOR GRAPHICS MODE LOWER 4 LINES AS TEXT WINDOW SET FOR 40 COL WINDOW TOP IN A-REG, BTTM AT LINE 24 VTAB TO ROW 23 VTABS TO ROW IN A-REG ABS VAL OF AC AUX INDEX FOR 16 BITS ACX * AUX + XTND TO AC, XTND IF NO CARRY, NO PARTIAL PROD. ADD MPLCND (AUX) TO PARTIAL PROD (XTND)
FB2E: FB2E: FB2E: FB33: FB36: FB39: FB39: FB39: FB39: FB39: FB40: FB49: FB49: FB49: FB49: FB49: FB49: FB49: FB49: FB51: FB55: FB55: FB55: FB55: FB55: FB55: FB66: FB66: FB66: FB66: FB66: FB66: FB67: FB77:	600 A95 ADD ADD FDD AD0 859 859 859 859 859 850 A05 A05 A05 A05 A05 A05 A05 A05 A05 A	00845654 5545100055364220022821832172522440500 5545547222440000000000000000000000000000	C0 C0 C0 F8 FC FB	RTS2D INIT SETTXT SETGR SETWND TABV MULPM MUL2 MUL3	RTS LDA STA LDA LDA LDA LDA LDA LDA LDA LDA LDA LD	<pre>#\$00 sTATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$14 WNDUFT #\$28 WNDWDTH #\$18 WNDBTM #\$17 CV VTAB MD1 #\$10 ACL A MUL4 #\$FE XTNDL+2,X AUXL+2,X XTNDL+2,X MUL3 #\$00</pre>	CLR STATUS FOR DEBUG SOFTWARE INIT VIDEO MODE SET FOR TEXT MODE FULL SCREEN WINDOW SET FOR GRAPHICS MODE LOWER 4 LINES AS TEXT WINDOW SET FOR 40 COL WINDOW TOP IN A-REG, BTTM AT LINE 24 VTAB TO ROW 23 VTABS TO ROW IN A-REG ABS VAL OF AC AUX INDEX FOR 16 BITS ACX * AUX + XTND TO AC, XTND IF NO CARRY, NO PARTIAL PROD. ADD MPLCND (AUX) TO PARTIAL PROD (XTND)
FB2E: FB2E: FB2E: FB33: FB36: FB39: FB39: FB39: FB39: FB39: FB39: FB40: FB40: FB40: FB40: FB40: FB40: FB55: FB55: FB55: FB55: FB55: FB55: FB55: FB55: FB66: FB66: FB66: FB66: FB67: FB71: FB73: FB74: FB74: FB74: FB74: FB74: FB74:	600 A95 ADD ADD FDD AD0 859 A95 A859 859 859 859 850 A05 A05 A05 A05 A05 A05 A05 A05 A05 A	00845654 55451000553642200228218237252244005005554 554555555555555555555555	C0 C0 C0 F8 FC FB	RTS2D INIT SETTXT SETGR SETWND TABV MULPM MUL2 MUL3 MUL4	RTS LDA STA LDA LDA LDA LDA LDA LDA LDA LDA LDA LD	<pre>#\$00 STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$00 WNDLFT #\$28 WNDWDTH #\$18 WNDBTM #\$17 CV VTAB MD1 #\$10 ACL A MUL4 #\$FE XTNDL+2,X AUXL+2,X XTNDL+2,X MUL3 #\$03 \$76</pre>	CLR STATUS FOR DEBUG SOFTWARE INIT VIDEO MODE SET FOR TEXT MODE FULL SCREEN WINDOW SET FOR GRAPHICS MODE LOWER 4 LINES AS TEXT WINDOW SET FOR 40 COL WINDOW TOP IN A-REG, BTTM AT LINE 24 VTAB TO ROW 23 VTABS TO ROW IN A-REG ABS VAL OF AC AUX INDEX FOR 16 BITS ACX * AUX + XTND TO AC, XTND IF NO CARRY, NO PARTIAL PROD. ADD MPLCND (AUX) TO PARTIAL PROD (XTND)
FB2E: FB2E: FB2E: FB33: FB36: FB39: FB39: FB39: FB39: FB39: FB40: FB49: FB49: FB49: FB49: FB49: FB49: FB49: FB49: FB49: FB49: FB55: FB55: FB55: FB55: FB55: FB59: FB63: FB63: FB66: FB66: FB66: FB66: FB67: FB67: FB71: FB71: FB74: FB77: FB78:	6095 8AD AD95 AD0 AD095 AD02 AD095 AD02 AD02 AD02 AD02 AD02 AD02 AD02 AD02	00 48 56 54 51 00 05 336 12 22 21 82 37 222 41 0 50 55 4 55 4 55 4 55 4 55 4 55 4 5	C0 C0 C0 F8 FC FB	RTS2D INIT SETTXT SETGR SETWND TABV MULPM MUL2 MUL3 MUL4 MUL5	RTS LDA STA LDA LDA LDA LDA LDA LDA LDA LDA LDA LD	<pre>#\$00 \$TATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$00 WNDLFT #\$28 WNDWDTH #\$18 WNDBTM #\$17 CV VTAB MD1 #\$10 ACL A MUL4 #\$FE XTNDL+2,X AUXL+2,X XTNDL+2,X MUL3 #\$03 \$76 \$50</pre>	CLR STATUS FOR DEBUG SOFTWARE INIT VIDEO MODE SET FOR TEXT MODE FULL SCREEN WINDOW SET FOR GRAPHICS MODE LOWER 4 LINES AS TEXT WINDOW SET FOR 40 COL WINDOW TOP IN A-REG, BTTM AT LINE 24 VTAB TO ROW 23 VTABS TO ROW IN A-REG ABS VAL OF AC AUX INDEX FOR 16 BITS ACX * AUX + XTND TO AC, XTND IF NO CARRY, NO PARTIAL PROD. ADD MPLCND (AUX) TO PARTIAL PROD (XTND)
FB2E: FB2E: FB2E: FB31: FB33: FB36: FB37: FB36: FB40: FB49: FB49: FB49: FB49: FB49: FB49: FB49: FB49: FB49: FB49: FB55: FB57: FB59: FB59: FB59: FB59: FB59: FB59: FB59: FB66: FB66: FB66: FB66: FB67: FB67: FB71: FB74: FB77:	600 A95 ADD AD0 AD0 AD0 AD0 AD0 AD0 AD0 AD0 AD0	00 456 54 50 00 50 536 14 200 20 82 18 20 20 20 20 22 12 52 24 10 50 50 50 53 6 14 200 20 82 18 21 50 50 50 50 53 6 6 50 50 53 6 6 50 50 50 50 53 6 6 50 20 20 20 20 20 20 50 50 50 50 50 50 50 50 50 50 50 50 50	C0 C0 C0 F8 FC FB	RTS2D INIT SETTXT SETGR SETWND TABV MULPM MUL2 MUL3 MUL3 MUL4	RTS LDA STA LDA LDA LDA LDA LDA LDA LDA LDA STA LDA LDA STA LDA LDA STA LDA LDA LDA LDA LDA LDA LDA LDA LDA LD	<pre>#\$00 STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDDTOP #\$00 WNDLFT #\$28 WNDWDTH #\$18 WNDBTM #\$17 CV VTAB MD1 #\$10 ACL A MUL4 #\$FE XTNDL+2,X AUXL+2,X XTNDL+2,X MUL3 #\$03 \$76 \$50</pre>	CLR STATUS FOR DEBUG SOFTWARE INIT VIDEO MODE SET FOR TEXT MODE FULL SCREEN WINDOW SET FOR GRAPHICS MODE LOWER 4 LINES AS TEXT WINDOW SET FOR 40 COL WINDOW TOP IN A-REG, BTTM AT LINE 24 VTAB TO ROW 23 VTABS TO ROW IN A-REG ABS VAL OF AC AUX INDEX FOR 16 BITS ACX * AUX + XTND TO AC, XTND IF NO CARRY, NO PARTIAL PROD. ADD MPLCND (AUX) TO PARTIAL PROD (XTND)
FB2E: FB2E: FB2E: FB2E: FB33: FB33: FB33: FB36: FB39: FB39: FB39: FB40: FB49: FB49: FB49: FB49: FB49: FB49: FB49: FB55: FB57: FB57: FB57: FB57: FB57: FB57: FB663: FB663: FB664: FB664: FB664: FB664: FB671: FB71: FB771: FB771: FB773: FB778: FB778:	600 A95 ADD ADD ADD ADD ADD ADD ADD ADD ADD AD	008456545100055366420022812522240500555645540005055655455655455655455655455545	C0 C0 C0 F8 FC FB	RTS2D INIT SETTXT SETGR SETWND TABV MULPM MUL2 MUL3 MUL3 MUL4 MUL5	RTS LDA STA LDA LDA LDA LDA LDA LDA LDA LDA LDA LD	<pre>#\$00 STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$14 WNDUTP #\$28 WNDWDTH #\$18 WNDWDTH #\$17 CV VTAB MD1 #\$10 ACL A MUL4 #\$FE XTNDL+2,X AUXL+2,X XTNDL+2,X MUL3 #\$03 \$76 \$50 MUL5</pre>	CLR STATUS FOR DEBUG SOFTWARE INIT VIDEO MODE SET FOR TEXT MODE FULL SCREEN WINDOW SET FOR GRAPHICS MODE LOWER 4 LINES AS TEXT WINDOW SET FOR 40 COL WINDOW TOP IN A-REG, BTTM AT LINE 24 VTAB TO ROW 23 VTABS TO ROW IN A-REG ABS VAL OF AC AUX INDEX FOR 16 BITS ACX * AUX + XTND TO AC, XTND IF NO CARRY, NO PARTIAL PROD. ADD MPLCND (AUX) TO PARTIAL PROD (XTND)
FB2E: FB2E: FB2E: FB2E: FB33: FB33: FB36: FB39: FB39: FB39: FB39: FB40: FB49: FB49: FB49: FB49: FB49: FB49: FB49: FB53: FB57: FB57: FB57: FB57: FB57: FB57: FB56: FB66: FB66: FB66: FB67: FB67: FB66: FB67: FB77:	600 A95 ADD ADD FDD AD0 8A95 A95 A95 A95 A95 A95 A95 A95 A95 A95	00045654 554510005336422002282118321725224401500 F546554703 FB	C0 C0 C0 F8 FC FB	RTS2D INIT SETTXT SETGR SETWND TABV MULPM MUL2 MUL3 MUL3	RTS LDA STA LDA LDA LDA LDA LDA LDA LDA LDA LDA LD	<pre>#\$00 STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$14 WNDUFT #\$28 WNDWDTH #\$17 CV VTAB MD1 #\$10 ACL A MUL4 #\$FE XTNDL+2,X AUXL+2,X XTNDL+2,X MUL3 #\$03 \$76 \$50 MUL5</pre>	CLR STATUS FOR DEBUG SOFTWARE INIT VIDEO MODE SET FOR TEXT MODE FULL SCREEN WINDOW SET FOR GRAPHICS MODE LOWER 4 LINES AS TEXT WINDOW SET FOR 40 COL WINDOW TOP IN A-REG, BTTM AT LINE 24 VTAB TO ROW 23 VTABS TO ROW IN A-REG ABS VAL OF AC AUX INDEX FOR 16 BITS ACX * AUX + XTND TO AC, XTND IF NO CARRY, NO PARTIAL PROD. ADD MPLCND (AUX) TO PARTIAL PROD (XTND)
FB2E:         FB2E:         FB2E:         FB33:         FB33:         FB36:         FB39:         FB39:         FB39:         FB39:         FB39:         FB39:         FB39:         FB40:         FB43:         FB44:         FB44:         FB48:         FB48:         FB55:         FB551:         FB552:         FB551:         FB552:         FB551:         FB552:         FB551:         FB551:         FB552:         FB552:         FB552:         FB652:         FB652:         FB663:         FB663:         FB663:         FB664:         FB671:         FB764:         FB774:	600 A95 ADD ADD FDD AD0 8A9 8A9 8A95 A95 A95 A95 A95 A95 A95 A95 A95 A95	008456545100055364220022812327252240050055654570005576554555455545555555555	C0 C0 C0 F8 FC FB	RTS2D INIT SETTXT SETGR SETWND TABV MULPM MUL2 MUL3 MUL3 MUL4	RTS LDA STA LDA LDA LDA LDA LDA LDA LDA LDA LDA LD	<pre>#\$00 STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$14 WNDUTH #\$18 WNDBTM #\$17 CV VTAB MD1 #\$10 ACL A MUL4 #\$FE XTNDL+2,X AUXL+2,X XTNDL+2,X MUL3 #\$03 \$76 \$50 MUL5 MUL2</pre>	CLR STATUS FOR DEBUG SOFTWARE INIT VIDEO MODE SET FOR TEXT MODE FULL SCREEN WINDOW SET FOR GRAPHICS MODE LOWER 4 LINES AS TEXT WINDOW SET FOR 40 COL WINDOW TOP IN A-REG, BTTM AT LINE 24 VTAB TO ROW 23 VTABS TO ROW IN A-REG ABS VAL OF AC AUX INDEX FOR 16 BITS ACX * AUX + XTND TO AC, XTND IF NO CARRY, NO PARTIAL PROD. ADD MPLCND (AUX) TO PARTIAL PROD (XTND)

LDOT:	20	A4	FΒ	DIVPM	JSR	MDl	ABS VAL OF AC, AUX.
FB84:	A0	10		DIV	LDY	#\$10	INDEX FOR 16 BITS
FB86:	06	50		DIV2	ASL	ACL	
FB88:	26	51			ROL	ACH	
FB8A:	26	52			ROL	XTNDL	XTND/AUX
FB8C:	26	53			ROL	XTNDH	TO AC.
FB8E:	38				SEC		
FB8F:	A5	52			LDA	XTNDL	
FB91:	E5	54			SBC	AUXL	MOD TO XTND.
FB93:	AA				TAX		
FB94:	A 5	53			T.DA	XTNDH	
FB96:	E5	55			SBC	AUXH	
FB98.	90	06			BCC		
FB91.	86	52			STX	XTNDI.	
FDOC.	00	52			CUN	VTNDU	
FDOF.	55	50			TNC	ACT	
	00	50		D T 17 2	TINC	ACL	
FDAU:	00	т <b>Э</b>		DIVS	DEI	DT170	
FDAL:	00	сэ			DINE	DIVZ	
FBA3:	00	~ ~		ND 1	RTS	# ¢ 0 0	
FBA4:	AU	00		MDI	LDY	#\$00	ABS VAL OF AC, AUX
FBA6:	84	2 F			STY	SIGN	WITH RESULT SIGN
FBA8:	A2	54			LDX	#AUXL	IN LSB OF SIGN.
FBAA:	20	AF	FΒ		JSR	MD3	
FBAD:	A2	50			LDX	#ACL	
FBAF:	В5	01		MD 3	LDA	LOCl,X	X SPECIFIES AC OR AUX
FBBl:	10	0 D			BPL	MDRTS	
FBB3:	38				SEC		
FBB4:	98				TYA		
FBB5:	F5	00			SBC	LOC0,X	COMPL SPECIFIED REG
FBB7:	95	00			STA	LOC0,X	IF NEG.
FBB9:	98				TYA	•	
FBBA:	F5	01			SBC	LOCL.X	
FBBC:	95	01			STA	LOCIX	
FBBE:	E6	2 F			TNC	STGN	
FBC0:	60			MDRTS	RTS	5101	
FBC1:	48			BASCALC	PHA		CALC BASE ADD IN BASL H
FBC2.	/ A			DIDCIILC	T.CD	Δ	FOR GIVEN LINE NO
FDC2.	20	03				#¢03	OZ-IINE NO Z-\$17
FDC5.	00	0.0				#\$0J #\$01	
FDCJ.	05	20			CULA	#904 DACH	PACH-000001CD
FBC/:	80	29			STA	вазн	BASH=000001CD
FBC9:	00	10			PLA	# <b>6</b> 7 0	AND
FBCA:	29	18			AND	#\$18	BASL=EABAB000
FBCC:	90	02			BCC	BSCLC2	
FBCE:	69	/ F			ADC	#\$/E	
	0 5	20		BSCLC2	CUDY		
FBD0:	0.0	20		DDCLCL	SIA	BASL	
FBD0: FBD2:	0A	20		DBCLCL	ASL	BASL	
FBD0: FBD2: FBD3:	0A 0A	20		DOLOZ	ASL ASL	BASL	
FBD0: FBD2: FBD3: FBD4:	0A 0A 05	28		DUDUL	ASL ASL ORA	BASL	
FBD0: FBD2: FBD3: FBD4: FBD6:	0A 0A 05 85	28 28		DUCLUE	ASL ASL ORA STA	BASL BASL BASL	
FBD0: FBD2: FBD3: FBD4: FBD6: FBD8:	0A 0A 05 85 60	28 28		Dellel	ASL ASL ORA STA RTS	BASL BASL	
FBD0: FBD2: FBD3: FBD4: FBD6: FBD8: FBD9:	0A 0A 05 85 60 C9	28 28 87		BELL1	ASL ASL ORA STA RTS CMP	BASL BASL #\$87	BELL CHAR? (CNTRL-G)
FBD0: FBD2: FBD3: FBD4: FBD6: FBD8: FBD9: FBDB:	0A 0A 05 85 60 C9 D0	28 28 28 87 12		BELLI	ASL ASL ORA STA RTS CMP BNE	BASL BASL BASL #\$87 RTS2B	BELL CHAR? (CNTRL-G) NO, RETURN
FBD0: FBD2: FBD3: FBD4: FBD6: FBD8: FBD9: FBDB: FBDD:	0A 0A 05 85 60 C9 D0 A9	28 28 87 12 40		BELLI	ASL ASL ORA STA RTS CMP BNE LDA	BASL BASL #\$87 #\$2B #\$40	BELL CHAR? (CNTRL-G) NO, RETURN DELAY .01 SECONDS
FBD0: FBD2: FBD3: FBD4: FBD6: FBD8: FBD9: FBDB: FBDD: FBDF:	0A 0A 05 85 60 C9 D0 A9 20	28 28 87 12 40 A8	FC	BELL1	ASL ASL ORA STA RTS CMP BNE LDA JSR	BASL BASL #\$87 RTS2B #\$40 WAIT	BELL CHAR? (CNTRL-G) NO, RETURN DELAY .01 SECONDS
FBD0: FBD2: FBD3: FBD4: FBD6: FBD8: FBD9: FBDB: FBDD: FBDF: FBE2:	0A 0A 05 85 60 C9 D0 A9 20 A0	28 28 87 12 40 A8 C0	FC	BELL1	ASL ASL ORA STA RTS CMP BNE LDA JSR LDY	BASL BASL #\$87 RTS2B #\$40 WAIT #\$C0	BELL CHAR? (CNTRL-G) NO, RETURN DELAY .01 SECONDS
FBD0: FBD2: FBD3: FBD4: FBD6: FBD8: FBD9: FBD9: FBDB: FBDD: FBDF: FBE2: FBE4:	0A 0A 05 85 60 C9 D0 A9 20 A0 A9	28 28 28 87 12 40 A8 C0 0C	FC	BELL1	ASL ASL ORA STA RTS CMP BNE LDA JSR LDY LDA	BASL BASL #\$87 RTS2B #\$40 WAIT #\$C0 #\$0C	BELL CHAR? (CNTRL-G) NO, RETURN DELAY .01 SECONDS TOGGLE SPEAKER AT
FBD0: FBD2: FBD3: FBD4: FBD6: FBD8: FBD9: FBD9: FBDB: FBDD: FBDF: FBE2: FBE4: FBE6:	0A 0A 05 85 60 C9 D0 A9 20 A0 A9 20	28 28 28 87 12 40 A8 C0 0C A8	FC	BELL1 BELL2	ASL ASL ORA STA RTS CMP BNE LDA JSR LDY LDA JSR	BASL BASL #\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT	BELL CHAR? (CNTRL-G) NO, RETURN DELAY .01 SECONDS TOGGLE SPEAKER AT
FBD0: FBD2: FBD3: FBD4: FBD6: FBD8: FBD9: FBD9: FBD5: FBDF: FBE2: FBE4: FBE6: FBE9:	0A 0A 05 85 60 C9 D0 A9 20 A0 A9 20 A0	28 28 28 87 12 40 A8 C0 0C A8 30	FC	BELL1 BELL2	ASL ASL ORA STA RTS CMP BNE LDA JSR LDA JSR LDA	BASL BASL #\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR	BELL CHAR? (CNTRL-G) NO, RETURN DELAY .01 SECONDS TOGGLE SPEAKER AT 1 KHZ FOR .1 SEC.
FBD0: FBD2: FBD2: FBD4: FBD6: FBD9: FBD9: FBD9: FBD9: FBD5: FBD7: FBE4: FBE4: FBE6: FBE9:	0A 0A 05 85 60 C9 D0 A9 20 A0 A9 20 AD 88	28 28 87 12 40 A8 C0 0C A8 30	FC FC C0	BELL1 BELL2	ASL ASL ORA STA RTS CMP BNE LDA JSR LDY LDA JSR LDA DEY	BASL BASL #\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR	<pre>BELL CHAR? (CNTRL-G) NO, RETURN DELAY .01 SECONDS TOGGLE SPEAKER AT 1 KHZ FOR .1 SEC.</pre>
FBD0: FBD2: FBD2: FBD4: FBD4: FBD9: FBD9: FBD9: FBD9: FBD7: FBE4: FBE4: FBE6: FBE9: FBE2: FBE4: FBE6: FBE9: FBE2:	0A 0A 05 85 60 C9 D0 A9 20 A0 A9 20 A0 88 D0	28 28 87 12 40 A8 C0 0C A8 30	FC FC C0	BELL1 BELL2	ASL ASL ASL ORA STA RTS CMP BNE LDA JSR LDA JSR LDA JSR LDA DEY BNE	BASL BASL #\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR BELL2	<pre>BELL CHAR? (CNTRL-G) NO, RETURN DELAY .01 SECONDS TOGGLE SPEAKER AT 1 KHZ FOR .1 SEC.</pre>
FBD0: FBD2: FBD2: FBD4: FBD4: FBD8: FBD9: FBD9: FBD9: FBD9: FBD9: FBE0: FBE4: FBE6: FBE9: FBEC: FBE2: FBE5: FBE5: FBE5: FBE5: FBE5: FBE5: FBE5: FBE5: FBD3: FBD5: FBD3: FBD3: FBD5: FBD3: FBD3: FBD5: FBD3: FBD5: FBD5: FBD5: FBD5: FBD5: FBD5: FBD5: FBD5: FBD5: FBD5: FBD5: FBD5: FBD5: FBD5: FBE5:	85 0A 05 85 60 20 A0 20 A0 A0 88 D0 60	28 28 87 12 40 A8 C0 0C A8 30 F5	FC CO	BELL1 BELL2 BTS2B	ASL ASL ORA STA RTS CMP BNE LDA JSR LDY LDA JSR LDA DEY BNE RTS	BASL BASL #\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR BELL2	<pre>BELL CHAR? (CNTRL-G) NO, RETURN DELAY .01 SECONDS TOGGLE SPEAKER AT 1 KHZ FOR .1 SEC.</pre>
FBD0: FBD2: FBD2: FBD4: FBD6: FBD8: FBD8: FBD9: FBD9: FBD9: FBD9: FBD7: FBE7: FBE6: FBE6: FBE6: FBE6: FBE6: FBE6: FBE6: FBE6: FBE6: FBE7: FBE6: FBE7:	85 0A 05 85 60 20 A9 20 A0 A0 20 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0	28 28 28 87 12 40 A8 C0 0C A8 30 F5 24	FC FC C0	BELL1 BELL2 RTS2B	ASL ASL ORA STA RTS CMP BNE LDA JSR LDY LDA JSR LDA DEY BNE RTS LDY	BASL BASL #\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR BELL2 CH	BELL CHAR? (CNTRL-G) NO, RETURN DELAY .01 SECONDS TOGGLE SPEAKER AT 1 KHZ FOR .1 SEC.
FBD0: FBD2: FBD2: FBD4: FBD6: FBD8: FBD9: FBD9: FBD9: FBD9: FBD7: FBE4: FBE4: FBE4: FBE5: FBE5: FBE7: FBE7:	85 0A 05 85 60 20 A9 20 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0	28 28 28 87 12 40 A8 C0 0C A8 30 F5 24	FC FC C0	BELL1 BELL2 RTS2B STOADV	ASL ASL ASL ORA STA RTS LDA JSR LDA JSR LDA DEY BNE RTS LDY STA	BASL BASL #\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR BELL2 CH	<pre>BELL CHAR? (CNTRL-G) NO, RETURN DELAY .01 SECONDS TOGGLE SPEAKER AT 1 KHZ FOR .1 SEC.</pre> CURSOR H INDEX TO Y-REG
FBD0: FBD2: FBD2: FBD4: FBD4: FBD9: FBD9: FBD9: FBD9: FBD9: FBD7: FBE4: FBE4: FBE6: FBE2: FBE5: FBE7: FBF0: FBF2:	0A 0A 05 85 60 20 A9 20 A0 20 A0 20 A0 20 A0 40 60 49 20 A0 5 5 60 20 20 20 20 20 5 5 5 5 5 60 20 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5	28 28 28 87 12 40 A8 C0 0C A8 30 F5 24 28 24	FC FC C0	BELL1 BELL2 RTS2B STOADV	ASL ASL ORA STA RTS CMP BNE LDA JSR LDA JSR LDA JSR LDA JSR LDA STA RTS LDY STA RTS LDY STA	BASL BASL BASL #\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR BELL2 CH (BASL),Y	<pre>BELL CHAR? (CNTRL-G) NO, RETURN DELAY .01 SECONDS TOGGLE SPEAKER AT 1 KHZ FOR .1 SEC. CURSOR H INDEX TO Y-REG STORE CHAR IN LINE INCREMENT CURSOR H INDEX</pre>
FBD0: FBD2: FBD2: FBD4: FBD4: FBD6: FBD9: FBD9: FBD9: FBD7: FBE4: FBE4: FBE6: FBE7: FBE7: FBF4:	0A 0A 05 85 60 20 A9 20 A0 A0 20 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0	28 28 28 87 12 40 87 24 20 75 24 28 24 24 24	FC FC C0	BELL1 BELL2 RTS2B STOADV ADVANCE	ASL ASL ASL ORA STA RTS CMP BNE LDA JSR LDY LDA JSR LDY BNE RTS LDY STA INC	BASL BASL #\$87 RTS2B #\$40 WAIT #\$00 #\$0C WAIT SPKR BELL2 CH (BASL),Y CH	<pre>BELL CHAR? (CNTRL-G) NO, RETURN DELAY .01 SECONDS TOGGLE SPEAKER AT 1 KHZ FOR .1 SEC. CURSOR H INDEX TO Y-REG STORE CHAR IN LINE INCREMENT CURSOR H INDEX (NOUE DECUTE)</pre>
FBD0:         FBD0:         FBD2:         FBD4:         FBD6:         FBD8:         FBD8:         FBD9:         FBD7:         FBE4:         FBE4:         FBE4:         FBE6:         FBE6:         FBF0:         FBF4:         FBF4:         FBF6:         FBF6:         FBF6:	0A 0A 05 85 60 0A 20 A0 20 A0 20 A0 20 A0 20 A0 20 A0 20 A0 20 A0 20 A0 20 A0 20 A0 20 A0 20 A0 20 A0 20 A0 20 A0 20 A0 20 A0 20 A0 20 20 A0 20 20 20 20 20 20 20 20 20 20 20 20 20	28 28 28 87 12 40 A8 C0 0C A8 30 F5 24 28 24 24 24	FC FC C0	BELL1 BELL2 RTS2B STOADV ADVANCE	ASL ASL ORA STA RTS CMP BNE LDA JSR LDA JSR LDA JSR LDA JSR LDA STA INC LDA STA	BASL BASL BASL #\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR BELL2 CH (BASL),Y CH CH	<pre>BELL CHAR? (CNTRL-G) NO, RETURN DELAY .01 SECONDS TOGGLE SPEAKER AT 1 KHZ FOR .1 SEC. CURSOR H INDEX TO Y-REG STORE CHAR IN LINE INCREMENT CURSOR H INDEX (MOVE RIGHT) DEVON UNDERNO</pre>
FBD0: FBD2: FBD2: FBD3: FBD4: FBD6: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBD7: FBE4: FBE4: FBE6: FBE7: FBF4: FBF6: FBF8:	0A 0A 05 85 60 20 A9 20 A0 20 A0 20 A0 20 A0 88 D0 60 4 91 E6 50 20 C9 20 C9 C9 C9 C9 C9 C9 C9 C9 C9 C9 C9 C9 C9	28 28 28 87 12 40 A8 C0 0C A8 30 F5 24 28 24 24 24 26 C	FC CO	BELL1 BELL2 RTS2B STOADV ADVANCE	ASL ASL ORA STA RTS CMP BNE LDA JSR LDA JSR LDA JSR LDA DEY BNE RTS LDA STA INC LDA CMP	BASL BASL BASL #\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR BELL2 CH (BASL),Y CH CH WNDWDTH	<ul> <li>BELL CHAR? (CNTRL-G) NO, RETURN DELAY .01 SECONDS</li> <li>TOGGLE SPEAKER AT 1 KHZ FOR .1 SEC.</li> <li>CURSOR H INDEX TO Y-REG STORE CHAR IN LINE INCREMENT CURSOR H INDEX (MOVE RIGHT)</li> <li>BEYOND WINDOW WIDTH?</li> </ul>
FBD0: FBD2: FBD2: FBD4: FBD4: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBE4: FBE4: FBE6: FBF6: FBF6: FBF8: FBF4:	0A 0A 05 80 00 80 00 00 00 00 00 00 00 00 00 00	28 28 28 87 12 40 A8 C0 0C A8 30 F5 24 28 24 24 24 24 26 66	FC FC C0	BELL1 BELL2 RTS2B STOADV ADVANCE	ASL ASL ASL ORA STA RTS CMP BNE LDA JSR LDA JSR LDA DEY BNE RTS LDY STA INC LDA CMP BCS	BASL BASL BASL #\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR BELL2 CH (BASL),Y CH CH WNDWDTH CR	<ul> <li>BELL CHAR? (CNTRL-G) NO, RETURN DELAY .01 SECONDS</li> <li>TOGGLE SPEAKER AT 1 KHZ FOR .1 SEC.</li> <li>CURSOR H INDEX TO Y-REG STORE CHAR IN LINE INCREMENT CURSOR H INDEX (MOVE RIGHT)</li> <li>BEYOND WINDOW WIDTH? YES CR TO NEXT LINE</li> </ul>
FBD0:         FBD0:         FBD2:         FBD2:         FBD4:         FBD8:         FBD8:         FBD9:         FBD9:         FBD7:         FBE4:         FBE4:         FBE4:         FBE6:         FBE7:         FBF0:         FBF4:         FBF4:         FBF8:         FBF8:         FBF8:         FBF6:	00 00 00 00 00 00 00 00 00 00 00 00 00	28 28 28 87 12 40 A8 C0 0C A8 30 F5 24 24 24 24 21 66	FC CO	BELL1 BELL2 RTS2B STOADV ADVANCE RTS3	ASL ASL ASL ORA STA RTS CMP BNE LDA JSR LDA JSR LDA JSR LDA JSR LDA STA LDA STA IDC LDA CMP BCS RTS	BASL BASL BASL #\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR BELL2 CH (BASL),Y CH WNDWDTH CR	<ul> <li>BELL CHAR? (CNTRL-G) NO, RETURN DELAY .01 SECONDS</li> <li>TOGGLE SPEAKER AT 1 KHZ FOR .1 SEC.</li> <li>CURSOR H INDEX TO Y-REG STORE CHAR IN LINE INCREMENT CURSOR H INDEX (MOVE RIGHT)</li> <li>BEYOND WINDOW WIDTH? YES CR TO NEXT LINE NO, RETURN</li> </ul>
FBD0: FBD2: FBD2: FBD4: FBD4: FBD6: FBD8: FBD9: FBD7: FBE4: FBE4: FBE4: FBE4: FBE7: FBF4: FBF6: FBF7: FBF7: FBF7: FBF7:	0 A 0 0 5 0 A 0 0 5 0 C 9 0 A 9 0 A 0 0 5 0 C 9 0 A 9 0 A 0 0 5 0 C 9 0 A 0 0 A 0 0 A 0 1 A 0 0	28 28 28 87 12 40 A8 C0 0C A8 30 F5 24 24 24 24 21 66 A0	FC FC C0	BELL1 BELL2 RTS2B STOADV ADVANCE RTS3 VIDOUT	ASL ASL ASL ORA STA RTS CMP BNE LDA JSR LDA JSR LDA JSR LDA JSR LDA DEY STA INC LDA STA INC CMP BCS RTS CMP	BASL BASL BASL #\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR BELL2 CH (BASL),Y CH CH CH WNDWDTH CR #\$A0	<ul> <li>BELL CHAR? (CNTRL-G) NO, RETURN</li> <li>DELAY .01 SECONDS</li> <li>TOGGLE SPEAKER AT 1 KHZ FOR .1 SEC.</li> <li>CURSOR H INDEX TO Y-REG STORE CHAR IN LINE INCREMENT CURSOR H INDEX (MOVE RIGHT)</li> <li>BEYOND WINDOW WIDTH? YES CR TO NEXT LINE NO, RETURN</li> <li>CONTROL CHAR?</li> </ul>
FBD0:         FBD0:         FBD2:         FBD3:         FBD4:         FBD8:         FBD8:         FBD7:         FB27         FB28         FB29:         FB29:         FB29:         FB29:         FB20:         FB20:         FB21:         FB54:         FB54: </td <td>00 00 00 00 00 00 00 00 00 00 00 00 00</td> <td>28 28 28 87 12 40 A8 20 0C A8 30 F5 24 28 24 22 24 22 4 21 66 A0 EF</td> <td>FC FC C0</td> <td>BELL1 BELL2 RTS2B STOADV ADVANCE RTS3 VIDOUT</td> <td>ASL ASL ASL ORA STA RTS CMP BNE LDA JSR LDA JSR LDA JSR LDA JSR LDA STA INC LDA STA INC LDA CMP BCS RTS CMP BCS</td> <td>BASL BASL BASL #\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR BELL2 CH (BASL),Y CH CH WNDWDTH CR #\$A0 STOADV</td> <td><ul> <li>BELL CHAR? (CNTRL-G) NO, RETURN</li> <li>DELAY .01 SECONDS</li> <li>TOGGLE SPEAKER AT 1 KHZ FOR .1 SEC.</li> <li>CURSOR H INDEX TO Y-REG STORE CHAR IN LINE</li> <li>INCREMENT CURSOR H INDEX (MOVE RIGHT)</li> <li>BEYOND WINDOW WIDTH? YES CR TO NEXT LINE NO, RETURN</li> <li>CONTROL CHAR? NO, OUTPUT IT.</li> </ul></td>	00 00 00 00 00 00 00 00 00 00 00 00 00	28 28 28 87 12 40 A8 20 0C A8 30 F5 24 28 24 22 24 22 4 21 66 A0 EF	FC FC C0	BELL1 BELL2 RTS2B STOADV ADVANCE RTS3 VIDOUT	ASL ASL ASL ORA STA RTS CMP BNE LDA JSR LDA JSR LDA JSR LDA JSR LDA STA INC LDA STA INC LDA CMP BCS RTS CMP BCS	BASL BASL BASL #\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR BELL2 CH (BASL),Y CH CH WNDWDTH CR #\$A0 STOADV	<ul> <li>BELL CHAR? (CNTRL-G) NO, RETURN</li> <li>DELAY .01 SECONDS</li> <li>TOGGLE SPEAKER AT 1 KHZ FOR .1 SEC.</li> <li>CURSOR H INDEX TO Y-REG STORE CHAR IN LINE</li> <li>INCREMENT CURSOR H INDEX (MOVE RIGHT)</li> <li>BEYOND WINDOW WIDTH? YES CR TO NEXT LINE NO, RETURN</li> <li>CONTROL CHAR? NO, OUTPUT IT.</li> </ul>
FBD0:         FBD0:         FBD2:         FBD3:         FBD4:         FBD8:         FBD8:         FBD9:         FBD8:         FBD9:         FBD8:         FBD9:         FBD9:         FBD1:         FBE4:         FBE4:         FBE7:         FBF4:         FBF4:         FBF4:         FBF4:         FBF6:         FBF7:         FBF7:         FBF7:         FC01:	00 00 00 00 00 00 00 00 00 00	28 28 28 87 12 40 A8 C0 0C A8 30 F5 24 28 24 24 21 66 A0 EF	FC FC C0	BELL1 BELL2 RTS2B STOADV ADVANCE RTS3 VIDOUT	ASL ASL ASL ORA STA RTS CMP BNE LDA JSR LDA JSR LDA DEY BNE RTS LDA STA LDA CMP BCS TAY	BASL BASL BASL #\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR BELL2 CH (BASL),Y CH CH WNDWDTH CR #\$A0 STOADV	<ul> <li>BELL CHAR? (CNTRL-G) NO, RETURN DELAY .01 SECONDS</li> <li>TOGGLE SPEAKER AT 1 KHZ FOR .1 SEC.</li> <li>CURSOR H INDEX TO Y-REG STORE CHAR IN LINE INCREMENT CURSOR H INDEX (MOVE RIGHT)</li> <li>BEYOND WINDOW WIDTH? YES CR TO NEXT LINE NO, RETURN</li> <li>CONTROL CHAR? NO, OUTPUT IT.</li> <li>INVERSE VIDEO?</li> </ul>
FBD0: FBD2: FBD2: FBD2: FBD4: FBD4: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBE4: FBE4: FBE4: FBE7: FBF4: FBF4: FBF4: FBF4: FBF4: FBF4: FBF7:	00A 005 00A 005 009 009 200 A09 200 200 200 200 200 200 200 200 200 2	28 28 28 87 12 40 40 87 12 40 40 87 12 40 87 12 40 87 12 40 87 12 40 87 12 40 87 12 40 87 12 40 87 12 40 87 12 40 87 12 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8	FC CO	BELL1 BELL2 RTS2B STOADV ADVANCE RTS3 VIDOUT	ASL ASL ASL ORA STA RTS CMP BNE LDA JSR LDA JSR LDA JSR LDA DEY BNE RTS LDY STA STA CMP BCS RTS CMP BCS TAY BPL	BASL BASL BASL #\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR BELL2 CH (BASL),Y CH CH CH WNDWDTH CR #\$A0 STOADV	<ul> <li>BELL CHAR? (CNTRL-G) NO, RETURN DELAY .01 SECONDS</li> <li>TOGGLE SPEAKER AT 1 KHZ FOR .1 SEC.</li> <li>CURSOR H INDEX TO Y-REG STORE CHAR IN LINE INCREMENT CURSOR H INDEX (MOVE RIGHT)</li> <li>BEYOND WINDOW WIDTH? YES CR TO NEXT LINE NO,RETURN CONTROL CHAR? NO, OUTPUT IT.</li> <li>INVERSE VIDEO? YES, OUTPUT IT.</li> </ul>
FBD0:         FBD0:         FBD2:         FBD2:         FBD4:         FBD8:         FBD8:         FBD9:         FBD9:         FBD9:         FBD9:         FBD9:         FBD9:         FBD0:         FBE0:         FBE4:         FBE7:         FBF6:         FBF4:         FBF4:         FBF8:         FBF7:         FC01:         FC02:         FC04:	00A 005 00A 005 00 00 00 00 00 00 00 00 00 00 00 00	28 28 28 87 12 40 A8 C0 0CC A8 30 F5 24 224 224 221 66 A0 EF EC 8D	FC FC C0	BELL1 BELL2 RTS2B STOADV ADVANCE RTS3 VIDOUT	ASL ASL ASL ASL ASL ASL ASL ASL ASL CMP BNE LDA JSR LDA LDA LDA LDA LDA LDA LDA STA INC LDA STA INC CMP BCS RTS CMP BCS RTS CMP BPL CMP CMP CMP CMP CMP CMP CMP CMP CMP CMP	BASL BASL BASL #\$87 RTS2B #\$40 WAIT #\$00 WAIT SPKR BELL2 CH (BASL),Y CH CH CH WNDWDTH CR #\$A0 STOADV \$TOADV #\$8D	<ul> <li>BELL CHAR? (CNTRL-G) NO, RETURN DELAY .01 SECONDS</li> <li>TOGGLE SPEAKER AT 1 KHZ FOR .1 SEC.</li> <li>CURSOR H INDEX TO Y-REG STORE CHAR IN LINE INCREMENT CURSOR H INDEX (MOVE RIGHT)</li> <li>BEYOND WINDOW WIDTH? YES CR TO NEXT LINE NO,RETURN</li> <li>CONTROL CHAR? NO,OUTPUT IT.</li> <li>INVERSE VIDEO? YES, OUTPUT IT.</li> <li>CR?</li> </ul>
FBD0: FBD2: FBD2: FBD2: FBD4: FBD4: FBD8: FBD8: FBD9: FBD7: FBE4: FBE4: FBE4: FBE4: FBE5: FBE7: FBF4: FBF6: FBF7: FBF7: FBF7: FBF7: FC01: FC04: FC04:	00A 005 00A 200 200 200 200 200 200 200 200 200	28 28 28 87 12 40 A8 C0 0C A8 30 F5 24 24 24 24 24 24 24 24 24 26 66 A0 EF EC 8D 5A	FC FC C0	BELL1 BELL2 RTS2B STOADV ADVANCE RTS3 VIDOUT	ASL ASL ASL ORA STA RTS CMP BNE LDA JSR LDA JSR LDA JSR LDA JSR LDA DEY BNE STA INC LDA STA INC LDY STA INC CMP BCS RTS CMP BPL CMP BEQ	BASL BASL BASL #\$87 RTS2B #\$40 WAIT #\$00 WAIT SPKR BELL2 CH (BASL),Y CH CH WNDWDTH CR #\$40 STOADV #\$8D CR	<ul> <li>BELL CHAR? (CNTRL-G) NO, RETURN</li> <li>DELAY .01 SECONDS</li> <li>TOGGLE SPEAKER AT 1 KHZ FOR .1 SEC.</li> <li>CURSOR H INDEX TO Y-REG STORE CHAR IN LINE</li> <li>INCREMENT CURSOR H INDEX (MOVE RIGHT)</li> <li>BEYOND WINDOW WIDTH? YES CR TO NEXT LINE NO, RETURN</li> <li>CONTROL CHAR? NO, OUTPUT IT.</li> <li>INVERSE VIDEO? YES, OUTPUT IT.</li> <li>CR? YES.</li> </ul>
FBD0: FBD2: FBD2: FBD2: FBD4: FBD8: FBD8: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBE4: FBE4: FBE4: FBE4: FBE7: FBF4: FBF6: FBF7:	00A05500000000000000000000000000000000	28 28 28 87 12 40 A8 C0 0 C0 A8 C0 0 C0 A8 C0 C0 C0 A8 C0 C0 C0 C0 C0 C0 C0 C0 C0 C0 C0 C0 C0	FC CO	BELL1 BELL2 RTS2B STOADV ADVANCE RTS3 VIDOUT	ASL ASL ASL ORA STA RTS CMP BNE LDA JSR LDY LDA JSR LDY BNE LDA JSR LDA JSR LDA STA INC LDA CMP BCS TAY BECS TAY BEQ CMP	BASL BASL BASL #\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR BELL2 CH (BASL),Y CH CH WNDWDTH CR #\$A0 STOADV #\$8D CR #\$8A	<ul> <li>BELL CHAR? (CNTRL-G) NO, RETURN DELAY .01 SECONDS</li> <li>TOGGLE SPEAKER AT 1 KHZ FOR .1 SEC.</li> <li>CURSOR H INDEX TO Y-REG STORE CHAR IN LINE INCREMENT CURSOR H INDEX (MOVE RIGHT)</li> <li>BEYOND WINDOW WIDTH? YES CR TO NEXT LINE NO, RETURN</li> <li>CONTROL CHAR? NO, OUTPUT IT.</li> <li>INVERSE VIDEO? YES, OUTPUT IT.</li> <li>CR? YES.</li> <li>LINE FEED?</li> </ul>
FBD0: FBD2: FBD2: FBD2: FBD3: FBD4: FBD4: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBE4: FBE4: FBE4: FBF6: FBF7: FBF7: FBF7: FC01: FC04: FC08: FC08:	00A 005 00A 200 200 200 200 200 200 200 200 200	28 28 28 87 40 87 40 87 40 87 24 24 21 66 87 24 22 4 21 66 85 88 55	FC FC C0	BELL1 BELL2 RTS2B STOADV ADVANCE RTS3 VIDOUT	ASL ASL ASL ORA STA RTS CMP BNE LDA JSR LDA JSR LDA JSR LDA DEY BNE RTS LDA STA LDA CMP BCS TAY BCS TAY BEQ CMP BEQ CMP	BASL BASL BASL #\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR BELL2 CH (BASL),Y CH CH CH WNDWDTH CR #\$A0 STOADV #\$8D CR #\$8A LF	<ul> <li>BELL CHAR? (CNTRL-G) NO, RETURN DELAY .01 SECONDS</li> <li>TOGGLE SPEAKER AT 1 KHZ FOR .1 SEC.</li> <li>CURSOR H INDEX TO Y-REG STORE CHAR IN LINE INCREMENT CURSOR H INDEX (MOVE RIGHT)</li> <li>BEYOND WINDOW WIDTH? YES CR TO NEXT LINE NO, RETURN CONTROL CHAR? NO, OUTPUT IT.</li> <li>INVERSE VIDEO? YES, OUTPUT IT.</li> <li>CR? YES.</li> <li>LINE FEED? IF SO, DO IT.</li> </ul>
FBD0: FBD2: FBD2: FBD2: FBD4: FBD5: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBE4: FBE4: FBE4: FBE4: FBE7: FBF4: FBF4: FBF4: FBF4: FBF4: FBF7: FBF7: FBF7: FBF7: FBF7: FC01: FC04: FC08: FC08: FC08: FC08: FC08: FC08: FC08: FC08: FC08: FC08: FC08: FC08: FC08: FC08: FB58: FC00:	00A 005 00A 200 200 200 200 200 200 200 200 200	28 28 28 87 240 A8 C0 00C A8 30 F5 24 224 224 224 224 224 224 224 224 224	FC CO	BELL1 BELL2 RTS2B STOADV ADVANCE RTS3 VIDOUT	ASL ASL ASL ASL ORA STA RTS CMP BNE LDA JSR LDA JSR LDA JSR LDA DEY BNE RTS LDY STA CMP BCS RTS CMP BCS TAY BPL CMP BEQ CMP	BASL BASL BASL #\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR BELL2 CH (BASL),Y CH CH CH WNDWDTH CR #\$A0 STOADV #\$8D CR #\$8A LF #\$88	<ul> <li>BELL CHAR? (CNTRL-G) NO, RETURN DELAY .01 SECONDS</li> <li>TOGGLE SPEAKER AT 1 KHZ FOR .1 SEC.</li> <li>CURSOR H INDEX TO Y-REG STORE CHAR IN LINE INCREMENT CURSOR H INDEX (MOVE RIGHT)</li> <li>BEYOND WINDOW WIDTH? YES CR TO NEXT LINE NO,RETURN</li> <li>CONTROL CHAR? NO,OUTPUT IT.</li> <li>INVERSE VIDEO? YES, OUTPUT IT.</li> <li>INVERSE VIDEO? YES, OUTPUT IT.</li> <li>CR? YES.</li> <li>LINE FEED? IF SO, DO IT.</li> <li>BACK SPACE? (CNTRL-H)</li> </ul>
FBD0:         FBD0:         FBD2:         FBD2:         FBD4:         FBD6:         FBD7:         FBD7:         FB20:         FC01:         FC00:         FC00:         FC00:	00A 00A 005 50 00A 005 50 00A 00 00 00 00 00 00 00 00 00 00 00 0	28 28 28 87 12 40 A8 COC A8 30 F5 24 228 24 226 A0 EF E8D 5A 85 88 C9	FC FC C0	BELL1 BELL2 RTS2B STOADV ADVANCE RTS3 VIDOUT	ASL ASL ASL ASL ORA STA RTS CMP BNE LDA JSR LDA JSR LDA JSR LDA DEY BNE LDA STA IDA STA IDA STA IDA CMP BCS RTS CMP BCS RTS CMP BCS RTS CMP BEQ CMP BEQ CMP BNE	BASL BASL BASL #\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR BELL2 CH (BASL),Y CH CH (BASL),Y CH CH CH WNDWDTH CR #\$A0 STOADV #\$80 STOADV #\$80 CR #\$88 LF #\$88 BELL1	<ul> <li>BELL CHAR? (CNTRL-G) NO, RETURN</li> <li>DELAY .01 SECONDS</li> <li>TOGGLE SPEAKER AT 1 KHZ FOR .1 SEC.</li> <li>CURSOR H INDEX TO Y-REG STORE CHAR IN LINE INCREMENT CURSOR H INDEX (MOVE RIGHT)</li> <li>BEYOND WINDOW WIDTH? YES CR TO NEXT LINE NO,RETURN</li> <li>CONTROL CHAR? NO,OUTPUT IT.</li> <li>INVERSE VIDEO? YES, OUTPUT IT.</li> <li>CR? YES.</li> <li>LINE FEED? IF SO, DO IT.</li> <li>BACK SPACE? (CNTRL-H) NO, CHECK FOR BELL.</li> </ul>
FBD0: FBD2: FBD2: FBD2: FBD3: FBD4: FBD8: FBD8: FBD9: FBD7: FBE4: FBE4: FBE4: FBE4: FBE4: FBE6: FBE7: FBF4: FBF6: FBF7: FC70:	00A 00A 005 00A 005 00 009 00 004 000 000 000 000 000 000 0	28 28 28 87 12 40 87 12 40 87 28 24 21 66 87 24 22 42 24 22 66 87 88 85 88 85 88 85 88 85 88 85 88 85 80 85 80 85 80 85 80 85 80 85 80 85 80 80 80 80 80 80 80 80 80 80 80 80 80	FC CO	BELL1 BELL2 RTS2B STOADV ADVANCE RTS3 VIDOUT	ASL ASL ASL ORA STA RTS CMP BNE LDA JSR LDA JSR LDA JSR LDA JSR LDA JSR LDA STA IDA STA STA CMP BNE CMP BCS TAY BPL CMP BEQ CMP BEQ CMP BNE CMP BNE CMP CMP CMP CMP	BASL BASL BASL #\$87 RTS2B #\$40 WAIT #\$00 WAIT SPKR BELL2 CH (BASL),Y CH CH WNDWDTH CR #\$40 STOADV #\$80 CR #\$88 LF CR #\$88 BELL1 CH	<ul> <li>BELL CHAR? (CNTRL-G) NO, RETURN DELAY .01 SECONDS</li> <li>TOGGLE SPEAKER AT 1 KHZ FOR .1 SEC.</li> <li>CURSOR H INDEX TO Y-REG STORE CHAR IN LINE INCREMENT CURSOR H INDEX (MOVE RIGHT)</li> <li>BEYOND WINDOW WIDTH? YES CR TO NEXT LINE NO, RETURN</li> <li>CONTROL CHAR? NO, OUTPUT IT.</li> <li>INVERSE VIDEO? YES, OUTPUT IT.</li> <li>INVERSE VIDEO? YES.</li> <li>LINE FEED? IF SO, DO IT.</li> <li>BACK SPACE? (CNTRL-H) NO, CHECK FOR BELL.</li> <li>DECREMENT CURSOR H INDEX</li> </ul>
FBD0: FBD2: FBD2: FBD2: FBD4: FBD8: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBE4: FBE4: FBE4: FBE4: FBE4: FBE5: FBF2: FBF4: FBF6: FBF7: FF700: FF700: FF7	00A05500000000000000000000000000000000	28 28 28 28 28 28 28 28 28 28 28 20 28 28 20 28 28 20 28 28 20 28 28 28 28 28 28 28 28 28 28 28 28 28	FC CO	BELL1 BELL2 RTS2B STOADV ADVANCE RTS3 VIDOUT	ASL ASL ASL ORA STA RTS CMP BNE LDA JSR LDY LDA JSR LDY BNE LDA JSR LDA JSR LDA STA IDA STA IDA STA INC LDA CMP BCS TAY BCS TAY BEQ CMP BEQ CMP BNE BEQ CMP BNE BEQ CMP BNE BEQ CMP BNE BEQ CMP BNE BEQ CMP BNE BEQ CMP BNE BEQ CMP BNE BEQ CMP BNE BEQ CMP BNE BEQ CMP BNE BEQ CMP BNE BEQ CMP BNE BEQ CMP BNE BEQ CMP BNE BEQ CMP BNE BEQ CMP BNE BEQ CMP BNE BEQ CMP BNE BEQ CMP BDE BEQ CMP BDE BEQ CMP BDE BEQ CMP BEQ CMP BDE BCS TA STA STA STA STA STA STA STA STA STA	BASL BASL BASL #\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR BELL2 CH (BASL),Y CH CH WNDWDTH CR #\$A0 STOADV #\$80 STOADV #\$80 STOADV #\$88 LF #\$88 BELL1 CH RTS3	<ul> <li>BELL CHAR? (CNTRL-G) NO, RETURN DELAY .01 SECONDS</li> <li>TOGGLE SPEAKER AT 1 KHZ FOR .1 SEC.</li> <li>CURSOR H INDEX TO Y-REG STORE CHAR IN LINE INCREMENT CURSOR H INDEX (MOVE RIGHT)</li> <li>BEYOND WINDOW WIDTH? YES CR TO NEXT LINE NO, RETURN</li> <li>CONTROL CHAR? NO, OUTPUT IT.</li> <li>INVERSE VIDEO? YES, OUTPUT IT.</li> <li>CR? YES.</li> <li>LINE FEED? IF SO, DO IT.</li> <li>BACK SPACE? (CNTRL-H) NO, CHECK FOR BELL.</li> <li>DECREMENT CURSOR H INDEX IF POS, OK. ELSE MOVE UP</li> </ul>
FBD0: FBD2: FBD2: FBD2: FBD3: FBD4: FBD4: FBD4: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBE4: FBE4: FBE4: FBE4: FBF6: FBF7: FBF7: FBF7: FBF7: FC01: FC02: FC04:	00A05500000000000000000000000000000000	28 28 28 28 28 28 28 28 28 28 28 20 28 28 20 28 28 20 28 28 28 28 28 28 28 28 28 28 28 28 28	FC CO	BELL1 BELL2 RTS2B STOADV ADVANCE RTS3 VIDOUT	ASL ASL ASL ORA STA RTS CMP BNE LDA JSR LDA JSR LDA JSR LDA DEY BNE RTS LDA STA LDA CMP BCS TAY BCS TAY BEQ CMP BEQ CMP BEQ CMP BEQ CMP BDE LDA	BASL BASL BASL #\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR BELL2 CH (BASL),Y CH CH WNDWDTH CR #\$A0 STOADV #\$8D CR #\$8A LF #\$88 LF #\$88 BELL1 CH RTS3 WNDWDTH	<ul> <li>BELL CHAR? (CNTRL-G) NO, RETURN DELAY .01 SECONDS</li> <li>TOGGLE SPEAKER AT 1 KHZ FOR .1 SEC.</li> <li>CURSOR H INDEX TO Y-REG STORE CHAR IN LINE INCREMENT CURSOR H INDEX (MOVE RIGHT)</li> <li>BEYOND WINDOW WIDTH? YES CR TO NEXT LINE NO, RETURN</li> <li>CONTROL CHAR? NO, OUTPUT IT.</li> <li>INVERSE VIDEO? YES, OUTPUT IT.</li> <li>CR? YES.</li> <li>LINE FEED? IF SO, DO IT.</li> <li>BACK SPACE? (CNTRL-H) NO, CHECK FOR BELL.</li> <li>DECREMENT CURSOR H INDEX IF POS, OK. ELSE MOVE UP SET CH TO WNDWDTH-1</li> </ul>
FBD0: FBD2: FBD2: FBD2: FBD3: FBD4: FBD4: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBE4: FBE4: FBE4: FBE4: FBE7: FBE7: FBF4: FBF4: FBF4: FBF7: FBF7: FC01: FC02: FC04: FC04: FC04: FC04: FC12: FC14: FC14: FC14:	00A05500000000000000000000000000000000	28 28 28 87 228 87 228 87 228 40 A8 COCA8 30 F5 228 224 224 226 60 E F 50 85 A 85 24 85 24 85 24 228 87 228 228	FC CO	BELL1 BELL2 RTS2B STOADV ADVANCE RTS3 VIDOUT	ASL ASL ASL ASL ASL ASL ASL ASL ASL ASL	BASL BASL BASL #\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR BELL2 CH (BASL),Y CH CH CH WNDWDTH CR #\$A0 STOADV #\$8D CR #\$8A LF #\$88 BELL1 CH RTS3 WNDWDTH CH	<ul> <li>BELL CHAR? (CNTRL-G) NO, RETURN DELAY .01 SECONDS</li> <li>TOGGLE SPEAKER AT 1 KHZ FOR .1 SEC.</li> <li>CURSOR H INDEX TO Y-REG STORE CHAR IN LINE INCREMENT CURSOR H INDEX (MOVE RIGHT)</li> <li>BEYOND WINDOW WIDTH? YES CR TO NEXT LINE NO,RETURN CONTROL CHAR? NO, OUTPUT IT.</li> <li>INVERSE VIDEO? YES, OUTPUT IT.</li> <li>INVERSE VIDEO? YES.</li> <li>LINE FEED? IF SO, DO IT.</li> <li>BACK SPACE? (CNTRL-H) NO, CHECK FOR BELL.</li> <li>DECREMENT CURSOR H INDEX IF POS, OK. ELSE MOVE UP SET CH TO WNDWDTH-1</li> </ul>
FBD0: FBD2: FBD2: FBD2: FBD2: FBD4: FBD4: FBD4: FBD4: FBD4: FBD5: FBD5: FBD5: FBE4: FBE4: FBE4: FBE4: FBE4: FBE4: FBE7: FBE7: FBF4: FBF6: FBF7: FBF7: FBF7: FC02: FC04:	00A05500000000000000000000000000000000	28 28 28 87 228 87 228 87 228 87 228 240 28 200 87 228 224 226 A0 EF ESDA8 5A8 5A8 5A8 224 224 226 A0 EF 228 228 228 228 228 228 228 228 228 22	FC CO	BELL1 BELL2 RTS2B STOADV ADVANCE RTS3 VIDOUT	ASL ASL ASL ASL ASL ASL ASL ASL ASL ASL	BASL BASL BASL #\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR BELL2 CH (BASL),Y CH CH (BASL),Y CH CH CH WNDWDTH CR #\$A0 STOADV #\$8D CR #\$88 BELL1 CH RTS3 WNDWDTH CH CH CH CH CH	<ul> <li>BELL CHAR? (CNTRL-G) NO, RETURN DELAY .01 SECONDS</li> <li>TOGGLE SPEAKER AT 1 KHZ FOR .1 SEC.</li> <li>CURSOR H INDEX TO Y-REG STORE CHAR IN LINE INCREMENT CURSOR H INDEX (MOVE RIGHT)</li> <li>BEYOND WINDOW WIDTH? YES CR TO NEXT LINE NO, RETURN</li> <li>CONTROL CHAR? NO, OUTPUT IT.</li> <li>INVERSE VIDEO? YES, OUTPUT IT.</li> <li>INVERSE VIDEO? YES.</li> <li>LINE FEED? IF SO, DO IT.</li> <li>BACK SPACE? (CNTRL-H) NO, CHECK FOR BELL.</li> <li>DECREMENT CURSOR H INDEX IF POS, OK. ELSE MOVE UP SET CH TO WNDWDTH-1</li> <li>(RIGHTMOST SCREEN POS)</li> </ul>
FBD0: FBD2: FBD2: FBD2: FBD2: FBD4: FBD4: FBD8: FBD7: FBD7: FBD7: FBE4: FBE4: FBE4: FBE4: FBE6: FBE7: FBF6: FBF7: FBF7: FBF7: FBF7: FBF7: FC01: FC04: FC04: FC04: FC04: FC14: FC14: FC14: FC14:	00A05500000000000000000000000000000000	28 28 28 87 12 40 87 28 87 12 40 87 28 87 22 87 87 87 87 87 87 87 87 87 87 87 87 87	FC CO	BELL1 BELL2 RTS2B STOADV ADVANCE RTS3 VIDOUT BS	ASL ASL ASL ASL ORA STA RTS CMP BNE LDA JSR LDA JSR LDA JSR LDA JSR LDA JSR LDA DEY BNE LDA STA INC LDA ECS RTS CMP BCS TAY BPL CMP BCS TAY BPL CMP BCS TAY BDE LDA LDA LDA LDA LDA LDA LDA LDA LDA LDA	BASL BASL BASL #\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR BELL2 CH (BASL),Y CH CH WNDWDTH CR #\$A0 STOADV #\$80 CR #\$80 STOADV #\$88 CR #\$88 BELL1 CH RTS3 WNDWDTH CH CH WNDTOP	<ul> <li>BELL CHAR? (CNTRL-G) NO, RETURN DELAY .01 SECONDS</li> <li>TOGGLE SPEAKER AT 1 KHZ FOR .1 SEC.</li> <li>CURSOR H INDEX TO Y-REG STORE CHAR IN LINE INCREMENT CURSOR H INDEX (MOVE RIGHT)</li> <li>BEYOND WINDOW WIDTH? YES CR TO NEXT LINE NO, RETURN</li> <li>CONTROL CHAR? NO, OUTPUT IT.</li> <li>INVERSE VIDEO? YES, OUTPUT IT.</li> <li>CR? YES.</li> <li>LINE FEED? IF SO, DO IT.</li> <li>BACK SPACE? (CNTRL-H) NO, CHECK FOR BELL.</li> <li>DECREMENT CURSOR H INDEX IF POS, OK. ELSE MOVE UP SET CH TO WNDWDTH-1</li> <li>(RIGHTMOST SCREEN POS) CURSOR V INDEX</li> </ul>
FBD0: FBD2: FBD2: FBD2: FBD3: FBD4: FBD4: FBD4: FBD4: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBE4: FBE4: FBE4: FBE4: FBE4: FBE7: FBF4: FBF6: FBF7: FC70:	00A05509090020A800416655009088090060000000000000000000000000	28 28 28 28 28 28 28 28 28 28 28 20 28 28 20 28 28 20 28 28 20 28 28 20 28 28 28 28 28 28 28 20 28 28 20 28 28 20 28 20 28 20 28 20 28 20 28 20 28 20 28 20 28 20 20 20 20 20 20 20 20 20 20 20 20 20	FC FC C0	BELL1 BELL2 RTS2B STOADV ADVANCE RTS3 VIDOUT BS	ASL ASL ASL ASL ORA STA RTS CMP BNE LDA JSR LDY LDA JSR LDY LDA JSR LDY BNE STA IDA STA INC LDA STA INC LDA CMP BCS TAY BEQ CMP BEQ CMP BEQ CMP BEQ CMP BEQ CMP ENE STA CMP LDA CMP CMP CMP CMP CMP CMP CMP CMP CMP CMP	BASL BASL BASL #\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR BELL2 CH (BASL),Y CH CH WNDWDTH CR #\$A0 STOADV #\$8D CR #\$8A LF #\$88 BELL1 CH RTS3 WNDWDTH CH CH WNDTOP CV	<ul> <li>BELL CHAR? (CNTRL-G) NO, RETURN DELAY .01 SECONDS</li> <li>TOGGLE SPEAKER AT 1 KHZ FOR .1 SEC.</li> <li>CURSOR H INDEX TO Y-REG STORE CHAR IN LINE INCREMENT CURSOR H INDEX (MOVE RIGHT)</li> <li>BEYOND WINDOW WIDTH? YES CT TO NEXT LINE NO, RETURN</li> <li>CONTROL CHAR? NO, OUTPUT IT.</li> <li>INVERSE VIDEO? YES.</li> <li>LINE FEED? IF SO, DO IT.</li> <li>BACK SPACE? (CNTRL-H) NO, CHECK FOR BELL.</li> <li>DECREMENT CURSOR H INDEX IF POS, OK. ELSE MOVE UP SET CH TO WNDWDTH-1</li> <li>(RIGHTMOST SCREEN POS) CURSOR V INDEX</li> </ul>

FClE:	в0	0B			BCS	RTS4	IF TOP LINE THEN RETURN
FC20:	C6	25			DEC	CV	DEC CURSOR V-INDEX
FC22:	A5	25	ਸ਼ਾਸ	VTAB	LDA	CV	GET CURSOR V-INDEX
FC24:	20	20	гЬ	VIADZ	ADC	WNDLFT	ADD WINDOW LEFT INDEX
FC29:	85	28			STA	BASL	TO BASL
FC2B:	60			RTS4	RTS		
FC2C:	49	C0		ESCl	EOR	#\$C0	ESC?
FC2E:	F0	28			BEQ	HOME	IF SO, DO HOME AND CLEAR
FC30:	69	FD			ADC	#\$FD	ESC-A OR B CHECK
FC32:	90	CU			BCC	ADVANCE	A, ADVANCE
FC36.	г U 6 9	DA FD				до #\$FD	ESC-C OR D CHECK
FC38:	90	2C			BCC	LF	C, DOWN
FC3A:	FO	DE			BEQ	UP	D, GO UP
FC3C:	69	FD			ADC	#\$FD	ESC-E OR F CHECK
FC3E:	90	5C			BCC	CLREOL	E, CLEAR TO END OF LINE
FC40:	DU D4	E9		CIDEOD	BNE	RTS4	NOT F, RETURN
FC42:	Δ5	24		CLREOP		CV	CURSOR V TO A-REGISTER
FC46:	48	20		CLEOP1	PHA		SAVE CURRENT LINE ON STK
FC47:	20	24	FC		JSR	VTABZ	CALC BASE ADDRESS
FC4A:	20	9E	FC		JSR	CLEOLZ	CLEAR TO EOL, SET CARRY
FC4D:	A0	00			LDY	#\$00	CLEAR FROM H INDEX=0 FOR REST
FC4F:	68	0.0			PLA	#¢00	INCREMENT CURRENT LINE
FC50:	09	23			CMD	#ŞUU WNDBUM	CARKI IS SEI)
FC54:	90	F0			BCC	CLEOP1	NO, KEEP CLEARING LINES
FC56:	в0	CA			BCS	VTAB	YES, TAB TO CURRENT LINE
FC58:	A5	22		HOME	LDA	WNDTOP	INIT CURSOR V
FC5A:	85	25			STA	CV	AND H-INDICES
FC5C:	A0	00			LDY	#\$00 GU	
FC5E:	84	24 124			STY	CH	THEN CLEAR TO END OF PAGE
FC62:	г 0 д 9	00		CR	T'DA	#\$00	CURSOR TO LEFT OF INDEX
FC64:	85	24		011	STA	CH	(RET CURSOR H=0)
FC66:	E6	25		LF	INC	CV	INCR CURSOR V(DOWN 1 LINE)
FC68:	A5	25			LDA	CV	
FC6A:	C5	23			CMP	WNDBTM	OFF SCREEN?
FC6C:	90 C6	80 25			DEC	VTABZ CV	NO, SET BASE ADDR DECP CURSOP V(BACK TO BOTTOM LINE)
FC70:	A5	22		SCROLL	LDA	WNDTOP	START AT TOP OF SCRL WNDW
FC72:	48				PHA		
FC73:	20	24	FC		JSR	VTABZ	GENERATE BASE ADR
FC76:	A5	28		SCRL1	LDA	BASL	COPY BASL,H
FC78:	85 75	2A 20			STA TDA	BASZL	TO BASZL, H
FC7C:	85	2 B			STA	BAS2H	
FC7E:	A4	21			LDY	WNDWDTH	INIT Y TO RIGHTMOST INDEX
FC80:	88				DEY		OF SCROLLING WINDOW
FC81:	68				PLA	" + 0 7	
FC82:	69 C5	01 23			ADC	#ŞUL WNDRTM	INCR LINE NUMBER
FC86:	B0	2 J 0 D			BCS	SCRL3	VES. FINISH
FC88:	48	00			PHA	Denillo	
FC89:	20	24	FC		JSR	VTABZ	FORM BASL,H (BASE ADDR)
FC8C:	Bl	28		SCRL2	LDA	(BASL),Y	MOVE A CHR UP ON LINE
FC8E:	91	2A			STA	(BAS2L),Y	
FC90:	88	ΨO			DEY	CODI 2	NEXT CHAR OF LINE
FC91:	30	гэ El			BMT	SCRL2	NEXT LINE (ALWAYS TAKEN)
FC95:	A0	00		SCRL3	LDY	#\$00	CLEAR BOTTOM LINE
FC97:	20	9E	FC		JSR	CLEOLZ	GET BASE ADDR FOR BOTTOM LINE
FC9A:	в0	86			BCS	VTAB	CARRY IS SET
FC9C:	A4	24		CLREOL	LDY	CH	CURSOR H INDEX
FC9E:	A9	A0 20		CLEOLZ	LDA	#ŞAU	CHODE DIANKS EDOM LHEDEL
FCAU:	C.8	20		CLEOLZ	TNY	(DASL),I	TO END OF LINES (WNDWDTH)
FCA3:		~ 1			CPY	WNDWDTH	10 200 01 21020 (0000010)
FCA5:	C4	21					
FCA7:	C4 90	21 F9			BCC	CLEOL2	
FCA8:	C4 90 60	21 F9			BCC RTS	CLEOL2	
ECNO.	C4 90 60 38	21 F9		WAIT	BCC RTS SEC	CLEOL2	
FCA9:	C4 90 60 38 48 E9	21 F9		WAIT WAIT2 WAIT3	BCC RTS SEC PHA SBC	CLEOL2	
FCA9: FCAA: FCAC:	C4 90 60 38 48 E9 D0	21 F9 01 FC		WAIT WAIT2 WAIT3	BCC RTS SEC PHA SBC BNE	CLEOL2 #\$01 WAIT3	1.0204 USEC
FCA9: FCAA: FCAC: FCAE:	C4 90 60 38 48 E9 D0 68	21 F9 01 FC		WAIT WAIT2 WAIT3	BCC RTS SEC PHA SBC BNE PLA	CLEOL2 #\$01 WAIT3	1.0204 USEC (13+2712*A+512*A*A)
FCA9: FCAA: FCAC: FCAE: FCAF:	C4 90 60 38 48 E9 D0 68 E9	21 F9 01 FC 01		WAIT WAIT2 WAIT3	BCC RTS SEC PHA SBC BNE PLA SBC	CLEOL2 #\$01 WAIT3 #\$01	1.0204 USEC (13+2712*A+512*A*A)
FCA9: FCAA: FCAC: FCAE: FCAF: FCB1:	C4 90 60 38 48 E9 D0 68 E9 D0 68	21 F9 01 FC 01 F6		WAIT WAIT2 WAIT3	BCC RTS SEC PHA SBC BNE PLA SBC BNE BNE	CLEOL2 #\$01 WAIT3 #\$01 WAIT2	1.0204 USEC (13+2712*A+512*A*A)
FCA9: FCAA: FCAC: FCAE: FCAF: FCB1: FCB3: FCB4.	C4 90 60 38 48 E9 D0 68 E9 D0 60 E6	21 F9 01 FC 01 F6 42		WAIT WAIT2 WAIT3	BCC RTS SEC PHA SBC BNE PLA SBC BNE RTS INC	CLEOL2 #\$01 WAIT3 #\$01 WAIT2 A41	1.0204 USEC (13+2712*A+512*A*A)
FCA9: FCAA: FCAC: FCAE: FCAF: FCB1: FCB3: FCB4: FCB6:	C4 90 60 38 48 E9 D0 68 E9 D0 60 E6 D0	21 F9 01 FC 01 F6 42 02		WAIT WAIT2 WAIT3 NXTA4	BCC RTS SEC PHA SBC BNE PLA SBC BNE RTS INC BNE	CLEOL2 #\$01 WAIT3 #\$01 WAIT2 A4L NXTA1	1.0204 USEC (13+2712*A+512*A*A) INCR 2-BYTE A4 AND A1
FCA9: FCAA: FCAC: FCAE: FCAF: FCB1: FCB3: FCB4: FCB6: FCB8:	C4 90 38 48 D0 68 D0 60 E6 D0 E6	21 F9 01 FC 01 F6 42 02 43		WAIT WAIT2 WAIT3 NXTA4	BCC RTS SEC PHA SBC BNE PLA SBC BNE RTS INC BNE INC	CLEOL2 #\$01 WAIT3 #\$01 WAIT2 A4L NXTA1 A4H	1.0204 USEC (13+2712*A+512*A*A) INCR 2-BYTE A4 AND A1
FCA9: FCAA: FCAC: FCAE: FCAF: FCB1: FCB3: FCB4: FCB6: FCB8: FCBA:	C4 90 60 38 48 D0 68 E9 D0 60 E6 D0 E6 D0 E6 S	21 F9 01 FC 01 F6 42 02 43 3C		WAIT WAIT2 WAIT3 NXTA4 NXTA1	BCC RTS SEC PHA SBC BNE PLA SBC BNE RTS INC BNE INC LDA	CLEOL2 #\$01 WAIT3 #\$01 WAIT2 A4L NXTA1 A4H A1L	1.0204 USEC (13+2712*A+512*A*A) INCR 2-BYTE A4 AND A1 INCR 2-BYTE A1.
FCA9: FCAA: FCAC: FCAF: FCB1: FCB3: FCB4: FCB6: FCB8: FCBA: FCBA: FCBC: FCBE:	C4 90 38 49 D6 20 60 E0 E0 E0 E0 E0 E5 E5 E5	21 F9 01 FC 01 F6 42 02 43 3C 3E 3D		WAIT WAIT2 WAIT3 NXTA4 NXTA1	BCC RTS SEC PHA SBC BNE PLA SBC BNE INC LDA	CLEOL2 #\$01 WAIT3 #\$01 WAIT2 A4L NXTA1 A4H A1L A2L A1H	1.0204 USEC (13+2712*A+512*A*A) INCR 2-BYTE A4 AND A1 INCR 2-BYTE A1. AND COMPARE TO A2

FCC0:	E5	3F			SBC	A2H	
FCC2:	E6	3C			TNC	Alt.	(CARRY SET IF >=)
FCC4.		02			DNE		(email bhi ii > )
FCC4:	00	02			DNC	RI54D	
FCC6:	E6	3D			INC	AIH	
FCC8:	60			RTS4B	RTS		
FCC9:	A0	4B		HEADR	LDY	#\$4B	WRITE A*256 'LONG l'
FCCB:	20	DB	FC		TSR	ZERDLY	HALF CYCLES
FCCF.	<u>۵</u>	F0	10		DNF	UENDD	(650 USEC EACH)
FCCE.	00	1 7			DNE		(050 OBEC EACH)
FCD0:	69	F.F.			ADC	#ŞFE	
FCD2:	в0	F5			BCS	HEADR	THEN A 'SHORT 0'
FCD4:	A0	21			LDY	#\$21	(400 USEC)
FCD6:	20	DB	FC	WRBTT	JSR	ZERDLY	WRITE TWO HALF CYCLES
FCDQ.	C 8	22			TNV		OF 250 IISEC (101)
TCDJ.	20				1 N 1 7 N 17		$\begin{array}{c} \text{OP}  \text{ZOO}  \text{USEC}  (  \text{O}  ) \\ \text{OP}  \text{EOO}  \text{USEC}  (  \text{O}  ) \end{array}$
FCDA:	6.8				ΤΝΥ		OR 500 OSEC (0.0)
FCDB:	88			ZERDLY	DEY		
FCDC:	D0	FD			BNE	ZERDLY	
FCDE:	90	05			BCC	WRTAPE	Y IS COUNT FOR
FCF0.	20	32			TUN	#\$32	TIMING LOOP
FCEO.	AU	52				πγυζ	TIMING DOOP
FCE2:	88			ONEDLY	DEY		
FCE3:	D0	FD			BNE	ONEDLY	
FCE5:	AC	20	C0	WRTAPE	LDY	TAPEOUT	
FCE8.	<u>م</u>	20			T.DY	#\$2C	
FCEN.	07	20			DEV	1920	
FCEA:	CA				DEA		
FCEB:	60				RTS		
FCEC:	A2	08		RDBYTE	LDX	#\$08	8 BITS TO READ
FCEE:	48			RDBYT2	PHA		READ TWO TRANSITIONS
FCFF.	20	FΔ	FC		TCD	יידפרס	(FIND FDCF)
FCEF.	20	ΓA	гC		DDA	RDZDII	(FIND EDGE)
FCF2:	68				PLA		
FCF3:	2A				ROL		NEXT BIT
FCF4:	A0	3A			LDY	#\$3A	COUNT FOR SAMPLES
FCF6:	CA				DEX		
FCF7.	D0	τ			DNE	משעתת	
FCF/:	00	гэ			DNE	RUDIIZ	
FCF9:	60				RTS		
FCFA:	20	FD	FC	RD2BIT	JSR	RDBIT	
FCFD:	88			RDBIT	DEY		DECR Y UNTIL
FCFE:	AD	60	CO		T.DA	ΤΑΡΕΙΝ	TAPE TRANSTITON
ED01.	1 5	25	00		FOD	тасшти	
FDUI:	40	21			LOK	DADIIN	
FD03:	ΤU	F.8			BLT	RDBIT	
FD05:	45	2F			EOR	LASTIN	
FD07:	85	2F			STA	LASTIN	
FD09:	C 0	80			CPY	#\$80	SET CARRY ON Y
EDOD.	60	00			DI	100	ber omder og i
FDUB:	00	~ .			RIS		
FDUC:	A4	24		RDKEY	LDY	CH	
FDOE •	B1	28					
TDOD.					LDA	(BASL),Y	SET SCREEN TO FLASH
FD10:	48				PHA	(BASL),Y	SET SCREEN TO FLASH
FD10: FD11:	48 29	3F			DDA PHA AND	(BASL),Y #\$3F	SET SCREEN TO FLASH
FD10: FD11: FD13:	48 29	3F			DA PHA AND	(BASL),Y #\$3F #\$40	SET SCREEN TO FLASH
FD10: FD11: FD13:	48 29 09	3F 40			DDA PHA AND ORA	(BASL),Y #\$3F #\$40	SET SCREEN TO FLASH
FD10: FD11: FD13: FD15:	48 29 09 91	3F 40 28			LDA PHA AND ORA STA	(BASL),Y #\$3F #\$40 (BASL),Y	SET SCREEN TO FLASH
FD10: FD11: FD13: FD15: FD17:	48 29 09 91 68	3F 40 28			LDA PHA AND ORA STA PLA	(BASL),Y #\$3F #\$40 (BASL),Y	SET SCREEN TO FLASH
FD10: FD11: FD13: FD15: FD17: FD18:	48 29 09 91 68 6C	3F 40 28 38	00		LDA PHA AND ORA STA PLA JMP	(BASL),Y #\$3F #\$40 (BASL),Y (KSWL)	GO TO USER KEY-IN
FD10: FD11: FD13: FD15: FD17: FD18: FD18:	48 29 09 91 68 6C E6	3F 40 28 38 4E	00	KEYIN	LDA PHA AND ORA STA PLA JMP INC	(BASL),Y #\$3F #\$40 (BASL),Y (KSWL) RNDL	GO TO USER KEY-IN
FD10: FD11: FD13: FD15: FD15: FD17: FD18: FD18: FD10:	48 29 09 91 68 6C E6	3F 40 28 38 4E 02	00	KEYIN	LDA PHA AND ORA STA PLA JMP INC BNE	(BASL),Y #\$3F #\$40 (BASL),Y (KSWL) RNDL KEYIN2	GO TO USER KEY-IN INCE END NUMBER
FD10: FD11: FD13: FD13: FD15: FD17: FD18: FD18: FD1D:	48 29 09 91 68 6C E6 D0	3F 40 28 38 4E 02	00	KEYIN	LDA PHA AND ORA STA PLA JMP INC BNE INC	(BASL),Y #\$3F #\$40 (BASL),Y (KSWL) RNDL KEYIN2 NDU	GO TO USER KEY-IN INCR RND NUMBER
FD10: FD11: FD13: FD15: FD15: FD17: FD18: FD18: FD1B: FD1D:	48 29 09 91 68 6C E6 D0 E6	3F 40 28 38 4E 02 4F	00	KEYIN	LDA PHA AND ORA STA PLA JMP INC BNE INC	(BASL),Y #\$3F #\$40 (BASL),Y (KSWL) RNDL KEYIN2 RNDH	GO TO USER KEY-IN INCR RND NUMBER
FD10: FD11: FD13: FD15: FD17: FD18: FD18: FD18: FD1B: FD1D: FD1F: FD21:	48 29 09 91 68 6C E6 D0 E6 2C	3F 40 28 38 4E 02 4F 00	00 C0	KEYIN KEYIN2	LDA PHA AND ORA STA PLA JMP INC BNE INC BIT	(BASL),Y #\$3F #\$40 (BASL),Y (KSWL) RNDL KEYIN2 RNDH KBD	GO TO USER KEY-IN INCR RND NUMBER KEY DOWN?
FD10: FD11: FD13: FD15: FD17: FD18: FD18: FD18: FD1B: FD1D: FD1F: FD21: FD24:	48 29 09 91 68 6C E6 D0 E6 2C 10	3F 40 28 38 4E 02 4F 00 F5	00 C0	KEYIN KEYIN2	LDA PHA AND ORA STA PLA JMP INC BNE INC BIT BPL	(BASL),Y #\$3F #\$40 (BASL),Y (KSWL) RNDL KEYIN2 RNDH KBD KEYIN	GO TO USER KEY-IN INCR RND NUMBER KEY DOWN? LOOP
FD10: FD11: FD13: FD15: FD15: FD18: FD18: FD18: FD18: FD11: FD11: FD21: FD24: FD26:	48 29 91 68 6C E6 D0 E6 2C 10 91	3F 40 28 38 4E 02 4F 00 F5 28	00 C0	KEYIN KEYIN2	LDA PHA AND ORA STA PLA JMP INC BNE INC BIT BIT BPL STA	(BASL),Y #\$3F #\$40 (BASL),Y (KSWL) RNDL KEYIN2 RNDH KBD KEYIN (BASL),Y	GO TO USER KEY-IN INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN
FD10: FD11: FD13: FD15: FD17: FD18: FD18: FD18: FD18: FD1F: FD21: FD24: FD26: FD28:	48 29 91 68 6C E6 D0 E6 2C 10 91 AD	3F 40 28 38 4E 02 4F 00 F5 28 00	00 C0	KEYIN KEYIN2	LDA PHA AND ORA STA PLA JMP INC BNE INC BIT BPL STA LDA	(BASL),Y #\$3F #\$40 (BASL),Y (KSWL) RNDL KEYIN2 RNDH KBD KEYIN (BASL),Y KBD	GO TO USER KEY-IN INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE
FD10: FD11: FD13: FD15: FD17: FD18: FD18: FD18: FD19: FD21: FD24: FD24: FD28: FD28:	48 29 91 68 6C E6 2C 10 91 AD	3F 40 28 38 4E 02 4F 00 F5 28 00	00 C0 C0	KEYIN KEYIN2	LDA PHA AND ORA STA PLA JMP INC BNE INC BIT BPL STA LDA BIT	(BASL),Y #\$3F #\$40 (BASL),Y (KSWL) RNDL KEYIN2 RNDH KBD KEYIN (BASL),Y KBD KEDB KEDB KEDB KEDB KEDB	GO TO USER KEY-IN INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE CLP KEY STROBE
FD10: FD11: FD13: FD15: FD17: FD18: FD18: FD18: FD18: FD19: FD21: FD22: FD24: FD28: FD28:	48 29 91 68 6C E6 D0 E6 2C 91 AD 2C	3F 40 28 38 4E 02 4F 00 F5 28 00 10	00 C0 C0 C0	KEYIN KEYIN2	LDA PHA AND ORA STA PLA JMP INC BNE INC BIT BIT STA LDA BIT	(BASL),Y #\$3F #\$40 (BASL),Y (KSWL) RNDL KEYIN2 RNDH KBD KEYIN (BASL),Y KBD KBDSTRB	GO TO USER KEY-IN INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE CLR KEY STROBE
FD10: FD11: FD13: FD15: FD15: FD18: FD18: FD18: FD21: FD24: FD24: FD28: FD28: FD28:	48 29 91 68 6C E6 D0 E6 2C 91 AD 2C 0	3F 40 28 38 4E 02 4F 00 F5 28 00 10	00 C0 C0 C0	KEYIN KEYIN2	LDA PHA AND ORA STA PLA JMP INC BNE INC BIT STA LDA BIT RTS	(BASL),Y #\$3F #\$40 (BASL),Y (KSWL) RNDL KEYIN2 RNDH KBD KEYIN (BASL),Y KBD KBDSTRB	GO TO USER KEY-IN INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE CLR KEY STROBE
FD10: FD13: FD13: FD15: FD17: FD18: FD18: FD18: FD21: FD24: FD24: FD28: FD28: FD28: FD28: FD28:	48 29 91 68 6C E6 2C 91 AD 2C 60 20	3F 40 28 38 4E 02 4F 00 F5 28 00 10 0C	00 C0 C0 C0 FD	KEYIN KEYIN2 ESC	LDA PHA AND ORA STA PLA JMP INC BNE INC BIT BPL STA LDA BIT RTS JSR	(BASL),Y #\$3F #\$40 (BASL),Y (KSWL) RNDL KEYIN2 RNDH KBD KEYIN (BASL),Y KBD KBDSTRB RDKEY	GO TO USER KEY-IN INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE GET KEYCODE
FD10: FD11: FD13: FD15: FD17: FD18: FD18: FD18: FD21: FD22: FD28: FD28: FD28: FD28: FD28: FD28: FD28: FD28: FD28: FD28:	48 29 91 68 6C 20 91 8 6C 20 91 20 20	3F 40 28 38 4E 02 4F 00 F5 28 00 10 0C 2C	00 C0 C0 FD FC	KEYIN KEYIN2 ESC	LDA PHA AND ORA STA PLA JMP INC BNE INC BNE INC BNE INC BNE INC BIT LDA BIT RTS JSR JSR	(BASL),Y #\$3F #\$40 (BASL),Y (KSWL) RNDL KEYIN2 RNDH KBD KEYIN (BASL),Y KBD KBDSTRB RDKEY ESC1	GO TO USER KEY-IN INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE CLR KEY STROBE GET KEYCODE HANDLE ESC FUNC.
FD10: FD11: FD13: FD15: FD17: FD18: FD18: FD18: FD24: FD24: FD28:	48 29 91 68 6C E6 D0 E6 2C 10 91 AD 2C 60 20 20	3F 40 28 38 4E 02 4F 00 F5 28 00 10 0C 2C 0C	00 C0 C0 FD FC FD	KEYIN KEYIN2 ESC RDCHAR	LDA PHA AND ORA STA PLA JMP INC BNE INC BIT STA LDA BIT RTS JSR JSR JSR	(BASL),Y #\$3F #\$40 (BASL),Y (KSWL) RNDL KEYIN2 RNDH KBD KEYIN (BASL),Y KBD KBDSTRB RDKEY ESC1 RDKEY	GO TO USER KEY-IN INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE CLR KEY STROBE GET KEYCODE HANDLE ESC FUNC. READ KEY
FD10: FD13: FD13: FD15: FD17: FD18: FD18: FD18: FD24: FD24: FD28: FD28: FD28: FD28: FD32: FD35:	429 991 68C 66C 106 200 91D 200 200 200 200 200	3F 40 28 38 4E 02 4F 00 F5 28 00 10 0C 2C 0C 9B	00 C0 C0 FD FC FD	KEYIN KEYIN2 ESC RDCHAR	LDA PHA AND ORA STA PLA JMP INC BNE INC BIT BPL STA LDA BIT RTS JSR JSR JSR CMD	(BASL),Y #\$3F #\$40 (BASL),Y (KSWL) RNDL KEYIN2 RNDH KBD KEYIN (BASL),Y KBD KBDSTRB RDKEY ESC1 RDKEY ESC1 RDKEY #\$9B	GO TO USER KEY-IN INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE CLR KEY STROBE GET KEYCODE HANDLE ESC FUNC. READ KEY ESC2
FD10: FD11: FD13: FD15: FD17: FD18: FD18: FD18: FD21: FD22: FD28: FD28: FD28: FD28: FD28: FD28: FD27: FD35: FD35:	48 29 91 68 6C 60 20 20 20 20 20 20 20 20 20 20	3F 40 28 38 4E 02 4F 00 F5 28 00 10 0C 2C 9B	00 C0 C0 FD FC FD	KEYIN KEYIN2 ESC RDCHAR	LDA PHA AND ORA STA PLA JMP INC BNE INC BNE INC BNE INC BNE INC BTA LDA BIT STA LDA BIT STA JSR JSR JSR DSR ORA	(BASL),Y #\$3F #\$40 (BASL),Y (KSWL) RNDL KEYIN2 RNDH KBD KEYIN (BASL),Y KBD KBDSTRB RDKEY ESC1 RDKEY #\$9B	GO TO USER KEY-IN INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE CLR KEY STROBE GET KEYCODE HANDLE ESC FUNC. READ KEY ESC?
FD10: FD11: FD13: FD15: FD17: FD18: FD18: FD18: FD21: FD24: FD24: FD28: FD28: FD28: FD28: FD28: FD28: FD35: FD33:	4 8 29 91 68 6C 20 91 8 6C 20 91 20 20 20 20 91 8 7 10 20 20 20 91 8 10 20 20 20 20 20 20 91 10 20 20 20 20 20 20 20 20 20 20 20 20 20	3F 40 28 38 4E 02 4F 00 55 28 00 10 0C 2C 0C 9B F3	00 C0 C0 FD FC FD	KEYIN KEYIN2 ESC RDCHAR	LDA PHA AND ORA STA PLA JMP INC BNE INC BNE INC BIT STA LDA BIT RTS JSR JSR JSR CMP BEQ	(BASL),Y #\$3F #\$40 (BASL),Y (KSWL) RNDL KEYIN2 RNDH KBD KEYIN (BASL),Y KBD KBDSTRB RDKEY #\$9B ESC	GO TO USER KEY-IN INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE CLR KEY STROBE GET KEYCODE HANDLE ESC FUNC. READ KEY ESC? YES, DON'T RETURN
FD10: FD13: FD13: FD15: FD17: FD18: FD18: FD17: FD24: FD24: FD24: FD24: FD28: FD28: FD28: FD28: FD35: FD35: FD38: FD38:	29 91 68 6C 2C 10 91 AD 20 20 20 50 60	3F 40 28 38 4E 02 4F 00 F5 28 00 10 0C 2C 0C 9B F3	00 C0 C0 FD FC FD	KEYIN KEYIN2 ESC RDCHAR	LDA PHA AND ORA STA PLA JMP INC BNE INC BIT BPL STA LDA BIT RTS JSR JSR JSR CMP BEQ RTS	(BASL),Y #\$3F #\$40 (BASL),Y (KSWL) RNDL KEYIN2 RNDH KBD KEYIN (BASL),Y KBD KBDSTRB RDKEY ESC1 RDKEY #\$9B ESC	GO TO USER KEY-IN INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE CLR KEY STROBE GET KEYCODE HANDLE ESC FUNC. READ KEY ESC? YES, DON'T RETURN
FD10: FD11: FD13: FD15: FD17: FD18: FD18: FD18: FD21: FD24: FD28: FD28: FD28: FD28: FD28: FD28: FD28: FD38: FD38: FD38: FD38: FD38: FD38:	4 29 91 68 60 60 20 91 AD 20 20 20 F0 A5	3F 40 28 38 4E 02 4F 00 F5 28 00 10 0C 2C 0C 9B F3 32	00 C0 C0 FD FC FD	KEYIN KEYIN2 ESC RDCHAR NOTCR	LDA PHA AND ORA STA PLA JMP INC BNE INC BNE INC BNE INC BNE INC BTA LDA BIT JSR JSR JSR JSR JSR CMP BEQ RTS LDA	(BASL),Y #\$3F #\$40 (BASL),Y (KSWL) RNDL KEYIN2 RNDH KBD KEYIN (BASL),Y KBD KBDSTRB RDKEY ESC1 RDKEY #\$9B ESC INVFLG	GO TO USER KEY-IN INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE CLR KEY STROBE GET KEYCODE HANDLE ESC FUNC. READ KEY ESC? YES, DON'T RETURN
FD10: FD11: FD13: FD15: FD17: FD18: FD18: FD18: FD21: FD24: FD24: FD28: FD28: FD28: FD28: FD28: FD37: FD35: FD33: FD33: FD37:	4829 991 68C 66 20 20 20 20 50 60 548 548 548 548 548 548 548 548 548 548	3F 40 28 38 4E 02 4F 00 F5 28 00 10 0C 2C 9B F3 32	00 C0 C0 FD FC FD	KEYIN KEYIN2 ESC RDCHAR NOTCR	LDA PHA AND ORA STA PLA JMP INC BNE INC BIT STA LDA BIT RTS JSR JSR JSR JSR CMP BEQ RTS LDA PHA	(BASL),Y #\$3F #\$40 (BASL),Y (KSWL) RNDL KEYIN2 RNDH KBD KEYIN (BASL),Y KBD KBDSTRB RDKEY ESC1 RDKEY #\$9B ESC INVFLG	GO TO USER KEY-IN INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE CLR KEY STROBE GET KEYCODE HANDLE ESC FUNC. READ KEY ESC? YES, DON'T RETURN
FD10: FD11: FD13: FD15: FD17: FD18: FD18: FD18: FD21: FD24: FD24: FD26: FD28: FD28: FD28: FD37: FD38: FD37: FD37: FD37:	4 2 9 9 1 8 6 C 6 0 9 A C 0 2 0 9 1 8 6 C 6 0 2 C 0 9 A C 0 2 0 9 F 0 0 A 5 8 4 8 9 4 5 6 7 5 6 7 5 6 7 5 6 7 5 6 7 5 6 7 5 7 5	3F 40 28 38 4E 02 4F 00 55 28 00 10 0C 2C 9B F3 32 FF	00 C0 C0 FD FC FD	KEYIN KEYIN2 ESC RDCHAR NOTCR	LDA PHA AND ORA STA PLA JMP INC BNE INC BIT BNE INC BIT BPL STA LDA BIT RTS JSR JSR JSR CMP BEQ RTS LDA PHA	(BASL),Y #\$3F #\$40 (BASL),Y (KSWL) RNDL KEYIN2 RNDH KBD KEYIN (BASL),Y KBD KBDSTRB RDKEY ESC1 RDKEY #\$9B ESC INVFLG #\$5FF	GO TO USER KEY-IN INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE CLR KEY STROBE GET KEYCODE HANDLE ESC FUNC. READ KEY ESC? YES, DON'T RETURN
FD10: FD11: FD13: FD15: FD17: FD18: FD18: FD18: FD21: FD24: FD24: FD28: FD28: FD28: FD28: FD28: FD38: FD38: FD38: FD30: FD30: FD37: FD30: FD37:	4899188C606E0062009180C000000000000000000000000000000000	3F 40 28 38 4E 02 4F 00 F5 28 00 10 0C 2C 9B F3 32 FF	00 C0 C0 FD FC FD	KEYIN KEYIN2 ESC RDCHAR NOTCR	LDA PHA AND ORA STA PLA JMP INC BNE INC BNE INC BNE INC BNE INC BTA LDA BIT STA LDA BIT STA JSR JSR JSR JSR JSR JSR DSR JSR DSR DSR DSR DSR DSR DSR DSR DSR DSR D	(BASL),Y #\$3F #\$40 (BASL),Y (KSWL) RNDL KEYIN2 RNDH KEYIN (BASL),Y KBD KBDSTRB RDKEY ESC1 RDKEY ESC1 RDKEY #\$9B ESC INVFLG #\$FF	GO TO USER KEY-IN INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE CLR KEY STROBE GET KEYCODE HANDLE ESC FUNC. READ KEY ESC? YES, DON'T RETURN
FD10: FD11: FD13: FD15: FD17: FD18: FD18: FD26: FD27: FD28: FD28: FD28: FD28: FD28: FD28: FD28: FD37: FD37: FD32: FD37:	48999188C66062009191086C60000000000000000000000000000000000	3F 40 28 38 4E 02 4F 02 28 00 10 0C 2C 0C 9B 73 32 FF 32	00 C0 C0 FD FC FD	KEYIN KEYIN2 ESC RDCHAR NOTCR	LDA PHA AND ORA STA PLA JMP INC BNE INC BNE INC BIT STA LDA BIT RTS JSR JSR JSR JSR CMP BEQ RTS LDA PHA LDA STA	(BASL),Y #\$3F #\$40 (BASL),Y (KSWL) RNDL KEYIN2 RNDH KBD KEYIN1 (BASL),Y KBD KBDSTRB RDKEY #\$9B ESC INVFLG #\$FF INVFLG	GO TO USER KEY-IN INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE CLR KEY STROBE GET KEYCODE HANDLE ESC FUNC. READ KEY ESC? YES, DON'T RETURN ECHO USER LINE
FD10: FD11: FD13: FD15: FD17: FD18: FD18: FD18: FD24: FD24: FD26: FD28: FD28: FD28: FD28: FD37: FD38: FD37: FD47:	48999186666006600000000000000000000000000	3F 40 28 38 4E 00 F5 28 00 10 0C 2C 0C 9B F3 32 FF 32 00	00 C0 C0 FD FC FD	KEYIN KEYIN2 ESC RDCHAR NOTCR	LDA PHA AND ORA STA PLA JMP INC BNE INC BIT BPL STA LDA BIT RTS JSR JSR JSR JSR JSR LDA BEQ RTS LDA LDA STA LDA	<pre>(BASL),Y #\$3F #\$40 (BASL),Y (KSWL) RNDL KEYIN2 RNDH KBD KEYIN (BASL),Y KBD KBDSTRB RDKEY ESC1 RDKEY #\$9B ESC INVFLG #\$FF INVFLG IN,X</pre>	GO TO USER KEY-IN INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE CLR KEY STROBE GET KEYCODE HANDLE ESC FUNC. READ KEY ESC? YES, DON'T RETURN ECHO USER LINE NON INVERSE
FD10: FD11: FD13: FD15: FD17: FD18: FD18: FD18: FD18: FD28: FD28: FD28: FD28: FD28: FD28: FD28: FD38: FD38: FD38: FD37: FD37: FD37: FD40: FD44: FD44: FD44: FD44:	4899168C66002009168C6002000000000000000000000000000000000	3F 40 28 38 4E 00 F5 28 00 10 0C 2C 0C 9B F3 32 FF 32 00 ED	000 C0 FD FC FD FD	KEYIN KEYIN2 ESC RDCHAR NOTCR	LDA PHA AND ORA STA PLA JMP INC BNE INC BNE INC BNE STA LDA BIT STA LDA BIT STA JSR JSR JSR JSR JSR JSR LDA PHA LDA STA LDA JSR	(BASL),Y #\$3F #\$40 (BASL),Y (KSWL) RNDL KEYIN2 RNDH KBD KEYIN (BASL),Y KBD KBDSTRB RDKEY ESC1 RDKEY ESC1 RDKEY #\$9B ESC INVFLG #\$FF INVFLG IN,X COUT	GO TO USER KEY-IN INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE CLR KEY STROBE GET KEYCODE HANDLE ESC FUNC. READ KEY ESC? YES, DON'T RETURN ECHO USER LINE NON INVERSE
FD10: FD11: FD13: FD15: FD17: FD18: FD18: FD18: FD24: FD24: FD28: FD28: FD28: FD28: FD28: FD28: FD37: FD47:	4899168C66002009168C66002009168C660000000000000000000000000000000000	3F 40 28 38 4E 02 4F 00 F5 28 00 10 0C 2C 0C 9B F3 32 FF 32 00 ED	000 C0 C0 FD FC FD	KEYIN KEYIN2 ESC RDCHAR NOTCR	LDA PHA AND ORA STA PLA AND ORA STA DINC BNE INC BNE INC BNE INC BIT STA LDA BIT RTS JSR JSR CMP BEQ RTS LDA STA LDA STA LDA STA LDA STA	(BASL),Y #\$3F #\$40 (BASL),Y (KSWL) RNDL KEYIN2 RNDH KBD KEYIN1 (BASL),Y KBD KBDSTRB RDKEY #\$9B ESC INVFLG INVFLG IN,X COUT	GO TO USER KEY-IN INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE CLR KEY STROBE GET KEYCODE HANDLE ESC FUNC. READ KEY ESC? YES, DON'T RETURN ECHO USER LINE NON INVERSE
FD10: FD11: FD13: FD15: FD17: FD18: FD18: FD18: FD26: FD27: FD28: FD28: FD28: FD28: FD28: FD28: FD28: FD38: FD38: FD30: FD30: FD30: FD37: FD37: FD37: FD37: FD37: FD40: FD40: FD44: FD47: FD44: FD47: FD44: FD47:	4899168C660000000000000000000000000000000000	3F 40 28 38 4E 00 F5 28 00 10 0C 2C 0C 9B F3 32 FF 32 00 ED	00 C0 C0 FD FC FD 02 FD	KEYIN2 KEYIN2 ESC RDCHAR NOTCR	LDA PHA AND ORA STA PLA JMP INC BIT INC BIT BNE INC BIT BPL JST JSR JSR JSR JSR JSR LDA BIT RTS JSR JSR LDA PHA LDA STA LDA STA STA	<pre>(BASL),Y #\$3F #\$40 (BASL),Y (KSWL) RNDL KEYIN2 RNDH KBD KEYIN (BASL),Y KBD KBDSTRB RDKEY ESC1 RDKEY #\$99B ESC INVFLG #\$FF INVFLG IN,X COUT INVELC</pre>	GO TO USER KEY-IN INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE CLR KEY STROBE GET KEYCODE HANDLE ESC FUNC. READ KEY ESC? YES, DON'T RETURN ECHO USER LINE NON INVERSE
FD10: FD11: FD13: FD15: FD17: FD18: FD18: FD18: FD18: FD28: FD28: FD28: FD28: FD28: FD28: FD28: FD28: FD38: FD37: FD37: FD37: FD37: FD37: FD40: FD44: FD44: FD44: FD44: FD44: FD44:	48999186C66D06C2099085 68C66D06C209005 68C66D06C2009005 68C66D06C2009005 68C66D06C2009005 68C60005 68C60005 68C60005 68C60005 68C60005 68C60005 68C60005 68C60005 68C60005 68C60005 68C60005 68C60005 68C60005 68C6005 68C6005 68C6005 68C005 88C005 88	3F 40 28 38 4E 02 4F 00 55 28 00 10 0C 2C 0C 9B 53 32 FF 32 20 ED	000 C0 FD FC FD FD 02 FD	KEYIN KEYIN2 ESC RDCHAR NOTCR	LDA PHA AND ORA STA PLA JMP INC BNE INC BNE INC BNE INC BNE INC BNE INC STA LDA STA LDA PHA LDA STA LDA STA LDA STA STA STA	(BASL),Y #\$3F #\$40 (BASL),Y (KSWL) RNDL KEYIN2 RNDH KEYIN2 (BASL),Y KBD KBDSTRB RDKEY ESC1 RDKEY ESC1 RDKEY #\$9B ESC INVFLG INVFLG INVFLG	GO TO USER KEY-IN INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE CLR KEY STROBE GET KEYCODE HANDLE ESC FUNC. READ KEY ESC? YES, DON'T RETURN ECHO USER LINE NON INVERSE
FD10: FD11: FD13: FD15: FD17: FD18: FD18: FD18: FD24: FD24: FD28: FD28: FD28: FD28: FD28: FD28: FD28: FD28: FD28: FD37: FD37: FD37: FD37: FD37: FD37: FD37: FD37: FD44: FD44: FD44: FD48: FD48: FD48:	48999186C66062009180000000000000000000000000000000000	3F 40 28 38 4E 02 4F 00 27 55 28 00 10 0C 2C 0C 9B F 3 2 00 ED 32 00	000 C0 C0 FD FC FD 02 FD 02	KEYIN2 KEYIN2 ESC RDCHAR NOTCR	LDA PHA AND ORA STA PLA STA JMP INC BNE INC BNE INC BIT RTS JSR JSR JSR JSR LDA BIT RTS JSR LDA STA LDA STA LDA STA LDA	(BASL),Y #\$3F #\$40 (BASL),Y (KSWL) RNDL KEYIN2 RNDH KBD KEYIN2 (BASL),Y KBD KBDSTRB RDKEY #\$9B ESC INVFLG IN,X COUT INVFLG IN,X	GO TO USER KEY-IN INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE CLR KEY STROBE GET KEYCODE HANDLE ESC FUNC. READ KEY ESC? YES, DON'T RETURN ECHO USER LINE NON INVERSE
FD10: FD11: FD13: FD15: FD17: FD18: FD18: FD18: FD24: FD24: FD26: FD28: FD28: FD28: FD28: FD28: FD28: FD38: FD37: FD38: FD30: FD37: FD37: FD38: FD37: FD38: FD37: FD37: FD40: FD44: FD45:	48999186C66D0622099085E000585D9	3F 40 28 38 4E 02 4F 00 55 28 00 10 0C 20C 9B F3 32 F52 00 ED 32 00 88	00 C0 C0 FD FC FD 02 FD 02 FD	KEYIN2 KEYIN2 ESC RDCHAR NOTCR	LDA PHA AND ORA STA PLA JMP INC BNE INC BIT BPL INC BIT BPL JSR JSR JSR JSR JSR JSR JSR LDA BIT RTS JSR JSR LDA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA	<pre>(BASL),Y #\$3F #\$40 (BASL),Y (KSWL) RNDL KEYIN2 RNDH KBD KEYIN (BASL),Y KBD KBDSTRB RDKEY ESC1 RDKEY #\$99B ESC INVFLG #\$FF INVFLG IN,X COUT INVFLG IN,X #\$88</pre>	GO TO USER KEY-IN INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE CLR KEY STROBE GET KEYCODE HANDLE ESC FUNC. READ KEY ESC? YES, DON'T RETURN ECHO USER LINE NON INVERSE
FD10: FD11: FD13: FD15: FD17: FD18: FD18: FD18: FD24: FD24: FD28: FD28: FD28: FD28: FD28: FD28: FD28: FD28: FD38: FD38: FD37: FD37: FD37: FD40: FD40: FD44: FD45: FD55:	4299186C66D6C2019DCC020090658BD9085D085BD9085B0085B0085B0000000000000000000000000	3F 40 28 38 4E 02 4F 05 528 00 10 02C 09B F 3 2 00 9B F 3 2 00 8D 52 00 81 0 2 0 2 0 0 8 8 7 32 8 2 0 0 10 0 2 8 8 10 2 8 10 10 10 10 10 10 10 10 10 10 10 10 10	000 C0 C0 FD FC FD FD 02 FD 02	KEYIN KEYIN2 ESC RDCHAR NOTCR	LDA PHA AND ORA STA PLA JMP INC BNE INC BNE INC BNE INC BPL STA LDA BIT RTS JSR JSR JSR JSR JSR CMP BEQ RTS LDA PHA LDA STA LDA STA LDA STA LDA STA LDA STA STA STA DSR STA STA DSR STA STA STA STA STA STA STA STA STA STA	(BASL),Y #\$3F #\$40 (BASL),Y (KSWL) RNDL KEYIN2 RNDH KBD KEYIN3 (BASL),Y KBD KBDSTRB RDKEY ESC1 RDKEY ESC1 RDKEY #\$9B ESC INVFLG IN,X COUT INVFLG IN,X #\$88 BCKSPC	GO TO USER KEY-IN INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE CLR KEY STROBE GET KEYCODE HANDLE ESC FUNC. READ KEY ESC? YES, DON'T RETURN ECHO USER LINE NON INVERSE
FD10: FD11: FD13: FD15: FD17: FD18: FD18: FD18: FD24: FD24: FD28: FD28: FD28: FD28: FD28: FD28: FD28: FD28: FD37: FD37: FD37: FD37: FD37: FD37: FD37: FD37: FD37: FD44: FD44: FD44: FD44: FD44: FD44: FD44: FD50: FD52: FD52: FD52: FD52: FD52: FD52: FD52: FD52: FD52: FD52: FD52: FD52: FD52: FD52: FD52: FD52: FD52: FD52: FD52: FD53:	4299186C6606C09D20090658BC90658BC90658BC906558BC906558BC906558BC906558BC906558BC90588BC90588BC90588BC90588B090588B090588B090588B090588B090588B090588B090588B090588B090588B090588B090588B090588B090588B090588B090588B090588B090588B090588B0905888B0905888B0905888B0905888B0905888B0905888B0905888B0905888B0905888B09058888B0905888B0905888B0905888B09058888B09058888B09058888B09058888B09058888B09058888B09058888888888	3F 40 28 38 4E 02 4F 00 28 00 10 0C 20 20 9B F 3 2 00 ED 32 00 88 10 9	000 C0 C0 FD FC FD 02 FD 02	KEYIN2 KEYIN2 ESC RDCHAR NOTCR	LDA PHA AND ORA STA PLA STA JMP INC BNE INC BNE INC BIT STA LDA BIT RTS JSR JSR JSR LDA BIT RTS JSR LDA LDA STA LDA STA LDA STA LDA CMP BEQ CMP	<pre>(BASL),Y #\$3F #\$40 (BASL),Y (KSWL) RNDL KEYIN2 RNDH KBD KEYIN (BASL),Y KBD KBDSTRB RDKEY #\$9B ESC INVFLG IN,X #\$88 BCKSPC #\$98</pre>	GO TO USER KEY-IN INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE CLR KEY STROBE GET KEYCODE HANDLE ESC FUNC. READ KEY ESC? YES, DON'T RETURN ECHO USER LINE NON INVERSE CHECK FOR EDIT KEYS BS, CTRL-X
FD10: FD11: FD13: FD15: FD17: FD18: FD18: FD18: FD26: FD26: FD28: FD28: FD28: FD28: FD28: FD28: FD28: FD28: FD37: FD38: FD30: FD30: FD37: FD37: FD38: FD37: FD37: FD37: FD40: FD44: FD45: FD55:	48999186C66006200996850850996850850996850850996	3F 40 28 38 4E 02 4F 528 00 10 0C 2C 0C 9B 32 F 52 00 ED 32 00 88 1D 98 32	00 C0 C0 FD FC FD 02 FD 02	KEYIN KEYIN2 ESC RDCHAR NOTCR	LDA PHA AND ORA STA PLA JMP INC BNE INC BIT STA LDA BIT RTS JSR JSR JSR JSR JSR LDA BIT RTS JSR LDA DA DTA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA STA STA STA STA STA STA STA STA ST	(BASL),Y #\$3F #\$40 (BASL),Y (KSWL) RNDL KEYIN2 RNDH KBD KEYIN2 RDKH KBD KBDSTRB RDKEY ESC1 RDKEY #\$9B ESC INVFLG INVFLG IN,X COUT INVFLG IN,X #\$88 BCKSPC #\$98	GO TO USER KEY-IN INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE CLR KEY STROBE GET KEYCODE HANDLE ESC FUNC. READ KEY ESC? YES, DON'T RETURN ECHO USER LINE NON INVERSE CHECK FOR EDIT KEYS BS, CTRL-X
FD10: FD11: FD13: FD13: FD17: FD17: FD18: FD18: FD18: FD28: FD28: FD28: FD28: FD28: FD28: FD28: FD28: FD38: FD38: FD37: FD37: FD37: FD37: FD37: FD40: FD40: FD44: FD44: FD44: FD44: FD44: FD44: FD44: FD56: FD56:	48999186C66D66C2099085D966C66D66C2099085D966C66B06C2099085B09085B0909085B09090750	3F 40 28 38 4E 02 4F 05 528 00 10 02C 09B F 3 2 00 8 7 32 F 5 200 8 0 2 00 8 32 8 00 8 00 2 00 9 8 32 8 20 00 10 00 20 8 32 8 20 20 20 20 20 20 20 20 20 20 20 20 20	000 C0 FD FC FD FD 02 FD 02	KEYIN KEYIN2 ESC RDCHAR NOTCR	LDA PHA AND ORA STA PLA JMP INC BNE INC BNE INC BNE INC BNE INC BTA LDA STA LDA PHA LDA STA LDA PHA LDA STA LDA STA LDA STA LDA STA STA LDA STA STA STA PLA STA STA PLA STA STA PLA STA STA PLA STA STA PLA STA STA PLA STA STA STA STA STA STA STA STA STA ST	(BASL),Y #\$3F #\$40 (BASL),Y (KSWL) RNDL KEYIN2 RNDH KBD KEYIN3 (BASL),Y KBD KBDSTRB RDKEY ESC1 RDKEY ESC1 RDKEY #\$9B ESC INVFLG IN,X COUT INVFLG IN,X #\$88 BCKSPC #\$98 CANCEL	GO TO USER KEY-IN INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE CLR KEY STROBE GET KEYCODE HANDLE ESC FUNC. READ KEY ESC? YES, DON'T RETURN ECHO USER LINE NON INVERSE CHECK FOR EDIT KEYS BS, CTRL-X
FD10: FD11: FD13: FD15: FD17: FD18: FD18: FD18: FD24: FD24: FD28: FD38: FD38: FD37: FD40: FD44: FD44: FD48: FD59:	4299186C606C019A2C0090658BC685D909085CF090	3F 40 28 38 4E 02 4F 00 55 28 00 10 02C 02B F 3 2 00 8B 73 2 00 88 10 80 88 10 80 87 80 80 80 80 80 80 80 80 80 80 80 80 80	000 C0 C0 FD FC FD 02 FD 02	KEYIN2 KEYIN2 ESC RDCHAR NOTCR	LDA PHA AND ORA STA PLA JMP INC BNE INC BNE INC BIT STA LDA BIT RTS JSR JSR JSR JSR LDA BIT RTS JSR LDA DA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA	(BASL),Y #\$3F #\$40 (BASL),Y (KSWL) RNDL KEYIN2 RNDH KBD KEYIN2 (BASL),Y KBD KBDSTRB RDKEY #\$9B ESC INVFLG INVFLG IN,X #\$88 BCKSPC #\$98 CANCEL #\$F8	GO TO USER KEY-IN INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE CLR KEY STROBE GET KEYCODE HANDLE ESC FUNC. READ KEY ESC? YES, DON'T RETURN ECHO USER LINE NON INVERSE CHECK FOR EDIT KEYS BS, CTRL-X
FD10: FD11: FD13: FD17: FD17: FD18: FD18: FD18: FD24: FD24: FD24: FD28: FD28: FD28: FD28: FD28: FD28: FD30: FD30: FD30: FD30: FD30: FD31: FD30: FD31: FD31: FD32: FD32: FD32: FD34: FD40: FD44: FD45: FD55: FD55: FD55: FD55: FD55: FD55: FD55: FD55: FD55: FD55: FD55: FD55: FD55:	48999186C6600620090605880085090000000000000000000000000	3F 40 28 38 4E 02 4F 528 00 10 02C 02B 53 2 F 52 800 10 02C 02B 53 32 F 52 00 20 88 1D 80 88 1D 80 88 1D 80 80 80 80 80 80 80 80 80 80 80 80 80	00 C0 C0 FD FC FD 02 FD 02	KEYIN KEYIN2 ESC RDCHAR NOTCR	LDA PHA AND ORA STA PLA JMP INC BIT INC BIT STA LDA BIT RTS JSR JSR JSR JSR JSR JSR JSR LDA BIT RTS JSR JSR LDA BIT RTS JSR JSR LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA STA STA STA STA STA STA STA STA ST	(BASL),Y #\$3F #\$40 (BASL),Y (KSWL) RNDL KEYIN2 RNDH KBD KEYIN2 RDKH KBD KBDSTRB RDKEY ESC INVFLG INVFLG IN,X COUT INVFLG IN,X #\$88 BCKSPC #\$98 CANCEL #\$F8 NOTCR1	GO TO USER KEY-IN INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE CLR KEY STROBE GET KEYCODE HANDLE ESC FUNC. READ KEY ESC? YES, DON'T RETURN ECHO USER LINE NON INVERSE CHECK FOR EDIT KEYS BS, CTRL-X
FD10: FD11: FD13: FD15: FD17: FD18: FD18: FD18: FD18: FD24: FD24: FD28: FD28: FD28: FD28: FD28: FD28: FD28: FD37: FD37: FD37: FD37: FD37: FD37: FD37: FD40: FD40: FD44: FD44: FD44: FD44: FD44: FD44: FD44: FD55: FD55: FD55: FD55:	4899186C60020090850850900000000000000000000000000	3F 40 28 38 4E 02 4F 05 528 00 10 02C 09B F 3 2 F 5 200 8 0 20 09B F 3 2 F 5 200 20 8 5 3 2 F 5 200 20 8 5 3 2 8 20 8 5 20 20 8 5 5 20 9 8 5 5 20 20 20 20 20 20 20 20 20 20 20 20 20	00 C0 FD FC FD 02 FD 02 FF	KEYIN KEYIN2 ESC RDCHAR NOTCR	LDA PHA AND ORA STA PLA JMP INC BNE INC BNE INC BNE INC BPL STA LDA BIT RTS JSR JSR JSR JSR JSR JSR LDA PHA LDA STA LDA STA LDA STA LDA STA LDA STA STA STA STA STA STA STA STA STA ST	(BASL),Y #\$3F #\$40 (BASL),Y (KSWL) RNDL KEYIN2 RNDH KBD KEYIN3 (BASL),Y KBD KBDSTRB RDKEY ESC1 RDKEY ESC1 RDKEY #\$9B ESC INVFLG IN,X COUT INVFLG IN,X #\$88 BCKSPC #\$98 CANCEL #\$F8 NOTCR1 BELL	GO TO USER KEY-IN INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE CLR KEY STROBE GET KEYCODE HANDLE ESC FUNC. READ KEY ESC? YES, DON'T RETURN ECHO USER LINE NON INVERSE CHECK FOR EDIT KEYS BS, CTRL-X MARGIN? YES, SOUND BELL
FD10: FD11: FD13: FD15: FD17: FD18: FD18: FD18: FD24: FD24: FD28:	4299186C606C019A2C00906588BC85BC70900000000000000000000000000000000000	3F 40 28 38 4E 02 4F 00 55 28 00 10 02C 0C 9B F3 32 FF 32 00 88 1D 80 88 1D 80 3 3 A	00 C0 C0 FD FC FD 02 FD 02 FF	KEYIN2 KEYIN2 ESC RDCHAR NOTCR	LDA PHA AND ORA STA PLA STA JMP INC BNE INC BNE INC BIT STA LDA BIT RTS JSR JSR JSR JSR LDA BIT RTS JSR JSR LDA EQ CMP BEQ CMP BEQ CMP BEQ CMP BEQ CMP BEQ CMP STA LDA STA STA INC STA STA STA STA STA STA STA STA STA STA	(BASL),Y #\$3F #\$40 (BASL),Y (KSWL) RNDL KEYIN2 RNDH KBD KEYIN2 RDA KEYIN (BASL),Y KBD KBDSTRB RDKEY #\$9B ESC INVFLG INVFLG IN,X #\$88 BCKSPC INVFLG IN,X #\$88 BCKSPC #\$78 NOTCR1 BELL	GO TO USER KEY-IN INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE CLR KEY STROBE GET KEYCODE HANDLE ESC FUNC. READ KEY ESC? YES, DON'T RETURN ECHO USER LINE NON INVERSE CHECK FOR EDIT KEYS BS, CTRL-X MARGIN? YES, SOUND BELL
FD10: FD11: FD13: FD15: FD17: FD18: FD18: FD18: FD24: FD24: FD24: FD28: FD28: FD28: FD28: FD28: FD34: FD37: FD57:	4299186C66D62C019A2C02009C0588D2685D90086CFC90088	3F 40 28 38 4E 02 4F 528 00 10 0C 2C 0C 9B 32 F 52 00 2C 0C 9B 32 F 32 7F 200 ED 32 00 88 1D 98 03 3A	00 C0 FD FC FD 02 FD 02 FF	KEYIN2 KEYIN2 ESC RDCHAR NOTCR	LDA PHA AND ORA STA PLA JMP INC BNE INC BIT STA LDA BIT RTS JSR JSR JSR JSR JSR JSR JSR LDA BIT RTS JSR JSR LDA BIT RTS JSR JSR LDA STA LDA STA LDA STA LDA STA LDA STA STA STA STA STA STA STA STA STA ST	(BASL),Y #\$3F #\$40 (BASL),Y (KSWL) RNDL KEYIN2 RNDH KBD KEYIN2 RDA KBDSTRB RDKEY #\$9B ESC INVFLG INVFLG INVFLG IN,X #\$88 BCKSPC #\$98 CANCEL #\$F8 NOTCR1 BELL	GO TO USER KEY-IN INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE CLR KEY STROBE GET KEYCODE HANDLE ESC FUNC. READ KEY ESC? YES, DON'T RETURN ECHO USER LINE NON INVERSE CHECK FOR EDIT KEYS BS, CTRL-X MARGIN? YES, SOUND BELL ADVANCE INPUT INDEX
FD10: FD11: FD13: FD15: FD17: FD17: FD18: FD18: FD18: FD28: FD28: FD28: FD28: FD28: FD28: FD28: FD28: FD37: FD37: FD37: FD37: FD37: FD37: FD37: FD37: FD40: FD40: FD44: FD44: FD44: FD44: FD44: FD44: FD56: FD58: FD58: FD57: FD57: FD57: FD56: FD57:	48999186C660020090658809085090000000000000000000000000	3F 40 28 38 4E 02 4F 05 528 00 10 02C 09B F 3 2 00 9B F 3 2 00 2 0 2 0 2 0 2 8 5 3 2 5 5 2 8 0 0 10 0 2 8 5 5 2 8 0 0 10 0 2 8 10 10 2 8 10 10 10 10 10 10 10 10 10 10 10 10 10	00 C0 FD FC FD FD 02 FD 02 FF	KEYIN KEYIN2 ESC RDCHAR NOTCR	LDA PHA AND ORA STA PLA JMP INC BNE INC BNE INC BNE INC BNE INC BTA LDA STA LDA BIT RTS LDA PHA LDA STA LDA PHA LDA STA LDA STA LDA STA LDA STA INC BEQ CMP BEQ CMP BEQ CPX BCC JSR INX BNE	(BASL),Y #\$3F #\$40 (BASL),Y (KSWL) RNDL KEYIN2 RNDH KEYIN2 RNDH KBD KEYIN3 (BASL),Y KBD KBDSTRB RDKEY ESC1 RDKEY ESC1 RDKEY ESC1 RDKEY ESC1 RDKEY ESC1 RDKEY ESC1 RDKEY ESC1 RDKEY ESC1 RDKEY ESC1 RDKEY ESC1 RDKEY ESC1 RDKEY ESC1 RDKEY ESC1 RDKEY ESC2 INVFLG IN,X #\$88 BCKSPC #\$98 CANCEL #\$F8 NOTCR1 BELL NXTCHAR	GO TO USER KEY-IN INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE CLR KEY STROBE GET KEYCODE HANDLE ESC FUNC. READ KEY ESC? YES, DON'T RETURN ECHO USER LINE NON INVERSE CHECK FOR EDIT KEYS BS, CTRL-X MARGIN? YES, SOUND BELL ADVANCE INPUT INDEX
FD10: FD13: FD13: FD15: FD17: FD18: FD18: FD18: FD24: FD24: FD28:	4899186C6606C019A2C020900588BC685D9028009 86C6006C019A2C0200900588BC685D9028009 86C600000000000000000000000000000000000	3F 40 28 38 4E 02 4F 00 528 00 10 02C 0C 9B F 3 2 00 8 7 32 7 7 32 8 00 20 8 8 00 20 8 8 7 32 7 8 00 20 8 8 10 20 8 8 10 20 20 8 10 20 20 20 20 20 20 20 20 20 20 20 20 20	00 C0 C0 FD FC FD 02 FD 02 FF	KEYIN KEYIN2 ESC RDCHAR NOTCR	LDA PHA AND ORA STA PLA STA JMP INC BNE INC BNE INC BIT STA LDA BIT RTS JSR JSR JSR JSR JSR LDA EQ RTS LDA PHA LDA STA STA LDA STA STA STA STA STA STA STA STA STA ST	(BASL),Y #\$3F #\$40 (BASL),Y (KSWL) RNDL KEYIN2 RNDH KEYIN2 RNDH KEYIN3 (BASL),Y KBD KEYIN4 (BASL),Y KBD KEYIN4 (BASL),Y KBD KEY10 (BASL),Y KBD KEY10 (BASL),Y (BASL),Y (SWL)6 (BASL),Y (SWL)6	GO TO USER KEY-IN INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE HANDLE ESC FUNC. READ KEY ESC? YES, DON'T RETURN ECHO USER LINE NON INVERSE CHECK FOR EDIT KEYS BS, CTRL-X MARGIN? YES, SOUND BELL ADVANCE INPUT INDEX BACKSLASH AFTER CANCELLED LINE

FD67:	20	8E	FD	GETLNZ	JSR	CROUT	OUTPUT CR
FD6C.	A5 20	33 ED	БD	GETLN	LDA	COUT	ΟΠΑΔΙΆ ΡΕΟΜΡΆ CHAR
FD6F:	A2	01	ГD		LDX	#\$01	INIT INPUT INDEX
FD71:	8A			BCKSPC	TXA		WILL BACKSPACE TO 0
FD72:	F0	F3			BEQ	GETLNZ	
FD74:	CA	25			DEX		
FD/5:	20	35	FD	NATCHAR	JSR	HDTCK	ILCE CODEEN CUAD
FD78:	00	02			BNE	CAPTST	FOR CTRL-U
FD7C:	Bl	28			LDA	(BASL),Y	
FD7E:	C9	ΕO		CAPTST	CMP	#\$E0	
FD80:	90	02			BCC	ADDINP	CONVERT TO CAPS
FD82:	29	DF			AND	#\$DF	
FD84:	9D	00	02	ADDINP	STA	IN,X	ADD TO INPUT BUF
FD8/:	0	8D 82			ENF	# \$ 8 D NOTCP	
FD8B:	20	9C	FC		JSR	CLREOL	CLR TO EOL IF CR
FD8E:	A9	8D		CROUT	LDA	#\$8D	
FD90:	D0	5B			BNE	COUT	
FD92:	A4	3D		PRAl	LDY	AlH	PRINT CR,Al IN HEX
FD94:	A6	3C	-	DD11110	LDX	ALL	
FD90:	20	8E 40	FD FQ	PRIXZ	JSK	DDNTVV	
FD9C:	20 A0	00	ĽĴ		LDY	#\$00	
FD9E:	A9	AD			LDA	#\$AD	PRINT '-'
FDA0:	4C	ED	FD		JMP	COUT	
FDA3:	A5	3C		XAM8	LDA	All	
FDA5:	09	07			ORA	#\$07	SET TO FINISH AT
FDA/:	85	3E 2D			STA	AZL	MOD 8=7
FDA9:	A5 85	3 F			STA	АІН	
FDAD:	A5	3C		MODSCHK	LDA	AlL	
FDAF:	29	07			AND	#\$07	
FDBl:	D0	03			BNE	DATAOUT	
FDB3:	20	92	FD	XAM	JSR	PRAL	
FDB6:	A9 20	AU FD	חיד	DATAOUT	LDA	#ŞAU COUT	OUTDUT DI ANK
FDBB:	20 Bl	3C	ГD		T.DA	(Alt.).Y	OULD PRAK
FDBD:	20	DA	FD		JSR	PRBYTE	OUTPUT BYTE IN HEX
FDC0:	20	BA	FC		JSR	NXTAl	
FDC3:	90	E8			BCC	MODSCHK	CHECK IF TIME TO,
FDC5:	60			RTS4C	RTS		PRINT ADDR
FDC6:	4A 90	FΔ		ХАМРМ	LSR	А	MODE IS XAM
FDC9:	4A	111			LSR	A	ADD, OR SUB
FDCA:	4A				LSR	A	
FDCB:	A5	3E			LDA	A2L	
FDCD:	90	02			BCC	ADD	
FDCF:	49	F.F.			EOR	#ŞE'E' אוא	SUB: FORM 2'S COMPLEMENT
FDD1:	48	50		ADD	PHA	ATD	
FDD4:	A9	BD			LDA	#\$BD	
FDD6:	20	ED	FD		JSR	COUT	PRINT '=', THEN RESULT
FDD9:	68				PLA		
FDDA:	48			PRBYTE	PHA	7	PRINT BYTE AS 2 HEX
FDDC:	4A 4A				LSR	A	DIGIIS, DESIROIS A-REG
FDDD:	4A				LSR	A	
FDDE:	4A				LSR	A	
FDDF:	20	E5	FD		JSR	PRHEXZ	
FDE2:	68	~ -			PLA	" +	
FDE3:	29	OF		PRHEX	AND	#\$0F #\$D0	PRINT HEX DIG IN A-REG
FDE5:	09 C9	BU BA		PRHEXZ	CMP	#\$B0 #\$BA	T2B.2
FDE7:	90	02			BCC	COUT	
FDEB:	69	06			ADC	#\$06	
FDED:	6C	36	00	COUT	JMP	(CSWL)	VECTOR TO USER OUTPUT ROUTINE
FDF0:	C9	A0		COUT1	CMP	#\$A0	
FDF2:	90	02			BCC	COUTZ	DON'T OUTPUT CTRL'S INVERSE
FDF4:	25	32		COUT	AND	INVFLG	MASK WITH INVERSE FLAG
FDF8:	48	55		0012	PHA	ISAVI	SAV A-REG
FDF9:	20	FD	FB		JSR	VIDOUT	OUTPUT A-REG AS ASCII
FDFC:	68				PLA		RESTORE A-REG
FDFD:	A4	35			LDY	YSAVl	AND Y-REG
FDFF:	60	2.4		ן דם	RTS	V C A 57	THEN RETURN
FE00:	С6 Е0	う4 0 F		вгт	DEC	I SAV XAM8	
FE04:	CA	75		BLANK	DEX	AAPIO	BLANK TO MON
FE05:	D0	16			BNE	SETMDZ	AFTER BLANK
FE07:	C9	BA			CMP	#\$BA	DATA STORE MODE?
FE09:	D0	BB			BNE	XAMPM	NO, XAM, ADD, OR SUB
FEOD.	85 25	১⊥ ২৮		STOR	P.L.A	MODE A 2 T.	KEEP IN STORE MODE
	110					لسل سته د د	

FEOF:	91	40			STA	(A3L),Y	STORE AS LOW BYTE AS (A3)
LEI3.	E6	40			INC	AJL DTC5	
FE15.	E6	41			TNC	73H	INCK AS, REIORN
FE17:	60	41		RTS5	RTS	AJII	
FE18:	A4	34		SETMODE	TDY	YSAV	SAVE CONVERTED ':'. '+'.
FELA:	В9	FF	01	021110022	LDA	IN-1,Y	'-', '.' AS MODE.
FE1D:	85	31		SETMDZ	STA	MODE	,
FElF:	60				RTS		
FE20:	A2	01		LT	LDX	#\$01	
FE22:	В5	3E		LT2	LDA	A2L,X	COPY A2 (2 BYTES) TO
FE24:	95	42			STA	A4L,X	A4 AND A5
FE26:	95	44			STA	A5L,X	
FE28:	CA				DEX		
FE29:	10	F7			BPL	LT2	
FE2B:	60				RTS		
FE2C:	BL	3C		MOVE	LDA	(ALL),Y	MOVE (AL TO A2) TO
FEZE:	91	4Z	ПQ		STA	(A4L),Y	(A4)
FE30:	20	B4 127	FC		JSK	NATA4 MOVE	
FE35.	60	r /			DUC	MOVE	
FE36:	B1	30		VFY	T.DA	(Alt.).Y	VERTEY (AL TO A2) WITH
FE38:	D1	42			CMP	$(A4T_{i})$ , Y	(A4)
FE3A:	FO	1C			BEO	VFYOK	()
FE3C:	20	92	FD		JSR	PRAL	
FE3F:	Bl	3C			LDA	(AlL),Y	
FE41:	20	DA	FD		JSR	PRBYTE	
FE44:	A9	A0			LDA	#\$A0	
FE46:	20	ED	FD		JSR	COUT	
FE49:	A9	A8			LDA	#\$A8	
FE4B:	20	ED	FD		JSR	COUT	
FE4E:	Bl	42			LDA	(A4L),Y	
FE50:	20	DA	FD		JSR	PRBYTE	
FE53:	A9	A9			LDA	#\$A9	
FE55:	20	ED D4	FD	VENOR	JSR	COUT	
FE58:	20	B4	FC	VFYOR	JSR	NXTA4	
FESD:	90 60	D9			DUC	VEI	
FE5E.	20	75	ਸ਼ਾਜ	T.TST	TSB	Alpc	MOVE AL (2 BYTES) TO
FE61:	20 29	14	ГĽ	птот	T.DA	#\$14	PC IF SPEC'D AND
FE63:	48			LTST2	PHA	1411	DISEMBLE 20 INSTRS
FE64:	20	D0	F8	21012	JSR	INSTDSP	
FE67:	20	53	F9		JSR	PCADJ	ADJUST PC EACH INSTR
FE6A:	85	3A			STA	PCL	
FE6C:	84	3B			STY	PCH	
FE6E:	68				PLA		
FE6F:	38				SEC		
FE70:	E9	01			SBC	#\$01 	NEXT OF 20 INSTRS
FE/2:	DO	EF			BNE	LIST2	
FE/4:	00				RTS mva		TE LICED CDECLD ADD
FE/3: FE76.	8A 50	07		AIPC	DEO	λιραρπα	CODY FROM AL TO DC
FE78.	P U B 5	30		AIPCLP		ALFCRIS	COFI FROM AL TO FC
FE7A:	95	50		TITI CITI		1111/1	
FE7C:	22	3 A			STA	PCT. X	
	CA	3A			STA DEX	PCL,X	
FE/D:	CA 10	3A F9			STA DEX BPL	PCL,X AlPCLP	
FE7D: FE7F:	CA 10 60	3A F9		Alpcrts	STA DEX BPL RTS	PCL,X AlPCLP	
FE7D: FE7F: FE80:	CA 10 60 A0	3A F9 3F		AlPCRTS SETINV	STA DEX BPL RTS LDY	PCL,X AlPCLP #\$3F	SET FOR INVERSE VID
FE7D: FE7F: FE80: FE82:	CA 10 60 A0 D0	3A F9 3F 02		Alpcrts SETINV	STA DEX BPL RTS LDY BNE	PCL,X AlPCLP #\$3F SETIFLG	SET FOR INVERSE VID VIA COUT1
FE7D: FE7F: FE80: FE82: FE84:	CA 10 60 A0 D0 A0	3A F9 3F 02 FF		AlpCRTS SETINV SETNORM	STA DEX BPL RTS LDY BNE LDY	PCL,X AlPCLP #\$3F SETIFLG #\$FF	SET FOR INVERSE VID VIA COUT1 SET FOR NORMAL VID
FE7D: FE7F: FE80: FE82: FE84: FE86:	CA 10 60 A0 D0 A0 84	3A F9 3F 02 FF 32		Alpcrts SETINV SETNORM SETIFLG	STA DEX BPL RTS LDY BNE LDY STY	PCL,X AlPCLP #\$3F SETIFLG #\$FF INVFLG	SET FOR INVERSE VID VIA COUT1 SET FOR NORMAL VID
FE7D: FE7F: FE80: FE82: FE84: FE86: FE88:	CA 10 60 A0 D0 A0 84 60	3A F9 3F 02 FF 32		Alpcrts SETINV SETNORM SETIFLG	STA DEX BPL RTS LDY BNE LDY STY RTS	PCL,X AlPCLP #\$3F SETIFLG #\$FF INVFLG	SET FOR INVERSE VID VIA COUT1 SET FOR NORMAL VID
FE7D: FE7F: FE80: FE82: FE84: FE86: FE88: FE88:	CA 10 60 A0 D0 A0 84 60 A9	3A F9 3F 02 FF 32 00		Alpcrts SETINV SETNORM SETIFLG	STA DEX BPL RTS LDY BNE LDY STY RTS LDA	PCL,X AlPCLP #\$3F SETIFLG #\$FF INVFLG #\$00	SET FOR INVERSE VID VIA COUT1 SET FOR NORMAL VID SIMULATE PORT #0 INPUT
FE7D: FE7F: FE80: FE82: FE84: FE86: FE88: FE88: FE88: FE88:	CA 10 60 A0 D0 A0 84 60 A9 85	3A F9 3F 02 FF 32 00 3E		Alpcrts SETINV SETNORM SETIFLG SETKBD INPORT	STA DEX BPL RTS LDY BNE LDY STY RTS LDA STA	PCL,X AlPCLP #\$3F SETIFLG #\$FF INVFLG #\$00 A2L	SET FOR INVERSE VID VIA COUT1 SET FOR NORMAL VID SIMULATE PORT #0 INPUT SPECIFIED (KEYIN ROUTINE)
FE7D: FE7F: FE80: FE82: FE84: FE86: FE88: FE88: FE89: FE8B: FE8D:	CA 10 60 A0 D0 A0 84 60 A9 85 A2	3A F9 3F 02 FF 32 00 3E 38		Alpcrts SETINV SETNORM SETIFLG SETKBD INPORT INPRT	STA DEX BPL RTS LDY BNE LDY STY RTS LDA STA LDX	PCL,X AlPCLP #\$3F SETIFLG #\$FF INVFLG #\$00 A2L #KSWL	SET FOR INVERSE VID VIA COUT1 SET FOR NORMAL VID SIMULATE PORT #0 INPUT SPECIFIED (KEYIN ROUTINE)
FE7D: FE7F: FE80: FE82: FE84: FE86: FE88: FE89: FE8B: FE8D: FE8D: FE80:	CA 10 60 A0 D0 A0 84 60 A9 85 A2 A0	3A F9 3F 02 FF 32 00 3E 38 1B		Alpcrts SETINV SETNORM SETIFLG SETKBD INPORT INPRT	STA DEX BPL RTS LDY BNE LDY STY RTS LDA STA LDX LDY DYE	PCL,X AlPCLP #\$3F SETIFLG #\$FF INVFLG #\$00 A2L #KSWL #KSWL #KEYIN	SET FOR INVERSE VID VIA COUT1 SET FOR NORMAL VID SIMULATE PORT #0 INPUT SPECIFIED (KEYIN ROUTINE)
FE7D: FE7F: FE80: FE82: FE82: FE84: FE86: FE88: FE89: FE89: FE8D: FE8F: FE91: FE93:	CA 10 60 A0 D0 A0 84 60 A9 85 A2 A0 D0	3A F9 3F 02 FF 32 00 3E 38 1B 08 00		Alpcrts SETINV SETNORM SETIFLG SETKBD INPORT INPRT	STA DEX BPL RTS LDY STY RTS LDY STY LDA STA LDX LDX LDX LDA	PCL,X AlPCLP #\$3F SETIFLG #\$FF INVFLG #\$00 A2L #KSWL #KEYIN IOPRT #\$00	SET FOR INVERSE VID VIA COUTI SET FOR NORMAL VID SIMULATE PORT #0 INPUT SPECIFIED (KEYIN ROUTINE)
FE7D: FE7F: FE80: FE82: FE84: FE86: FE88: FE89: FE8B: FE8D: FE8F: FE91: FE95:	CA 10 60 A0 D0 A0 84 60 A9 85 A2 A0 D0 A9 85	3A F9 3F 02 FF 32 00 3E 38 1B 08 00 3F		Alpcrts SETINV SETNORM SETIFLG SETKBD INPORT INPRT	STA DEX BPL RTS LDY BNE LDY STY RTS LDY STA LDX LDY BNE LDA STA	PCL,X AlPCLP #\$3F SETIFLG #\$FF INVFLG #\$00 A2L #KSWL #KEYIN IOPRT #\$00 A2L	SET FOR INVERSE VID VIA COUT1 SET FOR NORMAL VID SIMULATE PORT #0 INPUT SPECIFIED (KEYIN ROUTINE) SIMULATE PORT #0 OUTPUT
FE7D: FE7F: FE80: FE82: FE84: FE88: FE89: FE89: FE85: FE91: FE95: FE97:	CA 10 60 A0 D0 A0 84 60 A9 85 A2 A0 D0 A9 85 A2	3A F9 3F 02 FF 32 00 3E 38 1B 08 00 3E 36		Alpcrts SETINV SETNORM SETIFLG SETKBD INPORT INPRT SETVID OUTPORT OUTPRT	STA DEX BPL RTS LDY BNE LDY STY RTS LDA STA LDX LDY STA LDX	PCL,X AlPCLP #\$3F SETIFLG #\$FF INVFLG #\$00 A2L #KSWL #KEYIN IOPRT #\$00 A2L #CSWL	SET FOR INVERSE VID VIA COUT1 SET FOR NORMAL VID SIMULATE PORT #0 INPUT SPECIFIED (KEYIN ROUTINE) SIMULATE PORT #0 OUTPUT SPECIFIED (COUT1 ROUTINE)
FE7D: FE7T: FE80: FE82: FE84: FE88: FE88: FE89: FE8D: FE8D: FE91: FE93: FE95: FE97:	CA 10 60 A0 D0 84 60 A9 85 A2 A0 D0 A9 85 A2 A0	3A F9 3F 02 FFF 32 00 3E 38 1B 08 00 3E 36 F0		Alpcrts Setinv Setiflg SetkBD INPORT INPRT SETVID OUTPORT OUTPRT	STA DEX BPL RTS LDY BNE LDY STY RTS LDA LDX LDY BNE LDA STA LDX LDY	PCL,X AlPCLP #\$3F SETIFLG #\$FF INVFLG #\$00 A2L #KSWL #KEYIN IOPRT #\$00 A2L #CSWL #COUT1	SET FOR INVERSE VID VIA COUT1 SET FOR NORMAL VID SIMULATE PORT #0 INPUT SPECIFIED (KEYIN ROUTINE) SIMULATE PORT #0 OUTPUT SPECIFIED (COUT1 ROUTINE)
FE7D: FE7T: FE80: FE82: FE84: FE86: FE88: FE89: FE89: FE8D: FE91: FE91: FE95: FE97: FE99: FE99: FE99:	CA 10 60 A0 D0 A0 84 60 A9 85 A2 A0 D0 A9 85 A2 A0 A5	3A F9 3F 02 FFF 32 00 3E 38 1B 08 00 3E 36 F0 3E		Alpcrts Setinv Setnorm Setiflg Setkbd INPORT INPRT SETVID OUTPORT OUTPRT	STA DEX BPL RTS LDY BNE LDY STY RTS LDA STA LDX LDY LDA LDX LDY LDA	PCL,X AlPCLP #\$3F SETIFLG #\$FF INVFLG #\$00 A2L #KSWL #KEYIN IOPRT #\$00 A2L #CSWL #COUTL A2L	SET FOR INVERSE VID VIA COUT1 SET FOR NORMAL VID SIMULATE PORT #0 INPUT SPECIFIED (KEYIN ROUTINE) SIMULATE PORT #0 OUTPUT SPECIFIED (COUT1 ROUTINE) SET RAM IN/OUT VECTORS
FE7D: FE7T: FE82: FE82: FE84: FE86: FE88: FE89: FE89: FE85: FE91: FE91: FE95: FE95: FE99: FE99: FE99: FE90:	CA 10 60 A0 D0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0	3A F9 3F 02 FF 32 00 3E 38 1B 08 00 3E 36 F0 3E 0F		Alpcrts SETINV SETIFLG SETKBD INPORT INPRT SETVID OUTPORT OUTPRT	STA DEX BPL RTS LDY BNE LDY STY RTS LDY STA LDY LDY LDA STA LDY LDA AND	PCL,X AlPCLP #\$3F SETIFLG #\$FF INVFLG #\$00 A2L #KSWL #KEYIN IOPRT #\$00 A2L #CSWL #CSWL #CSWL #CSWL #\$0F	SET FOR INVERSE VID VIA COUT1 SET FOR NORMAL VID SIMULATE PORT #0 INPUT SPECIFIED (KEYIN ROUTINE) SIMULATE PORT #0 OUTPUT SPECIFIED (COUT1 ROUTINE) SET RAM IN/OUT VECTORS
FE7D: FE7F: FE80: FE82: FE84: FE84: FE86: FE89: FE89: FE89: FE91: FE91: FE95: FE99: FE99: FE99: FE95: FE95: FE95:	CA 10 60 A0 D0 A0 84 60 A9 85 A2 A0 D0 A9 85 A2 A0 A5 29 F0	3A F9 3F 02 FF 32 00 3E 38 1B 08 00 3E 36 F0 3E 0F 06		Alpcrts SETINV SETNORM SETIFLG SETKBD INPORT INPRT SETVID OUTPORT OUTPRT	STA DEX BPL RTS LDY BNE LDY STY RTS LDA STA LDA STA LDA STA LDA STA LDA AND BEQ	PCL,X AlPCLP #\$3F SETIFLG #\$FF INVFLG #\$00 A2L #KSWL #KSWL #KSWL #CSWL #COUT1 A2L #\$0F IOPRT	SET FOR INVERSE VID VIA COUT1 SET FOR NORMAL VID SIMULATE PORT #0 INPUT SPECIFIED (KEYIN ROUTINE) SIMULATE PORT #0 OUTPUT SPECIFIED (COUT1 ROUTINE) SET RAM IN/OUT VECTORS
FE7D: FE7F: FE80: FE82: FE84: FE84: FE86: FE89: FE8D: FE8D: FE91: FE91: FE95: FE97: FE99: FE99: FE95:	CA 10 60 A0 D0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0	3A F9 3F 02 FF 32 00 3E 38 1B 08 00 3E 36 F0 3E 06 C0		Alpcrts SETINV SETNORM SETIFLG SETKBD INPORT INPRT SETVID OUTPORT OUTPRT	STA DEX BPL RTS LDY BNE LDY STY LDA STA LDA STA LDA STA LDA STA LDA STA LDA BNE LDA STA LDA STA LDA STA LDA STA	PCL,X AlPCLP #\$3F SETIFLG #\$FF INVFLG #\$00 A2L #KSWL #KSWL #KSWL #KSWL #CSWL #COWT1 A2L #\$0F IOPRT1 #IOADR/256	SET FOR INVERSE VID VIA COUT1 SET FOR NORMAL VID SIMULATE PORT #0 INPUT SPECIFIED (KEYIN ROUTINE) SIMULATE PORT #0 OUTPUT SPECIFIED (COUT1 ROUTINE) SET RAM IN/OUT VECTORS
FE7D: FE7D: FE80: FE80: FE84: FE86: FE80: FE80: FE80: FE80: FE91: FE91: FE91: FE95: FE99: FE95: FE95: FE95: FE95: FE85: FE95: FE95: FE85:	CA 10 60 00 00 84 60 85 A2 00 85 A2 00 85 A2 00 85 A2 00 9 60 85 A2 00 09 A0	3A F9 3F 02 FF 32 00 3E 38 1B 08 00 3E 36 F0 3E 06 C0 00		Alpcrts SETINV SETNORM SETIFLG SETKBD INPORT INPRT SETVID OUTPORT OUTPRT	STA DEX BPL RTS LDY BNE LDY STY LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA LDY BNE LDA STA LDY BLA LDY LDA LDY LDY LDY LDY LDY STY LDY STY LDY STY LDY STY LDY STY LDY STY LDY STY LDY STY LDY STY LDY STY LDY STY LDY STY LDY STY LDY STA LDY STA LDY STA LDY STA LDY STA LDY STA LDY STA LDY STA LDY STA LDY STA LDY STA LDY STA LDY STA LDY STA LDY STA LDY STA LDY STA LDA STA LDY STA LDA STA STA STA STA STA STA STA STA STA ST	PCL,X AlPCLP #\$3F SETIFLG #\$FF INVFLG #\$00 A2L #KSWL #KEYIN IOPRT #\$00 A2L #CSWL #COUT1 A2L #\$0F IOPRT1 #IOADR/256 #\$00	SET FOR INVERSE VID VIA COUT1 SET FOR NORMAL VID SIMULATE PORT #0 INPUT SPECIFIED (KEYIN ROUTINE) SIMULATE PORT #0 OUTPUT SPECIFIED (COUT1 ROUTINE) SET RAM IN/OUT VECTORS
FE7D: FE7D: FE80: FE80: FE84: FE84: FE88: FE88: FE88: FE80: FE91: FE91: FE95:	CA 10 60 A0 D0 A0 84 60 A9 85 A2 A0 D0 A9 85 A2 A0 A5 29 F0 09 A0 F0	3A F9 3F 62 FF 32 00 3E 38 1B 08 00 3E 36 00 3E 50 00 00 00 00 00		Alpcrts SETINV SETNORM SETIFLG SETKBD INPORT INPRT SETVID OUTPORT OUTPORT	STA DEX BPL RTS LDY STS LDY STY RTS LDA STA LDX LDX LDX LDX LDX LDX LDX LDX LDY STA LDX LDY STA	PCL,X AlPCLP #\$3F SETIFLG #\$FF INVFLG #\$00 A2L #KSWL #KEYIN IOPRT #\$00 A2L #COUTI A2L #COUTI A2L #SOF IOPRT1 #IOADR/256 #\$00 IOPRT2	SET FOR INVERSE VID VIA COUT1 SET FOR NORMAL VID SIMULATE PORT #0 INPUT SPECIFIED (KEYIN ROUTINE) SIMULATE PORT #0 OUTPUT SPECIFIED (COUT1 ROUTINE) SET RAM IN/OUT VECTORS
FE7D: FE7D: FE80: FE80: FE84: FE84: FE88: FE89: FE80: FE80: FE91: FE91: FE92: FE92: FE92: FE92: FE92: FE92: FE93: FE92:	CA 10 60 A0 D0 A0 84 60 85 A2 A0 A9 85 A2 A0 A9 85 A2 A0 A5 9 F0 9 A0 F0 A0	3A F9 3F 62 FF 32 00 3E 38 1B 08 03 26 50 36 00 32 60 00 02 FD 60 00 02 FD		Alpcrts SETINV SETNORM SETIFLG SETKBD INPORT INPRT SETVID OUTPORT IOPRT IOPRT	STA DEX BPL RTS LDY STS LDY STY RTS LDY LDA STA LDX LDY LDA STA LDY LDX LDY LDA STA LDY LDY LDA STA LDY STA LDY STA	PCL,X AlPCLP #\$3F SETIFLG #\$FF INVFLG #\$00 A2L #KSWL #KEYIN IOPRT #\$00 A2L #CSWL #COUT1 A2L #\$0F IOPRT1 #IOADR/256 #\$00 IOPRT2 #COUT1/256	SET FOR INVERSE VID VIA COUT1 SET FOR NORMAL VID SIMULATE PORT #0 INPUT SPECIFIED (KEYIN ROUTINE) SIMULATE PORT #0 OUTPUT SPECIFIED (COUT1 ROUTINE) SET RAM IN/OUT VECTORS
FE7D: FE7D: FE80: FE80: FE84: FE86: FE88: FE89: FE89: FE80: FE80: FE91: FE93: FE91: FE92: FE92: FE92: FE92: FE92: FE92: FE93: FE92: FE92: FE92: FE92: FE92: FE92: FE92: FE92: FE92: FE93: FE92: FE93: FE92: FE93:	CA 10 60 00 00 80 00 85 20 09 85 20 09 85 20 09 09 60 99 20 09 20 20 20 20 20 20 20 20 20 20 20 20 20	3A F9 3F 02 FF 32 00 3E 38 00 3E 36 07 06 00 02 FD 00 02 FD		Alpcrts Setinv Setiflg SetkBD INPORT INPRT SETVID OUTPORT IOPRT IOPRT1 IOPRT1	STA DEX BPL RTS LDY BNE LDY STY RTS LDA STA LDX LDY LDA STA LDY LDA STA LDY LDA STA LDY LDA STA LDY STA STA STA STA STA STA STA STA STA STA	PCL,X AlPCLP #\$3F SETIFLG #\$FF INVFLG #\$00 A2L #KSWL #KEYIN IOPRT #\$00 A2L #COUTI A2L #COUTI A2L #\$0F IOPRT1 #10ADR/256 #\$00 IOPRT2 #COUTI/256 LOC0,X	SET FOR INVERSE VID VIA COUT1 SET FOR NORMAL VID SIMULATE PORT #0 INPUT SPECIFIED (KEYIN ROUTINE) SIMULATE PORT #0 OUTPUT SPECIFIED (COUT1 ROUTINE) SET RAM IN/OUT VECTORS
FE7D: FE7D: FE8D: FE82: FE84: FE84: FE86: FE88: FE89: FE8D: FE8D: FE91: FE93: FE95: FE95: FE95: FE95: FE95: FE4D: FEA3: FE35:	CA 10 60 00 00 80 00 85 20 85 20 85 85 85 85 85 85 85 85 85 85 85 85 85	3A F9 3F 02 FF 32 00 3E 38 00 3E 07 06 00 02 FD 00 01		Alpcrts Setinv Setiflg SetkBD INPORT INPRT SETVID OUTPORT IOPRT IOPRT1 IOPRT1	STA DEX BPL RTS LDY BNE LDY STY RTS LDX LDY LDX LDY LDX LDY LDA AND BEQ CORA LDY STA LDY STA STY STA	PCL,X AlPCLP #\$3F SETIFLG #\$FF INVFLG #\$00 A2L #KSWL #KEYIN IOPRT #\$00 A2L #COUT1 A2L #COUT1 A2L #\$0F IOPRT1 #IOADR/256 #\$00 IOPRT2 #COUT1/256 LOC0,X LOC1,X	SET FOR INVERSE VID VIA COUT1 SET FOR NORMAL VID SIMULATE PORT #0 INPUT SPECIFIED (KEYIN ROUTINE) SIMULATE PORT #0 OUTPUT SPECIFIED (COUT1 ROUTINE) SET RAM IN/OUT VECTORS
FE7D: FE7D: FE80: FE80: FE84: FE84: FE84: FE88: FE80: FE80: FE80: FE91: FE91: FE91: FE91: FE91: FE92: FE93: FE94:	CA 10 600 A00 A00	3A F9 3F 02 FF 32 00 3E 38 08 00 3E 60 00 3E 60 00 00 00 00 00 00 00 00		Alpcrts Setinv Setiflg SetkBD INPORT INPRT SETVID OUTPORT OUTPRT IOPRT IOPRT1 IOPRT2	STA DEX BPL RTS LDY STY RTS LDY STA LDY LDX LDY LDA STA LDY LDA AND BEQ CORA LDY STA STA STA STA	PCL,X AlPCLP #\$3F SETIFLG #\$FF INVFLG #\$00 A2L #KSWL #KEYIN IOPRT #\$00 A2L #COUT1 A2L #COUT1 A2L #\$0F IOPRT1 #\$00 IOPRT2 #COUT1/256 LOC0,X LOC1,X	SET FOR INVERSE VID VIA COUT1 SET FOR NORMAL VID SIMULATE PORT #0 INPUT SPECIFIED (KEYIN ROUTINE) SIMULATE PORT #0 OUTPUT SPECIFIED (COUT1 ROUTINE) SET RAM IN/OUT VECTORS
FE7D: FE7D: FE80: FE80: FE84: FE84: FE84: FE80: FE80: FE80: FE80: FE87: FE91:	CA 10 600 D00 85 A00 A00 A00 A00 A00 A00 A00 A00 A00 A0	3A F9 3F 62 FF 32 00 3E 38 1B 00 3E 36 50 60 00 02 FD 00 01		Alpcrts SETINV SETNORM SETIFLG SETKBD INPORT INPRT SETVID OUTPORT OUTPRT IOPRT IOPRT1 IOPRT2	STA DEX BPL RTS LDY BNE LDY STY RTS LDA STA LDA LDA LDA LDA LDA LDA AND BEQ ORA LDY BEQ CORA LDY STA RTS NOP NOP	PCL,X AlPCLP #\$3F SETIFLG #\$FF INVFLG #\$00 A2L #KSWL #KSWL #KSWL #COUTI A2L #COUTI A2L #\$0F IOPRT1 #\$00 IOPRT2 #COUT1/256 LOC0,X LOC1,X	SET FOR INVERSE VID VIA COUT1 SET FOR NORMAL VID SIMULATE PORT #0 INPUT SPECIFIED (KEYIN ROUTINE) SIMULATE PORT #0 OUTPUT SPECIFIED (COUT1 ROUTINE) SET RAM IN/OUT VECTORS
FE7D: FE7D: FE80: FE80: FE84: FE84: FE80: FE80: FE80: FE80: FE91: FE91: FE91: FE91: FE91: FE92: FE95: FE95: FE95: FE95: FE95: FE85: FE95: FE95: FE95: FE95: FE95: FE95: FE95: FE95: FE95: FE85: FE85: FE85: FE85: FE85: FE85: FE85: FE85: FE85: FE85: FE85: FE85: FE85: FE85: FE85: FE85: FE85: FE85: FE95: FE95: FE95: FE85: FE85: FE85: FE85: FE85: FE85: FE85: FE85: FE85: FE95: FE95: FE85: FE85: FE85: FE85: FE85: FE85: FE85: FE85: FE85: FE85: FE95: FE95: FE85:	CA 10 600 A00 A00 A00 A00 A00 A00 A00 A00 A0	3A F9 3F 2 FF 32 00 3E 38 1B 00 3E 36 F0 06 C0 00 2 FD 06 00 01 00	E0	Alpcrts Setinv Setiflg SetkBD INPORT INPRT SETVID OUTPORT IOPRT IOPRT1 IOPRT2 XBASIC	STA DEX BPL RTS LDY STY RTS LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA DA STA LDA DA STA LDA DA STA LDA DA STA LDA STA JDA STA JDA STA JDA STA JDA STA JDA STA JDA STA JDA STA JDA STA JDA STA JDA STA JDA STA JDA STA JDA STA STA STA STA STA STA STA STA STA ST	PCL,X AlPCLP #\$3F SETIFLG #\$FF INVFLG #\$00 A2L #KSWL #KSWL #KSWL #COUTI A2L #COUTI A2L #COUTI A2L #\$0F IOPRT1 #10ADR/256 #\$00 IOPRT2 #COUT1/256 LOC0,X LOC1,X BASIC	SET FOR INVERSE VID VIA COUT1 SET FOR NORMAL VID SIMULATE PORT #0 INPUT SPECIFIED (KEYIN ROUTINE) SIMULATE PORT #0 OUTPUT SPECIFIED (COUT1 ROUTINE) SET RAM IN/OUT VECTORS

T DDO.	20	75	ਸ਼ਾਸ਼	GO	TSB	Alpc	ADR TO PC IF SPECID
	20	210	D D	90	JCD	DECEODE	NDR TO FC IF SFEC D
FEB9:	20	31	FF		JSR	RESTORE	RESTORE META REGS
FEBC:	6C	3A	00		JMP	(PCL)	GO TO USER SUBR
FEBF:	4C	D7	FA	REGZ	JMP	REGDSP	TO REG DISPLAY
FEC2:	C6	34		TRACE	DEC	YSAV	
FEC4.	20	75	FE	STED7	TSR	AIPC	ADR TO PO IF SPECID
FEC7.	10	12	т D 17 Л	DIDI 2	TMD	CUED	MAKE ONE CHED
FEC/:	4C	43	ΓA		JMP	STEP	TAKE ONE STEP
FECA:	4C	F8	03	USR	JMP	USRADR	TO USR SUBR AT USRADR
FECD:	A9	40		WRITE	LDA	#\$40	
FECF:	20	C 9	FC		TSR	HEADR	WRITE 10-SEC HEADER
EED2.	20	27	10		TDV	#¢27	
FEDZ:	AU	21		7		# \$ 2 /	
FED4:	A2	00		WRI	LDX	#\$00	
FED6:	41	3C			EOR	(AlL,X)	
FED8:	48				рна	,	
FEDQ.	71	30			TDA	(A]T V)	
FED9.	AL	50			LDA	(AIL,A)	
FEDB:	20	ΕD	FΕ		JSR	WRBYTE	
FEDE:	20	ΒA	FC		JSR	NXTAl	
FEEl:	A0	1D			TIDY	#\$1D	
	6.8					. , ===	
PDDJ.	00					LTD ]	
FEE4:	90	EE			BCC	WRI	
FEE6:	A0	22			LDY	#\$22	
FEE8:	20	ED	FE		JSR	WRBYTE	
FFFD.	<b>₽</b> 0	4 D			PEO		
FEED.	10	40			BEQ		
FEED:	A2	Τ0		WRBYTE	LDX	#\$10	
FEEF:	0A			WRBYT2	ASL	A	
FEF0:	20	D6	FC		JSR	WRBTT	
FFF2.	50	<u>Б</u> О	10		DNE	WDDVM2	
rers:	00	гA			DINE	WRDIIZ	
FEF5:	60				RTS		
FEF6:	20	00	FE	CRMON	JSR	BLl	HANDLE A CR AS BLANK
FEF9:	68				PT.A		THEN POP STACK
	60						AND DEN EO NON
F.F.L.Y.	68				PLA		AND RIN TO MON
FEFB:	D0	6C			BNE	MONZ	
FEFD:	20	FA	FC	READ	JSR	RD2BIT	FIND TAPEIN EDGE
FF00.	λQ	16			TDA	#\$16	
<b>FF00</b> .	20	10	ПО				DEL NY 2 E GEGOVEG
F.E.07:	20	69	FC		JSR	HEADR	DELAY 3.5 SECONDS
FF05:	85	2E			STA	CHKSUM	INIT CHKSUM=\$FF
FF07:	20	FA	FC		JSR	RD2BIT	FIND TAPEIN EDGE
FFOA.	ΔÔ	21		202	TDY	#\$21	LOOK FOR SYNC BIT
FFOA.	20	27	ПО	KD2			LOOK FOR SINC DIT
F.F.OC :	20	F.D	FC		JSR	RDBIT	(SHORT 0)
FFOF:	в0	F9			BCS	RD2	LOOP UNTIL FOUND
FF11:	20	FD	FC		JSR	RDBIT	SKIP SECOND SYNC H-CYCLE
FF1/.	<u>م</u>	38			T.DV	#¢3₽	INDEX FOR 0/1 TEST
FF14.	A0	50		553		π950 Π02	INDEX FOR 0/1 IESI
F.F.T.0:	20	EC	FC	RD3	JSR	RDBYTE	READ A BYTE
FF19:	81	3C			STA	(AlL,X)	STORE AT (Al)
FF1B:	45	2E			EOR	CHKSUM	
. תושם	05	25			CULY	CURCIIM	UDDATE BUNNING CURCUM
FFID.	05				SIA	CHRSOM	UPDAIL KONNING CHRSOM
F. F. T F. :	20	BA	FC		JSR	NXTAL	INC AL, COMPARE TO AZ
FF22:	A0	35			LDY	#\$35	COMPENSATE 0/1 INDEX
FF24:	90	F0			BCC	RD3	LOOP UNTIL DONE
FF26.	20	FC	FC		TCP	PDBVTF	PEAD CHKSIIM BYTTE
	~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~	Ľ	гc		JAK	RDBIIL	KEAD CHRSOM BILL
FF20.	20	<u> </u>			CMP		
FF29:	C5	2E			0	СПКБОМ	
FF29: FF2B:	C5 F0	2E 0D			BEQ	BELL	GOOD, SOUND BELL AND RETURN
FF29: FF2B: FF2D:	C5 F0 A9	2E 0D C5		PRERR	BEQ	BELL #\$C5	GOOD, SOUND BELL AND RETURN
FF29: FF2B: FF2D:	C5 F0 A9	2E 0D C5	חש	PRERR	BEQ LDA	BELL #\$C5	GOOD, SOUND BELL AND RETURN
FF29: FF2B: FF2D: FF2F:	C5 F0 A9 20	2E 0D C5 ED	FD	PRERR	BEQ LDA JSR	BELL #\$C5 COUT	GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL
FF29: FF2B: FF2D: FF2F: FF32:	C5 F0 A9 20 A9	2E 0D C5 ED D2	FD	PRERR	BEQ LDA JSR LDA	BELL #\$C5 COUT #\$D2	GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL
FF29: FF2B: FF2D: FF2F: FF32: FF34:	C5 F0 A9 20 A9 20	2E 0D C5 ED D2 ED	FD FD	PRERR	BEQ LDA JSR LDA JSR	BELL #\$C5 COUT #\$D2 COUT	GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL
FF29: FF2B: FF2D: FF2F: FF32: FF34: FF37:	C5 F0 A9 20 A9 20 20	2E 0D C5 ED D2 ED ED	FD FD FD	PRERR	BEQ LDA JSR LDA JSR JSR	BELL #\$C5 COUT #\$D2 COUT COUT	GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL
FF29: FF2B: FF2D: FF2F: FF32: FF34: FF37:	C5 F0 A9 20 A9 20 20 20	2E 0D C5 ED D2 ED ED	FD FD FD	PRERR	BEQ LDA JSR LDA JSR JSR	BELL #\$C5 COUT #\$D2 COUT COUT	GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL
FF29: FF2B: FF2D: FF2F: FF32: FF34: FF37: FF3A:	C5 F0 A9 20 A9 20 20 20 A9	2E 0D C5 ED D2 ED ED 87	FD FD FD	PRERR	BEQ LDA JSR LDA JSR JSR LDA	BELL #\$C5 COUT #\$D2 COUT COUT #\$87 COUT	GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN
FF29: FF2B: FF2D: FF2F: FF32: FF34: FF37: FF3A: FF3C:	C5 F0 A9 20 A9 20 20 20 A9 4C	2E 0D C5 ED D2 ED ED 87 ED	FD FD FD FD	PRERR	BEQ LDA JSR LDA JSR JSR LDA JMP	BELL #\$C5 COUT #\$D2 COUT COUT #\$87 COUT	GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN
FF29: FF2B: FF2D: FF2F: FF32: FF34: FF37: FF3A: FF3C: FF3F:	C5 F0 A9 20 A9 20 20 20 A9 4C A5	2E 0D C5 ED 22 ED 87 ED 48	FD FD FD FD	PRERR BELL RESTORE	BEQ LDA JSR LDA JSR JSR LDA JMP LDA	BELL #\$C5 COUT #\$D2 COUT COUT #\$87 COUT STATUS	GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS
FF29: FF2B: FF2D: FF2F: FF32: FF34: FF37: FF3A: FF3C: FF3F: FF41:	C5 F0 A9 20 A9 20 20 20 A9 4C A5 48	2E 0D C5 ED 2 ED ED 87 ED 48	FD FD FD FD	PRERR BELL RESTORE	BEQ LDA JSR LDA JSR LDA JMP LDA PHA	BELL #\$C5 COUT #\$D2 COUT COUT #\$87 COUT STATUS	GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS USED BY DEBUG SOFTWARE
FF29: FF2B: FF2D: FF2F: FF32: FF34: FF37: FF3A: FF3A: FF3F: FF41: FF42:	C5 F0 A9 20 A9 20 20 A9 4C A5 48 A5	2E 0D C5 ED 22 ED ED 87 ED 48 45	FD FD FD	PRERR BELL RESTORE	BEQ LDA JSR LDA JSR LDA JMP LDA PHA LDA	BELL #\$C5 COUT #\$D2 COUT COUT \$\$87 COUT STATUS ACC	GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS USED BY DEBUG SOFTWARE
FF29: FF2B: FF2D: FF2F: FF32: FF34: FF37: FF3A: FF3C: FF3F: FF41: FF42:	C5 F0 A9 20 A9 20 20 A9 4C A5 48 A5	2E 0D C5 ED D2 ED ED 87 ED 48 45	FD FD FD	PRERR BELL RESTORE	BEQ LDA JSR LDA JSR LDA JMP LDA PHA LDA	BELL #\$C5 COUT #\$D2 COUT COUT #\$87 COUT STATUS ACC	GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS USED BY DEBUG SOFTWARE
FF29: FF2B: FF2D: FF2F: FF32: FF34: FF37: FF3A: FF3A: FF3F: FF3F: FF41: FF42: FF44:	C5 F0 A9 20 A9 20 20 A9 20 A9 4C A5 48 A5 A6	2E 0D C5 ED 22 ED ED 87 ED 48 45 46	FD FD FD	PRERR BELL RESTORE RESTR1	BEQ LDA JSR LDA JSR LDA JMP LDA PHA LDA LDA	BELL #\$C5 COUT #\$D2 COUT COUT #\$87 COUT STATUS ACC XREG WDFC	GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS USED BY DEBUG SOFTWARE
FF29: FF2B: FF2D: FF32F: FF32: FF37: FF37: FF37: FF37: FF3F: FF41: FF42: FF44: FF44:	C5 F0 A9 20 A9 20 A9 20 A9 4C A5 48 A5 A6 A4	2E 0D C5 ED 22 ED 87 ED 48 45 46 47	FD FD FD	PRERR BELL RESTORE RESTR1	BEQ LDA JSR LDA JSR LDA JMP LDA PHA LDA LDX LDY	BELL #\$C5 COUT #\$D2 COUT COUT COUT STATUS ACC XREG YREG	GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS USED BY DEBUG SOFTWARE
FF29: FF2B: FF2D: FF2F: FF32: FF32: FF37: FF3A: FF37: FF3F: FF41: FF42: FF46: FF48:	C5 F0 A9 20 A9 20 A9 4C A5 48 A5 A6 A4 28	2E 0D C5 ED ED ED 87 ED 48 45 46 47	FD FD FD	PRERR BELL RESTORE RESTR1	BEQ LDA JSR LDA JSR LDA JMP LDA PHA LDA LDA LDY PLP	BELL #\$C5 COUT #\$D2 COUT COUT #\$87 COUT STATUS ACC XREG YREG	GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS USED BY DEBUG SOFTWARE
FF29: FF29: FF20: FF27: FF32: FF37: FF37: FF37: FF37: FF37: FF41: FF44: FF44: FF48: FF49:	C5 F0 A9 20 A9 20 A9 20 A9 4C A5 48 A5 A6 A4 28 60	2E 0D ED ED 2 ED 87 ED 48 45 46 47	FD FD FD	PRERR BELL RESTORE RESTR1	BEQ LDA JSR LDA JSR LDA JMP LDA PHA LDA LDA LDA LDY PLP RTS	BELL #\$C5 COUT #\$D2 COUT #\$87 COUT STATUS ACC XREG YREG	GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS USED BY DEBUG SOFTWARE
FF29: FF2B: FF2D: FF2F: FF32: FF32: FF37: FF37: FF37: FF37: FF34: FF41: FF42: FF44: FF48: FF49: FF49:	C5 F0 A9 20 A9 20 20 A9 20 20 A9 20 20 A9 20 20 A9 4C A5 48 A5 A6 A4 28 60 85	2E 0D C5 ED 2ED ED 87 ED 48 45 46 47	FD FD FD	PRERR BELL RESTORE RESTR1	BEQ LDA JSR LDA JSR LDA JMP LDA LDA LDA LDX LDY PLP RTS STA	BELL #\$C5 COUT #\$D2 COUT COUT STATUS ACC XREG YREG	GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS USED BY DEBUG SOFTWARE
FF29: FF2B: FF2D: FF2F: FF32: FF32: FF37: FF3A: FF37: FF3F: FF41: FF42: FF46: FF46: FF49: FF42:	C5 F0 A9 20 A9 20 20 A9 4C A5 48 A5 A6 A4 28 60 5	2E 0D C5 ED ED ED ED 87 ED 48 45 46 47	FD FD FD	PRERR BELL RESTORE RESTR1	BEQ LDA JSR LDA JSR LDA JMP LDA PHA LDA LDA LDY PLP RTS STA	BELL #\$C5 COUT #\$D2 COUT COUT #\$87 COUT STATUS ACC XREG YREG ACC	GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS USED BY DEBUG SOFTWARE
FF29: FF29: FF227: FF327: FF37: FF37: FF37: FF37: FF37: FF41: FF42: FF44: FF48: FF48: FF48: FF44: FF44: FF44:	C5 F0 A9 20 A9 20 A9 20 A9 4C A5 48 A5 A6 A4 28 60 85 86	2E 0D C5 ED 2 ED 87 ED 48 45 46 47 45 46	FD FD FD	PRERR BELL RESTORE RESTR1 SAVE SAVE	BEQ LDA JSR LDA JSR LDA JMP LDA LDA LDA LDA LDA LDA LDY PHA LDA STA STA	BELL #\$C5 COUT #\$D2 COUT #\$87 COUT STATUS ACC XREG YREG ACC XREG	GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS USED BY DEBUG SOFTWARE SAVE 6502 REG CONTENTS
FF29: FF29: FF27: FF327: FF37: FF37: FF37: FF37: FF37: FF37: FF41: FF44: FF44: FF48: FF49: FF42: FF42:	C5 F0 20 20 20 20 20 20 20 20 20 20 20 20 20	2E 0D C5 ED 2E D2 ED 87 ED 48 45 46 47 45 46 47	FD FD FD	PRERR BELL RESTORE RESTR1 SAVE	BEQ LDA JSR LDA JSR LDA JMP LDA LDA LDA LDA LDY PLP RTS STA STX STY	BELL #\$C5 COUT #\$D2 COUT COUT STATUS ACC XREG YREG ACC XREG YREG	GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS USED BY DEBUG SOFTWARE SAVE 6502 REG CONTENTS
FF29: FF2B: FF2D: FF2F: FF32: FF32: FF37: FF37: FF37: FF37: FF41: FF42: FF46: FF48: FF42: FF42: FF42:	C5 F0 20 20 20 20 20 20 20 20 20 20 20 20 20	2E 0D C5 ED ED ED ED 87 ED 48 45 46 47 45 46 47	FD FD FD	PRERR BELL RESTORE RESTR1 SAVE SAV1	BEQ LDA JSR LDA JSR LDA JMP LDA LDA LDA LDA LDY PLP RTS STA STY PHP	BELL #\$C5 COUT #\$D2 COUT COUT #\$87 COUT STATUS ACC XREG YREG ACC XREG YREG	GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS USED BY DEBUG SOFTWARE SAVE 6502 REG CONTENTS
FF29: FF29: FF227: FF327: FF327: FF37: FF37: FF37: FF37: FF341: FF442: FF442: FF48: FF48: FF49: FF49: FF49: FF49: FF40:	C5 F0 20 20 20 20 20 20 20 20 20 20 20 20 20	2E 0D C5 ED ED ED 2 ED 48 45 46 47 45 46 47	FD FD FD	PRERR BELL RESTORE RESTR1 SAVE SAVE	BEQ LDA JSR LDA JSR LDA JMP LDA LDA LDA LDA LDA LDA LDA STA STA STA STY PHP PIA	ACC XREG YREG	GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS USED BY DEBUG SOFTWARE SAVE 6502 REG CONTENTS
FF29: FF29: FF227: FF327: FF37: FF37: FF37: FF37: FF37: FF37: FF41: FF42: FF44: FF44: FF49: FF44: FF42: FF450: FF50: FF51:	C 50 A 9 20 A 9 20 A 9 20 A 9 20 A 9 20 A 9 4C A 9 A 5 A 6 A 4 8 5 86 84 08 86 84 08 86 86 86 86 86 86 86 86 86 86 86 86 86	2E 0D C5 ED ED ED ED 87 ED 87 ED 48 45 46 47 45 46 47	FD FD FD	PRERR BELL RESTORE RESTR1 SAVE SAV1	BEQ LDA JSR LDA JSR LDA JMP LDA LDA LDA LDA LDA LDY PLP RTS STA STX STY PHP PLA	BELL #\$C5 COUT #\$D2 COUT COUT #\$87 COUT STATUS ACC XREG YREG ACC XREG YREG	GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS USED BY DEBUG SOFTWARE SAVE 6502 REG CONTENTS
FF29: FF29: FF20: FF27: FF32: FF37: FF37: FF37: FF37: FF37: FF41: FF42: FF44: FF44: FF44: FF44: FF44: FF45: FF50: FF51: FF51:	C 5 F 0 20 20 20 20 20 20 20 20 20 20 20 20 20	2E 0D C5 ED ED ED 87 ED 48 45 46 47 45 46 47 48	FD FD FD	PRERR BELL RESTORE RESTR1 SAVE SAV1	BEQ LDA JSR LDA JSR LDA JMP LDA LDA LDA LDA LDY PLP RTS STA STX STY PHP PLA STA	BELL #\$C5 COUT #\$D2 COUT COUT #\$87 COUT STATUS ACC XREG YREG ACC XREG YREG STATUS	GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS USED BY DEBUG SOFTWARE SAVE 6502 REG CONTENTS
FF29: FF29: FF27: FF32: FF37: FF37: FF37: FF37: FF37: FF37: FF41: FF44: FF48: FF48: FF48: FF48: FF48: FF450: FF50: FF51: FF52: FF52: FF524:	C 50 A 20 A 20	2E 0D C5 ED ED ED 87 ED 48 45 46 47 45 46 47 45	FD FD FD	PRERR BELL RESTORE RESTR1 SAVE SAV1	BEQ LDA JSR LDA JSR LDA JMP LDA LDA LDA LDA LDA LDA LDA STA STA STA STA STA STA STA STA STA ST	BELL #\$C5 COUT #\$D2 COUT COUT #\$87 COUT STATUS ACC XREG YREG ACC XREG YREG STATUS	GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS USED BY DEBUG SOFTWARE SAVE 6502 REG CONTENTS
FF29: FF29: FF227: FF327: FF37: FF37: FF37: FF37: FF37: FF37: FF37: FF41: FF42: FF44: FF49: FF44: FF42: FF42: FF51: FF52: FF52: FF55: FF55:	C 50 F 90 2 00 2 00 2 00 2 00 2 00 2 00 2 00 2	2E 0D C5 ED ED ED 87 ED 87 ED 48 45 46 47 45 46 47 48 49	FD FD FD	PRERR BELL RESTORE RESTR1 SAVE SAV1	BEQ LDA JSR LDA JSR LDA JMP LDA LDA LDA LDA LDA LDA LDA STA STA STX STY PHP PLA STX STX	BELL #\$C5 COUT #\$D2 COUT COUT #\$87 COUT STATUS ACC XREG YREG ACC XREG YREG STATUS SPNT	GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS USED BY DEBUG SOFTWARE SAVE 6502 REG CONTENTS
FF29: FF29: FF20: FF27: FF32: FF37: FF37: FF37: FF37: FF37: FF37: FF41: FF42: FF44: FF44: FF44: FF44: FF44: FF450: FF51: FF52: FF54: FF57.	C 50 C 50 20 20 20 20 20 20 20 20 20 2	2E 0D C5 ED ED ED 87 ED 48 45 46 47 45 46 47 48 49	FD FD FD	PRERR BELL RESTORE RESTR1 SAVE SAV1	BEQ LDA JSR LDA JSR JSR LDA JMP LDA LDA LDA LDA LDY PLP RTS STA STX STX STX STX STX STX STX STX CLD	BELL #\$C5 COUT #\$D2 COUT COUT STATUS ACC XREG YREG ACC XREG YREG STATUS SPNT	GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS USED BY DEBUG SOFTWARE SAVE 6502 REG CONTENTS
FF29: FF29: FF27: FF32: FF37: FF37: FF37: FF37: FF37: FF41: FF42: FF44: FF48: FF48: FF48: FF48: FF49: FF42: FF50: FF51: FF52: FF55:	205 CF09209209 4054 405 808 808 808 808 808 808 808 8	2E 0D C5 ED ED ED 87 ED 48 45 46 47 45 46 47 48 49	FD FD FD	PRERR BELL RESTORE RESTR1 SAVE SAV1	BEQ LDA JSR LDA JSR LDA JSR LDA JMP LDA LDA LDA LDA LDA LDA LDA STA STA STA STX STY PHP PLA STA TSX STX CLD	BELL #\$C5 COUT #\$D2 COUT COUT #\$87 COUT STATUS ACC XREG YREG ACC XREG YREG STATUS SPNT	GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS USED BY DEBUG SOFTWARE SAVE 6502 REG CONTENTS
FF29: FF29: FF27: FF32: FF37: FF37: FF37: FF37: FF37: FF37: FF41: FF42: FF44: FF44: FF49: FF44: FF44: FF45: FF52: FF52: FF55: FF55: FF55:	205 CF0 209 200 405 405 806 808 808 808 808 808 808 808	2E 0D C5 ED D2 ED 87 ED 48 45 46 47 45 46 47 48 49	FD FD FD	PRERR BELL RESTORE RESTR1 SAVE SAV1	BEQ LDA JSR LDA JSR LDA JMP LDA LDA LDA LDA LDA LDA LDA STA STA STX STY PHP PLA STA STX STX CLD RTS	BELL #\$C5 COUT #\$D2 COUT COUT STATUS ACC XREG YREG ACC XREG YREG STATUS SPNT	GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS USED BY DEBUG SOFTWARE SAVE 6502 REG CONTENTS
FF29: FF29: FF27: FF32: FF37: FF37: FF37: FF37: FF37: FF37: FF41: FF42: FF44: FF44: FF44: FF44: FF44: FF44: FF450: FF51: FF52: FF551: FF557: FF58: FF59:	2 C50 C F 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2	2E 0D C5 ED 22 ED 87 ED 48 45 46 47 45 46 47 48 49 84	FD FD FD FD	PRERR BELL RESTORE RESTR1 SAVE SAV1 RESET	BEQ LDA JSR LDA JSR JSR LDA JMP LDA LDA LDA LDA LDY PLP RTS STA STX STX STX STX STX STX STX CLD RTS JSR	BELL #\$C5 COUT #\$D2 COUT COUT STATUS ACC XREG YREG ACC XREG YREG STATUS SPNT SETNORM	GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS USED BY DEBUG SOFTWARE SAVE 6502 REG CONTENTS
FF29: FF29: FF27: FF32: FF37: FF37: FF37: FF37: FF37: FF48: FF48: FF48: FF48: FF48: FF48: FF48: FF48: FF48: FF48: FF50: FF55: FF55: FF55: FF58: FF58: FF55:	205 CF0920920920920920920920920920920920920920	2E 0D C5 ED 2D ED 87 ED 48 45 46 47 45 46 47 48 49 84 2F	FD FD FD FD	PRERR BELL RESTORE RESTR1 SAVE SAV1	BEQ LDA JSR LDA JSR LDA JSR LDA LDA LDA LDA LDA LDA LDA LDA LDA STA STA STA STA STA STA STA STA STA ST	BELL #\$C5 COUT #\$D2 COUT COUT COUT STATUS ACC XREG YREG ACC XREG YREG STATUS SPNT SETNORM INIT	GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS USED BY DEBUG SOFTWARE SAVE 6502 REG CONTENTS
FF29: FF29: FF27: FF327: FF37: FF37: FF37: FF37: FF37: FF37: FF41: FF42: FF44: FF44: FF44: FF44: FF44: FF450: FF551: FF552: FF5555: FF555: FF555: FF555: FF555: FF555: FF5555: FF555: FF555: FF	205 F0920920920920920920920920920920920920920	2E 0D C55 ED 2ED ED 2ED 87 ED 48 45 46 47 45 46 47 48 49 84 293	FD FD FD FD FD FD	PRERR BELL RESTORE RESTR1 SAVE SAV1 RESET	BEQ LDA JSR LDA JSR LDA JSR LDA JMP LDA LDA LDA LDA LDA LDA LDA STA STY PHP PLA STA STX STY PHP PLA STX STX CLD RTS JSR JSR JSR JSR LDA JSR JSR LDA LDA LDA LDA LDA LDA LDA LDA LDA LDA	BELL #\$C5 COUT #\$D2 COUT COUT #\$87 COUT STATUS ACC XREG YREG ACC XREG YREG STATUS SPNT SETNORM INIT SETVID	GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS USED BY DEBUG SOFTWARE SAVE 6502 REG CONTENTS
FF29: FF29: FF29: FF27: FF32: FF37: FF37: FF37: FF37: FF37: FF41: FF44: FF44: FF44: FF44: FF44: FF44: FF44: FF44: FF50: FF55: FF55: FF55: FF55: FF55:	205 F0920920920920920920920920920920920920920	2E 0D C5 ED 2 ED 2 ED 2 ED 2 ED 2 ED 2 ED 2 ED	FD FD FD FD FD FE	PRERR BELL RESTORE RESTR1 SAVE SAV1 RESET	BEQ LDA JSR LDA JSR JSR LDA JMP LDA LDA LDA LDA LDA LDY PLP RTS STA STX STY PHP PLA STA TSX STY CLD RTS JSR JSR JSR STA STX STX STA STX STA STX STA STA STA STA STA STA STA STA STA STA	BELL #\$C5 COUT #\$D2 COUT COUT #\$87 COUT STATUS ACC XREG YREG ACC XREG YREG STATUS SPNT SETNORM INIT SETVID SETVID	GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS USED BY DEBUG SOFTWARE SAVE 6502 REG CONTENTS
FF29: FF29: FF27: FF32: FF37: FF37: FF37: FF37: FF37: FF41: FF42: FF44: FF48: FF48: FF48: FF48: FF48: FF48: FF50: FF51: FF55:	205 F09209209 405 405 405 805 805 805 805 805 805 805 8	2E 0D C5 ED ED ED ED ED 87 ED 48 45 46 47 45 46 47 48 49 82 F 93 89	FD FD FD FD FD FE FE FE	PRERR BELL RESTORE RESTR1 SAVE SAV1 RESET	BEQ LDA JSR LDA JSR LDA JSR LDA JMP LDA LDA LDA LDA LDA LDA LDA LDA STA STA STX STY PHP PLA STA STX STX CLD RTSX JSR JSR JSR	BELL #\$C5 COUT #\$D2 COUT COUT STATUS ACC XREG YREG ACC XREG YREG STATUS SPNT SETNORM INIT SETVID SETKBD	GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS USED BY DEBUG SOFTWARE SAVE 6502 REG CONTENTS SAVE 6502 REG CONTENTS
FF29: FF29: FF27: FF327: FF37: FF37: FF37: FF37: FF37: FF37: FF37: FF41: FF42: FF44: FF44: FF44: FF44: FF44: FF45: FF55:	205 F0920920920920920920920920920920920920920	2E0DC5D2EDD2EDD2ED02ED02E0048845466474546647484984298389	FD FD FD FD FE FE FE FE FE	PRERR BELL RESTORE RESTR1 SAVE SAV1 RESET	BEQ LDA JSR LDA JSR LDA JSR LDA JMP LDA LDA LDA LDA LDA LDA LDA LDA STA STA STX STY PHP PLA STA STX STX CLD RTS JSR JSR JSR CLD	BELL #\$C5 COUT #\$D2 COUT COUT STATUS ACC XREG YREG ACC XREG YREG STATUS SPNT SETNORM INIT SETVID SETKBD	GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS USED BY DEBUG SOFTWARE SAVE 6502 REG CONTENTS SAVE 6502 REG CONTENTS
FF29: FF29: FF27: FF32: FF37: FF37: FF37: FF37: FF37: FF37: FF41: FF44: FF44: FF44: FF44: FF44: FF44: FF44: FF44: FF55: FF55: FF55: FF55: FF55: FF55: FF55: FF55: FF55: FF65: FF65: FF66: FF66:	205 50 209 209 209 209 209 209 209 20	2E0DC5DED2ED2ED2ED2ED2ED248447454664749842F93893A	FD FD FD FD FD FE FE FE FE FE FF	PRERR BELL RESTORE RESTR1 SAVE SAV1 RESET MON	BEQ LDA JSR LDA JSR LDA JSR LDA JMP LDA LDA LDA LDA LDA LDA LDA LDA LDA LDA	BELL #\$C5 COUT #\$D2 COUT COUT STATUS ACC XREG YREG ACC XREG YREG STATUS SPNT SETNORM INIT SETVID SETKBD BELL	GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS USED BY DEBUG SOFTWARE SAVE 6502 REG CONTENTS SAVE 6502 REG CONTENTS SET SCREEN MODE AND INIT KBD/SCREEN AS I/O DEV'S MUST SET HEX MODE!
FF29: FF29: FF27: FF37: FF37: FF37: FF37: FF37: FF37: FF47: FF48: FF48: FF48: FF48: FF48: FF48: FF48: FF48: FF48: FF50: FF55: FF55: FF55: FF55: FF55: FF56: FF66: FF66: FF66:	205 50 50 200 200 200 200 200 20	2E000 C55E02E00 E002E00 877E00 484546647 4546647 4849 82F9389 849 82F9389 3AA	FD FD FD FD FD FE FE FE FE FE	PRERR BELL RESTORE RESTR1 SAVE SAV1 RESET MON	BEQ LDA JSR LDA JSR LDA JSR LDA LDA LDA LDA LDA LDA LDA LDA LDA STA STX STY PHP PLA STA STX STY CLD RTSR JSR JSR CLD JSR LDA LDA LDA LDA LDA LDA LDA LDA LDA LDA	BELL #\$C5 COUT #\$D2 COUT COUT STATUS ACC XREG YREG ACC XREG YREG STATUS SPNT SETNORM INIT SETVID SETKBD BELL #\$AA	GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS USED BY DEBUG SOFTWARE SAVE 6502 REG CONTENTS SAVE 6502 REG CONTENTS SAVE 6502 REG CONTENTS MUST SET HEX MODE!
FF29: FF29: FF27: FF37: FF37: FF37: FF37: FF37: FF37: FF37: FF37: FF41: FF42: FF44: FF44: FF44: FF44: FF44: FF44: FF45: FF55: FF55: FF55: FF55: FF55: FF55: FF55: FF56: FF66: FF66: FF66: FF66: FF66:	205 50 50 50 50 50 50 50 50 50	2E0DC5D EDD2EDD2ED2ED2 EDD2488 4546647 4546647 4849 842F389 893AA22	FD FD FD FD FD FD FD FD FD FD FD FD FD F	PRERR BELL RESTORE RESTR1 SAVE SAVE RESET	BEQ LDA JSR LDA JSR LDA JSR LDA JMP LDA LDA LDA LDA LDA LDA LDA LDA LDA STA STY PHP PLA STA STX STY PHP PLA STX STX STX JSR JSR CLD JSR STX STX STX STX STX STX STX STX STX STX	BELL #\$C5 COUT #\$D2 COUT COUT STATUS ACC XREG YREG ACC XREG YREG STATUS SPNT SETNORM INIT SETVID SETKBD BELL #\$AA BROMDE	GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS USED BY DEBUG SOFTWARE SAVE 6502 REG CONTENTS SAVE 6502 REG CONTENTS SET SCREEN MODE AND INIT KBD/SCREEN AS I/O DEV'S MUST SET HEX MODE! '*' PROMPT FOR MON
FF29: FF29: FF27: FF32: FF37: FF37: FF37: FF37: FF37: FF37: FF42: FF44: FF44: FF44: FF44: FF44: FF44: FF44: FF44: FF50: FF55:	205 F090 2092 405 405 805 805 805 805 805 805 805 8	2ED C55 ED2 ED2 ED2 ED2 ED2 ED2 ED2 ED2	FD FD FD FD FD FD FD FD FD FD FD FD FD F	PRERR BELL RESTORE RESTR1 SAVE SAV1 RESET	BEQ LDA JSR LDA JSR LDA JSR LDA JMP LDA LDA LDA LDA LDA LDA LDA LDA LDA LDA	BELL #\$C5 COUT #\$D2 COUT COUT STATUS ACC XREG YREG ACC XREG YREG STATUS SPNT SETNORM INIT SETVID SETKBD BELL #\$AA PROMPT	GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS USED BY DEBUG SOFTWARE SAVE 6502 REG CONTENTS SAVE 6502 REG CONTENTS SAVE 6502 REG CONTENTS MUST SET HEX MODE '*' PROMPT FOR MON

FF70:	20	C7	FF		JSR	ZMODE	CLEAR MON MODE, SCAN IDX
FF73:	20	Α7	FF	NXTITM	JSR	GETNUM	GET ITEM, NON-HEX
FF76:	84	34			STY	YSAV	CHAR IN A-REG
FF78:	A0	17			LDY	#\$17	X-REG=0 IF NO HEX INPUT
FF7A:	88			CHRSRCH	DEY		
FF7B:	30	E8			BMI	MON	NOT FOUND, GO TO MON
FF7D:	D9	CC	FF		CMP	CHRTBL,Y	FIND CMND CHAR IN TEL
FF80:	D0	F8			BNE	CHRSRCH	
FF82:	20	BE	FF		JSR	TOSUB	FOUND, CALL CORRESPONDING
FF85:	A4	34			LDY	YSAV	SUBROUTINE
FF87:	4C	73	$\mathbf{FF}$		JMP	NXTITM	
FF8A:	A2	03		DIG	LDX	#\$03	
FF8C:	0A				ASL	A	
FF8D:	0A				ASL	A	GOT HEX DIG,
FF8E:	0A				ASL	A	SHIFT INTO AZ
F.F.8F.:	0A				ASL	A	
FF90:	0A	2.5		NXTBIT	ASL	A	
F.F.91:	26	3 E			ROL	AZL	
FF93:	26	3 F			ROL	AZH	TRAVE V-CER TR DIC
FF95:	CA	<b>T</b> O			DEX	NVDDTD	LEAVE X=\$FF IF DIG
FF90:	TO	Г Ö 2 I		NYEDAC	BPL	NATBIT	
FF98:	A5 D0	31		NXTBAS	LDA	MODE	TE MODE TO REDO
FF9A:		00			BNE	NATE52	IF MODE IS ZERO
FF9C:	82	25				AZH, A	THEN COPY AZ TO
FF9E:	95	3D 41			STA	ALH, A	AI AND AS
FFAU:	90	4 L		NVEDCO	DIA	AJH, A	
FFAZ:	E 8	т <b>Э</b>		NXTB52	INA	NVDDAC	
FFAJ:		г Э О С			DNE	NAIDAS	
FFAD:	DU 70	00		CEDNUM	BNE	H¢00	CLEAD AC
FFA/:	AZ	217		GETNOM		# \$ U U 7 2 T	CLEAR AZ
FFA9:	80	25			STA	AZL	
FFAB:	80 D0	31	0.2	NYMCUD	5TX TDA	AZH TN V	CEM CHAD
FFAD:	с 0	00	02	NAICHK	LDA	IN,I	GEI CHAR
FFDU:	10	۳O			TNI	#¢₽0	
FFD1.	49 C0	07			CMD	#\$D0 #\$07	
FFD5.	00	0A 20			BCC	#ŞUA	
FFDJ.	60	00			ADC	#000	IF HEA DIG, THEN
FFD0.	09	00 57			CMD	#\$00 #¢₽ħ	
FFDD.	D0	r A CD			DCC	#ŞFA DTC	
FFDD:	БU 6 0	CD				DIG	
FFDF.	70	$\nabla \nabla$		TOCITO	TDY	#CO/256	DIICU UTCU ODDED
FFCO.	A9 10	гь		10208		#GU/256	SUDD ADD ON STR
FFCU.	40 D0	52	<b>rr</b>			CIIDTDT V	DUCH ION ODDED
FFC1:	10	ъэ	гг			SUDIDL,I	SUDD ON STR
FFC5.	Δ5	31				MODE	SOBK ADK ON SIK
FFC7.	A0	00		ZMODE		#\$00	CLP MODE OLD MODE
FFC9.	81 81	31		ZHODE	STA	# \$ 0 0 MODE	TO A-PEC
FFCD.	60	51			DTTC	HODE	
FFCC.	BC			CHDTRL	VEB VID	ŚBC	GO IO SOBE VIA EIS E("CTRL_C")
FFCD.	BC B2			CHKIBL	DFB	\$BC \$B2	F(CTRL-C) F("CTPL-V")
FFCF.	BF				DFB	SBE SBE	F("CTRL-F")
FFCF.	ED				DFB	\$ED	F("T")
FFD0.	FF				DFB	900 \$FF	F("V")
FFD1.	C1				DFB	\$C1	F("CTPT_K")
FFD2:	EC				DFB	SEC	F("S")
FFD3.	ΔQ				DFB	\$29	F("CTRL_P")
FFD4.	BB				DFB	ŚBB	F("CTRL-B")
FFD5.	<b>D</b> D <b>A</b> 6				DFB	\$ <b>D</b> 6	F("-")
FFD6:	A4				DFB	\$A4	F("+")
FFD7:	06				DFB	\$06	F("M") (F=EX-OR \$B0+\$89)
FFD8.	95				DFB	\$95	F("<")
FFD9:	07				DFB	\$07	F("N")
FFDA:	02				DFB	\$02	F("T")
FFDB:	05				DFB	\$05	F("L")
FFDC:	F0				DFB	\$F0	F("W")
FFDD:	00				DFB	\$00	F("G")
FFDE:	EB				DFB	ŚEB	F("R")
FFDF:	93				DFB	\$93	F(":")
FFE0:	Δ7				DFB	\$A7	F(",")
FFEl:	C.6				DFB	\$C6	F("CR")
FFE2.	99				DFR	\$99	F(BLANK)
FFE3.	B2			SUBTRI.	DFR	BASCONT-1	- ( ,
FFE4.	CQ				DFR	USR-1	
FFE5.	BE				DFR	REGZ-1	
FFE6.	CI				DFR	TRACE-1	
FFE7.	35				DFR	VFY_1	
FFE8.	80				DFR	INPRT-1	
FFE9.	C3				DFR	STEPZ-1	
FFFA.	96				DFB	OUTPRT-1	
FFEB:	AF				DFB	XBASIC-1	
FFEC:	17				DFB	SETMODE-1	
FFED:	17				DFB	SETMODE-1	
FFEE:	2B				DFB	MOVE-1	
• चचचच	ן דו				DFB	ፒ.ም]	

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

			* * * * * * * * * * * * DD	*****	* * * * * * * * * T	* * *	
			* MINI-A	ASSEM	BLER	*	
			*	_		*	
			* COPYRIC	GHT 1	977 BY	*	
			* APPLE C	JMP01	ER INC.	*	
			* ALL RIG	HTS R	ESERVED	*	
			*			*	
			* S.I	WOZNI	AK	*	
			* A.	BAUM		*	
			*******	* * * * *	* * * * * * * *	* *	
			TITLE "AI FORMAT	PPLE- EPZ	II MINI- \$2E	-AS	SEMBLER"
			LENGTH	EPZ	\$2F		
			MODE	EPZ FD7	\$31 \$33		
			YSAV	EPZ	\$34		
			L	EPZ	\$35		
			PCL	EPZ	\$3A		
			PCH	EPZ ED7	\$3B \$3D		
			A2L	EPZ	\$3E		
			A2H	EPZ	\$3F		
			A4L	EPZ	\$42		
			A4H FMT	EPZ FD7	\$43 \$11		
			IN	EOU	\$200		
			INSDS2	EQU	\$F88E		
			INSTDSP	EQU	\$F8D0		
			PRBL2	EQU	\$F94A \$F053		
			CHAR]	EQU	\$F9B4		
			CHAR2	EQU	\$F9BA		
			MNEML	EQU	\$F9C0		
			MNEMR	EQU	\$FA00		
			GETINZ	EQU	SFD67		
			COUT	EQU	\$FDED		
			BLl	EQU	\$FE00		
			Alpclp	EQU	\$FE78		
			GETNUM	EQU	SFF3A SFF27		
			TOSUB	EQU	\$FFBE		
			ZMODE	EQU	\$FFC7		
			CHRTBL	EQU	\$FFCC		
F500.	ΨQ	81	DET	ORG	\$£500 #¢81		TS EMT COMPATIBLE
F502:	4A	01	KEL	LSR	A		WITH RELATIVE MODE?
F503:	D0	14		BNE	ERR3		NO.
F505:	A4	3F		LDY	A2H		
F50/:	A6	3E 01		LDX BNF	AZL REL2		DOUBLE DECREMENT
F50B:	88	01		DEY	RUUZ		
F50C:	CA		REL2	DEX			
F50D:	8A			TXA			
F50E: F50F•	18 E2	3 Δ		CLC	PCL		FORM ADDR-PC-2
F511:	85	3E		STA	A2L		I ONT ADDR-FC=2
F513:	10	01		BPL	REL3		
F515:	C8			INY			
F.270:	98		KEL3	ΊΥA			

P 1 1 / 7	ES	סר			SBC	PCH	
FE10.	50	60		2 תחש	DNE	EDD	EDDOD TE NI DYME DDANCU
F519:	00	08		ERRS	BNE	ERR	ERROR IF >1-BITE BRANCH
F.2TB:	A4	ΖF.		FINDOP	LDY	LENGTH	
F51D:	в9	3D	00	FNDOP2	LDA	AlH,Y	MOVE INST TO (PC)
F520:	91	3A			STA	(PCL),Y	
F522:	88				DEY	( ),	
T 522.	10	<b>T</b> O				ENDODO	
F 5 2 3 :	TO	гð			BPL	FNDOPZ	
F525:	20	ΤA	FC		JSR	CURSUP	
F528:	20	lA	FC		JSR	CURSUP	RESTORE CURSOR
F52B:	20	D0	F8		JSR	TNSTDSP	TYPE FORMATTED LINE
F52D.	20	50	F0		TCD	DCADT	UDDAME DC
FOZE:	20	55	<u> </u>		JSK	PCADJ	OPDAIL PC
F531:	84	3B			STY	PCH	
F533:	85	3A			STA	PCL	
F535:	4C	95	F5		JMP	NXTLINE	GET NEXT LINE
E530.	20	סד		EVKENON3	TCD	TOCILD	CO TO DELIM UNDIER
1550.	20		LL	TAKEHONS	TDI	10500	GO TO DELIM NANDLEK
F.23B:	A4	34			LDY	YSAV	RESTORE Y-INDEX
F53D:	20	Α7	$\mathbf{FF}$	FAKEMON	JSR	GETNUM	READ PARAM
F540:	84	34			STY	YSAV	SAVE Y-INDEX
F542:	A0	17			TDY	#\$17	TNTT DELIMITER INDEX
E544.	00	- /		ENKEWON2	DEV	" + <b>=</b> /	CUECE NEVE DELTM
FJ44.	20	4.5		FAREHONZ	DEI		CHECK NEXT DELIM
F545:	30	4 B			BWT	RESETZ	ERR IF UNRECOGNIZED DELIM
F547:	D9	CC	FF		CMP	CHRTBL,Y	COMPARE WITH DELIM TABLE
F54A:	D0	F8			BNE	FAKEMON2	NO MATCH
F54C:	C 0	15			CPY	#\$15	MATCH, IS IT CR?
FE4F.		- D			DNE	TAKEWON2	NO HANDLE IN MONITOD
F 54E:	00	E 8			BNE	FAREMONS	NO, HANDLE IT IN MONITOR
F550:	A5	31			LDA	MODE	
F552:	A0	00			LDY	#\$0	
F554:	C6	34			DEC	YSAV	
F556.	20	00	ਸ਼ਾਸ਼		TCP	BT.1	HANDLE CP OUTSTDE MONTTOP
T 550.	20	00			TVD		HANDLE CK OUISIDE MONIIOK
F559:	4C	95	F.2		JMP	NXTLINE	
F55C:	A5	3D		TRYNEXT	LDA	AlH	GET TRIAL OPCODE
F55E:	20	8E	F8		JSR	INSDS2	GET FMT+LENGTH FOR OPCODE
F561:	AA				ТАХ		
F562.	חם	00	ር እ		TDA	MNEMD V	CET TOWED MNEMONIC DVTE
F 502.	60	00	ΓA		LDA	MINEMA, A	GET LOWER MNEMONIC BITE
F.262:	C5	42			CMP	A4L	MATCH?
F567:	D0	13			BNE	NEXTOP	NO, TRY NEXT OPCODE.
F569:	BD	C0	F9		LDA	MNEML,X	GET UPPER MNEMONIC BYTE
F56C:	C5	43			CMP	А4Н	MATCH?
F56F.	<u>۵</u> 0	00			DNF	NEVTOD	
FJOE.	50	00			DNE	NEXIOF	NO, IKI NEXI OFCODE
F570:	A5	44			LDA	FMT	
F572:	Α4	2E			LDY	FORMAT	GET TRIAL FORMAT
F574:	C0	9D			CPY	#\$9D	TRIAL FORMAT RELATIVE?
F576.	F٥	88			BEO	RET.	VES
FE70.	0	25		NDET	CMD	EODMAE	CAME FORMADO
F 5 7 8 .	23	213		NKLL	CHF	FURMAI	SAME FORMAL:
F5/A:	F.0	9 F.			BEQ	FINDOP	YES.
F57C:	C6	3D		NEXTOP	DEC	AlH	NO, TRY NEXT OPCODE
F57E:	D0	DC			BNE	TRYNEXT	
F580.	E6	44			TNC	FMT	NO MORE TRY WITH LEN=2
T 500.	20	25			DEC	T 111	WAG 1-2 ALDEADY2
F DOZ:		30			DEC	ц 	WAS L-Z ALREADI:
F584:	F'0	D6			BEQ	TRYNEXT	NO.
F586:	Α4	34		ERR	LDY	YSAV	YES, UNRECOGNIZED INST.
F588:	98			ERR2	TYA		
	ΔΔ				TAΥ		
F589:					1		
F589:	20	4 م	гQ		TCD	DDDT 2	תגים האבר הפתואוו ^ האדפת
F589: F58A:	20	4A	F9		JSR	PRBL2	PRINT ^ UNDER LAST READ
F589: F58A: F58D:	20 A9	4A DE	F9		JSR LDA	PRBL2 #\$DE	PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR
F589: F58A: F58D: F58F:	20 A9 20	4A DE ED	F9 FD		JSR LDA JSR	PRBL2 #\$DE COUT	PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION.
F589: F58A: F58D: F58F: F592:	20 A9 20 20	4A DE ED 3A	F9 FD FF	RESETZ	JSR LDA JSR JSR	PRBL2 #\$DE COUT BELL	PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION.
F589: F58A: F58D: F58F: F592: F595:	20 A9 20 20 20	4A DE ED 3A Al	F9 FD FF	RESETZ	JSR LDA JSR JSR LDA	PRBL2 #\$DE COUT BELL #\$Al	PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION.
F589: F58A: F58D: F58F: F592: F595:	20 A9 20 20 A9	4A DE ED 3A Al	F9 FD FF	RESETZ NXTLINE	JSR LDA JSR JSR LDA	PRBL2 #\$DE COUT BELL #\$A1 DBCMDT	PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '!'
F589: F58A: F58D: F58F: F592: F595: F597:	20 A9 20 20 A9 85	4A DE ED 3A Al 33	F9 FD FF	RESETZ NXTLINE	JSR LDA JSR JSR LDA STA	PRBL2 #\$DE COUT BELL #\$A1 PROMPT	PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '!' INITIALIZE PROMPT
F589: F58A: F58D: F58F: F592: F595: F597: F599:	20 A9 20 20 A9 85 20	4A DE ED 3A A1 33 67	F9 FD FF FD	RESETZ NXTLINE	JSR LDA JSR JSR LDA STA JSR	PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ	PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '!' INITIALIZE PROMPT GET LINE.
F589: F58A: F58D: F58F: F592: F595: F597: F599: F59C:	20 A9 20 20 A9 85 20 20	4A DE ED 3A A1 33 67 C7	F9 FD FF FD FF	RESETZ NXTLINE	JSR LDA JSR JSR LDA STA JSR JSR	PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE	PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '!' INITIALIZE PROMPT GET LINE. INIT SCREEN STUFF
F589: F58A: F58D: F58F: F592: F595: F597: F599: F59C: F59F:	20 A9 20 20 A9 85 20 20 AD	4A DE ED 3A A1 33 67 C7 00	F9 FD FF FD FF 02	RESETZ NXTLINE	JSR LDA JSR LDA STA JSR JSR JSR LDA	PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN	PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '!' INITIALIZE PROMPT GET LINE. INIT SCREEN STUFF GET CHAR
F589: F58A: F58D: F58F: F592: F595: F597: F599: F59C: F59F: F592:	20 A9 20 20 A9 85 20 20 AD C9	4A DE ED 3A A1 33 67 C7 00 A0	F9 FD FF FD FF 02	RESETZ NXTLINE	JSR LDA JSR LDA STA JSR JSR LDA CMP	PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0	PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '!' INITIALIZE PROMPT GET LINE. INIT SCREEN STUFF GET CHAR ASCII BLANK?
F589: F58A: F58D: F58F: F592: F595: F597: F599: F59C: F59F: F597:	20 A9 20 20 A9 85 20 20 AD C9	4A DE ED 3A A1 33 67 C7 00 A0	F9 FD FF FD FF 02	RESETZ NXTLINE	JSR LDA JSR LDA STA JSR JSR LDA CMP	PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0 CDACE	PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '!' INITIALIZE PROMPT GET LINE. INIT SCREEN STUFF GET CHAR ASCII BLANK?
F589: F58A: F58D: F58F: F592: F597: F597: F597: F597: F597: F597: F597: F594:	20 A9 20 20 A9 85 20 20 AD C9 F0	4A DE ED 3A A1 33 67 C7 00 A0 13	F9 FD FF FD FF 02	RESETZ NXTLINE	JSR LDA JSR LDA STA JSR LDA CMP BEQ	PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0 SPACE	PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '!' INITIALIZE PROMPT GET LINE. INIT SCREEN STUFF GET CHAR ASCII BLANK? YES
F589: F58A: F58D: F58F: F592: F597: F597: F597: F597: F59F: F59F: F5A6:	20 A9 20 20 A9 85 20 20 20 AD C9 F0 C8	4A DE ED 3A A1 33 67 C7 00 A0 13	F9 FD FF FD FF 02	RESETZ NXTLINE	JSR JSR JSR LDA STA JSR JSR LDA CMP BEQ INY	PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0 SPACE	PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '!' INITIALIZE PROMPT GET LINE. INIT SCREEN STUFF GET CHAR ASCII BLANK? YES
F589: F58A: F58D: F58F: F592: F597: F597: F597: F597: F597: F597: F542: F544: F546: F547:	20 A9 20 20 A9 85 20 20 AD C9 F0 C8 C9	4A DE ED 3A A1 33 67 C7 00 A0 13 A4	F9 FD FF FF 02	RESETZ NXTLINE	JSR LDA JSR LDA STA JSR LDA CMP BEQ INY CMP	PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0 SPACE #\$A4	PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '!' INITIALIZE PROMPT GET LINE. INIT SCREEN STUFF GET CHAR ASCII BLANK? YES ASCII '\$' IN COL 1?
F589: F58D: F58D: F595: F595: F597: F597: F597: F597: F597: F597: F542: F544: F546: F547:	20 A9 20 20 A9 85 20 20 AD C9 F0 C8 C9 F0	4A DE ED 3A A1 33 67 C7 00 A0 13 A4 92	F9 FD FF FD FF 02	RESETZ NXTLINE	JSR JSR JSR JSR JSR JSR JSR LDA CMP BEQ INY CMP BEO	PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0 SPACE #\$A4 FAKEMON	PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '!' INITIALIZE PROMPT GET LINE. INIT SCREEN STUFF GET CHAR ASCII BLANK? YES ASCII '\$' IN COL 1? YES. SIMULATE MONITOR
F589: F587: F587: F597: F597: F597: F597: F597: F597: F597: F597: F547: F547: F547: F547: F547: F547: F547:	20 A9 20 20 A9 85 20 20 AD C9 F0 C8 C9 F0 88	4A DE 3A A1 33 67 C7 00 A0 13 A4 92	F9 FD FF O2	RESETZ NXTLINE	JSR LDA JSR LDA STA JSR LDA CMP BEQ INY CMP BEQ DEY	PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0 SPACE #\$A4 FAKEMON	PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '!' INITIALIZE PROMPT GET LINE. INIT SCREEN STUFF GET CHAR ASCII BLANK? YES ASCII '\$' IN COL 1? YES, SIMULATE MONITOR NO BACKUB & CHAR
F589: F588: F588: F592: F595: F597:	20 A9 20 20 A9 85 20 20 AD C9 F0 C8 C9 F0 88 20 20 A9 85 20 20 A9 85 20 20 A9 85 20 20 A9 85 20 20 A9 85 20 20 A9 85 20 20 A9 85 20 A9 85 20 A9 85 20 A9 85 20 A9 85 20 A9 85 20 A9 85 20 A9 85 20 A9 85 20 A0 A0 A0 A0 A0 A0 A0 A0 A0 A	4A DE ED 3A A1 33 67 C7 00 A0 13 A4 92	F9 FD FF 02	RESETZ NXTLINE	JSR JSR JSR JSR JSR JSR JSR JSR JSR JSR	PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0 SPACE #\$A4 FAKEMON	PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '!' INITIALIZE PROMPT GET LINE. INIT SCREEN STUFF GET CHAR ASCII BLANK? YES ASCII '\$' IN COL 1? YES, SIMULATE MONITOR NO, BACKUP A CHAR
F589: F588: F588: F592: F595: F597: F597: F597: F597: F597: F597: F542: F544: F544: F546: F549: F540:	20 A9 20 20 A9 85 20 20 AD C9 F0 C8 C9 F0 88 20 20	4A DE ED 3A A1 33 67 C7 00 A0 13 A4 92 A7	F9 FD FF O2 FF	RESETZ NXTLINE	JSR JSR JSR LDA JSR JSR LDA CMP BEQ DEY JSR	PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0 SPACE #\$A4 FAKEMON GETNUM	PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '!' INITIALIZE PROMPT GET LINE. INIT SCREEN STUFF GET CHAR ASCII BLANK? YES ASCII '\$' IN COL 1? YES, SIMULATE MONITOR NO, BACKUP A CHAR GET A NUMBER
F589: F58D: F58D: F597: F597: F597: F597: F597: F597: F597: F542: F544: F546: F547: F549: F549: F547: F547: F547: F547: F547:	20 A9 20 20 A9 85 20 20 AD C9 F0 88 20 88 20 C9 F0 88 20 C9	4A DE ED 3A A1 33 67 C7 00 A0 13 A4 92 A7 93	F9 FD FF 02 FF	RESETZ NXTLINE	JSR JSR JSR LDA STA JSR LDA CMP BEQ INY CMP BEQ DEY JSR CMP	PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0 SPACE #\$A4 FAKEMON GETNUM #\$93	PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '!' INITIALIZE PROMPT GET LINE. INIT SCREEN STUFF GET CHAR ASCII BLANK? YES ASCII '\$' IN COL 1? YES, SIMULATE MONITOR NO, BACKUP A CHAR GET A NUMBER ':' TERMINATOR?
F589: F588: F588: F592: F597:	20 A9 20 20 A9 85 20 20 AD C9 F0 88 20 C9 F0 88 20 C9 D0	4A DE ED 3A A1 33 67 C7 00 A0 13 A4 92 A7 93 D5	F9 FD FF 02 FF	RESETZ NXTLINE *	JSR JSR JSR JSR JSR JSR JSR JSR LDA CMP BEQ DEY CMP BEQ DEY CMP BNE	PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0 SPACE #\$A4 FAKEMON GETNUM #\$93 ERR2	PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '!' INITIALIZE PROMPT GET LINE. INIT SCREEN STUFF GET CHAR ASCII BLANK? YES ASCII '\$' IN COL 1? YES, SIMULATE MONITOR NO, BACKUP A CHAR GET A NUMBER ':' TERMINATOR? NO, ERR.
F589: F588: F588: F592: F595: F597:	20 A9 20 20 A9 85 20 20 AD C9 F0 C8 C9 F0 88 20 C9 F0 88 20 88 20 20 A9 85 20 A9 85 20 A9 85 20 A9 85 20 A9 85 20 A9 85 20 A9 85 20 A9 85 20 A9 85 20 A9 85 20 A9 85 20 A9 85 20 A9 85 20 A0 A0 A0 A0 A0 A0 A0 A0 A0 A	4A DE ED 3A 41 33 67 C7 00 A0 13 A4 92 A7 93 D5	F9 FD FF 02 FF	RESETZ NXTLINE *	JSR JSR JSR LDA STR JSR LDA CMP BEQ INY BEQ DEY JSR CMP BEQ DEY JSR CMP BNE TXA	PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0 SPACE #\$A4 FAKEMON GETNUM #\$93 ERR2	PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '!' INITIALIZE PROMPT GET LINE. INIT SCREEN STUFF GET CHAR ASCII BLANK? YES ASCII '\$' IN COL 1? YES, SIMULATE MONITOR NO, BACKUP A CHAR GET A NUMBER ':' TERMINATOR? NO, ERR.
F589: F588: F588: F592: F595: F597: F597: F597: F597: F597: F527: F524: F524: F524: F524: F524: F524: F524: F524: F524: F524: F524: F524: F524: F524: F524: F524: F524: F524: F525:	20 A9 20 20 A9 85 20 20 AD C9 F0 C8 C9 F0 88 20 C9 F0 88 20 C9 F0 88 20 C9 F0 C9 C9 C9 C9 C9 C9 C9 C9 C9 C9	4A DE ED 3A A1 33 67 C7 00 A0 13 A4 92 A7 93 D5	F9 FD FF 02 FF	RESETZ NXTLINE *	JSR JSR JSR JSR JSR JSR LDA JSR LDA CMP BEQ INY BEQ JSR CMP BNE DEY JSR CMP BNE SEO	PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0 SPACE #\$A4 FAKEMON GETNUM #\$93 ERR2 ERR2	PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '!' INITIALIZE PROMPT GET LINE. INIT SCREEN STUFF GET CHAR ASCII BLANK? YES ASCII '\$' IN COL 1? YES, SIMULATE MONITOR NO, BACKUP A CHAR GET A NUMBER ':' TERMINATOR? NO, ERR.
F589: F588: F588: F592: F597: F597: F597: F597: F597: F597: F597: F597: F542: F542: F543: F543: F543: F543: F543: F544: F554: F554:	20 A9 20 A9 20 20 20 20 20 20 20 20 20 20	4A DE ED 3A A1 33 67 C7 00 A0 13 A2 92 A7 93 D5 D2 272	F9 FD FF 02 FF	RESETZ NXTLINE *	JSR JSR JSR JSR JSR JSR JSR JSR LDA CMP BEQ DEY CMP BEQ DEY CMP BNE TXA BEQ	PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0 SPACE #\$A4 FAKEMON GETNUM #\$93 ERR2 ERR2	PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '!' INITIALIZE PROMPT GET LINE. INIT SCREEN STUFF GET CHAR ASCII BLANK? YES ASCII '\$' IN COL 1? YES, SIMULATE MONITOR NO, BACKUP A CHAR GET A NUMBER ':' TERMINATOR? NO, ERR. NO ADR PRECEDING COLON.
F589: F588: F588: F599: F597:	20 A9 20 20 A9 85 20 20 AD C9 F0 C8 88 20 C9 F0 88 20 C9 D0 8A F0 20	4A DE ED 3A A1 33 67 C7 00 A0 13 A4 92 A7 93 D5 D2 28	F9 FD FF 02 FF	RESETZ NXTLINE * ERR4	JSR JSR JSR JSR JSR JSR JSR JSR LDA CMP BEQ INY BEQ JSR CMP BEQ DEY JSR BNE TXA BEQ JSR	PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0 SPACE #\$A4 FAKEMON GETNUM #\$93 ERR2 ERR2 ERR2 ERR2	PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '!' INITIALIZE PROMPT GET LINE. INIT SCREEN STUFF GET CHAR ASCII BLANK? YES ASCII '\$' IN COL 1? YES, SIMULATE MONITOR NO, BACKUP A CHAR GET A NUMBER ':' TERMINATOR? NO, ERR. NO ADR PRECEDING COLON. MOVE ADR TO PCL, PCH.
F589: F588: F588: F592: F595: F597: F597: F597: F597: F597: F597: F527: F524:	20 A9 20 20 A9 85 20 20 AD C9 F0 C8 C9 F0 88 20 C9 F0 88 20 20 AD AD AD AD AD AD AD AD AD AD	4A DE ED 3A A1 33 67 C7 00 A0 13 A4 92 A7 93 D5 D2 78 03	F9 FD FF O2 FF	RESETZ NXTLINE * ERR4 SPACE	JSR JSR JSR JSR JSR JSR JSR LDA MP BEQ JSR CMP BNE JSR CMP BNE TXA BEQ JSR LDA	PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0 SPACE #\$A4 FAKEMON GETNUM #\$93 ERR2 ERR2 AlPCLP #\$3	PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '!' INITIALIZE PROMPT GET LINE. INIT SCREEN STUFF GET CHAR ASCII BLANK? YES ASCII '\$' IN COL 1? YES, SIMULATE MONITOR NO, BACKUP A CHAR GET A NUMBER ':' TERMINATOR? NO, ERR. NO ADR PRECEDING COLON. MOVE ADR TO PCL, PCH. COUNT OF CHARS IN MNEMONIC
F589: F588: F588: F592: F597: F597: F597: F597: F597: F597: F597: F597: F542: F542: F542: F543: F544: F544: F544: F5542: F5542: F5542: F5542: F5542: F5543:	20 A9 20 A9 20 A9 85 20 AD F0 C9 F0 88 20 C9 F0 88 C9 D0 88 C9 A9 80 C9 A9 80 C9 A9 80 C9 A9 80 C9 C9 C0 80 C9 C9 C9 C9 C9 C9 C9 C9 C9 C9	4A DE ED 3A A1 33 67 C7 00 A0 13 A2 92 A7 93 D5 D2 78 03 3D	F9 FD FF 02 FF	RESETZ NXTLINE * ERR4 SPACE	JSR JSR JSR JSR JSR JSR JSR JSR LDA CMP BEQ DEYR CMP BEQ DSY CMP BNE TXA BEQ JSR STA	PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0 SPACE #\$A4 FAKEMON GETNUM #\$93 ERR2 ERR2 AlPCLP #\$3 AlH	PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '!' INITIALIZE PROMPT GET LINE. INIT SCREEN STUFF GET CHAR ASCII BLANK? YES ASCII '\$' IN COL 1? YES, SIMULATE MONITOR NO, BACKUP A CHAR GET A NUMBER ':' TERMINATOR? NO, ERR. NO ADR PRECEDING COLON. MOVE ADR TO PCL, PCH. COUNT OF CHARS IN MNEMONIC
F589: F588: F588: F592: F597: F598:	20 A9 20 A9 20 A9 85 20 20 AD F0 C9 F0 88 20 C9 F0 88 20 C9 F0 88 20 20 85 20 20 AD 85 20 20 AD 85 20 20 AD 85 20 20 AD 85 20 20 AD 85 20 AD 85 20 AD 85 20 AD 85 20 AD 85 20 AD 85 20 AD 85 20 AD 85 20 AD 85 20 AD 85 20 AD 88 20 AD 88 20 AD 88 20 20 AD 88 20 20 AD 88 20 20 88 20 20 88 20 20 88 20 20 88 20 20 88 20 20 88 20 20 20 88 20 20 85 85 85 85 85 85 85 85 85 85	4A DE ED 3A A1 33 67 C7 00 13 A2 93 D5 D2 78 03 D2 78 03 23 24	F9 FD FF FF 02 FF FE	RESETZ NXTLINE * ERR4 SPACE NXTMN	JSR JSR JSR JSR JSR JSR JSR JSR LDA CMP BEQ JSR BEQ JSR BNE TXA BEQ JSR STA STA	PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0 SPACE #\$A4 FAKEMON GETNUM #\$93 ERR2 ERR2 ERR2 ERR2 AlPCLP #\$3 AlH GETNSP	PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '!' INITIALIZE PROMPT GET LINE. INIT SCREEN STUFF GET CHAR ASCII BLANK? YES ASCII '\$' IN COL 1? YES, SIMULATE MONITOR NO, BACKUP A CHAR GET A NUMBER ':' TERMINATOR? NO, ERR. NO ADR PRECEDING COLON. MOVE ADR TO PCL, PCH. COUNT OF CHARS IN MNEMONIC GET FIRST MNEM CHAR.
F589: F588: F588: F592: F595: F597: F598:	20 20 20 20 20 20 20 20 20 20	4A DE ED 3A 33 67 C7 00 A0 13 A4 92 A7 93 D5 D2 78 03 3D 34	F9 FD FF 02 FF FE FE	RESETZ NXTLINE * ERR4 SPACE NXTMN	JSR JSR JSR JSR JSR JSR JSR LDA CMP BEQ DEY JSR CMP BEQ JSR CMP BEQ JSR LDA STA JSR LDA	PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0 SPACE #\$A4 FAKEMON GETNUM #\$93 ERR2 ERR2 AlPCLP #\$3 AlH GETNSP A	PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '!' INITIALIZE PROMPT GET LINE. INIT SCREEN STUFF GET CHAR ASCII BLANK? YES ASCII '\$' IN COL 1? YES, SIMULATE MONITOR NO, BACKUP A CHAR GET A NUMBER ':' TERMINATOR? NO, ERR. NO ADR PRECEDING COLON. MOVE ADR TO PCL, PCH. COUNT OF CHARS IN MNEMONIC GET FIRST MNEM CHAR.
F589: F588: F588: F592: F597:	20 20 20 20 20 20 20 20 20 20 20 20 20 2	4A DE ED 3A A1 33 67 C7 00 A0 13 A4 92 A7 935 D2 78 03 3D 278 03 3D	F9 FD FF 02 FF FE F6	RESETZ NXTLINE * ERR4 SPACE NXTMN NXTM	JSR JSR JSR JSR JSR JSR JSR JSR JSR LDA CMP BEQ JSR STA BEQ JSR STA STA JSR LDA CMP BEQ JSR STA STA STA STA STA STA STA STA STA STA	PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0 SPACE #\$A4 FAKEMON GETNUM #\$93 ERR2 ERR2 AlPCLP #\$3 AlH GETNSP A	PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '!' INITIALIZE PROMPT GET LINE. INIT SCREEN STUFF GET CHAR ASCII BLANK? YES ASCII '\$' IN COL 1? YES, SIMULATE MONITOR NO, BACKUP A CHAR GET A NUMBER ':' TERMINATOR? NO, ERR. NO ADR PRECEDING COLON. MOVE ADR TO PCL, PCH. COUNT OF CHARS IN MNEMONIC GET FIRST MNEM CHAR.
F589: F588: F588: F592: F597: F597: F597: F597: F597: F597: F597: F597: F597: F597: F597: F597: F597: F584: F584: F584: F589: F589: F589: F588: F589: F588: F597: F588:	20 20 20 20 20 20 20 20 20 20 20 20 20 2	4A DE ED 3A 33 67 C7 00 A0 13 A4 92 A7 93 D5 D2 78 03 3D 34 BE	F9 FD FF FF 02 FF FE	RESETZ NXTLINE * ERR4 SPACE NXTMN NXTM	JSR JSR JSR JSR JSR JSR JSR JSR JSR LDA CMP BEQ JSR BEQ JSR BEQ JSR BEQ JSR STA STA STA STA	PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0 SPACE #\$A4 FAKEMON GETNUM #\$93 ERR2 ERR2 ERR2 ERR2 AlPCLP #\$3 AlH GETNSP A #\$BE	PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '!' INITIALIZE PROMPT GET LINE. INIT SCREEN STUFF GET CHAR ASCII BLANK? YES ASCII '\$' IN COL 1? YES, SIMULATE MONITOR NO, BACKUP A CHAR GET A NUMBER ':' TERMINATOR? NO, ERR. NO ADR PRECEDING COLON. MOVE ADR TO PCL, PCH. COUNT OF CHARS IN MNEMONIC GET FIRST MNEM CHAR.
F589: F588: F588: F599: F597: F597: F597: F597: F597: F597: F597: F584: F584: F584: F584: F584: F584: F584: F584: F584: F584: F584: F585: F597: F587: F587: F585: F585: F587: F585:	20         80         200         80         200         80         200         82         200         82         200         82         200         82         200         82         200         82         200         82         200         82         200         200         200         200         200         200         200         200         200         200         <	4A DE ED 3A 33 67 C7 00 A0 13 A4 92 A7 93 D5 D2 78 03 D2 78 03 234 BE C2	F9 FD FF 02 FF FE FE	RESETZ NXTLINE * ERR4 SPACE NXTMN NXTM	JSR JSR JSR JSR JSR JSR JSR JSR JSR LDA BEQ JSR CMP BEQ JSR CMP BNE JSR STA STA SSC CMP	PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0 SPACE #\$A4 FAKEMON GETNUM #\$93 ERR2 ERR2 AlPCLP #\$3 AlH GETNSP A #\$BE #\$C2	PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '!' INITIALIZE PROMPT GET LINE. INIT SCREEN STUFF GET CHAR ASCII BLANK? YES ASCII '\$' IN COL 1? YES, SIMULATE MONITOR NO, BACKUP A CHAR GET A NUMBER ':' TERMINATOR? NO, ERR. NO ADR PRECEDING COLON. MOVE ADR TO PCL, PCH. COUNT OF CHARS IN MNEMONIC GET FIRST MNEM CHAR. SUBTRACT OFFSET LEGAL CHAR?
F589: F588: F588: F592: F597:	A9         20         A0         C9         F0         C9         B0         F0         C9         B0         S0         A0         A0         A0         A0         A0         A0         A0         A0         B0	4A DE ED 3A 33 67 C7 00 A0 13 A1 92 A7 93 D5 D2 78 03 3D 278 03 34 BE 2C1	F9 FD FF 02 FF FE F6	RESETZ NXTLINE * ERR4 SPACE NXTMN NXTM	JSR JSR JSR JSR JSR JSR JSR JSR JSR JSR	PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0 SPACE #\$A4 FAKEMON GETNUM #\$93 ERR2 ERR2 A1PCLP #\$3 A1H GETNSP A #\$BE #\$C2 ERR2	PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '!' INITIALIZE PROMPT GET LINE. INIT SCREEN STUFF GET CHAR ASCII BLANK? YES ASCII '\$' IN COL 1? YES, SIMULATE MONITOR NO, BACKUP A CHAR GET A NUMBER ':' TERMINATOR? NO, ERR. NO ADR PRECEDING COLON. MOVE ADR TO PCL, PCH. COUNT OF CHARS IN MNEMONIC GET FIRST MNEM CHAR. SUBTRACT OFFSET LEGAL CHAR? NO.
F589: F588: F588: F592: F597: F507:	20 20 20 20 20 20 20 20 20 20 20 20 20 2	4A DED 3A A1 33 67 C7 00 A0 13 A4 92 A7 93 D5 D2 78 30 35 BE C2 C1	F9 FD FF 02 FF FE	RESETZ NXTLINE * ERR4 SPACE NXTMN NXTM	JSR JSR JSR JSR JSR JSR JSR JSR JSR LDA CMP BEQ JSR BEQ JSR BEQ JSR BEQ JSR STA STA STA SBC CMP BEQ JSR STA SBC SC SC SC SC SC SC SC SC SC SC SC SC SC	PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0 SPACE #\$A4 FAKEMON GETNUM #\$93 ERR2 ERR2 ERR2 AlPCLP #\$3 AlH GETNSP A #\$BE #\$C2 ERR2 A	PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '!' INITIALIZE PROMPT GET LINE. INIT SCREEN STUFF GET CHAR ASCII BLANK? YES ASCII '\$' IN COL 1? YES, SIMULATE MONITOR NO, BACKUP A CHAR GET A NUMBER ':' TERMINATOR? NO, ERR. NO ADR PRECEDING COLON. MOVE ADR TO PCL, PCH. COUNT OF CHARS IN MNEMONIC GET FIRST MNEM CHAR. SUBTRACT OFFSET LEGAL CHAR? NO. COMPRESS-LEFT JUSTIFY
F589: F588: F588: F592: F597: F598: F598: F598: F598: F598: F598: F507:	20 20 20 20 20 20 20 20 20 20 20 20 20 2	4A DEE 3A 33 67 C7 00 A0 13 A4 92 A7 93 D5 D2 78 03 3D 278 03 32 BE C2 C1	F9 FD FF 02 FF FE FE	RESETZ NXTLINE * ERR4 SPACE NXTMN NXTM	JSR JSR JSR JSR JSR JSR JSR JSR JSR LDA BEQ DEY BEQ JSR CMP BEQ JSR STA STA STA STA STA STA STA STA STA STA	PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0 SPACE #\$A4 FAKEMON GETNUM #\$93 ERR2 ERR2 AlPCLP #\$3 AlH GETNSP A #\$BE #\$C2 ERR2 A A	PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '!' INITIALIZE PROMPT GET LINE. INIT SCREEN STUFF GET CHAR ASCII BLANK? YES ASCII '\$' IN COL 1? YES, SIMULATE MONITOR NO, BACKUP A CHAR GET A NUMBER ':' TERMINATOR? NO, ERR. NO ADR PRECEDING COLON. MOVE ADR TO PCL, PCH. COUNT OF CHARS IN MNEMONIC GET FIRST MNEM CHAR. SUBTRACT OFFSET LEGAL CHAR? NO. COMPRESS-LEFT JUSTIFY
F589: F588: F588: F592: F597: F507:	20 20 20 20 20 20 20 20 20 20 20 20 20 2	4A DED 3A A1 33 67 C7 0 00 13 A4 92 A7 93 D5 D2 78 03 3D 34 BE 2 C1	F9 FD FF 02 FF FE F6	RESETZ NXTLINE * ERR4 SPACE NXTMN NXTM	JSR JSR JSR JSR JSR JSR JSR JSR JSR JSR	PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0 SPACE #\$A4 FAKEMON GETNUM #\$93 ERR2 ERR2 AlPCLP #\$3 AlH GETNSP A #\$BE #\$C2 ERR2 A A	PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '!' INITIALIZE PROMPT GET LINE. INIT SCREEN STUFF GET CHAR ASCII BLANK? YES ASCII '\$' IN COL 1? YES, SIMULATE MONITOR NO, BACKUP A CHAR GET A NUMBER ':' TERMINATOR? NO, ERR. NO ADR PRECEDING COLON. MOVE ADR TO PCL, PCH. COUNT OF CHARS IN MNEMONIC GET FIRST MNEM CHAR. SUBTRACT OFFSET LEGAL CHAR? NO. COMPRESS-LEFT JUSTIFY
F589: F588: F588: F592: F597: F598: F598: F598: F507:	A9         20         A0         C0         B0         C0	4A DE ED 3A A1 33 67 C00 A0 13 A2 92 A7 93 D5 D2 78 03 D2 78 03 D3 4 BE C2 C1 04	F9 FD FF 02 FF FE	RESETZ NXTLINE * ERR4 SPACE NXTMN NXTM	JSR JSR JSR JSR JSR JSR JSR JSR JSR LDA CMP BEQ JSR LDA CMP BEQ JSR STA SEC CMP BNE STA SEC CMP CMP BEQ JSR LDA STA SEC STA SEC SEC SEC SEC SEC SEC SEC SEC SEC SEC	PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0 SPACE #\$A4 FAKEMON GETNUM #\$93 ERR2 ERR2 ERR2 AlPCLP #\$3 AlH GETNSP A #\$BE #\$C2 ERR2 A A #\$4	PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '!' INITIALIZE PROMPT GET LINE. INIT SCREEN STUFF GET CHAR ASCII BLANK? YES ASCII '\$' IN COL 1? YES, SIMULATE MONITOR NO, BACKUP A CHAR GET A NUMBER ':' TERMINATOR? NO, ERR. NO ADR PRECEDING COLON. MOVE ADR TO PCL, PCH. COUNT OF CHARS IN MNEMONIC GET FIRST MNEM CHAR. SUBTRACT OFFSET LEGAL CHAR? NO. COMPRESS-LEFT JUSTIFY

F5CC: F5CE:	26 26 CD	42 43			ROL ROL	A4L A4H	
F5D1: F5D3:	10 C6	F8 3D			BPL DEC	NXTM2 AlH	DONE WITH 3 CHARS?
F5D5: F5D7:	F0 10	F4 E4			BEQ BPL	NXTM2 NXTMN	YES, BUT DO I MORE SHIFT NO
F5D9:	A2	05		FORM1	LDX	#\$5	5 CHARS IN ADDR MODE
F5DB:	20 84	34 34	F6	FORM2	JSR STV	GETNSP	GET FIRST CHAR OF ADDR
F5E0:	DD	B4	F9		CMP	CHAR1,X	FIRST CHAR MATCH PATTERN?
F5E3:	D0	13	7		BNE	FORM3	NO
F5E5:	20 DD	34 BA	F9		CMP	CHAR2.X	MATCHES SECOND HALF?
F5EB:	FO	0 D			BEQ	FORM5	YES.
F5ED:	BD	BA	F9		LDA	CHAR2,X	NO, IS SECOND HALF ZERO?
F5F0:	F0 C9	07 A4			CMP	FORM4 #SA4	NO.SECOND HALF OPTIONAL?
F5F4:	FO	03			BEQ	FORM4	YES.
F5F6:	A4	34		TODWO	LDY	YSAV	CLEAD DIE NO MAECH
F5F9:	88			FORM4	DEY		BACK UP 1 CHAR
F5FA:	26	44		FORM5	ROL	FMT	FORM FORMAT BYTE
F5FC:	E0	03 00			CPX	#\$3 ЕОРМ7	TIME TO CHECK FOR ADDR.
F600:	20	A7	FF		JSR	GETNUM	YES
F603:	A5	3F			LDA	A2H	
F605:	F0 E8	01			BEQ TNX	FORM6	HIGH-ORDER BYTE ZERO
F608:	86	35		FORM6	STX	L	STORE LENGTH
F60A:	A2	03			LDX	#\$3	RELOAD FORMAT INDEX
F60C:	88 86	3D		FORM7	DEY STX	AlH	SAVE INDEX
F60F:	CA				DEX		DONE WITH FORMAT CHECK?
F610:	10	C9			BPL	FORM2	NO.
F614:	A5 0A	44			ASL	A	IN LOW BITS
F615:	0A				ASL	A	
F616:	05	35			ORA	L #\$20	
F61A:	B0	20			BCS	#\$20 FORM8	ADD "\$" IF NONZERO LENGTH
F6lC:	A6	35			LDX	L	AND DON'T ALREADY HAVE IT
F61E:	F0 AQ	02 80			BEQ	FORM8 #\$80	
F622:	85	44		FORM8	STA	FMT	
F624:	84	34			STY	YSAV	
F626:	B9 C9	00 BB	02		LDA CMP	IN,Y #SBB	GET NEXT NONBLANK
F62B:	F0	04			BEQ	FORM9	YES
F62D:	C9	8D			CMP	#\$8D	CARRIAGE RETURN?
F62F:	D0 4C	80 5C	F5	FORM9	JMP	ERR4 TRYNEXT	NO, ERR.
F634:	в9	00	02	GETNSP	LDA	IN,Y	
F637:	C8	70			INY	#¢ 7.0	CEM NEVM NON DIANU CHAD
гозо: F63A:	C9 F0	AU F8			BEO	#ŞAU GETNSP	GEI NEXT NON BLANK CHAR
F63C:	60				RTS		
RCCC	10	0.2		MINACM	ORG	\$F666	
10001	40	92	гЭ	MINASM	JMP	RESETZ	

				*******	* * * * * *	******	* *
				*			*
				* APPLE-	II FLO	DATING	*
				* POINT	ROUT	INES	*
				*			*
				* COPYRI	GHT 19	977 BY	*
				* APPLE CO	OMPUTI	ER INC.	*
				*			*
				* ALL RIG	HTS RI	ESERVED	*
				*			*
				* 5.1	NOZNTZ	AK	*
				*	1021111		*
				*******	*****	******	* *
				<b>ΨΤΨΤ.Ε. "F</b> I	.02771	IG POINT	T BOUTTNES"
				STGN	EPZ	SF3	
				8201	EPZ	\$F4	
				M2	EPZ	\$F5	
				x1	EPZ	\$F8	
				м1	EPZ	\$F9	
				E	EPZ	SFC	
				OVLOC	EOU	\$3F5	
				01100	ORG	\$F425	
F425.	18			חח∆	CLC	<b>VI 42</b> 5	CLEAR CARRY
F426.	A2	02		ndb		#\$2	INDEX FOR 3-BYTE ADD
F128.	B5	F0		וחחג		лу2 м] У	INDER TOR S-DITE MDD.
F120.	75	F 5		ADDI	ADC	M2 Y	א תתא א איי איי א איי א א א א א א א א א
F12C.	95	F0			STA	MI Y	ADD A DITE OF MANIZ TO MANTI
F12C.	CA	1 )			DEX	111 <i>,</i> A	TNDEY TO NEYT MODE STONIE BYTE
F12E.	10	F7			BDL	וחחג	LOOP UNTIL DONE
F/31.	60	L /			DTD	ADDI	DEMIDN
E431.	00	52		MDI	ACT	STON	CIEND ICD OF SICN
E432.	20	27	<b>F</b> /	HDI	TCD	ADGWAD	VDC MAT OF MI TUFN CWAD WITH MO
F137.	21	л Т	1.4	ABGWAD	BTT	MI	MANTI NECATIVE?
E437.	10	05		ADSWAL		ADGWAD1	
C439.	20	74	<b>F</b> /		TCD	ECOMDI	VEC COMDIEMENT TT
C43D.	20	R4 F2	1.4		TNC	STCN	THER STON COMPLEMENTING ISP
	20	гJ		ותגשמת	CEC	BIGN	CET CARRY FOR RETURN TO MUL
	70	04		CWAD	TDV	#¢1	TNDEV FOR A RVTE CWAD
C113.	AZ	04 50		CWADI	CULV	#94 5 ] V	INDEX FOR 4 BILL SWAF.
	74	F D		SWAFT	TDY	<u>v</u> i i v	CHAD A DYME OF FYD/MANM] WIMH
C44J.		E 3			TDA	$x_2 + x_2$	SWAF A BILL OF EAF/MANIE WITH EVD/MANT2 AND LEAVE A CODY OF
C110.	01	F 3			CUL	x1 1 X	MANUTI IN F (3 DUTEC) F13 HEFD
	94	E 2			CUV DII	X2 1 X	MANII IN E (5 BIIES). E+5 05ED
C44D:	95	гэ			DEX	AZ-1,A	ADVANCE INDEX DO NEVD DVDE
C44D:	CA D0	ъЭ			DEA	CWADI	ADVANCE INDEX IO NEXI BILE
	60	гJ				SWAFT	DEMUDN
	70	017		ET O M	KID TDA	# ¢ 0 ₽	KEIUKN INIM EVDI MO 14
C451:	A9 0E	0£		FLOAT		#205 V1	INII EAFI IO 14, MUEN NORMALIZE MO ELOAM
	7 2	F 0		NODMI	TDA	M1	UTCH ODDED MANUEL DYME
E433:	A5	F 9		NORMI		MI #¢QQ	HIGH-ORDER MANTI BITE.
C457:	20	00			DMT	# \$C0	VEC DEMUDN WIMH MANMI NODMALIZED
C439:	30				DEC	KIDI VI	IES, REIORN WIIN MANII NORMALIZED
E43B:		F 8			DEC	AL M1+2	DECREMENT EXPL.
C45D:	26				ROL	MITI	CUIEM MANUEL (2 DYMEC) IEEM
E45E:	20	FA			ROL	MI+I	SHIFT MANTI (3 BITES) LEFT.
E461:	26	F'9		NODY	ROL	ML	
E403:	A5	F 8		NORM	LDA	AL NODAJ	EARL ZERU:
r405:	00	ЕE			BNE	NORMI	NO, CONTINUE NORMALIZING.
E467:	60		- /	RTSI	RTS		KETURN.
E 468:	20	A4	F4	FSUB	JSR	F'COMPL	CMPL MANTI, CLEARS CARRY UNLESS 0
F46B:	20	7B	F4	SWPALGN	JSR	ALGNSWF	P RIGHT SHIFT MANTI OR SWAP WITH
F46E:	A5	F4		FADD	LDA	X2	· · · · · · · · · · · ·
F470:	C5	F8			CMP	Xl	COMPARE EXP1 WITH EXP2.
F472:	D0	F7			BNE	SWPALGN	N 1F #,SWAP ADDENDS OR ALIGN MANTS.
F474:	20	25	F4		JSR	ADD	ADD ALIGNED MANTISSAS.
F477:	50	EA		ADDEND	BVC	NORM	NO OVERFLOW, NORMALIZE RESULT.
F479:	70	05			BVS	RTLOG	OV: SHIFT M1 RIGHT, CARRY INTO SIGN

F47B:	90	C4		ALGNSWP *	BCC	SWAP	SWAP IF CARRY CLEAR,
F47D:	A5	F9		RTAR	LLSE SI	Ml	SIGN OF MANTL INTO CARRY FOR
F47F:	0A				ASL		RIGHT ARITH SHIFT.
F480:	E6	F8		RTLOG	INC	Xl	INCR X1 TO ADJUST FOR RIGHT SHIFT
F482:	F U A 2	75 FA		RTLOG1	T'DX RFÖ	UVFL #SFA	INDEX FOR 6.BYTE RIGHT SHIFT.
F486:	76	FF		RORL	ROR	E+3,X	INDER FOR O'DITE RIGHT DHITT.
F488:	E8				INX		NEXT BYTE OF SHIFT.
F489:	D0	FΒ			BNE	ROR1	LOOP UNTIL DONE.
F48B:	20	32	F4	FMUT.	JSR	мр1	ABS VAL OF MANTL. MANT2
F48F:	65	F8		11102	ADC	Xl	ADD EXP1 TO EXP2 FOR PRODUCT EXP
F491:	20	E2	F4		JSR	MD2	CHECK PROD. EXP AND PREP. FOR MUL
F494:	18	0.4	TT 4	MITT 3	CLC		CLEAR CARRY FOR FIRST BIT.
F495: F498:	20	03	F 4	MOLI	BCC	MUL2	TF CARRY CLEAR, SKIP PARTIAL PROD
F49A:	20	25	F4		JSR	ADD	ADD MULTIPLICAND TO PRODUCT.
F49D:	88			MUL2	DEY		NEXT MUL ITERATION.
F49E:	10 16	F5 F3		MDEND	BPL	MULI	LOOP UNTIL DONE. TEST SIGN LSB
F4A2:	90	BF		NORMX	BCC	NORM	IF EVEN,NORMALIZE PROD,ELSE COMP
F4A4:	38			FCOMPL	SEC		SET CARRY FOR SUBTRACT.
F4A5:	A2	03		CONDE 1	LDX	#\$3	INDEX FOR 3 BYTE SUBTRACT.
F4A/: F4A9:	А9 F5	00 F8		COMPLI	SBC	#\$0 X1.X	SUBTRACT BYTE OF EXPL.
F4AB:	95	F8			STA	xl,x	RESTORE IT.
F4AD:	CA	_			DEX		NEXT MORE SIGNIFICANT BYTE.
F4AE:	D0 F0	F7			BNE	COMPLI	LOOP UNTIL DONE.
F4B0:	20	32	F4	FDTV	JSR	MD1	TAKE ABS VAL OF MANTL. MANT2.
F4B5:	E5	F 8			SBC	Xl	SUBTRACT EXPl FROM EXP2.
F4B7:	20	E2	F4		JSR	MD2	SAVE AS QUOTIENT EXP.
F4BA:	38	02		DIVI	SEC	#¢2	SET CARRY FOR SUBTRACT.
F4BD:	B5	F5		DIV2	LDA	M2,X	INDEX FOR 5-BITE DOBINACTION.
F4BF:	F5	$\mathbf{FC}$			SBC	E,X	SUBTRACT A BYTE OF E FROM MANT2.
F4C1:	48				PHA		SAVE ON STACK.
F4C2:	10	F8			BPL	DIV2	LOOP UNTIL DONE.
F4C5:	A2	FD			LDX	#\$FD	INDEX FOR 3-BYTE CONDITIONAL MOVE
F4C7:	68			DIV3	PLA		PULL BYTE OF DIFFERENCE OFF STACK
F4C8:	90	02 F8			BCC	DIV4 M2+3 X	IF M2 <e don't="" m2.<="" restore="" td="" then=""></e>
F4CC:	E8	10		DIV4	INX	112 + 5 / 12	NEXT LESS SIGNIFICANT BYTE.
F4CD:	D0	F8			BNE	DIV3	LOOP UNTIL DONE.
F4CF:	26	FB EA			ROL	M1+2	DATI AHATENT LEET CADDY INTA ICD
F4D1:	26	F9			ROL	MITI	KOLL QUOTIENT LEFT, CARRI INTO LSB
F4D5:	06	F7			ASL	M2+2	
F4D7:	26	F6			ROL	M2+1 M2	SHIFT DIVIDEND LEFT
F4DB:	в0	1C			BCS	OVFL	OVFL IS DUE TO UNNORMED DIVISOR
F4DD:	88				DEY	_	NEXT DIVIDE ITERATION.
F4DE:	D0 F0	DA			BNE	DIV1 MDEND	LOOP UNTIL DONE 23 ITERATIONS.
F4E0:	86	FB		MD2	STX	MDEND Ml+2	NORM. GOOTIENT AND CORRECT SIGN.
F4E4:	86	FA			STX	Ml+1	CLEAR MANT1 (3 BYTES) FOR MUL/DIV.
F4E6:	86	F9			STX	Ml	
F4E8: F4EA:	во 30	01			BCS	MD3	IF NEG THEN NO UNDERFLOW.
F4EC:	68				PLA		POP ONE RETURN LEVEL.
F4ED:	68	-			PLA		
F4EE:	90 49	80 80		мпз	EOR	NORMX #\$80	CLEAR XI AND RETURN. COMPLEMENT SIGN BIT OF EXPONENT
F4F2:	85	F8		1120	STA	xl	STORE IT.
F4F4:	A0	17			LDY	#\$17	COUNT 24 MUL/23 DIV ITERATIONS.
F4F6:	60	<b>5</b> 7		OVCHK	RTS	MD 3	RETURN.
F4F9:	4C	F5	03	OVEIK	JMP	OVLOC	IT FOSTIVE EXF THEN NO OVEL.
					ORG	\$F63D	
F63D:	20	7D	F4	FIXL	JSR	RTAR	
F642:	10	13		LIX	BPL	UNDFL	
F644:	C9	8E			CMP	#\$8E	
F646:	D0	F5			BNE	FIX1	
F648:	∠4 10	Г.Э Г.Э			BIT RPT.	MT Flxbuc	
F64C:	A5	FB			LDA	Ml+2	
F64E:	F0	06			BEQ	FIXRTS	
F650:	E6	FA			INC	MI+1 EIVDmc	
F654:	D0 Е6	02 F9			BNE INC	Ml	
F656:	60	-		FIXRTS	RTS		
F657:	A9	00		UNDFL	LDA	#\$0	
r059: F65B:	б5 85	г9 FA			STA STA	Ml+l	
F65D:	60				RTS		

APPLE-II PSEUDO \* MACHINE INTERPRETER COPYRIGHT 1977 \* APPLE COMPUTER INC \* ALL RIGHTS RESERVED S. WOZNIAK \*\*\*\*\*\* TITLE "SWEET16 INTERPRETER" ROL EPZ \$0 EPZ R0H \$1 R14H EP7 \$1D R15L EPZ \$lE R15H EPZ \$1F S16PAG EQU \$F7 SAVE EÕU \$FF4A RESTORE EQU \$FF3F ORG \$F689 PRESERVE 6502 REG CONTENTS F689: 20 4A FF SW16 JSR SAVE F68C: 68 PLA F68D: 85 1E STA R15L INIT SWEET16 PC FROM RETURN F68F: 68 PT-A F690: 85 lF STA R15H ADDRESS INTERPRET AND EXECUTE F692: 20 98 F6 SW16B JSR SW16C F695: 4C 92 F6 JMP SW16B ONE SWEET16 INSTR. F698: E6 lE SW16C INC R15L F69A: D0 02 SW16D INCR SWEET16 PC FOR FETCH BNE F69C: E6 1F INC R15H F69E: A9 F7 SW16D LDA #SW16PAG F6A0: 48 PHA PUSH ON STACK FOR RTS F6A1: A0 00 LDY #\$0 F6A3: B1 1E LDA (R15L),Y FETCH INSTR F6A5: 29 OF AND MASK REG SPECIFICATION #\$F F6A7: 0A ASL A DOUBLE FOR TWO BYTE REGISTERS F6A8: AA TAX TO X REG FOR INDEXING F6A9: 4A LSR Α F6AA: 51 1E EOR (R15L), Y NOW HAVE OPCODE IF ZERO THEN NON-REG OP F6AC: F0 OB BEQ TOBR F6AE: 86 1D INDICATE'PRIOR RESULT REG' STX R14H F6B0: 4A LSR Α F6Bl: 4A LSR А OPCODE\*2 TO LSB'S F6B2: 4A LSR А F6B3: A8 TAY TO Y REG FOR INDEXING OPTBL-2,Y LOW ORDER ADR BYTE F6B4: B9 E1 F6 LDA F6B7: 48 ONTO STACK PHA F6B8: 60 RTS GOTO REG-OP ROUTINE F6B9: E6 1E TOBR INC R15L TOBR2 F6BB: D0 02 BNE INCR PC F6BD: E6 lF R15H INC F6BF: BD E4 F6 TOBR2 LDA BRTBL,X LOW ORDER ADR BYTE F6C2: 48 PHA ONTO STACK FOR NON-REG OP F6C3: A5 1D LDA Rl4H 'PRIOR RESULT REG' INDEX F6C5: 4A PREPARE CARRY FOR BC, BNC. LSR А F6C6: 60 RTS GOTO NON-REG OP ROUTINE F6C7: 68 POP RETURN ADDRESS RTNZ PLA F6C8: 68 PLA F6C9: 20 3F FF JSR RESTORE RESTORE 6502 REG CONTENTS F6CC: 6C 1E 00 JMP (R15L) RETURN TO 6502 CODE VIA PC F6CF: Bl lE SETZ (R15L), Y HIGH-ORDER BYTE OF CONSTANT LDA

10010	95	01			STA	ROH,X	
F6D3:	88	1 17			DEY	(DIET) V	LON ODDED DYME OF CONCMAND
F6D4:	ьт 95	0.0			STA	ROL X	LOW-ORDER BILE OF CONSIANT
F6D8:	98				TYA		Y-REG CONTAINS 1
F6D9:	38				SEC		
F6DA:	65 85	1E 1F			ADC	RI5L Pl5L	ADD 2 TO PC
F6DE:	90	02			BCC	SET2	
F6E0:	E6	lF			INC	R15H	
F6E2:	60			SET2	RTS		
F6E3:	02			OPTBL	DFB	SET-1	(1X)
F6E5:	гэ 04			DRIDL	DFB	LD-1	(0) (2X)
F6E6:	9D				DFB	BR-1	(1)
F6E7:	0 D				DFB	ST-1	(3X)
F6E8:	9E				DFB	BNC-1	(2)
F6EA·	20 25				DFB	LDAT-I	(4X)
F6EB:	16				DFB	STAT-1	(5X)
F6EC:	В2				DFB	BP-1	(4)
F6ED:	47				DFB	LDDAT-1	(6X)
F6EE: F6EF·	В9 51				DFB	BM-I STDAT-1	(5) (7X)
F6F0:	C0				DFB	BZ-1	(6)
F6Fl:	2F				DFB	POP-1	(8X)
F6F2:	C9				DFB	BNZ-1	(7)
F6F3:	5B				DFB	STPAT-1	(9X)
F6F5:	85				DFB	ADD-1	(8) (AX)
F6F6:	DD				DFB	BNM1-1	(9)
F6F7:	6E				DFB	SUB-1	(BX)
F6F8:	05				DFB	BK-1	
F6FA:	55 E8				DFB	RS-1	(CX) (B)
F6FB:	70				DFB	CPR-1	(DX)
F6FC:	93				DFB	BS-1	(C)
F6FD:	lE				DFB	INR-1	(EX)
FOFE: FOFF:	些 / 6 5				DFB	NUL-1 DCR-1	(D) (FX)
F700:	E7				DFB	NUL-1	(E)
F701:	Ε7				DFB	NUL-1	(UNUSED)
F702:	E7	~ -		~~~	DFB	NUL-1	(F)
F703:	10 P5	CA		SET	BPL	SETZ BOT Y	ALWAYS TAKEN
1705.	ЪЭ	00		BK	EOU	*-1	
F707:	85	00			STA	ROL	
F709:	B5	01			גמד	ROH X	MOVE DY MO DO
					LDA		MOVE RX TO RU
F70B:	85	01			STA	ROH	MOVE KX TO RU
F70B: F70D: F70E:	85 60 A5	01		ST	STA RTS LDA	ROH ROL	MOVE KA TO RU
F70B: F70D: F70E: F710:	85 60 A5 95	01 00 00		ST	STA RTS LDA STA	ROL,X	MOVE R0 TO RX
F70B: F70D: F70E: F710: F712:	85 60 A5 95 A5	01 00 00 01		ST	STA RTS LDA STA LDA	ROH ROL ROL,X ROH	MOVE R0 TO RX
F70B: F70D: F70E: F710: F712: F714:	85 60 A5 95 A5 95	01 00 00 01 01		ST	STA RTS LDA STA LDA STA STA	ROH ROL ROL,X ROH ROH,X	MOVE RA TO RU MOVE RO TO RX
F70B: F70D: F70E: F710: F712: F714: F716: F717:	85 60 A5 95 A5 95 60 A5	01 00 00 01 01 00		ST	STA RTS LDA STA LDA STA RTS LDA	ROH ROL ROL,X ROH ROH,X ROL	MOVE RA TO RU MOVE RO TO RX
F70B: F70D: F70E: F710: F712: F714: F716: F717: F719:	85 60 A5 95 A5 95 60 A5 81	01 00 00 01 01 01 00 00		ST STAT STAT2	STA RTS LDA STA LDA STA RTS LDA STA	ROH ROL ROL,X ROH ROH,X ROL (ROL,X)	MOVE R0 TO RX STORE BYTE INDIRECT
F70B: F70D: F70E: F710: F712: F714: F716: F717: F719: F71B:	85 60 A5 95 A5 95 60 A5 81 A0	01 00 01 01 01 00 00 00		ST STAT STAT2	STA RTS LDA STA LDA STA RTS LDA STA LDY	ROH ROL ROL,X ROH ROH,X ROL (ROL,X) #\$0	MOVE RA TO RU MOVE RO TO RX STORE BYTE INDIRECT
F70B: F70D: F70E: F710: F712: F714: F716: F717: F719: F71B: F71D: F71D:	85 60 A5 95 A5 95 60 A5 81 A0 84	01 00 00 01 01 01 00 00 00 1D		ST STAT STAT2 STAT3	STA RTS LDA STA LDA STA RTS LDA STA LDY STY LNC	ROH ROL ROL,X ROH ROH,X ROL (ROL,X) #\$0 Rl4H POT Y	MOVE RX TO RU MOVE R0 TO RX STORE BYTE INDIRECT INDICATE R0 IS RESULT NEG
F70B: F70D: F70E: F712: F714: F716: F717: F719: F71B: F71D: F71D: F71F: F721:	85 60 A5 95 A5 95 60 A5 81 A0 84 F6 D0	01 00 01 01 01 01 00 00 00 00 1D 00 02		ST STAT STAT2 STAT3 INR	STA RTS LDA STA LDA STA RTS LDA STA LDY STY INC BNE	ROH ROL ROL,X ROH ROH,X ROL (ROL,X) #\$0 R14H ROL,X INR2	MOVE RX TO RU MOVE R0 TO RX STORE BYTE INDIRECT INDICATE R0 IS RESULT NEG INCR RX
F70B: F70D: F70E: F710: F712: F714: F716: F717: F719: F719: F71B: F71D: F71D: F71F: F721: F723:	85 60 A5 95 60 A5 95 60 A5 81 A0 84 F6 D0 F6	01 00 01 01 01 00 00 00 00 1D 00 02 01		ST STAT STAT2 STAT3 INR	STA RTS LDA STA LDA STA RTS LDA STA LDY STY INC BNE INC	ROH ROL ROL,X ROH ROH,X ROL (ROL,X) #\$0 R14H ROL,X INR2 ROH,X	MOVE RX TO RU MOVE R0 TO RX STORE BYTE INDIRECT INDICATE R0 IS RESULT NEG INCR RX
F70B: F70D: F710: F712: F714: F714: F716: F717: F719: F719: F7110: F7110: F7121: F7211: F7221: F7225:	85 60 95 95 60 85 95 60 85 81 84 F6 D0 F60	01 00 00 01 01 00 00 00 00 1D 00 02 01		ST STAT STAT2 STAT3 INR INR2	STA RTS LDA STA LDA STA RTS LDA STA RTS LDA STA STA LDY STY INC RTS	ROH ROL ROL,X ROH ROH,X ROL (ROL,X) #\$0 R14H ROL,X INR2 ROH,X	MOVE RX TO RU MOVE R0 TO RX STORE BYTE INDIRECT INDICATE R0 IS RESULT NEG INCR RX
F70B: F70D: F70E: F71C: F712: F714: F716: F717: F71B: F71B: F71B: F71D: F71F: F721: F722: F725: F726:	85 60 95 85 95 85 85 85 85 85 85 85 81 84 84 56 84 56 84 84 84 84 84 84 85 84 84 85 85 85 85 85 85 85 85 85 85 85 85 85	01 00 00 01 01 01 00 00 00 1D 00 02 01		ST STAT STAT2 STAT3 INR INR2 LDAT	STA RTS LDA STA LDA STA LDA STA LDA STA LDY STY INC RTS LDA	ROH ROL ROL,X ROH ROH,X ROL (ROL,X) #\$0 R14H ROL,X INR2 ROH,X (ROL,X) POL	MOVE RX TO RU MOVE R0 TO RX STORE BYTE INDIRECT INDICATE R0 IS RESULT NEG INCR RX LOAD INDIRECT (RX)
F70B: F70D: F70E: F71C: F712: F714: F716: F717: F71B: F71B: F71B: F71D: F71E: F721: F722: F722: F726: F728: F728:	85 60 95 85 95 85 80 85 80 84 F6 80 85 80	01 00 00 01 01 01 00 00 00 00 00 00 00 0		ST STAT STAT2 STAT3 INR INR2 LDAT	STA STA STA LDA STA LDA STA LDA STA LDA STA LDY STY INC RTS LDA STA LDY	ROH ROL ROL,X ROH ROH,X ROL (ROL,X) #\$0 R14H ROL,X INR2 ROH,X (ROL,X) ROL,X ROL,X ROL ROL,X ROL ROL ROL ROH ROH ROH ROH ROH ROH ROH ROH	MOVE RX TO RU MOVE R0 TO RX STORE BYTE INDIRECT INDICATE R0 IS RESULT NEG INCR RX LOAD INDIRECT (RX) TO R0
F70B: F70D: F70E: F71C: F712: F714: F716: F717: F71B: F71B: F71B: F71D: F71F: F721: F722: F722: F726: F728: F722: F722:	850 850 850 850 810 840 840 850 850 840 850 840 850 840 850 840 850 850 840 850 850 840 850 840 850 840 850 850 840 850 840 850 850 840 850 840 850 850 850 850 850 850 850 85	01 00 00 01 01 01 00 00 00 00 00 00 00 0		ST STAT STAT2 STAT3 INR INR2 LDAT	STA STA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDY STY	ROH ROL ROL,X ROH ROH,X ROL (ROL,X) #\$0 R14H ROL,X INR2 ROH,X (ROL,X) ROL,X ROL,X ROL #\$0 ROL,X ROH ROH,X ROH ROH,X ROH ROH,X ROH ROH,X ROH ROH,X ROH ROH,X ROH ROH,X ROH ROH,X ROH ROH,X ROH ROH,X ROH ROH,X ROH ROH,X ROH ROH,X ROH ROH,X ROH ROL,X ROH ROH,X ROL ROL,X ROH ROL,X ROH ROL,X ROL ROL,X ROL ROL,X ROL ROL ROL,X ROL ROL,X ROL ROL,X ROL ROL,X ROL ROL,X ROL ROL,X ROL ROL,X ROL ROL,X ROL ROL ROL,X ROL ROL,X ROL ROL,X ROL ROL,X ROL ROL,X ROL ROL,X ROL ROL,X ROL ROL,X ROL ROL ROL ROL ROL ROL ROL ROL	MOVE RX TO RU MOVE R0 TO RX STORE BYTE INDIRECT INDICATE R0 IS RESULT NEG INCR RX LOAD INDIRECT (RX) TO R0 ZERO HIGH-ORDER R0 BYTE
F70B: F70D: F70E: F712: F712: F714: F716: F717: F719: F71B: F71D: F71B: F71D: F721: F722: F722: F722: F722: F722: F722:	85 60 95 95 60 81 84 F6 00 85 84 85 84 F0	01 00 00 01 01 00 00 00 00 00 00 00 00 0		ST STAT STAT2 STAT3 INR INR2 LDAT	STA STA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA STA STY BEQ	ROH ROL ROL,X ROH ROH,X ROL (ROL,X) #\$0 R14H ROL,X INR2 ROH,X (ROL,X) ROL,X ROH,X (ROL,X) ROL ROH STAT3	MOVE RX TO RU MOVE R0 TO RX STORE BYTE INDIRECT INDICATE R0 IS RESULT NEG INCR RX LOAD INDIRECT (RX) TO R0 ZERO HIGH-ORDER R0 BYTE ALWAYS TAKEN
F70B: F70D: F70C: F712: F712: F714: F716: F717: F719: F71B: F71D: F71B: F71D: F721: F722: F72: F7	850 A55 A55 A55 A50 A51 A50 A50 A50 A50 A50 A50 A50 A50	01 00 00 01 01 00 00 00 00 00 00 00 00 0		ST STAT STAT2 STAT3 INR INR2 LDAT	STA STA STA LDA STA LDA STA LDA STA LDA STA LDA STA STA STA STA STA STA STA STA STA ST	ROH ROL ROL,X ROH,X ROL (ROL,X) #\$0 R14H ROL,X INR2 ROH,X (ROL,X) ROL,X (ROL,X) ROL ROH,X (ROL,X) ROH STAT3 #\$0 ROH STAT3 #\$0 ROH STAT3	MOVE RX TO RU MOVE R0 TO RX STORE BYTE INDIRECT INDICATE R0 IS RESULT NEG INCR RX LOAD INDIRECT (RX) TO R0 ZERO HIGH-ORDER R0 BYTE ALWAYS TAKEN HIGH ORDER BYTE = 0
F70B: F70D: F70C: F712: F712: F714: F714: F716: F717: F719: F718: F717: F721: F722: F722: F722: F722: F722: F722: F722: F732: F732:	850 A55 A55 A50 A51 A50 A50 A50 A50 A50 A50 A50 A50	01 00 00 01 01 00 00 00 00 00 00 00 00 0	F7	ST STAT STAT2 STAT3 INR INR2 LDAT POP POPD	STA STA STA LDA STA LDA STA LDA STA LDA STA LDA STA STA STA STA STA STA STA STA STA ST	ROH ROL ROL,X ROH,X ROH,X ROL (ROL,X) #\$0 R14H ROL,X INR2 ROH,X (ROL,X) ROL X INR2 ROH,X (ROL,X) #\$0 ROH STAT3 #\$0 POP2 DCR	MOVE RX TO RU MOVE R0 TO RX STORE BYTE INDIRECT INDICATE R0 IS RESULT NEG INCR RX LOAD INDIRECT (RX) TO R0 ZERO HIGH-ORDER R0 BYTE ALWAYS TAKEN HIGH ORDER BYTE = 0 ALWAYS TAKEN DECR RX
F70B: F70D: F70C: F712: F712: F714: F714: F717: F719: F717: F719: F718: F721: F722: F722: F722: F722: F722: F722: F722: F730: F732: F737:	850 A55 A55 A55 A50 A51 A50 A50 A50 A50 A50 A50 A50 A50	01 00 00 01 01 00 00 00 00 00 00 00 00 0	F7	ST STAT STAT2 STAT3 INR INR2 LDAT POP POPD	STA STA STA STA STA STA STA LDA STA LDA STA LDA STA STA STA STA STA STA STA STA STA ST	ROH ROL ROL,X ROH,X ROH,X ROL (ROL,X) #\$0 R14H ROL,X INR2 ROH,X (ROL,X) ROL #\$0 ROH STAT3 #\$0 POP2 DCR (ROL,X)	MOVE RX TO RU MOVE R0 TO RX STORE BYTE INDIRECT INDICATE R0 IS RESULT NEG INCR RX LOAD INDIRECT (RX) TO R0 ZERO HIGH-ORDER R0 BYTE ALWAYS TAKEN HIGH ORDER BYTE = 0 ALWAYS TAKEN DECR RX POP HIGH ORDER BYTE @RX
F70B: F70D: F70C: F712: F712: F714: F714: F717: F719: F717: F719: F717: F721: F722: F722: F722: F722: F722: F722: F722: F730: F732: F739: F739:	850 A55 A55 A55 A50 A51 A50 A50 A50 A50 A50 A50 A50 A50 A50 A50	01 00 00 01 01 00 00 00 00 00 00 00 00 0	F7	ST STAT STAT2 STAT3 INR INR2 LDAT POP POPD	STA STA STA STA STA STA STA LDA STA LDA STA LDA STA STA STA STA STA STA STA STA STA ST	ROH ROL ROL,X ROH,X ROH,X ROL (ROL,X) #\$0 R14H ROL,X INR2 ROH,X (ROL,X) ROL #\$0 ROH STAT3 #\$0 POP2 DCR (ROL,X)	MOVE RX TO RU MOVE R0 TO RX STORE BYTE INDIRECT INDICATE R0 IS RESULT NEG INCR RX LOAD INDIRECT (RX) TO R0 ZERO HIGH-ORDER R0 BYTE ALWAYS TAKEN HIGH ORDER BYTE = 0 ALWAYS TAKEN DECR RX POP HIGH ORDER BYTE @RX SAVE IN Y-REG
F70B: F70D: F70C: F712: F712: F714: F714: F717: F719: F717: F719: F717: F721: F722: F722: F722: F722: F722: F722: F722: F730: F732: F739: F739: F739:	850 A55 A55 A55 A55 A55 A55 A55 A	01 00 00 01 01 00 00 00 00 00 00 00 00 0	F7 F7	ST STAT STAT2 STAT3 INR INR2 LDAT POP POPD POP2	STA STA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA STA LDA STA STA LDA STA STA LDA STA LDA STA LDA STA LDA STA STA LDA STA STA LDA STA STA STA LDA STA STA LDA STA STA LDA STA STA LDA STA STA LDA STA STA LDA STA STA LDA STA STA LDA STA STA LDA STA STA LDA STA STA STA LDA STA STA STA STA STA STA STA STA STA ST	ROH ROL ROL, X ROH, X ROH, X ROL (ROL, X) #\$0 R14H ROL, X INR2 ROH, X (ROL, X) ROL #\$0 ROH STAT3 #\$0 POP2 DCR (ROL, X) DCR (POL X)	MOVE RX TO RU MOVE R0 TO RX STORE BYTE INDIRECT INDICATE R0 IS RESULT NEG INCR RX LOAD INDIRECT (RX) TO R0 ZERO HIGH-ORDER R0 BYTE ALWAYS TAKEN HIGH ORDER BYTE = 0 ALWAYS TAKEN DECR RX POP HIGH ORDER BYTE @RX SAVE IN Y-REG DECR RX LOW-ORDEP BYTE
F70B: F70D: F70C: F712: F712: F714: F714: F717: F719: F715: F715: F725: F726: F726: F726: F726: F726: F726: F726: F727: F730: F737: F737: F739: F737: F737: F737: F737: F737:	850 850 850 850 850 850 850 850	01 00 00 01 01 00 00 00 00 00 00 00 00 0	F7 F7	ST STAT STAT2 STAT3 INR INR2 LDAT POP POPD POP2	STA STA STA LDA STA STA STA LDA STA STA STA LDA STA STA STA STA STA STA STA STA STA ST	ROH ROL ROL, X ROH, X ROH, X ROL (ROL, X) #\$0 R14H ROL, X INR2 ROH, X (ROL, X) ROL #\$0 ROH STAT3 #\$0 POP2 DCR (ROL, X) DCR (ROL, X) ROL	MOVE RX TO RU MOVE R0 TO RX STORE BYTE INDIRECT INDICATE R0 IS RESULT NEG INCR RX LOAD INDIRECT (RX) TO R0 ZERO HIGH-ORDER R0 BYTE ALWAYS TAKEN HIGH ORDER BYTE = 0 ALWAYS TAKEN DECR RX POP HIGH ORDER BYTE @RX SAVE IN Y-REG DECR RX LOW-ORDER BYTE TO R0
F70B: F70D: F70C: F712: F712: F714: F716: F717: F719: F71B: F71B: F71D: F721: F722: F722: F722: F722: F722: F722: F722: F730: F732: F739: F739: F739: F732:	850 455 455 455 455 455 455 455 4	01 00 00 01 01 00 00 00 00 00 00 00 00 0	F7 F7	ST STAT STAT2 STAT3 INR INR2 LDAT POP POPD POP2	STA STA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA STA LDA STA STA LDA STA STA STA STA STA STA STA STA STA ST	ROH ROL ROL, X ROH, X ROL (ROL, X) #\$0 R14H ROL, X INR2 ROH, X (ROL, X) ROL #\$0 ROH STAT3 #\$0 POP2 DCR (ROL, X) DCR (ROL, X) ROL ROH	MOVE RX TO RU MOVE R0 TO RX STORE BYTE INDIRECT INDICATE R0 IS RESULT NEG INCR RX LOAD INDIRECT (RX) TO R0 ZERO HIGH-ORDER R0 BYTE ALWAYS TAKEN HIGH ORDER BYTE = 0 ALWAYS TAKEN DECR RX POP HIGH ORDER BYTE @RX SAVE IN Y-REG DECR RX LOW-ORDER BYTE TO R0
F70B: F70D: F70C: F712: F712: F714: F716: F717: F719: F71B: F71D: F71D: F721: F722: F722: F722: F722: F722: F722: F722: F730: F732: F739: F739: F739: F734: F737: F739: F734: F737: F734:	860A5555560A804600 860A555555555555555555555555555555555555	01 00 00 01 01 00 00 00 00 00 00 00 00 0	F7 F7	ST STAT STAT2 STAT3 INR INR2 LDAT POP POPD POP2 POP3	STA STA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA STA LDA STA STA STA STA STA STA STA STA STA ST	ROH ROL ROL, X ROH, X ROL (ROL, X) #\$0 R14H ROL, X) #\$0 R14H ROL, X) ROL, X) ROL #\$0 ROH, X (ROL, X) ROL #\$0 ROH STAT3 #\$0 POP2 DCR (ROL, X) DCR (ROL, X) ROL ROH #\$0 ROH ROH ROH ROH ROH ROH ROH ROH	MOVE RX TO RU MOVE R0 TO RX STORE BYTE INDIRECT INDICATE R0 IS RESULT NEG INCR RX LOAD INDIRECT (RX) TO R0 ZERO HIGH-ORDER R0 BYTE ALWAYS TAKEN HIGH ORDER BYTE = 0 ALWAYS TAKEN DECR RX POP HIGH ORDER BYTE @RX SAVE IN Y-REG DECR RX LOW-ORDER BYTE TO R0 INDICATE R0 AS LAST RESULT REG
F70B: F70D: F70C: F712: F714: F714: F714: F717: F719: F715: F715: F725: F726: F726: F726: F726: F726: F726: F726: F727: F730: F737: F739: F739: F737: F739: F737: F739: F737: F739: F734: F737: F734: F737: F734: F737: F734: F737:	860A555556A804860166018504860000000000000000000000000000000000	0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	F7 F7	ST STAT STAT2 STAT3 INR INR2 LDAT POP POPD POP2 POP3	LDA STAS STAS STAS STAS STAS STAS STAS ST	ROH ROL ROL, X ROH, X ROL (ROL, X) #\$0 R14H ROL, X) #\$0 R14H ROL, X) ROL, X) ROL #\$0 ROH, X (ROL, X) ROL #\$0 ROH STAT3 #\$0 POP2 DCR (ROL, X) ROL ROL X) DCR (ROL, X) ROL ROH #\$0 ROL ROH #\$0 ROL ROH ROH ROH ROH ROH ROH ROH ROH	MOVE RX TO RU MOVE R0 TO RX STORE BYTE INDIRECT INDICATE R0 IS RESULT NEG INCR RX LOAD INDIRECT (RX) TO R0 ZERO HIGH-ORDER R0 BYTE ALWAYS TAKEN HIGH ORDER BYTE = 0 ALWAYS TAKEN DECR RX POP HIGH ORDER BYTE @RX SAVE IN Y-REG DECR RX LOW-ORDER BYTE TO R0 INDICATE R0 AS LAST RESULT REG
F70B: F70D: F70C: F71C: F712: F714: F714: F717: F715: F715: F725: F726: F726: F726: F726: F726: F727: F727: F730: F737: F739: F737: F739: F737: F739: F737: F739: F737: F739: F734: F737: F747:	860 860 860 855 860 850 860 860 860 860 860 860 860 86	0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	F7 F7	ST STAT STAT2 STAT3 INR INR2 LDAT POP POPD POP2 POP3 LDDAT	LDA STAS STAS STAS STAS STAS STAS STAS ST	ROH ROL ROL, X ROH, X ROL (ROL, X) #\$0 R14H ROL, X) #\$0 R14H ROL, X) ROL, X) ROL #\$0 ROH, X (ROL, X) ROL #\$0 ROH STAT3 #\$0 POP2 DCR (ROL, X) POP2 DCR (ROL, X) ROL ROH #\$0 ROH #\$0 ROH #\$0 ROH #\$0 ROH ROH ROH ROH ROH ROH ROH ROH	MOVE RX TO RU MOVE R0 TO RX STORE BYTE INDIRECT INDICATE R0 IS RESULT NEG INCR RX LOAD INDIRECT (RX) TO R0 ZERO HIGH-ORDER R0 BYTE ALWAYS TAKEN HIGH ORDER BYTE = 0 ALWAYS TAKEN DECR RX POP HIGH ORDER BYTE @RX SAVE IN Y-REG DECR RX LOW-ORDER BYTE TO R0. INCR RX
F70B: F70D: F70D: F710: F712: F714: F714: F715: F715: F715: F725: F726: F726: F726: F726: F726: F726: F726: F727: F730: F737: F739: F739: F739: F739: F737: F739: F739: F739: F734: F745: F745: F748: F748: F748: F748:	860 860 850 850 850 850 850 850 850 85	0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	F7 F7	ST STAT STAT2 STAT3 INR INR2 LDAT POP POPD POP2 POP3 LDDAT	LDA STAS LDA STTS LDA STAS LA STAS LDA STAS LDA STAS LDA STAS LDA STAS LDA STAS LDA STAS LDA	ROH ROL ROL, X ROH, X ROL (ROL, X) #\$0 R14H ROL, X) #\$0 R14H ROL, X) ROL, X) ROL #\$0 ROH, X (ROL, X) ROH STAT3 #\$0 POP2 DCR (ROL, X) POP2 DCR (ROL, X) ROL ROH #\$0 ROH #\$0 ROH ROH ROH ROH ROH ROH ROH ROH	MOVE RX TO RU MOVE R0 TO RX STORE BYTE INDIRECT INDICATE R0 IS RESULT NEG INCR RX LOAD INDIRECT (RX) TO R0 ZERO HIGH-ORDER R0 BYTE ALWAYS TAKEN HIGH ORDER BYTE = 0 ALWAYS TAKEN DECR RX POP HIGH ORDER BYTE = 0 ALWAYS TAKEN DECR RX POP HIGH ORDER BYTE = 0 ALWAYS TAKEN DECR RX LOW-ORDER BYTE TO R0 INDICATE R0 AS LAST RESULT REG LOW-ORDER BYTE TO R0, INCR RX HIGH-ORDER BYTE TO R0
F70B: F70D: F70D: F710: F712: F714: F714: F715: F715: F715: F715: F725: F726: F726: F726: F726: F726: F727: F726: F727: F727: F727: F727: F730: F737: F737: F739: F737: F739: F737:	860A5556A8048F0F601850485048540486001854084600185408460018540840000000000000000000000000000000	0         1           0         0           0         0           0         1           0         0           0         1           0         0	F7 F7	ST STAT STAT2 STAT3 INR INR2 LDAT POP POPD POP2 POP3 LDDAT	LDA STA STA LDA STA STA STA STA STA STA STA STA STA ST	ROH ROL ROL, X ROH, X ROL (ROL, X) #\$0 R14H ROL, X) #\$0 R14H ROL, X) ROL, X) ROL #\$0 ROH, X (ROL, X) ROH STAT3 #\$0 POP2 DCR (ROL, X) ROL ROL ROL ROH #\$0 ROL ROL X) ROL ROH ROH ROH ROH ROH ROH ROH ROH	MOVE RX TO RU MOVE R0 TO RX STORE BYTE INDIRECT INDICATE R0 IS RESULT NEG INCR RX LOAD INDIRECT (RX) TO R0 ZERO HIGH-ORDER R0 BYTE ALWAYS TAKEN HIGH ORDER BYTE = 0 ALWAYS TAKEN DECR RX POP HIGH ORDER BYTE = 0 ALWAYS TAKEN DECR RX POP HIGH ORDER BYTE = 0 ALWAYS TAKEN DECR RX POP HIGH ORDER BYTE = 0 ALWAYS TAKEN DECR RX LOW-ORDER BYTE TO R0 INDICATE R0 AS LAST RESULT REG LOW-ORDER BYTE TO R0, INCR RX HIGH-ORDER BYTE TO R0
F700B: F700E: F700E: F7102: F7122: F714: F714: F714: F717: F717: F717: F717: F717: F717: F717: F717: F722: F722: F722: F722: F722: F722: F722: F722: F722: F722: F722: F722: F722: F722: F722: F723: F723: F734: F737: F741: F747: F	860 860 860 850 860 860 860 860 860 860 860 86	0 1 00 00 00 00 00 00 00 00 00 00 00 00	F7 F7 F7 F7	ST STAT STAT2 STAT3 INR INR2 LDAT POP POPD POP2 POP3 LDDAT	LDA SRTS SRTS SRTS SRTS SRTS SRTS SRTS SRT	ROH ROL ROL, X ROH, X ROH, X ROL (ROL, X) #\$0 R14H ROL, X INR2 ROH, X (ROL, X) ROL #\$0 ROH STAT3 #\$0 POP2 DCR (ROL, X) ROL ROL ROH #\$0 ROL X) POP2 DCR (ROL, X) ROL ROL X) ROL ROH #\$0 ROL X) ROL ROL X) ROL ROL X) ROL ROL X) ROL	MOVE RX TO RU MOVE R0 TO RX STORE BYTE INDIRECT INDICATE R0 IS RESULT NEG INCR RX LOAD INDIRECT (RX) TO R0 ZERO HIGH-ORDER R0 BYTE ALWAYS TAKEN HIGH ORDER BYTE = 0 ALWAYS TAKEN DECR RX POP HIGH ORDER BYTE = 0 ALWAYS TAKEN DECR RX POP HIGH ORDER BYTE = 0 ALWAYS TAKEN DECR RX POP HIGH ORDER BYTE = 0 ALWAYS TAKEN DECR RX LOW-ORDER BYTE TO R0 INDICATE R0 AS LAST RESULT REG LOW-ORDER BYTE TO R0, INCR RX HIGH-ORDER BYTE TO R0

F755:	Α5	01			LDA	ROH	BYTE AND INCR RX. THEN
F757:	81	00	_		STA	(ROL,X)	STORE HIGH-ORDER BYTE.
F759:	4C	1F	F7		JMP	INR	INCR RX AND RETURN
F75C:	20	66	F7	STPAT	JSR	DCR	DECR RX
F/5F:	A5	00				RUL	CHODE DO LOW DUME ADV
F763.	0 I 1 C	13	F7		TMD	(RUL,A)	TNDTCATE DO AS LAST DELT DEC
F766:	85	00	1 /	DCR	T.DA	ROLX	INDICATE NO NO ENDI NOEI NEG
F768:	D0	02		DOI	BNE	DCR2	DECR RX
F76A:	D6	01			DEC	ROH,X	
F76C:	D6	00		DCR2	DEC	ROL,X	
F76E:	60				RTS	•	
F76F:	A0	00		SUB	LDY	#\$0	RESULT TO RO
F771:	38			CPR	SEC		NOTE Y-REG = $13 \times 2$ FOR CPR
F772:	Α5	00			LDA	ROL	
F774:	F5	00			SBC	ROL,X	
F776:	99	00	00		STA	ROL,Y	R0-RX TO RY
F779:	A5	10			LDA	ROH	
F//B:	F 5	01	0.0	CUD 2	SBC	ROH,X	
F700.	99	01	00	SUBZ	5TA TVA	ROH, Y	
F781.	69	0.0			ADC	#\$O	CAPPY TO LCR
F783:	85	10			STA	R14H	
F785:	60	10			RTS		
F786:	A5	00		ADD	LDA	ROL	
F788:	75	00			ADC	ROL,X	
F78A:	85	00			STA	ROL	R0+RX TO R0
F78C:	A5	01			LDA	ROH	
F78E:	75	01			ADC	ROH,X	
F790:	A0	00			LDY	#\$0	R0 FOR RESULT
F792:	FO	Ε9			BEQ	SUB2	FINISH ADD
F794:	A5	lE		BS	LDA	R15L	NOTE X-REG IS 12*2!
F796:	20	19	F7		JSR	STAT2	PUSH LOW PC BYTE VIA R12
F799:	A5	1F			LDA	RISH	
F/9B:	20	19	F. /	תת	JSR	STAT2	PUSH HIGH-ORDER PC BYTE
F79E:	TO BU	0 ፑ		BNC	BCC	BNC2	NO CAPPY TEST
F721.	BI	1E		BRI		(R15L) V	DISPLACEMENT BYTE
F7A3:	10	01		DRI	BPL	BR2	DIDI DACEMENT DITE
F7A5:	88	• -			DEY	2112	
F7A6:	65	lE		BR2	ADC	R15L	ADD TO PC
F7A8:	85	lE			STA	R15L	
F7AA:	98				TYA		
F7AB:	65	lF			ADC	R15H	
F7AD:	85	lF			STA	R15H	
F7AF:	60			BNC2	RTS		
F7B0:	в0	EC		BC	BCS	BR	
F7B2:	60				RTS		
F/B3:	0A			ВР	ASL	A	DOUBLE RESULT-REG INDEX
F7B4:	~ ~				TAX		$\Pi(A) = V = [D = D(A) + D = T = D = D = V = D = D = D = D = D = D = D$
	AA D5	01			TDA	DAU V	TO X REG FOR INDEXING
F7B7:	AA B5 10	01 E8			LDA BPL	ROH,X BRl	TO X REG FOR INDEXING TEST FOR PLUS BRANCH IF SO
F7B7:	AA B5 10 60	01 E8			LDA BPL RTS	ROH,X BRl	TO X REG FOR INDEXING TEST FOR PLUS BRANCH IF SO
F7B7: F7B9: F7BA:	AA B5 10 60 0A	01 E8		ВМ	LDA BPL RTS ASL	ROH,X BRl A	TO X REG FOR INDEXING TEST FOR PLUS BRANCH IF SO DOUBLE RESULT-REG INDEX
F7B7: F7B7: F7B9: F7BA: F7BB:	AA B5 10 60 0A AA	01 E8		ВМ	LDA BPL RTS ASL TAX	ROH,X BRl A	TO X REG FOR INDEXING TEST FOR PLUS BRANCH IF SO DOUBLE RESULT-REG INDEX
F7B3: F7B7: F7B9: F7BA: F7BB: F7BC:	AA B5 10 60 0A AA B5	01 E8 01		ВМ	LDA BPL RTS ASL TAX LDA	ROH,X BR1 A ROH,X	TO X REG FOR INDEXING TEST FOR PLUS BRANCH IF SO DOUBLE RESULT-REG INDEX TEST FOR MINUS
F7B7: F7B7: F7B9: F7BA: F7BB: F7BC: F7BE:	AA B5 10 60 0A AA B5 30	01 E8 01 E1		ВМ	LDA BPL RTS ASL TAX LDA BMI	ROH,X BR1 A ROH,X BR1	TO X REG FOR INDEXING TEST FOR PLUS BRANCH IF SO DOUBLE RESULT-REG INDEX TEST FOR MINUS
F7B7: F7B7: F7B9: F7BA: F7BB: F7BC: F7BE: F7C0:	AA B5 10 60 0A AA B5 30 60	01 E8 01 E1		ВМ	LDA BPL RTS ASL TAX LDA BMI RTS	ROH,X BR1 A ROH,X BR1	TO X REG FOR INDEXING TEST FOR PLUS BRANCH IF SO DOUBLE RESULT-REG INDEX TEST FOR MINUS
F7B3: F7B7: F7B9: F7BA: F7BB: F7BC: F7BC: F7BE: F7C0: F7C1:	AA B5 10 60 0A AA B5 30 60 0A	01 E8 01 E1		BM BZ	LDA BPL RTS ASL TAX LDA BMI RTS ASL	ROH, X BR1 A ROH, X BR1 A	TO X REG FOR INDEXING TEST FOR PLUS BRANCH IF SO DOUBLE RESULT-REG INDEX TEST FOR MINUS DOUBLE RESULT-REG INDEX
F7B5: F7B7: F7B9: F7BA: F7B8: F7BC: F7BC: F7C0: F7C1: F7C2:	AA B5 10 60 0A AA B5 30 60 0A AA	01 E8 01 E1		BM BZ	LDA BPL RTS ASL TAX LDA BMI RTS ASL TAX	ROH, X BR1 A ROH, X BR1 A	TO X REG FOR INDEXING TEST FOR PLUS BRANCH IF SO DOUBLE RESULT-REG INDEX TEST FOR MINUS DOUBLE RESULT-REG INDEX
F7B3: F7B7: F7B9: F7B8: F7B8: F7B6: F7B6: F7C0: F7C1: F7C2: F7C3:	AA B5 10 60 0A AA B5 30 60 0A AA B5	01 E8 01 E1		BM BZ	LDA BPL RTS ASL TAX LDA BMI RTS ASL TAX LDA	ROH, X BRI A ROH, X BRI A ROL, X	TO X REG FOR INDEXING TEST FOR PLUS BRANCH IF SO DOUBLE RESULT-REG INDEX TEST FOR MINUS DOUBLE RESULT-REG INDEX TEST FOR ZERO
F7B3: F7B9: F7BA: F7BB: F7BC: F7BC: F7BC: F7C0: F7C1: F7C2: F7C3: F7C5:	AA B5 10 60 0A AB5 30 60 0A AA B5 15	01 E8 01 E1 00 01		BM BZ	LDA BPL RTS ASL TAX LDA BMI RTS ASL TAX LDA ORA	ROH,X BRI A ROH,X BRI A ROL,X ROH,X	TO X REG FOR INDEXING TEST FOR PLUS BRANCH IF SO DOUBLE RESULT-REG INDEX TEST FOR MINUS DOUBLE RESULT-REG INDEX TEST FOR ZERO (BOTH BYTES)
F7B3: F7B9: F7BA: F7BB: F7BC: F7BC: F7BC: F7C0: F7C1: F7C2: F7C3: F7C5: F7C7:	AA B5 10 60 0A AB5 30 60 0A AB5 15 F0 60	01 E8 01 E1 00 01 D8		BM BZ	LDA BPL RTS ASL TAX LDA BMI RTS ASL TAX LDA ORA BEQ	ROH, X BR1 A ROH, X BR1 A ROL, X ROH, X BR1	TO X REG FOR INDEXING TEST FOR PLUS BRANCH IF SO DOUBLE RESULT-REG INDEX TEST FOR MINUS DOUBLE RESULT-REG INDEX TEST FOR ZERO (BOTH BYTES) BRANCH IF SO
F7B7: F7B7: F7B8: F7B8: F7B8: F7B2: F7C0: F7C1: F7C2: F7C3: F7C5: F7C7: F7C9:	AA B5 10 60 0A AB5 30 60 0A AB5 15 F0 60 00	01 E8 01 E1 00 01 D8		BM BZ	LDA BPL RTS ASL TAX LDA BMI RTS ASL TAX LDA ORA BEQ RTS ASL	ROH, X BR1 A ROH, X BR1 A ROL, X ROH, X BR1	TO X REG FOR INDEXING TEST FOR PLUS BRANCH IF SO DOUBLE RESULT-REG INDEX TEST FOR MINUS DOUBLE RESULT-REG INDEX TEST FOR ZERO (BOTH BYTES) BRANCH IF SO
F7B7: F7B7: F7B8: F7B8: F7B8: F7B8: F7B8: F7C0: F7C0: F7C1: F7C2: F7C3: F7C5: F7C7: F7C9: F7C8: F7C8:	AA B5 10 60 0A AA B5 30 60 0A AA B5 15 F0 0A 2A	01 E8 01 E1 00 01 D8		BM BZ BNZ	LDA BPL RTS ASL TAX LDA BMI RTS ASL TAX LDA ORA BEQ RTS ASL TAX	ROH, X BR1 A ROH, X BR1 A ROL, X ROH, X BR1 A	TO X REG FOR INDEXING TEST FOR PLUS BRANCH IF SO DOUBLE RESULT-REG INDEX TEST FOR MINUS DOUBLE RESULT-REG INDEX TEST FOR ZERO (BOTH BYTES) BRANCH IF SO DOUBLE RESULT-REG INDEX
F7B7: F7B7: F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F7C0: F7C0: F7C2: F7C2: F7C7: F7C7: F7C7: F7C7: F7C8: F7C8: F7C8:	AA B5 10 0A A5 30 0A A5 50 60 0A A5 50 60 0A A5 55 60 0A A5 55	01 E8 01 E1 00 01 D8		BM BZ BNZ	LDA BPL RTS ASL TAX LDA BMI RTS ASL TAX LDA ORA BEQ RTS ASL TAX LDA	ROH, X BR1 A ROH, X BR1 A ROL, X ROH, X BR1 A ROL, X	TO X REG FOR INDEXING TEST FOR PLUS BRANCH IF SO DOUBLE RESULT-REG INDEX TEST FOR MINUS DOUBLE RESULT-REG INDEX TEST FOR ZERO (BOTH BYTES) BRANCH IF SO DOUBLE RESULT-REG INDEX TEST FOR NON-ZERO
F7B7: F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F7C0: F7C0: F7C2: F7C2: F7C7: F7C7: F7C7: F7C8:	AA B5 10 0A A5 30 60 0A AA B5 15 F0 0A AA B5 15	01 E8 01 E1 00 01 D8 00 01		BM BZ BNZ	LDA BPL RTS ASL TAX LDA BMI RTS ASL LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA ORA	ROH, X BR1 A ROH, X BR1 A ROL, X ROH, X BR1 A ROL, X ROH, X	TO X REG FOR INDEXING TEST FOR PLUS BRANCH IF SO DOUBLE RESULT-REG INDEX TEST FOR MINUS DOUBLE RESULT-REG INDEX TEST FOR ZERO (BOTH BYTES) BRANCH IF SO DOUBLE RESULT-REG INDEX TEST FOR NON-ZERO (BOTH BYTES)
F7B7: F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F7C8: F7C2: F7C2: F7C7: F7C7: F7C7: F7C7: F7C8:	AA B5 10 60 0A B5 30 60 0A A5 50 60 0A A5 15 60 0A A5 15 00	01 E8 01 E1 00 01 D8 00 01 CF		BM BZ BNZ	LDA BPL RTS ASL TAX LDA BMI RTS ASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL	ROH,X BR1 A ROH,X BR1 A ROL,X ROH,X BR1 A ROL,X ROH,X BR1	TO X REG FOR INDEXING TEST FOR PLUS BRANCH IF SO DOUBLE RESULT-REG INDEX TEST FOR MINUS DOUBLE RESULT-REG INDEX TEST FOR ZERO (BOTH BYTES) BRANCH IF SO DOUBLE RESULT-REG INDEX TEST FOR NON-ZERO (BOTH BYTES) BRANCH IF SO
F7B7: F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F708: F702: F702: F702: F707: F707: F707: F702: F702: F702:	AA B5 10 60 0A B5 30 60 0A B5 50 60 0A B5 15 60 0A B5 15 00 60	01 E8 01 E1 00 01 D8 00 01 CF		BM BZ BNZ	LDA BPL RTS ASL LDA BMI RTS LDA BMI RTS LDA BES ASL TAX LDA BNE RTS	ROH, X BR1 A ROH, X BR1 A ROL, X ROH, X BR1 A ROL, X ROH, X BR1	TO X REG FOR INDEXING TEST FOR PLUS BRANCH IF SO DOUBLE RESULT-REG INDEX TEST FOR MINUS DOUBLE RESULT-REG INDEX TEST FOR ZERO (BOTH BYTES) BRANCH IF SO DOUBLE RESULT-REG INDEX TEST FOR NON-ZERO (BOTH BYTES) BRANCH IF SO
F7B7: F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F7C1: F7C2: F7C2: F7C2: F7C7: F7C7: F7C7: F7C7: F7C8: F7C7: F7C7: F7C7: F7C7: F7C8: F7C8: F7C8: F7C8: F7C9:	AA B5 10 0A B5 30 0A A5 55 50 0A A5 55 50 0A A5 50 0A A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0	01 E8 01 E1 00 01 D8 00 01 CF		BM BZ BNZ	LDA BPL RTS ASL LDA BMI RTS LDA ORA BEQ RTS LDA ORA BEQ RTS ASL	ROH, X BR1 A ROH, X BR1 A ROL, X ROH, X BR1 A ROL, X ROH, X BR1 A	TO X REG FOR INDEXING TEST FOR PLUS BRANCH IF SO DOUBLE RESULT-REG INDEX TEST FOR MINUS DOUBLE RESULT-REG INDEX TEST FOR ZERO (BOTH BYTES) BRANCH IF SO DOUBLE RESULT-REG INDEX TEST FOR NON-ZERO (BOTH BYTES) BRANCH IF SO DOUBLE RESULT-REG INDEX
F7B7: F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F7C1: F7C2: F7C2: F7C2: F7C7: F7C5: F7C7: F7C7: F7C8: F7C8: F7C8: F7C8: F7C9:	AA B5 10 0A A5 30 0A A5 50 0A A5 15 60 0A A5 15 00 AA A5 15 00 AA AA	01 E8 01 E1 00 01 D8 00 01 CF		BM BZ BNZ BM1	LDA BPL RTS ASL LDA BMI RTS ASL LDA ORA BEQ RTS ASL LDA ORA BRTS ASL TAX	ROH, X BR1 A ROH, X BR1 A ROL, X ROH, X BR1 A ROL, X ROH, X BR1 A	TO X REG FOR INDEXING TEST FOR PLUS BRANCH IF SO DOUBLE RESULT-REG INDEX TEST FOR MINUS DOUBLE RESULT-REG INDEX TEST FOR ZERO (BOTH BYTES) BRANCH IF SO DOUBLE RESULT-REG INDEX TEST FOR NON-ZERO (BOTH BYTES) BRANCH IF SO DOUBLE RESULT-REG INDEX
F7B7: F7B8: F7B8: F7B8: F7B8: F7B7: F7B7: F7C1: F7C2: F7C2: F7C2: F7C5: F7C7: F7C7: F7C7: F7C7: F7C7: F7C7: F7C7: F7C7: F7C7: F7C7: F7C8: F7C9:	AA B5 10 0A AB5 30 0A AB5 15 60 0A AB5 15 00 AA B5 10 00 AA B5 500 80 80 80 80 80 80 80 80 80 80 80 80 8	01 E8 01 E1 00 01 D8 00 01 CF		BM BZ BNZ BM1	LDA BPL RTS ASL LDA BMI RTS ASL LDA ORA BEQ RTS LDA ORA BNE RTSL LDA DAA DAA DAA	ROH, X BR1 A ROH, X BR1 A ROL, X ROH, X BR1 A ROL, X ROH, X BR1 A ROL, X	TO X REG FOR INDEXING TEST FOR PLUS BRANCH IF SO DOUBLE RESULT-REG INDEX TEST FOR MINUS DOUBLE RESULT-REG INDEX TEST FOR ZERO (BOTH BYTES) BRANCH IF SO DOUBLE RESULT-REG INDEX TEST FOR NON-ZERO (BOTH BYTES) BRANCH IF SO DOUBLE RESULT-REG INDEX CHECK BOTH BYTES
F7B7: F7B7: F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F7C8: F7C2: F7C2: F7C7: F7C7: F7C7: F7C7: F7C7: F7C7: F7C7: F7C7: F7C7: F7C7: F7C7: F7C8: F7C9: F7C8: F7C9: F7C9: F7C9: F7C8: F7C8: F7C8: F7C8: F7C8: F7C9: F7C8: F7C9: F7C8: F7C9: F7C8: F7C9: F7C8: F7C9: F7C8: F7C9: F7C8: F7C9: F7C8: F7C9: F7C8: F7C8: F7C9: F7C8: F7C8: F7C9: F7C8: F7C8: F7C9: F7C8: F7C8: F7C8: F7C8: F7C8: F7C9: F7C8:	AA B5 10 0A AB5 30 0A AB5 15 60 0A AB5 15 00 AA B5 15 00 AA B5 50 0A AB5 50 0A 30 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	01 E8 01 E1 00 01 D8 00 01 CF		BM BZ BNZ BM1	LDA BPL RTS ASL LDA BMI RTS ASL LDA BMI RTS ASL LDA BEQ RTS LDA BNE RTSL LDA BNE SASL LDA BNE SASL LDA BAN RTS ASL TAX LDA ASL ASL TAX LDA ASL TAX LDA ASL ASL ASL ASL ASL ASL ASL ASL ASL AS	ROH, X BR1 A ROH, X BR1 A ROL, X ROH, X BR1 A ROL, X ROH, X BR1 A ROL, X ROH, X	TO X REG FOR INDEXING TEST FOR PLUS BRANCH IF SO DOUBLE RESULT-REG INDEX TEST FOR MINUS DOUBLE RESULT-REG INDEX TEST FOR ZERO (BOTH BYTES) BRANCH IF SO DOUBLE RESULT-REG INDEX TEST FOR NON-ZERO (BOTH BYTES) BRANCH IF SO DOUBLE RESULT-REG INDEX CHECK BOTH BYTES FOR \$FF (MINUS 1)
F7B7: F7B7: F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F7C0: F7C1: F7C2:	AA B5 60 AA B5 30 60 AA B5 50 60 AA B5 50 60 AA B5 150 60 AA B5 530 60 AA B5 50 60 AA B5 50 60 AA AA B5 80 AA AA B5 80 AA AA B5 80 AA AA B5 80 AA AA B5 80 AA AA B5 80 AA AA B5 80 AA AA B5 80 AA AA B5 80 AA AA AA B5 80 AA AA B5 80 AA AA B5 80 AA AA B5 80 AA AA B5 80 AA AA B5 80 AA AA B5 80 AA AA AA AA AA AA AA AA AA AA AA AA AA	01 E8 01 E1 00 01 D8 00 01 CF 00 01 FF		BM BZ BNZ BM1	LDA BPL RTS ASL LDA BMI RTS ASL LDA BMI RTS ASL CORA BEQ RTS ASL CORA BNE RTS ASL LDA BNE RTS ASL CORA BNE RTS ASL CORA BEQ RTS CORA CORA CORA CORA CORA CORA CORA CORA	ROH, X BR1 A ROH, X BR1 A ROL, X ROH, X BR1 A ROL, X ROH, X BR1 A ROL, X ROH, X FFF	TO X REG FOR INDEXING TEST FOR PLUS BRANCH IF SO DOUBLE RESULT-REG INDEX TEST FOR MINUS DOUBLE RESULT-REG INDEX TEST FOR ZERO (BOTH BYTES) BRANCH IF SO DOUBLE RESULT-REG INDEX TEST FOR NON-ZERO (BOTH BYTES) BRANCH IF SO DOUBLE RESULT-REG INDEX CHECK BOTH BYTES FOR \$FF (MINUS 1)
F7B3: F7B4: F7B5: F7B5: F7B5: F7B5: F7C0: F7C2: F7C2: F7C2: F7C3: F7C5: F7C7: F7C7: F7C7: F7C7: F7C7: F7C7: F7D2: F7D2: F7D4: F7D5: F7D5: F7D5:	AA B5 60 AA B5 30 60 AA B5 50 60 AA B5 150 60 AA B5 150 60 AA B5 549 60 AA	01 E1 00 01 D8 00 01 CF 00 01 FF C4		BM BZ BNZ BM1	LDA BPL RTS ASL LDA BMI RTS ASL LDA BMI RTS ASL LDA BEQ RTS ASL LDA BEQ RTS ASL LDA BNE RTS ASL LDA BEQ RTS ASL CRAS LDA BEQ RTS RTS ASL RTS RTS RTS RTS RTS RTS RTS RTS RTS RTS	ROH,X BR1 A ROH,X BR1 A ROL,X ROH,X BR1 A ROL,X ROH,X BR1 A ROL,X ROH,X #\$FF BR1	TO X REG FOR INDEXING TEST FOR PLUS BRANCH IF SO DOUBLE RESULT-REG INDEX TEST FOR MINUS DOUBLE RESULT-REG INDEX TEST FOR ZERO (BOTH BYTES) BRANCH IF SO DOUBLE RESULT-REG INDEX TEST FOR NON-ZERO (BOTH BYTES) BRANCH IF SO DOUBLE RESULT-REG INDEX CHECK BOTH BYTES FOR \$FF (MINUS 1) BRANCH IF SO
F7B7 F7B7 F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F708: F7C1: F7C2: F7C2: F7C2: F7C2: F7C7: F7C7: F7C7: F7C8: F7C7: F7C9: F7C7: F7C9: F7C7: F7C9: F7C7: F7C9: F7C7: F7C9: F7	AA B500A B5300A B5500A B5500A B5500A B5500A B5500A B5549 F000A B5549	01 E8 01 E1 00 01 D8 00 01 CF 00 01 FF C4		BM BZ BNZ BM1	LDA BPL RTSL ASL LDA BMI RTSL ASL LDA BMI RTSL TAX LDA BRTS ASL TAX LDA BRTS ASL TAX LDA BRTS ASL TAX LDA BRTS RTSL TAX LDA RTSL TAX LDA RTSL RTSL RTSL RTSL RTSL RTSL RTSL RTSL	ROH, X BR1 A ROH, X BR1 A ROL, X ROH, X BR1 A ROL, X ROH, X BR1 A ROL, X ROH, X FFF BR1	TO X REG FOR INDEXING TEST FOR PLUS BRANCH IF SO DOUBLE RESULT-REG INDEX TEST FOR MINUS DOUBLE RESULT-REG INDEX TEST FOR ZERO (BOTH BYTES) BRANCH IF SO DOUBLE RESULT-REG INDEX TEST FOR NON-ZERO (BOTH BYTES) BRANCH IF SO DOUBLE RESULT-REG INDEX CHECK BOTH BYTES FOR \$FF (MINUS 1) BRANCH IF SO
F7B7 F7B7 F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F7C8: F7C2: F7C2: F7C2: F7C7: F7C7: F7C7: F7C7: F7C7: F7C7: F7C7: F7C7: F7D2: F7	AA B500A B5300A B5500A B5500A B5500A B5500A B5500A B5549 F00A B5549 F00A	01 E8 01 E1 00 01 D8 00 01 CF 00 01 FF C4		BM BZ BNZ BM1 BNM1	LDA BPL RTSL ASL LDA BMI RTSL LDA BMI RTSL LDA BRTSL LDA BRTSL LDA BRTSL LDA BRTSL LDA BRTSL LDA BRTSL LDA BRTSL LDA BRTSL LDA BRTSL TAX BRTSL TAX LDA BRTSL TAX LDA BRTSL TAX LDA BRTSL TAX LDA BRTSL TAX LDA BRTSL TAX LDA BRTSL TAX LDA BRTSL TAX LDA BRTSL TAX LAX LAX LAX LAX LAX LAX LAX LAX LAX L	ROH, X BR1 A ROH, X BR1 A ROL, X ROH, X BR1 A ROL, X ROH, X BR1 A ROL, X ROH, X #\$FF BR1 A	TO X REG FOR INDEXING TEST FOR PLUS BRANCH IF SO DOUBLE RESULT-REG INDEX TEST FOR MINUS DOUBLE RESULT-REG INDEX TEST FOR ZERO (BOTH BYTES) BRANCH IF SO DOUBLE RESULT-REG INDEX TEST FOR NON-ZERO (BOTH BYTES) BRANCH IF SO DOUBLE RESULT-REG INDEX CHECK BOTH BYTES FOR \$FF (MINUS 1) BRANCH IF SO DOUBLE RESULT-REG INDEX
F7B7: F7B7: F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F708: F702: F7C2: F7C2: F7C7: F7C7: F7C7: F7C7: F7C7: F7C7: F7D7: F7D7: F7D7: F7D7: F7D7:	AA B500A B500A B500A B550A B57	01 E8 01 E1 00 01 D8 00 01 CF 00 01 FFF C4		BM BZ BNZ BM1 BNM1	LDA BPL RTSL ASL LDA BMI RTSL LDA BMI RTSL LDA BEQ RTSL LDA BEQ RTSL LDA BEQ RTSL LDA BEQ RTSL LDA BEQ RTSL LDA BEQ RTSL LDA RTSL LDA BEQ RTSL LDA BEQ RTSL LDA RTSL LDA RTSL LDA RTSL LDA RTSL LDA RTSL LDA RTSL LDA RTSL LDA RTSL LDA RTSL LDA RTSL LDA RTSL LDA RTSL LDA RTSL LDA RTSL LDA RTSL LDA RTSL LDA RTSL RTSL LDA RTSL RTSL RTSL RTSL RTSL RTSL RTSL RTSL	ROH, X BR1 A ROH, X BR1 A ROL, X ROH, X BR1 A ROL, X ROH, X BR1 A ROL, X ROH, X #\$FF BR1 A ROL, X	TO X REG FOR INDEXING TEST FOR PLUS BRANCH IF SO DOUBLE RESULT-REG INDEX TEST FOR MINUS DOUBLE RESULT-REG INDEX TEST FOR ZERO (BOTH BYTES) BRANCH IF SO DOUBLE RESULT-REG INDEX TEST FOR NON-ZERO (BOTH BYTES) BRANCH IF SO DOUBLE RESULT-REG INDEX CHECK BOTH BYTES FOR \$FF (MINUS 1) BRANCH IF SO DOUBLE RESULT-REG INDEX
F7B7: F7B7: F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F7C1: F7C2: F7C2: F7C2: F7C2: F7C2: F7C5: F7C7: F7C7: F7C7: F7C8: F7C8: F7C9: F7C9: F7D2:	AA B500AA B500AA B500AA B500AAA B500AAA B500AAA B500AAA B500AAA B500AAA B500AAA B500AAA B500AAAA B500AAAAAA B500AAAAAAA B500AAAAAAAAAA	01 E8 01 E1 00 01 D8 00 01 CF 00 01 FFF C4 00 01		BM BZ BNZ BM1 BNM1	LDA BPL RTSL ASL LDA BMI RTSL ASL LDA BMI RTSL ASL LDA BASL CRA BEQ SASL LDA BRTSL AND RTSL AND RTSL AND RTSL ASL ASL ASL ASL ASL ASL ASL ASL ASL A	ROH, X BR1 A ROH, X BR1 A ROL, X ROH, X BR1 A ROL, X ROH, X BR1 A ROL, X ROH, X #\$FF BR1 A ROL, X ROH, X #\$FF	TO X REG FOR INDEXING TEST FOR PLUS BRANCH IF SO DOUBLE RESULT-REG INDEX TEST FOR MINUS DOUBLE RESULT-REG INDEX TEST FOR ZERO (BOTH BYTES) BRANCH IF SO DOUBLE RESULT-REG INDEX TEST FOR NON-ZERO (BOTH BYTES) BRANCH IF SO DOUBLE RESULT-REG INDEX CHECK BOTH BYTES FOR \$FF (MINUS 1) BRANCH IF SO DOUBLE RESULT-REG INDEX
F7B7: F7B7: F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F708: F7C2:	AA B500AA B500AA B500AA B550AA B1500A B55490 AB55490 AB5549 C0AA B55490 AB55490 AA B55490 AB55490 AA B55490 AA B55490 AA B55490 AA B55490 AA B55490 AA B55400 AA B555400 AA B555400 AA B555400 AA B555400 AA B555400 AA B555400 AA B555400 AA B555400 AA B555400 AA B555400 AA B555400 AA B555400 AA B555400 AA B555400 AA B555400 AA B555400 AA B555700 AA B555700 AA B55700 AA B55700 AA B55700 AA B55700 AA B55700 AA B55700 AA B55700 AA B55700 AA B55700 AA B55700 AA B55700 AA B570000 AA B57000 AA B5700	01 E1 00 01 D8 00 01 CF 00 01 FF C4 00 01 FF		BM BZ BNZ BM1 BNM1	LDA BPL RTSL ASL LDA BMI RTSL ASL LDA BMI RTSL ASL LDA BASL CRA BEQ SASL LDA BNE SASL LDA BNE SASL LDA BEQ SASL LDA BEQ SASL LDA BEQ SASL LDA BEQ SASL LDA BEQ SASL CRA SASL LDA BEQ SASL LDA BEQ SASL LDA BEQ SASL LDA BEQ SASL LDA BEQ SASL LDA BEQ SASL LDA BEQ SASL LDA BEQ SASL ASL ASL ASL ASL ASL ASL ASL ASL AS	ROH, X BR1 A ROH, X BR1 A ROL, X ROH, X BR1 A ROL, X ROH, X BR1 A ROL, X ROH, X #\$FF BR1 A ROL, X ROH, X #\$FF	TO X REG FOR INDEXING TEST FOR PLUS BRANCH IF SO DOUBLE RESULT-REG INDEX TEST FOR MINUS DOUBLE RESULT-REG INDEX TEST FOR ZERO (BOTH BYTES) BRANCH IF SO DOUBLE RESULT-REG INDEX TEST FOR NON-ZERO (BOTH BYTES) BRANCH IF SO DOUBLE RESULT-REG INDEX CHECK BOTH BYTES FOR \$FF (MINUS 1) BRANCH IF SO DOUBLE RESULT-REG INDEX CHECK BOTH BYTES FOR NO \$FF
F7B7 F7B7 F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F708: F702: F7C2: F7	AA B500AA B500AA B500AA B5500A B5500A B55490 C0AA B53490 C0AA B53490 C0AA B53490 C0AA B53490 C0AA B53490 C0AA B53490 C0AA B53490 C0AA B53490 C0AA B53490 C0AA B53490 C0AA B53490 C0AA B53490 C0AA B53490 C0AA B53490 C0AA B53490 C0AA B53490 C0AA B53490 C0AA B53400 C0AA B53400 C0AA B53400 C0AA B53500 C0AA B53500 C0AA B53500 C0AA B53500 C0AA B53500 C0AA B53500 C0AA B53500 C0AA B53500 C0AA C0AA B5300 C0AA C0AA B5300 C0AA B5300 C0AA B5300 C0AA C0AA B5300 C0AA C0AA C0AA C0AA C0AA C0AA C0AA C	01 E8 01 E1 00 01 D8 00 01 CF 00 01 FFF C4 00 01 FFF B9		BM BZ BNZ BM1 BNM1	LDA BPL RTSL ASL LDA BMI RTS ASL LDA BMI RTSL ASL LDA BMI RTSL ASL LDA BNE RTSL ASL LDA BNE RTSL ASL LDA BNE RTSL ASL LDA BNE RTSL ASL LDA BAL RTSL ASL LDA BMI RTSL ASL ASL ASL ASL ASL ASL ASL ASL ASL A	ROH, X BR1 A ROH, X BR1 A ROL, X ROH, X BR1 A ROL, X ROH, X BR1 A ROL, X ROH, X #\$FF BR1 A ROL, X ROH, X #\$FF BR1	TO X REG FOR INDEXING TEST FOR PLUS BRANCH IF SO DOUBLE RESULT-REG INDEX TEST FOR MINUS DOUBLE RESULT-REG INDEX TEST FOR ZERO (BOTH BYTES) BRANCH IF SO DOUBLE RESULT-REG INDEX TEST FOR NON-ZERO (BOTH BYTES) BRANCH IF SO DOUBLE RESULT-REG INDEX CHECK BOTH BYTES FOR \$FF (MINUS 1) BRANCH IF SO DOUBLE RESULT-REG INDEX CHECK BOTH BYTES FOR NO \$FF BRANCH IF NOT MINUS 1
F7B7: F7B7: F7B8: F7B8: F7B8: F7B8: F7B8: F7B8: F7C0: F7C1: F7C2:	AA B500AA B500AA B500AA B5500AA B5500AA B55400AA B55500AA B55400AAA B55400A	01 E8 01 E1 00 01 CF 00 01 FFF C4 00 01 FF9		BM BZ BNZ BM1 BNM1	LDA BPL RTSL ASL LDA BMI RTS ASL ADA BEQ RTSL ASL ADA BEQ RTSL ASL ADA BRES ASL ADA BRES LDA BRES ASL ADA BRES ASL ADA BRES RTSL ASL ADA BASL RTSL ASL ADA BASL RTSL ASL ADA BASL ADA BASL ASL ADA BASL ADA BASL ADA BASL ADA BASL ADA BASL ASL ADA BASL ASL ADA BASL ADA ADA BASL ADA ADA BASL ADA ADA BASL ADA BASL ADA ADA ADA BASL ADA ADA BASL ADA BASL ADA ADA ADA ADA ADA ADA ADA BASL ADA ADA ADA ADA ADA ADA ADA ADA ADA AD	ROH, X BR1 A ROH, X BR1 A ROL, X ROH, X BR1 A ROL, X ROH, X BR1 A ROL, X ROH, X #\$FF BR1 A ROL, X ROH, X #\$FF BR1	TO X REG FOR INDEXING TEST FOR PLUS BRANCH IF SO DOUBLE RESULT-REG INDEX TEST FOR MINUS DOUBLE RESULT-REG INDEX TEST FOR ZERO (BOTH BYTES) BRANCH IF SO DOUBLE RESULT-REG INDEX TEST FOR NON-ZERO (BOTH BYTES) BRANCH IF SO DOUBLE RESULT-REG INDEX CHECK BOTH BYTES FOR \$FF (MINUS 1) BRANCH IF SO DOUBLE RESULT-REG INDEX CHECK BOTH BYTES FOR NO \$FF BRANCH IF NOT MINUS 1

F7EB:	20	66	F7		JSR	DCR	DECR STACK POINTER
F7EE:	Al	00			LDA	(ROL,X)	POP HIGH RETURN ADR TO PC
F7F0:	85	lF			STA	R15H	
F7F2:	20	66	F7		JSR	DCR	SAME FOR LOW-ORDER BYTE
F7F5:	Al	00			LDA	(ROL,X)	
F7F7:	85	lE			STA	R15L	
F7F9:	60				RTS		
F7FA:	4C	C7	F6	RTN	JMP	RTNZ	

## 6502 MICROPROCESSOR INSTRUCTIONS

LDX

ADC	Add Memory to Accumulator with
	Carry
AND	"AND" Memory with Accumulator
ASL	Shift Left One Bit (Memory or
	Accumulator)
BCC	Branch on Carry Clear
BCS	Branch on Carry Set
BEQ	Branch on Result Zero
BIT	Test Bits in Memory with
	Accumulator
BMI	Branch on Result Minus
BNE	Branch on Result not Zero
BPL	Branch on Result Plus
BRK	Force Break
BVC	Branch on Overflow Clear
BVS	Branch on Overflow Set
CLC	Clear Carry Flag
CLD	Clear Decimal Mode
CLI	Clear Interrupt Disable Bit
CLV	Clear Overflow Flag
CMP	Compare Memory and Accumulator
CPX	Compare Memory and Index X
CPY	Compare Memory and Index Y
DEC	Decrement Memory by One
DEX	Decrement index X by One
DEY	Decrement Index Y by One
EOR	"Exclusive-Or" Memory with
	Accumulator
INC	Increment Memory by One
INX	Increment Index X by One
INY	Increment Index Y by One
JMP	Jump to New Location
JSR	Jump to New Location Saving
	Return Address

LDY Load Index Y with Memory LSR Shift Right one Bit (Memory or Accumulator) NOP No Operation ORA OR Memory with Accumulator PHA Push Accumulator on Stack PHP Push Processor Status on Stack PLA Pull Accumulator from Stack PLP Pull Processor Status from Slack ROL Rotate One Bit Left (Memory or Accumulator) ROR Rotate One Bit Right (Memory or Accumulator) RTI Return from Interrupt RTS Return from Subroutine SBC Subtract Memory from Accumulator with Borrow SEC Set Carry Flag SED Set Decimal Mode SEI Set Interrupt Disable Status STA Store Accumulator in Memory STX Store Index X in Memory STY Store Index Y in Memory TAX Transfer Accumulator to Index X TAY Transfer Accumulator to Index Y тѕх Transfer Stack Pointer to Index X ТХА Transfer Index X to Accumulator TXS Transfer Index X to Stack Pointer TYA Transfer Index Y to Accumulator

LDA Load Accumulator with Memory

Load Index X with Memory
## THE FOLLOWING NOTATION APPLIES TO THIS SUMMARY:

A	Accumulator
X, Y	Index Registers
М	Memory
Ē	Borrow
Р	Processor Status Register
S	Stack Pointer
1	Change
_	No Change
+	Add
Λ	Logical AND
-	Subtract
¥	Logical Exclusive OR
ŧ	Transfer From Slack
¥	Transfer To Stack
+	Transfer To
*	Transfer To
V	Logical OR
PC	Program Counter
PCH	Program Conter High
PCL	Program Counter low
OPER	Operrand
#	Immediate Addressing Mode

FIGURE 1. ASL-SHIFT LEFT ONE BIT OPERATION



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



FIGURE 3.



NOTE 1: BIT – TEST BITS

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

### **PROGRAMMING MODEL**



ACCUMULATOR

INDEX REGISTER Y

INDEX REGISTER X

PROGRAM COUNTER

STACK POINTER



# INSTRUCTION CODES

Name Description	Operation	Addressing Mode	Assembly Language Form	HEX OP Code	No. Bytes	"P" States Reg. N Z C I D V
ADC Add memory to accumulator with carry	A-M-C -+ A.C	Immediate Zero Page Zero Page X Absolute Absolute X Absolute Y (Indirect,X) (Indirect,Y)	ADC #Oper ADC Oper ADC Oper,X ADC Oper,X ADC Oper,X ADC Oper,Y ADC Oper,Y ADC Oper,Y	69 65 75 70 79 61 71	000000000	<u> </u>
AND AND" memory with accumulator	AAM + A	Immediate Zero Page Zero Page X Absolute Absolute X Absolute Y (Indirect,X) (Indirect,X)	AND #Oper AND Oper AND Oper,X AND Oper,X AND Oper,X AND Oper,X AND (Oper,X) AND (Oper,X)	29 25 35 30 39 31	5 2 3 3 3 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5	<i>^</i> /
<b>ASL</b> Shift left one bit (Memory or Accumulator)	(See Figure 1)	Accumulator Zero Page Zero Page X Absolute Absolute	ASL A ASL Oper ASL Oper,X ASL Oper,X ASL Oper,X	0A 06 16 0E 1E	3357	^^^
BCC Branch on carry clear	Branch on C=0	Relative	BCC Oper	06	2	
<b>BCS</b> Branch on carry set	Branch on C=1	Relative	BCS Oper	BO	2	
<b>BEQ</b> Branch on result zero	Branch on Z=1	Relative	BEQ Oper	F0	2	
<b>BIT</b> Test bits in memory with accumulator	AAM, M7→N, M5→V		BIT* Oper BIT* Oper	24 2C	3 7	M7√M6
<b>BMI</b> Branch on result minus	Branch on N=1	Relative	BMI Oper	30	2	
<b>BNE</b> Branch on result not zero	Branch on Z=0	Relative	BNE Oper	00	2	
<b>BPL</b> Branch on result plus	Branch on N=0	Relative	BPL oper	10	2	
<b>BRK</b> Force Break	Forced Interrupt PC+2 \$ P \$	Implied	BRK*	00	-	
<b>BVC</b> Branch on overflow clear	Branch on V=0	Relative	BVC Oper	50	2	
e 1: Bit 6 and 7 are transferred to the then then $Z = 1$ ; otherwise $Z = 0$ .	status register. If the res	ult of A AND M is zero	Note 2: AB	RK comme	and canno	t be masked by setting

Name Description	Operation	Addressing Mode	Assembly Language Form	HEX OP Code	No. Bytes	"P" States Reg. N Z C I D V
<b>BVS</b> Branch on overflow set	Branch on V=1	Relative	BVS Oper	70	2	
<b>CLC</b> Clear carry flag	0 <del>+</del> C	Implied	CLC	18	-	0
<b>CLD</b> Clear decimal mode	<b>□</b> + 0	Implied	CLD	D8	-	0
CLI	<b>-+</b> 0	Implied	CLI	58	-	0
<b>CLV</b> Clear overflow flag	∧ <del>+</del> -0	lm plied	CLV	B8	-	00
<b>CMP</b> Compare memory and accumulator	A — M	Immediate Zero Page Zero Page, X Absolute, X Absolute, X Absolute, Y (Indirect, X) (Indirect, X)	CMP #Oper CMP Oper CMP Oper,X CMP Oper,X CMP Oper,X CMP Oper,X CMP (Oper,X) CMP (Oper),Y	C9 C5 C0 C0 D0 C1 C1 C1	5 5 5 5 5 5 5 5	V V V
<b>GPX</b> compare memory and index X	w—x	lmmediate Zero Page Absolute	CPX #0per CPX 0per CPX 0per	E0 E4 EC	2 3	///
<b>GPY</b> Compare memory and index Y	м — Х	lmmediate Zero Page Absolute	CPY #Oper CPY Oper CPY Oper	C0 C4 CC	2 2 3	~~~^^
DEC Decrement memory by one	M → 1 → M	Zero Page Zero Page,X Absolute Absolute,X	DEC Oper DEC Oper,X DEC Oper,X DEC Oper,X	C6 D6 CE DE	2 3 3	~~~/
<b>DEX</b> Decrement index X by one	X → 1 → X	Implied	DEX	CA	1	~~~^^
<b>DEY</b> Decrement index Y by one	Υ — 1 → Υ	lmplied	DEY	88	1	~~~^ <i>\</i>

102

INSTRUCTION CODES

"P" States Reg. N Z C I D V	<i>^ /</i>	//	V	//			<i>//</i>	<i>/</i> /	//
No. Bytes	2 2 3 3 3 2 2 2	2 3 3	-	١	3 3	3	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0	0 0 0 0 0 0
HEX OP Code	49 55 40 50 59 41	E6 F6 FE	E8	C.8	4C 6C	20	A9 A5 A0 B1 A1 B1	A2 A6 B6 BE	A0 A4 AC BC
Assembly Language Form	EOR #Oper EOR Oper EOR Oper,X EOR Oper,X EOR Oper,Y EOR (Oper,Y) EOR (Oper,Y)	INC Oper INC Oper,X INC Oper INC Oper,X	INX	INY	JMP Oper JMP (Oper)	JSR Oper	LDA #Oper LDA Oper LDA Oper,X LDA Oper,X LDA Oper,X LDA Oper,Y LDA (Oper,X) LDA (Oper,X)	LDX #Oper LDX Oper LDX Oper,Y LDX Oper,Y LDX Oper,Y	LDY #Oper LDY Oper LDY Oper,X LDY Oper LDY Oper,X
Addressing Mode	Immediate Zero Page Zero Page,X Absolute Absolute,X Absolute,X Absolute,Y (Indirect,X) (Indirect,Y)	Zero Page Zero Page,X Absolute Absolute,X	Implied	Implied	Abs olute Indirect	Absolute	Immediate Zero Page Zero Page,X Absolute Absolute,X Absolute,X Absolute,Y (Indirect,X)	Immediate Zero Page Zero Page, Absolute Absolute,Y	Immediate Zero Page Zero Page,X Absolute Absolute,X
Operation	A V M + A	M + 1 ₩ <b>→</b> M	X + 1 - X	Υ + 1 <del>- Υ</del>	(PC+1) → PCL (PC+2) → PCH	PC+2 ↓ (PC+1) → PCL (PC+2) → PCH	A + M	X ↔ M	λ <b>+</b> W
Name Description	EOR "Exclusive-Or' memory with accumulator	INC Increment memory by one	INX Increment index X by one	INY Increment index Y by one	<b>JMP</b> Jump to new location	<b>JSR</b> Jump to new location saving return address	LDA Load accumulator with memory	LDX Load index X with memory	LDY Load index Y with memory

Name Description	Operation	Addressing Mode	Assembly Language Form	LEX Code	No. Bytes	"P" States Reg. N Z C I D V
<b>LSR</b> Shift right one but (memory or accumulator)	(See Figure 1)	Accumulator Zero Page Zero Page,X Absolute Absolute,X	LSR A LSR Oper LSR Oper,X LSR Oper,X LSR Oper,X	44 46 46 46 56 56	33557	/ /0
NOP No operation	No operation	lm plied	AON	EA	1	
<b>ORA</b> "OR" memory with accumulator	A V M → A	Immediate Zero Page Zero Page,X Absolute_X Absolute,X Absolute,Y (Indirect,X) (Indirect,X)	0RA #0per 0RA 0per 0RA 0per,X 0RA 0per,X 0RA 0per,X 0RA 0per,Y 0RA (0per,Y) 0RA (0per,)Y	09 05 15 00 110 119 11	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	//
PHA Push accumulator on stack	4 t	lm plied	рна	48		
PHP Push processor status on stack	+ d	Implied	dHd	08	1	
PLA Push accumulator from stack	A +	Implied	PLA	68	1	<i>\</i> /
PLP Pull processor status from stack	P 4	Implied	PLP	28	-	From Stack
ROL Rotate one bit left (memory or accumulator)	(See Figure 2)	Accumulator Zero Page Zero Page,X Absolute Absolute,X	ROL A ROL Oper ROL Oper,X ROL Oper,X ROL Oper,X	2A 26 36 3E	1 2 3	^^
ROR Rotate one bit right (memory or accumulator)	(See Figure 3)	Accumulator Zero Page Zero Page,X Absolute Absolute,X	ROR A ROR Oper ROR Oper,X ROR Oper,X ROR Oper,X	6A 66 76 6E	3322	^/

# INSTRUCTION CODES

Name Description	Operation	Addressing Mode	Assembly Language Form	HEX Code	No. Bytes	"P" States Reg. N Z C I D V
<b>RTI</b> Return from interrupt	P † PC †	Implied	RTI	40	-	From Stack
<b>RTS</b> Return from subroutine	PC∳, PC→1 → PC	lm plied	RTS		1	
SBC Subtract memory from accumulator with borrow	A - M - <del>C</del> A	Immediate Zero Page Zero Page,X Absolute,X Absolute,X Absolute,Y (Indirect,X) (Indirect,X)	SBC #Oper SBC Oper,X SBC Oper,X SBC Oper,Y SBC Oper,Y SBC (Oper,X) SBC (Oper,Y SBC #Oper SBC #Oper	E E E E E E E E E E E E E E E E E E E	~~~~~~	VV
<b>SEC</b> Set carry flag	1 <del>- </del> C	Implied	SEC	38	1	1
<b>SED</b> Set decimal mode	1 <del>- •</del> D	lm plied	SED	F8	-	1
<b>SEI</b> Set interrupt disable status	Ť.	Implied	SEI	78	-	
<b>STA</b> store accumulator in memory	A + M	Zero Page Zero Page,X Absolute Absolute,X Absolute,Y (Indirect,X) (Indirect,X)	STA Oper STA Oper, X STA Oper, X STA Oper, Y STA Oper, Y STA (Oper, X) STA (Oper), Y	85 95 99 91 91	0 0 0 0 0 0 0 0	
<b>STX</b> Store index X in memory	₩ <del>+</del> X	Zero Page Zero Page,Y Absolute	STX Oper STX Oper,Y STX Oper	86 96 8E	0 0 0	
<b>STY</b> Store index Y in memory	М ↔ Ү	Zero Page Zero Page,X Absolute	STY Oper STY Oper,X STY Oper	84 94 BC	2 3	
<b>TAX</b> Transfer accumulator to index X	A → X	lmplied	ТАХ	АА	1	<i>√√</i>
<b>TAY</b> Transfer accumulator to index Y	A Y	Implied	ТАҮ	A8	-	/
<b>TSX</b> Transfer stack pointer to index X	S ➡ X	Implied	TSX	BA	-	<i>\</i>

Name Description	Operation	Addressing Mode	Assembly Language Form	HEX OP Code	No. Bytes	"P" States Reg. N Z C I D V
<b>TXA</b> Transfer index X to accumulator	X X	Implied	ТХА	8A	-	~~~^ <i>\</i>
<b>TXS</b> Transfer index X to stack pointer	S <del>↓</del> X	Implied	TXS	9A	1	
<b>TYA</b> Transfer index Y to accumulator	Y A	Implied	ТҮА	98	1	~~~^ <i>\</i>

# HEX OPERATION CODES

00 — BRK	2F — NOP	5ELSR Absolute, X	8D — STA — Absolute	84 — LDY — Zero Page, X	D8 — NOR
01 — ORA — (Indirect. XI	30 — BM!	SF — NOP	BE — STX — Absolute	85 — LDA — Zero Page, X	DCMOP
02 — NOP	31 — AND — (Indirect), V	60 — RTS	8F — NOP	B6 — LOX — Zero Page, Y	DO
03 — NOR	32 — NOP	61 — ADC — Indirect, X	90 — BCC	87 — NOP	DE — DEC — Absolute, X
04 — NOR	33 — NOP	62 — NOR	91 — STA — (Indirect), Y	B8 — CLV	OF - NOP
05 — ORA — Zero Page	34 — NOP	63 — NOP	92 — NOP	89 — LDA — Absolute. Y	E0 — CPX — Immediate
06 — ASL — Zero Page	35 — AND — Zero Page, X	64 — NOR	93 — NOR	BA — TSX	El - SBC - (Indirect, X)
07 — NOP	36 — ROL — Zero Page. X	65 — ADC — Zero Page	94 — STY — Zero Page. X	BB — NOP	E2 — NOP
08 — PHP	37 — NOP	66 — ROR — Zero Page	95 — STA — Zero Page, X	BC — LDY — Absolute.X	E3 — NOP
09 — ORA — Immediate	38 — SEC	67 — NOP	96 — STX — Zero Page, Y	BD — LDA — Absolute, X	E4 — CPX — Zero Page
OA — ASL — Accumulator	39 — AND — Absolute, Y	68 — PLA	97 — NOP	BE — LOX — Absolute, Y	E5 — SBC —Zero Page
OB — NOP	3A — NOP	69 — ADC — Immediate	98 — TVA	BF — NOP	E6 — INC—Zero Page
OC - NOP	3B — NOP	6A — ROR — Accumulator	99 — STA — Absolute, Y	CO — CPY — Immediate	E7 — NOP
OD — ORA — Absolute	3C — NOP	6B — NOP	9A — TXS	C1 — CMP — (Indirect, X	eb — Inx
OEASLAbsolute	3D — AND — Absolute, X	6C — JMP — Indirect	9B — MOP	C2 — NOP	E9 — SBC — Immediate
OF - NOP	3E — ROL — Absolute, X	6D — ADC — Absolute	9C - NOP	C3 — NOP	EA — NOP
10 — BPL	3F — NOP	6E — ROR — Absolute	9D — STA — Absolute, X	C4 — CPY — Zero Page	EB – NOP
11 — ORA — (Indirect), Y	40 — RTI	6F — NOP	9E — NOP	C5 — CMP — Zero Page	EC — CPX — Absolute
12 — NOP	41 — EOR — Indirect. X	70 — BVS	9F — NOP	C6 — DEC — Zero Page	ED — SBC — Absolute
13 — NOP	42 — NOP	71 — ADC — (Indirect), Y	AO — LDY — Immediate	C7 — NOP	EE — INC — Absolute
14 — NOR	43 — NOP	72 — NOP	AI — LDA —(Indirect, XI	C8 — INY	EE - NOP
15 — ORA — Zero Page, X	44 — NOR	73 — МОР	A2 —LOX — Immediate	C9 — CMP — Immediate	FO — BM
16 — ASL — Zero Page. X	45 — EOR — Zero Page	74 — NOP	A3 — NOR	CA — DEX	F1 — SBC — (Indirect), Y
17 — NOR	46 — LSR — Zero Page	75 — ADC — Zero Page, X	A4 — LDY — Zero Page	CBMOP	F2 — NOP
18 — CLC	47 — NOP	76 — ROR — Zero Page. X	AS — LDA — Zero Page	CC — CPY — Absolute	F3 — NOR
19 — ORA — Absolute, Y	48 — PHA	77 — NOP	A6 — LDX — Zero Page	CD —CMP — Absolute	F4 — NOP
IA — NOR	49 — EOR — Immediate	78 — SEI	AI — NOP	CE — DEC DEC — Absolute	F5 — SBC — Zero Page, X
1B — NOP	4A — LSR — Accumulator	79 — ADC — Absolute, Y	А8 — ТАҮ	CF - NOP	F6 — INC — Zero Page. X
1C	4B —NOR	7A — NOP	A9 — LDA — Immediate	DO — BNE	F7 — NOP
10 — ORA — Absolute, X	4C — JMP — Absolute	7B — NOP	АА — ТАХ	D1 — CMP — (Indirect), V	F8 — SED
1E — ASL — Absolute.X	4D — EOR — Absolute	7C — NOP	AB — NOP	D2 — NOP	F9 — SBC — Absolute. Y
1F — NOP	4E — LSR — Absolute	7D — ADC — Absolute, X NOP	AC – LDY – Absolute	D3 — NOR	FA — NOP
20 — JSR	4F —MOP	7E — 808 — Absolute, X NOP	AD —Absolute	D4 — NOP	FB — NOP
21 — AND —(Indirect, X)	50 — BVC	7F — NOP	AE — LDX — Absolute	05 — CMP — Zero Page. X	FC - NOP
22 – NOR	51 — EOR Indirect, Y	80 — NOR	AF	D6 — DEC — Zero Page, X	FD — SBC — Absolute. X
23 — NOP	52 — NOP	81 — STA — (Indirect, Xi	B0 – BCS	07 —NOR	FE — INC — Absolute, X
24 — BIT — Zero Page	53 — NOP	82 — NOP	81 — LDA — (Indirect), Y	08 — CLD	FF — NOP
25 — AND — Zero Page	54 — NOP	83 — NOP	B2 — NOP	D9 —CMP — Absolute. Y	
26 — ROL — Zero Page	55 — EOR — Zero Page, X	84 —STY — Zero Page	B3 — NOP	DA — NOP	
27 — NOP	56 — LSR — Zero Page, X	85 — STA — Zero Page			
28 — PLP	57 — NOP	86 — STX — Zero Page			
29 — AND — Immediate	58 — CLI	87 — NOP			
2A — ROL — Accumulator	59 — FOR Absolute, Y	88 — DEY			
2B — NOP	5A — NOP	89 — NOP			
2C — BIT — Absolute	5B — NOP	8A — TXA			
2D — AND — Absolute	5C — NOP	88 — NOP			
2E — ROL — Absolute	50 — EOR — Absolute, X	8C — STY — Absolute			

## APPLE II HARDWARE

- 1. Getting Started with Your APPLE II Board
- 2. APPLE II Switching Power Supply
- 3. Interfacing with the Home TV
- 4. Simple Serial Output
- 5. Interfacing the APPLE Signals, Loading, Pin Connections
- 6. Memory Options, Expansion, Map, Address
- 7. System Timing
- 8. Schematics

### INTRODUCTION

### ITEMS YOU WILL NEED:

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

- a. 1 ea. APPLE II P.C. Board complete with specified RAM memory.
- b. l ea. d.c. power connector with cable.
- c. l ea. 2" speaker with cable.
- d. l ea. Preliminary Manual
- e. l ea. Demonstration cassette tapes. (For 4K: 1 cassette (2 programs); 16K or greater: 3 cassettes.
- f. 2 ea. 16 pin headers plugged into locations A7 and J14.

In addition you will need:

- g. A color TV set (or B & W) equipped with a direct video input connector for best performance or a commercially available RF modulator such as a "Pixi-verter"<sup>tm</sup> Higher channel (7-13) modulators generally provide better system performance than lower channel modulators (2-6).
- h. The following power supplies (NOTE: current ratings do not include any capacity for peripheral boards.):
  - 1. +12 Volts with the following current capacity!

a. For 4K or 16K systems - 350mA.

- b. For 8K, 20K or 32K 550mA.
- c. For 12K, 24K, 36K or 48K 850mA.
- 2. +5 Volts at 1.6 amps
- 3. -5 Volts at 10mA.
- 4. OPTIONAL: If -12 Volts is reouired by your keyboard. (If using an APPLE II supplied keyboard, you will need -12V at 50mA.)

- i. An audio cassette recorder such as a Panasonic model RQ-309 DS which is used to load and save programs.
- j. An ASCII encoded keyboard equipped with a "reset" switch.

k.Cable for the following:

- 1. Keyboard to APPLE II P.C.B.
- 2. Video out 75 ohm cable to TV or modulator
- 3. Cassette to APPLE II P.C.B. (1 or 2)

Optionally you may desire:

- Game paddles or pots with cables to APPLE II Game I/O connector. (Several demo programs use PDL(O) and "Pong" also uses PDL(1).
- m. Case to hold all the above

Final Assembly Steps

- Using detailed information on pin functions in hardware section of manual, connect power supplies to d.c. cable assembly. Use both ground wires to miminize resistance. With cable assembly disconnected from APPLE II mother board, turn on power supplies and verify voltages on connector pins. Improper supply connections such as reverse polarity can severely damage your APPLE II.
- 2. Connect keyboard to APPLE II by unplugging leader in location A7 and wiring keyboard cable to it, then plug back into APPLE II P.C.B.
- 3. Plug in speaker cable.
- 4. Optionally connect one or two game paddles using leader supplied in socket located at J14.
- 5. Connect video cable.
- 6. Connect cable from cassette monitor output to APPLE II cassette input.
- 7. Check to see that APPLE II board is not contacting any conducting surface.
- 8. With power supplies turned off, plug in power connector to mother board then recheck all cableing.

### POWER UP

- Turn power on. If power supplies overload, immediately turn off and recheck power cable wiring. Verify operating supply voltages are within +3% of nominal value.
- You should now have random video display. If not check video level pot on mother board, full clockwise is maximum video output. Also check video cables for opens and shorts. Check modulator if you are using one.
- Press reset button. Speaker should beep and a "\*" prompt character with a blinking cursor should appear in lower left on screen.
- 4. Press "esc" button, release and type a "@" (shift-P) to clear screen.. You may now try "Monitor" commands if you wish. See details in "Monitor" software section.

### RUNNING BASIC

- Turn power on; press reset button; type "control B" and press return button. A ">" prompt character should appear on screen indicating that you are now in BASIC.
- 2. Load one of the supplied demonstration cassettes into recorder. Set recorder level to approximately 5 and start recorder. Type "LOAD" and return. First beep indicates that APPLE II has found beginning of program; second indicates end of program followed by ">" character on screen. If error occurs on loading, try a different demo tape or try changing cassette volume level.
- 3. Type RUN and carriage return to execute demonstration program. Listings of these are included in the last section of this manual.

### THE APPLE II SWITCHING POWER SUPPLY

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

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

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

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

All DC outputs are regulated at the same time and one of the four outputs (the +5 volt supply) is compared to a reference voltage with the difference error fed to a feedback loop to assist the oscillator in running at the needed frequency. Since all DC outputs are regulated together, their voltages will reflect to some extent unequal loadings. For example; if the +5 supply is loaded very heavily, then all other supply voltages will increase in voltage slightly; conversely, very light loading on the +5 supply and heavy loading on the +12 supply will cause both it and the others to sag lightly. If precision reference voltages are needed for peripheral applications, they should be provided for in the peripheral design.

In general, the APPLE II design is conservative with respect to component ratings and operating termperatures. An over-voltage crowbar shutdown system and an auxilliary control feedback loop are provided to ensure that even very unlikely failure modes will not cause damage to the APPLE II computer system. The over-voltage protection references to the DC output voltages only. The AC line voltage input must be within the specified limits, i.e., 107V to 132V.

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

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

Nominal load current ratios are: The +12V supply load is  $\frac{1}{2}$  that of the +5V. The - 5V supply load is 1/10 that of the +5V. The -12V supply load is 1/10 that of the +5V.

The supply voltages are  $+5.0 \pm 0.15$  volts,  $+11.8 \pm 0.5$  volts,  $-12.0 \pm 1V$ ,  $-5.2 \pm 0.5$  volts. The tolerances are greatly reduced when the loads are close to nominal.

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

> + 12V - use no more than 250 mA + 5V - use no more than 500 mA - 5V - use no more than 200 mA - 12V - use no more than 200 mA

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

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

111

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

Users report success with the following RF modulators:

the "PixieVerter" (a kit) ATV Research 13th and Broadway Dakota City, Nebraska 68731 the "TV-1" (a kit) **UHF** Associates 6037 Haviland Ave. Whittier, CA 90601 the "Sup-r-Mod" by (assembled & tested) M&R Enterprises P.O. Box 1011 Sunnyvale, CA 94088 the RF Modulator (a P.C. board) Electronics Systems P.O. Box 212 Burlingame, CA 94010

Most of the above are available through local computer stores.

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

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

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

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

carrier of the RF modulator to produce faint spurious background patterns (called "worms") This RFI "trash" must be of quite a low level if worms are to be prevented. In fact, these spurious beats must be 40 to 50db below the signal level to reduce worms to an acceptable level. When it is remembered that only 2 to 6 mV (across  $300\Omega$ ) is presented to the VHF input of the TV set, then stray RFI getting into the TV must be less than  $50\mu$ V to obtain a clean picture. Therefore we recommend that a good, co-ax cable be used to carry the signal from any modulator to the TV set, such as RG/59u (with copper shield), Belden #8241 or an equivalent miniature type such as Belden #8218. We also recommend that the RF modulator been enclosed in a tight metal box (an unpainted die cast aluminum box such as Pomona #2428). Even with these precautions, some trouble may be encountered with worms, and can be greatly helped by threading the coax cable connecting the modulator to the TV set repeatedly through a Ferrite toroid core Apple Computer supplies these cores in a kit, along with a 4 circuit connector/cable assembly to match the auxilliary video connector found on the Apple II board. This kit has order number A2MØ1ØX. The M&R "Sup-r-Mod" is supplied with a coax cable and toroids.

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

Another caution concerns possible long term effects on the TV picture tube. Most home TV sets have "Brightness" and "Contrast" controls with a very wide range of adjustment. When an un-changing picture is displayed with high brightness for a long period ,a faint discoloration of the TV CRT may occur as an inverse pattern observable with the TV set turned off. This condition may be avoided by keeping the "Brightness" turned down slightly and "Contrast" moderate. The Apple II is equipped with a 16 pin DIP socket most frequently used to connect potentiometers, switches, etc. to the computer for paddle control and other game applications. This socket, located at J-14, has outputs available as well. With an appropriate machine language program, these output lines may be used to serialize data in a format suitable for a teletype. A suitable interface circuit must be built since the outputs are merely LSTTL and won't run a teletype without help. Several interface circuits are discussed below and the user may pick the one best suited to his needs.

The ASR - 33 Teletype

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

- a) The yellow wire and purple wire should both go to terminal 9 of Terminal Strip X. If the purple wire is going to terminal 8, then remove it and relocate it at terminal 9. This is necessary to change from the 60mA current loop to the 20mA current loop.
- b) Above Terminal Strip X is a connector socket identified as "2". Pin 8 is the input line + or high; Pin 7 is the input line or low. This connector mates with a Molex receptacle model 1375 #Ø3-Ø9-2151 or #03-09-2153. Recommended terminals are Molex #Ø2-Ø9-2136. An alternate connection method is via spade lugs to Terminal Strip X, terminal 7 (the + input line) and 6 (the input line).
- c) The following circuit can be built on a 16 pin DIP component carrier and then plugged into the Apple's l6 pin socket found at J-14: (The junction of the 3.3k resistor and the transistor base lead is floating). Pins 16 and 9 are used as tie points as they are unconnected on the Apple board. (Figure la).

The "RS - 232 Interface"

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

### A Serial Out Machine Center Language Program

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

Entering Machine Language Program

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

Executing this Program

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

- 2. PR#Ø will inactivate the printer transfering control back to the Video monitor as the primary output device.
- 3. In Monitor mode \$37ØG activates the printer and hitting the "RESET" key exits the program.

Saving the Machine Language Program

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

The way it is saved is as follows:

- 1. Insert a blank program cassette into the tape recorder and rewind it.
- Hit the "RESET" key. The system should move into Monitor mode. An asterick "\*" and flashing cursor should appear on the left-hand side of the screen.
- 3. Type in "370.03E9W 370.03E9W".
- Start the tape recorder in record mode and depress the "RETURN" key.
- 5. When the program has been written to tape, the asterick and flashing cursor will reappear.

The Program

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

- 1. B<sup>C</sup> (control B) into BASIC
- 2. Turn the teletype (printer on)
- 3. Type in the following
  - 10 CALL 880
  - 15 PRINT "ABCD...XYZØ1123456789"
  - 2Ø PR#Ø
  - 25 END
- Type in RUN and hit the "RETURN" key. The text in line 15 should be printed on the teletype and control is returned to the keyboard and Video monitor

Line 10 activates the teletype machine routine and all "PRINT" statements following it will be printed to the teletype until a PR#0 statement is encountered. Then the text in line 15 will appear on the teletype's output. Line 20 deactivates the printer and the program ends on line 25.

### Conclusion

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









FIGURE 2 RS-232

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

			1	T	ITLE	' TEL	ETYPE	E DRIVER	ROU	TINES'
			2	**:	*****	***	*****	******	**	
			3	*					*	
			4	*		TTY	YDRIVE	ER:	*	
			5	*	TEI	LETY	YPE OL	JTPUT	*	
			6	*	RO	UTIN	NE FOR	72	*	
			7	*	CO	LUM	V PRI	NT WITH	*	
			8	*	BAS	SIC	LIST		*	
			9	*					*	
			10	*	COPY	RIG	HT 19	77 BY:	*	
			11	*	APPLI	E CO	OMPUTE	ER INC.	*	
			12	*		11,	/18/7	7	*	
			13	*					*	
			14	*	R	• W1	IGGIN	ron	*	
			15	*	S	• W(	DZNIA	<	*	
			16	*					*	
			17	**:	*****	***	*****	******	**	
			18	WNI	DWDTH		EQU	\$21		FOR APPLE-II
			19	СН			EQU	\$24		CURSOR HORIZ.
			20	CSI	WL		EQU	\$36		CHAR. OUT SWITCH
			21	YSI	AVE		EQU	\$778		
			22	COI	_CNT		EQU	\$7F8		;COLUMN COUNT LOC.
			23	MAI	ЯK		EQU	\$C058		
			24	SP	ACE		EQU	\$C059		
			25	WA	IT		EQU	\$FCA8		
			26				ORG	\$370		
***WAR	NIN	G: OP	ERAND	OVE	RFLOW	IN	LINE	27		
0370:	A9	82	27	TT	INIT:		LDA	#TTOUT		
0372:	85	36	28				STA	CSWL		POINT TO TTY ROUTINES
0374:	A9	Ø3	29				LDA	#TTOUT/	256	;HIGH BYTE
0376:	85	37	3Ø				STA	CSWL+1		
Ø378:	A9	48	31				LDA	#72		SET WINDOW WIDTH
Ø37A:	85	21	32				STA	WNDWDTH		; TO NUMBER COLUMNS ONT
Ø37C:	A5	24	33				LDA	СН		
Ø37E:	8D	F8	34				STA	COLCNT		;WHERE WE ARE NOW.
0381:	60		35				RTS			
0382:	48		36	TT(	50T :		РНА			SAVE TWICE
0383:	48	-	37	PH	e e e e e e e e e e e e e e e e e e e					JON STACK.
0384:	AD	18	38	11(	0012:		LDA	COLONT		CHECK FOR A TAB.
03871	05	24	39				CMP	CH		ADDECTODE OUTDUT OUAD
03891	68	<i>a</i> 0	40				PLA	<b>TECTOTO</b>		JRESIDRE DUIPUI CHAR.
030H ·	BU 40	03	41				BUS	IESICIRI	L	JIF C SETJ NU TAB
0300.	40	<u>^</u>	46					# <b>6</b> \ \ \ \		DDINT A SDACE
0300.	H7 00	AU CO	43	TE	5 TO TO	•	LUA	#JAU DTC1		TRICK TO DETERMINE
0301 •	20	00 Ø3	44	I E.	SIGIRI	L •	DII	RIJI		JELCONTROL CHAR
0372.	7 U	03 F0	43				INC	COLONT		IF NOT, ADD ONE TO CM
03974	00	r 0 C 1	40	DD				DOCHAR		PRINT THE CHAR ON TTY
03941	68	01	~ / ⁄\X	ΓKI	*1 1 1 *			DOONAK		PRESTORE CHAP
03931	<u>78</u>		40				PHA			AND PHT BACK ON STAC
03901	90	F.6	50				BCC	ττομτο		DO MORE SPACES FOR TA
039F:	49	00	51				EOR	#\$OD		CHECK FOR CAR RET.
0340:	0A		52				ASL	A		ELIM PARITY
Ø3A1:	DO	OD	53				BNE	FINISH		JIF NOT CR. DONE.

					TELETYP	E DRI	VER ROUTINES	
3:42 P	• M • 3	11	1/18	3/1977	7			PAGE: 2
Ø3A3:	8D	F8	Ø7	54		STA	COLCNT	CLEAR COLUMN COUNT
Ø3A6:	A9	8A		55		LDA	#\$8A	SNOW DO LINE FEED
Ø3A8:	20	C 1	Ø3	56		JSR	DOCHAR	
Ø3AB:	A9	58		57		LDA	#\$53	
Ø3AD:	20	A8	FC	58		JSR	WAIT	3200MSEC DELAY FOR LIB
Ø3BØ:	AD	F8	Ø7	59	FINISH:	LDA	COLCNT	CHECK IF IN MARGIN
Ø3B3:	FØ	Ø8		60		3EØ	SETCH	FOR CR, RESET CH
Ø385:	E5	21		61		\$3C	WVDWDTH	JIF SO, CARRY SET.
Ø3B7:	E9	F7		62		SSC	#\$F7	
Ø389:	<b>9</b> Ø	Ø4		63		BCC	RETURN	
Ø388:	69	1 F		64		ADC	#\$1F	JADJUST CH
Ø3BD:	85	24		65	SETCH:	STA	СН	
Ø38F:	68			66	RETURN:	PLA		
Ø3CØ:	60			67	RTS1:	RTS		RETURN TO CALLER
				68	* HERE IS	THE T	ELETYPE PRINT	A CHARACTER ROUTINE:
Ø3C1:	8C	78	Ø7	69	DOCHAR:	STY	YSAVE	
Ø3C4:	Ø8			70		PHP		SAVE STATUS.
Ø3C5:	AØ	ØВ		71		LDY	#\$0B	;11 BITS (1 START, XXXXXXXXXXXXXXX
Ø3C7:	18			72		CLC		BEGIN 7ITH SPACE (STXXXXXX
0308:	48			73	TTOUT3:	PHA		SAVE A REG AND SET FOR
Ø3C9:	ВØ	Ø5		74		BCS	MARKOUT	
Ø3CB:	AD	59	CØ	75		LDA	SPACE	SEND A SPACE
Ø3CE:	<b>9</b> Ø	ØЗ		76		BCC	TTOUT4	
03D0:	AD	58	CØ	77	MARKOUT:	LDA	MARK	SEND A MARK
Ø3D3:	A9	D7		78	TTOUT4:	LDA	#\$D7	JDELAY 9.091 MSEC FOR
Ø3D5:	48			7 <b>9</b>	DLY1:	PHA		3110 BAUD
Ø3D6:	A9	20		80		LDA	#\$20	
Ø3D8:	4A			81	DLY2:	LSR	Α	
Ø3DB:	<b>9</b> Ø	FD		82		BCC	DLY2	
Ø3DB:	68			83		PLA		
Ø3DC:	E9	Øl		84		SBC	# <b>\$</b> Ø1	
Ø3DE:	DØ	F 5		85		3NE	DLY1	
Ø3EØ:	68			86		PLA		
Ø3E1:	6A			87		ROR	Α	;NEXT BIT (STOP BITS R
Ø3E2:	88			88		DEY		LOOP 11 BITS.
Ø3E3:	DØ	E3		89		BNE	TTOUT3	
Ø3E5:	AC	78	Ø7	<b>9</b> Ø		LDY	YSAVE	RESTORE Y-REG.
Ø3E8:	28			91		PLP		RESTORE STATUS
Ø3E9:	60			92		RTS		;RETURN
*****	**SL	JCCE	ESSF	TUL AS	SSEMBLY: NO	ERRO	RS	

FIGURE 3b

CROSS-REF	ERNCE:	TELETYF	PEDR	IVER	ROUTIN	NES
СН	0024	0033	ØØ39	0065		
COLCNT	Ø718	0034	ØØ38	0046	0054	ØØ59
C5WL	0036	0028	ØØ3Ø			
DLYI	0305	0085				
DLY2	0308	0082				
DOCHAR	0301	0047	0056			
FINISH	0330	0053				
MARK	C058	0077				
MARKOUT	0300	0074				
PRNTIT	Ø397	0045				
RETURN	Ø38F	0063				
RTS1	0300	0044				
SETCH	0330	0060				
SPACE	C059	0075				
TESTCTRL	Ø33F	0041				
TTINIT	0370					
TTOUT	Ø332	0027	ØØ29			
TTOUT2	Ø384	0050				
TTOUT3	Ø3C8	ØØ89				
TTOUT4	0303	0076				
WAIT	FCAB	0058				
WNDWDTH	0021	ØØ32	0061			
YSAVE	Ø778	ØØ69	ØØ <b>9</b> Ø			
ILE:						

FIGURE 3c

### INTERFACING THE APPLE

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

### TABLE OF CONTENTS

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



Figure 1A APPLE II Board-Complete View



### CASSETTE JACKS

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

<u>CASSETTE DATA IN JACK</u>: Designed for connection to the "EARPHONE" or "MONITOR" output found on most audio cassette tape recorders. V<sub>IN</sub>=1Vpp (nominal), Z<sub>IN</sub>=12K Ohms. Located at K12 as illustrated in Figure

CASSETTE DATA OUT JACK: Designed for connection to the "MIC" or "MICROPHONE" input found on most audio cassette tape recorders. V<sub>OUT</sub> =25 mV into 100 Ohms, Z<sub>OUT</sub> =100 Ohms. Located at K13 as illustrated in In Figure 1.

### GAME I/O CONNECTOR

The Game I/O Connector provides a means for connecting paddle controls, lights and switches to the APPLE II for use in controlling video games, etc. It is a 16 pin IC socket located at Jl4 and is illustrated in Figure 1 and 2.

Figure 2

### GAME I/O CONNECTOR

( Front Edge of PC Board )

+5V	1	•	16	N.C.
SWO	2		15	ANO
SW1	3		14	AN1
SW2	4		13	AN2
CO40 STB	5		12	AN3
PDLO	6		11	PDL3
PDL2	7		10	PDL1
GND	8		9	N.C.

LOCATION J14

- 8 addresses (CØ58-CØ5F) are assigned to selectively ANO-AN3: "SET" or "CLEAR" these four "ANNUNCIATOR" outputs. Envisioned to control indicator lights, each is a 74LSxx series TTL output and must be buffered if used to drive lamps.
- CØ4Ø STB: A utility strobe output. Will go low during  $\emptyset_2$  of a read or write cycle to addresses CØ40-CØ4F. This is a 74LSxx series TTL output.
- GND: System circuit ground. O Volt line from power supply.
- NC: No connection.
- PDLØ-PDL3: Paddle control inputs. Requires a Ø-15ØK ohm variable resistance and +5V for each paddle. Internal 100 ohm resistors are provided in series with external pot to prevent excess current if pot goes completely to zero ohms.
- Switch inputs. Testable by reading from addresses SWØ-SW2: CØ61-CØ63 (or CØ69-CØ6B). These are uncommitted 74LSxx series inputs.
- Positive 5-Volt supply. To avoid burning out the connector +5V: pin, current drain MUST be less than 100mA.

### **KEYBOARD CONNECTOR**

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

Figure 3

### **KEYBOARD CONNECTOR**

TOP VIEW 

	01 1121	•	
( Front E	dge of	PC Board	)
+5V 1	•	16 N.C.	
STROBE 2		15 -12V	
RESET 3		14 N.C.	
N.C. 4		13 B2	
B6 5		12 B1	
B5 6		11 B4	
B7 7		10 B3	
GND 8		9 N.C.	

LOCATION A7

### SIGNAL DESCRIPTION FOR KEYBOARD INTERFACE

- <u>Bl-B7</u>: 7 bit ASCII data from keyboard, positive logic (high level= "1"), TTL logic levels expected.
- <u>GND</u>: System circuit ground. Ø Volt line from power supply.
- NC: No connection.
- <u>RESET</u>: System reset input. Requires switch closure to ground.
- <u>STROBE</u>: Strobe output from keyboard. The APPLE II recognizes the positive going edge of the incoming strobe.
- <u>+5V</u>: Positive 5-Volt supply. To avoid burning out the connector pin, current drain <u>MUST</u> be less than 100mA.
- <u>-12V</u>: Negative 12-Volt supply. Keyboard should draw less than 50mA.

### PERIPHERAL CONNECTORS

The eight Peripheral Connectors mounted near the back edge of the APPLE II board provide a convenient means of connecting expansion hardware and peripheral devices to the APPLE II I/O Bus. These are Winchester #2HW25CQ-111 (or equivalent) 50 pin card edge connectors with pins on  $.10^{"}$  centers. Location and pin outs are illustrated in Figures 1 and 4.

### SIGNAL DESCRIPTION FOR PERIPHERAL I/O

- AØ-A15: 16 bit system address bus. Addresses are set up by the 6502 within 300nS after the beginning of  $\emptyset_1$ . These lines will drive up to a total of 16 standard TTL loads.
- <u>DEVICE SELECT</u>: Sixteen addresses are set aside for each peripheral connector. A read or write to such an address will send pin 41 on the selected connector low during  $\emptyset_2$  (500nS). Each will drive 4 standard TTL loads.
- <u>DØ-D7:</u> 8 bit system data bus. During a write cycle data is set up by the 65Ø2 less than 3ØØnS after the beginning of  $\emptyset_2$ . During a read cycle the 65Ø2 expects data to be ready no less than 1ØØnS before the end of  $\emptyset_2$ . These lines will drive up to a total of 8 total low power schottky TTL loads.

- <u>DMA</u>: Direct Memory Access control output. This line has a 3K Ohm pullup to +5V and should be driven with an open collector output.
- <u>DMA IN</u>: Direct Memory Access daisy chain input from higher priority peripheral devices. Will present no more than 4 standard TTL loads to the driving device.
- <u>DMA OUT</u>: Direct Memory Access daisy chain output to lower priority peripheral devices. This line will drive 4 standard TTL loads.
- GND: System circuit ground. Ø Volt line from power supply.
- <u>INH</u>: Inhibit Line. When a device pulls this line low, all ROM's on board are disabled (Hex addressed DØØØ through FFFF). This line has a 3K Ohm pullup to +5V and should be driven with an open collector output.
- <u>INT IN</u>: Interrupt daisy chain input from higher priority peripheral devices. Will present no more than 4 standard TTL loads to the driving device.
- INT OUT: Interrupt daisy chain output to lower priority peripheral devices. This line will drive 4 standard TTL loads.
- <u>I/O SELECT</u>: 256 addresses are set aside for each peripheral connector (see address map in "MEMORY" section). A read or write of such an address will send pin 1 on the selected connector low during  $\emptyset_2$  (500nS). This line will drive 4 standard TTL loads.
- <u>I/O STROBE</u>: Pin 20 on all peripheral connectors will go low during  $\emptyset_2$ , of a read or write to any address C8ØØ-CFFF. This line will drive a total of 4 standard TTL loads.
- <u>IRQ</u>: Interrupt request line to the 6502. This line has a 3K Ohm pullup to +5V and should be driven with an open collector output. It is active low.
- NC: No connection.
- <u>NMI</u>: Non Maskable Interrupt request line to the 6502. This line has a 3K Ohm pullup to +5V and should be driven with an open collector output. It is active low.
- <u>Q</u><sub>3</sub>: A 1MHz (nonsymmetrical) general purpose timing signal. Will drive up to a total of 16 standard TTL loads.
- <u>RDY</u>: "Ready" line to the 65 $\emptyset$ 2. This line should change only during  $\emptyset_1$ , and when low will halt the microprocessor at the next READ cycle. This line has a 3K Ohm pullup to +5V and should be driven with an open collector output.
- <u>RES</u>: Reset line from "RESET" key on keyboard. Active low. Will drive 2 MOS loads per Peripheral Connector.

- <u>R/W:</u> READ/WRITE line from 6502. When high indicates that a read cycle is in progress, and when low that a write cycle is in progress. This line will drive up to a total of 16 standard TTL loads.
- USER 1: The function of this line will be described in a later document.
- <u>Ø\_0:</u> Microprocessor phase V clock. Will drive up to a total of 16 standard TTL loads.
- $\underline{\emptyset}_1: \qquad \text{Phase 1 clock, complement of } \emptyset_0. \text{ Will drive up to a total of 16 standard TTL loads. }$
- <u>7M:</u> Seven MHz high frequency clock. Will drive up to a total of 16 standard TTL loads.
- +12V: Positive 12-Volt supply.
- +5V: Positive 5-Volt supply
- -5V: Negative 5-Volt supply.
- -12V: Negative 12-Volt supply.

### POWER CONNECTOR

The four voltages required by the APPLE II are supplied via this AMP #9-35028-1,6 pin connector. See location and pin out in Figures 1 and 5.

### PIN DESCRIPTION

- GND:(2 pins) system circuit ground. Ø Volt line from power<br/>supply.+12V:Positive 12-Volt line from power supply.+5V:Positive 5-Volt line from power supply.-5V:Negative 5-Volt line from power supply.
- -12V: Negative 5-Volt line from power supply.







LOCATION K1

### SPEAKER CONNECTOR

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

### SIGNAL DESCRIPTION FOR SPEAKER

- <u>+5V</u>: System +5 Volts
- <u>SPKR</u>: Output line to speaker. Will deliver about .5 watt into 8 Ohms.



### VIDEO OUTPUT JACK

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

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

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

See locations illustrated in Figure 1.

### AUXILIARY VIDEO OUTPUT CONNECTOR

This is a MOLEX KK 100 series connector with four .25" square pins on .10" centers. It provides composite video and two power supply voltages. Video out on this connector is not adjustable by the on board 200 Ohm trim pot. See Figures 1 and 7.

### SIGNAL DESCRIPTION

<u>GND</u>: System circuit ground. Ø Volt line from power supply.

- <u>VIDEO</u>: NTSC compatible positive composite VIDEO. DC coupled emitter follower output (not short circuit protected). SYNC TIP is Ø Volts, black level is about .75 Volts, and white level is about 2.0 Volts into 470 Ohms. Output level is non-adjustable.
- +12V: +12 Volt line from power supply.
- +5V: -5 Volt line from power supply.



LOCATION J14B

### INSTALLING YOUR OWN RAM

### THE POSSIBILITIES

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

### RESTRICTIONS

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

- 1. All sockets in a row must have the same type (4K or 16K) RAMs.
- 2. There MUST be RAM assigned to the zero block of addresses.

### ASSIGNING RAM

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

Figure 8

### MEMORY SELECT SOCKETS TOP VIEW

### PINOUT

(0000-OFFF) 4K "0" BLOCK1	•	14	RAM ROW	С		
(1000-1FFF) 4K "1" BLOCK 2		13	RAM ROW	D		
(2000-2FFF) 4K "2" BLOCK 3	· · · · · · · · · · · · · · · · · · ·	12	RAM ROW	Е		
(3000-3FFF) 4K "3" BLOCK 4	· · · · · ·	11	N.C.			
(4000-4FFF) 4K "4" BLOCK 5	· · · · · ·	10	16K "0" BL	.OCK	(0000-3	FFF)
(5000-5FFF) 4K "5" BLOCK 6		9	16K "4" BL	.OCK	(4000-7	FFF)
(6000-EFFF) 4K "6" BLOCK7		8	16K "8" BL	OCK	(8000-B	FFF)

LOCATIONS D1, E1, F1

### MEMORY

TABLE OF CONTENTS

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

### INTRODUCTION

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

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

A set of three each of one of the above is supplied with the board. Type 1 is supplied with 4K or 8K systems. Both type 1 and 2 are supplied with 12K systems. Type 1 is a contiguous memory range for maximum BASIC program size. Type 2 is non-contiguous and allows 8K dedicated to HIRES screen memory with approximately 2K of user BASIC space. Type 3 is supplied with 16K, 20K and 24K systems. Type 4 with 30K and 36K systems and type 5 with 48K systems.

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

The 6502 microprocessor generates a 16 bit address, which allows 65536 (commonly called 65K) different memory locations to be specified. For convenience we represent each 16 bit (binary) address as a 4-digit hexadecimal number. Hexadecimal notation (hex) is explained in the Monitor section of this manual.

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

### MEMORY SELECT SOCKETS

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

### HOW TO USE

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

Let us learn by example:

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

If in addition you have inserted 4K RAMs into rows "D" and "E", and you want them each to occupy the first 4K addresses starting at 4000 and 5000 respectively, jumper pin 13 to pin 5 (thereby assigning row "D" to the 4000-4FFF range of memory), and jumper pin 12 to pin 6 (thereby assigning row "E" to the 5000-5FFF range of memory). Remember to jumper all three MEMORY SELECT sockets the same.

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

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

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

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

### Memory Address Allocations in 4K Bytes

0000	text and color graphics display pages, 6502 stack, pointers, etc.	8000	
1000	-	9000	-
2000	high res graphics display primary page	A000	
3000		B000	
4000	high res. graphics display secondary page	C000	addresses dedicated to hardware functions
5000		D000	ROM socket D0: spare " ROM socket D8: spare
6000		E000	ROM socket EO: BASIC
7000	_	F000	ROM socket E8: BASIC " ROM socket F0: BASIC utility
			ROM socket F8; monitor

Memory Map Pages Ø to BFF

USED		CONVENTS	
BA	USED FOR	CURRENTS	
UTILITY	register area for "sweet 16" 16 bit firmware processor.		
MONITOR			
MONITOR	holds a 16 bit number that is randomized with each key entry.		
UTILITY	integer multiply and divide work space.		
BASIC			
UTILITY	floating point work space.		
6502	subroutine return stack.		
	character input buffer.		
MONITOR	Y (control Y) will cause a <sup>c</sup> JSR to this location.		
	NMI's are vectored to this location.		
	IRQ's are vectored to the address pointed to by these locations.		
DISPLAY	text or color graphics primary page.		
DISPLAY	text or color graphics secondary page.	BASIC initializes LOMEM to location 0800.	
	USED BY UTILITY MONITOR MONITOR UTILITY BASIC UTILITY 6502 MONITOR DISPLAY DISPLAY	USED BYUSED FORUTILITYregister area for "sweet 16" 16 bit firmware processor.MONITORholds a 16 bit number that is randomized with each key entry.MONITORholds a 16 bit number that is randomized with each key entry.UTILITYinteger multiply and divide work space.BASICInteger multiply and divide work space.G502subroutine return stack. character input buffer.WONITORY (control Y) will cause a CJSR to this location. NMI's are vectored to this location.NONITORY text or color graphics primary page.DISPLAYtext or color graphics secondary page.	
HEX ADDRESS	ASSIGNED FUNCTION	COMMENTS	
----------------	-----------------------------------	---	--
COOX	Keyboard input.	Keyboard strobe appears in bit 7. ASCII data from keyboard appears in the 7 lower bits.	
CO1X	Clear keyboard strobe.		
C02X	Toggle cassette output.		
C03X	Toggle speaker output.	Output strobe to Game I/O connector.	
C04X	" $\overline{C040 \text{ STB}}$ "		
C050	Set graphics mode		
C051	" text "		
C052	Set bottom 4 lines graphics		
C053	" " " text		
C054	Display primary page		
C055	" secondary page		
C056	Set high res. graphics		
C057	" color "		
C058	Clear "ANO"	Annunciator 0 output to Game I/O connector	
C059	Set "	dame 170 connector.	
C05A	Clear "AN1"	Annunciator 1 output to Game I/O connector.	
C05B	Set "		
C05C	Clear "AN2"	Annunciator 2 output to $Game I/O$ connector	
C05D	Set "		
C05E	Clear "AN3"	Annunciator 3 output to Game I/O connector.	
C05F	Set "		

HEX ADDRESS	ASSIGNED FUNCTION	COMMENTS
C060/8	Cassette input	State of "Cassette Data In" appears in bit 7
C061/9	"SW1"	input on State of Switch 1 ∧ Game I/O connector appears in bit 7.
C062/A	"SW2"	State of Switch 2 input on Game I/O connector appears in bit 7.
C063/B	"SW3"	State of Switch 3 input on Game I/O connector appears in bit 7.
C064/C	Paddle 0 timer output	State of timer output for Paddle 0 appears in bit 7.
C065/D	"1""	State of timer output for Paddle 1 appears in bit 7.
C066/E	" $2$ ""	State of timer output for Paddle 2 appears in bit 7.
C067/F	" 3 " "	State of timer output for Paddle 3 appears in bit 7.
C07X	"PDL STB"	Triggers paddle timers during ${                                   $
C08X	DEVICE SELECT 0	Pin 41 on the selected
C09X	" 1	Peripheral Connector goes low during $\phi_2^{}.$
COAX	" 2	
COBX	" 3	
COCX	" 4	
CODX	" 5	
COEX	" 6	
COFX	" 7	
C10X	" 8	Expansion connectors.
C11X	" 9	"
C12X	" A	"

HEX ADDRESS	ASSIGNED FUN	CTION	COMMENTS
C13X	DEVICE SELECT	В	n
C14X	"	С	"
C15X	"	D	"
C16X	"	Е	"
C17X	"	F	n
C1XX	I/O SELECT	1	Pin 1 on the selected
C2XX	"	2	Peripheral Connector goes low during $\phi_2$ .
СЗХХ	"	3	NOTES:
C4XX	"	4	0 does not get this
C5XX	"	5	$\frac{1}{1}$ SIgnal
C6XX	"	6	2. 1/0 SELECT I uses the same addresses as
C7XX	"	7	DEVICE SELECI 8-F
C8XX	"	8, $\overline{I/O \text{ STROBE}}$	Expansion connectors.
C9XX	"	9, "	
CAXX	"	Α, "	
CBXX	"	Β, "	
CCXX	"	C, "	
CDXX	"	D, "	
CEXX	"	Ε, "	
CFXX	"	F, "	
D000-D7FF	ROM socket DO		Spare.
D800-DFFF	" " D8		Spare.
E000-E7FF	" " EO		BASIC.
E800-DFFF	" " E8		BASIC.
F000-F7FF	" " F0		1K of BASIC, 1K of utility.
F800-FFFF	" " F8		Monitor
E000-E7FF E800-DFFF F000-F7FF F800-FFFF	" " E0 " " E8 " " F0 " " F8		BASIC. BASIC. 1K of BASIC, 1K of utility. Monitor

## SYSTEM TIMING

## SIGNAL DESCRIPTIONS

- 14M: Master oscillator output, 14.318 MHz +/- 35 ppm. All other timing signals are derived from this one.
- 7M: Intermediate timing signal, 7.159 MHz.

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

- $\emptyset_0$ : Phase  $\emptyset$  clock to microprocessor, 1. $\emptyset$ 23 MHz nominal.
- $\underline{\emptyset}_1$ : Microprocessor phase l clock, complement of  $\underline{\emptyset}_0$ , l.023 Mhz nominal.
- $\underline{\emptyset_2}$ : Same as  $\underline{\emptyset_0}$ . Included here because the 65 $\underline{\emptyset}$ 2 hardware and programming manuals use the designation  $\underline{\emptyset_2}$  instead of  $\underline{\emptyset_0}$ .
- <u>Q3</u>: A general purpose timing signal which occurs at the same rate as the microprocessor clocks but is nonsymmetrical.

## MICROPROCESSOR OPERATIONS

- <u>ADDRESS</u>: The address from the microprocessor changes during  $\emptyset_1$ , and is stable about 300nS after the start of  $\emptyset_1$ .
- <u>DATA WRITE</u>: During a write cycle, data from the microprocessor appears on the data bus during  $\emptyset_2$ , and is stable about  $3\emptyset$  on S after the start of  $\emptyset_2$ .
- <u>DATA READ</u>: During a read cycle, the microprocessor will expect data to appear on the data bus no less than 100nS prior to the end of  $\emptyset_2$ .

SYSTEM TIMING DIAGRAM

TIMING CIRCUITRY BLOCK DIAGRAM

TIMING RELATIONSHIPS





FIGURE S-1 APPLE II SYSTEM DIAGRAM



FIGURE S-2 MPU AND SYSTEM BUS







FIGURE S-4 SYNC COUNTER











FIGURE S-7 RAM ADDRESS MUX



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



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



FIGURE S-10 ON-BOARD I/O



FIGURE S-11 VIDEO GENERATOR



khaibitgfx@gmail.com







10260 Bandley Drive Cupertino, California 95014 (408) 996-1010

Steven Jobs Vice President, New Product Development

























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