

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

Any donations for better software and hardware, what I have is old and slow, can be made to marketplace@seaside.ns.ca, Thanks!



Current Donators to this project are: Dan Chisarick





Apple I Reference Manual

January 1978

leference Manual

Apple II Reference Manual

January 1978

Manual Reference

APPLE II Reference Manual

January 1978

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

APPLE II Reference Manual **TABLE OF CONTENTS**

A. GETTING STARTED WITH YOUR

APPLE II	1
1. Unpacking	1
2. Warranty Registration Card	1
3. Check for Shipping Damage	2
4. Power Up	2
5. APPLE II Speaks Several Languages	2
6. APPLE Integer BASIC	3
7. Running Your First	
and Second Programs	3
8. Running 16K Startrek	3
9. Loading a Program Tape	4
10. Breakout and Color Demos Tapes	6
11. Breakout and Color	
Demos Program Listings	12
12. How to Play Startrek	14
13. Loading HIRES Demo Tape	15
B. APPLE II INTEGER BASIC	17
1. BASIC Commands	18
2. BASIC Operators	19
3. BASIC Functions	22
4. BASIC Statements	23
5. Special Control and Editing	28
6. Table A- Graphics Colors	29
7. Special Controls and Features	30
8. BASIC Error Messages	32
9. Simplified Memory Map	33
10. Data Read/Save Subroutines	34
11. Simple Tone Subroutines	43
12. High Resolution Graphics Subroutines and Listings	46

13. Additional BASIC Program	
Examples	55
a. Rod's Color Pattern (4K)	55
b. Pong (4K)	56
c. Color Sketch (4K)	57
d. Mastermind (8K)	59
e. Biorhythm (4K)	61
f. Dragon Maze (4K)	63
C. APPLE II FIRMWARE	67
1. System Monitor Commands	68
2. Control and Editing Characters	72
3. Special Controls and Features	74
4. Annotated Monitor and	
Dis-assembler Listing	76
5. Binary Floating Point Package	94
6. Sweet 16 Interpreter Listing	96
7. 6502 Op Codes	100
D. APPLE II HARDWARE	106
1. Getting Started with Your APPLE II Board	107 110
2. APPLE II Switching Power Supply	112
3. Interfacing with the Home TV	114
4. Simple Serial Output	
5. Interfacing the APPLE -	
Signals, Loading, Pin Connections	122
6. Memory -	
Options, Expansion, Map,	100
Address	133
7. System Timing	140
8. Schematics	141

Unpacking

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

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

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

Warranty Registration Card

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

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.
- 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.
- 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.
- 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.

7

Memory Size too Small

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

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

Cassette Recorder Inadjustment

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

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

Often times new cassette recorders will not need this adjustment.

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

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

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

Dealing with the Loading Problem

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

- Check the program tape for its memory requirements. Be sure that you have a large enough system.
- 2. Before loading a program reset the memory pointers with the B_C (control B) command.
- 3. In special cases have the tape head azimuth checked and adjusted.
- 4. Check the program tape by listening to it.
 - a) Replace it if it is defective, or
 - b) start it at the beginning of the program.

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

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

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

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.

BREAKOUT GAME PROGRAM LISTING

PROGRAM LISTING

5 GOTO 15

- 10 Q=(PDL (0)-20)/6: IF Q<0 THEN Q=0: IF Q>=34 THEN Q=34: COLOR= D: YLIN Q.Q+5 AT 0: COLOR=A: IF P>Q THEN 175: IF Q THEN VLIN 0.0-1 AT 0:P=0:RETURN
- 15 DIM A\$(15).B\$(10):A=1:B=13: C=9:D=6:E=15: TEXT : CALL -936: YTAB 4: TAB 10: PRINT **** BREAKOUT ****:PRINT
- 28 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): ¥LIN 0,39 AT I
- 30 NEXT I: POKE 34,20: PRINT : PRINT : PRINT : FOR I=0 TO 15: ¥TAB 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' 88 V=-V ::GOSUB 95:B=E: PRINT 'ODD BRIC K':: GOSUB 95:C=E: PRINT 'PADDLE ';: GOSUB 95:D=E: PRINT 'BALL' ::GOSUB 95
- 40 POKE 34,20: COLOR=A: FOR I= 0 TO 39: YLIN 0.39 AT I: NEXT I: FOR I=20 TO 34 STEP 2: TAB I+1: PRINT I/2-9:: COLOR=8: YLIN 0,39 AT I: COLOR=C: FOR J=I MOD 4 TO 39 STEP 4

- 45 YLIN J,J+1 AT I: NEXT J, I: TAB 180 IF N THEN Y= ABS (Y): YLIN 5: PRINT 'SCORE =0':PRINT : PRINT : POKE 34,21:S=0:P= S:L=S:X=10:Y=10:L=6
- 50 COLOR=A: PLOT X,Y/3:X=19:Y= RND (128):V=-1:W= RND (5)-2:L=L-1: IF L<1 THEN 120: TAB 6: IF L>1 THEN PRINT L:"BALLS L EFT
- 55 IF L=1 THEN PRINT 'LAST BALL, ' ;A\$: PRINT : FOR I=1 TO 100 : GOSUB 10: NEXT I:M=1:N=0
- 60 J=Y+W: IF J≻=0 AND J≺120 THEN 65:₩=-₩:J=Y: FOR I-1 TO 6:K= PEEK (-16336): NEXT I
- 65 I-X+V: IF I<0 THEN 180: GOSUB 178: COLOR=A:K=J/3: IF I>39 THEN 75: IF SCRH(I.K)=A THEN 85: IF I THEN 100:N=N+1:V=(N>5)+1:W=(K-P)*2-5:M=1
- 70 Z= PEEK (-16336)-PEEK (-16336)+ PEEK (-16336)- PEEK (-16336)+ PEEK (-16336)- PEEK (-16336)+ PEEK (-16336): GOTO 85
- 75 FOR I=1 TO 6:M= PEEK (−16336): NEXT I:I=X:M=0
- 85 PLOT X,Y/3: COLOR=E: PLOT I, K:X=I:Y=J: 60TO 60
- 98 PRINT 'INVALID, REENTER'; 95 INPUT ' COLOR (0, TO 15)',E: IF E<8 OR E>15 THEN 98: RETURN
- K/2*2,K/2*2+1 AT I:S=S+I/2-9: YTAB 21: TAB 13: PRING S 105 Q= PEEK (-16336)- PEEK (-16336)+ PEEK (-16336)- PEEK (-16336)+ PEEK (-16336)- PEEK (-16336))+ PEEK (-16336)- PEEK (-16336))+ PEEK (-16336)- PEEK (-16336)) 110 IF S<720 THEN 80 115 PRINT "CONGRATULATONS. ":A\$;' YOU WIN!': GOTO 165 120 PRINT YOUR SCORE OF ':S:' IS ' ;: 60TO 125+(S∕100)*5 125 PRINT "TERRIBLE!": GOTO 165 130 PRINT 'LOUSY.': GOTO 165 135 PRINT "POOR.": GOTO 165 140 PRINT 'GOOD.': GOTO 165 145 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: YTAB 10: TAB 10: PRINT 'GAME OV ER": END 170 Q=(PDL (0)-20)/6: IF Q<0 THEN 0=0: IF 0>=34 THEN 0=34: COLOR= D: VLIN 0,0+5 AT 0: COLOR=A: IF P>0 THEN 175: IF 0 THEN YLIN 0,0-1 AT 0:P=0: RETURN 175 IF P=0 THEN RETURN : IF 0*84
 - THEN YLIN 0+6,39 AT 0:P=0: RETURH
 - 180 FOR I=1 TO 80:0= PEEK (-16336): NEXT I: GOTO 50

PROGRAM LISTING

- 10 DIM C(4): 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
- 20 POKE 13,4: POKE 14,198: POKE 15,24: POKE 16,240: POKE 17 ,5: POKE 18,198: POKE 19,1: POKE 28,76: POKE 21,2: POKE 22.0: POKE 23,96
- 30 TEXT : CALL -936: VTAB 4: TAB 8: PRINT *4K COLOR DEMOS*: PRINT : PRINT *1 LINES*: PRINT *2 CROS S": PRINT "3 WEAVING"
- 48 PRINT *4 TUNNEL*: PRINT *5 CIRCL 580 2=20: GOTO 980 E*: PRINT *6 SPIRAL ***: PRINT 600 COLOR= RND (16): FOR I=0 TO "7 TONES ** ": PRINT "8 SPRING"
- 58 PRINT "9 HYPERBOLA": PRINT *10 COLOR BARS*: PRINT : PRINT *** NEEDS PDL(0) CONNECTED* : PRINT
- 60 PRINT "HIT ANY KEY FOR HEW DEMO" :Z=0: PRINT : INPUT "WHICH DENO \$ ".I: GR : IF I)0 AND I(11 THEN GOTO 100+1: GOTO 30
- 70 INPUT "WHICH DEMO WOULD YOU LIKE ",I: GR : IF I AND IK20 THEN GOTO 109*1: GOTO 30
- 108 I=1+I MOD 79: J=1+(1)39)*(79 -I-I): GOSUB 2000: GOSUB 10000 : GOTO 100
- 200 I=1+I MOD 39:J=I: GOSUB 2000 :J=39-1: GOSUB 2000: GOSUB 18888: GOTO 208

- 300 J=J+1:J=J MOD 22+1: FOR I=1 TO 1295: COLOR=1 MOD J+7: PLOT (2*1) MOD 37,(3*1) MOD 35: NEXT I: GOSUB 10000: GOTO 300 488 FOR 1=1 TO 4:C(1)= RND (16)
 - : NEXT 1
 - 410 FOR 1=3 TO 1 STEP -1:C(1+1) =C(I): NEXT I:C(1)= RND (16): FOR 1=1 TO 5: FOR J=1 TO 4
 - 420 COLOR=C(J):L=J*5+14+1:K=39-L: HLIN K,L AT K: YLIN K,L AT L: HLIN K,L AT L: VLIN K,L AT K: NEXT J,I: GOSUB 10000: GOTO 418
 - 18 STEP 2: J=39-1: HLIN 1, J AT I: GOSUB 640: YLIN I,J AT J: GOSUB 648
 - 610 HLIN 1+2, J AT J: GOSUB 640: VLIN 1+2, J AT 1+2: GOSUB 640 : HEXT I
 - 620 COLOR= RND (16): FOR I=18 TO 0 STEP -2: J=39-1: VLIN I+2, J AT I+2: GOSUB 640: HLIN I+ 2, J AT J: GOSUB 640
- 638 YLIN I, J AT J: GOSUB 648: HLIN 1,J AT 1: GOSUB 640: NEXT I: GOSUB 10000: GOTO 600
- 648 K=1+7:L=K*K*5+K*26+70:L=32767 /L*(PDL (8)/18): POKE 8.K: POKE 1,L MOD 256: POKE 24, L/256+1: CALL 2: RETURN

- 788 I= RND (38)+3; J=I*I*5+1*26+ 70:K=32767/J*(PDL (0)/10): POKE 0,1: POKE 1,K MOD 256 : POKE 24,(K)255)+1: CALL 2 : GOSUB 10000: GOTO 700
 - 800 X=3:A=1880:P=A:L=20:W=4:Y=8 :J=1: COLOR=6: HLIN 0,39 AT 4: COLOR=9: GOSUB 880: COLOR= 12: YLIN 5,M-2 AT X
 - 810 N=2*A-P-A/W: COLOR=0: GOSUB 880: VLIN 5,39 RT X:X=X+1: IF X(39 THEN 820:X=3: VLIN 5,39 AT 1: YLIN 5,39 AT 2
 - 828 P=A:A=N:Y=A/108: COLOR=12: GOSUB 888: 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: YLIN 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 (28-X)-Z)*(RB5 (28-Y)-Z)/25 : PLOT X,Y: NEXT X,Y: GOSUB 18888: GOTO 988
- 1880 CALL -936
- 1010 J=1+J NOD 32: COLOR=J/2: YLIH 8,39 RT 3+J: YTAB 21+(J/2) NOD 2: TAB 3+J: IF J MOD 2 THEN PRINT J/2;: GOSUB 10000: GOTO 1010
- 2000 COLOR= RND (16); HLIN 8,39 AT
- J: COLOR= RND (16): VLIN 0,
 - 39 AT J: RETURN
- 10000 IF PEEK (-16384)(128 THEN RETURN
 - : POKE -16368.0: POP : GOTO

30

THIS IS A SHORT DESCRIPTION OF HOW TO PLAY STARTREK ON THE APPLE COMPUTER. THE UNIVERSE IS MADE UP OF 64 QUADRANTS IN AN 8 BY 8 MATRIX. THE QUADRANT IN WHICH YOU THE ENTERPRISE ' ARE, IS IN WHITE, AND A BLOW UP OF THAT QUADRANT IS FOUND IN THE LOWER LEFT Corner. Your space ship status is found in a table to THE RIGHT SIDE OF THE QUADRANT BLOW UP. THIS IS A SEARCH AND DESTROY MISSION. THE OBJECT IS TO LONG-RANGE SENSE FOR INFORMATION AS TO WHERE KLINGONS (K) ARE MOVE TO THAT QUADRANT, AND DESTROY. AND DESTROY. NUMBERS DISPLAYED FOR EACH QUADRANT DENOTE: * OF STARS IN THE ONES PLACE * OF BASES IN THE TENS PLACE * OF KLINGONS IN THE HUNDREDS PLACE AT ANY TIME DURING THE GAME, FOR INSTANCE BEFORE ONE TOTALLY RUNS OUT OF ENERGY, OR NEEDS TO REGENERATE ALL SYSTEMS, ONE MOVES TO A QUADRANT WHICH INCLUDES A BASE, IONS NEXT TO THAT BASE (B) AT WHICH TIME THE BASE SELF-DESTRUCTS AND THE ENTERPRISE (E) HAS ALL SYSTEMS 'GO' AGAIN. TO PLAY: 1. THE COMMANDS CAN BE OBTAINED BY TYPING A '0' (ZERO) AND RETURN. THEY ARE: 1. PROPULSION 2. REGENERATE 3. LONG RANGE SENSORS 5. PHOTON TORPEDOES 4. PHASERS 6. GALAXY RECORD 8. PROBE 7. COMPUTER . SHIELD ENERGY 10.DAMAGE REPORT 11.LOAD PHOTON TORPEDOES 11.LOAD PHOTON TORPEDOES 2.THE COMANDS ARE INVOKED BY TYPING 1HE NUMBER REFERING TO THEM FOLLOWED BY A 'RETURN'. A.IF RESPONSE IS 1 THE COMPUTER WILL ASK WARP OR ION AND EXPECTS 'W' IF ONE WANTS TO TRAVEL IN THE GALAXY BETWEEN QUADRANTS AND AN 'I' IF ONE WANTS ONLY INTERNAL QUADRANT TRAVEL. DURATION OF WARP FACTOR IS THE NUMBER OF SPACES OR QUADRANTS THE ENTERPRISE WILL MOVE. COURSE IS COMPASS READING IN DEGREES FOR THE DESI-RED DESTINATION. B.A 2 REGENERATES THE ENERGY AT 1HE EXPENSE OF TIME. C.A 3 GIVES THE CONTENTS OF THE IMMEDIATE. ADJACENT QUADRANTS. THE GALAXY IS WRAP-AROUND IN ALL DIRECTIONS. D.4 FIRES PHASERS AT THE EXPENSE OF AVAILABLE ENERGY. D. 4 FIRES PHASERS AT THE EXPENSE OF AVAILABLE ENERGY. E.5 INITIATES A SET OF QUESTIONS FOR TORPEDO FIRING. They can be fired automatically if they have been locked on target while in the computer mode, or may be fired manually if the tragectory angle ISKNOWN. F.6, 8 AND 18 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 7. RETURN TO COMAND MODE 6.STATUS 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: А 270-----90 180 -.-.- THIS EXPLANATION WAS WRITTEN BY ELWOOD -.-.-.-.-.-. Not responsible for ERRORS

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. Do not depress the "RETURN" key yet.
- 5. Start the tape recorder in playback mode and depress the "RETURN" key. The flashing cursor disappears.
- 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 <i>num2</i> .
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 <i>num1</i> .
LIST num1, num2	Lists program line number numl through line number num2.

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

NO DSP var Clears DSP mode for variable var.

NO TRACE Clears TRACE mode.

- <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^C .

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.
NOT	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	
1	$6\emptyset \ Y = X \ 3$	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.

<u>Symbol</u>	Sample Statement	Explanation
=	160 IF D = E THEN 500	Expression "equals" expression.
=	17Ø IF A\$(1,1)= "Y" THEN 5VV	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.
>=	220 IF A>=B THEN 100	Expression "is greater than or equal to" expression.
<=	230 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 l "and" expression 2 must both be "true" for statements to be true.</td></d>	Expression l "and" expression 2 must both be "true" for statements to be true.
OR	250 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\$) LEN (str\$)	320 330 335	PRINT PRINT PRINT	ASC(3\$)	Gives decimal ASCII value of designated string variable str. If more than one character is in designated string or sub-string, it gives decimal ASCII value of first character. Gives current length of designated string variable str\$;i.e., number of
				characters.
PDL (<i>expr</i>)	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 (<i>expr</i>)	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(exprl, expr2)	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 (<i>expr</i>)	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

<u>CALL</u> 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	30 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 <i>expr</i> bytes of memory limited by available memory. For string arrays <i>-str\$-(expr)</i> 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	11Ø END	Stops program execution. Sends carriage return and "> " BASIC prompt) to screen.
FOR 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>expr1</i> then increments it by amount in expression <i>expr3</i> each time the corresponding "NEXT" statement is encountered, until value of expression <i>expr3</i> is reached. If STEP <i>expr3</i> is omitted, a STEP of +1 is assumed. Negative numbers are allowed.
<u>GOSUB</u> expr	14Ø GOSUB 5ØØ	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 2ØØ 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.
<u>HLIN</u> 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:		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.

	220 IF A> B THEN PRINT A 230 IF X=0 THEN C=1 240 IF A#10 THEN GOSUB 200 250 IF A\$(1,1)# "Y" THEN 100 legal: 260 IF L> 5 THEN 50: ELSE 60 egal: 270 IF L> 5 THEN 50	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.
<u>INPUT</u> varl, var2, str\$	GO TO 6Ø 280 INPUT X,Y,Z(3) 290 INPUT "AMT", DLLR 300 INPUT "Y or N?", A\$	Enters data into memory from I/O device. If number input is expected, APPLE 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	310 IN# 6 320 IN# Y+2 330 IN# 0	Transfers source of data for subsequent INPUT statements to peripheral I/O slot (1-7) as specified as by expression <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>	340 LET X=5	Assignment operator. "LET" is optional
LIST num1, num2	350 IF X>6 THEN	Causes program from line number <i>numl</i> through line number num2 to be displayed on screen.
NEXT varl, var2	360 NEXT I 370 NEXT J,K	Increments corresponding "FOR" variable and loops back to statement following "FOR" until variable exceeds limit.
<u>NO DSP</u> var	380 NO DSP I	Turns-off DSP debug mode for variable
<u>NO TRACE</u>	390 NO TRACE	Turns-off TRACE debug mode

PLOT expr1, expr2	400 PLOT 15, 25 400 PLT XV,YV	In standard resolution color graphics, this command plots a small square of a predefined color (set by COLOR=) at horizontal location specified by expression <i>exprl</i> in range Ø to 39 and vertical location specified by expression <i>exprl</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, XMOD25E	Stores <u>decimal</u> number defined by expression <i>expr</i> 2 in range of Ø 255 at <u>decimal</u> memory location specified by expression <i>exprl</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 L1, X2 470 PRINT "AMT=";DX 480 PRINT A\$;B\$; 490 PRINT 492 PRINT "HELLO" 494 PRINT 2+3	Outputs data specified by variable var or string variable str\$ starting at current cursor location. If there is not trailing "," or ";" (Ex 450) a carriage return will be generated. Commas (Ex. 460) outputs data in 5
		left justified columns. Semi-colon (Ex. 470) inhibits print of any spaces. Text imbedded in " " will be printed and may appear multiple times.
<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	530 TAB 24 540 TAB 1+24 550 IF A#B THEN TAB 20	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
<u>TRACE</u>	570 TRACE 580 IFN >32000	Sets debug mode that displays each line number as it is executed. THEN TRACE
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> .
<u>VTAB</u> expr	610 VTAB 18 620 VTAB Z+2	Similar to TAB. Moves cursor to absolute vertical position specified by expression expr in the range 1 to 24. VTAB 1 is top line on screen; VTAB24 is bottom.

SPECIAL CONTROL AND EDITING CHARACTERS

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

<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 transfered to System Monitor and Apple prompts with a "*" (asterisk) and a bell. Hitting RESET key does NOT destroy existing BASIC or machine language program.
Control B	If in System Monitor (as indicated by a "*"), a control B and a carriage return will transfer control to BASIC, <u>scratching (killing) any existing BASIC program</u> and set HIMEM: to maximum installed user memory and LOMEM: to 2048.
Control C	If in BASIC, halts program and displays line number where stop occurred*. Program may be continued with a CON command. If in <u>System</u> Monitor, (as indicated by "*"), control C and a carraige return will enter BASIC <u>without</u> killing current program.
Control G	Sounds bell (beeps speaker)
Control H	Backspaces cursor and deletes any overwritten characters from computer but not from screen. Apply supplied keyboards have special key "÷" on right side of keyboard that provides this functions without using control button.
Control 3	Issues line feed only
Control V	Compliment to H ^C . Forward spaces cursor and copies over written characters. Apple keyboards have H-O 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.

<u>CHARACTER</u>

DESCRIPTION OF ACTION

A _E	Move cursor to right
BE	Move cursor to left
C _E	Move cursor down
D _E	Move cursor up
E	Clear text from cursor to end of line
F E	Clear text from cursor to end of page
@ E	Home cursor to top of page, clear text to end of page.

Table A: APPLE II COLORS AS SET BY COLOR =

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

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

<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 use Page 1 only)					
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					
TEXT Mode Controls							
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<40. Wl>Ø					
ØØ22	110 POKE 34,11	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					
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. 14Ø and 15Ø perform identical function.					
ØØ25	16Ø CV=PEEK (37) 17Ø POKE 37,CV 18Ø VTAB(CV+1)	Similar to above. Read/set cusor vertical position in the range Ø to 23.					
ØØ32	190 POKE 50,127 200 POKE 50,255	Set inverse flag if 127 (Ex. 190) Set normal flag if 255(Ex. 200)					
FC58	210 CALL -936	(@ _E) Home cusor, clear screen					
FC42	220 CALL -958	(F _E) Clear from cusor to end of page					

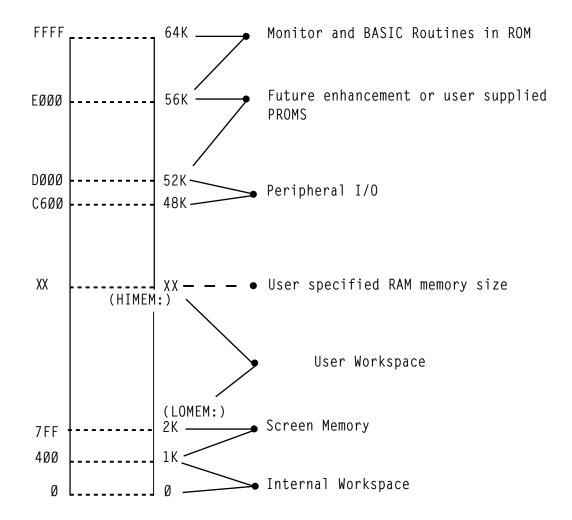
<u>Hex</u>	BASIC Example	Description
FC9C	230 CALL -868	(EE) Clear from cusor to end of line
FC66	240 CALL -922	(J ^C) Line feed
FC7Ø	250 CALL -912	Scroll up text one line

<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	41Ø 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 ₂		hn+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 _{n+l}	
		represent	ted in memory	100 characters as ASCII equi- order bit set.		
		DSP set i that disp	in BASIC, init blays this var ber every time	et to Øl when iates a proces iable with the it is changed	<u>j</u>	
		bytes (fi high orde	「VARIABLE ADD irst low order er) indicating of the next v	, the second the memory		
		DATA - AS order bit	SCII equivalen t set.	ts with high		
			ERMINATOR (ST) t set characte tring.			

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
LW=	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.

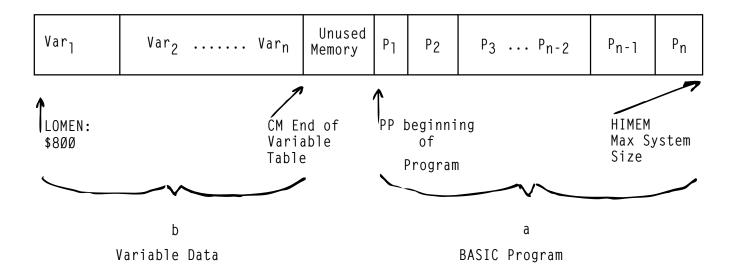


Figure 1

a)	PRINT	PEEK(2Ø4)	+	PEEK(2Ø5)*256	PP
b)	PRINT	PEEK(202)	+	PEEK(2Ø3)*256	СМ



800	8Ø1	8Ø2	8Ø3	8Ø4	8Ø5	8Ø6	8Ø7	808	8Ø9	8ØA	8ØB	8ØC
C1	ØØ	Ø6	Ø8	FF	FF	C2	ØØ	0C	Ø8	ØØ	ØØ	ØØ
		L	Н	L	Н			L	Н			
VAR	DSP	Ν	VA	DA	TA	VAR	DSP	Ν	VA	DA	TA	
NAM		•	V			NAM		N	!			
			1			1						1
				\rightarrow				T		\rightarrow		

Figure 3 \$800.80C rewritten with labelling

FIGURE 4b

REA	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.
1Ø	GOTO 100	This statement moves command to the main program.
2Ø	PRINT "REWIND TAPE THEN START TAPE RECORDER": INPUT "THEN HIT RETURN", B\$	Lines 20-26 are the write data to tape subroutine.
22	A=CM-LM: POKE 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	Writing data table to tape
26	PRINT "DATA TABLE SAVED": RETURN	Returning control to main program.
3Ø	PRINT "REWIND THE TAPE THEN START TAPE RECORDER": INPUT "AND HIT RETURN", B\$	Lines 30-38 are the READ data from tape subroutine.
32	POKE 60,4: POKE 61,8: POKE 62,5: POKE 63,8: CALL -259	
34	IF A<01 THEN 38: P=LM+A: IF P>HM THEN 38: CM=P: POKE 6Ø, LM MOD 256: POKE 61, LM/256: POKE 52, CM MOD 256: POKE 63, CM/256: CALL -259	Checking the record length (A) for memory requirements if everything is satisfactory the data is READ in.
36	PRINT "DATA READ IN": RETURN	
38	PRINT "***TOO MUCH DATA BASE***": RETURN	Returning control to main program.

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

1	>A=1 >	Define variable A=-1, then hit RETURN
2	B=Ø >	Define variable B=Ø, then hit RETURN
3	>PRINT PEEK (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:	
	10- C1 00 86 08 FF FF C2 00	
Ø80	00 00 00 00 00 00 00 00 00 00 00 00 00	

Example 1

Example 2

XLIST

0 A=0

- 18 GOTO 188
- 20 REM WRITE DATA TO TAPE ROUTINE 22 A=CN-LM: POKE 60,4: POKE 61
- ,8: POKE 62,5: POKE 63,8: CALL
- -387
- 24 POKE 60,LM MOD 256: POKE 61 ,LM/256: POKE 62,CM MOD 256 : POKE 63, CM/256: CALL -307
- 26 RETURN
- 30 REM READ DATA SUBROUTINE
- 32 POKE 60,4: POKE 61,8: POKE 62,5: POKE 63,8: CALL -259
- 34 IF A<0 THEN 38:P=LM+A: IF P> HM THEN 38: CM=P: POKE 60,LM MOD 256: POKE 61,LM/256: POKE 62 ,CM MOD 256: POKE 63,CM/256 : CALL - 259
- 36 RETURN
- 38 PRINT **** TOO MUCH DATA BASE ** *':END
- 100 DIM A\$(1),X(20)
- 105 FOR I=1 TO 20:X(I)=I: NEXT I
- 108 LM=2048:CM=2106:A=58:HM=16383

110 PRINT '20 NUMBERS GENERATED'

- 120 PRINT 'NOW WE ARE GOING TO SAVE THE DATA': PRINT 'WHEN YOU ARE R EADY START THE RECORDER IN RECOR D MORE': INPUT 'AND HIT RETURN' ,A\$
- 130 CALL -936: PRINT 'NOW WRITING DA TA TO TAPE': GOSUB 20
- 135 PRINT "NOW THE DATA IS SAVE"
 - 140 PRINT "NOW WE ARE GOING TO CLEAR THE X(20) TABLE AND READ THE DA TA FROM TAPE"
- 150 FOR I=1 TO 20:X(I): NEXT I
- 160 PRINT 'NOW START TAPE RECORDER' INPUT 'AND THEN HIT RETURN' ,A\$
- 165 PRINT 'A ',A
- 170 GOSUB 30
- 188 PRINT 'ALL THE DATA READ IN'
- 198 FOR I-1 TO 28: PRINT 'X(';I; ')=';X(I): NEXT I 195 PRINT 'THIS IS THE END' 288 END

A SIMPLE TONE SUBROUTINE

INTRODUCTION

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

Tone Generation

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

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

Machine Language Program

This machine language program resides in page \emptyset of memory from \$92 (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.

8888-				222	
0000-				222	
0002-	ΑD	30	CØ	LDA	\$0030
0005-	88			DEY	
0006-	00	84		BHE	\$000C
0008-	C6	81		DEC	\$01
000A-	FØ	88		BEQ	\$8814
000C-	CA			DEX	
0000-	00	FS		BHE	\$0005
000F-	A6	00		LDX	\$88
0011-	40	02	00	.JMP	\$8882
0014-	68			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: P0KE 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 β 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 ۰1 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:

J		-	-7	14	
\mathbf{T}				Ψ	
4		-1		2	_
5	Z		1	t C	r -
14	5		-5-		بد

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.

Shape vectors: CODES START A CBA B C $\overline{\Psi}$ ØØ Ø ØØ F 个 01 0 1 0 010 Ø ٥ ١ 44) 001 t 111 ١. 1 ٦ Ť T ſΦ î 2 Ø | Ø 000 00 ۱ 3 11 → 1 1 t 0 I I 01 100 00 ١ 4 Ô ۱ ۱ οl 岭 ۱ T 5 100 L ۱ 01 0 0 IØI \$ J っ L ٢ Ô L ١ O 7 L 11 Ø ١ **€**↓ 01 1 L O 8 4 L t 000 ۱ ۱ L P Emptri 000 0 00 ð O (This vector cannot be a plot vector Figure 1. A Move Up (1) 90 B Hex-becimal Codes < 12 = 0001 ØOI æ φ 3 F 0000 \rightarrow 0,0 L ۱ ۱ l ۱ 0 1 ١. 0001 ÷ 00 1 0 O. 000 ZΦ t Z ~> 64 0010 2 01100 3 1 00 3 2 D 4 001 ラ 1 001 0 l L 01 15 ラ ٩ 0100 5 0001 0 0 I ł 36 5 01 ~> O t 001 U 0 l 10 6 ١E 0110 ラ 6 0001 ł Ł 10 7 00000 7 Ø 7 シ 8 11 01 L ł ł 8 1000 ⇒ 00000000 O \$ + Entri ٩ denotes end of vector table. 00 1 → ٩ ·L 2. A Figure 1010 ~ マ οιι B t 00 -7 C ١ ⇒ ٤ υι Ď 7 E 10 I F -> t t l t

TALL A STATE A STATE A STATE

>REM HIRES DEMO-BASIC LISTING

X157

- 1 INIT=3072:CLEAR=3086:PO5N=3761 :PLOT=3780:LINE=3786:SHAPE= 3805:FIND=3667:SINTBL=3840
- 5 0IK X(10),Y(10)
- 18 TEXT : CALL -936: VTAB 4: TAB
 18: PRINT "*** 16K APPLE II ***"
 PRINT " *** HIGH RESOLUTION G
 RAPKICS DENOS ***": PRINT
- 15 PRINT "1 RANDON LINE DRAW AT BRS IC SPEED": PRINT "2 RANDON SWAPE PROJECTED INTO CORNER"
- 20 FRINT "3 CHRIS' NAD FOLLY": PRINT "4 RANDOM SHAPE SPIRALING INTO POINT": PRINT "5 SPIROGRAP H"
- 25 PRINT "6 HI-RES DONUT": PRINT 17 RANDOM WAYE FORM": PRINT "8 SUM OF TWO SINE WAYES"
- 38 PRINT : PRINT "HIT ANY KEY FOR N EW DEMC": PRINT "TYPE 'CONTROL C ' ; RETURN BUITON THEN TYPE 'T EXT AND RETURN BUITON TO STOP"
- 50 PRINT : INPUT "WHICH DEMO # 50 Y OU WANT ",X1
- 90 IF XIXI OR X1/8 THEN 10: CALL INIT: GOTO 100*X1
- 108 CALL INIT:X=40:7=X: 59508 2000 : POKE 812,255: CALL PLOT
- 110 X= RKD (200):Y= RHD (160): GOSUB 2000: CALL LINE: IF NOT RHD (200) THEN POKE 23,(PEEK (20)+ RHD (3)+1) MOD 4+85: GOSUB-3000: GOTO 110
- 200 GOSUB 1000:X= RNG (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 POK<u>e</u> 28 , RHD (4)≭85;¥=Y+YDIR≉B; IF Y>=9 AND Y<166 THEN 519:YD18= -YDIR:Y=-Y: IF YK® THER Y=Y+ 318: GOSU**G 3000:** GOTO 510 600 POKE -16302.0: POKE 768.5: POKE 769,0: POKE 880,148: POKE 801 ,0: POKE 802,0: POKE 804,9; POKE 805.3: POKE \$12.255: CALL POSN 618 FOR R=0 TO 4160: POKE 807.R MOD 64: POKE 306,2+6* NOT (R MCD 65): CALL SHAPE: NEXT R: GOSUB 3080: 69T0 610 700 J= RND (10)+ RND (10);K= RND (33)+ 別級 (31)+ 別級 (68):1= RND (9)/8: PRINT "FRED#1= " ;;;* FRE8≇2= *;K 710 SOSU8 4000: GOSUB 3000: GOTO 760 800 INPUT "REL FREQ \$1=",1: INPUT *REL FRER #2=*,K; INPUT *MODE (0 =SOLID, 1=POINTS>".L 816 GOSUB 4000: GOSUB 3000: GOTO 89R 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): HEXT 1 1818 X=X(1):Y=Y(1): GOSUE 2008: RETURN 2000 POKE 800.X NOD 256: POKE 801 .X>255: POKE 882.Y: RETURN 3000 IF PEEK (~16384)(128 THEN RETURN : POXE -16368.0: POP : GOTO 10 4000 CALL INIT: POKE 812,255;8=0 :8=6: FOR 1=0 TO 279:A=(A+J) MOD 256:8=(8+%) MOD 256:Y= (PEEK (SINTBL+A)+ PEEK (SINTBL+ 8))#5/16

4010 POKE 800,1 HOD 256: POKE 801 ,1)255: POKE 802,Y: CALL LINE-64(NOT 1 OR L): NEXT 1: RETURN

- 218 X(I)=(X(I)-X)*9/10+X:Y(I)=(
 Y(I)-Y)*9/10+Y: NEXT I,J: GOSUB
 3000: GOTO 200
 308 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
 HOT RND (200) THEN 90KE 20,
 RND (4)*85
 320 X1=X+(RND (3)-1)*25:Y1=Y+(
 RND (3)-1)*15: IF X1(0 OR
 X1)279 OR Y1(0 OR Y1)159.THEN
 320
- 338 X=X1:Y=Y1: GOSU8 2008; CALL LINE: GOSU8 3000; GOTO 310
- 400 GOSUB 1000: POKE 812, RHD (3)+85+85: CALL PLOT
- 410 FOR J=1 TO 25: FOR I=1 TO R: POKE 800,X(I) MOD 255: FOKE 881,X)255: POKE 802,Y(I): CALL LINE
 - 428 X=(X(I)-88+(Y(I)-80)/8)*9/10 +88:Y(I)=(Y(I)-88-(X(I)-80) /8)*9/10+88:X(I)=X: NEXT I, J: GOSUB 2008: GOTC 400
 - 500 CALL INIT: POKE 899,9: CALL PLOT:X=0:Y=0:XDIR=1:YDIR=1: R=5:8=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 (\$00) THEN 520:R=1+ RND (13):B=2+ RND (8):C=4+ RND (7)
- 520 POKE 800,X MOD 236: POKE 801 ,X>255: POKE 802,0: CALL LINE: X=X+XDIR*8: IF X>=0 RND X<280 THEN 530:XDIR=-XDIR:X=-X: IF X<0 THEN X=X+358

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 0=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 MEXT J,I 145 MEXT W: GOTO 105

5 REH PONG BY WENDELL BITTER 128 IF Y=PP+3 THEN V=-1: IF Y=PP+ 235 IF H THEN 245:P(1)=((PDL (10 REM 77777 15 REM PRODLE SWITCHES CONTROL PADDLE SIZE AFTER A MISS OR DURING & HIT 20 GR 25 DIM P(3): DIM HP\$(10) 30 A=38:8=1:C=-1 35 COLOR=13: HLIN 1,38 AT 0: HLIN 140 IF X=0 THEN VVO= AB5 (V) 1,38 RT 39 49 CALL -936: VTAB 23: INPUT "HANDB ALL OR PONG ? ".₩P\$ 45 INPUT *PADDLE SIZE (1-6) *, 155 IF PEEK (-16287))127 AND 50 PS: IF PS(1 OR PS)6 THEN 45 :5=85-1 58 CALL -936 55 IF HP\$(1)&"H" THEN 205 60 H=1: COLOR=13: VLIN-0.33 AT 39: 5070 205 65 FOR X≠A TO 8 STEP C 70 Y=YY+V: 17 Y>1 AND YKO8 THEN THEH Y=38 75 V=-V: FOR T=1 TO 5:M= PEEK 185 VTAB 23: TAB 7: PRINT SL;: TAB 260 PRINT **: 200 (-16336): NEXT T 80 TF X=C OR X=39+C THEN 85: COLOR= 8: PLOT X-C.YY: COLGR=15: PLOT 8.8 85 YY=Y: IF X NOD 2=8 THEN GOSUB 235: NEXT X 98 GOSVB 235 95 IF SCRNCX,Y+V+(7+V(40 AND Y+)127 AND S#5 THEN S=S+1: IF ÿ>-1))=0 THEN 165 100 FOR T=1 TO 10:M= PEEK (-16336): WEXT T 105 IF H RND C>8 THEN 136 119 PP=P(%/38) 115 IF Y=PP THEN V=3: IF Y=PP+1 THEN V=2: IF Y=PP+2 THEN V= i

4 THEN V=+2; IF Y=PP+5 THEN ¥=-3 125 IF S=8 THEN V=3- RND (7) 138 COLOR=0: PLOT X-C.Y 135 IF (K AND C)0) OR (VYO= ABS 240 COLOR=6; VLIN P(1),P(1)+5 AT (V) AND X=8) THEN V=4- RND (9) 145 8=39-8:B=39-8:C=-C 158 IF PEEK (-16286)>127 RNC S# 245 P(0)=((POL (0)-24)#20)/115 5 THEN S=S+i 0 THEN 5-5-1 160 GOTO 65 165 COLOR=0: FLOT X-C,Y 170 COLOR=15: PLOT X,Y+Y*(Y+V)-1 AND Y+V(40) 175 FOR T=1 TC 75:M= PEEK (-16336)+ PEEK (-16336)- PEEK (-16336 255 COLOR=0: IF P(0))P(2) THEN >: NEXT T 88: IF Y(1 THEN Y=1: IF Y)38 188 IF X=8 THEN SR=SR+1: IF X=39 (P(2) THEN VLIN P(6)+5+1,39 THEM SL=SL+1 33: PRINT SR 198 COLOR=8: PLOT X-C,Y 195 IF SL=15 OR SR=15 THEN 260 200 COLOR=0: PLOT X, Y+V+(Y+V)-1 AND 7+VY(48) 205 FOR T=1 TO 75: IF T MOD 5#0 THEN 218: IF PEEK (-16286) PEEK (-16287))127 AND 580 THEN S≠S+1 218 GOSUB 235: NEXT T 215 YY=P(8): IF X=0 THEN YY=P(1) 220 IF H THEN YY= RHD (37)+1 225 ¥=1- RND (3) 238 6070 65

- 1)-24)+29)/115: IF P(1)=P(3) THEN 245; IF P(1)(8 THEN P(1)=0: IF P(1)+S>39 THEN P(1)=39-5
- 39: COLOR=8: IF P(1)>P(3) THEN VLIN 0.P(1)-1 AT 39: IF P(1 XP(3) THEN VLIN P(1)+5+1,39 AT 39:P(3)=P(1)
- : IF P(0)X0 THEN P(0)=0: IF P(8)=P(2) THEN RETURN : IF P(0)+S>39 THEN P(0>=39-S
- 250 COLOR=6: VLIN P(0),P(0)+S AT 0: COLOR=0: IF P(0))P(2) THEN VLIH 8,P(0)-1 AT 8: IF P(0) (P(2) THEN VLIN P(0)+5+1.39 AT 9
 - VLHH 8.P(0)-1 AT 9: IF P(8) 87 9: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.

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 18 POKE 14,198: POKE 15,24: POKE 16,248: POKE 17,5: POKE 18. 198: POKE 19,1: POKE 28.76: POKE 21,2: POKE 22,0: "POKE 23,96 15 DIN B\$(40); TEXT : CALL -936 : GOTO 90 28 CALL -936: 6010 98 25 A= LEN(B\$): FOR Z=1 TO A: GOSUB 65: PRINT B\$(Z,Z);: #EXT 2: GOSUE 76: RETURN 35 B‡="COLOR SKETCH": RETURN 48 B\$="COPYRIGHT APPLE COMPUTER 197 7°: RETURN 45 B\$="THIS PROGRAM ALLOUS YOU TO " RETURN 58 S\$="SKETCH COLORED FIGURES IN" : RETURN 55 8\$="LOW RESOLUTION GRAPHICS WITH PRODLES": RETURN 68 KK=28:TOH=28: GOSUB 85: RETURN 65 KK=10:TOH=10: GOSUB 85: RETURN 70 KK=20:TON=50: GDSUB 85:KK=30 :TON=90: GOSUB 85: RETURN 75 KK=28:TON=28: GOSUB 85: RETURN 88 KK=8:TON=250: G850B 85:KK=9 :TON=259: GOSUB 85: RETURN

85 POKE 1,TON MOD 256: POKE 24 ,TON/256+1: POKE 0,KK: CALL 2: RETURN 98 605UB 38: 605UB 25: PRINT : 788 13: 60568 35: 60568 25 : PRINT : 605UB 30: 605UB 25 : PRINT : THE 5: GOSUE 40: GOSUE 25: PRINT : GOSUB 30: 60508 25 95 PRINT : GOSUB 70: GOSUB 45: GOSUB 25: PRINT : GOSUB 58 : GOSUB 25: PRINT : GOSUB 55 : GOSUB 25: PRINT 100 PRINT : PRINT : GOSUB 70: INPUT *#HEN RERDY HIT RETURN*,B\$ 185 GR 118 B\$="ABCDEFGHIJKLANDP": CALL -936 115 FOR 2=8 TO 15: COLOR=Z: PLOT 2+2+4,39: YTAB 21: GOSUB 75 : TAB Z#2+5: PRINT B\$(2+1.2+ 1):: GOSU8 75: NEXT 2: TAB 1 120 YTAB 22:8\$="TYPE A LETTER TO CH ANGE COLOR.": GOSUB 25: PRINT :B\$="TYPE SPACE BAR TO STOP PLAT .*: GOSUB 25: PRINT 125 Y= PDL (1)#38/255;X= PDL (0)#39/255: VTAB 24: TAB 1: PRINT "CURSOR POSITION: X=";X;" Y=" <u>بر</u> ۱۳ 138 1F PEEK (~16384))127 THEN 145 : IF X1=X AND Y1=Y THEN 125 : COLOR=C2: PLOT X1.Y1: IF NOT FLAG THEN 135: COLOR=C: PLOT X.Y

135 C2= 5CRH(X,Y):C3=15: IF C2= 15 THEN C3=5: COLOR=C3: PLOT X,Y:X1=X:Y1=Y

148 GOTO 125

- 145 IF PEEK (-16384)#160 THEN 155 :Flag=0: Poke -16368,0: Poke 34,28: Color=0: Hlin 0,39 At 39: Crll -936
- 150 PRINT :B\$="CONTINUE OR STOP" : VTAB 24: GOSUB 25: INPUT " (C/S) ",B\$: IF B\$(1,1)="C" THEN 110: PRINT "END": END
- 155 FLAG=1:C= PEEK (~16384)-193 : POKE -16368,0: GOTO 125

MASTERMIND PROGRAM

PROGRAM DESCRIPTION

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

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

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

```
0 REN GAME OF MASTERNIND 8-25-77
      ¥OZ (APPLE COMPUTER)
   10 DIM R(E),C(8),D(5),X(8),X$(
      8):X(1)=2:X(2)=12:X(3)=1:X(
      4)=13:X(5)=3:X(6)=9:X(7)=15
     :X(8)=5:X$="66RYY0¥X"
   20 TEXT : CALL -936: PRINT "
  ¥E1.00
      ME TO THE GRME OF NASTERMIND!
      YOUR OBJECT IS TO GUESS 5 COLOR
     S (WHICH*
   30 PRINT "I WILL MAKE UP) IN THE MJ
      NIMUN NUMBER OF GUESSES. THER
     E ARE EIGHT DIFFERENT COLORS TO
     CHOSE FROM."
   46 PRINT "
FEWER THRM 7 GUESSES--EXC
     ELLENT": PRINT " 7 TO 9 GUESSE
     S----GOOD": PRINT " 10 TO 14 G
     UESSES----AVERAGE"
   59 PRINT "NORE THAN 14 GUESSES--POC
     Ē
": CALL -384; TAB 7: PRINT
     "HIT ANY KEY TO BEGIN FLAY"
  188 CALL -386: IF PEEK (-16384)
     <132 THEN 180: POKE -16368,
     8: GR : PRINT : FOR I=1 TO
     8:0(1)= RND (8)+1: COLOR=X(
     i): WLIN I*4-2,I*4 AT 39: PRINT
     " ";X$(I,I);: HEXT 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 QUESS #":
```

208 Y=TRY*2 HOD 36+1:TRY=TRY+1; TRE 32: PRINT TRY:: COLOR= 0: KLIN 0,39 BT Y;FLASH=1; FOR N=1 TO 5:A(X)=8: GOSUB 1000 : NEXT H:H=1 300 FOR WAIT=1 TO 10:KEY= PEEK (-16384): IF KEY(132 THEN 310 : POKE -16368.0:FLASH=1: FOR I=1 TO S: IF KEY() ASC(X\$(I)) THEN HEXT I: IF I=9 THEN 310;A(N)=1:KEY=149 310 GOSUS 1000: IF KEY=141 THEN 400: IF KEY=136 AND H>1 OR KEY=149 AND AGG THEN N=N+KEY/ 5-28: NEXT WAIT:FLASH=1-FLASH: 6070 588 488 COLOR=15:X=0: FOR I=1 TO 5: D(1)=C(1);J=1: G05UB 2000: NEXT I: IF M=5 THEN 580: COLOR=5 : FOR J=1 70 5: FOR I=1 TO 5: GOSUB 2000; NEXT I.J: GOTO 268 500 PRINT : PRINT " YOU GOT IT IN " ;TRY;" TRIES (";: IF TRY(7 THEN PRINT "EXCELLENT": IF TRY> 6 BHD TRYVIN THEN PRINT "GOOD" Ĵ 519 IF TRY>9 AND TRY(15 THEN PRINT "AVERAGE";: 1F TRY>14 THEN PRINT "POOR";: PRINT ")": CALL -384: TRE 5: PRINT *HIT ANY KEY TO PLAY AGAIN": GOTO 100 1888 IF H=6 THEN RETURN : COLOR= X(A(N))*FLASH: HLIN N*4-2.H* 4 AT Y: RETURN 2000 IF A(I)()D(J) THEN RETURN ; N=M+1: PLOT 21+M+N,Y: PRINT **;:A(1)=0:D(1)=9: RETURN

3000 REN CALL -384 SETS INVERSE VID 3010 REN CALL -380 SETS NORMAL VID 3020 REM PEEK(-16384) IS KRD (ASC11) (IF) 127 THEN STROBE SET) 3830 REN POKE-16368 CLRS KBD STROBE 3040 REN CALL-936 CLEARS SCREEN AND TABS CURSOR TO UPPER LEFT. 3050 REM IN 310, KEY/5-28= -1 OR +1 (ARPOW KEY=136 OR 149 ASCII) 4000 REN STNTS 10-50 INTRO-4818 REN STHTS 196-118 NEW SETUP 4820 REN STHT 200 NEW GUESS 4830 REN STNTS 300-318 USER INPUT 4040 REN STAT 400 GUESS EVAL 4058 REN STATS 509-518 WIN 4868 REN SUBR 1808 COLOR LINE 4873 REN SUBR 2000 NRTCH TEST

PROGRAM DESCRIPTION

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

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

Physical

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

Emotional

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

Mental

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

Biorhythms

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

REQUIREMENTS

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

PROGRAM LISTING: BIORHYTHM

5 POKE 2,173: POKE 3,48: POKE 4,192: POKE 5,165: POKE 6,8 : POKE 7,32: POKE 8,168: POKE 9,252: POKE 10,165: POKE 11 ,1: POKE 12,208: POKE 13,4 18 POKE 14.198: POKE 15.24: POKE 16,240: POKE 17,5: POKE 18, 198: POKE 19,1: POKE 28,75: POKE 21,2: POKE 22,8: POKE 23,96. 15 GOTO 85 20 TT=3: G05UB 30: RETURN 38 KK=8:TON=589: GOSUB 45: RETURN 35 KK=8:TOH=258: GOSØ8 45: RETURN 40 KK=8:TON=250: GOSUB 45:KK=9 :TON=250: GOSUB 45: RETURN 45 POKE 1, TON MOD 256: POKE 24 ,TON/256+1: POKE 8,KK: CALL 2: RETURN 50 a=(19-(P*B(1)/100))*(P*100(C(1))+(P+18B)C(1))+(P+108(= 34C(1))4((P*100-C(1))/100*B(

55 A=A+(P*100)3*C(1))*(38-((P*
100~3*C(1))/100*8(1)/100));
A=39*(A)39)*A*(A(40); RETURN

I)/100)

60 KK=8:TN=500: GOSUB 70:KK=9: TN=250: GOSUB 70: RETURH 65 KK=7:TN=10: GOSUB 70: RETURN 70 POKE 1,TN NOD 256: POKE 24, TM/256+1: POKE 0,KK: CHLL 2 : RETURN 75 GOSUB 60: INPUT "DATE (M,D,Y) "

- 3 60508 601 10701 "DALE (A,D,T)" ,N,D,Y:Y=Y+(Y(100)*1988
- 80 A=Y-(M(3):N=Y MOD 58*365-Y/ 58*82+A/4-A/400+M*31-H/12-M/ 7-N/5-3*(M)2)+D: IF N(0 THEN N=H+21252: RETURN
- 85 DIN H\$(10),B\$(3),B(3),C(3), BY(3):B(1)=348:B(2)=286:B(3)=242:C(1)=575:C(2)=708:C(3)=825:BY(1)=23:BY(2)=28
- 98 BW(3)=33: TEXT : CALL -936: POKE 34,20: GOSUB 26: GOSUB 25: GOSUB 20: FRINT : TAB 10 : PRINT "APPLE II BIORNYTHM (4K) ": TAB 15: PRINT
- 95 GOSUB 25: TAB 5: PRINT "COPYRIGH I 1977 APPLE COMPUTER INC." : POKE 34,24: VTAB 24 100 GOSUB 60: INPUT "NRME ",N\$:
- VTAB 22: PRINT N\$: VTAB 24 : PRINT "BIRTH ";: GOSUB 75 : VTAB 22: TAB 21: PRINT "BIRTH DATE ";N;",";D;",";Y: VTAB 24:N1=N: CALL -868 105 PRINT "FORECAST ";: GOSUB 75 :N=N-N1: IF N<0 THEN N=N+21252 - (100 02) TAB 40 OPTIME 150000
- : VTAB 23: TAB 18: PRINT "FORECA St date ";m;",";d;",";y: vtab 24: Call -068

- 110 J=1: GR : POKE 34,23: FOR X= 18 TO 20: COLOR=3: HLIN 0,31 AT X: NEXT X; HLIN 1,3 AT 3: HLIN 1,3 AT 37: VLIN 2,4 AT 2: VTAB 21
- 115 FOR Y=1 TO 31 STEP 3: PRINT Y;: IF Y(10 THEN PRINT * "; : PRINT " ";: NEXT Y: PRINT " P E N": VTAB 24
- 120 VTAB 23: PRINT "DAYS LIVED " ;N: FOR I=1 TO 3: COLOR=1*(I=1)+6*(I=2)+3*(I=3): VLIN 0,39 AT 33+1+1: VTAB 24
- 125 FOR X=0 TO 31:P=(N HOD 8V(I) +X) HOD 8V(I): GOSUB 50: PLOT X,A: GOSUB 65: NEXT X: HEXT I
 - 130 PRINT : INPUT "ANOTHER PLOT (Y/H) ",8\$: IF 8\$(1,1)="Y* THEN 90: END

Þ

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 8751 3 PRINT "YOU MAY WATCH WHILE I BUI LD A MAZE." 4 PRINT "BUT WHEN IT'S COMPLETE, I 'LL FRASE* 5 PRINT "THE PICTURE. THEN YOU'LL OHLY SEE THE WALLS AS YOU BUMP I NTG THEM." 6 PRINT "TO NOVE, 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 18 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 0 DOT) EATS" 13 PRINT "YOU." 14 PRINT "BEWARE!!!!!!!! SOMETIMES THE DREGON" 15 PRINT "GETS REAL MAD, AND CLIMBS OVER A WALL," 16 PRINT "BUT NOST OF THE TIME. HE CAN'T GO OVER" 17 PRINT "AND HAS TO GO AROUND," 18 PRINT 19 PRINT *(WINT: YOU CRN OFTEN TELL

WHERE A WALL*

20 PRINT PIS. EVEN BEFORE YOU CAN S 1998 Q=R+D+L+U EE IT, BYª 21 PRINT "THE FACT THAT THE DRAGON CBN'T GET" 22 PRINT "THROUGH IT!)" 23 PRINT 89 DIN 8\$(3) 90 PRINT "TYPE 'GO' TO BEGIN " :: INPUT AS 180 GR : COLOR=15 195 CALL -936; PRINT "DRAGON WAZE" :: TAB (25): PRINT "GARY J. SHAN KON™ 110 FOR 1=0 TO 39 STEP 3: VLIN 8,39 AT I: HLIN 8,39 AT I: HEXT 1146 GOTO 1035 Ī 120 COLOR=# 130 S=1008 1000 DIN N(169),T(169) 1981 FOR I=1 TO 169;T(I)=8; NEXT Ţ 1010 FOR I=1 TO 169:1X(I)=11: NEXT // 1165 HLIN 3*X-2,3*X-1 AT 3*Y: 60TO I 1838 X= RND (13)+1;Y= RND (13)+1 1178 X= RND (13)+1;Y= RND (13)+1 :0=169 1035 IF C=1 THEN 1200 1040 R=0:D=8:L=0:U=0:K=X+13*(Y-1):N(K)=- ABS (N(K)):C=C+1 1050 IF X=13 THEH 1060:R=N(K+1)> ŝ 1960 IF Y=13 THEN 1978:D=WK+13) }₿ 1070 IF X=1 THEN 1080:L=N(K-1))@ 1089 IF Y=1 THEN 1099:0=N(K-13)> ü

1108 IF (Q(3 AND RND (18)(2) OR Q=0 THEN 1170 1118 DR= RND (4) 1120 GOTO 1130+16*DR 1130 IF NOT & THEH 1110:N(K)=N(K) +1+X=X+1 1135 VLIN 3*Y-2.3*Y-1 AT 3*(X-1) 1136 GOTO 1935 1140 IF NOT D THEN 1110:M(K)=H(K) +10:Y=Y+1 1145 HLIN 3#X-2,3*X-1 8T 3*(Y-1) 1150 IF NOT L THEN 1110:N(K-1)=N(K-1)-1:X=X-1 1155 YLIN 3*Y-2.3*Y-1 AT 3*X 1156 6070 1935 1160 IF NOT U THEN 1110:N(K-13)= M(K-13)-18:Y=7-1 1835 1180 IF N(X+13*(Y-1))>0 THEN 1170 1198 C=C+1: GOID 1035 1200 GOSUB 5000: PRINT "THE MAZE IS R EADAu 1285 GR : COLOR=15 1210 VLIN 0.39 AT 6: VLIN 0.39 AT 🔆 39: HLIN 8,39 AT 8: KLIN 9, 39 87 39 1220 X=1:Y= RND (13)+1; COLOR=8; PLOT 3#X-2,3*Y-2

1225 HX=3*X-2;HY=3*Y-2 1230 ¥Y= RND (13)+1 1240 COLOR=0: YLIN 3*8Y-2,3+8Y-1 AT 39 1250 SX=13:SY=#Y 1268 QX=3*5X-2:QY=3*5Y-2 1278 RD=1 1500 K= PEEK (-16384): IF K(128 THEm 1599 1518 POKE -16368,8 1515 QQ=K: GOSUB 7060:K=QQ 1516 IF SX=X AND SY=Y THEN 8006 1520 IF K= RSC("R") THEN 2000 1530 IF K= ASC("L") THEN 2500 1540 IF K= ASC("U") THEN 3000 1550 IF K= ASC("D") THEN 3500 1569 GOSUB 5980: GOTO 1580 2000 DX=1:DY=0 2010 IF M(X+13*(Y-1)) MOD 10 THEN 49392020 FX=3*X-2:FY=3*Y-2: FOR I=1 TO 3 2038 FX=FX+DX:FY=FY+DY 2840 COLOR=9 2060 FOR K=0 TO 1: FOR L=0 TO 1: PLOT HX+K,HY+L: HEXT L,K: COLOR= 4320 HLIN 3*(X-1),5*X AT 3*Y 8: FOR K=0 TO 1: FOR L=0 TO 1: PLOT FX+K,FY+L: NEXT L,K: 8X=FX;8Y=FY 2110 NEXT I 2115 X=X+DX:Y=Y+DY 2116 IF X=13 AND Y=NY THEN 6000 2128 6010 1589 2500 DX=-1:0Y=0 2518 IF N(X+13*(Y-1)-L) NOD 10 THEN 4100

2520 5078 2028 3000 DX=0:DY=-1 3018 IF W(X+13+(Y-2))/10 THEN 4200 3828 6010 2828 3588 0X=0:0Y=1 3510 IF N(X+13+(Y-1))/10 THER 430s 3528 GOTO 2629 4888 GOSUB 5888 4010 COLOR=15 4926 VLIH 3*(Y-1),3*Y AT 3*X 4939 6010 1598 4180 50508 5989 4118 COLOR=15 4120 YLIN 3#(Y-1),3#Y AT 3#(X-1) 4138 GOTO 1588 4288 GOSUB 5868 4218 COLOR=15 4220 HLIN 3*(X-1),3*X BT 3*(Y-1) 4230 6010 1588 4388 50SUB 5888 4318 COLOR=15 4338 GOT0 1566 5000 S=S-1: FOR I=1 TO 20:9= PEEK (-16336)+ PEEK (-16336)+ PEEK (-16336)+ PEEK (-16336): #EXT I: RETURN 6060 PRINT "YOU WIN!" 6810 GOSUE 5000: GOSUE 5000: GOSUE 5888 6020 PRINT "SEORE=":5+3

7000 IF XXSX THEN 7005: IF YXSY THEN 7858 7001 IF K(SX THEN 7100: IF Y(SY THEN 7158 7005 IF SX=13 THEN 7050; IF T(SX+ 13*(5Y-1)))9 THEN 7018: IF N(SX+13+(SY-1)) HOD 10 THEN 76567018 DX=1:DY=0 7028 COLOR=8 7822 RX=3*5X-2:RY=3*5Y-2 7023 FOR I=1 TO 3: RX=RX+DX: RY=RY+ Ŵ٧ 7824 COLOR=9 7025 FOR K=0 TO 1: FOR L=0 TO 1: PLOT \$X+K, \$Y+L: HEXT L.K; COLOR= RD: FOR K=8 TO 1: FOR L=0 TO 1: PLOT RX+K, RY+L: NEXT L.K: QX=RX:QY=RY 7638 REXT 1 7835 SX=SX+DX: SY=SY+DY 7848 T(SX+13*(SY-1))=T(SX+13*(SY-1))+i 7845 RETURN 7858 IF SY=13 THEN 7188: IF T(5X+ 13*(5Y-1))>9 THEN 7060: 1F N(SX+13+(SY-1))/18 THEN 7100 7868 DX=8:DY=1: 6010 7628 7100 IF SX=1 THEN 7150: IF T(SX+ 13*(5Y-1))>9 THEN 7110: IF

H(5X+13+(5Y-1)-1) NOD 10 THEN

7158

6038 END

DRAGON MAZE cont.

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

APPLE II FIRMWARE

- **1.** System Monitor Commands
- 2. Control and Editing Characters
- **3. Special Controls and Features**
- 4. Annotated Monitor and Dis-assembler Listing
- 5. Binary Floating Point Package
- 6. Sweet 16 Interpreter Listing
- 7.6502 Op Codes

System Monitor Commands

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

Command Format	Example	Description
Examine Memory		
adrs	*CØF2	Examines (displays) single memory location of (adrs)
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)
М	*N	Set normal video mode. (White characters on black background)
Dis-assembler		
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.
<u>Mini-assembler</u>		
(Turn-on)	*F666G	Turns-on mini-assembler. Prompt character is now a "!" (exclamation point).
\$(monitor: command)	\$C800L	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	<u>Example</u>	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)	*γC	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	Arithmetic	
datal+data2	*78+34	Performs hexidecimal sum of datal plus data2.
datal-data2	*AE-34	Performs hexidecimal difference of datal minus data2.

<u>Command Format</u>	Example	Description
Set Input/Output	Ports	
(X) (Control P)	*5PC	Sets printer output to I/O slot number (X). (see Note 2 below)
(X) (Control K)	*2KC	Sets keyboard input to I/O slot number (X). (see Note 2 below)

Note 2:

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

Multiple Commands

*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_C . 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 ^C . Forward spaces cursor and copies over written characters. Apple keyboards have "+" key on right side which also performs this function.
Control X	Immediately deletes current line.
	* If BASIC program is expecting keyboard input, you will have

SPECIAL CONTROL AND EDITING CHARACTERS

(continued)

CHARACTER	DESCRIPTION OF	ACTION
-----------	----------------	--------

- A_F Move cursor to right
- B_E Move cursor to left
- C_F Move cursor down
- D_F Move cursor up
- E_F Clear text from cursor to end of line
- F_E Clear text from cursor to end of page

Hex	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 use Page 1 only)	
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	
TEXT Mode	Controls		
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 Wl. Ll+Wl<4Ø. Wl>Ø	
ØØ22	110 POKE 34,11	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	(@ _E) Home cusor, clear screen	
FC42	220 CALL -958	(F _E) Clear from cusor to end of page	

<u>Hex</u>	BASIC Example	Description
FC9C	230 CALL -868	(E _E) Clear from cusor to end of line
FC66	240 CALL -922	(J ^C) 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	*	
	STEM M	ONITOR	*	
*			*	
		1977 BY		
* APPL *	E COMP	UTER, INC	· * *	
	RIGHTS	RESERVED		
*			*	
*	s. WOZ	NIAK	*	
*	A. BA	UM	*	
*			*	
	TITLE	*******	"APPLE II SYSTEM	MONTTODU
LOC0	EPZ	\$00	AFFDE II DIDIEM	MONTION
LOC1		\$01		
WNDLFT	EPZ	\$20		
WNDWDTH		\$21		
WNDTOP WNDBTM	EPZ	\$22		
CH CV	EPZ EPZ	\$24 ¢25		
GBASL		\$25 \$26		
GBASH BASL	EPZ	\$28		
BASH	EPZ	\$29		
	EPZ			
BAS2H		\$2B		
H2 LMNEM	EPZ EPZ	\$2C		
RTNL	EPZ	\$2C \$2C		
V2	EPZ			
RMNEM		\$2D		
RTNH				
RTNH MASK CHKSUM	EPZ EPZ			
	EPZ	\$2E		
FORMAT				
LASTIN		\$2F		
LENGTH SIGN	EPZ	\$2F		
COLOR	EPZ	\$30		
MODE	EPZ	\$31		
INVFLG	EPZ	\$32		
PROMPT	EPZ EPZ	\$33		
YSAV				
YSAV1 CSWL	EPZ EPZ	\$35 \$36		
CSWH	EPZ	\$37		
KSWL				
KSWH	EPZ EPZ	\$39		
PCL	EPZ	\$3A		
PCH	EPZ			
XQT	EPZ	\$3C		
A1L A1H	EPZ EPZ	\$3C \$3D		
AIH A2L	EPZ	\$3D \$3E		
A2H	EPZ	\$3F		
A3L	EPZ	\$40		
A3H	EPZ	\$41		
A4L	EPZ	\$42		
A4H	EPZ	\$43		
A5L A5H	EPZ EPZ	\$44 \$45		
АЭП	БГД	940		

	ACC	EQU	\$45	
		EQU	\$46	
		EQU		
		EQU	\$48 \$49	
		EQU		
			\$4F	
		EQU		
			\$51	
		EQU		
		EQU		
		EQU	\$54	
		EQU		
			\$95 \$95	
		EQU		
	USRADR			
			\$03FB	
	IRQLOC			
			\$C000	
			\$C000	
	KBDSTRB			
	TAPEOUT			
	SPKR			
			\$C050	
	TXTSET			
	MIXCLR	EOU	\$C052	
	MIXSET			
	LOWSCR			
			\$C055	
	LORES	EQU	\$C056	
	HIRES	EQU	\$C057	
	TAPEIN	EQU	\$C060	
	PADDL0	EQU	\$C064	
	PTRIG	EQU	\$C070	
	BASIC	EQU	\$E000	
	BASIC2	EQU	\$E003	
		ORG		ROM START ADDRESS
F800: 4A	PLOT	LSR		Y-COORD/2
F801: 08		PHP		SAVE LSB IN CARRY
F802: 20 47 F8				CALC BASE ADR IN GBASL,H
F805: 28		PLP		RESTORE LSB FROM CARRY
F806: A9 0F				MASK \$0F IF EVEN
F808: 90 02			RTMASK #CRO	MACK CEO IE ODD
F80A: 69 E0		ADC	#\$E0	MASK \$F0 IF ODD
F80A: 69 E0 F80C: 85 2E	RTMASK	ADC STA	#\$E0 MASK	
F80A: 69 E0 F80C: 85 2E F80E: B1 26		ADC STA LDA	#\$E0 MASK (GBASL),Y	DATA
F80A: 69 E0 F80C: 85 2E F80E: B1 26 F810: 45 30	RTMASK	ADC STA LDA EOR	#\$E0 MASK (GBASL),Y COLOR	DATA EOR COLOR
F80A: 69 E0 F80C: 85 2E F80E: B1 26	RTMASK	ADC STA LDA EOR AND	#\$E0 MASK (GBASL),Y COLOR MASK	DATA EOR COLOR
F80A: 69 E0 F80C: 85 2E F80E: B1 26 F810: 45 30 F812: 25 2E	RTMASK	ADC STA LDA EOR AND EOR	#\$E0 MASK (GBASL),Y COLOR MASK (GBASL),Y	DATA EOR COLOR AND MASK XOR DATA
F80A: 69 E0 F80C: 85 2E F80E: B1 26 F810: 45 30 F812: 25 2E F814: 51 26	RTMASK	ADC STA LDA EOR AND EOR	#\$E0 MASK (GBASL),Y COLOR MASK (GBASL),Y	DATA EOR COLOR AND MASK
F80A: 69 E0 F80C: 85 2E F80E: B1 26 F810: 45 30 F812: 25 2E F814: 51 26 F816: 91 26	RTMASK PLOT1	ADC STA LDA EOR AND EOR STA RTS	#\$E0 MASK (GBASL),Y COLOR MASK (GBASL),Y	DATA EOR COLOR AND MASK XOR DATA
F80A: 69 E0 F80C: 85 2E F80E: B1 26 F810: 45 30 F812: 25 2E F814: 51 26 F816: 91 26 F818: 60	RTMASK PLOT1	ADC STA LDA EOR AND EOR STA RTS	#\$E0 MASK (GBASL),Y COLOR MASK (GBASL),Y (GBASL),Y	DATA EOR COLOR AND MASK XOR DATA TO DATA
F80A: 69 E0 F80C: 85 2E F80E: B1 26 F810: 45 30 F812: 25 2E F814: 51 26 F816: 91 26 F818: 60 F819: 20 00 F8	RTMASK PLOT1 HLINE	ADC STA LDA EOR AND EOR STA RTS JSR CPY	#\$E0 MASK (GBASL),Y COLOR MASK (GBASL),Y (GBASL),Y PLOT	DATA EOR COLOR AND MASK XOR DATA TO DATA PLOT SQUARE
F80A: 69 E0 F80C: 85 2E F80E: B1 26 F810: 45 30 F812: 25 2E F814: 51 26 F816: 91 26 F818: 60 F819: 20 00 F8 F819: C4 2C	RTMASK PLOT1 HLINE	ADC STA LDA EOR AND EOR STA RTS JSR CPY BCS INY	<pre>#\$E0 MASK (GBASL),Y COLOR MASK (GBASL),Y (GBASL),Y PLOT H2 RTS1</pre>	DATA EOR COLOR AND MASK XOR DATA TO DATA PLOT SQUARE DONE?
F80A: 69 E0 F80C: 85 2E F80E: B1 26 F810: 45 30 F812: 25 2E F814: 51 26 F816: 91 26 F818: 60 F819: 20 00 F8 F81C: C4 2C F81E: B0 11 F820: C8 F821: 20 0E F8	RTMASK PLOT1 HLINE	ADC STA LDA EOR AND EOR STA RTS JSR CPY BCS INY	#\$E0 MASK (GBASL),Y COLOR MASK (GBASL),Y (GBASL),Y PLOT H2 RTS1	DATA EOR COLOR AND MASK XOR DATA TO DATA PLOT SQUARE DONE? YES, RETURN
F80A: 69 E0 F80C: 85 2E F80E: B1 26 F810: 45 30 F812: 25 2E F814: 51 26 F816: 91 26 F816: 91 26 F819: 20 00 F8 F81C: C4 2C F81E: B0 11 F820: C8 F821: 20 0E F8 F824: 90 F6	RTMASK PLOT1 HLINE HLINE1	ADC STA LDA EOR AND EOR STA RTS JSR CPY BCS INY JSR BCC	#\$E0 MASK (GBASL),Y COLOR MASK (GBASL),Y (GBASL),Y PLOT H2 RTS1 PLOT1 HLINE1	DATA EOR COLOR AND MASK XOR DATA TO DATA PLOT SQUARE DONE? YES, RETURN NO, INCR INDEX (X-COORD)
F80A: 69 E0 F80C: 85 2E F80E: B1 26 F810: 45 30 F812: 25 2E F814: 51 26 F816: 91 26 F816: 91 26 F818: 60 F819: 20 00 F8 F81C: C4 2C F81E: B0 11 F820: C8 F821: 20 0E F8 F824: 90 F6 F824: 69 01	RTMASK PLOT1 HLINE HLINE1 VLINEZ	ADC STA LDA EOR AND EOR STA CPY BCS INY JSR BCC ADC	<pre>#\$E0 MASK (GBASL),Y COLOR MASK (GBASL),Y (GBASL),Y PLOT H2 RTS1 PLOT1 HLINE1 #\$01</pre>	DATA EOR COLOR AND MASK XOR DATA TO DATA PLOT SQUARE DONE? YES, RETURN NO, INCR INDEX (X-COORD) PLOT NEXT SQUARE ALWAYS TAKEN NEXT Y-COORD
F80A: 69 E0 F80C: 85 2E F80E: B1 26 F810: 45 30 F812: 25 2E F814: 51 26 F816: 91 26 F816: 91 26 F818: 60 F819: 20 00 F8 F81C: C4 2C F81E: B0 11 F820: C8 F821: 20 0E F8 F8221: 20 0E F8 F824: 90 F6 F826: 69 01 F828: 48	RTMASK PLOT1 HLINE HLINE1 VLINEZ VLINE	ADC STA LDA EOR AND EOR STA CPY BCS INY JSR BCC ADC PHA	<pre>#\$E0 MASK (GBASL),Y COLOR MASK (GBASL),Y (GBASL),Y PLOT H2 RTS1 PLOT1 HLINE1 #\$01</pre>	DATA EOR COLOR AND MASK XOR DATA TO DATA PLOT SQUARE DONE? YES, RETURN NO, INCR INDEX (X-COORD) PLOT NEXT SQUARE ALWAYS TAKEN NEXT Y-COORD SAVE ON STACK
F80A: 69 E0 F80C: 85 2E F80E: B1 26 F810: 45 30 F812: 25 2E F814: 51 26 F816: 91 26 F818: 60 F819: 20 00 F8 F812: C4 2C F81E: B0 11 F820: C8 F821: 20 0E F8 F821: 20 0E F8 F824: 90 F6 F826: 69 01 F828: 48 F829: 20 00 F8	RTMASK PLOT1 HLINE HLINE1 VLINEZ VLINE	ADC STA LDA EOR AND EOR STA RTS JSR CPY BCS INY JSR BCC ADC PHA JSR	<pre>#\$E0 MASK (GBASL),Y COLOR MASK (GBASL),Y (GBASL),Y PLOT H2 RTS1 PLOT1 HLINE1 #\$01</pre>	DATA EOR COLOR AND MASK XOR DATA TO DATA PLOT SQUARE DONE? YES, RETURN NO, INCR INDEX (X-COORD) PLOT NEXT SQUARE ALWAYS TAKEN NEXT Y-COORD
F80A: 69 E0 F80C: 85 2E F80E: B1 26 F810: 45 30 F812: 25 2E F814: 51 26 F814: 51 26 F818: 60 F819: 20 00 F8 F812: 20 00 F8 F821: 20 0E F8 F821: 20 0E F8 F824: 90 F6 F826: 69 01 F828: 48 F829: 20 00 F8 F822: 68	RTMASK PLOT1 HLINE HLINE1 VLINEZ VLINE	ADC STA LDA EOR AND EOR STA RTS JSR CPY BCS INY BCS INY JSR BCC ADC PHA JSR PLA	<pre>#\$E0 MASK (GBASL),Y COLOR MASK (GBASL),Y (GBASL),Y PLOT H2 RTS1 PLOT1 HLINE1 #\$01 PLOT</pre>	DATA EOR COLOR AND MASK XOR DATA TO DATA PLOT SQUARE DONE? YES, RETURN NO, INCR INDEX (X-COORD) PLOT NEXT SQUARE ALWAYS TAKEN NEXT Y-COORD SAVE ON STACK PLOT SQUARE
F80A: 69 E0 F80C: 85 2E F80E: B1 26 F810: 45 30 F812: 25 2E F814: 51 26 F814: 51 26 F816: 91 26 F819: 20 00 F8 F819: 20 00 F8 F811: 60 11 F820: C8 F821: 20 0E F8 F821: 20 0E F8 F824: 90 F6 F826: 69 01 F828: 48 F829: 20 00 F8 F822: 68 F82D: C5 2D	RTMASK PLOT1 HLINE HLINE1 VLINEZ VLINE	ADC STA LDA EOR STA RTS JSR CPY BCS JSR BCC ADC PHA JSR PLA CMP	#\$E0 MASK (GBASL),Y COLOR MASK (GBASL),Y (GBASL),Y PLOT H2 RTS1 PLOT1 HLINE1 #\$01 PLOT V2	DATA EOR COLOR AND MASK XOR DATA TO DATA PLOT SQUARE DONE? YES, RETURN NO, INCR INDEX (X-COORD) PLOT NEXT SQUARE ALWAYS TAKEN NEXT Y-COORD SAVE ON STACK PLOT SQUARE DONE?
F80A: 69 E0 F80C: 85 2E F80E: B1 26 F810: 45 30 F812: 25 2E F814: 51 26 F816: 91 26 F818: 60 F819: 20 00 F8 F81C: C4 2C F81E: B0 11 F820: C8 F821: 20 0E F8 F821: 20 0E F8 F824: 90 F6 F826: 69 01 F828: 48 F829: 20 00 F8 F820: C5 2D F82F: 90 F5	RTMASK PLOT1 HLINE HLINE1 VLINEZ VLINE	ADC STA LDA EOR STA RTS JSR CPY BCS INY JSR BCC ADC PHA JSR PLA CMP BCC	<pre>#\$E0 MASK (GBASL),Y COLOR MASK (GBASL),Y (GBASL),Y PLOT H2 RTS1 PLOT1 HLINE1 #\$01 PLOT</pre>	DATA EOR COLOR AND MASK XOR DATA TO DATA PLOT SQUARE DONE? YES, RETURN NO, INCR INDEX (X-COORD) PLOT NEXT SQUARE ALWAYS TAKEN NEXT Y-COORD SAVE ON STACK PLOT SQUARE DONE?
F80A: 69 E0 F80C: 85 2E F80E: B1 26 F810: 45 30 F812: 25 2E F814: 51 26 F816: 91 26 F816: 91 26 F818: 60 F819: 20 00 F8 F81C: C4 2C F81E: B0 11 F820: C8 F821: 20 0E F8 F821: 20 0E F8 F824: 90 F6 F826: 69 01 F828: 48 F829: 20 00 F8 F829: 20 00 F8 F820: C5 2D F82F: 90 F5 F831: 60	RTMASK PLOT1 HLINE HLINE1 VLINEZ VLINE RTS1	ADC STA LDA EOR STA RTS JSR CPY BCS INY JSR BCC ADC PHA JSR PLA CMP BCC RTS	#\$E0 MASK (GBASL),Y COLOR MASK (GBASL),Y (GBASL),Y (GBASL),Y PLOT H2 RTS1 PLOT HLINE1 #\$01 PLOT PLOT V2 VLINEZ	DATA EOR COLOR AND MASK XOR DATA TO DATA PLOT SQUARE DONE? YES, RETURN NO, INCR INDEX (X-COORD) PLOT NEXT SQUARE ALWAYS TAKEN NEXT Y-COORD SAVE ON STACK PLOT SQUARE DONE? NO, LOOP
F80A: 69 E0 F80C: 85 2E F80E: B1 26 F810: 45 30 F812: 25 2E F814: 51 26 F814: 51 26 F818: 60 F819: 20 00 F8 F81C: C4 2C F81E: B0 11 F820: C8 F821: 20 0E F8 F824: 90 F6 F824: 90 F6 F826: 69 01 F828: 48 F829: 20 00 F8 F82C: 68 F82D: C5 2D F821: 60 F831: 60 F832: A0 2F	RTMASK PLOT1 HLINE HLINE1 VLINE2 VLINE2 VLINE	ADC STA LDA EOR STA RTS JSR CPY BCS INY JSR BCC ADC PHA JSR PHA CMP BCC RTS LDY	<pre>#\$E0 MASK (GBASL),Y COLOR MASK (GBASL),Y (GBASL),Y PLOT H2 RTS1 PLOT1 HLINE1 #\$01 PLOT V2 VLINEZ #\$2F</pre>	DATA EOR COLOR AND MASK XOR DATA TO DATA PLOT SQUARE DONE? YES, RETURN NO, INCR INDEX (X-COORD) PLOT NEXT SQUARE ALWAYS TAKEN NEXT Y-COORD SAVE ON STACK PLOT SQUARE DONE? NO, LOOP MAX Y, FULL SCRN CLR
F80A: 69 E0 F80C: 85 2E F80E: B1 26 F810: 45 30 F812: 25 2E F814: 51 26 F816: 91 26 F818: 60 F819: 20 00 F8 F812: C4 2C F81E: B0 11 F820: C8 F821: 20 0E F8 F824: 90 F6 F824: 90 F6 F822: 48 F829: 20 00 F8 F829: 20 00 F8 F820: C5 2D F82F: 90 F5 F831: 60 F832: A0 2F F834: D0 02	RTMASK PLOT1 HLINE HLINE1 VLINE2 VLINE2 VLINE2 RTS1 CLRSCR	ADC STA LDA EOR AND EOR STA RTS JSR CPY BCS INY JSR BCC ADC PHA JSR PHA CMP BCC RTS LDY BNE	<pre>#\$E0 MASK (GBASL),Y COLOR MASK (GBASL),Y (GBASL),Y PLOT H2 RTS1 PLOT1 HLINE1 #\$01 PLOT V2 V1 V2 V1 V2 V1 V2 V1 V2 V1 V2 V1 V2 V1 V2 V1 V2 V1 V2 V2 V1 V2 V1 V2 V1 V2 V1 V2 V1 V2 V1 V2 V1 V2 V1 V1 V2 V1 V1 V2 V1 V1 V2 V1 V1 V2 V1 V1 V2 V1 V1 V2 V1 V1 V2 V1 V1 V1 V1 V1 V1 V1 V1 V1 V1 V1 V1 V1</pre>	DATA EOR COLOR AND MASK XOR DATA TO DATA PLOT SQUARE DONE? YES, RETURN NO, INCR INDEX (X-COORD) PLOT NEXT SQUARE ALWAYS TAKEN NEXT Y-COORD SAVE ON STACK PLOT SQUARE DONE? NO, LOOP MAX Y, FULL SCRN CLR ALWAYS TAKEN
F80A: 69 E0 F80C: 85 2E F80E: B1 26 F810: 45 30 F812: 25 2E F814: 51 26 F814: 51 26 F818: 60 F819: 20 00 F8 F812: 20 00 F8 F812: C4 2C F81E: B0 11 F820: C8 F821: 20 0E F8 F824: 90 F6 F826: 69 01 F828: 48 F829: 20 00 F8 F820: C5 2D F82F: 90 F5 F831: 60 F832: A0 2F F834: D0 02 F836: A0 27	RTMASK PLOT1 HLINE HLINE1 VLINEZ VLINE RTS1 CLRSCR CLRTOP	ADC STA LDA EOR STA RTS JSR CPY BCS INY JSR BCC ADC PHA JSR PHA JSR PHA CMP BCC RTS LDY ENE LDY	<pre>#\$E0 MASK (GBASL),Y COLOR MASK (GBASL),Y (GBASL),Y PLOT H2 RTS1 PLOT1 HLINE1 #\$01 PLOT V2 VLINEZ #\$2F CLRSC2 #\$27</pre>	DATA EOR COLOR AND MASK XOR DATA TO DATA PLOT SQUARE DONE? YES, RETURN NO, INCR INDEX (X-COORD) PLOT NEXT SQUARE ALWAYS TAKEN NEXT Y-COORD SAVE ON STACK PLOT SQUARE DONE? NO, LOOP MAX Y, FULL SCRN CLR ALWAYS TAKEN MAX Y, TOP SCREEN CLR
F80A: 69 E0 F80C: 85 2E F80E: B1 26 F810: 45 30 F812: 25 2E F814: 51 26 F814: 51 26 F818: 60 F819: 20 00 F8 F812: 20 00 F8 F812: C4 2C F81E: B0 11 F820: C8 F821: 20 0E F8 F824: 90 F6 F826: 69 01 F828: 48 F829: 20 00 F8 F820: C5 2D F82F: 90 F5 F831: 60 F832: A0 2F F834: D0 02 F836: A0 27	RTMASK PLOT1 HLINE HLINE1 VLINEZ VLINE RTS1 CLRSCR CLRTOP	ADC STA LDA EOR STA RTS JSR BCS JSR BCC ADC PHA SCC PHA CMP BCC RTS LDY BNE LDY STY	<pre>#\$E0 MASK (GBASL),Y COLOR MASK (GBASL),Y (GBASL),Y PLOT H2 RTS1 PLOT1 HLINE1 #\$01 PLOT V2 VLINEZ #\$2F CLRSC2 #\$27 V2</pre>	DATA EOR COLOR AND MASK XOR DATA TO DATA PLOT SQUARE DONE? YES, RETURN NO, INCR INDEX (X-COORD) PLOT NEXT SQUARE ALWAYS TAKEN NEXT Y-COORD SAVE ON STACK PLOT SQUARE DONE? NO, LOOP MAX Y, FULL SCRN CLR ALWAYS TAKEN MAX Y, TOP SCREEN CLR STORE AS BOTTOM COORD
F80A: 69 E0 F80C: 85 2E F80E: B1 26 F810: 45 30 F812: 25 2E F814: 51 26 F814: 51 26 F818: 60 F819: 20 00 F8 F812: 20 00 F8 F812: C4 2C F81E: B0 11 F820: C8 F821: 20 0E F8 F824: 90 F6 F826: 69 01 F828: 48 F829: 20 00 F8 F820: C5 2D F82F: 90 F5 F831: 60 F832: A0 2F F834: D0 02 F836: A0 27	RTMASK PLOT1 HLINE HLINE1 VLINEZ VLINE RTS1 CLRSCR CLRTOP	ADC STA LDA EOR STA RTS JSR CPY BCS JSR BCC ADC PHA CMP BCC RTS LDY BNE LDY STY FOR	<pre>#\$E0 MASK (GBASL),Y COLOR MASK (GBASL),Y (GBASL),Y PLOT H2 RTS1 PLOT HLINE1 #\$01 PLOT V2 VLINEZ #\$2F CLRSC2 #\$27 V2 VLINE CALLS</pre>	DATA EOR COLOR AND MASK XOR DATA TO DATA PLOT SQUARE DONE? YES, RETURN NO, INCR INDEX (X-COORD) PLOT NEXT SQUARE ALWAYS TAKEN NEXT Y-COORD SAVE ON STACK PLOT SQUARE DONE? NO, LOOP MAX Y, FULL SCRN CLR ALWAYS TAKEN MAX Y, TOP SCREEN CLR STORE AS BOTTOM COORD
F80A: 69 E0 F80C: 85 2E F80E: B1 26 F810: 45 30 F812: 25 2E F814: 51 26 F816: 91 26 F819: 20 00 F8 F81C: C4 2C F81E: B0 11 F820: C8 F821: 20 0E F8 F824: 90 F6 F826: 69 01 F828: 48 F820: C5 2D F821: 60 F832: A0 2F F834: D0 02 F836: A0 27 F83A: A0 27	RTMASK PLOT1 HLINE HLINE1 VLINEZ VLINE RTS1 CLRSCR CLRTOP CLRSC2	ADC STA LDA EOR AND EOR STA CPY BCS INY JSR BCC ADC PHA JSR PLA CMP BCC CMP BCC CMP BCC CMP BCC STY STY FOR STY FOR	<pre>#\$E0 MASK (GBASL),Y COLOR MASK (GBASL),Y (GBASL),Y PLOT H2 RTS1 PLOT1 HLINE1 #\$01 PLOT V2 VLINEZ #\$2F CLRSC2 #\$27 V2 VLINE CALLS #\$27</pre>	DATA EOR COLOR AND MASK XOR DATA TO DATA PLOT SQUARE DONE? YES, RETURN NO, INCR INDEX (X-COORD) PLOT NEXT SQUARE ALWAYS TAKEN NEXT Y-COORD SAVE ON STACK PLOT SQUARE DONE? NO, LOOP MAX Y, FULL SCRN CLR ALWAYS TAKEN MAX Y, TOP SCREEN CLR STORE AS BOTTOM COORD
F80A: 69 E0 F80C: 85 2E F80E: B1 26 F810: 45 30 F812: 25 2E F814: 51 26 F818: 60 F819: 20 00 F8 F810: C4 2C F81E: B0 11 F820: C8 F821: 20 0E F8 F824: 90 F6 F824: 90 F6 F822: 48 F829: 20 00 F8 F822: 69 F822: 69 F822: 68 F821: 50 F831: 60 F832: A0 2F F834: D0 02 F836: A0 27 F83A: A0 27	RTMASK PLOT1 HLINE HLINE1 VLINEZ VLINE RTS1 CLRSCR CLRTOP CLRSC2	ADC STA LDA EOR AND EOR STA RTS CPY BCS INY JSR BCC ADC PHA JSR PHA JSR PHA CMP BCC RTS LDY STY FOR LDY LDY	<pre>#\$E0 MASK (GBASL),Y COLOR MASK (GBASL),Y (GBASL),Y PLOT H2 RTS1 PLOT1 HLINE1 #\$01 PLOT V2 VLINEZ #\$2F CLRSC2 #\$27 V2 VLINE CALLS #\$27 #\$00</pre>	DATA EOR COLOR AND MASK XOR DATA TO DATA PLOT SQUARE DONE? YES, RETURN NO, INCR INDEX (X-COORD) PLOT NEXT SQUARE ALWAYS TAKEN NEXT Y-COORD SAVE ON STACK PLOT SQUARE DONE? NO, LOOP MAX Y, FULL SCRN CLR ALWAYS TAKEN MAX Y, TOP SCREEN CLR STORE AS BOTTOM COORD RIGHTMOST X-COORD (COLUMN)
F80A: 69 E0 F80C: 85 2E F80E: B1 26 F810: 45 30 F812: 25 2E F814: 51 26 F818: 60 F819: 20 00 F8 F816: C4 2C F81E: B0 11 F820: C8 F821: 20 0E F8 F824: 90 F6 F824: 90 F6 F826: 69 01 F828: 48 F829: 20 00 F8 F820: C5 2D F821: 60 F832: A0 2F F831: 60 F832: A0 27 F838: A0 27 F83A: A0 27	RTMASK PLOT1 HLINE HLINE1 VLINEZ VLINE RTS1 CLRSCR CLRTOP CLRSC2	ADC STA LDA EOR AND EOR STA RTS JSR CPY BCS INY JSR BCC ADC PHA JSR PHA CMP BCC RTS LDY ENE LDY STY FOR STA	<pre>#\$E0 MASK (GBASL),Y COLOR MASK (GBASL),Y (GBASL),Y PLOT H2 RTS1 PLOT1 HLINE1 #\$01 PLOT V2 VLINEZ #\$2F CLRSC2 #\$27 V2 VLINE CALLS #\$27 V2 VLINE CALLS #\$27 K</pre>	DATA EOR COLOR AND MASK XOR DATA TO DATA PLOT SQUARE DONE? YES, RETURN NO, INCR INDEX (X-COORD) PLOT NEXT SQUARE ALWAYS TAKEN NEXT Y-COORD SAVE ON STACK PLOT SQUARE DONE? NO, LOOP MAX Y, FULL SCRN CLR ALWAYS TAKEN MAX Y, TOP SCREEN CLR STORE AS BOTTOM COORD S RIGHTMOST X-COORD (COLUMN) TOP COORD FOR VLINE CALLS
F80A: 69 E0 F80C: 85 2E F80E: B1 26 F810: 45 30 F812: 25 2E F814: 51 26 F818: 60 F819: 20 00 F8 F816: C4 2C F81E: B0 11 F820: C8 F821: 20 0E F8 F824: 90 F6 F824: 90 F6 F826: 69 01 F828: 48 F829: 20 00 F8 F829: 20 00 F8 F820: C5 2D F82F: 90 F5 F831: 60 F832: A0 2F F834: D0 02 F834: A0 27 F838: 84 2D F83A: A0 27 F83A:	RTMASK PLOT1 HLINE HLINE1 VLINEZ VLINE RTS1 CLRSCR CLRTOP CLRSC2	ADC STA LDA EOR AND EOR STA RTS JSR CPY BCS INY JSR BCC ADC PHA JSR PHA CMP BCC RTS LDY ENE LDY STY FOR STA	<pre>#\$E0 MASK (GBASL),Y COLOR MASK (GBASL),Y (GBASL),Y PLOT H2 RTS1 PLOT HLINE1 #\$01 PLOT V2 VLINEZ #\$2F CLRSC2 #\$27 V2 VLINE CALLS #\$27 #\$00 COLOR VLINE</pre>	DATA EOR COLOR AND MASK XOR DATA TO DATA PLOT SQUARE DONE? YES, RETURN NO, INCR INDEX (X-COORD) PLOT NEXT SQUARE ALWAYS TAKEN NEXT Y-COORD SAVE ON STACK PLOT SQUARE DONE? NO, LOOP MAX Y, FULL SCRN CLR ALWAYS TAKEN MAX Y, TOP SCREEN CLR STORE AS BOTTOM COORD RIGHTMOST X-COORD (COLUMN) TOP COORD FOR VLINE CALLS CLEAR COLOR (BLACK)
F80A: 69 E0 F80C: 85 2E F80E: B1 26 F811: 45 30 F812: 25 2E F814: 51 26 F814: 51 26 F814: 51 26 F815: 91 26 F818: 60 F819: 20 00 F8 F812: 20 00 F8 F821: 20 0E F8 F822: 20 02 F8 F822: 68 F822: 68 F822: 20 00 F8 F822: 62 F8 F822: 62 50 F5 F831: 60 F8 F832: A0 2F F834: D0 02 F834: A0 27 F833: A0 27 F833: A0 27 F832: A0 27 F833: A0 27 F835: 30 F840: 20 28 58 </td <td>RTMASK PLOT1 HLINE HLINE1 VLINEZ VLINE RTS1 CLRSCR CLRTOP CLRSC2</td> <td>ADC STA LDA EOR AND EOR STA RTS JSR CPY BCS JSR BCC ADC ADC ADC PHA JSR PLA CMP BCC RTS LDY BNE LDY STY FOR LDY LDY LDY STA STA STA</td> <td><pre>#\$E0 MASK (GBASL),Y COLOR MASK (GBASL),Y (GBASL),Y PLOT H2 RTS1 PLOT H2 RTS1 PLOT HLINE1 #\$01 PLOT V2 VLINE2 #\$27 V2 VLINE CALLS #\$27 V2 VLINE CALLS #\$27 #\$27 V2 VLINE CALLS #\$27 V2 VLINE CALLS #\$27 V2 VLINE CALLS #\$27 V2 VLINE CALLS #\$27 V2 VLINE CALLS #\$27 V2 VLINE CALLS #\$27 V2 VLINE CALLS #\$27 V2 VLINE CALLS</pre></td> <td>DATA EOR COLOR AND MASK XOR DATA TO DATA PLOT SQUARE DONE? YES, RETURN NO, INCR INDEX (X-COORD) PLOT NEXT SQUARE ALWAYS TAKEN NEXT Y-COORD SAVE ON STACK PLOT SQUARE DONE? NO, LOOP MAX Y, FULL SCRN CLR ALWAYS TAKEN MAX Y, TOP SCREEN CLR STORE AS BOTTOM COORD RIGHTMOST X-COORD (COLUMN) TOP COORD FOR VLINE CALLS CLEAR COLOR (BLACK) DRAW VLINE</td>	RTMASK PLOT1 HLINE HLINE1 VLINEZ VLINE RTS1 CLRSCR CLRTOP CLRSC2	ADC STA LDA EOR AND EOR STA RTS JSR CPY BCS JSR BCC ADC ADC ADC PHA JSR PLA CMP BCC RTS LDY BNE LDY STY FOR LDY LDY LDY STA STA STA	<pre>#\$E0 MASK (GBASL),Y COLOR MASK (GBASL),Y (GBASL),Y PLOT H2 RTS1 PLOT H2 RTS1 PLOT HLINE1 #\$01 PLOT V2 VLINE2 #\$27 V2 VLINE CALLS #\$27 V2 VLINE CALLS #\$27 #\$27 V2 VLINE CALLS #\$27 V2 VLINE CALLS #\$27 V2 VLINE CALLS #\$27 V2 VLINE CALLS #\$27 V2 VLINE CALLS #\$27 V2 VLINE CALLS #\$27 V2 VLINE CALLS #\$27 V2 VLINE CALLS</pre>	DATA EOR COLOR AND MASK XOR DATA TO DATA PLOT SQUARE DONE? YES, RETURN NO, INCR INDEX (X-COORD) PLOT NEXT SQUARE ALWAYS TAKEN NEXT Y-COORD SAVE ON STACK PLOT SQUARE DONE? NO, LOOP MAX Y, FULL SCRN CLR ALWAYS TAKEN MAX Y, TOP SCREEN CLR STORE AS BOTTOM COORD RIGHTMOST X-COORD (COLUMN) TOP COORD FOR VLINE CALLS CLEAR COLOR (BLACK) DRAW VLINE
F80A: 69 E0 F80C: 85 2E F80E: B1 26 F811: 45 30 F812: 25 2E F814: 51 26 F814: 51 26 F815: 20 00 F8 F819: 20 00 F8 F815: 60 F8 F8 F812: 20 00 F8 F821: 20 02 F8 F822: 68 F822: 68 F822: 60 F5 F831: 60 F832: A0 27 F8334: A0 27 F833: A0 27 F8335: A2 D F83A: A0 27 F8335: 30 F840: 20 F843: 85 30 F840: 20 28 F8	RTMASK PLOT1 HLINE HLINE1 VLINEZ VLINE RTS1 CLRSCR CLRTOP CLRSC2	ADC STA LDA EOR AND EOR STA RTS JSR CPY BCS JSR BCC ADC ADC ADC PHA JSR PLA CMP BCC RTS LDY BNE LDY STY FOR LDY LDY LDY STA STA STA	<pre>#\$E0 MASK (GBASL),Y COLOR MASK (GBASL),Y (GBASL),Y PLOT H2 RTS1 PLOT H2 RTS1 PLOT HLINE1 #\$01 PLOT V2 VLINE2 #\$27 V2 VLINE CALLS #\$27 V2 VLINE CALLS #\$27 #\$27 V2 VLINE CALLS #\$27 V2 VLINE CALLS #\$27 V2 VLINE CALLS #\$27 V2 VLINE CALLS #\$27 V2 VLINE CALLS #\$27 V2 VLINE CALLS #\$27 V2 VLINE CALLS #\$27 V2 VLINE CALLS</pre>	DATA EOR COLOR AND MASK XOR DATA TO DATA PLOT SQUARE DONE? YES, RETURN NO, INCR INDEX (X-COORD) PLOT NEXT SQUARE ALWAYS TAKEN NEXT Y-COORD SAVE ON STACK PLOT SQUARE DONE? NO, LOOP MAX Y, FULL SCRN CLR ALWAYS TAKEN MAX Y, TOP SCREEN CLR STORE AS BOTTOM COORD S RIGHTMOST X-COORD (COLUMN) TOP COORD FOR VLINE CALLS CLEAR COLOR (BLACK) DRAW VLINE NEXT LEFTMOST X-COORD
F80A: 69 E0 F80C: 85 2E F80E: B1 26 F810: 45 30 F812: 25 2E F814: 51 26 F814: 21 20 00 F812: 20 02 F8 F821: 20 00 F8 F822: 20 00 F8 F822: 20 00 F8 F822: 20 00 F8 F822: 20 00 F8 F831: 60 F5 F831: F832: A0 27 F833: F834: D0 22 F834: 20 F833: A0 27 F833: 84 F834: A0 27 F834: 20	RTMASK PLOT1 HLINE HLINE1 VLINEZ VLINE RTS1 CLRSCR CLRTOP CLRSC2	ADC STA LDA EOR AND EOR STA RTS JSR CPY BCS INY JSR BCC ADC PHA JSR PHA JSR PHA CMP BCC RTS LDY STY FOR LDY LDY LDA STA DEY BPL RTS	<pre>#\$E0 MASK (GBASL),Y COLOR MASK (GBASL),Y (GBASL),Y (GBASL),Y PLOT H2 RTS1 PLOT HLINE1 #\$01 PLOT V2 VLINE1 #\$01 PLOT V2 VLINEZ #\$27 *\$27 V2 VLINE CALLS #\$27 #\$00 COLOR VLINE CLRSC3</pre>	DATA EOR COLOR AND MASK XOR DATA TO DATA PLOT SQUARE DONE? YES, RETURN NO, INCR INDEX (X-COORD) PLOT NEXT SQUARE ALWAYS TAKEN NEXT Y-COORD SAVE ON STACK PLOT SQUARE DONE? NO, LOOP MAX Y, FULL SCRN CLR ALWAYS TAKEN MAX Y, TOP SCREEN CLR STORE AS BOTTOM COORD S RIGHTMOST X-COORD (COLUMN) TOP COORD FOR VLINE CALLS CLEAR COLOR (BLACK) DRAW VLINE NEXT LEFTMOST X-COORD
F80A: 69 E0 F80C: 85 2E F80E: B1 26 F811: 25 2E F814: 51 26 F814: 51 26 F814: 51 26 F814: 51 26 F815: 91 26 F818: 60 F819: 20 00 F8 F812: 20 00 F8 F821: 20 02 F824: 90 F6 F824: 90 F6 F826: 69 01 F826: 68 F8220: C5 2D F825: 20 00 F8 F821: 20 00 F8 F826: 60 F5 F831: 60 F8 F834: D0 27 F833: A0 27 F833: 84 2D F8 F843: 88 F844: A0 27 F835: 30 F842: 88 F844: 10 F6	RTMASK PLOT1 HLINE HLINE1 VLINE2 VLINE2 VLINE RTS1 CLRSCR CLRSC2 CLRSC3	ADC STA LDA EOR AND EOR STA RTS JSR CPY BCS INY JSR BCC ADC PHA JSR PHA CMP BCC RTS LDY BNE LDY STY FOR STA JSR DEY BDA STA JSR LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LSR	<pre>#\$E0 MASK (GBASL),Y COLOR MASK (GBASL),Y (GBASL),Y PLOT H2 RTS1 PLOT1 HLINE1 #\$01 PLOT V2 VLINEZ #\$2F CLRSC2 #\$27 V2 VLINE CALLS #\$27 V2 VLINE CALLS #\$27 V2 CLRSC3</pre>	DATA EOR COLOR AND MASK XOR DATA TO DATA PLOT SQUARE DONE? YES, RETURN NO, INCR INDEX (X-COORD) PLOT NEXT SQUARE ALWAYS TAKEN NEXT Y-COORD SAVE ON STACK PLOT SQUARE DONE? NO, LOOP MAX Y, FULL SCRN CLR ALWAYS TAKEN MAX Y, TOP SCREEN CLR STORE AS BOTTOM COORD S RIGHTMOST X-COORD (COLUMN) TOP COORD FOR VLINE CALLS CLEAR COLOR (BLACK) DRAW VLINE NEXT LEFTMOST X-COORD LOOP UNTIL DONE
F80A: 69 E0 F80C: 85 2E F80E: B1 26 F811: 45 30 F812: 25 2E F814: 51 26 F814: 51 26 F815: 60 F819: 20 F814: 51 26 F815: 60 F819: 20 F812: 20 00 F8 F821: 20 02 F8 F822: 20 00 F8 F822: 20 00 F8 F822: 20 00 F8 F822: 20 00 F8 F821: 50 F5 F831: 60 F832: A0 27 F833: 84 20 F833: A0 27 F833: 84 20 F833: A0 27 F833: 85 30 F844: A0 27 F835: 30 F843: <td< td=""><td>RTMASK PLOT1 HLINE HLINE1 VLINE2 VLINE2 VLINE RTS1 CLRSCR CLRSC2 CLRSC3</td><td>ADC STA LDA EOR AND EOR STA RTS JSR CPY BCS INY JSR BCC ADC PHA JSR PHA LDY STY FOR LDY STY LDY LDA STA JSR PIA LDY STA LDY LDA STA LDY LDA STA LDY LDA STA ADC RTS LDY LDA STA LDY LDA STA ADC RTS LDY LDA STA LDY LDA STA ADC RTS LDY R STA ADC RTS LDY R STA STA STA STA STA STA STA STA STA STA</td><td><pre>#\$E0 MASK (GBASL),Y COLOR MASK (GBASL),Y (GBASL),Y PLOT H2 RTS1 PLOT H2 RTS1 PLOT1 HLINE1 #\$01 PLOT V2 VLINEZ #\$2F CLRSC2 #\$27 V2 VLINE CALLS #\$27 #\$00 COLOR VLINE CLRSC3 #\$03</pre></td><td>DATA EOR COLOR AND MASK XOR DATA TO DATA PLOT SQUARE DONE? YES, RETURN NO, INCR INDEX (X-COORD) PLOT NEXT SQUARE ALWAYS TAKEN NEXT Y-COORD SAVE ON STACK PLOT SQUARE DONE? NO, LOOP MAX Y, FULL SCRN CLR ALWAYS TAKEN MAX Y, TOP SCREEN CLR STORE AS BOTTOM COORD S RIGHTMOST X-COORD (COLUMN) TOP COORD FOR VLINE CALLS CLEAR COLOR (BLACK) DRAW VLINE NEXT LEFTMOST X-COORD LOOP UNTIL DONE FOR INPUT 000DEFGH</td></td<>	RTMASK PLOT1 HLINE HLINE1 VLINE2 VLINE2 VLINE RTS1 CLRSCR CLRSC2 CLRSC3	ADC STA LDA EOR AND EOR STA RTS JSR CPY BCS INY JSR BCC ADC PHA JSR PHA LDY STY FOR LDY STY LDY LDA STA JSR PIA LDY STA LDY LDA STA LDY LDA STA LDY LDA STA ADC RTS LDY LDA STA LDY LDA STA ADC RTS LDY LDA STA LDY LDA STA ADC RTS LDY R STA ADC RTS LDY R STA STA STA STA STA STA STA STA STA STA	<pre>#\$E0 MASK (GBASL),Y COLOR MASK (GBASL),Y (GBASL),Y PLOT H2 RTS1 PLOT H2 RTS1 PLOT1 HLINE1 #\$01 PLOT V2 VLINEZ #\$2F CLRSC2 #\$27 V2 VLINE CALLS #\$27 #\$00 COLOR VLINE CLRSC3 #\$03</pre>	DATA EOR COLOR AND MASK XOR DATA TO DATA PLOT SQUARE DONE? YES, RETURN NO, INCR INDEX (X-COORD) PLOT NEXT SQUARE ALWAYS TAKEN NEXT Y-COORD SAVE ON STACK PLOT SQUARE DONE? NO, LOOP MAX Y, FULL SCRN CLR ALWAYS TAKEN MAX Y, TOP SCREEN CLR STORE AS BOTTOM COORD S RIGHTMOST X-COORD (COLUMN) TOP COORD FOR VLINE CALLS CLEAR COLOR (BLACK) DRAW VLINE NEXT LEFTMOST X-COORD LOOP UNTIL DONE FOR INPUT 000DEFGH
F80A: 69 E0 F80E: B1 26 F80E: B1 26 F811: 25 22 F814: 51 26 F815: 20 0 F816: 91 26 F812: 20 00 F8 F812: 20 00 F8 F812: 20 00 F8 F821: 20 00 F8 F822: 69 01 F8 F822: 69 01 F8 F822: 20 00 F8 F821: 20 00 F8 F822: 22 20 00 F8 F821: 60 F8 F834: 00 F834: 00 27 F838: 84 2D F834: A0 27 F838: 84 2D F834: A0 27 F838: 84 2D F834: A0 27 F838: 88	RTMASK PLOT1 HLINE HLINE1 VLINEZ VLINE RTS1 CLRSCR CLRSC2 CLRSC3 GBASCALC	ADC STA LDA EOR AND EOR STA RTS JSR CCPY BCS INY JSR BCC ADC PHA JSR PLA CMP BCC RTS LDY STY FOR STY FOR STY LDY LDA STY FOR STY STY STY STY STY FOR ADC STY STY STY STY STY STY STY STY STY STY	<pre>#\$E0 MASK (GBASL),Y COLOR MASK (GBASL),Y (GBASL),Y (GBASL),Y PLOT H2 RTS1 PLOT1 HLINE1 #\$01 PLOT V2 VLINE1 #\$01 PLOT V2 VLINE2 #\$27 #\$00 COLOR VLINE CLRSC3 #\$03 #\$04</pre>	DATA EOR COLOR AND MASK XOR DATA TO DATA PLOT SQUARE DONE? YES, RETURN NO, INCR INDEX (X-COORD) PLOT NEXT SQUARE ALWAYS TAKEN NEXT Y-COORD SAVE ON STACK PLOT SQUARE DONE? NO, LOOP MAX Y, FULL SCRN CLR ALWAYS TAKEN MAX Y, TOP SCREEN CLR STORE AS BOTTOM COORD S RIGHTMOST X-COORD (COLUMN) TOP COORD FOR VLINE CALLS CLEAR COLOR (BLACK) DRAW VLINE NEXT LEFTMOST X-COORD LOOP UNTIL DONE
F80A: 69 E0 F80C: 85 2E F80E: B1 26 F810: 45 30 F812: 25 2E F814: 51 26 F814: 21 20 00 F817: 20 00 F8 F821: 20 02 F8 F822: 20 00 F8 F822: 20 00 F8 F822: 20 00 F8 F821: 20 00 F8 F822: 20 20 F8 F821: 60 F8 F8 F832: A0 27 F833: A0 27 F833: A0 27 F834: A0 27 F835: 30 F844: 10 F6	RTMASK PLOT1 HLINE HLINE1 VLINEZ VLINE RTS1 CLRSCR CLRSC2 CLRSC3 GBASCALC	ADC STA LDA EOR AND EOR STA RTS JSR CPY BCS INY JSR BCC ADC PHA JSR PHA CMP BCC RTS LDY STY FOR LDY LDY STY FOR STA STA STA STA STA	<pre>#\$E0 MASK (GBASL),Y COLOR MASK (GBASL),Y (GBASL),Y PLOT H2 RTS1 PLOT HLINE1 #\$01 PLOT V2 VLINE2 #\$27 V2 VLINE CALLS #\$27 *\$27 V2 VLINE CALLS #\$27 #\$00 COLOR VLINE CLRSC3 #\$03 #\$04 GBASH</pre>	DATA EOR COLOR AND MASK XOR DATA TO DATA PLOT SQUARE DONE? YES, RETURN NO, INCR INDEX (X-COORD) PLOT NEXT SQUARE ALWAYS TAKEN NEXT Y-COORD SAVE ON STACK PLOT SQUARE DONE? NO, LOOP MAX Y, FULL SCRN CLR ALWAYS TAKEN MAX Y, FULL SCRN CLR ALWAYS TAKEN MAX Y, TOP SCREEN CLR STORE AS BOTTOM COORD S RIGHTMOST X-COORD (COLUMN) TOP COORD FOR VLINE CALLS CLEAR COLOR (BLACK) DRAW VLINE NEXT LEFTMOST X-COORD LOOP UNTIL DONE FOR INPUT 000DEFGH
F80A: 69 E0 F80C: 85 2E F80E: B1 26 F811: 51 26 F812: 25 2E F814: 51 26 F815: 20 00 F8 F817: 20 00 F8 F821: 20 00 F8 F822: 20 00 F8 F822: 20 00 F8 F822: 20 00 F8 F822: 20 00 F8 F821: 20 02 F8 F831: 60 27 F838: F834: 20 27 F833: 84 20 F834: 20 28 F844: 20 28 F844: 20 <td>RTMASK PLOT1 HLINE HLINE1 VLINEZ VLINEZ CLRSCR CLRSC2 CLRSC3 GBASCALC</td> <td>ADC STA LDA EOR AND EOR STA RTS JSR CCPY BCS INY BCC PHA JSR CMP BCC RTS LDY ENC RTS LDY ENC STY LDY LDY STY JSR CMP BCC RTS LDY ENC RTS LDY ENC RTS LDY ENC RTS LDY ENC RTS LDY ENC RTS LDY ENC RTS LDY ENC RTS LDY ENC RTS LDY ENC RTS LDY ENC RTS LDY ENC RTS RTS RTS RTS RTS RTS RTS RTS RTS RTS</td> <td><pre>#\$E0 MASK (GBASL),Y COLOR MASK (GBASL),Y (GBASL),Y (GBASL),Y PLOT H2 RTS1 PLOT HLINE1 #\$01 PLOT V2 VLINE1 #\$01 PLOT V2 VLINEZ #\$27 V2 VLINE CALLS #\$27 #\$00 COLOR VLINE CLRSC3 #\$03 #\$04 GBASH</pre></td> <td>DATA EOR COLOR AND MASK XOR DATA TO DATA PLOT SQUARE DONE? YES, RETURN NO, INCR INDEX (X-COORD) PLOT NEXT SQUARE ALWAYS TAKEN NEXT Y-COORD SAVE ON STACK PLOT SQUARE DONE? NO, LOOP MAX Y, FULL SCRN CLR ALWAYS TAKEN MAX Y, TOP SCREEN CLR STORE AS BOTTOM COORD S RIGHTMOST X-COORD (COLUMN) TOP COORD FOR VLINE CALLS CLEAR COLOR (BLACK) DRAW VLINE NEXT LEFTMOST X-COORD LOOP UNTIL DONE FOR INPUT 000DEFGH</td>	RTMASK PLOT1 HLINE HLINE1 VLINEZ VLINEZ CLRSCR CLRSC2 CLRSC3 GBASCALC	ADC STA LDA EOR AND EOR STA RTS JSR CCPY BCS INY BCC PHA JSR CMP BCC RTS LDY ENC RTS LDY ENC STY LDY LDY STY JSR CMP BCC RTS LDY ENC RTS LDY ENC RTS LDY ENC RTS LDY ENC RTS LDY ENC RTS LDY ENC RTS LDY ENC RTS LDY ENC RTS LDY ENC RTS LDY ENC RTS LDY ENC RTS RTS RTS RTS RTS RTS RTS RTS RTS RTS	<pre>#\$E0 MASK (GBASL),Y COLOR MASK (GBASL),Y (GBASL),Y (GBASL),Y PLOT H2 RTS1 PLOT HLINE1 #\$01 PLOT V2 VLINE1 #\$01 PLOT V2 VLINEZ #\$27 V2 VLINE CALLS #\$27 #\$00 COLOR VLINE CLRSC3 #\$03 #\$04 GBASH</pre>	DATA EOR COLOR AND MASK XOR DATA TO DATA PLOT SQUARE DONE? YES, RETURN NO, INCR INDEX (X-COORD) PLOT NEXT SQUARE ALWAYS TAKEN NEXT Y-COORD SAVE ON STACK PLOT SQUARE DONE? NO, LOOP MAX Y, FULL SCRN CLR ALWAYS TAKEN MAX Y, TOP SCREEN CLR STORE AS BOTTOM COORD S RIGHTMOST X-COORD (COLUMN) TOP COORD FOR VLINE CALLS CLEAR COLOR (BLACK) DRAW VLINE NEXT LEFTMOST X-COORD LOOP UNTIL DONE FOR INPUT 000DEFGH
F80A: 69 E0 F80C: 85 2E F80E: B1 26 F811: 51 25 2E F814: 51 26 F814: 50 F8 F812: 20 00 F8 F821: 20 02 F8 F822: 20 00 F8 F822: 20 00 F8 F822: 20 00 F8 F822: 20 00 F8 F822: 20 02 F8 F831: 60 27 F838: F834: 20 27 F835: F834: 20 28 F8 F844: 10 F6 F845: F844: 20 28<	RTMASK PLOT1 HLINE HLINE1 VLINEZ VLINEZ CLRSCR CLRSC2 CLRSC3 GBASCALC	ADC STA LDA EOR AND EOR STA RTS JSR CPY BCS INY JSR BCC ADC PHA JSR PHA CMP BCC RTS LDY ENE LDY STY FOR STA JSR DEY BNE LDY STY STA JSR CPP PHA LDY STA STA STA STA AND STA AND STA AND	<pre>#\$E0 MASK (GBASL),Y COLOR MASK (GBASL),Y (GBASL),Y PLOT H2 RTS1 PLOT1 HLINE1 #\$01 PLOT V2 VLINEZ #\$27 V2 VLINE CALLS #\$27 V2 VLINE CALLS</pre>	DATA EOR COLOR AND MASK XOR DATA TO DATA PLOT SQUARE DONE? YES, RETURN NO, INCR INDEX (X-COORD) PLOT NEXT SQUARE ALWAYS TAKEN NEXT Y-COORD SAVE ON STACK PLOT SQUARE DONE? NO, LOOP MAX Y, FULL SCRN CLR ALWAYS TAKEN MAX Y, FULL SCRN CLR ALWAYS TAKEN MAX Y, TOP SCREEN CLR STORE AS BOTTOM COORD S RIGHTMOST X-COORD (COLUMN) TOP COORD FOR VLINE CALLS CLEAR COLOR (BLACK) DRAW VLINE NEXT LEFTMOST X-COORD LOOP UNTIL DONE FOR INPUT 000DEFGH
F80A: 69 E0 F80C: 85 2E F80E: B1 26 F811: 25 2E F814: 51 26 F814: 51 26 F812: 25 2E F814: 51 26 F815: 20 00 F818: 60 F819: 20 00 F812: 20 02 F821: 20 02 F822: 20 00 F822: 68 F822: 20 00 F825: 90 F834: 00 F834: 00 F834: 20 F834: 20 F834: 20 F835: 30 F843: 88 F844: 10 F843: 88 F844: 10 F844: 10 F844: 40 F844: 41 F844: 42	RTMASK PLOT1 HLINE HLINE1 VLINEZ VLINEZ CLRSCR CLRSC2 CLRSC3 GBASCALC	ADC STA LDA EOR AND EOR STA RTS JSR BCC ADC ADC PHA JSR PLA CMP BCC RTS LDY ENE LDY STY LDY ENE LDY STY JSR BRE LDY STY STA JSR DEY BPLA STA JSR DEY BPLA STA JSR DEY BNE LDY STA JSR ADD STA JSR DEY BA STA STA JSR DEY BA STA STA STA STA STA STA STA STA STA ST	<pre>#\$E0 MASK (GBASL),Y COLOR MASK (GBASL),Y (GBASL),Y PLOT H2 RTS1 PLOT HLINE1 #\$01 PLOT V2 VLINEZ #\$2F CLRSC2 #\$27 V2 VLINE CALLS #\$27 #\$00 COLOR VLINE CLRSC3 #\$03 #\$04 GBASH #\$18 GBCALC</pre>	DATA EOR COLOR AND MASK XOR DATA TO DATA PLOT SQUARE DONE? YES, RETURN NO, INCR INDEX (X-COORD) PLOT NEXT SQUARE ALWAYS TAKEN NEXT Y-COORD SAVE ON STACK PLOT SQUARE DONE? NO, LOOP MAX Y, FULL SCRN CLR ALWAYS TAKEN MAX Y, FULL SCRN CLR ALWAYS TAKEN MAX Y, TOP SCREEN CLR STORE AS BOTTOM COORD S RIGHTMOST X-COORD (COLUMN) TOP COORD FOR VLINE CALLS CLEAR COLOR (BLACK) DRAW VLINE NEXT LEFTMOST X-COORD LOOP UNTIL DONE FOR INPUT 000DEFGH
F80A: 69 E0 F80C: 85 2E F80E: B1 26 F811: 51 25 2E F814: 51 26 F814: 50 F8 F812: 20 00 F8 F821: 20 02 F8 F822: 20 00 F8 F822: 20 00 F8 F822: 20 00 F8 F822: 20 00 F8 F822: 20 02 F8 F831: 60 27 F838: F834: 20 27 F835: F834: 20 28 F8 F844: 10 F6 F845: F844: 20 28<	RTMASK PLOT1 HLINE HLINE1 VLINEZ VLINE CLRSCR CLRSCR CLRSC3 GBASCALC	ADC STA LDA EOR AND EOR STA RTS JSR CCPY BCS INY JSR BCC ADC PHA JSR PHA CMP BCC RTS LDY BNE LDY STY FOR STY FOR STY STY FOR STY STY STY ADC STA STA AND ORA STA AND ORA STA	<pre>#\$E0 MASK (GBASL),Y COLOR MASK (GBASL),Y (GBASL),Y PLOT H2 RTS1 PLOT1 HLINE1 #\$01 PLOT V2 VLINE2 #\$2F CLRSC2 #\$27 V2 VLINE CALLS #\$27 #\$00 COLOR VLINE CLRSC3 #\$03 #\$04 GBASH #\$18 GBCALC #\$7F</pre>	DATA EOR COLOR AND MASK XOR DATA TO DATA PLOT SQUARE DONE? YES, RETURN NO, INCR INDEX (X-COORD) PLOT NEXT SQUARE ALWAYS TAKEN NEXT Y-COORD SAVE ON STACK PLOT SQUARE DONE? NO, LOOP MAX Y, FULL SCRN CLR ALWAYS TAKEN MAX Y, FULL SCRN CLR ALWAYS TAKEN MAX Y, TOP SCREEN CLR STORE AS BOTTOM COORD S RIGHTMOST X-COORD (COLUMN) TOP COORD FOR VLINE CALLS CLEAR COLOR (BLACK) DRAW VLINE NEXT LEFTMOST X-COORD LOOP UNTIL DONE FOR INPUT 000DEFGH

F858:						A	
F859:	0A				ASL	A	
F85A:	05	26			ORA	GBASL	
F85C:					STA		
		20				GDADD	
F85E:					RTS		
		30		NXTCOL		COLOR	INCREMENT COLOR BY 3
F861:	18				CLC		
F862:	69	03			ADC	#\$03	
F864:	29	0F		SETCOL	AND	#\$0F	SETS COLOR=17*A MOD 16
F866:						COLOR	
F868:		50					DOMU HALE DUMES OF GOLOD BOHAL
					ASL		BOTH HALF BYTES OF COLOR EQUAL
F869:					ASL		
F86A:	0A				ASL	A	
F86B:	0A				ASL	A	
F86C:	05	30			ORA	COLOR	
F86E:	85	30			STA	COLOR	
F870:					RTS		
				SCRN		7	READ SCREEN Y-COORD/2
F872:			-		PHP		SAVE LSB (CARRY)
F873:							CALC BASE ADDRESS
F876:	В1	26			LDA	(GBASL),Y	GET BYTE
F878:	28				PLP		RESTORE LSB FROM CARRY
F879:	90	04		SCRN2	BCC	RTMSKZ	IF EVEN, USE LO H
F87B:						A	
F87C:					LSR		
							CUITER UTCU UNTE DURE DOWN
F87D:					LSR		SHIFT HIGH HALF BYTE DOWN
F87E:					LSR		
		0F		RTMSKZ	AND	#\$0F	MASK 4-BITS
F881:	60				RTS		
F882:	A6	3A		INSDS1	LDX	PCL	PRINT PCL,H
F884:					LDY		
F886:						PRYX2	
F889:							FOLLOWED BY A BLANK
F88C:	A1					(PCL,X)	GET OP CODE
F88E:	A8			INSDS2	TAY		
F88F:	4A				LSR	A	EVEN/ODD TEST
F890:	90	09				IEVEN	
F892:					ROR		BIT 1 TEST
		10					
F893:					BCS		XXXXXX11 INVALID OP
F895:					CMP		
F897:	FO	0C			BEQ		OPCODE \$89 INVALID
F899:	29	87			AND	#\$87	MASK BITS
F89B:	4A			IEVEN	LSR	A	LSB INTO CARRY FOR L/R TEST
F89C:					TAX		· · · · , · ·
F89D:			FO			EMT1 V	GET FORMAT INDEX BYTE
F8A0:							R/L H-BYTE ON CARRY
F8A3:						GETFMT	
FONE.	A0	80		ERR	LDY	#\$80	SUBSTITUTE \$80 FOR INVALID OPS
FORD:					LDA		SET PRINT FORMAT INDEX TO 0
F8A7:	A9			GETFMT			
F8A7:					TAX		
F8A7: F8A9:	AA					FMT2 X	INDEX INTO PRINT FORMAT TABLE
F8A7: F8A9: F8AA:	AA BD	A6			LDA		INDEX INTO PRINT FORMAT TABLE
F8A7: F8A9: F8AA: F8AD:	AA BD 85	A6 2E			LDA STA	FORMAT	SAVE FOR ADR FIELD FORMATTING
F8A7: F8A9: F8AA:	AA BD 85	A6 2E			LDA STA AND	FORMAT #\$03	SAVE FOR ADR FIELD FORMATTING MASK FOR 2-BIT LENGTH
F8A7: F8A9: F8AA: F8AD:	AA BD 85	A6 2E			LDA STA AND	FORMAT #\$03	SAVE FOR ADR FIELD FORMATTING
F8A7: F8A9: F8AA: F8AD:	AA BD 85 29	A6 2E 03			LDA STA AND	FORMAT #\$03	SAVE FOR ADR FIELD FORMATTING MASK FOR 2-BIT LENGTH
F8A7: F8A9: F8AA: F8AD: F8AF:	AA BD 85 29 85	A6 2E 03			LDA STA AND	FORMAT #\$03 (P=1 BYTE, 3	SAVE FOR ADR FIELD FORMATTING MASK FOR 2-BIT LENGTH
F8A7: F8A9: F8AA: F8AD: F8AF: F8B1:	AA BD 85 29 85 98	A6 2E 03 2F	F9		LDA STA AND STA TYA	FORMAT #\$03 (P=1 BYTE, 3	SAVE FOR ADR FIELD FORMATTING MASK FOR 2-BIT LENGTH 1=2 BYTE, 2=3 BYTE)
F8A7: F8A9: F8AA: F8AD: F8AF: F8B1: F8B3: F8B4:	AA BD 85 29 85 98 29	A6 2E 03 2F 8F	F9		LDA STA AND STA TYA AND	FORMAT #\$03 (P=1 BYTE, : LENGTH	SAVE FOR ADR FIELD FORMATTING MASK FOR 2-BIT LENGTH 1=2 BYTE, 2=3 BYTE) OPCODE MASK FOR 1XXX1010 TEST
F8A7: F8A9: 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	FORMAT #\$03 (P=1 BYTE, : LENGTH	SAVE FOR ADR FIELD FORMATTING MASK FOR 2-BIT LENGTH 1=2 BYTE, 2=3 BYTE) OPCODE MASK FOR 1XXX1010 TEST SAVE IT
F8A7: F8A9: F8AA: F8AD: F8AF: F8B1: F8B3: F8B4: F8B6: F8B7:	AA BD 85 29 85 98 29 AA 98	A6 2E 03 2F 8F	F9		LDA STA AND STA TYA AND TAX TYA	FORMAT #\$03 (P=1 BYTE, 1 LENGTH #\$8F	SAVE FOR ADR FIELD FORMATTING MASK FOR 2-BIT LENGTH 1=2 BYTE, 2=3 BYTE) OPCODE MASK FOR 1XXX1010 TEST
F8A7: F8A9: F8AA: F8AD: F8AF: F8B1: F8B1: F8B4: F8B4: F8B6: F8B7: F8B8:	AA BD 85 29 85 98 29 AA 98 A0	A6 2E 03 2F 8F	F9		LDA STA AND STA TYA AND TAX TYA LDY	FORMAT #\$03 P=1 BYTE, : LENGTH #\$8F #\$03	SAVE FOR ADR FIELD FORMATTING MASK FOR 2-BIT LENGTH 1=2 BYTE, 2=3 BYTE) OPCODE MASK FOR 1XXX1010 TEST SAVE IT
F8A7: F8A9: F8AA: F8AD: F8AF: F8B1: F8B1: F8B4: F8B4: F8B6: F8B7: F8B8: F8BA:	AA BD 85 29 85 98 29 AA 98 A0 E0	A6 2E 03 2F 8F 03 8A	F9		LDA STA AND STA TYA AND TAX TYA LDY CPX	FORMAT #\$03 P=1 BYTE, : LENGTH #\$8F #\$03 #\$8A	SAVE FOR ADR FIELD FORMATTING MASK FOR 2-BIT LENGTH 1=2 BYTE, 2=3 BYTE) OPCODE MASK FOR 1XXX1010 TEST SAVE IT
F8A7: F8A9: F8AA: F8AD: F8AF: F8B1: F8B1: F8B4: F8B4: F8B6: F8B7: F8B8:	AA BD 85 29 85 98 29 AA 98 A0 E0	A6 2E 03 2F 8F 03 8A	F9		LDA STA AND STA TYA AND TAX TYA LDY CPX BEQ	FORMAT #\$03 P=1 BYTE, : LENGTH #\$8F #\$03 #\$8A MNNDX3	SAVE FOR ADR FIELD FORMATTING MASK FOR 2-BIT LENGTH 1=2 BYTE, 2=3 BYTE) OPCODE MASK FOR 1XXX1010 TEST SAVE IT
F8A7: F8A9: F8AA: F8AD: F8AF: F8B1: F8B1: F8B4: F8B4: F8B6: F8B7: F8B8: F8BA:	AA BD 85 29 85 98 29 AA 98 A0 E0 F0	A6 2E 03 2F 8F 03 8A 03	F9		LDA STA AND STA TYA AND TAX TYA LDY CPX	FORMAT #\$03 P=1 BYTE, : LENGTH #\$8F #\$03 #\$8A MNNDX3	SAVE FOR ADR FIELD FORMATTING MASK FOR 2-BIT LENGTH 1=2 BYTE, 2=3 BYTE) OPCODE MASK FOR 1XXX1010 TEST SAVE IT
F8A7: F8A9: F8AA: F8A7: F8AF: F8B1: F8B4: F8B4: F8B6: F8B6: F8B8: F8BA: F8BC:	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	FORMAT #\$03 P=1 BYTE, : LENGTH #\$8F #\$03 #\$8A MNNDX3 A	SAVE FOR ADR FIELD FORMATTING MASK FOR 2-BIT LENGTH 1=2 BYTE, 2=3 BYTE) OPCODE MASK FOR 1XXX1010 TEST SAVE IT
F8A7: F8A9: F8AA: F8A7: F8AF: F8B1: F8B4: F8B4: F8B6: F8B6: F8B8: F8B4: F8B2: F8B2:	AA BD 85 29 85 98 29 AA 98 A0 E0 F0 4A 90	A6 2E 03 2F 8F 03 8A 0B 08	F9		LDA STA AND STA TYA AND TAX TYA LDY CPX BEQ LSR	FORMAT #\$03 P=1 BYTE, : LENGTH #\$8F #\$03 #\$8A MNNDX3 A MNNDX3	SAVE FOR ADR FIELD FORMATTING MASK FOR 2-BIT LENGTH 1=2 BYTE, 2=3 BYTE) OPCODE MASK FOR 1XXX1010 TEST SAVE IT OPCODE TO A AGAIN
F8A7: F8A9: F8A7: F8A7: F8B1: F8B4: F8B4: F8B4: F8B6: F8B7: F8B8: F8B8: F8B2: 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 LDY LDY CPX BEQ LSR BCC LSR	FORMAT #\$03 P=1 BYTE, : LENGTH #\$8F #\$03 #\$8A MNNDX3 A MNNDX3 A	SAVE FOR ADR FIELD FORMATTING MASK FOR 2-BIT LENGTH 1=2 BYTE, 2=3 BYTE) OPCODE MASK FOR 1XXX1010 TEST SAVE IT OPCODE TO A AGAIN FORM INDEX INTO MNEMONIC TABLE
F8A7: F8A9: F8A7: F8A7: F8B1: F8B4: F8B4: F8B4: F8B4: F8B7: F8B8: F8B4: F8B5: F8B5: F8B5: F8B5: F822:	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 AND TYA AND TAX TYA LDY CPX BEQ LSR BCC LSR LSR	FORMAT #\$03 P=1 BYTE, : LENGTH #\$8F #\$03 #\$8A MNNDX3 A MNNDX3 A A	SAVE FOR ADR FIELD FORMATTING MASK FOR 2-BIT 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
F8A7: F8A9: F8A7: F8AF: F8B1: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B2: F8B2: F8B5: F8B5: F8C1: F8C2: F8C2: F8C3:	AA BD 85 29 85 98 29 AA 98 A0 E0 F0 4A 90 4A 4A 09	A6 2E 03 2F 8F 03 8A 0B 08	F9	MNNDX1	LDA STA AND TYA AND TAX TYA LDY CPX BEQ LSR BCC LSR LSR ORA	FORMAT #\$03 P=1 BYTE, : LENGTH #\$8F #\$03 #\$8A MNNDX3 A MNNDX3 A	SAVE FOR ADR FIELD FORMATTING MASK FOR 2-BIT 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
F8A7: F8A9: F8A1: F8A1: F8A7: F8B4: F8B4: F8B4: F8B4: F8B7: F8B7: F8B7: F8B7: F8B7: F8B7: F8B7: F8B7: F8B7: F827: F8C2: F8C5:	AA BD 85 29 85 98 29 AA 98 A0 F0 4A 90 4A 4A 09 88	A6 2E 03 2F 8F 03 8A 0B 08 08	F9	MNNDX1	LDA STA AND STA TYA AND TAX LDY CPX BEQ LSR BCC LSR LSR LSR DEY	FORMAT #\$03 P=1 BYTE, : LENGTH #\$8F #\$03 #\$8A MNNDX3 A MNNDX3 A A #\$20	SAVE FOR ADR FIELD FORMATTING MASK FOR 2-BIT 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
F8A7: F8A9: F8AD: F8AD: F8AF: F8B4: F8B4: F8B4: F8B4: F8B4: F8B5: F8B5: F8B5: F8B5: F8B5: F8B5: F8C1: F8C2: F8C5: F8C5: F8C5:	AA BD 85 29 85 98 29 AA 98 A0 E0 F0 4A 90 4A 40 88 D0	A6 2E 03 2F 8F 03 8A 0B 08 08 20 FA	F9	MNNDX1	LDA STA AND STA TYA AND TAX TYA LDY CPX BEQ LSR BEQ LSR BCC LSR DEY BNE	FORMAT #\$03 P=1 BYTE, : LENGTH #\$8F #\$03 #\$8A MNNDX3 A MNNDX3 A A	SAVE FOR ADR FIELD FORMATTING MASK FOR 2-BIT 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) XXXYYY01->00110XXX 4) XXXYY100->00100XXX
F8A7: F8A9: F8A5: F8A7: F8B6: F8B4: F8B4: F8B4: F8B4: F8B7: F8B7: F8B7: F8B7: F8B7: F8B7: F8B7: F8B7: F8B7: F8C2: F8C3: F8C5: F8C6: F8C8: F8C8:	AA BD 85 29 85 98 29 AA 98 A0 E0 F0 4A 90 4A 40 88 D0 C8	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 BEQ LSR BECC LSR LSR ORA DEY ENE INY	FORMAT #\$03 P=1 BYTE, : LENGTH #\$8F #\$03 #\$8A MNNDX3 A MNNDX3 A A #\$20	SAVE FOR ADR FIELD FORMATTING MASK FOR 2-BIT 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
F8A7: F8A9: F8AD: F8AD: F8AF: F8B4: F8B4: F8B4: F8B4: F8B4: F8B5: F8B5: F8B5: F8B5: F8B5: F8B5: F8C1: F8C2: F8C5: F8C5: F8C5:	AA BD 85 29 85 98 29 AA 98 A0 E0 F0 4A 90 4A 40 88 D0 C8	A6 2E 03 2F 8F 03 8A 0B 08 08 20 FA	F9	MNNDX1	LDA STA AND STA TYA AND TAX TYA LDY CPX BEQ LSR BEQ LSR BECC LSR LSR ORA DEY ENE INY	FORMAT #\$03 P=1 BYTE, : LENGTH #\$8F #\$03 #\$8A MNNDX3 A MNNDX3 A A #\$20	SAVE FOR ADR FIELD FORMATTING MASK FOR 2-BIT 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) XXXYYY01->00110XXX 4) XXXYY100->00100XXX
F8A7: F8A9: F8A5: F8A7: F8B6: F8B4: F8B4: F8B4: F8B4: F8B7: F8B7: F8B7: F8B7: F8B7: F8B7: F8B7: F8B7: F8B7: F8C2: F8C3: F8C5: F8C6: F8C8: F8C8:	AA BD 85 29 85 98 29 AA 98 A0 E0 F0 4A 90 4A 4A 09 88 D0 C8 88	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 BEQ LSR LSR LSR LSR CPX BEQ LSR LSR LSR INY DEY	FORMAT #\$03 P=1 BYTE, : LENGTH #\$8F #\$03 #\$8A MNNDX3 A MNNDX3 A A #\$20	SAVE FOR ADR FIELD FORMATTING MASK FOR 2-BIT 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) XXXYYY01->00110XXX 4) XXXYY100->00100XXX
F8A7: F8A9: F8A9: F8A7: F8A7: F8B1: F8B4: F8B4: F8B4: F8B7: F8B4: F8B7: F8B8: F8B7: F8B8: F8B7: F8B7: F8C1: F8C2: F8C2: F8C3: F8C5: F8C6: F8C8: F8C9: F8C9:	AA BD 85 29 85 98 29 AA 98 A0 E0 F0 4A 90 4A 4A 09 88 D0 C8 88 D0	A6 2E 03 2F 8F 03 8A 0B 08 20 FA	F9	MNNDX1 MNNDX2	LDA STA AND STA TYA AND TAX LDY CPX BEQ LSR BEQ LSR BEQ LSR BEC LSR USR DEY BNE	FORMAT #\$03 P=1 BYTE, : LENGTH #\$8F #\$03 #\$8A MNNDX3 A MNNDX3 A A #\$20 MNNDX2	SAVE FOR ADR FIELD FORMATTING MASK FOR 2-BIT 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) XXXYYY01->00110XXX 4) XXXYY100->00100XXX
F8A7: F8A9: F8A1: F8A1: F8A7: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B5: F8B5: F8C1: F8C2: F8C3: F8C4: F8C9: F8C9: F8C9: F8C2: F8C2: F8C2:	AA BD 85 29 85 98 29 AA 98 A0 E0 F0 4A 90 4A 4A 09 88 D0 C8 88 D0 60	A6 2E 03 2F 8F 03 8A 0B 08 20 FA F2	F9	MNNDX1 MNNDX2 MNNDX3	LDA STA AND STA TYA AND TAX LDY CPX BEQ CPX BEQ LSR BCQ LSR BCQ LSR BCQ LSR DEY BNE INY DEY ENE ENE RTS	FORMAT #\$03 (P=1 BYTE, : LENGTH #\$8F #\$03 #\$8A MNNDX3 A MNNDX3 A #\$20 MNNDX2 MNNDX1	SAVE FOR ADR FIELD FORMATTING MASK FOR 2-BIT 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->0000XXXXX
F8A7: F8A9: F8A9: F8A1: F8A7: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B5: F8B5: F8C1: F8C2: F8C2: F8C2: F8C3: F8C5: F8C9: F8C4: F8C2: F8C4: F8C4: F8C4: F8C5:	AA BD 85 29 85 98 29 AA 98 A0 F0 4A 90 4A 4A 09 88 D0 C8 88 D0 C8 88 D0 FF	A6 2E 03 2F 8F 03 8A 0B 08 20 FA F2 FF	F9	MNNDX1 MNNDX2 MNNDX3	LDA STA AND STA TYA AND TAX TYA LDY CPX BEQ LSR BCC LSR DCY BNE LSR ORA DEY BNE INY BNE RTS DFB	FORMAT #\$03 P=1 BYTE, : LENGTH #\$8F #\$03 #\$8A MINNDX3 A MINNDX3 A #\$20 MINNDX2 MINNDX1 \$FF,\$FF,\$FT	SAVE FOR ADR FIELD FORMATTING MASK FOR 2-BIT 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->000XXXXX
F8A7: F8A9: F8A1: F8A1: F8A7: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B5: F8B5: F8C2: F8C2: F8C2: F8C4: F8C5: F8C4: F8C9: F8C4: F8C2: F8C4: F8C2: F8C4: F8C5: F8C4: F8C5:	AA BD 85 29 AA 98 29 AA 90 F0 4A 90 4A 4A 90 88 80 C8 88 D0 C8 88 D0 FF 20	A6 2E 03 2F 8F 03 8A 0B 08 08 20 FA F2 FF	F9	MNNDX1 MNNDX2 MNNDX3	LDA STA AND STA TYA AND TAX TYA LDY CPX BEQ LSR BEQ LSR BEQ LSR BEQ LSR BEQ LSR BEQ LSR BEQ LSR BEQ LSR BEQ LSR BEQ LSR BEQ LSR STA STA TYA AND TAX TYA AND TAX TYA AND TAX TYA AND TAX TYA BEQ LSR BEQ DEY BNE INY DEY BNE STA STA AND TAX TYA AND TAX TYA AND TAX TYA AND TAX TYA AND TAX TYA AND TAX TYA AND TAX TYA CPX BEQ DY STA STA AND TAX TYA CPX BEQ DY STA STA TAX TYA CPX BEQ DY STA STA STA STA AND TAX TYA AND TAX TYA CPX BEQ DY STA STA STA TAX TYA CPX BEQ DY STA STA STA TAX TYA CPX BEQ DY STA STA STA TAX TYA STA TAX TYA BEQ DY STA STA STA TAX TYA STA STA TAX TYA STA TAX TYA STA STA STA TAX TYA STA STA STA STA STA STA STA STA STA ST	FORMAT #\$03 P=1 BYTE, : LENGTH #\$8F #\$03 #\$8A MINNDX3 A MINNDX3 A #\$20 MINNDX2 MINNDX1 \$FF,\$FF,\$FT	SAVE FOR ADR FIELD FORMATTING MASK FOR 2-BIT 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->000XXXXX
F8A7: F8A9: F8A9: F8A7: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B5: F8B5: F825: F8C5: F8C5: F8C6: F8C2: F8C3:	AA BD 85 29 AA 98 29 AA 98 A0 F0 4A 90 4A 40 98 80 00 60 FF 20 48	A6 2E 03 2F 8F 03 8A 0B 08 20 FA F2 F2 F7 82	F9 FF F8	MNNDX1 MNNDX2 MNNDX3 INSTDSP	LDA STA AND STA TYA AND TAX TYA LDY CPX BEQ LSR BEQ LSR BEQ LSR BEQ LSR BEQ LSR BEQ LSR BEQ LSR BEQ LSR BEQ LSR BEQ LSR CPX BEQ LSR BEQ DEY STA TAX TYA	FORMAT #\$03 P=1 BYTE, :: LENGTH #\$8F #\$03 #\$8A MNNDX3 A MNNDX3 A A #\$20 MNNDX2 MNNDX1 \$FF, \$FF, \$FI INSDS1	SAVE FOR ADR FIELD FORMATTING MASK FOR 2-BIT 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->000XXXXX
F8A7: F8A9: F8A9: F8A7: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B5: F8B5: F825: F8C5: F8C5: F8C6: F8C2: F8C3:	AA BD 85 29 AA 98 29 AA 98 A0 F0 4A 90 4A 40 98 80 00 60 FF 20 48	A6 2E 03 2F 8F 03 8A 0B 08 20 FA F2 F2 F7 82	F9 FF F8	MNNDX1 MNNDX2 MNNDX3 INSTDSP	LDA STA AND STA TYA AND TAX TYA LDY CPX BEQ LSR BEQ LSR BEQ LSR BEQ LSR BEQ LSR BEQ LSR BEQ LSR BEQ LSR BEQ LSR BEQ LSR CPX BEQ LSR BEQ DEY STA TAX TYA	FORMAT #\$03 P=1 BYTE, : LENGTH #\$8F #\$03 #\$8A MINNDX3 A MINNDX3 A #\$20 MINNDX2 MINNDX1 \$FF,\$FF,\$FT	SAVE FOR ADR FIELD FORMATTING MASK FOR 2-BIT 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->000XXXXX
F8A7: F8A9: F8A9: F8A7: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B5: F8B5: F825: F8C5: F8C5: F8C6: F8C2: F8C3:	AA BD 85 29 AA 85 29 AA 85 29 AA 80 40 40 40 40 40 88 D0 88 D0 88 D0 88 D0 88 D0 88 D0 88 D0 88 D0 88 D0 88 D0 88 D0 80 80 80 80 80 80 80 80 80 80 80 80 80	A6 2E 03 2F 8F 03 8A 08 08 20 FA F2 F2 F2 3A	F9 FF F8	MNNDX1 MNNDX2 MNNDX3 INSTDSP	LDA STA AND STA TYA AND TAX LDY CPX BEQ LSR BEQ LSR BEQ LSR BEC LSR USR DEY BNE RTS DFB JSR PHA LDA	FORMAT #\$03 P=1 BYTE, :: LENGTH #\$8F #\$03 #\$8A MNNDX3 A MNNDX3 A A #\$20 MNNDX2 MNNDX1 \$FF, \$FF, \$FI INSDS1	SAVE FOR ADR FIELD FORMATTING MASK FOR 2-BIT 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->000XXXXX
F8A7: F8A9: F8A9: F8A1: F8A7: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B5: F8C2: F8C4: F8C5: F8C4:	AA BD 85 29 85 29 85 29 80 80 80 80 80 80 80 80 88 D0 60 88 D0 60 FF 20 48 B1 20	A6 2E 03 2F 8F 03 8A 0B 08 20 FA F2 F2 F2 SA DA	F9 FF F8 FD	MNNDX1 MNNDX2 MNNDX3 INSTDSP PRNTOP	LDA STA AND STA TYA AND TAX LDY CPX BEQ CPX BEQ CPX BEQ LSR BCC LSR BCC LSR ULSR USR USR USR DEY BNE RTS DFB JSR PHA LDA JSR	FORMAT #\$03 (P=1 BYTE, : LENGTH #\$8F #\$03 #\$8A MNNDX3 A #\$20 MNNDX3 A #\$20 MNNDX1 \$FF,\$FF,\$FJ INSDS1 (PCL),Y PRBYTE	SAVE FOR ADR FIELD FORMATTING MASK FOR 2-BIT 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 F GEN FMT, LEN BYTES SAVE MNEMONIC TABLE INDEX
F8A7: F8A9: F8A9: F8A1: F8A7: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B5: F8B5: F8B5: F8C1: F8C2: F8C5: F8C5: F8C5: F8C6: F8C9: F8C4: F8C2: F8C4: F8C5:	AA BD 85 29 85 29 A0 E0 F0 4A 90 4A 90 4A 90 88 D0 C8 88 D0 C8 88 D0 60 FF 20 48 L0 20 20 20 20 20 20 20 20 20 20 20 20 20	A6 2E 03 2F 8F 03 8A 0B 08 20 FA F2 F2 F2 3A DA 01	F9 FF F8 FD	MNNDX1 MNNDX2 MNNDX3 INSTDSP PRNTOP	LDA STA AND STA TYA AND TAX LDY CPX BEQ LSR BCC LSR BCC LSR BCC LSR BCC LSR BCC LSR DFB JSR LDA JSR LDA JSR LDX	FORMAT #\$03 (P=1 BYTE, : LENGTH #\$8F #\$03 #\$8A MINNDX3 A MINNDX3 A #\$20 MINNDX1 \$FF,\$FF,\$FI INSDS1 (PCL),Y PRBYTE #\$01	SAVE FOR ADR FIELD FORMATTING MASK FOR 2-BIT 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->000XXXXX
F8A7: F8A9: F8A1: F8A1: F8A7: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B5: F8B5: F8C1: F8C2: F8C2: F8C3: F8C4: F8C5: F8C4: F8C5: F8C4: F8C5:	AA BD 85 29 AA 98 A0 F0 40 40 40 40 88 D0 60 FF 20 48 BD 60 FF 20 48 20 20	A6 2E 03 2F 8F 03 8A 0B 08 20 FA F2 FF 82 3A 01 4A	F9 FF F8 FD	MNNDX1 MNNDX2 MNNDX3 INSTDSP PRNTOP	LDA STA AND STA TYA AND TAX TYA LDY CPX BEQ LSR BCC LSR BCC LSR DEY BNE INY BNE RTS DFB JSR PHA LDA JSR LDX JSR	FORMAT #\$03 (P=1 BYTE, : LENGTH #\$8F #\$03 #\$8A MINNDX3 A MINNDX3 A #\$20 MINNDX1 \$FF,\$FF,\$FI INSDS1 (PCL),Y PRBYTE #\$01 PRBL2	SAVE FOR ADR FIELD FORMATTING MASK FOR 2-BIT 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->000XXXXX F GEN FMT, LEN BYTES SAVE MNEMONIC TABLE INDEX PRINT 2 BLANKS
F8A7: F8A9: F8A1: F8A1: F8A1: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B5: F8B5: F8C2: F8C3: F8C4: F8C5: F8C4: F8C4: F8C4: F8C4: F8C5: F8C4: F8C5: F8C6: F8C5:	AA BD 85 29 AA 98 A0 F0 4A 40 88 D0 60 FF 20 48 BD 60 FF 20 48 B1 20 C4	A6 2E 03 2F 8F 03 8A 0B 08 20 FA F2 FF 82 3DA 01 4A 2F	F9 FF F8 FD	MNNDX1 MNNDX2 MNNDX3 INSTDSP PRNTOP	LDA STA AND STA TYA AND TAX TYA LDY CPX BEQ LSR BCC LSR BCC LSR BCC LSR BCC LSR BCC LSR BCC LSR BCC LSR BCC LSR BCC LSR BCC LSR BCC LSR DEY BNE INY DEY BNE INY DFB JSR LDA JSR CPY	FORMAT #\$03 (P=1 BYTE, : LENGTH #\$8F #\$03 #\$8A MINNDX3 A MINNDX3 A #\$20 MINNDX1 \$FF,\$FF,\$FI INSDS1 (PCL),Y PRBYTE #\$01 PRBL2	SAVE FOR ADR FIELD FORMATTING MASK FOR 2-BIT 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->00110XXX 4) XXXYY10->00100XXX 5) XXXXX000->00100XXX 5) XXXXX000->0000XXXX 5) XXXXX000->000XXXXX F GEN FMT, LEN BYTES SAVE MNEMONIC TABLE INDEX PRINT 2 BLANKS PRINT 1NST (1-3 BYTES)
F8A7: F8A9: F8A8: F8A7: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B5: F8B5: F8C5: F8C5: F8C62: F8C62: F8C62: F8C62: F8C62: F8C7: F8C62: F8C62: F8C62: F8C7: F8C62: F8C7: F8C7: F8C7: F8C8: F8C9: F8C	AA BD 85 29 AA 98 29 AA 98 40 40 40 40 40 88 D0 60 F 70 48 BD 20 20 48 B1 20 20 48 C 82 C 48 C 82 C 88 C 88 C 88 C 88 C	A6 2E 03 2F 8F 03 8A 0B 08 20 FA F2 FF 82 3A 01 4A 2F	F9 FF F8 FD	MNNDX1 MNNDX2 MNNDX3 INSTDSP PRNTOP	LDA STA AND STA TYA AND TAX LDY LDY CPX BEQ LSR BCQ LSR BCQ LSR BCQ LSR BCQ LSR BCQ LSR BCQ LSR BCQ LSR BCQ LSR BCQ LSR BCQ LSR DEY BNE RTS DFB JSR LDA JSR LDA LSR LDA LDA LDA LDA LDA LDA LDA LDA LDA LDA	FORMAT #\$03 (P=1 BYTE, : LENGTH #\$8F #\$03 #\$8A MNNDX3 A #\$20 MNNDX3 A MNNDX1 \$FF,\$FF,\$FI INSDS1 (PCL),Y PRBYTE #\$01 PRB12 LENGTH	SAVE FOR ADR FIELD FORMATTING MASK FOR 2-BIT 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->000XXXXX F GEN FMT, LEN BYTES SAVE MNEMONIC TABLE INDEX PRINT 2 BLANKS
F8A7: F8A9: F8A9: F8A1: F8A1: F8A1: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8C4: F8C5: F8C4:	AA BD 85 29 85 29 AA 98 40 40 40 40 40 80 60 60 60 88 00 60 88 00 60 57 20 48 80 60 67 20 48 20 20 48 50 20 48 50 20 40 50 20 40 50 20 40 50 20 40 50 20 40 50 20 40 50 20 40 50 20 40 50 20 40 50 20 40 50 20 40 50 20 40 40 50 20 40 40 50 20 40 40 50 20 40 40 50 20 40 40 50 20 40 40 50 20 40 40 50 20 40 40 50 20 40 40 50 20 40 40 50 20 40 40 50 20 40 40 50 20 40 40 20 40 40 20 40 40 20 40 40 20 40 20 40 20 40 20 40 20 40 20 40 40 20 20 40 20 40 20 40 20 20 40 20 20 40 20 20 40 20 20 40 20 20 20 40 20 20 20 20 20 20 20 20 20 20 20 20 20	A6 2E 03 2F 8F 03 8A 08 08 20 FA F2 F7 82 3A DA 01 42F F1	F9 FF F8 FD F9	MNNDX1 MNNDX2 MNNDX3 INSTDSP PRNTOP	LDA STA AND STA TYA AND TAX LDY LDY CPX BEQ LSR BCQ LSR BCQ LSR BCQ LSR BCQ LSR BCQ LSR BCQ LSR BCQ LSR BCQ LSR BCQ LSR BCQ LSR DEY BNE RTS DFB JSR LDA JSR LDA LSR LDA LDA LDA LDA LDA LDA LDA LDA LDA LDA	FORMAT #\$03 (P=1 BYTE, : LENGTH #\$8F #\$03 #\$8A MNNDX3 A #\$20 MNNDX3 A #\$20 MNNDX1 \$FF,\$FF,\$F INSDS1 (PCL),Y PRBYTE #\$01 PRBL2 LENGTH PRNTOP	SAVE FOR ADR FIELD FORMATTING MASK FOR 2-BIT 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->00110XXX 4) XXXYY10->00110XXX 4) XXXYY10->00100XXX 5) XXXXX000->00100XXX 5) XXXXX000->0000XXXX
F8A7: F8A9: F8A8: F8A7: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B5: F8B5: F8C5: F8C5: F8C62: F8C62: F8C62: F8C62: F8C62: F8C7: F8C62: F8C62: F8C62: F8C7: F8C62: F8C7: F8C7: F8C7: F8C8: F8C9: F8C	AA BD 85 29 85 29 AA 98 40 40 40 40 40 80 60 60 60 88 00 60 88 00 60 57 20 48 80 60 67 20 48 20 20 48 50 20 48 50 20 40 50 20 40 50 20 40 50 20 40 50 20 40 50 20 40 50 20 40 50 20 40 50 20 40 50 20 40 50 20 40 50 20 40 40 50 20 40 40 50 20 40 40 50 20 40 40 50 20 40 40 50 20 40 40 50 20 40 40 50 20 40 40 50 20 40 40 50 20 40 40 50 20 40 40 50 20 40 40 50 20 40 40 20 40 40 20 40 40 20 40 40 20 40 20 40 20 40 20 40 20 40 20 40 40 20 20 40 20 40 20 40 20 20 40 20 20 40 20 20 40 20 20 40 20 20 20 40 20 20 20 20 20 20 20 20 20 20 20 20 20	A6 2E 03 2F 8F 03 8A 08 08 20 FA F2 F7 82 3A DA 01 42F F1	F9 FF F8 FD F9	MNNDX1 MNNDX2 MNNDX3 INSTDSP PRNTOP PRNTBL	LDA STA AND STA TYA AND TAX LDY LDY CPX BEQ LSR BCQ CPX LSR CPY LSR DFB USY DEY BNE RTS DFB JSR PHA LDA JSR LDX LDX CPY LDX CPY LDX CPY LDX	FORMAT #\$03 (P=1 BYTE, : LENGTH #\$8F #\$03 #\$8A MNNDX3 A #\$20 MNNDX3 A #\$20 MNNDX1 \$FF,\$FF,\$F INSDS1 (PCL),Y PRBYTE #\$01 PRBL2 LENGTH PRNTOP	SAVE FOR ADR FIELD FORMATTING MASK FOR 2-BIT 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->00110XXX 4) XXXYY10->00100XXX 5) XXXXX000->00100XXX 5) XXXXX000->0000XXXX 5) XXXXX000->000XXXXX F GEN FMT, LEN BYTES SAVE MNEMONIC TABLE INDEX PRINT 2 BLANKS PRINT 1NST (1-3 BYTES)
F8A7: F8A9: F8A9: F8A1: F8A1: F8A1: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8B4: F8C4: F8C5: F8C4:	AA BD 85 29 85 29 AA 98 29 AA 98 29 40 F0 40 40 88 D0 60 FF 20 88 D0 60 FF 20 88 20 C8 80 C8 20 C8 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 A 20 AA AA 20 AA 20 AA 20 A 20 A AA 20 A AA 20 AA AA 20 AA AA AA AA AA AA AA AA AA AA AA AA AA	A6 2E 03 2F 8F 03 8A 08 08 20 FA F2 FF 82 3A 01 4A 2F 51 03	F9 FF F8 FD F9	MNNDX1 MNNDX2 MNNDX3 INSTDSP PRNTOP PRNTBL	LDA STA AND STA TYA AND TAX LDY LDY CPX BEQ LSR BCQ CPX LSR CPY LSR DFB USY DEY BNE RTS DFB JSR PHA LDA JSR LDX LDX CPY LDX CPY LDX CPY LDX	FORMAT #\$03 P=1 BYTE, : LENGTH #\$8F #\$03 #\$8A MNNDX3 A #\$20 MNNDX3 A MNNDX1 \$FF,\$FF,\$FI INSDS1 (PCL),Y PRBYTE #\$01 PRBL2 LENGTH PRNTOP #\$03	SAVE FOR ADR FIELD FORMATTING MASK FOR 2-BIT 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->00110XXX 4) XXXYY10->00110XXX 4) XXXYY10->00100XXX 5) XXXXX000->00100XXX 5) XXXXX000->0000XXXX

POP7 .	0.0	ΕЭ			DCC	PRNTBL	
F8E7: F8E9:					PLA		RECOVER MNEMONIC INDEX
F8EA:					TAY		RECOVER MINEMONIC INDEX
F8EB:			τo			MNEMT V	
F8EE:					STA	MNEML,Y LMNEM MNEMR,Y	FETCH 3-CHAR MNEMONIC
F8F0:					TDA	MNEMD V	(PACKED IN 2-BYTES)
F8F3:					STA	RMNEM	(FACKED IN Z-BIIES)
				PRMN1			
F8F7:					LDY		
							SHIFT 5 BITS OF
F8FB:	26	20		PRMN2	ROL	LMNEM	CHARACTER INTO A
F8FD:					ROL	DUINISH	(CLEARS CARRY)
F8FE:					DEY		(ellinde ender)
F8FF:					BNF	PRMN2	
F901:					ADC		ADD "?" OFFSET
F903:					TSR	COUT	OUTPUT A CHAR OF MNEM
F906:					DEX	0001	
F907:						PRMN1	
F909:							OUTPUT 3 BLANKS
F90C:						LENGTH	
F90E:							CNT FOR 6 FORMAT BITS
F910:				PRADR1			
F912:							IF X=3 THEN ADDR.
F914:	06	2E		PRADR2	ASL	FORMAT	
F916:	90	0E			BCC	PRADR3	
F918:	BD	В3	F9		LDA	PRADR3 CHAR1-1,X COUT	
F91B:	20	ED	FD		JSR	COUT	
F91E:	BD	В9	F9		LDA	CHAR2-1,X	
F921:	FO	03				PRADR3	
F923:	20	ED	FD		JSR	COUT	
F926:	CA			PRADR3	DEX		
F927:	D0	E7			BNE	PRADR1	
F929:	60				RTS		
F92A:	88			PRADR4	DEY		
F92B:	30	E7				PRADR2	
F92D:						PRBYTE	
				PRADR5	LDA	FORMAT	
F932:					CMP	#\$E8	HANDLE REL ADR MODE SPECIAL (PRINT TARGET,
F934:					LDA	(PCL),Y	SPECIAL (PRINT TARGET,
F936:						PRADR4	NOT OFFSET)
			F9	RELADR			
F93B:					TAX		PCL, PCH+OFFSET+1 TO A, Y
F93C:	E8				INX		
	-	~ ~					
F93D:					BNE	PRNTYX	+1 TO Y,X
F93D: F93F:	C8				BNE INY	PRNTYX	+1 TO Y,X
F93D: F93F: F940:	C8 98			PRNTYX	BNE INY TYA		
F93D: F93F: F940: F941:	C8 98 20	DA	FD	PRNTYX PRNTAX	BNE INY TYA JSR		OUTPUT TARGET ADR
F93D: F93F: F940: F941: F944:	C8 98 20 8A	DA	FD	PRNTYX PRNTAX PRNTX	BNE INY TYA JSR TXA	PRBYTE	
F93D: F93F: F940: F941: F944: F945:	C8 98 20 8A 4C	DA DA	FD FD	PRNTYX PRNTAX PRNTX	BNE INY TYA JSR TXA JMP	PRBYTE PRBYTE	OUTPUT TARGET ADR OF BRANCH AND RETURN
F93D: F93F: F940: F941: F944: F945:	C8 98 20 8A 4C	DA DA	FD FD	PRNTYX PRNTAX PRNTX	BNE INY TYA JSR TXA JMP	PRBYTE PRBYTE	OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT
F93D: F93F: F940: F941: F944: F945: F948: F94A:	C8 98 20 8A 4C A2 A9	DA DA 03 A0	FD FD	PRNTYX PRNTAX PRNTX PRBLNK PRBL2	BNE INY TYA JSR TXA JMP LDX LDA	PRBYTE PRBYTE #\$03 #\$A0	OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE
F93D: F93F: F940: F941: F944: F945: F948: F94A: F94C:	C8 98 20 8A 4C A2 A9 20	DA DA 03 A0 ED	FD FD	PRNTYX PRNTAX PRNTX	BNE INY TYA JSR TXA JMP LDX LDA JSR	PRBYTE PRBYTE #\$03 #\$A0	OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT
F93D: F93F: F940: F941: F944: F945: F948: F948: F94A: F94C: F94F:	C8 98 20 8A 4C A2 A9 20 CA	DA DA 03 A0 ED	FD FD	PRNTYX PRNTAX PRNTX PRBLNK PRBL2	BNE INY TYA JSR TXA JMP LDX LDA JSR DEX	PRBYTE PRBYTE #\$03 #\$A0 COUT	OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK
F93D: F93F: F940: F941: F944: F945: F948: F948: F94A: F94C: F94F: F950:	C8 98 20 8A 4C A2 A9 20 CA D0	DA DA 03 A0 ED	FD FD	PRNTYX PRNTAX PRNTX PRBLNK PRBL2	BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE	PRBYTE PRBYTE #\$03 #\$A0	OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE
F93D: F940: F941: F944: F945: F948: F948: F948: F947: F947: F947: F947: F950:	C8 98 20 8A 4C A2 A9 20 CA D0 60	DA DA 03 A0 ED	FD FD	PRNTYX PRNTAX PRNTX PRBLNK PRBL2 PRBL3	BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE RTS	PRBYTE PRBYTE #\$03 #\$A0 COUT	OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK LOOP UNTIL COUNT=0
F93D: F940: F941: F944: F945: F948: F948: F94A: F94C: F94F: F950: F952: F953:	C8 98 20 8A 4C A2 20 CA D0 60 38	DA DA 03 A0 ED F8	FD FD	PRNTYX PRNTAX PRNTX PRBLNK PRBL2 PRBL3 PCADJ	BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE RTS SEC	PRBYTE PRBYTE #\$03 #\$A0 COUT PRBL2	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
F93D: F940: F941: F944: F945: F948: F948: F948: F947: F947: F947: F947: F950:	C8 98 20 8A 4C A2 20 CA D0 60 38 A5	DA DA 03 A0 ED F8 2F	FD FD	PRNTYX PRNTAX PRNTX PRBLNK PRBL2 PRBL3	BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE RTS	PRBYTE PRBYTE #\$03 #\$A0 COUT	OUTPUT TARGET ADR OF BRANCH AND RETURN BLANK COUNT LOAD A SPACE OUTPUT A BLANK LOOP UNTIL COUNT=0
F93D: F940: F941: F944: F945: F948: F948: F94A: F94C: F94F: F950: F952: F953: F954:	C8 98 20 8A 4C A2 20 CA D0 60 38 A5 A4	DA DA 03 A0 ED F8 2F	FD FD	PRNTYX PRNTAX PRNTX PRBLNK PRBL2 PRBL3 PCADJ PCADJ	BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE RTS SEC LDA	PRBYTE PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH	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
F93D: F940: F941: F944: F945: F948: F948: F948: F947: F947: F950: F952: F953: F953: F954: F956:	C8 98 20 8A 4C A2 20 CA D0 60 38 A5 A4 AA	DA DA 03 A0 ED F8 2F 3B	FD FD	PRNTYX PRNTAX PRNTX PRBLNK PRBL2 PRBL3 PCADJ PCADJ	BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE RTS SEC LDA LDY	PRBYTE PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH	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
F93D: F940: F941: F944: F945: F948: F948: F947: F947: F952: F952: F953: F953: F954: F956: F958:	C8 98 20 8A 4C A2 20 CA D0 60 38 A5 A4 AA 10	DA DA 03 A0 ED F8 2F 3B	FD FD	PRNTYX PRNTAX PRNTX PRBLNK PRBL2 PRBL3 PCADJ PCADJ	BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE RTS SEC LDA LDY TAX	PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCH	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
F93D: F940: F941: F941: F944: F944: F944: F944: F944: F946: F950: F952: F952: F952: F955: F	C8 98 20 8A 4C A2 20 CA 20 60 38 A5 A4 AA 10 88	DA 03 A0 ED F8 2F 3B 01	FD FD	PRNTYX PRNTAX PRNTX PRBLNK PRBL2 PRBL3 PCADJ PCADJ	BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE RTS SEC LDA LDY TAX BPL	PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCH	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)
F93D: F940: F940: F941: F944: F945: F948: F948: F948: F948: F950: F	C8 98 20 8A 4C A2 20 CA 20 60 38 A5 A4 AA 10 88 65	DA DA 03 A0 ED F8 2F 3B 01 3A	FD FD	PRNTYX PRNTAX PRNTX PRBL2 PRBL3 PCADJ PCADJ2 PCADJ2 PCADJ3	BNE INY TYA JSR TXA JMP LDX LDA JSR DEX ENE RTS SEC LDA LDY TAX BPL DEY	PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCH PCADJ4	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)
F93D: F940: F9440: F9441: F9442: F9442: F948: F947: F947: F9502:	C8 98 20 8A 4C A2 20 CA 20 CA 60 38 A5 A4 10 88 65 90	DA DA 03 A0 ED F8 2F 3B 01 3A	FD FD	PRNTYX PRNTAX PRNTX PRBL2 PRBL3 PCADJ PCADJ2 PCADJ2 PCADJ3	BNE INY TYA JSR TXA JMP LDX LDA JSR BNE RTS SEC LDA LDY TAX BPL DEY ADC	PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCH PCADJ4 PCL	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
F93D: F940: F9441: F9442: F9442: F9442: F9442: F9442: F9442: F9502: F9552: F9554: F9558: F9552: F9555:	C8 98 20 8A 4C A2 20 CA D0 60 38 A5 A4 AA 10 88 65 90 C8	DA DA 03 A0 ED F8 2F 3B 01 3A	FD FD	PRNTYX PRNTAX PRNTX PRBL2 PRBL3 PCADJ PCADJ2 PCADJ2 PCADJ3	BNE INY TYA JSR LDX LDA JSR DEX BNE RTS SEC LDA LDY TAX BPL DEY ADC BCC	PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCH PCADJ4 PCL	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
F93D: F940: F941: F941: F944: F944: F944: F944: F944: F945: F950: F952: F955: F959: F950: F	C8 98 20 8A 4C A2 20 CA D0 60 38 A5 A4 AA 10 88 65 90 C8	DA DA 03 A0 ED F8 2F 3B 01 3A	FD FD	PRNTYX PRNTAX PRNTX PRBLNK PRBL2 PRBL3 PCADJ PCADJ2 PCADJ2 PCADJ3	BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE RTS SEC LDA LDY TAX BPL DEY ADC BCC INY RTS	PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCH PCADJ4 PCL	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
F93D: F940: F941: F941: F944: F944: F944: F944: F944: F945: F950: F952: F955: F959: F950: F	C8 98 20 8A 4C A2 20 CA D0 60 38 A5 A4 AA 10 88 65 90 C8	DA DA 03 A0 ED F8 2F 3B 01 3A	FD FD	PRNTYX PRNTAX PRNTX PRBL2 PRBL3 PCADJ PCADJ2 PCADJ2 PCADJ3 PCADJ4 RTS2 *	BNE INY TYA JSR TXA JMP LDX LDA JSR DEX BNE RTS SEC LDA LDY TAX BPL DEY ADC BCC INY RTS	PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCH PCADJ4 PCL RTS2 BYTES:	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
F93D: F940: F941: F941: F944: F944: F944: F944: F944: F945: F950: F952: F955: F959: F950: F	C8 98 20 8A 4C A2 20 CA D0 60 38 A5 A4 AA 10 88 65 90 C8	DA DA 03 A0 ED F8 2F 3B 01 3A	FD FD	PRNTYX PRNTAX PRNTX PRBL2 PRBL3 PCADJ PCADJ2 PCADJ2 PCADJ3 PCADJ4 RTS2 * *	BNE INY TYA JSR TXA JMP LDX LDA DEX BNE RTS SEC LDA LDY TAX BPL DEY ADC INY RTS FMT1	PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCH PCADJ4 PCL RTS2 BYTES:	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 THEN RIGHT HALF BYTE
F93D: F93F: F940: F941: F944: F944: F948: F947: F947: F947: F950: F9552: F9554: F9554: F9558: F9558: F9552: F9551: F9560: F961:	C8 98 20 8A 4C A9 20 CA D0 60 38 A5 A4 A0 88 65 90 C8 60	DA DA 03 A0 ED F8 2F 3B 01 3A 01	FD FD	PRNTYX PRNTAX PRNTX PRBL2 PRBL3 PCADJ PCADJ2 PCADJ2 PCADJ3 PCADJ4 RTS2 *	BNE INY TYA JSR TXA JMP LDX LDA DEX BNE RTS SEC LDA BNE RTS BPL DEY ADC BCC INY RTS FMT1 IF Y=	PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCH PCADJ4 PCL RTS2 BYTES:	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
F93D: F93F: F940: F941: F945: F945: F948: F947: F947: F950: F957: F957: F958:	C8 98 20 8A 4C A2 20 CA D0 60 38 A5 A4 A10 88 65 90 C8 60 04	DA DA 03 A0 ED F8 2F 3B 01 3A 01	FD FD	PRNTYX PRNTAX PRNTX PRBL2 PRBL3 PCADJ PCADJ2 PCADJ3 PCADJ4 RTS2 * * *	BNE INY TYA JSR TXA JMP LDX LDA LDA BNE RTS SEC LDA LDY TAX BPL DEY ADC BCC INY RTS FMT1 IF Y= IF Y=	PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCADJ4 PCL RTS2 BYTES: 0 -1	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 THEN RIGHT HALF BYTE (X=INDEX)
F93D: F93F: F940: F944: F944: F948: F948: F948: F948: F954: F950: F950: F9550: F9550: F9550: F9550: F9550: F9550: F9550: F9551: F9551: F9562: F962: F9655:	C8 98 20 8A 4C A2 20 CA D0 60 38 A5 A4 A0 88 65 90 C8 60 04 30	DA DA 03 A0 ED F8 2F 3B 01 3A 01 20 0D	FD FD FD	PRNTYX PRNTAX PRNTX PRBL2 PRBL3 PCADJ PCADJ2 PCADJ2 PCADJ3 PCADJ4 RTS2 * *	BNE INY TYA JSR TXA JMP LDX LDA DEX BNE RTS SEC LDA BNE RTS BPL DEY ADC BCC INY RTS FMT1 IF Y=	PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCH PCADJ4 PCL RTS2 BYTES:	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 THEN RIGHT HALF BYTE (X=INDEX)
F93D: F93F: F940: F944: F944: F948: F948: F948: F948: F954: F950: F950: F9550:	C8 98 20 8A 4C A9 20 CA D0 60 38 A5 A4 AA 10 88 65 90 C8 60 04 30 80	DA DA 03 A0 ED F8 2F 3B 01 3A 01 20 0D 04	FD FD FD	PRNTYX PRNTAX PRNTX PRBL2 PRBL3 PCADJ PCADJ2 PCADJ3 PCADJ4 RTS2 * * *	BNE INY TYA JSR TXA JMP LDX LDA DEX BNE RTS SEC LDA LDY TAX BPL DEY ADC BCC INY RTS FMT1 IF Y IF Y DFB	PRBYTE PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCH PCADJ4 PCL RTS2 BYTES: 0 -1 \$04,\$20,\$54	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 THEN RIGHT HALF BYTE (X=INDEX)
F93D: F93F: F940: F944: F944: F948: F948: F947: F947: F950: F950: F9552: F9554: F9554: F9558: F9558: F9558: F9551: F9561: F9601: F9622: F9625: F9655: F9671: F9643:	C8 98 20 8A 4C 20 CA 20 CA D0 60 38 A5 A4 A0 88 65 90 C8 60 04 30 80 03	DA 03 A0 ED F8 2F 3B 01 3A 01 20 0D 04 22	FD FD 54 90	PRNTYX PRNTAX PRNTX PRBL2 PRBL3 PCADJ PCADJ2 PCADJ3 PCADJ4 RTS2 * * *	BNE INY TYA JSR TXA JMP LDX LDA LDA BNE RTS SEC LDA LDY TAX BPL DEY ADC BCC INY RTS FMT1 IF Y= IF Y=	PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCADJ4 PCL RTS2 BYTES: 0 -1	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 THEN RIGHT HALF BYTE (X=INDEX)
F93D: F93F: F940: F944: F944: F947: F947: F947: F950: F950: F9550: F9550: F9550: F9550: F9550: F9550: F9550: F9560: F9601: F9602: F967: F967: F967: F967: F967: F967:	C8 98 20 8A 4C 20 20 20 20 20 20 20 20 20 20 20 20 20	DA DA 03 A0 ED F8 2F 3B 01 3A 01 20 0D 04 22 33	FD FD 54 90	PRNTYX PRNTAX PRNTX PRBL2 PRBL3 PCADJ PCADJ2 PCADJ3 PCADJ4 RTS2 * * *	BNE INY TYA JSR TXA JMP LDX LDA JSR BNE RTS SEC LDA LDY TAX BPL DEY ADC BCC BCC INY RTS FMT1 IF Y= IF Y= DFB	PRBYTE PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCADJ4 PCL RTS2 BYTES: 0 1 \$04,\$20,\$54 \$80,\$04,\$90	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) XXXXXXYO INSTRS THEN LEFT HALF BYTE THEN RIGHT HALF BYTE (X=INDEX) 4,\$30,\$0D 0,\$03,\$22
F93D: F93F: F940: F944: F945: F948: F947: F947: F947: F950: F957: F957: F958: F968:	C8 98 20 8A 4C 20 CA D0 60 38 A2 20 CA D0 60 38 54 4 AA 10 88 65 90 C8 60 04 30 80 80 354 80	DA 03 A0 ED F8 2F 3B 01 3A 01 20 00 04 22 33 04	FD FD 54 90	PRNTYX PRNTAX PRNTX PRBL2 PRBL3 PCADJ PCADJ2 PCADJ3 PCADJ4 RTS2 * * *	BNE INY TYA JSR TXA JMP LDX LDA DEX BNE RTS SEC LDA LDY TAX BPL DEY ADC BCC INY RTS FMT1 IF Y IF Y DFB	PRBYTE PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCH PCADJ4 PCL RTS2 BYTES: 0 -1 \$04,\$20,\$54	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) XXXXXXYO INSTRS THEN LEFT HALF BYTE THEN RIGHT HALF BYTE (X=INDEX) 4,\$30,\$0D 0,\$03,\$22
F93D: F93F: F940: F944: F944: F944: F944: F944: F944: F954: F950: F950: F955: F955: F955: F955: F955: F955: F955: F961: F962: F962: F965: F967: F967: F967:	C8 98 20 8A 4C 20 CA D0 60 38 A5 20 CA D0 60 38 A5 90 C8 60 04 30 80 354 80 90 034 90	DA 03 A0 ED F8 2F 3B 01 3A 01 20 00 04 22 33 04 04	FD FD 54 90	PRNTYX PRNTAX PRNTX PRBL2 PRBL3 PCADJ PCADJ2 PCADJ3 PCADJ4 RTS2 * * *	BNE INY TYA JSR TXA JMP LDX LDA DEX ENE RTS SEC LDA LDY TAX BPL DEY ADC BCC INY RTS FMT1 IF Y= IF Y= DFB DFB	PRBYTE PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCH PCADJ4 PCL RTS2 BYTES: 0 1 \$04,\$20,\$54 \$80,\$04,\$90 \$54,\$33,\$01	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) XXXXXYO INSTRS THEN LEFT HALF BYTE THEN RIGHT HALF BYTE (X=INDEX) 4,\$30,\$0D 0,\$03,\$22
F93D: F93F: F940: F944: F944: F944: F944: F944: F944: F954: F950: F952: F955: F955: F955: F955: F955: F956: F956: F960: F961: F966: F967: F964: F967: F964:	C8 98 20 8A 4C 20 CA D0 60 83 84 A3 10 88 65 0 C8 60 04 30 80 03 54 90 54	DA 03 A0 ED F8 2F 3B 01 3A 01 20 0D 04 220 33 04 02 33 04 04 33	FD FD 54 90 20	PRNTYX PRNTAX PRNTX PRBL2 PRBL3 PCADJ PCADJ2 PCADJ3 PCADJ4 RTS2 * * *	BNE INY TYA JSR TXA JMP LDX LDA JSR BNE RTS SEC LDA LDY TAX BPL DEY ADC BCC BCC INY RTS FMT1 IF Y= IF Y= DFB	PRBYTE PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCADJ4 PCL RTS2 BYTES: 0 1 \$04,\$20,\$54 \$80,\$04,\$90	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) XXXXXYO INSTRS THEN LEFT HALF BYTE THEN RIGHT HALF BYTE (X=INDEX) 4,\$30,\$0D 0,\$03,\$22
F93D: F93F: F940: F944: F944: F944: F947: F947: F950: F950: F950: F955: F955: F958: F958: F958: F958: F958: F958: F956: F960: F961: F966: F967: F966: F967: F967: F974:	C8 98 20 8A 4C 20 CA D0 60 83 85 A4 AD 20 CA D0 60 88 60 03 80 03 54 80 03 54 00 54 00	DA DA 03 A0 ED F8 2F 3B 01 3A 01 20 0D 04 22 33 04 23 34 04 33 80	FD FD 54 90 20	PRNTYX PRNTAX PRNTX PRBL2 PRBL3 PCADJ PCADJ2 PCADJ3 PCADJ4 RTS2 * * *	BNE INY TYA JSR TXA JMP LDX LDA DEX BNE RTS SEC LDA LDY TAX BPL DEY ADC BCC INY RTS FMT1 IF Y= IF Y= DFB DFB DFB	PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCADJ4 PCL RTS2 BYTES: 0 =1 \$04,\$20,\$54 \$80,\$04,\$90 \$54,\$33,\$01 \$90,\$04,\$20	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) XXXXXXYO INSTRS THEN LEFT HALF BYTE THEN RIGHT HALF BYTE (X=INDEX) 4,\$30,\$0D 0,\$03,\$22 0,\$80,\$04 0,\$54,\$33
F93D: F93F: F940: F944: F944: F944: F947: F947: F947: F950: F950: F955: F955: F958: F958: F958: F958: F956: F956: F960: F961: F966: F966: F967: F967: F967: F976: F976:	C8 98 20 84 20 20 CA 20 CA 20 CA 20 60 38 A5 A4 AA 20 CA 50 60 04 380 90 C8 60 04 380 90 54 80 90 20 03 90 20 03 90 20 00 20 00 20 00 20 20 20 20 20 20 20	DA DA 03 A0 ED F8 2F 3B 01 3A 01 3A 01 20 0D 04 22 33 04 04 33 80 04	FD FD 54 90 00 200	PRNTYX PRNTAX PRNTX PRBL2 PRBL3 PCADJ PCADJ2 PCADJ3 PCADJ4 RTS2 * * *	BNE INY TYA JSR TXA JMP LDX LDA DEX ENE RTS SEC LDA LDY TAX BPL DEY ADC BCC INY RTS FMT1 IF Y= IF Y= DFB DFB	PRBYTE PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCH PCADJ4 PCL RTS2 BYTES: 0 1 \$04,\$20,\$54 \$80,\$04,\$90 \$54,\$33,\$01	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) XXXXXXYO INSTRS THEN LEFT HALF BYTE THEN RIGHT HALF BYTE (X=INDEX) 4,\$30,\$0D 0,\$03,\$22 0,\$80,\$04 0,\$54,\$33
F93D: F93F: F940: F944: F945: F948: F948: F947: F947: F950: F957: F957: F957: F958: F957: F978:	C8 98 20 A2 A2 A9 20 CD0 60 38 A5 A4 A10 88 65 90 80 80 30 80 90 54 00 54 00 20	DA 03 A0 ED F8 2F 3B 01 3A 01 20 004 22 33 04 04 33 04 54	FD FD 54 90 00 200	PRNTYX PRNTAX PRNTX PRBL2 PRBL3 PCADJ PCADJ2 PCADJ3 PCADJ4 RTS2 * * *	BNE INY TYA JSR TXA JMP LDX LDA LDA ENE RTS SEC LDA LDY TAX BPL DEY ADC BCC INY RTS FMT1 IF Y= IF Y= DFB DFB DFB DFB	PRBYTE PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCADJ4 PCL RTS2 BYTES: 0 \$04,\$20,\$54 \$80,\$04,\$90 \$54,\$33,\$01 \$90,\$04,\$20 \$04,\$20	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) XXXXXXYO INSTRS THEN LEFT HALF BYTE THEN RIGHT HALF BYTE (X=INDEX) 4,\$30,\$0D 0,\$03,\$22 0,\$80,\$04 0,\$54,\$33 4,\$90,\$04
F93D: F93F: F940: F944: F944: F944: F947: F947: F947: F950: F950: F955: F955: F958: F958: F958: F958: F956: F956: F960: F961: F966: F966: F967: F967: F967: F976: F976:	C8 98 20 A2 A2 A9 20 CD0 60 38 A5 A4 A0 00 88 65 90 80 60 04 30 80 054 00 54 00 54 00 00 00 00 00 00 00 00 00 00 00 00 00	DA 03 A0 ED F8 01 3A 01 20 00D 04 223 04 04 33 80 04 54 80	FD FD 54 90 0D 20 04 3B	PRNTYX PRNTAX PRNTX PRBL2 PRBL3 PCADJ PCADJ2 PCADJ3 PCADJ4 RTS2 * * *	BNE INY TYA JSR TXA JMP LDX LDA DEX BNE RTS SEC LDA LDY TAX BPL DEY ADC BCC INY RTS FMT1 IF Y= IF Y= DFB DFB DFB	PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCADJ4 PCL RTS2 BYTES: 0 =1 \$04,\$20,\$54 \$80,\$04,\$90 \$54,\$33,\$01 \$90,\$04,\$20	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) XXXXXXYO INSTRS THEN LEFT HALF BYTE THEN RIGHT HALF BYTE (X=INDEX) 4,\$30,\$0D 0,\$03,\$22 0,\$80,\$04 0,\$54,\$33 4,\$90,\$04
F93D: F93F: F940: F944: F944: F944: F944: F944: F944: F944: F954: F950: F950: F955: F955: F955: F955: F955: F955: F9561: F9661: F9662: F9662: F9662: F9662: F9661: F9662: F9661: F9671: F9771: F9771: F9771: F9775: F9775: F9775: F9775:	C8 98 20 84 20 CA 20 CA 20 38 A5 A4 AA 20 CA 38 60 CA 30 80 54 00 54 00 20 00 20 00 4	DA 03 A0 ED F8 01 3A 01 20 0D 04 222 33 04 54 33 80 04 54 90	FD FD 54 90 0D 20 04 3B	PRNTYX PRNTAX PRNTX PRBL2 PRBL3 PCADJ PCADJ2 PCADJ3 PCADJ4 RTS2 * * *	BNE INY TYA JSR TXA JMP LDX LDA LDA ENE RTS SEC LDA LDY TAX BPL DEY ADC BCC INY RTS FMT1 IF Y= IF Y= DFB DFB DFB DFB	PRBYTE PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCADJ4 PCL RTS2 BYTES: 0 \$04,\$20,\$54 \$80,\$04,\$90 \$54,\$33,\$01 \$90,\$04,\$20 \$04,\$20	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 THEN RIGHT HALF BYTE (X=INDEX) 4,\$30,\$0D 0,\$03,\$22 0,\$80,\$04 0,\$54,\$33 4,\$90,\$04 3,\$0D,\$80
F93D: F93F: F940: F944: F944: F944: F944: F944: F944: F944: F954: F956: F955: F955: F955: F955: F955: F956: F956: F961: F966: F967: F966: F966: F966: F967: F964: F976: F976: F976: F976: F977: F977: F977: F977: F977: F977: F978: F968: F978:	C8 98 20 8AC A2 20 CA D0 60 38 A5 A4 AA 20 CA 60 38 A5 60 C8 03 54 80 03 54 80 020 00 20 004 22	DA DA 03 A0 ED F8 01 3A 01 200 04 22 33 04 54 80 04 54 80 04 54 80 04 54 80 04 54 80 04 54 80 10 10 10 10 10 10 10 10 10 1	FD FD 54 90 0D 20 04 3B 00	PRNTYX PRNTAX PRNTX PRBL2 PRBL3 PCADJ PCADJ2 PCADJ3 PCADJ4 RTS2 * * *	BNE INY TYA JSR TXA JMP LDX LDA DEX ENE RTS SEC LDA LDY TAX BPL DEY ADC BCC INY RTS FMT1 IF Y= IF Y= DFB DFB DFB DFB DFB	PRBYTE PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCH PCADJ4 PCL RTS2 BYTES: 0 -1 \$04,\$20,\$54 \$80,\$04,\$20 \$54,\$33,\$01 \$90,\$04,\$20 \$54,\$33,\$01 \$90,\$04,\$20 \$54,\$33,\$01 \$90,\$04,\$20	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 THEN RIGHT HALF BYTE (X=INDEX) 4,\$30,\$0D 0,\$03,\$22 0,\$80,\$04 0,\$54,\$33 4,\$90,\$04 3,\$0D,\$80
F93D: F93F: F940: F944: F944: F944: F944: F947: F957: F957: F957: F957: F958: F958: F958: F958: F958: F958: F958: F958: F958: F958: F961: F961: F967: F967: F967: F967: F967: F968: F977: F977: F977: F977: F978:	C8 98 20 84C A9 20 CA D0 60 38 A5 A4 AA 00 38 65 90 C8 60 04 380 90 20 004 22 004 22 33	DA DA 03 A0 ED F8 2F 3B 01 3A 01 20 004 223 33 04 04 54 80 04 54 80 04 54 80 04 04 00 04 04 04 04 04 04 0	FD FD 54 90 0D 20 04 3B 00	PRNTYX PRNTAX PRNTX PRBL2 PRBL3 PCADJ PCADJ2 PCADJ3 PCADJ4 RTS2 * * *	BNE INY TYA JSR TXA JMP LDX LDA DEX ENE RTS SEC LDA LDY TAX BPL DEY ADC BCC INY RTS FMT1 IF Y= IF Y= DFB DFB DFB DFB DFB	PRBYTE PRBYTE #\$03 #\$A0 COUT PRBL2 LENGTH PCH PCADJ4 PCL RTS2 BYTES: 0 -1 \$04,\$20,\$54 \$80,\$04,\$20 \$54,\$33,\$01 \$90,\$04,\$20 \$54,\$33,\$01 \$90,\$04,\$20 \$54,\$33,\$01 \$90,\$04,\$20	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 THEN RIGHT HALF BYTE (X=INDEX) 4,\$30,\$0D 0,\$54,\$33 4,\$90,\$04 3,\$0D,\$80 0,\$22,\$44

F98A: 1	1 22	44				
F98D: 3	3 0D			DFB	\$11,\$22,	\$44,\$33,\$0D
F98F: C	8 44	A9				
F992: 0	1 22			DFB	\$C8,\$44,	\$A9,\$01,\$22
F994: 4	4 33	0D				
F997: 8	0 04			DFB	\$44,\$33,	\$0D,\$80,\$04
F999: 9	0 01	22				
F99C: 4	4 33			DFB	\$90,\$01,	\$22,\$44,\$33
F99E: 01	D 80	04				
F9A1: 9	0			DFB	\$0D,\$80,	\$04,\$90
F9A2: 2		87				
F9A5: 92				DFB	\$26,\$31,	\$87,\$9A \$ZZXXXY01 INSTR'S
F9A6: 0			FMT2	DFB	\$00	ERR
F9A7: 2					\$21	IMM
F9A8: 8				DFB	\$81	Z-PAGE
F9A9: 8				DFB	\$82	ABS
F9AA: 0					\$00	IMPLIED
F9AB: 0				DFB	\$00	ACCUMULATOR
F9AC: 5				DFB		(ZPAG,X)
F9AD: 41				DFB	\$4D	(ZPAG), Y
F9AE: 9				DFB	\$91	ZPAG,X
F9AF: 9					\$92	ABS,X
F9B0: 8				DFB	\$86	ABS,Y
F9B1: 4				DFB		(ABS)
F9B2: 8				DFB	\$85	ZPAG,Y
F9B3: 91				DFB	2AD	RELATIVE
F9B4: A						
F9B7: A	3 A8	A4				
			CHAR1	ASC	",),#(\$"	
F9BA: D						
F9BD: A	4 A4	00	CHAR2	DFB	\$D9,\$00,	\$D8,\$A4,\$A4,\$00
			*CHAR2:	"Y",0,	"X\$\$",0	
			*	MNEMI		IS OF FORM:
			*	(A) >	XXXXX000	
			*	(B) X	XXXYY100	
			*		LXXX1010	
			*		XXXYYY10	
			*		XXXYYY01	
			*		X=INDEX)	
F9C0: 1	C 84	1C		,		
			MNEML	DFB	\$1C \$8A	\$1C,\$23,\$5D,\$
F9C6: 11				212	φ±0,φ011,	<i>\\</i> 20 <i>/\</i> 20 <i>/\</i> 02 <i>/\</i>
F9C9: 82				DFB	¢18 ¢11	\$9D,\$8A,\$1D,\$23
F9CC: 91				DFB	γıd,γai,	, , , , , , , , , , , , , , , , , , ,
FUCC: 9	л ор	ΤD				
	1 00	20				¢15 ¢31 ¢00 ¢00
F9CF: A				DFB	\$9D,\$8B,	\$1D,\$A1,\$00,\$29
F9D2: 1	9 AE	69				
F9D2: 1 F9D5: A	9 AE 8 19	69 23		DFB DFB		\$1D,\$A1,\$00,\$29 \$69,\$A8,\$19,\$23
F9D2: 1 F9D5: A F9D8: 2	9 AE 8 19 4 53	69 23 1B		DFB	\$19,\$AE,	\$69,\$A8,\$19,\$23
F9D2: 1 F9D5: A F9D8: 2 F9D8: 2	9 AE 8 19 4 53 3 24	69 23 1B 53		DFB DFB	\$19,\$AE, \$24,\$53,	\$69,\$A8,\$19,\$23 \$1B,\$23,\$24,\$53
F9D2: 1 F9D5: A F9D8: 2 F9DB: 2 F9DB: 2	9 AE 8 19 4 53 3 24 9 A1	69 23 1B 53		DFB	\$19,\$AE, \$24,\$53,	\$69,\$A8,\$19,\$23
F9D2: 1 F9D5: A F9D8: 2 F9DB: 2 F9DE: 1 F9DE: 1	9 AE 8 19 4 53 3 24 9 A1 0 1A	69 23 1B 53 5B		DFB DFB	\$19,\$AE, \$24,\$53, \$19,\$A1	\$69,\$A8,\$19,\$23 \$1B,\$23,\$24,\$53 (A) FORMAT ABOVE
F9D2: 1 F9D5: A F9D8: 2 F9DB: 2 F9DB: 2	9 AE 8 19 4 53 3 24 9 A1 0 1A	69 23 1B 53 5B		DFB DFB	\$19,\$AE, \$24,\$53, \$19,\$A1	\$69,\$A8,\$19,\$23 \$1B,\$23,\$24,\$53
F9D2: 1 F9D5: A F9D8: 2 F9DB: 2 F9DE: 1 F9E0: 0 F9E3: 5 F9E6: 2	9 AE 8 19 4 53 3 24 9 A1 0 1A B A5 4 24	69 23 1B 53 5B 69		DFB DFB DFB DFB	\$19,\$AE, \$24,\$53, \$19,\$A1 \$00,\$1A,	\$69,\$A8,\$19,\$23 \$1B,\$23,\$24,\$53 (A) FORMAT ABOVE
F9D2: 1 F9D5: A F9D8: 2 F9DB: 2 F9DE: 1 F9E0: 0 F9E3: 5	9 AE 8 19 4 53 3 24 9 A1 0 1A B A5 4 24	69 23 1B 53 5B 69		DFB DFB DFB DFB	\$19,\$AE, \$24,\$53, \$19,\$A1 \$00,\$1A,	\$69,\$A8,\$19,\$23 \$1B,\$23,\$24,\$53 (A) FORMAT ABOVE \$5B,\$5B,\$A5,\$69
F9D2: 1 F9D5: A F9D8: 2 F9DB: 2 F9DE: 1 F9E0: 0 F9E3: 5 F9E6: 2	9 AE 8 19 4 53 3 24 9 A1 0 1A B A5 4 24 E AE	69 23 1B 53 5B 69 A8		DFB DFB DFB DFB DFB	\$19,\$AE, \$24,\$53, \$19,\$A1 \$00,\$1A, \$24,\$24	\$69,\$A8,\$19,\$23 \$1B,\$23,\$24,\$53 (A) FORMAT ABOVE \$5B,\$5B,\$A5,\$69
F9D2: 1 F9D5: A F9D8: 2 F9DB: 2 F9DE: 1 F9E0: 0 F9E3: 5 F9E6: 2 F9E6: 2	9 AE 8 19 4 53 3 24 9 A1 0 1A B A5 4 24 E AE D 29	69 23 1B 53 5B 69 A8 00		DFB DFB DFB DFB DFB	\$19,\$AE, \$24,\$53, \$19,\$A1 \$00,\$1A, \$24,\$24 \$AE,\$AE,	\$69,\$A8,\$19,\$23 \$1B,\$23,\$24,\$53 (A) FORMAT ABOVE \$5B,\$5B,\$A5,\$69 (B) FORMAT
F9D2: 1 F9D5: A F9D8: 2 F9DB: 2 F9DE: 1 F9E0: 0 F9E3: 5 F9E6: 2 F9E6: 2 F9E8: A F9E8: A	9 AE 8 19 4 53 3 24 9 A1 0 1A B A5 4 24 E AE D 29 C 00	69 23 1B 53 5B 69 A8 00		DFB DFB DFB DFB DFB	\$19,\$AE, \$24,\$53, \$19,\$A1 \$00,\$1A, \$24,\$24 \$AE,\$AE,	\$69,\$A8,\$19,\$23 \$1B,\$23,\$24,\$53 (A) FORMAT ABOVE \$5B,\$5B,\$A5,\$69 (B) FORMAT \$A8,\$AD,\$29,\$00
F9D2: 1 F9D5: A F9D8: 2 F9DB: 2 F9DE: 1 F9E0: 0 F9E3: 5 F9E6: 2 F9E6: 2 F9E8: A F9EB: A F9EE: 7	9 AE 8 19 4 53 3 24 9 A1 0 1A B A5 4 24 E AE D 29 C 00 5 9C	69 23 1B 53 5B 69 A8 00 6D		DFB DFB DFB DFB DFB DFB DFB	\$19,\$AE, \$24,\$53, \$19,\$A1 \$00,\$1A, \$24,\$24 \$AE,\$AE, \$7C,\$00	\$69,\$A8,\$19,\$23 \$1B,\$23,\$24,\$53 (A) FORMAT ABOVE \$5B,\$5B,\$A5,\$69 (B) FORMAT \$A8,\$AD,\$29,\$00
F9D2: 1 F9D5: A F9D8: 2 F9DB: 2 F9DE: 1 F9E0: 0 F9E3: 5 F9E6: 2 F9E6: 2 F9E8: A F9EB: A F9EB: A F9EB: 7 F9E0: 1	9 AE 8 19 4 53 3 24 9 A1 0 1A B A5 4 24 E AE D 29 C 00 5 9C C A5	 69 23 1B 53 5B 69 A8 00 6D 69 		DFB DFB DFB DFB DFB DFB DFB DFB	\$19,\$AE, \$24,\$53, \$19,\$A1 \$00,\$1A, \$24,\$24 \$AE,\$AE, \$7C,\$00 \$15,\$9C,	\$69,\$A8,\$19,\$23 \$1B,\$23,\$24,\$53 (A) FORMAT ABOVE \$5B,\$5B,\$A5,\$69 (B) FORMAT \$A8,\$AD,\$29,\$00 (C) FORMAT
F9D2: 1 F9D5: A F9D8: 2 F9DB: 2 F9DE: 1 F9E0: 0 F9E3: 5 F9E6: 2 F9E8: A F9E8: A F9E8: A F9E8: 7 F9E9: 1 F9F0: 1 F9F3: 9	9 AE 8 19 4 53 3 24 9 A1 0 1A B A5 4 24 E AE D 29 C 00 5 9C C A5 9 53	69 23 1B 53 5B 69 A8 00 6D 69		DFB DFB DFB DFB DFB DFB DFB DFB	\$19,\$AE, \$24,\$53, \$19,\$A1 \$00,\$1A, \$24,\$24 \$AE,\$AE, \$7C,\$00 \$15,\$9C,	\$69,\$A8,\$19,\$23 \$1B,\$23,\$24,\$53 (A) FORMAT ABOVE \$5B,\$5B,\$A5,\$69 (B) FORMAT \$A8,\$AD,\$29,\$00 (C) FORMAT \$6D,\$9C,\$A5,\$69
F9D2: 1 F9D5: A F9D8: 2 F9DB: 2 F9DE: 1 F9E0: 0 F9E3: 5 F9E6: 2 F9E8: A F9E8: A F9E8: A F9E8: 7 F9F0: 1 F9F0: 1 F9F3: 9 F9F6: 2	9 AE 8 19 4 53 3 24 9 A1 0 1A B A5 4 24 C 00 5 9C C 00 5 9C C A5 9 53 4 13	 69 23 1B 53 5B 69 A8 00 6D 69 34 		DFB DFB DFB DFB DFB DFB DFB DFB DFB	\$19,\$AE, \$24,\$53, \$19,\$A1 \$00,\$1A, \$24,\$24 \$AE,\$AE, \$7C,\$00 \$15,\$9C, \$29,\$53	\$69,\$A8,\$19,\$23 \$1B,\$23,\$24,\$53 (A) FORMAT ABOVE \$5B,\$5B,\$A5,\$69 (B) FORMAT \$A8,\$AD,\$29,\$00 (C) FORMAT \$6D,\$9C,\$A5,\$69
F9D2: 1 F9D5: A F9D8: 2 F9DB: 2 F9DE: 1 F9E0: 0 F9E3: 5 F9E6: 2 F9E8: A F9EB: A F9EB: 7 F9E5: 7 F9F0: 1 F9F70: 1 F9F70: 2 F9F6: 2 F9F8: 8	9 AE 8 19 4 53 3 24 9 A1 0 1A B A5 4 24 E AE D 29 C 00 5 9C C A5 9 53 4 13 1 A5	 69 23 1B 53 5B 69 A8 00 6D 69 34 69 		DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	<pre>\$19,\$AE, \$24,\$53, \$19,\$A1 \$00,\$1A, \$24,\$24 \$AE,\$AE, \$7C,\$00 \$15,\$9C, \$29,\$53 \$84,\$13,</pre>	\$69,\$A8,\$19,\$23 \$1B,\$23,\$24,\$53 (A) FORMAT ABOVE \$5B,\$5B,\$A5,\$69 (B) FORMAT \$A8,\$AD,\$29,\$00 (C) FORMAT \$6D,\$9C,\$A5,\$69 (D) FORMAT \$34,\$11,\$A5,\$69
F9D2: 1 F9D5: A F9D8: 2 F9DB: 2 F9DE: 1 F9E0: 0 F9E3: 5 F9E6: 2 F9E8: A F9EB: A F9EB: A F9EE: 7 F9F0: 1 F9F0: 1 F9F8: 8 F9FB: 1	9 AE 8 19 4 53 3 24 9 A1 0 1A B A5 4 24 E AE D 29 C 00 5 9C C A5 9 53 4 13 1 A5 3 A0	 69 23 1B 53 5B 69 A8 00 6D 69 34 69 		DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	<pre>\$19,\$AE, \$24,\$53, \$19,\$A1 \$00,\$1A, \$24,\$24 \$AE,\$AE, \$7C,\$00 \$15,\$9C, \$29,\$53 \$84,\$13,</pre>	\$69,\$A8,\$19,\$23 \$1B,\$23,\$24,\$53 (A) FORMAT ABOVE \$5B,\$5E,\$A5,\$69 (B) FORMAT \$A8,\$AD,\$29,\$00 (C) FORMAT \$6D,\$9C,\$A5,\$69 (D) FORMAT
F9D2: 1 F9D5: A F9D8: 2 F9DB: 2 F9DE: 1 F9E0: 0 F9E3: 5 F9E6: 2 F9E8: A F9E8: A F9E8: A F9E8: 7 F9F0: 1 F9F7: 9 F9F6: 2 F9F8: 8 F9F8: 1 F9F8: 2 FA00: D	9 AE 8 19 4 53 3 24 9 A1 0 1A B A5 4 24 E AE D 29 C 00 5 9C C A5 9 53 4 13 1 A5 3 A0 8 62	 69 23 1B 53 5B 69 A8 00 6D 69 34 69 5A 	MNEMR	DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	<pre>\$19,\$AE, \$24,\$53, \$19,\$A1 \$00,\$1A, \$24,\$24 \$AE,\$AE, \$7C,\$00 \$15,\$9C, \$29,\$53 \$84,\$13, \$23,\$A0</pre>	\$69,\$A8,\$19,\$23 \$1B,\$23,\$24,\$53 (A) FORMAT ABOVE \$5B,\$5B,\$A5,\$69 (B) FORMAT \$A8,\$AD,\$29,\$00 (C) FORMAT \$6D,\$9C,\$A5,\$69 (D) FORMAT \$34,\$11,\$A5,\$69 (E) FORMAT
F9D2: 1 F9D5: A F9D8: 2 F9DB: 2 F9DE: 1 F9E0: 0 F9E3: 5 F9E6: 2 F9E8: A F9E8: A F9E8: A F9E8: 7 F9F0: 1 F9F0: 1 F9F0: 2 F9F6: 2 F9F8: 8 F9F8: 1 F9F8: 2 FA00: D FA03: 4	9 AE 4 53 3 24 9 A1 53 9 A1 6 1 A5 9 A1 6 1 A5 9 A1 6 2 9 9 C 7 9 C 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7	 69 23 1B 53 5B 69 A8 00 6D 69 34 69 5A 62 	MNEMR	DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	<pre>\$19,\$AE, \$24,\$53, \$19,\$A1 \$00,\$1A, \$24,\$24 \$AE,\$AE, \$7C,\$00 \$15,\$9C, \$29,\$53 \$84,\$13, \$23,\$A0</pre>	\$69,\$A8,\$19,\$23 \$1B,\$23,\$24,\$53 (A) FORMAT ABOVE \$5B,\$5B,\$A5,\$69 (B) FORMAT \$A8,\$AD,\$29,\$00 (C) FORMAT \$6D,\$9C,\$A5,\$69 (D) FORMAT \$34,\$11,\$A5,\$69
F9D2: 1 F9D5: A F9D8: 2 F9DB: 2 F9DB: 1 F9E0: 0 F9E3: 55 F9E6: 2 F9E8: A F9E8: A F9E8: A F9E8: 7 F9F0: 1 F9F3: 9 F9F6: 2 F9F8: 8 F9F8: 1 F9F8: 1 F9F8: 1 F9F8: 1 F9F8: 2 FA00: D FA03: 4 FA06: 9	9 AE 4 533 3 244 533 9 A1 4 533 9 A1 4 533 9 A1 4 53 9 A1 53 9 A	69 23 1B 53 5B 69 A8 00 60 69 34 69 5A 62 54	MNEMR	DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	<pre>\$19,\$AE, \$24,\$53, \$19,\$A1 \$00,\$1A, \$24,\$24 \$AE,\$AE, \$7C,\$00 \$15,\$9C, \$29,\$53 \$84,\$13, \$23,\$A0 \$D8,\$62,</pre>	\$69,\$A8,\$19,\$23 \$1B,\$23,\$24,\$53 (A) FORMAT ABOVE \$5B,\$5E,\$A5,\$69 (B) FORMAT \$A8,\$AD,\$29,\$00 (C) FORMAT \$6D,\$9C,\$A5,\$69 (D) FORMAT \$34,\$11,\$A5,\$69 (E) FORMAT \$5A,\$48,\$26,\$62
F9D2: 1 F9D5: A F9D8: 2 F9DB: 2 F9DE: 1 F9E0: 0 F9E3: 5 F9E6: 2 F9E8: A F9E8: A F9E8: A F9E8: A F9E9: 1 F9F6: 2 F9F6: 2 F9F8: 8 F9F8: 1 F9F8: 2 F9F8: 1 F9F8: 2 FA00: D FA03: 4 FA09: 4	9 AE 4 5333 244 5333 244 5333 244 5333 244 533 244 5 AE 5 4 24 5 AE 5 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7	 69 23 1B 53 5B 69 A8 00 6D 69 34 69 5A 62 54 54 	MNEMR	DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	<pre>\$19,\$AE, \$24,\$53, \$19,\$A1 \$00,\$1A, \$24,\$24 \$AE,\$AE, \$7C,\$00 \$15,\$9C, \$29,\$53 \$84,\$13, \$23,\$A0 \$D8,\$62,</pre>	\$69,\$A8,\$19,\$23 \$1B,\$23,\$24,\$53 (A) FORMAT ABOVE \$5B,\$5B,\$A5,\$69 (B) FORMAT \$A8,\$AD,\$29,\$00 (C) FORMAT \$6D,\$9C,\$A5,\$69 (D) FORMAT \$34,\$11,\$A5,\$69 (E) FORMAT
F9D2: 1 F9D3: 2 F9D8: 2 F9D8: 2 F9D8: 1 F9E0: 0 F9E3: 5 F9E6: 2 F9E6: 2 F9E8: A F9E8: A F9E8: A F9E8: A F9E8: 1 F9F6: 2 F9F6: 2 F3F8: 8 F9F8: 1 F9F8: 2 FA00: D FA03: 4 FA06: 9 FA00: 4 FA0C: 6	9 AE 4 53 3 24 9 A1 9 A1 0 1A B A55 9 A1 1 A5 0 1A 4 24 4 24 4 24 4 24 5 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	69 23 1B 53 5B 69 80 60 60 60 69 34 69 54 54 54 54 54	MNEMR	DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	<pre>\$19,\$AE, \$24,\$53, \$19,\$A1 \$00,\$1A, \$24,\$24 \$AE,\$AE, \$7C,\$00 \$15,\$9C, \$29,\$53 \$84,\$13, \$23,\$A0 \$D8,\$62, \$94,\$88,</pre>	\$69,\$A8,\$19,\$23 \$1B,\$23,\$24,\$53 (A) FORMAT ABOVE \$5B,\$5E,\$A5,\$69 (B) FORMAT \$A8,\$AD,\$29,\$00 (C) FORMAT \$6D,\$9C,\$A5,\$69 (D) FORMAT \$34,\$11,\$A5,\$69 (E) FORMAT \$5A,\$48,\$26,\$62 \$54,\$44,\$C8,\$54
F9D2: 1 F9D2: A F9D8: 2 F9DB: 2 F9DE: 1 F9E0: 0 F9E3: 5 F9E6: 2 F9E6: 2 F9E8: A F9EB: A F9EB: A F9EB: 1 F9F6: 2 F9F6: 2 F9F6: 2 F9F6: 2 F9F6: 2 F9F8: 8 F9FB: 1 F9FE: 2 FA00: D FA03: 4 FA06: 9 FA09: 4 FA0C: 6 FA0F: 9	9 AE 4 53 3 24 9 A1 9 A1 0 1A 4 24 4 24 4 24 4 24 5 3 0 0 1A 5 29 5 33 4 13 1 A5 5 3 4 13 1 A5 5 3 4 13 3 A0 2 4 4 24 4 24 4 24 5 3 5 3 5 3 5 3 4 13 1 A5 5 3 4 13 4 24 4 24 4 24 5 3 5 3 5 3 5 3 5 3 6 26 5 3 5 3 6 26 6 29 5 3 3 4 6 26 6 29 5 3 3 4 6 26 6 29 5 3 3 4 6 26 6 29 5 3 7 4 1 A5 5 3 6 26 6 29 5 3 7 4 1 A5 6 26 6 29 5 3 7 4 1 A5 6 26 6 29 5 3 7 4 1 A5 7 4 8 8 8 8 8 8 8 8 4 4 8 8 8 4 4 4 4	69 23 1B 53 5B 69 A8 00 6D 69 34 69 34 69 5A 62 54 54 88 84	MNEMR	DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	<pre>\$19,\$AE, \$24,\$53, \$19,\$A1 \$00,\$1A, \$24,\$24 \$AE,\$AE, \$7C,\$00 \$15,\$9C, \$29,\$53 \$84,\$13, \$23,\$A0 \$D8,\$62, \$94,\$88,</pre>	\$69,\$A8,\$19,\$23 \$1B,\$23,\$24,\$53 (A) FORMAT ABOVE \$5B,\$5E,\$A5,\$69 (B) FORMAT \$A8,\$AD,\$29,\$00 (C) FORMAT \$6D,\$9C,\$A5,\$69 (D) FORMAT \$34,\$11,\$A5,\$69 (E) FORMAT \$5A,\$48,\$26,\$62
F9D2: 1 F9D5: A F9D8: 2 F9DB: 2 F9DE: 1 F9E0: 0 F9E3: 5 F9E6: 2 F9E8: A F9E8: A F9E8: A F9E8: 7 F9F0: 1 F9F0: 1 F9F6: 2 F9F6: 2 F9F8: 8 F9F8: 1 F9F8: 1 F9F8: 2 FA00: D FA03: 4 FA06: 9 FA09: 4 FA06: 6 FA0F: 9 FA12: 0	9 AE 4 53 3 244 53 9 A1 4	69 23 1B 53 69 A8 00 6D 69 34 69 34 69 5A 62 54 E8 B4 74	MNEMR	DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	<pre>\$19,\$AE, \$24,\$53, \$19,\$A1 \$00,\$1A, \$24,\$24 \$AE,\$AE, \$7C,\$00 \$15,\$9C, \$29,\$53 \$84,\$13, \$23,\$A0 \$D8,\$62, \$94,\$88, \$68,\$44,</pre>	\$69,\$A8,\$19,\$23 \$1B,\$23,\$24,\$53 (A) FORMAT ABOVE \$5B,\$5B,\$A5,\$69 (B) FORMAT \$A8,\$AD,\$29,\$00 (C) FORMAT \$6D,\$9C,\$A5,\$69 (D) FORMAT \$34,\$11,\$A5,\$69 (E) FORMAT \$5A,\$48,\$26,\$62 \$54,\$44,\$C8,\$54 \$E8,\$94,\$00,\$E4
F9D2: 1 F9D5: A F9D8: 2 F9DB: 2 F9DE: 1 F9E0: 0 F9E3: 55 F9E6: 2 F9E8: A F9E8: A F9E8: A F9E8: 7 F9F0: 1 F9F0: 1 F9F0: 2 F9F8: 8 F9F6: 2 F9F8: 8 F9F8: 1 F9F8: 2 FA00: D FA03: 4 FA06: 9 FA09: 4 FA05: 9 FA12: 0 FA15: B	9 AE 4 53 4 53 9 A1 53 9 A1 4 53 9 C 5 9 C 53 9 C	 69 23 1B 53 5B 69 A8 00 6D 69 34 69 34 69 5A 62 54 <	MNEMR	DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	<pre>\$19,\$AE, \$24,\$53, \$19,\$A1 \$00,\$1A, \$24,\$24 \$AE,\$AE, \$7C,\$00 \$15,\$9C, \$29,\$53 \$84,\$13, \$23,\$A0 \$D8,\$62, \$94,\$88, \$68,\$44,</pre>	\$69,\$A8,\$19,\$23 \$1B,\$23,\$24,\$53 (A) FORMAT ABOVE \$5B,\$5E,\$A5,\$69 (B) FORMAT \$A8,\$AD,\$29,\$00 (C) FORMAT \$6D,\$9C,\$A5,\$69 (D) FORMAT \$34,\$11,\$A5,\$69 (E) FORMAT \$5A,\$48,\$26,\$62 \$54,\$44,\$C8,\$54
F9D2: 1 F9D5: A F9D8: 2 F9DB: 2 F9DB: 1 F9E0: 0 F9E3: 55 F9E6: 2 F9E8: A F9E8: A F9E8: A F9E8: 7 F9F0: 1 F9F3: 99 F9F6: 2 F9F8: 8 F9F8: 1 F9F8: 2 F3F8: 8 F9F8: 1 F9F8: 2 FA00: D FA03: 4 FA06: 9 FA09: 4 FA06: 9 FA09: 4 FA06: 9 FA09: 4 FA06: 9 FA12: 0 FA12: 0 FA12: 0 FA13: F	9 AE 4 53 3 24 9 0 D A 53 9 0 D A 54 4 24 4 2 29 0 0 0 C 5 0 2 53 1 3 3 62 2 6 2 8 8 4 4 4 4 8 4 0 8 4 4 8 4 8 4 4 9 0 0 4 1 3 0 0 2 5 1 3 1 0	69 23 1B 53 5B 69 A8 00 60 60 34 69 34 69 5A 62 54 54 88 4 74 6E CC		DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	<pre>\$19,\$AE, \$24,\$53, \$19,\$A1 \$00,\$1A, \$24,\$24 \$AE,\$AE, \$7C,\$00 \$15,\$9C, \$29,\$53 \$84,\$13, \$23,\$A0 \$D8,\$62, \$94,\$88, \$68,\$44, \$08,\$84,</pre>	\$69,\$A8,\$19,\$23 \$1B,\$23,\$24,\$53 (A) FORMAT ABOVE \$5B,\$5B,\$A5,\$69 (B) FORMAT \$A8,\$AD,\$29,\$00 (C) FORMAT \$6D,\$9C,\$A5,\$69 (D) FORMAT \$34,\$11,\$A5,\$69 (E) FORMAT \$5A,\$48,\$26,\$62 \$54,\$44,\$C8,\$54 \$E8,\$94,\$00,\$E4 \$74,\$E4,\$28,\$6E
F9D2: 1 F9D2: 1 F9D5: A F9D8: 2 F9DE: 2 F9DE: 1 F9E0: 0 F9E3: 5 F9E6: 2 F9E6: 2 F9E8: A F9E8: A F9E8: A F9E8: 1 F9F7: 9 F9F8: 8 F9F8: 1 F9F8: 8 F9F8: 1 F9F8: 2 FA00: D FA03: 4 FA06: 9 FA09: 4 FA06: 9 FA09: 4 FA06: 9 FA12: 0 FA15: B FA18: 7 FA18: 7	9 AE 4 533 9 AE 4 533 9 AE 4 534 9 AE 4 54 9 AE 4 5	69 23 1B 53 5B 69 A8 00 6D 69 34 69 34 69 5A 69 5A 62 54 54 54 54 54 54 54 54 54 54 54 54 54		DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	<pre>\$19,\$AE, \$24,\$53, \$19,\$A1 \$00,\$1A, \$24,\$24 \$AE,\$AE, \$7C,\$00 \$15,\$9C, \$29,\$53 \$84,\$13, \$23,\$A0 \$D8,\$62, \$94,\$88, \$68,\$44, \$08,\$84, \$74,\$F4,</pre>	\$69,\$A8,\$19,\$23 \$1B,\$23,\$24,\$53 (A) FORMAT ABOVE \$5B,\$5B,\$A5,\$69 (B) FORMAT \$A8,\$AD,\$29,\$00 (C) FORMAT \$6D,\$9C,\$A5,\$69 (D) FORMAT \$34,\$11,\$A5,\$69 (E) FORMAT \$5A,\$48,\$26,\$62 \$54,\$44,\$C8,\$54 \$E8,\$94,\$00,\$B4 \$74,\$B4,\$28,\$6E \$CC,\$4A,\$72,\$F2
F9D2: 1 F9D2: 2 F9D8: 2 F9D8: 2 F9D8: 2 F9D8: 2 F9D8: 2 F9E8: 1 F9E9: 0 F9E3: 5 F9E6: 2 F9E8: A F9E8: A F9E8: A F9E9: 1 F9F6: 2 F9F6: 2 F9F6: 2 F9F6: 2 F9F8: 8 F9F8: 1 F9F8: 8 F9F8: 1 F9F8: 2 FA00: D FA03: 4 FA06: 9 FA09: 4 FA06: 9 FA12: 0 FA15: B FA18: 7 FA18: 7 FA18: 4 FA18: A	9 AE 4 53 9 AE 4 53	69 23 1B 53 5B 69 A8 00 60 69 34 69 34 69 5A 62 54 E8 B4 74 6E CC F2		DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	<pre>\$19,\$AE, \$24,\$53, \$19,\$A1 \$00,\$1A, \$24,\$24 \$AE,\$AE, \$7C,\$00 \$15,\$9C, \$29,\$53 \$84,\$13, \$23,\$A0 \$D8,\$62, \$94,\$88, \$68,\$44, \$08,\$84, \$74,\$F4,</pre>	\$69,\$A8,\$19,\$23 \$1B,\$23,\$24,\$53 (A) FORMAT ABOVE \$5B,\$5B,\$A5,\$69 (B) FORMAT \$A8,\$AD,\$29,\$00 (C) FORMAT \$6D,\$9C,\$A5,\$69 (D) FORMAT \$34,\$11,\$A5,\$69 (E) FORMAT \$5A,\$48,\$26,\$62 \$54,\$44,\$C8,\$54 \$E8,\$94,\$00,\$E4 \$74,\$E4,\$28,\$6E
F9D2: 1 F9D5: A F9D8: 2 F9DB: 2 F9DE: 1 F9E0: 0 F9E3: 55 F9E6: 2 F9E8: A F9E8: A F9E8: A F9E8: 7 F9F0: 1 F9F7: 2 F9F7: 2 F9F8: 8 F9F8: 1 F9F8: 2 FA00: D FA03: 4 FA06: 9 FA09: 4 FA06: 9 FA12: 0 FA15: B FA18: 7 FA18: 4 FA18: 4 FA18: 4 FA20: 0	9 AE 4 533 9 AE 4	69 23 1B 53 5B 69 A8 00 6D 69 34 69 34 69 5A 62 54 54 54 88 84 74 6E CC F2 A2		DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	<pre>\$19,\$AE, \$24,\$53, \$19,\$A1 \$00,\$1A, \$24,\$24 \$AE,\$AE, \$7C,\$00 \$15,\$9C, \$29,\$53 \$84,\$13, \$23,\$A0 \$D8,\$62, \$94,\$88, \$68,\$44, \$08,\$84, \$08,\$84, \$74,\$F4, \$A4,\$8A</pre>	\$69,\$A8,\$19,\$23 \$1B,\$23,\$24,\$53 (A) FORMAT ABOVE \$5B,\$5B,\$A5,\$69 (B) FORMAT \$A8,\$AD,\$29,\$00 (C) FORMAT \$6D,\$9C,\$A5,\$69 (D) FORMAT \$34,\$11,\$A5,\$69 (E) FORMAT \$5A,\$48,\$26,\$62 \$54,\$44,\$C8,\$54 \$E8,\$94,\$00,\$B4 \$74,\$B4,\$28,\$6E \$CC,\$4A,\$72,\$F2 (A) FORMAT
F9D2: 1 F9D5: A F9D8: 2 F9DB: 2 F9DE: 1 F9E0: 0 F9E3: 55 F9E6: 2 F9E8: A F9E8: A F9E8: A F9E8: 7 F9F0: 1 F9F3: 99 F9F6: 2 F9F6: 2 F9F8: 8 F9F8: 1 F9F8: 1 F9F8: 2 FA00: D FA03: 4 FA06: 9 FA09: 4 FA06: 9 FA12: 0 FA15: B FA18: 7 FA18: 7 FA18: 4 FA22: 0 FA23: A	9 AE 4 533 9 AE 4 533 9 0 AA 544 9 0 AA 4 4 4 4 4 9 0 AA 4 4 0 AA 4 9 0 AA 54 9 0 AA 544 9 0 0 000 9 0 0000 9 0 000 9 0 0000 9 0 000 9 0000 9 0 000 9 0 0000 9 0 000 9 0 0000 9 0 000 9	 69 23 1B 53 5B 69 A8 00 6D 69 34 69 34 69 54 <		DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	<pre>\$19,\$AE, \$24,\$53, \$19,\$A1 \$00,\$1A, \$24,\$24 \$AE,\$AE, \$7C,\$00 \$15,\$9C, \$29,\$53 \$84,\$13, \$23,\$A0 \$D8,\$62, \$94,\$88, \$68,\$44, \$08,\$84, \$74,\$F4, \$A4,\$8A \$00,\$AA,</pre>	\$69,\$A8,\$19,\$23 \$1B,\$23,\$24,\$53 (A) FORMAT ABOVE \$5B,\$5B,\$A5,\$69 (B) FORMAT \$A8,\$AD,\$29,\$00 (C) FORMAT \$6D,\$9C,\$A5,\$69 (D) FORMAT \$34,\$11,\$A5,\$69 (E) FORMAT \$5A,\$48,\$26,\$62 \$54,\$44,\$C8,\$54 \$E8,\$94,\$00,\$B4 \$74,\$B4,\$28,\$6E \$CC,\$4A,\$72,\$F2 (A) FORMAT \$A2,\$A2,\$74,\$74
F9D2: 1 F9D8: 2 F9D8: 2 F9D8: 2 F9D8: 2 F9D8: 2 F9D8: 2 F9E8: A F9E8: A F9E8: A F9E8: A F9E7: 9 F9F6: 2 F9F8: 8 F9F8: 1 F9F8: 2 FA00: D FA03: 4 FA06: 9 FA15: B FA18: 7 FA18: 7 FA18: 4 FA20: 0 FA23: A FA26: 7	9 AE 4 533 9 AE 4 533 9 0 A 544 4 2 4 5 2 4 4 5 2 4 4 5 2 4 4 5 2 4 4 5 2 4 4 5 2 4 5 2 6 2 9 5 3 1 3 5 2 6 5 3 1 3 5 2 6 5 5 2 5 2 5 2 5 2 5 2 5 2 5 2 5 2	69 23 1B 53 5B 69 A8 00 60 69 34 69 34 69 5A 62 54 54 54 54 54 54 54 54 54 274		DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	<pre>\$19,\$AE, \$24,\$53, \$19,\$A1 \$00,\$1A, \$24,\$24 \$AE,\$AE, \$7C,\$00 \$15,\$9C, \$29,\$53 \$84,\$13, \$23,\$A0 \$D8,\$62, \$94,\$88, \$68,\$44, \$08,\$84, \$74,\$F4, \$A4,\$8A \$00,\$AA,</pre>	\$69,\$A8,\$19,\$23 \$1B,\$23,\$24,\$53 (A) FORMAT ABOVE \$5B,\$5B,\$A5,\$69 (B) FORMAT \$A8,\$AD,\$29,\$00 (C) FORMAT \$6D,\$9C,\$A5,\$69 (D) FORMAT \$34,\$11,\$A5,\$69 (E) FORMAT \$5A,\$48,\$26,\$62 \$54,\$44,\$C8,\$54 \$E8,\$94,\$00,\$B4 \$74,\$B4,\$28,\$6E \$CC,\$4A,\$72,\$F2 (A) FORMAT
F9D2: 1 F9D2: 2 F9D8: 2 F9D8: 2 F9D8: 2 F9D8: 2 F9D8: 2 F9E9: 1 F9E0: 0 F9E3: 5 F9E6: 2 F9E8: A F9E8: A F9E8: A F9E9: 1 F9F6: 2 FA00: D FA07: 9 FA00: 4 FA06: 9 FA09: 4 FA06: 9 FA09: 4 FA06: 9 FA12: 0 FA12: 0 FA15: B FA18: 7 FA18: 4 FA18: A FA20: 0 FA23: A FA22: 7 FA23: A FA22: 7 FA23: A	9 AE 4 533 9 AE 4 533 9 AE 4 524 1 A54 9 0 B 4 24 4 E 2 90 0 0 C 5 0 45 1 A 54 4 2 4 E 0 0 0 C 5 0 0 C 5 0 4 1 A 50 2 6 4 4 8 4 8 4 4 8 4 8 4 4 7 2 4 4 4 8 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 4 4 4 4 4 5 4 5 4 5 6 6 6 6 6 6 6 6 6 6 6 6 6 6	69 23 1B 53 5B 69 A8 00 6D 69 34 69 34 69 5A 62 54 54 54 54 54 54 54 54 54 54 74 88 84 74 82		DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	<pre>\$19,\$AE, \$24,\$53, \$19,\$A1 \$00,\$1A, \$24,\$24 \$AE,\$AE, \$7C,\$00 \$15,\$9C, \$29,\$53 \$84,\$13, \$23,\$A0 \$D8,\$62, \$94,\$88, \$68,\$44, \$08,\$84, \$08,\$84, \$74,\$F4, \$A4,\$8A \$00,\$AA, \$74,\$72</pre>	\$69,\$A8,\$19,\$23 \$1B,\$23,\$24,\$53 (A) FORMAT ABOVE \$5B,\$5B,\$A5,\$69 (B) FORMAT \$A8,\$AD,\$29,\$00 (C) FORMAT \$6D,\$9C,\$A5,\$69 (D) FORMAT \$34,\$11,\$A5,\$69 (E) FORMAT \$5A,\$48,\$26,\$62 \$54,\$44,\$C8,\$54 \$E8,\$94,\$00,\$E4 \$74,\$E4,\$28,\$6E \$CC,\$4A,\$72,\$F2 (A) FORMAT \$A2,\$A2,\$74,\$74 (B) FORMAT
F9D2: 1 F9D2: 2 F9D8: 2 F9D8: 2 F9D8: 2 F9D8: 2 F9D8: 2 F9D8: 2 F9E8: A F9E8: A F9E8: A F9E8: A F9E70: 1 F9F8: 8 F9F8: 1 F9F8: 1 F9F8: 2 FA00: D FA03: 4 FA06: 9 FA12: 0 FA12: 0 FA18: 7 FA18: 7 FA18: 7 FA18: 7 FA18: 4 FA20: 0 FA23: A FA26: 7 FA28: 4 FA28: 3	9 AE 4 53 9 AE 4 53 9 0 B 4 24 4 E 2 00 C 5 0 4 3 1 A 5 4 4 E 2 00 C 5 0 4 1 3 3 8 2 6 8 8 8 4 4 8 1 8 2 8 4 4 4 7 2 8 A 4 4 7 2 8 A 4 4 7 2 8 A 4 4 7 2 6 8 2 8 2 8 2 8 2 8 2 8 2 8 2 8 2 8 2	69 23 1B 53 5B 69 A8 00 6D 69 34 69 34 69 5A 62 54 54 E8 B4 74 6E CC F2 A2 74 B2 00		DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	<pre>\$19,\$AE, \$24,\$53, \$19,\$A1 \$00,\$1A, \$24,\$24 \$AE,\$AE, \$7C,\$00 \$15,\$9C, \$29,\$53 \$84,\$13, \$23,\$A0 \$D8,\$62, \$94,\$88, \$68,\$44, \$08,\$84, \$08,\$84, \$74,\$F4, \$A4,\$8A \$00,\$AA, \$74,\$72 \$44,\$68,</pre>	\$69,\$A8,\$19,\$23 \$1B,\$23,\$24,\$53 (A) FORMAT ABOVE \$5B,\$5B,\$A5,\$69 (B) FORMAT \$A8,\$AD,\$29,\$00 (C) FORMAT \$6D,\$9C,\$A5,\$69 (D) FORMAT \$34,\$11,\$A5,\$69 (E) FORMAT \$5A,\$48,\$26,\$62 \$54,\$44,\$C8,\$54 \$E8,\$94,\$00,\$B4 \$74,\$B4,\$28,\$6E \$CC,\$4A,\$72,\$F2 (A) FORMAT \$A2,\$A2,\$74,\$74 (B) FORMAT \$B2,\$32,\$B2,\$00
F9D2: 1 F9D2: 2 F9D8: 2 F9D8: 2 F9D8: 2 F9D8: 1 F9E0: 0 F9E3: 53 F9E6: 2 F9E8: A F9E8: A F9E7: 1 F9F8: 8 F9F6: 2 FA06: 9 FA00: 0 FA06: 9 FA12: 0 FA15: B FA15: 7 FA15: 7 FA20: 0 FA23: A FA20: 0 FA23: A FA26: 7 FA28: 4 FA28: 4 FA28: 4	9 AE	69 23 1B 53 5B 69 A8 00 60 69 34 69 34 69 54 54 54 54 54 54 254 54 254 254 274 82 74 82 00		DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	<pre>\$19,\$AE, \$24,\$53, \$19,\$A1 \$00,\$1A, \$24,\$24 \$AE,\$AE, \$7C,\$00 \$15,\$9C, \$29,\$53 \$84,\$13, \$23,\$A0 \$D8,\$62, \$94,\$88, \$68,\$44, \$08,\$84, \$08,\$84, \$74,\$F4, \$A4,\$8A \$00,\$AA, \$74,\$72 \$44,\$68,</pre>	\$69,\$A8,\$19,\$23 \$1B,\$23,\$24,\$53 (A) FORMAT ABOVE \$5B,\$5B,\$A5,\$69 (B) FORMAT \$A8,\$AD,\$29,\$00 (C) FORMAT \$6D,\$9C,\$A5,\$69 (D) FORMAT \$34,\$11,\$A5,\$69 (E) FORMAT \$5A,\$48,\$26,\$62 \$54,\$44,\$C8,\$54 \$E8,\$94,\$00,\$E4 \$74,\$E4,\$28,\$6E \$CC,\$4A,\$72,\$F2 (A) FORMAT \$A2,\$A2,\$74,\$74 (B) FORMAT
F9D2: 1 F9D2: A F9D8: 2 F9D8: 2 F9D8: 1 F9E0: 0 F9E2: 1 F9E3: 53 F9E6: 2 F9E8: A F9E7: 1 F9E7: 2 F9F6: 2 F9F8: 8 F9F8: 1 F9F8: 2 FA00: 0 FA03: 4 FA06: 9 FA12: 0 FA18: 7 FA18: 7 FA18: 7 FA18: 4 FA20: 0 FA23: A FA26: 7 FA28: 4 FA28: 3 FA28: 3 FA28: 3 FA28: 3 FA28: 1	9 AE	69 23 1B 53 5B 69 A8 00 6D 69 34 69 34 69 5A 62 54 54 54 54 54 54 54 254 74 6E CC2 72 74 B2 00 26		DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	<pre>\$19,\$AE, \$24,\$53, \$19,\$A1 \$00,\$1A, \$24,\$24 \$AE,\$AE, \$7C,\$00 \$15,\$9C, \$29,\$53 \$84,\$13, \$23,\$A0 \$D8,\$62, \$94,\$88, \$68,\$44, \$08,\$84, \$08,\$84, \$74,\$F4, \$A4,\$8A \$00,\$AA, \$74,\$72 \$44,\$68, \$22,\$00</pre>	\$69,\$A8,\$19,\$23 \$1B,\$23,\$24,\$53 (A) FORMAT ABOVE \$5B,\$5E,\$A5,\$69 (B) FORMAT \$A8,\$AD,\$29,\$00 (C) FORMAT \$6D,\$9C,\$A5,\$69 (D) FORMAT \$34,\$11,\$A5,\$69 (E) FORMAT \$5A,\$48,\$26,\$62 \$54,\$44,\$C8,\$54 \$E8,\$94,\$00,\$B4 \$74,\$B4,\$28,\$6E \$CC,\$4A,\$72,\$F2 (A) FORMAT \$A2,\$A2,\$74,\$74 (B) FORMAT \$B2,\$32,\$B2,\$00 (C) FORMAT
F9D2: 1 F9D2: 1 F9D8: 2 F9D8: 2 F9D8: 1 F9E0: 0 F9E2: 2 F9E8: A F9E8: A F9E7: 7 F9F6: 2 F9F8: 8 F9F8: 8 F9F8: 1 F9F8: 1 F9F8: 1 F9F8: 2 FA00: 0 FA03: 4 FA06: 9 FA12: 0 FA15: B FA18: 7 FA18: 7 FA18: 4 FA20: 0 FA23: A FA26: 7 FA28: 4 FA28: 4 FA28: 2 FA33: 2	9 AE 4 533 9 0 B 4 24 E 290 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	69 23 1B 53 5B 69 A8 00 6D 69 34 69 34 69 5A 62 54 54 54 54 54 54 54 254 74 6E CC F2 A2 74 B2 00 26 72		DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	<pre>\$19,\$AE, \$24,\$53, \$19,\$A1 \$00,\$1A, \$24,\$24 \$AE,\$AE, \$7C,\$00 \$15,\$9C, \$29,\$53 \$84,\$13, \$23,\$A0 \$D8,\$62, \$94,\$88, \$68,\$44, \$08,\$84, \$08,\$84, \$74,\$F4, \$A4,\$8A \$00,\$AA, \$74,\$72 \$44,\$68, \$22,\$00 \$1A,\$1A,</pre>	\$69,\$A8,\$19,\$23 \$1B,\$23,\$24,\$53 (A) FORMAT ABOVE \$5B,\$5B,\$A5,\$69 (B) FORMAT \$A8,\$AD,\$29,\$00 (C) FORMAT \$6D,\$9C,\$A5,\$69 (D) FORMAT \$34,\$11,\$A5,\$69 (E) FORMAT \$5A,\$48,\$26,\$62 \$54,\$44,\$C8,\$54 \$E8,\$94,\$00,\$E4 \$74,\$E4,\$28,\$6E \$CC,\$4A,\$72,\$F2 (A) FORMAT \$A2,\$A2,\$74,\$74 (B) FORMAT \$B2,\$32,\$E2,\$00 (C) FORMAT
F9D2: 1 F9D2: 2 F9D8: 2 F9D8: 2 F9D8: 2 F9D8: 2 F9D8: 2 F9E8: A F9E8: A F9E8: A F9E8: A F9E8: A F9F6: 2 F9F8: B F9F8: B F9F8: B F9F8: 1 F000: D FA06: 9 FA12: 0 FA18: 7 FA18: 7 FA18: 7 FA18: 4 FA20: 0 FA23: A FA26: 7 FA28: 4 FA28: 4 FA28: <td< td=""><td>9 AE 93 4 533 4 1933 4 1933 4 2414 5</td><td>69 23 1B 53 5B 69 A8 00 6D 69 34 69 34 69 5A 69 5A 69 5A 69 5A 24 54 54 54 54 54 54 24 74 82 00 26 72</td><td></td><td>DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB</td><td><pre>\$19,\$AE, \$24,\$53, \$19,\$A1 \$00,\$1A, \$24,\$24 \$AE,\$AE, \$7C,\$00 \$15,\$9C, \$29,\$53 \$84,\$13, \$23,\$A0 \$D8,\$62, \$94,\$88, \$68,\$44, \$08,\$84, \$08,\$84, \$74,\$F4, \$A4,\$8A \$00,\$AA, \$74,\$72 \$44,\$68, \$22,\$00 \$1A,\$1A,</pre></td><td>\$69,\$A8,\$19,\$23 \$1B,\$23,\$24,\$53 (A) FORMAT ABOVE \$5B,\$5E,\$A5,\$69 (B) FORMAT \$A8,\$AD,\$29,\$00 (C) FORMAT \$6D,\$9C,\$A5,\$69 (D) FORMAT \$34,\$11,\$A5,\$69 (E) FORMAT \$5A,\$48,\$26,\$62 \$54,\$44,\$C8,\$54 \$E8,\$94,\$00,\$B4 \$74,\$B4,\$28,\$6E \$CC,\$4A,\$72,\$F2 (A) FORMAT \$A2,\$A2,\$74,\$74 (B) FORMAT \$B2,\$32,\$B2,\$00 (C) FORMAT</td></td<>	9 AE 93 4 533 4 1933 4 1933 4 2414 5	69 23 1B 53 5B 69 A8 00 6D 69 34 69 34 69 5A 69 5A 69 5A 69 5A 24 54 54 54 54 54 54 24 74 82 00 26 72		DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	<pre>\$19,\$AE, \$24,\$53, \$19,\$A1 \$00,\$1A, \$24,\$24 \$AE,\$AE, \$7C,\$00 \$15,\$9C, \$29,\$53 \$84,\$13, \$23,\$A0 \$D8,\$62, \$94,\$88, \$68,\$44, \$08,\$84, \$08,\$84, \$74,\$F4, \$A4,\$8A \$00,\$AA, \$74,\$72 \$44,\$68, \$22,\$00 \$1A,\$1A,</pre>	\$69,\$A8,\$19,\$23 \$1B,\$23,\$24,\$53 (A) FORMAT ABOVE \$5B,\$5E,\$A5,\$69 (B) FORMAT \$A8,\$AD,\$29,\$00 (C) FORMAT \$6D,\$9C,\$A5,\$69 (D) FORMAT \$34,\$11,\$A5,\$69 (E) FORMAT \$5A,\$48,\$26,\$62 \$54,\$44,\$C8,\$54 \$E8,\$94,\$00,\$B4 \$74,\$B4,\$28,\$6E \$CC,\$4A,\$72,\$F2 (A) FORMAT \$A2,\$A2,\$74,\$74 (B) FORMAT \$B2,\$32,\$B2,\$00 (C) FORMAT
F9D2: 1 F9D2: 2 F9D8: 2 F9D8: 2 F9D8: 2 F9D8: 2 F9D8: 2 F9D8: 2 F9E8: A F9E70: 1 F9F8: 8 F9F8: 8 F9F8: 1 F9F8: 2 FA00: D FA03: 4 FA00: 0 FA12: 0 FA15: B FA15: B FA18: 7 FA20: 0 FA23: A FA20: 0 FA23: A FA28: 4 FA28: 4 FA38: 2 FA38: 2	9 AE 4 533 9 AE 4 533 9 AE 4 534 9 AE 4 535 9 AE 4 535 1 A50 1 A50 2 AE 4 535 1 A50 2 AE 4 535 2 AE 4 5355 2 AE 4 53555 2 AE 4 535555 2 AE 4 535555 2 AE 4 5355555 2 AE 4 53555555 2 AE 4 535555555555555555555555555555555555	69 23 1B 53 5B 69 A8 00 6D 69 34 69 34 69 5A 62 54 54 54 54 54 54 254 74 82 74 82 00 26 72 26		DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	<pre>\$19,\$AE, \$24,\$53, \$19,\$A1 \$00,\$1A, \$24,\$24 \$AE,\$AE, \$7C,\$00 \$15,\$9C, \$29,\$53 \$84,\$13, \$23,\$A0 \$D8,\$62, \$94,\$88, \$68,\$44, \$08,\$84, \$08,\$84, \$74,\$F4, \$A4,\$8A \$00,\$AA, \$74,\$72 \$44,\$68, \$22,\$00 \$1A,\$1A, \$88,\$C8</pre>	\$69,\$A8,\$19,\$23 \$1B,\$23,\$24,\$53 (A) FORMAT ABOVE \$5B,\$5B,\$A5,\$69 (B) FORMAT \$A8,\$AD,\$29,\$00 (C) FORMAT \$6D,\$9C,\$A5,\$69 (D) FORMAT \$34,\$11,\$A5,\$69 (E) FORMAT \$5A,\$48,\$26,\$62 \$54,\$44,\$C8,\$54 \$E8,\$94,\$00,\$B4 \$74,\$B4,\$28,\$6E \$CC,\$4A,\$72,\$F2 (A) FORMAT \$A2,\$A2,\$74,\$74 (B) FORMAT \$26,\$26,\$72,\$72 (D) FORMAT
F9D2: 1 F9D2: 2 F9D8: 2 F9D8: 2 F9D8: 2 F9D8: 2 F9D8: 2 F9D8: 1 F9E0: 0 F9E3: 51 F9E6: 2 F9E8: A F9E8: A F9E8: A F9E7: 1 F9F6: 2 F9F8: 8 F9F8: 1 F9F8: 2 FA00: D FA03: 4 FA06: 9 FA15: B FA15: B FA15: R FA20: 0 FA23: A FA20: 0 FA23: A FA20: 0 FA23: A FA20: 0 FA23: A FA26: 7 FA28: 4 FA28: 3 FA28: 4 FA30: 1 FA33: 2 FA36: 8 FA38: 7	9 AE	69 23 1B 53 5B 69 A8 00 6D 69 34 69 34 69 54 54 54 54 54 54 54 254 24 74 62 74 80 74 82 74 82 74 82 74 82 74 82 74 82 74 82 74 82 74 82 74 82 74 82 74 83 74 83 74 74 74 74 74 74 74 74 74 74 74 74 74		DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	<pre>\$19,\$AE, \$24,\$53, \$19,\$A1 \$00,\$1A, \$24,\$24 \$AE,\$AE, \$7C,\$00 \$15,\$9C, \$29,\$53 \$84,\$13, \$23,\$A0 \$D8,\$62, \$94,\$88, \$68,\$44, \$08,\$84, \$08,\$84, \$74,\$F4, \$A4,\$8A \$00,\$AA, \$74,\$72 \$44,\$68, \$22,\$00 \$1A,\$1A, \$88,\$C8 \$C4,\$CA,</pre>	\$69,\$A8,\$19,\$23 \$1B,\$23,\$24,\$53 (A) FORMAT ABOVE \$5B,\$5B,\$A5,\$69 (B) FORMAT \$A8,\$AD,\$29,\$00 (C) FORMAT \$6D,\$9C,\$A5,\$69 (D) FORMAT \$34,\$11,\$A5,\$69 (E) FORMAT \$5A,\$48,\$26,\$62 \$54,\$44,\$C8,\$54 \$E8,\$94,\$00,\$B4 \$74,\$B4,\$28,\$6E \$CC,\$4A,\$72,\$F2 (A) FORMAT \$A2,\$A2,\$74,\$74 (B) FORMAT \$22,\$32,\$B2,\$00 (C) FORMAT \$26,\$26,\$72,\$72 (D) FORMAT
F9D2: 1 F9D2: 2 F9D8: 2 F9D8: 2 F9D8: 2 F9D8: 2 F9D8: 2 F9D8: 2 F9E8: A F9E70: 1 F9F8: 8 F9F8: 8 F9F8: 1 F9F8: 2 FA00: D FA03: 4 FA00: 0 FA12: 0 FA15: B FA15: B FA18: 7 FA20: 0 FA23: A FA20: 0 FA23: A FA28: 4 FA28: 4 FA38: 2 FA38: 2	9 AE	69 23 1B 53 5B 69 A8 00 6D 69 34 69 34 69 54 54 54 54 54 54 54 254 24 74 62 74 80 74 82 74 82 74 82 74 82 74 82 74 82 74 82 74 82 74 82 74 82 74 82 74 83 74 83 74 74 74 74 74 74 74 74 74 74 74 74 74		DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	<pre>\$19,\$AE, \$24,\$53, \$19,\$A1 \$00,\$1A, \$24,\$24 \$AE,\$AE, \$7C,\$00 \$15,\$9C, \$29,\$53 \$84,\$13, \$23,\$A0 \$D8,\$62, \$94,\$88, \$68,\$44, \$08,\$84, \$08,\$84, \$74,\$F4, \$A4,\$8A \$00,\$AA, \$74,\$72 \$44,\$68, \$22,\$00 \$1A,\$1A, \$88,\$C8 \$C4,\$CA,</pre>	\$69,\$A8,\$19,\$23 \$1B,\$23,\$24,\$53 (A) FORMAT ABOVE \$5B,\$5B,\$A5,\$69 (B) FORMAT \$A8,\$AD,\$29,\$00 (C) FORMAT \$6D,\$9C,\$A5,\$69 (D) FORMAT \$34,\$11,\$A5,\$69 (E) FORMAT \$5A,\$48,\$26,\$62 \$54,\$44,\$C8,\$54 \$E8,\$94,\$00,\$B4 \$74,\$B4,\$28,\$6E \$CC,\$4A,\$72,\$F2 (A) FORMAT \$A2,\$A2,\$74,\$74 (B) FORMAT \$26,\$26,\$72,\$72 (D) FORMAT

FA40:						\$FF,\$FF,\$FI	
FA43:	20	D0	F8	STEP	JSR	INSTDSP	DISASSEMBLE ONE INST
FA46:					PLA		AT (PCL,H)
FA47:		2C					ADJUST TO USER
FA49:		Ъ			PLA		STACK. SAVE
FA4A: FA4C:					STA		RTN ADR.
			FB		LDX LDA		INIT XEQ AREA
FA51:				AQINII	STA		INTI ABQ ANBA
FA53:					DEX	ngi ,n	
FA54:						XQINIT	
FA56:							USER OPCODE BYTE
FA58:	FO	42					SPECIAL IF BREAK
FA5A:	A4	2F			LDY	LENGTH	LEN FROM DISASSEMBLY
FA5C:	C9	20			CMP	#\$20	
FA5E:	FO	59			BEQ	XJSR	HANDLE JSR, RTS, JMP,
FA60:	C9	60					JMP (), RTI SPECIAL
FA62:					BEQ		
FA64:					CMP		
FA66:					BEQ		
FA68: FA6A:					CMP		
FA6A: FA6C:					CMP	XJMPAT #\$40	
FA6E:					BEQ		
FA70:					AND		
FA72:					EOR		
FA74:	C9	04					COPY USER INST TO XEQ AREA
FA76:	FO	02					
FA78:	B1	ЗA		XQ1	LDA	(PCL),Y	CHANGE REL BRANCH DISP TO 4 FOR
FA7A:	99	3C	00	XQ2	STA	XQT,Y	DISP TO 4 FOR
FA7D:					DEY		JMP TO BRANCH OR
FA7E:					BPL	XQ1	NBRANCH FROM XEQ. RESTORE USER REG CONTENTS.
FA80:							
FA83:				TDO	JMP	XQ.I.	XEQ USER OP FROM RAM
FA86: FA88:					STA PLA	ACC	(RETURN TO NBRANCH)
FA89:					PHA		**IRQ HANDLER
FA8A:					ASL		
FA8B:					ASL		
FA8C:	0A				ASL	A	
FA8D:	30	03					TEST FOR BREAK
FA8F:	6C	FE	03		JMP	(TROLOC)	USER ROUTINE VECTOR IN RAM
						(
FA92:	28			BREAK	PLP	(
					PLP JSR		SAVE REG'S ON BREAK
FA92: FA93: FA96:	20 68	4C	FF		PLP JSR PLA	SAV1	
FA92: FA93: FA96: FA97:	20 68 85	4C	FF		PLP JSR PLA STA	SAV1	SAVE REG'S ON BREAK
FA92: FA93: FA96: FA97: FA99:	20 68 85 68	4C 3A	FF		PLP JSR PLA STA PLA	SAV1 PCL	SAVE REG'S ON BREAK
FA92: FA93: FA96: FA97: FA99: FA9A:	20 68 85 68 85	4C 3A 3B	FF		PLP JSR PLA STA PLA STA	SAV1 PCL PCH	SAVE REG'S ON BREAK INCLUDING PC
FA92: FA93: FA96: FA97: FA99: FA9A: FA9C:	20 68 85 68 85 20	4C 3A 3B 82	FF F8	XBRK	PLP JSR PLA STA PLA STA JSR	SAV1 PCL PCH INSDS1	SAVE REG'S ON BREAK INCLUDING PC PRINT USER PC.
FA92: FA93: FA96: FA97: FA99: FA9A: FA9C: FA9F:	20 68 85 68 85 20 20	4C 3A 3B 82 DA	FF F8 FA	XBRK	PLP JSR PLA STA PLA STA JSR JSR	SAV1 PCL PCH INSDS1 RGDSP1	SAVE REG'S ON BREAK INCLUDING PC PRINT USER PC. AND REG'S
FA92: FA93: FA96: FA97: FA99: FA9A: FA9C: FA9F: FA92:	20 68 85 68 85 20 20 4C	4C 3A 3B 82 DA 65	FF F8 FA	XBRK	PLP JSR PLA STA PLA STA JSR JSR	SAV1 PCL PCH INSDS1 RGDSP1	SAVE REG'S ON BREAK INCLUDING PC PRINT USER PC.
FA92: FA93: FA96: FA97: FA99: FA9A: FA9C: FA9F:	20 68 85 68 20 20 4C 18	4C 3A 3B 82 DA 65	FF F8 FA	XBRK	PLP JSR PLA STA PLA STA JSR JSR JMP	SAV1 PCL PCH INSDS1 RGDSP1	SAVE REG'S ON BREAK INCLUDING PC PRINT USER PC. AND REG'S
FA92: FA93: FA96: FA97: FA99: FA9A: FA9C: FA9F: FAA2: FAA5:	20 68 85 20 20 4C 18 68	4C 3A 3B 82 DA 65	FF F8 FA	XBRK	PLP JSR PLA STA PLA STA JSR JSR JMP CLC PLA	SAV1 PCL PCH INSDS1 RGDSP1	SAVE REG'S ON BREAK INCLUDING PC PRINT USER PC. AND REG'S GO TO MONITOR
FA92: FA93: FA96: FA97: FA99: FA94: FA97: FA97: FA42: FAA5: FAA6:	20 68 85 20 20 4C 18 68 85	4C 3A 3B 82 DA 65 48	FF F8 FA FF	XBRK	PLP JSR PLA STA PLA STA JSR JSR JMP CLC PLA	SAV1 PCL PCH INSDS1 RGDSP1 MON	SAVE REG'S ON BREAK INCLUDING PC PRINT USER PC. AND REG'S GO TO MONITOR SIMULATE RTI BY EXPECTING
FA92: FA96: FA96: FA97: FA99: FA94: FA97: FA97: FAA2: FAA5: FAA6: FAA7:	20 68 85 20 20 4C 18 68 85 68	4C 3A 3B 82 DA 65 48	FF F8 FA FF	XBRK XRTI	PLP JSR PLA STA PLA STA JSR JSR JMP CLC PLA STA	SAV1 PCL PCH INSDS1 RGDSP1 MON STATUS PCL	SAVE REG'S ON BREAK INCLUDING PC PRINT USER PC. AND REG'S GO TO MONITOR SIMULATE RTI BY EXPECTING STATUS FROM STACK, THEN RTS RTS SIMULATION EXTRACT PC FROM STACK
FA92: FA93: FA96: FA97: FA99: FA94: FA97: FA95: FAA5: FAA5: FAA5: FAA7: FAA9: FAA3: FAA3: FAA2: FAA3:	20 68 85 20 20 4C 18 68 85 68	4C 3A 3B 82 DA 65 48 3A	FF FA FF	XBRK XRTI XRTS	PLP JSR PLA STA PLA STA JSR JSR JMP CLC CLC PLA STA PLA	SAV1 PCL PCH INSDS1 RGDSP1 MON STATUS PCL	SAVE REG'S ON BREAK INCLUDING PC PRINT USER PC. AND REG'S GO TO MONITOR SIMULATE RTI BY EXPECTING STATUS FROM STACK, THEN RTS RTS SIMULATION
FA92: FA93: FA96: FA97: FA99: FA94: FA94: FA42: FAA5: FAA6: FAA7: FAA7: FAA9: FAA3: FAA3: FAA3:	20 68 85 68 85 20 20 4C 18 68 85 68 85 68	4C 3A 82 DA 65 48 3A 3B	FF FA FF	XBRK XRTI XRTS PCINC2	PLP JSR PLA STA PLA STA JSR JSR JSR CLC PLA STA PLA STA PLA STA	SAV1 PCL PCH INSDS1 RGDSP1 MON STATUS PCL PCH	SAVE REG'S ON BREAK INCLUDING PC PRINT USER PC. AND REG'S GO TO MONITOR SIMULATE RTI BY EXPECTING STATUS FROM STACK, THEN RTS RTS SIMULATION EXTRACT PC FROM STACK AND UPDATE PC BY 1 (LEN=0)
FA92: FA93: FA96: FA97: FA99: FA94: FA94: FA94: FA94: FA45: FAA5: FAA6: FAA7: FAA9: FAA7: FAA2: FAA2: FAA5:	20 68 85 68 85 20 20 4C 18 68 85 68 85 68 85 A5	4C 3A 3B 82 DA 65 48 3A 3B 2F	FF FA FF	XBRK XRTI XRTS	PLP JSR PLA STA PLA STA JSR JSR JSR JSR JSR JSR STA PLA STA PLA STA LDA	SAV1 PCL PCH INSDS1 RGDSP1 MON STATUS PCL PCH LENGTH	SAVE REG'S ON BREAK INCLUDING PC PRINT USER PC. AND REG'S GO TO MONITOR SIMULATE RTI BY EXPECTING STATUS FROM STACK, THEN RTS RTS SIMULATION EXTRACT PC FROM STACK
FA92: FA93: FA96: FA97: FA97: FA97: FA97: FA97: FAA2: FAA5: FAA5: FAA7: FAA7: FAAA2: FAAA2: FAAA2: FAAA1: FAA1: FAA1: FAA1: FAA1:	20 68 85 68 85 20 20 20 4C 18 68 85 68 85 68 85 20	4C 3A 3B 82 DA 65 48 3A 3B 2F 56	FF F8 FA FF	XBRK XRTI XRTS PCINC2	PLP JSR PLA STA PLA STA JSR JSR JSR JSR CLC CLC PLA STA PLA STA STA LDA JSR	SAV1 PCL PCH INSDS1 RGDSP1 MON STATUS PCL PCH LENGTH PCADJ3	SAVE REG'S ON BREAK INCLUDING PC PRINT USER PC. AND REG'S GO TO MONITOR SIMULATE RTI BY EXPECTING STATUS FROM STACK, THEN RTS RTS SIMULATION EXTRACT PC FROM STACK AND UPDATE PC BY 1 (LEN=0)
FA92: FA93: FA96: FA97: FA99: FA97: FA97: FA97: FAA2: FAA5: FAA6: FAA7: FAA7: FAAC: FAAC: FAAD: FAAD: FAAD: FAAD: FAAD: FAAD:	20 68 85 68 85 20 20 4C 18 68 85 68 85 68 85 20 84	4C 3A 3B 82 DA 65 48 3A 3B 2F 56	FF F8 FA FF	XBRK XRTI XRTS PCINC2	PLP JSR PLA STA PLA STA JSR JSR JSR CLC PLA STA PLA STA PLA STA LDA JSR STY	SAV1 PCL PCH INSDS1 RGDSP1 MON STATUS PCL PCH LENGTH	SAVE REG'S ON BREAK INCLUDING PC PRINT USER PC. AND REG'S GO TO MONITOR SIMULATE RTI BY EXPECTING STATUS FROM STACK, THEN RTS RTS SIMULATION EXTRACT PC FROM STACK AND UPDATE PC BY 1 (LEN=0)
FA92: FA93: FA96: FA97: FA97: FA97: FA97: FA97: FAA2: FAA5: FAA6: FAA7: FAA7: FAA7: FAA1: FAA1: FAB1: FAB4: FAB4:	20 68 85 20 20 4C 18 68 85 68 85 68 85 20 84 18	4C 3A 3B 82 DA 65 48 3A 3B 2F 56 3B	FF F8 FA FF	XBRK XRTI XRTS PCINC2 PCINC3	PLP JSR PLA STA PLA STA JSR JSR JMP CLC STA PLA STA PLA STA LDA JSR STY CLC	SAV1 PCL PCH INSDS1 RGDSP1 MON STATUS PCL PCH LENGTH PCADJ3 PCH	SAVE REG'S ON BREAK INCLUDING PC PRINT USER PC. AND REG'S GO TO MONITOR SIMULATE RTI BY EXPECTING STATUS FROM STACK, THEN RTS RTS SIMULATION EXTRACT PC FROM STACK AND UPDATE PC BY 1 (LEN=0)
FA92: FA93: FA96: FA97: FA99: FA97: FA97: FA97: FAA2: FAA5: FAA6: FAA7: FAA7: FAAC: FAAC: FAAD: FAAD: FAAD: FAAD: FAAD: FAAD:	20 68 85 20 20 4C 18 68 85 68 85 68 85 20 84 18 90	4C 3A 3B 82 DA 65 48 3A 3A 3B 2F 56 3B 14	FF F8 FA FF F9	XBRK XRTI XRTS PCINC2 PCINC3	PLP JSR PLA STA PLA STA JSR JSR JMP CLC STA PLA STA PLA STA PLA STA LDA JSR STY CLC	SAV1 PCL PCH INSDS1 RGDSP1 MON STATUS PCL PCH LENGTH PCADJ3	SAVE REG'S ON BREAK INCLUDING PC PRINT USER PC. AND REG'S GO TO MONITOR SIMULATE RTI BY EXPECTING STATUS FROM STACK, THEN RTS RTS SIMULATION EXTRACT PC FROM STACK AND UPDATE PC BY 1 (LEN=0)
FA92: FA93: FA96: FA97: FA97: FA97: FA92: FA45: FA45: FA45: FA45: FA47: FA47: FA47: FA47: FA47: FA481: FA81: FA86: FA87:	20 68 85 68 85 20 20 4C 18 68 85 68 85 68 85 20 84 18 90 18	4C 3A 3B 82 DA 65 48 3A 3B 2F 56 3B 14	FF F8 FA FF	XBRK XRTI XRTS PCINC2 PCINC3	PLP JSR PLA STA PLA STA JSR JSR JMP CLC PLA STA PLA STA PLA STA JSR STA JSR STA DLA STA CLC CLC	SAV1 PCL PCH INSDS1 RGDSP1 MON STATUS PCL PCH LENGTH PCADJ3 PCH NEWPCL	SAVE REG'S ON BREAK INCLUDING PC PRINT USER PC. AND REG'S GO TO MONITOR SIMULATE RTI BY EXPECTING STATUS FROM STACK, THEN RTS RTS SIMULATION EXTRACT PC FROM STACK AND UPDATE PC BY 1 (LEN=0)
FA92: FA93: FA96: FA97: FA97: FA97: FA97: FA97: FA97: FA45: FAA5: FAA6: FAA7: FAA9: FAA7: FAA2: FAA7: FAA6: FAA7: FAA8: FAA7: FAA8: FA86: FA97:	20 68 85 20 20 4C 18 68 85 68 85 68 85 20 84 18 90 18 20	4C 3A 3B 82 DA 65 48 3A 3B 2F 56 3B 14 54	FF F8 FA FF	XBRK XRTI XRTS PCINC2 PCINC3	PLP JSR PLA STA PLA STA JSR JSR JMP CLC PLA STA PLA STA PLA STA JSR STA JSR STA DLA STA CLC CLC	SAV1 PCL PCH INSDS1 RGDSP1 MON STATUS PCL PCH LENGTH PCADJ3 PCH NEWPCL PCADJ2	SAVE REG'S ON BREAK INCLUDING PC PRINT USER PC. AND REG'S GO TO MONITOR SIMULATE RTI BY EXPECTING STATUS FROM STACK, THEN RTS RTS SIMULATION EXTRACT PC FROM STACK AND UPDATE PC BY 1 (LEN=0) UPDATE PC BY LEN
FA92: FA93: FA96: FA97: FA97: FA97: FA97: FA97: FAA2: FAA2: FAA5: FAA5: FAA7: FAA7: FAA7: FAA7: FAA7: FAA8: FAB1: FAB6: FAB7: FAB4: FAB6: FAB4: FAB4:	20 68 85 68 85 20 20 4C 18 68 85 68 85 68 85 68 85 20 84 18 90 18 20 AA	4C 3A 3B 82 DA 65 48 3A 3B 2F 56 3B 14 54	FF F8 FA FF	XBRK XRTI XRTS PCINC2 PCINC3	PLP JSR PLA STA JSR JSR JSR JSR CLC PLA STA PLA STA PLA STA LDA JSR STY CLC BCC CLC JSR	SAV1 PCL PCH INSDS1 RGDSP1 MON STATUS PCL PCH LENGTH PCADJ3 PCH NEWPCL PCADJ2	SAVE REG'S ON BREAK INCLUDING PC PRINT USER PC. AND REG'S GO TO MONITOR SIMULATE RTI BY EXPECTING STATUS FROM STACK, THEN RTS RTS SIMULATION EXTRACT PC FROM STACK AND UPDATE PC BY 1 (LEN=0) UPDATE PC BY LEN UPDATE PC AND PUSH
FA92: FA93: FA96: FA97: FA96: FA97: FA97: FA97: FA97: FA45: FAA5: FAA6: FAA7: FAA9: FAA7: FAA9: FAA7: FAA9: FAA7: FAA9: FAA7: FAB1: FAB1: FAB2: FAB2: FAB2: FAB2: FAB3: FAB5: FAB5: FAB5:	20 68 85 20 20 4C 18 68 85 68 85 68 85 20 4C 18 85 68 85 20 20 4C 18 85 68 85 20 20 4C 18 85 68 85 20 20 20 4C 18 85 85 20 20 20 4C 18 85 20 20 20 4C 18 85 20 20 20 20 20 20 20 20 20 20 20 20 20	4C 3A 3B 82 DA 65 48 3A 3B 2F 56 3B 14 54	FF F8 FA FF	XBRK XRTI XRTS PCINC2 PCINC3	PLP JSR PLA STA JSR JSR JSR JSR CLC PLA STA PLA STA LDA JSR STY CLC BCC CLC JSR TAX	SAV1 PCL PCH INSDS1 RGDSP1 MON STATUS PCL PCH LENGTH PCADJ3 PCH NEWPCL PCADJ2	SAVE REG'S ON BREAK INCLUDING PC PRINT USER PC. AND REG'S GO TO MONITOR SIMULATE RTI BY EXPECTING STATUS FROM STACK, THEN RTS RTS SIMULATION EXTRACT PC FROM STACK AND UPDATE PC BY 1 (LEN=0) UPDATE PC BY LEN UPDATE PC AND PUSH ONTO STACH FOR
FA92: FA93: FA96: FA97: FA97: FA97: FA97: FA97: FA97: FA45: FAA7: FAA7: FAA9: FAA7: FAA9: FAA7: FAA9: FAA7: FAA9: FA81: FA81: FA86: FA87: FA89: FA85:	20 68 85 20 20 4C 18 85 68 85 68 85 20 84 18 90 84 18 90 84 85 85 85 85 85 85 85 85 85 85 85 85 85	4C 3A 3B 82 DA 65 48 3A 3B 2F 56 3B 14 54	FF F8 FA FF	XBRK XRTI XRTS PCINC2 PCINC3	PLP JSR PLA STA JSR JSR JSR JSR PLA STA PLA STA PLA STA PLA STA JSR LDA JSR CLC CLC JSR TAX TYA PHA TXA	SAV1 PCL PCH INSDS1 RGDSP1 MON STATUS PCL PCH LENGTH PCADJ3 PCH NEWPCL PCADJ2	SAVE REG'S ON BREAK INCLUDING PC PRINT USER PC. AND REG'S GO TO MONITOR SIMULATE RTI BY EXPECTING STATUS FROM STACK, THEN RTS RTS SIMULATION EXTRACT PC FROM STACK AND UPDATE PC BY 1 (LEN=0) UPDATE PC BY LEN UPDATE PC AND PUSH ONTO STACH FOR
FA92: FA93: FA96: FA97: FA97: FA97: FA97: FA97: FA42: FA42: FA42: FA42: FA42: FA44:	20 68 85 68 85 20 20 4C 18 68 85 68 85 20 84 18 90 18 20 84 18 90 18 20 84 85 85 85 85 85 84 85 84 85 84 84 83 84 83 84 84 83 84 84 84 84 85 84 85 85 85 85 85 85 85 85 85 85 85 85 85	4C 3A 3B 82 DA 65 48 3A 3B 2F 56 3B 14 54	FF F8 FA FF F9	XBRK XRTI XRTS PCINC2 PCINC3	PLP JSR PLA STA JSR JSR JSR JSR CLC PLA STA PLA STA PLA STA LDA JSR STY CLC BCC CLC BCC CLC JSR TAX TYA PHA TXA PHA	SAV1 PCL PCH INSDS1 RGDSP1 MON STATUS PCL PCH LENGTH PCADJ3 PCH NEWPCL PCADJ2	SAVE REG'S ON BREAK INCLUDING PC PRINT USER PC. AND REG'S GO TO MONITOR SIMULATE RTI BY EXPECTING STATUS FROM STACK, THEN RTS RTS SIMULATION EXTRACT PC FROM STACK AND UPDATE PC BY 1 (LEN=0) UPDATE PC BY LEN UPDATE PC AND PUSH ONTO STACH FOR
FA92: FA93: FA96: FA97: FA97: FA97: FA92: FA97: FAA2: FAA2: FAA2: FAA2: FAA2: FAA3: FAA3: FAA3: FAA4: FAA4: FAB4: FAB4: FAB5: FAB5: FAB5: FAB5: FAB5: FAB5: FAB5: FAC0: FAC1: FAC2:	20 68 85 68 85 20 20 4C 85 68 85 68 85 68 85 20 84 18 90 18 20 84 18 90 84 85 85 84 85 85 84 85 84 84 82 84 83 84 83 84 83 84 84 83 84 84 84 84 85 84 85 85 85 85 85 85 85 85 85 85 85 85 85	4C 3A 3B 82 DA 65 48 3A 3B 2F 56 3B 14 54	FF FA FF F9	XBRK XRTI XRTS PCINC2 PCINC3 XJSR	PLP JSR PLA STA JSR JSR JSR JSR CLC PLA STA PLA STA PLA STA LDA JSR STY CLC BCC CLC CLC JSR TAX TYA PHA TXA PHA LDY	SAV1 PCL PCH INSDS1 RGDSP1 MON STATUS PCL PCH LENGTH PCADJ3 PCH NEWPCL PCADJ2	SAVE REG'S ON BREAK INCLUDING PC PRINT USER PC. AND REG'S GO TO MONITOR SIMULATE RTI BY EXPECTING STATUS FROM STACK, THEN RTS RTS SIMULATION EXTRACT PC FROM STACK AND UPDATE PC BY 1 (LEN=0) UPDATE PC BY LEN UPDATE PC AND PUSH ONTO STACH FOR
FA92: FA93: FA96: FA97: FA97: FA97: FA97: FA97: FAA2: FAA2: FAA2: FAA2: FAA3: FAA2: FAA4: FAA7: FAA3: FAA2: FAA4: FAA5: FAA5: FAB1: FAB4: FAB5: FAB5: FAB5: FAB5: FAB5: FAB5: FAB5: FAB5: FAC1: FAC2: FAC2: FAC2: FAC2: FAC2: FAC2: FAC2: FAC2: FAC2: FAC2: FAC2: FAC2: FAC2: FAC2: FAC2: FAC2: FAC3:	20 68 85 200 20 4C 85 68 85 68 85 20 20 4C 85 68 85 20 84 18 90 18 20 84 84 85 20 20 4C 84 18 90 18 20 20 20 20 20 20 20 20 20 20 20 20 20	4C 3A 3B 82 DA 65 48 3A 3B 2F 56 3B 14 54	FF F8 FA FF F9	XBRK XRTI XRTS PCINC2 PCINC3 XJSR	PLP JSR PLA STA PLA STA JSR JSR JSR JSR PLA STA PLA STA PLA STA LDA JSR STY CLC CLC CLC JSR TAX TYA PHA LDY CLC	SAV1 PCL PCH INSDS1 RGDSP1 MON STATUS PCL PCH LENGTH PCADJ3 PCH NEWPCL PCADJ2	SAVE REG'S ON BREAK INCLUDING PC PRINT USER PC. AND REG'S GO TO MONITOR SIMULATE RTI BY EXPECTING STATUS FROM STACK, THEN RTS RTS SIMULATION EXTRACT PC FROM STACK AND UPDATE PC BY 1 (LEN=0) UPDATE PC BY LEN UPDATE PC AND PUSH ONTO STACH FOR
FA92: FA93: FA96: FA97: FA99: FA94: FA96: FAA2: FAA2: FAA5: FAA6: FAA7: FAA6: FAA7: FAA7: FAA7: FAA7: FAA7: FAA7: FAA7: FAA7: FAA8: FAA8: FAB1: FAB1: FAB9: FAB9: FAB9: FAB2:	20 68 85 20 20 4C 18 68 85 20 84 18 90 84 18 90 84 20 AA 98 84 85 20 84 18 90 84 18 90 84 85 20 84 85 20 84 85 85 20 20 84 85 85 20 20 20 20 20 20 20 20 20 20 20 20 20	4C 3A 3B 82 DA 65 48 3A 3B 2F 53B 14 54 02 3A	FF F8 FA FF F9	XBRK XRTI XRTS PCINC2 PCINC3 XJSR	PLP JSR PLA STA PLA STA JSR JSR JSR JSR CLC PLA STA PLA STA PLA STA LDA JSR STY CLC CLC JSR TAX TYA PHA TXA PHA LDY CLC LDA	SAV1 PCL PCH INSDS1 RGDSP1 MON STATUS PCL PCH LENGTH PCADJ3 PCH NEWPCL PCADJ2	SAVE REG'S ON BREAK INCLUDING PC PRINT USER PC. AND REG'S GO TO MONITOR SIMULATE RTI BY EXPECTING STATUS FROM STACK, THEN RTS RTS SIMULATION EXTRACT PC FROM STACK AND UPDATE PC BY 1 (LEN=0) UPDATE PC BY LEN UPDATE PC BY LEN UPDATE PC AND PUSH ONTO STACH FOR JSR SIMULATE
FA92: FA93: FA96: FA97: FA97: FA97: FA97: FA97: FA97: FA45: FA45: FAA7: FAA9: FAA7: FAA9: FAA7: FAA9: FAA7: FA47: FA47: FA47: FA47: FA481: FA4	20 68 85 20 20 20 4C 18 68 85 68 85 20 84 18 90 84 18 90 84 85 20 84 85 20 84 18 90 84 85 20 84 18 90 84 85 20 20 20 20 20 20 20 20 20 20 20 20 20	4C 3A 3B 82 DA 65 48 3A 3B 2F 53B 14 54 02 3A	FF F8 FA FF F9	XBRK XRTI XRTS PCINC2 PCINC3 XJSR	PLP JSR PLA STA PLA STA JSR JSR JMP CLC PLA STA PLA STA PLA STA PLA STA DLA STA STA CLC CLC JSR CLC CLC JSR TXA PHA TXA PHA LDY CLC CLC JSR TYA	SAV1 PCL PCH INSDS1 RGDSP1 MON STATUS PCL PCH LENGTH PCADJ3 PCH NEWPCL PCADJ2	SAVE REG'S ON BREAK INCLUDING PC PRINT USER PC. AND REG'S GO TO MONITOR SIMULATE RTI BY EXPECTING STATUS FROM STACK, THEN RTS RTS SIMULATION EXTRACT PC FROM STACK AND UPDATE PC BY 1 (LEN=0) UPDATE PC BY LEN UPDATE PC AND PUSH ONTO STACH FOR JSR SIMULATE
FA92: FA93: FA96: FA97: FA97: FA97: FA92: FA97: FAA2: FAA2: FAA5: FAA5: FAA7: FAA7: FAA7: FAA7: FAA7: FAB4: FAB4: FAB4: FAB5: FAB5: FAB5: FAB5: FAB5: FAB5: FAB5: FAB5: FAC5:	20 68 85 68 85 20 20 4C 18 68 85 68 85 68 85 68 85 68 85 20 20 4C 18 85 84 85 84 84 82 0 84 84 83 84 84 85 84 85 84 85 84 85 84 85 85 85 85 85 85 85 85 85 85 85 85 85	4C 3A 3B 82 DA 65 48 3A 3B 2F 56 3B 14 54 02 3A	FF F8 FA FF F9	XBRK XRTI XRTS PCINC2 PCINC3 XJSR	PLP JSR PLA STA JSR JSR JSR JSR CLC PLA STA PLA STA PLA STA JSR STY CLC STA JSR STY CLC BCC CLC JSR TAX PLA LDA JSR CLC LDA TXA PHA LDY CLC ZDA TAX PHA	SAV1 PCL PCH INSDS1 RGDSP1 MON STATUS PCL PCH LENGTH PCADJ3 PCH NEWPCL PCADJ2 #\$02 (PCL),Y	SAVE REG'S ON BREAK INCLUDING PC PRINT USER PC. AND REG'S GO TO MONITOR SIMULATE RTI BY EXPECTING STATUS FROM STACK, THEN RTS RTS SIMULATION EXTRACT PC FROM STACK AND UPDATE PC BY 1 (LEN=0) UPDATE PC BY LEN UPDATE PC BY LEN UPDATE PC AND PUSH ONTO STACH FOR JSR SIMULATE
FA92: FA93: FA96: FA97: FA97: FA97: FA97: FA97: FA97: FA45: FA45: FA45: FA47: FA45: FA47: F447:	20 68 85 68 20 20 4C 18 68 85 68 85 68 85 68 85 20 84 18 90 84 82 90 84 820 20 84 85 85 84 18 90 84 85 84 85 85 84 85 85 84 85 85 85 85 85 85 85 85 85 85 85 85 85	4C 3A 3B 82 DA 65 48 3A 3B 2F 56 3B 14 54 02 3A 3A	FF F8 FA FF F9	XBRK XRTI XRTS PCINC2 PCINC3 XJSR	PLP JSR PLA STA JSR JSR JSR JSR CLC PLA STA PLA STA PLA STA LDA JSR STY CLC BCC CLC JSR TAX TYA PHA LDY CLC LDA TAX DEY LDA	SAV1 PCL PCH INSDS1 RGDSP1 MON STATUS PCL PCH LENGTH PCADJ3 PCH NEWPCL PCADJ2	SAVE REG'S ON BREAK INCLUDING PC PRINT USER PC. AND REG'S GO TO MONITOR SIMULATE RTI BY EXPECTING STATUS FROM STACK, THEN RTS RTS SIMULATION EXTRACT PC FROM STACK AND UPDATE PC BY 1 (LEN=0) UPDATE PC BY LEN UPDATE PC AND PUSH ONTO STACH FOR JSR SIMULATE
FA92: FA93: FA96: FA97: FA97: FA97: FA97: FA97: FA42: FA42: FA42: FA45: FA45: FA47: FA47: FA44:	20 68 85 68 20 20 4C 18 68 85 68 85 68 85 68 85 68 85 68 85 68 85 20 84 18 90 20 84 18 82 84 84 848 848 848 848 848 848 848 8	4C 3A 3B 2DA 65 48 3A 3B 2F 56 3B 14 54 02 3A 3B	FF F8 FA FF F9	XBRK XRTI XRTS PCINC2 PCINC3 XJSR	PLP JSR PLA STA JSR JSR JSR JSR JSR PLA STA PLA STA PLA STA LDA JSR STY CLC BCC CLC JSR TAX TYA PHA LDY CLC LDA TXA PHA LDY CLC LDA STX	SAV1 PCL PCH INSDS1 RGDSP1 MON STATUS PCL PCH LENGTH PCADJ3 PCH NEWPCL PCADJ2 #\$02 (PCL),Y	SAVE REG'S ON BREAK INCLUDING PC PRINT USER PC. AND REG'S GO TO MONITOR SIMULATE RTI BY EXPECTING STATUS FROM STACK, THEN RTS RTS SIMULATION EXTRACT PC FROM STACK AND UPDATE PC BY 1 (LEN=0) UPDATE PC BY LEN UPDATE PC AND PUSH ONTO STACH FOR JSR SIMULATE
FA92: FA93: FA96: FA97: FA97: FA97: FA97: FA97: FA97: FA25: FAA5: FAA6: FAA7: FAA9: FAA7: FAA9: FAA7: FAA9: FAA7: FAA9: FAA7: FAA7: FAB1: FAB4: FAB7: FAB7: FAB7: FAB7: FAB7: FAB7: FAC1: FAC2: FAC2: FAC2: FAC3: FAC9:	20 68 85 68 20 20 4C 18 85 68 85 68 85 20 84 18 20 84 18 20 84 18 20 84 85 85 85 85 88 5 88 5 88 5 88 5 88	4C 3A 3B 2DA 65 48 3A 3B 2F6 3B 14 54 02 3A 3B 3A 3B 54 54	FF F8 FA FF F9 F9	XBRK XRTI XRTS PCINC2 PCINC3 XJSR XJSR	PLP JSR PLA STA JSR JSR JSR JMP CLC PLA STA PLA STA PLA STA PLA STA DLA STA DLA STA DLA STA DLA STA DLA STA DLA STA DLA STA DLA STA DLA STA STA DLA STA STA STA STA STA STA STA STA STA ST	SAV1 PCL PCH INSDS1 RGDSP1 MON STATUS PCL PCH LENGTH PCADJ3 PCH NEWPCL PCADJ2 #\$02 (PCL),Y (PCL),Y PCH LCD,Y PCH	SAVE REG'S ON BREAK INCLUDING PC PRINT USER PC. AND REG'S GO TO MONITOR SIMULATE RTI BY EXPECTING STATUS FROM STACK, THEN RTS RTS SIMULATION EXTRACT PC FROM STACK AND UPDATE PC BY 1 (LEN=0) UPDATE PC BY LEN UPDATE PC AND PUSH ONTO STACH FOR JSR SIMULATE
FA92: FA93: FA96: FA97: FA97: FA97: FA97: FA97: FA97: FA45: FA45: FA45: FA45: FA47: FA47: FA47: FA44:	20 68 85 68 20 20 4C 18 85 68 85 68 85 68 85 68 85 68 85 20 84 18 90 84 820 AA 98 84 83 84 84 83 84 84 85 84 85 84 85 84 85 84 85 85 84 85 85 85 85 85 85 85 85 85 85 85 85 85	4C 3A 3B 82 DA 65 48 3A 3B 2F 56 3B 14 54 02 3A 3B 3A 3B 3A 3B 2D 2D	FF F8 FA FF F9 F9	XBRK XRTI XRTS PCINC2 PCINC3 XJSR XJSR	PLP JSR PLA STA JSR JSR JSR JMP CLC PLA STA PLA STA PLA STA PLA STA DLA STA DLA STA DLA STA DLA STA DLA STA DLA STA DLA STA DLA STA DLA STA STA DLA STA STA STA STA STA STA STA STA STA ST	SAV1 PCL PCH INSDS1 RGDSP1 MON STATUS PCL PCH LENGTH PCADJ3 PCH NEWPCL PCADJ2 #\$02 (PCL),Y (PCL),Y PCH LCD,Y PCH	SAVE REG'S ON BREAK INCLUDING PC PRINT USER PC. AND REG'S GO TO MONITOR SIMULATE RTI BY EXPECTING STATUS FROM STACK, THEN RTS RTS SIMULATION EXTRACT PC FROM STACK AND UPDATE PC BY 1 (LEN=0) UPDATE PC BY LEN UPDATE PC AND PUSH ONTO STACH FOR JSR SIMULATE
FA92: FA93: FA96: FA97: FA97: FA97: FA97: FA97: FA42: FA42: FA42: FA42: FA42: FA42: FA42: FA44:	20 68 85 68 20 20 4C 18 68 5 68 85 68 85 68 85 20 84 18 90 84 20 20 84 18 90 84 85 84 84 80 84 84 80 84 84 80 84 80 84 80 80 80 80 80 80 80 80 80 80 80 80 80	4C 3A 3B 82 DA 65 48 3A 3B 2F 56 3B 14 54 02 3A 3B 3A 3B 3A 52 D	FF F8 FA FF F9 F9	XBRK XRTI XRTS PCINC2 PCINC3 XJSR XJSR	PLP JSR PLA STA JSR JSR JSR JMP CLC PLA STA PLA STA PLA STA PLA STA DLA STA DLA STA DLA STA DLA STA DLA STA DLA STA DLA STA DLA STA DLA STA STA DLA STA STA STA STA STA STA STA STA STA ST	SAV1 PCL PCH INSDS1 RGDSP1 MON STATUS PCL PCH LENGTH PCADJ3 PCH NEWPCL PCADJ2 #\$02 (PCL),Y (PCL),Y PCH LCD,Y PCH	SAVE REG'S ON BREAK INCLUDING PC PRINT USER PC. AND REG'S GO TO MONITOR SIMULATE RTI BY EXPECTING STATUS FROM STACK, THEN RTS RTS SIMULATION EXTRACT PC FROM STACK AND UPDATE PC BY 1 (LEN=0) UPDATE PC BY LEN UPDATE PC AND PUSH ONTO STACH FOR JSR SIMULATE
FA92: FA93: FA96: FA97: FA97: FA97: FA92: FA97: FAA2: FAA2: FAA2: FAA2: FAA2: FAA2: FAA2: FAA3: FAA3: FAA4: FAA9: FAA4: FAB4: FAB4: FAB4: FAB5: FAB5: FAB5: FAB5: FAB5: FAB5: FAB5: FAC6: FAC7: FAC6: FAC7:	20 68 85 68 20 20 4C 18 68 85 68 85 68 85 20 84 18 90 82 20 84 18 90 820 84 18 90 84 84 84 84 84 84 84 84 84 85 85 84 84 84 84 84 85 84 84 84 84 84 84 84 84 84 84 84 84 84	4C 3A 3B 82 DA 65 48 3A 3B 2F 56 3B 14 54 02 3A 3B 3A 3B 3A 52 D	FF F8 FA FF F9 F9	XBRK XRTI XRTS PCINC2 PCINC3 XJSR XJSR	PLP JSR PLA STA JSR JSR JSR JSR JSR STA PLA STA PLA STA PLA STA LDA JSR STY CLC BCC CLC JSR TAX TYA PHA TXA PHA LDY CLC LDA STX STX ECC CLC JSR TAX TYA PLA STA STA DSTA STA STA DSTA STA STA STA STA STA STA STA STA STA	SAV1 PCL PCH INSDS1 RGDSP1 MON STATUS PCL PCH LENGTH PCADJ3 PCH NEWPCL PCADJ2 #\$02 (PCL),Y CPCL),Y PCH PCH XJMP RTNH	SAVE REG'S ON BREAK INCLUDING PC PRINT USER PC. AND REG'S GO TO MONITOR SIMULATE RTI BY EXPECTING STATUS FROM STACK, THEN RTS RTS SIMULATION EXTRACT PC FROM STACK AND UPDATE PC BY 1 (LEN=0) UPDATE PC BY LEN UPDATE PC AND PUSH ONTO STACH FOR JSR SIMULATE
FA92: FA93: FA96: FA97: FA97: FA97: FA97: FA97: FA97: FA97: FA45: FAA2: FAA2: FAA3: FAA4: FAA7: FAA9: FAA7: FAA9: FAA7: FAA9: FAA7: FA81: FA84: FA81: FA84: FA84: FA84: FA85:	20 68 85 68 85 20 20 4C 18 68 85 68 85 68 85 68 85 68 85 68 85 82 0 84 18 90 18 20 20 4C 84 85 85 85 84 85 85 84 85 85 84 85 85 84 85 85 84 85 85 84 85 85 84 85 85 84 85 85 84 85 84 85 84 85 84 85 84 85 84 85 84 85 84 84 85 84 84 85 84 84 85 84 84 84 85 84 84 84 84 84 84 84 84 84 84 84 84 84	4C 3A 3B 82 DA 65 48 3A 3E 56 3B 14 54 02 3A 3B 3A 3B 3A 3B 3A 2D 2C	FF F8 FA FF F9	XBRK XRTI XRTS PCINC2 PCINC3 XJSR XJSR XJMP XJMPAT NEWPCL RTNJMP	PLP JSR PLA STA PLA STA JSR JSR JSR JMP CLC PLA STA PLA STA PLA STA PLA STA DEX CLC CLC CLC CLC CLC CLC JSR TXA PHA TXA PHA TXA PHA CLC STX STA PLA STY CLC CLC STA STY CLC STA STY DA STY CLC STA STY DA STY DA STY DA STY DA STA STA PLA STA STA STA PLA STA STA STA STA STA STA STA PLA STA STA STA STA STA STA STA STA STA ST	SAV1 PCL PCH INSDS1 RGDSP1 MON STATUS PCL PCH LENGTH PCADJ3 PCH NEWPCL PCADJ2 #\$02 (PCL),Y (PCL),Y PCH PCL XJMP RTNH RTNL	SAVE REG'S ON BREAK INCLUDING PC PRINT USER PC. AND REG'S GO TO MONITOR SIMULATE RTI BY EXPECTING STATUS FROM STACK, THEN RTS RTS SIMULATION EXTRACT PC FROM STACK AND UPDATE PC BY 1 (LEN=0) UPDATE PC BY LEN UPDATE PC AND PUSH ONTO STACH FOR JSR SIMULATE LOAD PC FOR JMP, (JMP) SIMULATE.
FA92: FA93: FA96: FA97: FA97: FA97: FA97: FA97: FA97: FA97: FA45: FAA5: FAA6: FAA7: FAA9: FAA7: FAA9: FAA7: FAA9: FAA7: FAA9: FAB1: FAB4: FAB4: FAB7: FAB7: FAB7: FAB7: FAB7: FAB7: FAB7: FAB7: FAB7: FAB7: FAB7: FAB7: FAB7: FAB7: FAB7: FAB7: FAB7: FAC1: FAC2: FAC2: FAC2: FAC3: FAC3: FAC3: FAC3: FAC4: FAC5: FAC5: FAC5: FAC5: FAC7: FAC5: FAC7: FAC8: FAC7: FAC8: FAC7: FAC8: FAC7: FAC8: FAC7: FAC9:	20 68 85 68 20 20 4C 18 85 68 85 68 85 68 85 68 85 68 85 20 84 18 20 84 18 20 84 85 85 88 5 88 5 88 5 88 5 88 5 88 5	4C 3A 3B 82 DA 65 48 3A 3B 2F 56 3B 14 54 02 3A 3B 3A 3B 2D 2C 8E	FF F8 FA FF F9 F9	XBRK XRTI XRTS PCINC2 PCINC3 XJSR XJSR XJMPAT NEWPCL RTNJMP	PLP JSR PLA STA JSR JSR JSR JSR JMP CLC PLA STA PLA STA PLA STA PLA STA PLA STA DLA JSR CLC CLC JSR TAX PLA STA DLA PLA STA DLA PLA STA DLA PLA STA DLA PLA STA DLA PLA STA DLA PLA STA DLA PLA STA DLA PLA STA DLA PLA STA DLA DLA PLA STA DLA DLA DLA DLA DLA DLA DLA DLA DLA DL	SAV1 PCL PCH INSDS1 RGDSP1 MON STATUS PCL PCH LENGTH PCADJ3 PCH NEWPCL PCADJ2 #\$02 (PCL),Y (PCL),Y (PCL),Y PCH RTNH RTNL CROUT	SAVE REG'S ON BREAK INCLUDING PC PRINT USER PC. AND REG'S GO TO MONITOR SIMULATE RTI BY EXPECTING STATUS FROM STACK, THEN RTS RTS SIMULATION EXTRACT PC FROM STACK AND UPDATE PC BY 1 (LEN=0) UPDATE PC BY LEN UPDATE PC AND PUSH ONTO STACH FOR JSR SIMULATE LOAD PC FOR JMP, (JMP) SIMULATE.
FA92: FA93: FA96: FA97: FA97: FA97: FA97: FA97: FA97: FA25: FA45: FA45: FA45: FA47: FA47: FA47: FA47: FA47: FA47: FA47: FA47: FA44:	20 68 85 68 20 20 4C 18 85 68 85 68 85 68 85 68 85 20 84 18 90 84 18 90 84 820 AA 98 84 8A 84 85 68 85 84 18 90 84 84 85 85 84 85 85 84 85 85 84 85 85 84 85 85 84 85 85 84 85 85 84 85 85 84 85 85 85 85 84 85 85 85 84 85 85 85 85 84 85 85 85 85 85 84 85 85 88 85 85 88 85 88 80 88 85 80 88 80 88 80 88 80 80 88 80 80 80 80	4C 3A 3B 82 DA 65 48 3A 3B 2F 56 3B 14 54 02 3A 3B 3A 3B 3A 3B 2C 8E 45	FF F8 FA FF F9 F9 F9	XBRK XRTI XRTS PCINC2 PCINC3 XJSR XJSR XJMP XJMPAT NEWPCL RTNJMP	PLP JSR PLA STA PLA STA JSR JMP CLC PLA STA PLA STA PLA STA PLA STA DLA JSR CLC JSR CLC JSR CLC JSR TAX PLA STA DLA DCLC JSR TAX PLA STA DLA STA DLA DCLC JSR TAX DLA DCLC JSR TAX DLA DCLC JSR TAX DLA DLA DCLC DLA STA DLA DLA DLA DLA DLA DLA DLA DLA DLA DL	SAV1 PCL PCH INSDS1 RGDSP1 MON STATUS PCL PCH LENGTH PCADJ3 PCH NEWPCL PCADJ2 #\$02 (PCL),Y (PCL),Y PCL XJMP RTNH RTNL CROUT #ACC	SAVE REG'S ON BREAK INCLUDING PC PRINT USER PC. AND REG'S GO TO MONITOR SIMULATE RTI BY EXPECTING STATUS FROM STACK, THEN RTS RTS SIMULATION EXTRACT PC FROM STACK AND UPDATE PC BY 1 (LEN=0) UPDATE PC BY LEN UPDATE PC AND PUSH ONTO STACH FOR JSR SIMULATE LOAD PC FOR JMP, (JMP) SIMULATE.
FA92: FA93: FA96: FA97: FA97: FA97: FA97: FA97: FA97: FA97: FA45: FAA5: FAA6: FAA7: FAA9: FAA7: FAA9: FAA7: FAA9: FAA7: FAA9: FAB1: FAB4: FAB7: FAB7: FAB7: FAB7: FAB7: FAB7: FAB7: FAB7: FAB7: FAB7: FAB7: FAB7: FAB7: FAB7: FAB7: FAB7: FAB7: FAB7: FAC1: FAC2: FAC2: FAC7: FAC3: FAC7: FAC8: FAC7: FAC8: FAC9:	20 68 85 68 20 20 4C 18 85 68 85 68 85 68 85 68 85 20 84 18 90 84 18 90 84 820 AA 98 84 8A 84 85 68 85 84 18 90 84 84 85 85 84 85 85 84 85 85 84 85 85 84 85 85 84 85 85 84 85 85 84 85 85 84 85 85 85 85 84 85 85 85 84 85 85 85 85 84 85 85 85 85 85 84 85 85 88 85 85 88 85 88 80 88 85 80 88 80 88 80 88 80 80 88 80 80 80 80	4C 3A 3B 82 DA 65 48 3A 3B 2F 56 3B 14 54 02 3A 3B 3A 3B 3A 3B 2C 8E 45	FF F8 FA FF F9 F9 F9	XBRK XRTI XRTS PCINC2 PCINC3 XJSR XJSR XJMPAT NEWPCL RTNJMP	PLP JSR PLA STA JSR JSR JSR JSR JMP CLC PLA STA PLA STA PLA STA PLA STA PLA STA DLA JSR CLC CLC JSR TAX PLA STA DLA PLA STA DLA PLA STA DLA PLA STA DLA PLA STA DLA PLA STA DLA PLA STA DLA PLA STA DLA PLA STA DLA PLA STA DLA DLA PLA STA DLA DLA DLA DLA DLA DLA DLA DLA DLA DL	SAV1 PCL PCH INSDS1 RGDSP1 MON STATUS PCL PCH LENGTH PCADJ3 PCH NEWPCL PCADJ2 #\$02 (PCL),Y (PCL),Y PCH XJMP RTNH RTNL CROUT #ACC A3L	SAVE REG'S ON BREAK INCLUDING PC PRINT USER PC. AND REG'S GO TO MONITOR SIMULATE RTI BY EXPECTING STATUS FROM STACK, THEN RTS RTS SIMULATION EXTRACT PC FROM STACK AND UPDATE PC BY 1 (LEN=0) UPDATE PC BY LEN UPDATE PC AND PUSH ONTO STACH FOR JSR SIMULATE LOAD PC FOR JMP, (JMP) SIMULATE.

FADE:	Α9	00				#ACC/256	
FAE0:	85	41			STA LDX	A3H	
FAE2:	A2	FB			LDX	#\$FB	
FAE4:	A9	A0		RDSP1	LDA	#\$A0	
FAE6:	20	ED	FD		JSR	COUT	
FAE9:						RTBL-\$FB,X	
FAEC:					JSR		
FAEF:					LDA		
FAF1:					JSR		
FAF4:	B5	4A			LDA	ACC+5,X	
FAF6:	20	DA	FD		JSR	PRBYTE	
FAF9:	E8				INX		
FAFA:					BMI	RDSP1	
FAFC:					RTS	REDITI	
FAFC:	60				RIS		
FAFD:	18			BRANCH	СГС		BRANCH TAKEN,
FAFE:	A0	01			LDY	#\$01	ADD LEN+2 TO PC
FB00:	Β1	ЗA			LDA	(PCL),Y PCADJ3	
FB02:	20	56	F9		JSR	PCADJ3	
FB05:	85	3A			STA	PCL	
FB07:					TYA		
FB08:					SEC		
						DOTNOO	
FB09:	<u>Б</u> О	AZ		NBRNCH	BCS TOD	PCINCZ	
LROB:	20	4A	F.F.	NBRNCH	JSR	SAVE	NORMAL RETURN AFTER
FB0E:					SEC		XEQ USER OF
FB0F:	В0	9E			BCS	PCINC3	GO UPDATE PC
FB11:	ΕA			INITBL	NOP		
FB12:	ΕA				NOD		DUMMY FILL FOR
FB13:			FB		TMP	NBRNCH	XEQ AREA
FB15:						BRANCH	
FB16:	4U	rυ	гА	DUDI	UMP		
FB19:	C1			RTBL	DFB	ŞC1	
FB1A:	D8					\$D8	
FB1B:	D9				DFB	\$D9	
FB1C:	D0				DFB DFB	\$D0	
FB1D:	D3						
FB1E.		70	CO	PREAD		PTRIC	TRIGGER PADDLES
ED11.	7.0	00	00	I ICLI ID	TDV	#\$00	TRIGGER PADDLES INIT COUNT
FDZI:	AU	00			LDI	#\$00	INII COUNI
FD25:	БА				NOP		COMPENSATE FOR 1ST COUNT
FB24:					NOP		
FB25:	BD	64	C0	PREAD2	LDA	PADDL0,X	COUNT Y-REG EVERY 12 USEC
FB28:	10	04			BPL	RTS2D	12 USEC
FB2A:					INY		
FB2B:						PREAD2	EXIT AT 255 MAX
FB2D:					DEY	1 KD/ID2	DATI III 200 IIIM
FBZD:					DEI		
					-		
FB2E:	60				RTS		
FB2E: FB2F:	60 A9	00		INIT	LDA	#\$00	CLR STATUS FOR DEBUG
FB2E: FB2F: FB31:	A9	00		INIT	LDA	#\$00 STATUS	CLR STATUS FOR DEBUG SOFTWARE
FB2F:	A9 85	00 48		INIT	LDA STA	STATUS	SOFTWARE
FB2F: FB31: FB33:	A9 85 AD	00 48 56	CO	INIT	LDA STA	STATUS	SOFTWARE
FB2F: FB31: FB33: FB36:	A9 85 AD AD	00 48 56 54	C0 C0	INIT	LDA STA LDA LDA	STATUS LORES LOWSCR	SOFTWARE INIT VIDEO MODE
FB2F: FB31: FB33: FB36:	A9 85 AD AD	00 48 56 54	C0 C0	INIT	LDA STA LDA LDA	STATUS LORES LOWSCR	SOFTWARE INIT VIDEO MODE SET FOR TEXT MODE
FB2F: FB31: FB33: FB36: FB39: FB3C:	A9 85 AD AD AD A9	00 48 56 54 51 00	C0 C0 C0	INIT SETTXT	LDA STA LDA LDA LDA LDA	STATUS LORES LOWSCR TXTSET #\$00	SOFTWARE INIT VIDEO MODE
FB2F: FB31: FB33: FB36: FB39: FB3C: FB3C:	A9 85 AD AD AD A9 F0	00 48 56 54 51 00 0B	C0 C0 C0	INIT SETTXT	LDA STA LDA LDA LDA LDA BEQ	STATUS LORES LOWSCR TXTSET #\$00 SETWND	SOFTWARE INIT VIDEO MODE SET FOR TEXT MODE FULL SCREEN WINDOW
FB2F: FB31: FB33: FB36: FB39: FB3C: FB3E: FB40:	A9 85 AD AD AD F0 AD	00 48 56 54 51 00 0B 50	C0 C0 C0 C0	INIT SETTXT SETGR	LDA STA LDA LDA LDA LDA BEQ LDA	STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR	SOFTWARE INIT VIDEO MODE SET FOR TEXT MODE FULL SCREEN WINDOW SET FOR GRAPHICS MODE
FB2F: FB31: FB33: FB36: FB39: FB3C: FB3E: FB40:	A9 85 AD AD AD F0 AD	00 48 56 54 51 00 0B 50	C0 C0 C0 C0	INIT SETTXT SETGR	LDA STA LDA LDA LDA LDA BEQ LDA	STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR	SOFTWARE INIT VIDEO MODE SET FOR TEXT MODE FULL SCREEN WINDOW
FB2F: FB31: FB33: FB36: FB39: FB3C: FB3E: FB40:	A9 85 AD AD AD A9 F0 AD AD	00 48 56 51 00 0B 50 53	C0 C0 C0 C0	INIT SETTXT SETGR	LDA STA LDA LDA LDA BEQ LDA LDA	STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR	SOFTWARE INIT VIDEO MODE SET FOR TEXT MODE FULL SCREEN WINDOW SET FOR GRAPHICS MODE
FB2F: FB31: FB33: FB36: FB39: FB3C: FB3E: FB40: FB43:	A9 85 AD AD A9 F0 AD AD 20	00 48 56 51 00 0B 50 53 36	C0 C0 C0 C0	INIT SETTXT SETGR	LDA STA LDA LDA LDA BEQ LDA LDA JSR	STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET	SOFTWARE INIT VIDEO MODE SET FOR TEXT MODE FULL SCREEN WINDOW SET FOR GRAPHICS MODE LOWER 4 LINES AS
FB2F: FB31: FB33: FB36: FB39: FB3C: FB3E: FB40: FB43: FB46: FB49:	A9 85 AD AD AD F0 AD AD 20 A9	00 48 56 51 00 0B 50 53 36 14	C0 C0 C0 C0 C0 F8	INIT SETTXT SETGR	LDA STA LDA LDA LDA BEQ LDA LDA JSR LDA	STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14	SOFTWARE INIT VIDEO MODE SET FOR TEXT MODE FULL SCREEN WINDOW SET FOR GRAPHICS MODE LOWER 4 LINES AS TEXT WINDOW
FB2F: FB31: FB33: FB36: FB39: FB3C: FB3E: FB40: FB43: FB46: FB49: FB4B:	A9 85 AD AD A0 F0 AD 20 A9 85	00 48 56 51 00 0B 50 53 36 14 22	C0 C0 C0 C0 F8	INIT SETTXT SETGR	LDA STA LDA LDA LDA BEQ LDA LDA JSR LDA STA	STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP	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
FB2F: FB31: FB33: FB36: FB39: FB3C: FB3E: FB40: FB43: FB44: FB49: FB4B: FB4D:	A9 85 AD AD A0 F0 AD AD 20 A9 85 A9	00 48 56 54 51 00 0B 50 53 36 14 22 00	C0 C0 C0 C0 F8	INIT SETTXT SETGR	LDA STA LDA LDA LDA BEQ LDA LDA JSR LDA STA LDA	STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$00	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,
FB2F: FB31: FB33: FB36: FB39: FB37: FB32: FB40: FB40: FB440: FB49: FB49: FB49: FB49: FB49: FB45:	 A9 85 AD AD A0 <	00 48 56 54 51 00 0B 50 53 36 14 22 00 20	C0 C0 C0 C0 F8	INIT SETTXT SETGR	LDA STA LDA LDA LDA LDA BEQ LDA LDA STA LDA STA	STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$00 WNDLFT	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
FB2F: FB31: FB33: FB36: FB39: FB3C: FB3E: FB40: FB40: FB46: FB49: FB49: FB4B: FB4D: FB45: FB45:	 A9 85 AD AD A0 <	00 48 56 54 51 00 0B 50 53 36 14 22 00 20 28	C0 C0 C0 C0 F8	INIT SETTXT SETGR	LDA STA LDA LDA LDA BEQ LDA LDA JSR LDA STA LDA STA LDA	STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$00 WNDLFT #\$28	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,
FB2F: FB31: FB33: FB36: FB37: FB36: FB37: FB40: FB47: FB49: FB47: FB45: FB45: FB53:	 A9 85 AD AD AD A0 <	00 48 56 54 51 00 0B 50 53 36 14 22 00 20 28 21	C0 C0 C0 C0 F8	INIT SETTXT SETGR	LDA STA LDA LDA LDA BEQ LDA LDA JSR LDA STA LDA STA LDA STA	STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$14 WNDTOP #\$00 WNDLFT #\$28 WNDWDTH	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,
FB2F: FB31: FB33: FB36: FB39: FB39: FB40: FB40: FB46: FB49: FB449: FB449: FB447: FB47: FB51: FB53:	 A9 85 AD AD AD AD AD A0 <	00 48 56 54 51 00 50 53 36 14 22 00 20 28 21 18	C0 C0 C0 C0 F8	INIT SETTXT SETGR	LDA STA LDA LDA LDA BEQ LDA LDA STA LDA STA LDA STA LDA	STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$14 WNDTOP #\$28 WNDLFT #\$28 WNDWDTH #\$18	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
FB2F: FB31: FB33: FB36: FB37: FB36: FB37: FB40: FB47: FB49: FB47: FB45: FB45: FB53:	 A9 85 AD AD AD AD AD A0 <	00 48 56 54 51 00 50 53 36 14 22 00 20 28 21 18	C0 C0 C0 C0 F8	INIT SETTXT SETGR	LDA STA LDA LDA LDA BEQ LDA LDA STA LDA STA LDA STA LDA	STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$14 WNDTOP #\$00 WNDLFT #\$28 WNDWDTH	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,
FB2F: FB31: FB33: FB36: FB39: FB39: FB40: FB40: FB46: FB49: FB449: FB449: FB447: FB47: FB51: FB53:	 A9 85 AD AD A0 <	00 48 56 54 51 00 53 36 14 22 00 20 20 28 21 18 23	C0 C0 C0 C0 F8	INIT SETTXT SETGR	LDA STA LDA LDA LDA BEQ LDA LDA JSR LDA STA LDA STA LDA STA LDA STA STA	STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$14 WNDTOP #\$28 WNDLFT #\$28 WNDWDTH #\$18	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
FB2F: FB31: FB33: FB36: FB37: FB40: FB47: FB47: FB47: FB47: FB47: FB51: FB55: FB55: FB57: FB59:	 A9 85 AD AD A0 <	00 48 56 54 51 00 08 50 53 36 14 22 00 20 20 20 28 21 18 23 17	C0 C0 C0 C0 F8	INIT SETTXT SETGR	LDA STA LDA LDA LDA LDA BEQ LDA LDA STA LDA STA LDA STA LDA STA LDA	STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$14 WNDTOP #\$00 WNDLFT #\$28 WNDWDTH #\$18 WNDBTM #\$17	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
FB2F: FB31: FB33: FB36: FB39: FB30: FB32: FB40: FB49: FB49: FB49: FB49: FB48: FB48: FB51: FB55: FB55: FB55: FB59:	 A9 85 AD AD A0 <	00 48 56 54 51 00 08 50 53 36 14 22 00 20 20 20 28 21 18 23 17 25	C0 C0 C0 C0 F8	INIT SETTXT SETGR SETWND	LDA STA LDA LDA LDA BEQ LDA LDA JSR LDA STA LDA STA LDA STA LDA STA LDA STA STA	STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$14 WNDLFT #\$28 WNDUFT #\$18 WNDBTM #\$17 CV	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
FB2F: FB31: FB33: FB36: FB37: FB36: FB37: FB40: FB49: FB49: FB49: FB49: FB49: FB49: FB47: FB51: FB55: FB55: FB55: FB55: FB55: FB55: FB55: FB55:	 A9 85 AD AD A0 <	00 48 56 54 51 00 0B 50 53 36 14 22 00 20 20 20 20 20 20 21 18 23 17 25 22	C0 C0 C0 F8 F2	INIT SETTXT SETGR SETWND	LDA STA LDA LDA LDA BEQ LDA JSR LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA JMP	STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$14 WNDDTOP #\$00 WNDLFT #\$28 WNDWDTH #\$18 WNDBTM #\$17 CV VTAB	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
FB2F: FB31: FB33: FB36: FB36: FB39: FB40: FB40: FB46: FB49: FB47: FB47: FB51: FB57: FB57: FB59: FB59: FB50: FB50:	 A9 85 AD AD A0 <	00 48 56 54 51 00 0B 50 53 36 14 22 00 20 20 20 20 20 28 21 18 23 17 25 22 A4	C0 C0 C0 F8 FC FB	INIT SETTXT SETGR SETWND TABV MULPM	LDA STA LDA LDA LDA LDA LDA LDA JSR LDA STA LDA STA LDA STA LDA STA LDA STA JMP JSR	STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$14 WNDTOP #\$14 WNDTOP #\$14 WNDTOP #\$14 WNDLFT #\$28 WNDUFT #\$18 WNDUFT #\$18 WNDDETM #\$17 CV VTAB MD1	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
FB2F: FB31: FB33: FB36: FB36: FB37: FB40: FB43: FB47: FB47: FB47: FB51: FB55: FB57: FB57: FB59: FB59: FB59: FB50: FB40:	 A9 85 AD AD A9 F0 AD 20 85 A9 85 A20 A0 A0	00 48 56 51 00 53 36 20 20 20 28 21 18 23 17 25 22 A4 10	C0 C0 C0 F8 FC FB	INIT SETTXT SETGR SETWND TABV MULPM MUL	LDA STA LDA LDA LDA BEQ LDA LDA JSR LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA	STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$14 WNDTOP #\$14 WNDTOP #\$14 WNDTOP #\$14 WNDTOP #\$14 WNDUFT #\$18 WNDBTM #\$17 CV VTAB MD1 #\$10	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
FB2F: FB31: FB33: FB36: FB36: FB37: FB40: FB47: FB47: FB47: FB47: FB51: FB55: FB55: FB57: FB59: FB59: FB50: FB40: FB50:	 A9 85 AD AD A9 F0 A0 A9 85 A9 85 A9 85 A9 85 A9 85 40 20 A0 A5 	00 48 56 51 00 53 36 20 20 20 28 21 18 23 17 25 22 A4 10	C0 C0 C0 F8 FC FB	INIT SETTXT SETGR SETWND TABV MULPM	LDA STA LDA LDA LDA LDA LDA LDA JDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA	STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$14 WNDTOP #\$14 WNDTOP #\$14 WNDTOP #\$14 WNDTOP #\$14 WNDUFT #\$15 WNDWDTH #\$17 CV VTAB MD1 #\$10 ACL	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
FB2F: FB31: FB33: FB36: FB37: FB37: FB40: FB49: FB49: FB49: FB49: FB49: FB55: FB57:	 A9 85 AD <	00 48 56 51 00 08 50 53 36 14 22 00 20 28 21 18 23 17 25 22 A4 10 50	C0 C0 C0 F8 FC FB	INIT SETTXT SETGR SETWND TABV MULPM MUL	LDA STA LDA LDA LDA EQ LDA LDA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA	STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$14 WNDTOP #\$10 WNDBTM #\$10 ACL A	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
FB2F: FB31: FB33: FB36: FB37: FB37: FB40: FB47: FB47: FB47: FB47: FB57:	 A9 85 AD AD A0 <	00 48 56 51 00 08 50 53 36 14 22 00 20 28 21 18 23 17 25 22 A4 10 50	C0 C0 C0 F8 FC FB	INIT SETTXT SETGR SETWND TABV MULPM MUL	LDA STA LDA LDA LDA EQ LDA LDA JSR LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA	STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$14 WNDTOP #\$10 WNDBTM #\$10 ACL A	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,
FB2F: FB31: FB33: FB36: FB37: FB37: FB40: FB49: FB49: FB49: FB49: FB49: FB55: FB57:	 A9 85 AD AD A0 <	00 48 56 51 00 08 50 53 36 14 22 00 20 28 21 18 23 17 25 22 A4 10 50	C0 C0 C0 F8 FC FB	INIT SETTXT SETGR SETWND TABV MULPM MUL	LDA STA LDA LDA LDA LDA LDA LDA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA	STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$14 WNDTOP #\$10 WNDBTM #\$10 ACL A	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
FB2F: FB31: FB33: FB36: FB37: FB37: FB40: FB47: FB47: FB47: FB47: FB57:	 A9 85 AD AD A0 <	00 48 56 54 51 00 53 36 14 22 00 20 23 21 12 25 22 24 10 50 0C	C0 C0 C0 F8 FC FB	INIT SETTXT SETGR SETWND TABV MULPM MUL	LDA STA LDA LDA LDA LDA LDA LDA JSR LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA	STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$14 WNDTOP #\$10 WNDBTM #\$10 ACL A	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,
FB2F: FB31: FB33: FB36: FB36: FB37: FB40: FB40: FB40: FB44: FB47: FB57:	 A9 85 AD AD AD A0 <	00 48 56 54 51 00 53 36 14 22 00 20 23 21 18 23 17 25 22 24 10 50 70 FE	C0 C0 C0 F8 F8 F8	INIT SETTXT SETGR SETWND TABV MULPM MUL	LDA STA LDA LDA LDA LDA LDA LDA JSR LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA LDA	STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$18 WNDDFTM #\$18 WNDDETM #\$18 WNDDETM #\$11 WNDTOP #\$10 WNDLFT #\$18 WNDDETM #\$11 WNDTOP #\$10 WNDLFT #\$18 WNDDETM #\$11 WNDTOP #\$10 WNDLFT #\$18 WNDDETM #\$11 WNDTOP #\$14 WNDTOP #\$14 WNDTOP #\$14 WNDTOP #\$14 WNDTOP #\$14 WNDTOP #\$14 WNDTOP #\$14 WNDTOP #\$14 WNDTOP #\$14 WNDTOP #\$14 WNDTOP #\$14 WNDTOP #\$14 WNDTOP #\$14 WNDTOP #\$14 WNDTOP #\$14 WNDTOP #\$14 WNDTOP #\$14 WNDTOP #\$14 WNDTOP #\$18 WNDETM #\$18 WNDETM #\$17 CV VTAB MD1 #\$10 ACL A WUL4	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,
FB2F: FB31: FB33: FB36: FB36: FB40: FB47: FB47: FB47: FB47: FB51: FB55: FB57: FB57: FB58: FB57: FB58: FB57: FB58: FB57: FB58: FB57: FB58: FB58: FB58: FB58: FB58: FB67: FB67: FB68: FB68: FB68: FB68: FB68: FB66: FB66:	A9 85 AD AD A0 20 85 A9 85 A9 85 A9 85 A9 85 A9 85 A0 85 A0 85 A0 85 A0 85 A0 85 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0	00 48 56 54 51 00 20 20 20 20 20 20 20 20 20 20 20 20	C0 C0 C0 F8 F8 F8	INIT SETTXT SETGR SETWND TABV MULPM MUL MUL2	LDA STA LDA LDA LDA BEQ LDA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA	STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$14 WNDTOP #\$14 WNDTOP #\$14 WNDDFT #\$28 WNDUFT #\$18 WNDBTM #\$17 CV VTAB MD1 #\$10 ACL A MUL4 #\$FE XTNDL+2,X	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)
FB2F: FB31: FB33: FB36: FB37: FB40: FB47: FB47: FB47: FB47: FB47: FB57:	A9 85 AD AD AD 20 A9 85 A9 85 A9 85 A9 85 40 A0 A5 A9 85 40 A0 A5 A9 85 40 A0 A5 A9 85 75	00 48 56 54 51 00 0B 50 36 14 22 00 20 28 21 18 23 17 25 222 A4 10 50 6 FE 54 56	C0 C0 C0 F8 F8 F8	INIT SETTXT SETGR SETWND TABV MULPM MUL MUL2	LDA STA LDA LDA LDA EQ LDA LDA JDR LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA LDA LDA LDA LDA LDA LDA LDA LDA LD	STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$14 WNDTOP #\$10 WNDUFT #\$18 WNDWDTH #\$18 WNDBTM #\$17 CV VTAB MD1 #\$10 ACL A MUL4 #\$FE XTNDL+2,X AUXL+2,X	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
FB2F: FB31: FB33: FB36: FB39: FB39: FB40: FB49: FB49: FB49: FB49: FB49: FB55: FB57:	A9 85 AD AD AD 20 A9 85 A9 85 A9 85 A9 85 40 A0 A0 A5 A9 85 40 A0 A5 40 A5 A9 85 5 75 5	00 48 56 54 51 00 0B 50 36 14 22 00 20 28 21 18 23 17 25 222 A4 10 50 6 FE 54 56	C0 C0 C0 F8 F8 F8	INIT SETTXT SETGR SETWND TABV MULPM MUL MUL2	LDA STA LDA LDA LDA LDA LDA LDA 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 STA LDA STA STA STA STA STA	STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$14 WNDTOP #\$14 WNDTOP #\$14 WNDDFT #\$28 WNDUFT #\$18 WNDBTM #\$17 CV VTAB MD1 #\$10 ACL A MUL4 #\$FE XTNDL+2,X	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
FB2F: FB31: FB33: FB36: FB36: FB37: FB40: FB40: FB40: FB44: FB47: FB57: FB57: FB57: FB57: FB57: FB57: FB57: FB57: FB57: FB57: FB57: FB67: FB67: FB67: FB67: FB67: FB67: FB77: FB77: FB77:	A9 85 AD AD A0 40 85 85 85 85 40 85 85 85 85 40 85 40 85 40 85 40 85 85 85 85 85 85 85 85 85 85 85 85 85	00 48 56 54 51 00 85 36 14 22 00 20 28 21 82 17 25 22 A4 10 50 7 54 50 7 50 50 50 50 50 28 21 50 50 50 50 50 50 50 50 50 50 50 50 50	C0 C0 C0 F8 F8 F8	INIT SETTXT SETGR SETWND TABV MULPM MUL MUL2	LDA STA LDA LDA LDA LDA LDA LDA JSR LDA STA STA STA STA STA STA STA STA STA ST	STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$14 WNDTOP #\$14 WNDLFT #\$28 WNDWDTH #\$18 WNDBTM #\$18 WNDBTM #\$10 ACL A MUL4 #\$FE XTNDL+2,X XTNDL+2,X	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
FB2F: FB31: FB33: FB36: FB37: FB40: FB47: FB47: FB47: FB47: FB57: FB57: FB57: FB57: FB57: FB57: FB57: FB57: FB57: FB57: FB57: FB67: FB67: FB64: FB67: FB67: FB67: FB67: FB77:	A9 85 AD AD A0 20 85 A9 85 A9 85 A9 85 A9 85 A9 85 40 A5 40 A5 40 A5 40 A5 40 A5 40 A5 40 A5 A5 A5 A5 A5 A5 A5 A5 A5 A5 A5 A5 A5	$\begin{array}{c} 00\\ 48\\ 56\\ 54\\ 51\\ 00\\ 50\\ 53\\ 36\\ 14\\ 22\\ 00\\ 20\\ 23\\ 17\\ 25\\ 224\\ 10\\ 50\\ 0\\ FE\\ 56\\ 54\\ F7\\ \end{array}$	C0 C0 C0 F8 FC FB	INIT SETTXT SETGR SETWND TABV MULPM MUL2 MUL3	LDA STA LDA LDA LDA LDA LDA LDA JDA STA LDA STA STA STA STA STA STA STA STA STA ST	STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$14 WNDTOP #\$14 WNDTOP #\$14 WNDUFT #\$28 WNDUFT #\$18 WNDUFT #\$18 WNDUFT #\$18 WNDBTM #\$17 CV VTAB MD1 #\$10 ACL A MUL4 #\$FE XTNDL+2,X XTNDL+2,X XTNDL+2,X MUL3	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
FB2F: FB31: FB33: FB36: FB37: FB40: FB43: FB49: FB49: FB49: FB47: FB51: FB55: FB57: FB55: FB57: FB59: FB58: FB66: FB67: FB68: FB68: FB68: FB68: FB67: FB68: FB68: FB67: FB68: FB67: FB77:	A9 85 AD AD AD 20 A9 85 A9 85 A9 85 40 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0	$\begin{array}{c} 00\\ 48\\ 56\\ 54\\ 51\\ 00\\ 50\\ 53\\ 36\\ 14\\ 22\\ 00\\ 20\\ 23\\ 17\\ 25\\ 224\\ 10\\ 50\\ 0\\ FE\\ 56\\ 54\\ F7\\ \end{array}$	C0 C0 C0 F8 FC FB	INIT SETTXT SETGR SETWND TABV MULPM MUL MUL2	LDA STA LDA LDA LDA LDA LDA LDA JDA STA LDA STA STA STA STA STA STA STA STA STA ST	STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$14 WNDTOP #\$14 WNDLFT #\$28 WNDWDTH #\$18 WNDBTM #\$18 WNDBTM #\$10 ACL A MUL4 #\$FE XTNDL+2,X XTNDL+2,X	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
FB2F: FB31: FB33: FB36: FB37: FB40: FB47: FB47: FB47: FB47: FB57: FB57: FB57: FB57: FB57: FB57: FB57: FB57: FB57: FB57: FB57: FB67: FB67: FB64: FB67: FB67: FB67: FB67: FB77:	A9 85 AD AD AD 20 A9 85 A9 85 A9 85 40 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0	$\begin{array}{c} 00\\ 48\\ 56\\ 54\\ 51\\ 00\\ 50\\ 53\\ 36\\ 14\\ 22\\ 00\\ 20\\ 23\\ 17\\ 25\\ 224\\ 10\\ 50\\ 0\\ FE\\ 56\\ 54\\ F7\\ \end{array}$	C0 C0 C0 F8 FC FB	INIT SETTXT SETGR SETWND TABV MULPM MUL2 MUL3	LDA STA LDA LDA LDA LDA LDA LDA JDA 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 STA LDA	STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$14 WNDTOP #\$14 WNDTOP #\$14 WNDUFT #\$28 WNDUFT #\$18 WNDUFT #\$18 WNDUFT #\$18 WNDBTM #\$17 CV VTAB MD1 #\$10 ACL A MUL4 #\$FE XTNDL+2,X XTNDL+2,X XTNDL+2,X MUL3	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
FB2F: FB31: FB33: FB36: FB37: FB40: FB43: FB49: FB49: FB49: FB47: FB51: FB55: FB57: FB55: FB57: FB59: FB58: FB66: FB67: FB68: FB68: FB68: FB68: FB67: FB68: FB68: FB67: FB68: FB67: FB77:	A9 85 ADD AD 209 85 A9 85 40 209 85 85 40 200 85 40 200 85 85 200 85 200 85 200 85 85 200 85 200 85 200 85 85 200 80 85 200 80 80 80 80 80 80 80 80 80 80 80 80 8	$\begin{array}{c} 00\\ 48\\ 56\\ 54\\ 51\\ 00\\ 50\\ 53\\ 36\\ 14\\ 22\\ 00\\ 20\\ 23\\ 17\\ 25\\ 224\\ 10\\ 50\\ 0\\ FE\\ 56\\ 54\\ F7\\ \end{array}$	C0 C0 C0 F8 FC FB	INIT SETTXT SETGR SETWND TABV MULPM MUL2 MUL3 MUL4	LDA STA LDA LDA LDA LDA LDA LDA LDA STA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA STA STA STA STA STA STA STA STA ST	STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$14 WNDTOP #\$14 WNDTOP #\$14 WNDTOP #\$14 WNDUFT #\$18 WNDWDTH #\$18 WNDWDTH #\$18 WNDWDTH #\$17 CV VTAB MD1 #\$10 ACL A MUL4 #\$FE XTNDL+2,X XTNDL+2,X XTNDL+2,X	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
FB2F: FB31: FB33: FB36: FB36: FB37: FB40: FB47: FB47: FB47: FB47: FB57: FB57: FB57: FB57: FB57: FB57: FB57: FB57: FB57: FB57: FB57: FB58: FB57: FB66: FB67: FB68: FB67: FB67: FB67: FB67: FB67: FB67: FB67: FB71: FB73: FB74: FB74: FB74: FB74: FB74: FB75: FB79:	A9 85 ADD AD2 9 85 A9 85 A9 85 40 AD2 9 85 A9 85 40 A5 40 A5 40 A5 40 A5 50 50 A5 50 50 50 50 50 50 50 50 50 50 50 50 50	$\begin{array}{c} 00\\ 48\\ 56\\ 54\\ 51\\ 00\\ 50\\ 53\\ 36\\ 14\\ 22\\ 00\\ 20\\ 23\\ 17\\ 25\\ 224\\ 10\\ 50\\ 0\\ FE\\ 56\\ 54\\ F7\\ \end{array}$	C0 C0 C0 F8 FC FB	INIT SETTXT SETGR SETWND TABV MULPM MUL2 MUL3 MUL4	LDA STA LDA LDA LDA LDA LDA LDA LDA STA STA STA LDA STA STA STA STA STA STA STA STA STA ST	STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$14 WNDTOP #\$14 WNDDTP #\$18 WNDWDTH #\$18 WNDWDTH #\$17 CV VTAB MD1 #\$10 ACL A MU14 #\$FE XTNDL+2,X XTNDL+2,X XTNDL+2,X MUL3 #\$03 \$76	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
<pre>FB2F: FB31: FB33: FB36: FB36: FB40: FB40: FB40: FB44: FB41: FB47: FB57: FB57: FB57: FB57: FB57: FB57: FB57: FB58: FB66: FB66: FB67: FB67: FB67: FB67: FB71: FB74: FB74: FB74: FB74: FB74:</pre>	A9 85 ADD ADD A0 85 A9 85 A9 85 A9 85 A9 85 A9 85 A9 85 A9 85 A9 85 A0 ADD ADD ADD ADD ADD ADD ADD ADD ADD	00 48 56 54 50 00 20 20 20 20 20 20 20 20 20 20 20 20	C0 C0 C0 F8 FC FB	INIT SETTXT SETGR SETWND TABV MULPM MUL2 MUL3 MUL4	LDA STA LDA LDA LDA LDA LDA LDA JSR LDA STA DA STA DA STA DA STA DA STA DA STA DA STA DA STA DA STA DA STA DA STA DA STA DA STA DA STA DA STA STA DA STA STA STA DA STA STA STA STA STA STA STA STA STA ST	STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$14 WNDTOP #\$14 WNDLFT #\$28 WNDWDTH #\$18 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	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
<pre>FB2F: FB31: FB33: FB36: FB37: FB40: FB40: FB40: FB47: FB47: FB47: FB51: FB57: FB57: FB57: FB57: FB57: FB57: FB58: FB66: FB67: FB67: FB67: FB68: FB67: FB67: FB67: FB67: FB67: FB67: FB767: FB77:</pre>	A9 85 AD AD A0 F0 AD 20 85 A9 85 A9 85 A9 85 A9 85 A9 85 A9 85 A9 85 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0	00 48 56 54 50 00 20 20 20 20 20 20 20 20 20 20 20 20	C0 C0 C0 F8 FC FB	INIT SETTXT SETGR SETWND TABV MULPM MUL2 MUL3 MUL4	LDA STA LDA LDA LDA LDA LDA LDA LDA STA STA STA STA STA STA STA STA STA ST	STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDTOP #\$14 WNDTOP #\$14 WNDLFT #\$28 WNDWDTH #\$18 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	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
<pre>FB2F: FB31: FB33: FB36: FB36: FB40: FB43: FB40: FB43: FB47: FB47: FB51: FB55: FB57: FB57: FB57: FB59: FB58: FB66: FB66: FB67: FB71: FB71: FB71: FB77: F</pre>	A9 85 ADD AD9 F0 AD0 209 85 985 A95 A95 A95 A02 A05 A02 A05 A05 A05 A05 A05 A05 A05 A05 A05 A05	00 48 56 51 00 00 53 61 22 00 20 20 20 20 20 20 20 20 20 20 20	C0 C0 C0 F8 FC FB	INIT SETTXT SETGR SETWND TABV MULPM MUL2 MUL3 MUL4	LDA STA LDA LDA LDA LDA LDA LDA LDA STA STA LDA STA STA LDA STA LDA STA LDA STA LDA STA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA STA LDA STA LDA STA STA STA STA STA STA STA STA STA ST	STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDDETM #\$18 WNDBETM #\$10 ACL A MUL4 #\$FE XTNDL+2,X XTNDL+2,X XTNDL+2,X XTNDL+2,X XTNDL+2,X WUL3 #\$00 \$50 WNL5	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
<pre>FB2F: FB31: FB33: FB36: FB36: FB40: FB47: FB47: FB47: FB47: FB47: FB51: FB57: FB57: FB57: FB57: FB57: FB57: FB57: FB57: FB67: FB67: FB67: FB67: FB67: FB67: FB71: FB71: FB71: FB71: FB71: FB77: F</pre>	A9 85 ADD AD 209 85 A9 85 40 209 85 85 85 40 209 85 85 85 20 85 85 20 85 85 85 20 85 85 85 20 85 85 85 85 85 85 85 85 85 85 85 85 85	00 48 56 51 00 00 53 61 22 00 20 20 20 20 20 20 20 20 20 20 20	C0 C0 C0 F8 FC FB	INIT SETTXT SETGR SETWND TABV MULPM MUL2 MUL3 MUL4	LDA STA LDA LDA LDA LDA LDA BEQ LDA STA STA STA STA STA STA STA STA STA ST	STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDDETM #\$18 WNDBETM #\$10 ACL A MUL4 #\$FE XTNDL+2,X XTNDL+2,X XTNDL+2,X XTNDL+2,X XTNDL+2,X WUL3 #\$00 \$50 WNL5	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
<pre>FB2F: FB31: FB33: FB36: FB36: FB40: FB43: FB40: FB43: FB47: FB47: FB51: FB55: FB57: FB57: FB57: FB59: FB58: FB66: FB66: FB67: FB71: FB71: FB71: FB77: F</pre>	A9 85 ADD AD 209 85 A9 85 40 209 85 85 85 40 209 85 85 85 20 85 85 20 85 85 85 20 85 85 85 20 85 85 85 85 85 85 85 85 85 85 85 85 85	00 48 56 51 00 00 53 61 22 00 20 20 20 20 20 20 20 20 20 20 20	C0 C0 C0 F8 FC FB	INIT SETTXT SETGR SETWND TABV MULPM MUL2 MUL3 MUL4	LDA STA LDA LDA LDA LDA LDA LDA LDA STA STA LDA STA STA LDA STA LDA STA LDA STA LDA STA STA LDA STA LDA STA LDA STA LDA STA LDA STA LDA STA STA LDA STA LDA STA STA STA STA STA STA STA STA STA ST	STATUS LORES LOWSCR TXTSET #\$00 SETWND TXTCLR MIXSET CLRTOP #\$14 WNDDETM #\$18 WNDBETM #\$10 ACL A MUL4 #\$FE XTNDL+2,X XTNDL+2,X XTNDL+2,X XTNDL+2,X XTNDL+2,X WUL3 #\$00 \$50 WNL5	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

FB81:	20	A4	FB	DIVPM	JSR	MD1	ABS VAL OF AC, AUX. INDEX FOR 16 BITS
FB84: FB86:					ASL		INDEX FOR 16 BITS
FB86: FB88:							
FB88: FB8A:					ROL ROL		XTND/AUX
FB8C:						XTNDH	TO AC.
FB8E:					SEC	ATTADIT	10 110.
FB8F:						XTNDL	
FB91:							MOD TO XTND.
FB93:					TAX		
FB94:	A5	53			LDA	XTNDH	
FB96:					SBC		
FB98:						DIV3	
FB9A:	86	52			STX	XTNDL	
FB9C:	85	53			STA	XTNDH	
FB9E:	E6	50			INC	ACL	
FBA0:	88			DIV3	DEY		
FBA1:	D0	E3			BNE	DIV2	
FBA3:					RTS		
FBA4:				MD1	LDY	#\$00	ABS VAL OF AC, AUX
FBA6:						SIGN	WITH RESULT SIGN
FBA8:						#AUXL	IN LSB OF SIGN.
FBAA:						MD3	
FBAD:				MDD	LDX	#ACL	V OPECIFICA AC OD AVV
FBAF:							X SPECIFIES AC OR AUX
FBB1:						MDRTS	
FBB3:					SEC		
FBB4:		0.0			TYA	TOCO Y	COMDI ODECIEIED DEC
FBB5: FBB7:							COMPL SPECIFIED REG IF NEG.
FBB9:					TYA	LOC0,X	IF NEG.
FBBA:						LOC1,X	
FBBC:						LOC1,X	
FBBE:					INC		
FBC0:				MDRTS	RTS	5101	
FBC1:				BASCALC			CALC BASE ADR IN BASL,H
FBC2:					LSR		FOR GIVEN LINE NO
FBC3:						#\$03	0<=LINE NO.<=\$17
FBC5:							ARG=000ABCDE, GENERATE
FBC7:						BASH	BASH=000001CD
FBC9:	68				PLA		AND
FBCA:						#\$18	BASL=EABAB000
FBCC:	90	02			BCC	BSCLC2	
FBCE:	69	7F			ADC	#\$7F	
FBD0:	85	28		BSCLC2	STA	BASL	
FBD2:	0A				ASL		
FBD3:	0A				ASL		
	05	28			ORA	BASL	
FBD4:	0.0	20					
FBD4: FBD6:					STA	BASL	
FBD6: FBD8:	85 60	28			RTS		
FBD6: FBD8:	85 60	28			RTS		BELL CHAR? (CNTRL-G)
FBD6: FBD8: FBD9: FBDB:	85 60 C9 D0	28 87 12		BELL1	RTS CMP BNE	#\$87 RTS2B	NO, RETURN
FBD6: FBD8: FBD9: FBDB: FBDD:	85 60 C9 D0 A9	28 87 12 40		BELL1	RTS CMP BNE LDA	#\$87 RTS2B #\$40	BELL CHAR? (CNTRL-G) NO, RETURN DELAY .01 SECONDS
FBD6: FBD8: FBD9: FBDB: FBDD: FBDF:	85 60 C9 D0 A9 20	28 87 12 40 A8	FC	BELL1	RTS CMP BNE LDA JSR	#\$87 RTS2B #\$40 WAIT	
FBD6: FBD8: FBD9: FBDB: FBDD: FBDF: FBE2:	85 60 C9 D0 A9 20 A0	28 87 12 40 A8 C0	FC	BELL1	RTS CMP BNE LDA JSR LDY	#\$87 RTS2B #\$40 WAIT #\$C0	DELAY .01 SECONDS
FBD6: FBD8: FBD9: FBDB: FBDD: FBDF: FBE2: FBE4:	85 60 C9 D0 A9 20 A0 A9	28 87 12 40 A8 C0 0C	FC	BELL1	RTS CMP BNE LDA JSR LDY LDA	#\$87 RTS2B #\$40 WAIT #\$C0 #\$0C	DELAY .01 SECONDS TOGGLE SPEAKER AT
FBD6: FBD9: FBD9: FBDD: FBDF: FBE2: FBE4: FBE6:	85 60 C9 D0 A9 20 A0 A9 20	28 87 12 40 A8 C0 0C A8	FC FC	BELL1 BELL2	RTS CMP BNE LDA JSR LDY LDA JSR	#\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT	DELAY .01 SECONDS
FBD6: FBD8: FBD9: FBDB: FBDD: FBDF: FBE2: FBE4: FBE6: FBE9:	85 60 C9 D0 A9 20 A0 A9 20 AD	28 87 12 40 A8 C0 0C A8	FC FC	BELL1 BELL2	RTS CMP BNE LDA JSR LDY LDA JSR LDA	#\$87 RTS2B #\$40 WAIT #\$C0 #\$0C	DELAY .01 SECONDS TOGGLE SPEAKER AT
FBD6 : FBD8 : FBD9 : FBDB : FBD7 : FBD7 : FBE2 : FBE4 : FBE6 : FBE9 : FBEC :	85 60 C9 D0 A9 20 A0 A9 20 AD 88	28 87 12 40 A8 C0 0C A8 30	FC FC C0	BELL1 BELL2	RTS CMP BNE LDA JSR LDY LDA JSR LDA DEY	#\$87 RTS2B #\$40 WAIT #\$CO #\$OC WAIT SPKR	DELAY .01 SECONDS TOGGLE SPEAKER AT
FBD6 : FBD8 : FBD9 : FBD7 : FBD7 : FBE2 : FBE4 : FBE6 : FBE9 : FBEC : FBED :	85 60 C9 D0 A9 20 A0 A9 20 AD 88 D0	28 87 12 40 A8 C0 0C A8 30 F5	FC FC C0	BELL1 BELL2	RTS CMP BNE LDA JSR LDY LDA JSR LDA DEY BNE	#\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR BELL2	DELAY .01 SECONDS TOGGLE SPEAKER AT 1 KHZ FOR .1 SEC.
FBD6 : FBD8 : FBD9 : FBD7 : FBD7 : FBE2 : FBE4 : FBE9 : FBE2 : FBE2 : FBE2 : FBE5 :	85 60 C9 D0 A9 20 A0 A0 A0 AD 88 D0 60	28 87 12 40 A8 C0 0C A8 30 F5	FC FC C0	BELL1 BELL2	RTS CMP BNE LDA JSR LDY LDA JSR LDA DEY BNE	#\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR BELL2	DELAY .01 SECONDS TOGGLE SPEAKER AT 1 KHZ FOR .1 SEC.
FBD6 : FBD8 : FBD9 : FBD7 : FBD7 : FBE2 : FBE4 : FBE6 : FBE9 : FBEC : FBED :	85 60 C9 D0 A9 20 A0 A0 A0 A0 A0 60 A4	28 87 12 40 A8 C0 0C A8 30 F5 24	FC FC C0	BELL1 BELL2	RTS CMP BNE LDA JSR LDY LDA JSR LDA DEY BNE RTS LDY	#\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR BELL2 CH	DELAY .01 SECONDS TOGGLE SPEAKER AT 1 KHZ FOR .1 SEC. CURSOR H INDEX TO Y-REG
FBD6 : FBD8 : FBD9 : FBD7 : FBD7 : FBD7 : FBE2 : FBE2 : FBE4 : FBE6 : FBE7 : FBE7 : FBE7 :	85 60 C9 D0 A9 20 A0 A0 A0 A0 88 D0 60 A4 91	28 87 12 40 A8 C0 0C A8 30 F5 24 28	FC FC C0	BELL1 BELL2 RTS2B STOADV	RTS CMP BNE LDA JSR LDY LDA JSR LDA DEY BNE RTS LDY STA	#\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR BELL2 CH (BASL),Y	DELAY .01 SECONDS TOGGLE SPEAKER AT 1 KHZ FOR .1 SEC.
FBD6 : FBD8 : FBD9 : FBD7 : FBD7 : FBD7 : FBE2 : FBE2 : FBE4 : FBE6 : FBE7 : FBE7 : FBE7 :	85 60 C9 D0 A9 20 A0 A0 A0 88 D0 60 A4 91 E6	28 87 12 40 00 00 A8 30 F5 24 28 24	FC FC C0	BELL1 BELL2	RTS CMP BNE LDA JSR LDY LDA JSR LDA DEY BNE RTS LDY STA INC	#\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR BELL2 CH (BASL),Y	DELAY .01 SECONDS TOGGLE SPEAKER AT 1 KHZ FOR .1 SEC. CURSOR H INDEX TO Y-REG STORE CHAR IN LINE
FBD6: FBD8: FBD9: FBD5: FBD5: FBD7: FBD7: FBE2: FBE4: FBE6: FBE7: FBE7: FBE7: FBE7: FBE7: FBE7:	85 60 C9 D0 A9 20 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0	28 87 12 40 00 00 A8 30 F5 24 28 24 24 24	FC FC C0	BELL1 BELL2 RTS2B STOADV	RTS CMP BNE LDA JSR LDY LDA JSR LDA DEY BNE RTS LDY STA INC LDA	#\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR BELL2 CH (BASL),Y CH CH	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)
FBD6 : FBD9 : FBD9 : FBD5 : FBD7 : FBD7 : FBE4 : FBE4 : FBE6 : FBE7 : FBE7 : FBF7 : FBF6 :	85 60 C9 D0 A9 20 A0 A0 A0 A0 A0 A0 60 A1 60 A4 91 E6 A5 C5	28 87 12 40 A8 C0 0C A8 30 F5 24 28 24 24 24 21	FC FC C0	BELL1 BELL2 RTS2B STOADV	RTS CMP BNE LDA JSR LDY LDA JSR LDA DEY BNE RTS LDY STA INC LDA	#\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR BELL2 CH (BASL),Y CH CH CH WNDWDTH	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
FBD6 : FBD9 : FBD9 : FBD9 : FBD9 : FBD9 : FBD7 : FBE4 : FBE4 : FBE6 : FBE7 : FBE7 : FBF6 : FBF6 : FBF8 : FBF6 : FBF6 : FBF6 : FBF6 : FBF6 : FBF6 :	85 60 C9 D0 A9 20 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0	28 87 12 40 A8 C0 0C A8 30 F5 24 28 24 24 24 21 66	FC FC C0	BELL1 BELL2 RTS2B STOADV ADVANCE RTS3	RTS CMP BNE LDA JSR LDY LDA JSR LDA DEY BNE RTS LDY STA INC LDY CMP BCS RTS	#\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR BELL2 CH (BASL),Y CH CH CH WNDWDTH CR	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) BEYOND WINDOW WIDTH?
FBD6 : FBD9 : FBD9 : FBD9 : FBD9 : FBD9 : FBD7 : FBE4 : FBE4 : FBE6 : FBE7 : FBE7 : FBF6 : FBF6 : FBF8 : FBF6 : FBF6 : FBF6 : FBF6 : FBF6 : FBF6 :	85 60 C9 D0 A9 20 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0	28 87 12 40 A8 C0 0C A8 30 F5 24 28 24 24 24 21 66	FC FC C0	BELL1 BELL2 RTS2B STOADV ADVANCE RTS3	RTS CMP BNE LDA JSR LDY LDA JSR LDA DEY BNE RTS LDY STA INC LDY CMP BCS RTS	#\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR BELL2 CH (BASL),Y CH CH CH WNDWDTH CR	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) BEYOND WINDOW WIDTH? YES CR TO NEXT LINE
FBD6 : FBD9 : FBD9 : FBD9 : FBD9 : FBD9 : FBD7 : FBE4 : FBE4 : FBE6 : FBE7 : FBE7 : FBF6 : FBF6 : FBF8 : FBF6 : FBF6 : FBF6 : FBF6 : FBF6 : FBF6 :	85 60 C9 D0 A9 20 A0 A0 88 D0 60 A4 91 E6 A5 C5 B0 60 C9	28 87 12 40 A8 C0 0C A8 30 F5 24 24 24 24 21 66 A0	FC CO	BELL1 BELL2 RTS2B STOADV ADVANCE	RTS CMP BNE LDA JSR LDY LDA JSR LDA DEY BNE RTS LDY STA INC LDA CMP BCS RTS CMP	#\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR BELL2 CH (BASL),Y CH CH CH WNDWDTH CR	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) BEYOND WINDOW WIDTH? YES CR TO NEXT LINE NO, RETURN
FBD6 : FBD8 : FBD9 : FBD9 : FBD9 : FBD9 : FBD7 : FBE4 : FBE4 : FBE6 : FBE7 : FBE7 : FBF6 : FBF6 : FBF7 : FBF7 : FBF7 : FBFF : FBFF : FBFF : FBFF :	85 60 C9 D0 A9 20 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0	28 87 12 40 A8 C0 OC A8 30 F5 24 28 24 24 21 66 A0 EF	FC CO	BELL1 BELL2 RTS2B STOADV ADVANCE RTS3	RTS CMP BNE LDA JSR LDY LDA JSR LDY BNE RTS LDA INC LDA CMP BCS CMP BCS TAY	#\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR BELL2 CH (BASL),Y CH CH CH WNDWDTH CR #\$A0 STOADV	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) BEYOND WINDOW WIDTH? YES CR TO NEXT LINE NO, RETURN CONTROL CHAR? NO, OUTPUT IT. INVERSE VIDEO?
FBD6 : FBD8 : FBD9 : FBD8 : FBD7 : FBD7 : FBE4 : FBE6 : FBE6 : FBE7 : FBE7 : FBF6 : FBF7 : FBF7 : FBF7 : FBF7 : FBF7 : FBF7 : FBF7 : FBF7 : FBF7 : FC01 : FC02 :	85 60 C9 D0 A9 20 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0	28 87 12 40 A8 C0 0C A8 30 F5 24 28 24 24 24 21 66 A0 EF EC	FC CO	BELL2 BELL2 RTS2B STOADV ADVANCE RTS3 VIDOUT	RTS CMP BNE LDA JSR LDY LDA JSR LDY BNE RTS LDY STA CMP BCS RTS CMP BCS TAY BPL	#\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR BELL2 CH (BASL),Y CH CH CH WNDWDTH CR #\$A0 STOADV STOADV	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) BEYOND WINDOW WIDTH? YES CR TO NEXT LINE NO, RETURN CONTROL CHAR? NO, OUTPUT IT. INVERSE VIDEO? YES, OUTPUT IT.
FBD6 : FBD8 : FBD9 : FBD9 : FBD9 : FBD9 : FBD0 : FBE2 : FBE4 : FBE6 : FBE7 : FBE7 : FBF6 : FBF7 : FBF7 : FBF7 : FBF7 : FC01 : FC02 : FC04 :	85 60 C9 D0 A9 20 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0	28 87 12 40 A8 C0 0C A8 30 F5 24 28 24 24 24 24 21 66 A0 EF EC 8D	FC CO	BELL1 BELL2 RTS2B STOADV ADVANCE RTS3 VIDOUT	RTS CMP BNE LDA JSR LDY LDA JSR LDY BNE RTS LDY STA LDY STA LDA CMP BCS RTS CMP BCS TAY BPL CMP	#\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR BELL2 CH (BASL),Y CH (BASL),Y CH WNDWDTH CR #\$A0 STOADV #\$8D	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) BEYOND WINDOW WIDTH? YES CR TO NEXT LINE NO, RETURN CONTROL CHAR? NO, OUTPUT IT. INVERSE VIDEO? YES, OUTPUT IT. CR?
FBD6 : FBD8 : FBD9 : FBD7 : FBD7 : FBD7 : FBD7 : FBE4 : FBE4 : FBE6 : FBE7 : FBF4 : FBF6 : FBF7 : FBF7 : FBF7 : FBF7 : FBF7 : FC01 : FC01 : FC04 : FC06 :	85 60 C9 D0 A9 20 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0	28 87 12 40 A8 C0 0C A8 30 F5 24 24 24 24 24 24 26 66 A0 EF EC 8D 5A	FC CO	BELL1 BELL2 RTS2B STOADV ADVANCE RTS3 VIDOUT	RTS CMP BNE LDA JSR LDY LDA JSR LDY BNE RTS LDY STA INC LDA CMP BCS RTS CMP BCS TAY BPL CMP BEQ	#\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR BELL2 CH (BASL),Y CH CH CH CH CH CH CH CH CH STOADV #\$8D CR	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) BEYOND WINDOW WIDTH? YES CR TO NEXT LINE NO,RETURN CONTROL CHAR? NO,OUTPUT IT. INVERSE VIDEO? YES, OUTPUT IT. CR? YES.
FBD6 : FBD8 : FBD9 : FBD9 : FBD9 : FBD9 : FBD9 : FBD7 : FBE4 : FBE4 : FBE4 : FBE7 : FBE7 : FBF4 : FBF4 : FBF4 : FBF4 : FBF5 : FBF5 : FBF7 : FBF7 : FBF7 : FC01 : FC04 : FC08 :	85 60 C9 D0 A9 20 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0	28 87 12 40 A8 C0 0C A8 30 F5 24 24 21 66 A0 EF EC 8D 5A 8A	FC CO	BELL1 BELL2 RTS2B STOADV ADVANCE RTS3 VIDOUT	RTS CMP BNE LDA JSR LDY LDA DEY BNE RTS LDY STA INC LDA CMP BCS RTS CMP BCS TAY BPL CMP BEQ CMP	#\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR BELL2 CH (BASL),Y CH CH CH CH CH CH CH CH STOADV #\$80 CR #\$80 CR #\$88	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) BEYOND WINDOW WIDTH? YES CR TO NEXT LINE NO,RETURN CONTROL CHAR? NO,OUTPUT IT. INVERSE VIDEO? YES, OUTPUT IT. CR? YES. LINE FEED?
FBD6 : FBD8 : FBD9 : FBD9 : FBD9 : FBD9 : FBD9 : FBD7 : FBE4 : FBE6 : FBE7 : FBE7 : FBF4 : FBF6 : FBF7 : FBF7 : FBF7 : FBF7 : FBF7 : FC01 : FC04 : FC08 : FC08 :	85 60 C9 D0 A9 20 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0	28 87 12 40 A8 C0 0C A8 30 F5 24 28 24 24 24 24 24 26 66 A0 EF EC 8D 5A 8A 5A	FC CO	BELL2 BELL2 RTS2B STOADV ADVANCE RTS3 VIDOUT	RTS CMP BNE LDA JSR LDY LDA JSR LDY BNE RTS LDA INC LDA CMP BCS RTS CMP BCS CMP BCS BPL CMP BCP BEQ	#\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR BELL2 CH (BASL),Y CH CH CH WNDWDTH CR #\$A0 STOADV #\$8D CR #\$8A LF	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) BEYOND WINDOW WIDTH? YES CR TO NEXT LINE NO, RETURN CONTROL CHAR? NO, OUTPUT IT. INVERSE VIDEO? YES, OUTPUT IT. CR? YES. LINE FEED? IF SO, DO IT.
FBD6 : FBD8 : FBD9 : FBD9 : FBD9 : FBD9 : FBD9 : FBD9 : FBD7 : FBE4 : FBE6 : FBE7 : FBE7 : FBF4 : FBF7 : FBF7 : FBF7 : FBF7 : FC01 : FC02 : FC04 : FC08 : FC08 :	85 60 C9 D0 A9 20 A0 20 A0 20 A0 88 D0 60 44 91 E65 C5 B0 60 C9 B0 A8 10 9 F0 C9 F0 C9	28 87 12 40 A8 C0 0C A8 30 F5 24 28 24 24 21 66 A0 EF EC 8D 5A 88 88	FC	BELL2 BELL2 RTS2B STOADV ADVANCE RTS3 VIDOUT	RTS CMP BNE LDA JSR LDY LDA JSR LDY BNE RTS LDY STA LDA CMP BCS RTS CMP BCS CMP BCS CMP BCS CMP BEQ CMP	#\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR BELL2 CH (BASL),Y CH CH CH WNDWDTH CR #\$A0 STOADV #\$8D CR #\$8A LF #\$88	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) BEYOND WINDOW WIDTH? YES CR TO NEXT LINE NO, RETURN CONTROL CHAR? NO, OUTPUT IT. INVERSE VIDEO? YES. LINE FEED? IF SO, DO IT. BACK SPACE? (CNTRL-H)
FBD6 : FBD8 : FBD9 : FBD8 : FBD7 : FBD7 : FBE4 : FBE6 : FBE7 : FBE7 : FBF6 : FBF7 : FBF6 : FBF7 : FBF7 : FBF7 : FBF7 : FC01 : FC02 : FC03 : FC03 : FC03 : FC03 : FC04 :	85 60 C9 D0 A9 20 A0 20 A0 20 A0 88 D0 60 44 91 E65 C5 B0 60 C9 B0 A8 10 C9 E0 C9 D0 C9 D0 C9 D0 C9 D0 A0 20 A0 C9 A0 A0 C9 C0 A0 C9 C0 A0 C9 C0 A0 C9 C0 A0 C0 C9 C0 C0 C0 C0 C0 C0 C0 C0 C0 C0 C0 C0 C0	28 87 12 40 A8 COC A8 30 F5 24 24 24 24 24 21 66 A0 EF EC 8D 5A 88 C9	FC CO	BELL2 BELL2 RTS2B STOADV ADVANCE RTS3 VIDOUT	RTS CMP BNE LDA JSR LDY LDA JSR LDY BNE RTS LDY STA LDA CMP BCS RTS CMP BCS RTS BPL CMP BEQ CMP BEQ CMP BNE	#\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR BELL2 CH (BASL),Y CH (BASL),Y CH WNDWDTH CR #\$A0 STOADV #\$A0 STOADV #\$8D CR #\$8A LF #\$88 BELL1	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) BEYOND WINDOW WIDTH? YES CR TO NEXT LINE NO, RETURN CONTROL CHAR? NO, OUTPUT IT. INVERSE VIDEO? YES, OUTPUT IT. CR? YES. LINE FEED? IF SO, DO IT. BACK SPACE? (CNTRL-H) NO, CHECK FOR BELL.
FBD6 : FBD8 : FBD9 : FBD9 : FBD9 : FBD9 : FBD1 : FBD2 : FBE4 : FBE4 : FBE6 : FBE7 : FBE7 : FBF4 : FBF6 : FBF7 : FBF7 : FBF7 : FC01 : FC01 : FC03 : FC04 : FC04 : FC04 : FC04 : FC04 : FC04 : FC04 : FC04 : FC04 : FC05 : FC	85 60 C9 D0 A9 20 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0	28 87 12 40 A8 C0 C A8 30 F5 24 24 24 21 66 A0 EF EC 8D 5A 88 C9 24	FC CO	BELL2 BELL2 RTS2B STOADV ADVANCE RTS3 VIDOUT	RTS CMP BNE LDA JSR LDY LDA JSR LDY BDE RTS LDY STA LDA CMP BCS TAY BPL CMP BCS CMP BEQ CMP BEQ CMP BEQ CMP BEQ CMP	#\$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	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) BEYOND WINDOW WIDTH? YES CR TO NEXT LINE NO,RETURN CONTROL CHAR? NO,OUTPUT IT. INVERSE VIDEO? YES, OUTPUT IT. CR? YES. LINE FEED? IF SO, DO IT. BACK SPACE? (CNTRL-H) NO, CHECK FOR BELL. DECREMENT CURSOR H INDEX
FBD6 : FBD8 : FBD9 : FBD9 : FBD9 : FBD9 : FBD9 : FBD9 : FBD7 : FBE4 : FBE4 : FBE7 : FBE7 : FBF4 : FBF4 : FBF4 : FBF7 : FBF7 : FBF7 : FC01 : FC02 : FC03 : FC03 : FC03 : FC03 : FC04 : FC05 : FC5	85 60 C9 D0 A9 20 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0	28 87 12 40 87 60 00 83 00 83 00 83 24 24 24 24 24 24 24 24 24 24 24 24 80 5A 80 5A 80 5A 80 5A 80 5A 80 5A 80 80 80 80 80 80 80 80 80 80 80 80 80	FC CO	BELL2 BELL2 RTS2B STOADV ADVANCE RTS3 VIDOUT	RTS CMP BNE LDA JSR LDY LDA JSR LDY BDE RTS LDY STA STA CMP BCS TAY BCS CMP BCS CMP BEQ CMP BEQ CMP BEQ CMP BEQ CMP BEQ CMP	#\$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	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) BEYOND WINDOW WIDTH? YES CR TO NEXT LINE NO, RETURN CONTROL CHAR? NO, OUTPUT IT. INVERSE VIDEO? YES. LINE FEED? IF SO, DO IT. BACK SPACE? (CNTRL-H) NO, CHECK FOR BELL. DECREMENT CURSOR H INDEX IF POS, OK. ELSE MOVE UP
FBD6: FBD8: FBD9: FBD5: FBD7: FBD7: FBE4: FBE4: FBE6: FBE7: FBE7: FBF7: FBF7: FBF7: FBF7: FBF7: FC01: FC04: FC04: FC08: FC08: FC04:	85 60 C9 D0 A9 20 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0	28 87 12 40 A8 30 F5 24 24 24 24 24 24 24 24 24 24 24 24 24	FC C0	BELL2 BELL2 RTS2B STOADV ADVANCE RTS3 VIDOUT	RTS CMP BNE LDA JSR LDY LDA JSR LDY BNE RTS LDY STA LDA CMP BCS TAY BPL CMP BCS CMP BCS CMP BEQ CMP BEQ CMP BEQ CMP EDE LDA	#\$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	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) BEYOND WINDOW WIDTH? YES CR TO NEXT LINE NO,RETURN CONTROL CHAR? NO,OUTPUT IT. INVERSE VIDEO? YES, OUTPUT IT. CR? YES. LINE FEED? IF SO, DO IT. BACK SPACE? (CNTRL-H) NO, CHECK FOR BELL. DECREMENT CURSOR H INDEX
FBD6: FBD8: FBD9: FBD9: FBD7: FBD7: FBE4: FBE4: FBE6: FBE7: FBE7: FBF6: FBF7: FBF7: FBF7: FBF7: FC01: FC02: FC04:	85 60 C9 D0 A9 20 A0 A0 20 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0	28 87 12 40 87 60 830 75 24 28 24 21 66 80 87 80 87 80 80 80 80 80 80 80 80 80 80 80 80 80	FC C0	BELL2 BELL2 RTS2B STOADV ADVANCE RTS3 VIDOUT BS	RTS CMP BNE LDA JSR LDY LDA JSR LDY BNE RTS LDA INC LDA CMP BCS TAY BPL CMP BCS TAY BPL CMP BCS CMP BCS CMP BCS TAY BPL CMP BCS STA	#\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR BELL2 CH (BASL),Y CH CH CH WNDWDTH CR #\$A0 STOADV #\$8D CR #\$80 STOADV #\$8D CR #\$88 BELL1 CH RTS3 WNDWDTH CH	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) BEYOND WINDOW WIDTH? YES CR TO NEXT LINE NO, RETURN CONTROL CHAR? NO, OUTPUT IT. INVERSE VIDEO? YES. LINE FEED? IF SO, DO IT. BACK SPACE? (CNTRL-H) NO, CHECK FOR BELL. DECREMENT CURSOR H INDEX IF POS, OK. ELSE MOVE UP SET CH TO WNDWDTH-1
FBD6: FBD8: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBD9: FBE4: FBE6: FBE7: FBE7: FBF4: FBF6: FBF7: FBF7: FC01: FC04: FC06: FC08: FC08: FC12: FC14: FC16: FC18:	85 60 C9 D0 A9 20 A0 A0 20 A0 A0 60 A1 20 A0 A0 60 A1 60 C9 B0 A0 C9 B0 A0 C9 C9 C9 C0 C9 C0 C9 C0 C9 C0 C0 C0 C0 C0 C0 C0 C0 C0 C0 C0 C0 C0	28 87 12 40 A8 C0 0C A8 30 F5 24 24 22 4 21 66 A0 FF EC 8D 5A 88 C9 24 E8 24 22 4 22 4 22 4 22 4 22 4 22 4	FC CO	BELL2 BELL2 RTS2B STOADV ADVANCE RTS3 VIDOUT BS	RTS CMP BNE LDA JSR LDY LDA JSR LDY BNE RTS LDY STA LDA INC LDA CMP BCS RTS CMP BCS CMP BCS CMP BPL CMP BPL CMP BEQ CMP BDE CMP BDE CMP BDE CMP BDE CMP CMP CMP CMP CMP CMP CMP CMP CMP CMP	#\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR BELL2 CH (BASL),Y CH (CH WNDWDTH CR #\$A0 STOADV #\$8D CR #\$A0 STOADV #\$8D CR #\$8A LF #\$88 BELL1 CH RTS3 WNDWDTH CH CH	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) BEYOND WINDOW WIDTH? YES CR TO NEXT LINE NO, RETURN CONTROL CHAR? NO, OUTPUT IT. INVERSE VIDEO? YES, OUTPUT IT. CR? YES. LINE FEED? IF SO, DO IT. BACK SPACE? (CNTRL-H) NO, CHECK FOR BELL. DECREMENT CURSOR H INDEX IF POS, OK. ELSE MOVE UP SET CH TO WNDWDTH-1 (RIGHTMOST SCREEN POS)
FBD6: FBD8: FBD9: FBD9: FBD7: FBD7: FBE4: FBE4: FBE6: FBE7: FBE7: FBF6: FBF7: FBF7: FBF7: FBF7: FC01: FC02: FC04:	85 60 C9 D0 A9 20 A0 A0 20 A0 40 60 40 50 60 40 50 60 C9 B0 80 C9 B0 80 C9 B0 80 C9 B0 80 C9 C9 C9 A9 20 A0 A9 20 A0 A9 20 A0 A9 20 A0 A9 20 A0 A9 20 A0 A9 20 A0 A9 20 A0 A9 20 A0 A9 20 A0 A9 20 A0 A9 20 A0 A9 20 A0 A9 20 A0 A9 20 A0 A9 20 A0 A0 A9 20 A0 A0 A9 20 A0 A9 20 A0 A9 20 A0 A9 20 A0 A9 20 A0 A9 20 A0 A9 20 A0 A9 20 A0 A9 20 A0 A9 20 A0 A9 20 A0 A9 20 A0 A9 20 A0 A9 20 A0 A9 20 A0 A9 20 A0 A9 20 A0 A9 20 A0 A0 20 A0 A0 20 A0 A0 20 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0	28 87 12 40 87 20 67 87 24 28 24 21 66 80 57 80 57 80 57 80 57 80 57 80 57 80 57 80 57 80 57 80 57 80 57 80 57 80 57 80 80 80 80 80 80 80 80 80 80 80 80 80	FC CO	BELL2 BELL2 RTS2B STOADV ADVANCE RTS3 VIDOUT BS	RTS CMP BNE LDA JSR LDY LDA JSR LDY BNE RTS LDY STA LDA INC LDA CMP BCS RTS CMP BCS CMP BCS CMP BPL CMP BPL CMP BEQ CMP BDE CMP BDE CMP BDE CMP BDE CMP CMP CMP CMP CMP CMP CMP CMP CMP CMP	#\$87 RTS2B #\$40 WAIT #\$C0 #\$0C WAIT SPKR BELL2 CH (BASL),Y CH (BASL),Y CH WNDWDTH CR #\$A0 STOADV #\$80 CR #\$80 CR #\$88 ELL1 CH RTS3 WNDWDTH CH CH WNDTOP	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) BEYOND WINDOW WIDTH? YES CR TO NEXT LINE NO, RETURN CONTROL CHAR? NO, OUTPUT IT. INVERSE VIDEO? YES. LINE FEED? IF SO, DO IT. BACK SPACE? (CNTRL-H) NO, CHECK FOR BELL. DECREMENT CURSOR H INDEX IF POS, OK. ELSE MOVE UP SET CH TO WNDWDTH-1

FC1E: FC20: FC22:							
						RTS4	IF TOP LINE THEN RETURN
FC22:					DEC		DEC CURSOR V-INDEX
				VTAB			GET CURSOR V-INDEX
			FB	VTABZ	JSR	BASCALC	GENERATE BASE ADR ADD WINDOW LEFT INDEX
FC27: FC29:						BASL	TO BASL
FC29: FC2B:				RTS4	RTS	DASL	IO BASL
FC2B:						#\$C0	ESC?
FC2E:						HOME	IF SO, DO HOME AND CLEAR
FC30:					ADC	#\$FD	ESC-A OR B CHECK
FC32:					BCC	ADVANCE	A, ADVANCE
FC34:	FO	DA			BEQ	BS	B, BACKSPACE
FC36:	69	FD			ADC	#\$FD	ESC-C OR D CHECK
FC38:	90	2C			BCC	LF	C, DOWN
FC3A:	FO	DE			BEQ	UP	D, GO UP
FC3C:							ESC-E OR F CHECK
FC3E:						CLREOL	E, CLEAR TO END OF LINE
FC40:						RTS4	NOT F, RETURN
FC42:				CLREOP			CURSOR H TO Y INDEX
FC44:					LDA	CV	CURSOR V TO A-REGISTER
FC46:				CLEOP1	TOD	VTABZ	SAVE CURRENT LINE ON STK
FC47: FC4A:						CLEOLZ	CALC BASE ADDRESS CLEAR TO EOL, SET CARRY
FC4A: FC4D:			гC			#\$00	CLEAR FROM H INDEX=0 FOR REST
FC4D: FC4F:					PLA	#\$00	INCREMENT CURRENT LINE
FC50:						#\$00	(CARRY IS SET)
FC52:							DONE TO BOTTOM OF WINDOW?
FC54:						CLEOP1	NO, KEEP CLEARING LINES
FC56:						VTAB	YES, TAB TO CURRENT LINE
FC58:				HOME			INIT CURSOR V
FC5A:	85	25			STA	CV	AND H-INDICES
FC5C:	A0	00			LDY	#\$00	
FC5E:	84	24			STY	CH	THEN CLEAR TO END OF PAGE
FC60:	FO	E4			BEQ	CLEOP1	
FC62:				CR			CURSOR TO LEFT OF INDEX
FC64:						CH	(RET CURSOR H=0)
FC66:				LF	INC		INCR CURSOR V(DOWN 1 LINE)
FC68:						CV	
FC6A:						WNDBTM	
FC6C:						VTABZ	NO, SET BASE ADDR
FC6E:				CCDOLI	DEC		DECR CURSOR V (BACK TO BOTTOM) START AT TOP OF SCRL WNDW
FC70: FC72:				SCRULL	PHA	WNDIOP	START AT TOP OF SCRL WNDW
FC72:			FC			VTABZ	GENERATE BASE ADR
FC76:				SCRL1		BASL	COPY BASL,H
FC78:				Donali		BAS2L	TO BAS2L,H
FC7A:							
FC7C:					LDA	BASH	
	85	2B			LDA STA	BASH BAS2H	
FC7E:					STA		INIT Y TO RIGHTMOST INDEX
FC7E: FC80:	A4				STA	BAS2H	INIT Y TO RIGHTMOST INDEX OF SCROLLING WINDOW
	A4 88				STA LDY DEY PLA	BAS2H WNDWDTH	
FC80: FC81: FC82:	A4 88 68 69	21 01			STA LDY DEY PLA ADC	BAS2H WNDWDTH #\$01	OF SCROLLING WINDOW
FC80: FC81: FC82: FC84:	A4 88 68 69 C5	21 01 23			STA LDY DEY PLA ADC CMP	BAS2H WNDWDTH #\$01 WNDBTM	OF SCROLLING WINDOW INCR LINE NUMBER DONE?
FC80: FC81: FC82: FC84: FC86:	A4 88 68 69 C5 B0	21 01 23 0D			STA LDY DEY PLA ADC CMP BCS	BAS2H WNDWDTH #\$01	OF SCROLLING WINDOW
FC80: FC81: FC82: FC84: FC86: FC88:	A4 88 69 C5 B0 48	21 01 23 0D			STA LDY DEY PLA ADC CMP BCS PHA	BAS2H WNDWDTH #\$01 WNDBTM SCRL3	OF SCROLLING WINDOW INCR LINE NUMBER DONE? YES, FINISH
FC80: FC81: FC82: FC84: FC86: FC88: FC89:	A4 88 69 C5 B0 48 20	21 01 23 0D 24	FC		STA LDY DEY PLA ADC CMP BCS PHA JSR	BAS2H WNDWDTH #\$01 WNDBTM SCRL3 VTABZ	OF SCROLLING WINDOW INCR LINE NUMBER DONE? YES, FINISH FORM BASL,H (BASE ADDR)
FC80: FC81: FC82: FC84: FC86: FC88: FC89: FC82:	A4 88 69 C5 B0 48 20 B1	21 01 23 0D 24 28	FC	SCRL2	STA LDY DEY PLA ADC CMP BCS PHA JSR LDA	BAS2H WNDWDTH #\$01 WNDBTM SCRL3 VTABZ (BASL),Y	OF SCROLLING WINDOW INCR LINE NUMBER DONE? YES, FINISH FORM BASL,H (BASE ADDR) MOVE A CHR UP ON LINE
FC80: FC81: FC82: FC84: FC86: FC88: FC89: FC8C: FC8E:	A4 88 69 C5 B0 48 20 B1 91	21 01 23 0D 24 28 2A	FC	SCRL2	STA LDY DEY PLA ADC CMP BCS PHA JSR LDA STA	BAS2H WNDWDTH #\$01 WNDBTM SCRL3 VTABZ (BASL),Y (BAS2L),Y	OF SCROLLING WINDOW INCR LINE NUMBER DONE? YES, FINISH FORM BASL,H (BASE ADDR) MOVE A CHR UP ON LINE
FC80: FC81: FC82: FC84: FC86: FC88: FC89: FC8C: FC8E: FC90:	A4 88 69 C5 B0 48 20 B1 91 88	21 01 23 0D 24 28 2A	FC	SCRL2	STA LDY DEY PLA ADC CMP BCS PHA JSR LDA STA DEY	BAS2H WNDWDTH #\$01 WNDBTM SCRL3 VTABZ (BASL),Y (BAS2L),Y	OF SCROLLING WINDOW INCR LINE NUMBER DONE? YES, FINISH FORM BASL,H (BASE ADDR) MOVE A CHR UP ON LINE
FC80: FC81: FC82: FC84: FC86: FC88: FC89: FC8C: FC8E: FC90: FC91:	A4 88 69 C5 B0 48 20 B1 91 88 10	21 01 23 0D 24 28 2A F9	FC	SCRL2	STA LDY DEY PLA ADC CMP BCS PHA JSR LDA STA DEY BPL	BAS2H WNDWDTH #\$01 WNDBTM SCRL3 VTABZ (BASL),Y (BAS2L),Y (BAS2L),Y	OF SCROLLING WINDOW INCR LINE NUMBER DONE? YES, FINISH FORM BASL,H (BASE ADDR) MOVE A CHR UP ON LINE NEXT CHAR OF LINE
FC80: FC81: FC82: FC84: FC86: FC88: FC89: FC8C: FC8E: FC90: FC91: FC93:	A4 88 69 C5 B0 48 20 B1 91 88 10 30	21 01 23 0D 24 28 2A F9 E1	FC	SCRL2	STA LDY DEY PLA ADC CMP BCS PHA JSR LDA STA DEY BPL BMI	BAS2H WNDWDTH #\$01 WNDBTM SCRL3 VTABZ (BASL),Y (BAS2L),Y (BAS2L),Y SCRL2 SCRL1	OF SCROLLING WINDOW INCR LINE NUMBER DONE? YES, FINISH FORM BASL,H (BASE ADDR) MOVE A CHR UP ON LINE NEXT CHAR OF LINE NEXT LINE (ALWAYS TAKEN)
FC80: FC81: FC82: FC84: FC86: FC88: FC89: FC8C: FC8E: FC90: FC91: FC93:	A4 88 69 C5 B0 48 20 B1 91 88 10 30 A0	21 01 23 0D 24 28 2A F9 E1 00	FC	SCRL2 SCRL3	STA LDY DEY PLA ADC CMP BCS PHA JSR LDA STA DEY BPL BMI LDY	BAS2H WNDWDTH #\$01 WNDBTM SCRL3 VTABZ (BASL),Y (BAS2L),Y (BAS2L),Y SCRL2 SCRL1 #\$00	OF SCROLLING WINDOW INCR LINE NUMBER DONE? YES, FINISH FORM BASL,H (BASE ADDR) MOVE A CHR UP ON LINE NEXT CHAR OF LINE
FC80: FC81: FC82: FC84: FC86: FC88: FC82: FC82: FC90: FC91: FC91: FC95: FC97: FC94:	A4 88 69 C5 B0 48 20 B1 91 88 10 30 A0 20 B0	21 01 23 0D 24 28 2A F9 E1 00 9E 86	FC	SCRL2 SCRL3	STA LDY DEY PLA ADC CMP BCS PHA JSR LDA STA DEY BPL BMI LDY JSR BCS	BAS2H WNDWDTH #\$01 WNDBTM SCRL3 VTABZ (BASL),Y (BAS2L),Y (BAS2L),Y SCRL2 SCRL1 #\$00 CLEOLZ VTAB	OF SCROLLING WINDOW INCR LINE NUMBER DONE? YES, FINISH FORM BASL,H (BASE ADDR) MOVE A CHR UP ON LINE NEXT CHAR OF LINE NEXT LINE (ALWAYS TAKEN) CLEAR BOTTOM LINE GET BASE ADDR FOR BOTTOM LINE CARRY IS SET
FC80: FC81: FC82: FC84: FC86: FC88: FC82: FC82: FC90: FC91: FC91: FC95: FC97: FC94:	A4 88 69 C5 B0 48 20 B1 91 88 10 30 A0 20 B0	21 01 23 0D 24 28 2A F9 E1 00 9E 86	FC	SCRL2 SCRL3	STA LDY DEY PLA ADC CMP BCS PHA JSR LDA STA DEY BPL BMI LDY JSR BCS	BAS2H WNDWDTH #\$01 WNDBTM SCRL3 VTABZ (BASL),Y (BAS2L),Y (BAS2L),Y SCRL2 SCRL1 #\$00 CLEOLZ VTAB	OF SCROLLING WINDOW INCR LINE NUMBER DONE? YES, FINISH FORM BASL,H (BASE ADDR) MOVE A CHR UP ON LINE NEXT CHAR OF LINE NEXT LINE (ALWAYS TAKEN) CLEAR BOTTOM LINE GET BASE ADDR FOR BOTTOM LINE CARRY IS SET
FC80: FC81: FC82: FC84: FC86: FC86: FC85: FC85: FC90: FC91: FC91: FC95: FC97: FC92: FC92:	A4 88 69 C5 B0 48 20 B1 91 88 10 30 20 B0 A0 A0 A0 A0	21 01 23 0D 24 28 2A F9 E1 00 9E 86 24 A0	FC	SCRL2 SCRL3 CLREOL CLEOLZ	STA LDY DEY PLA ADC CMP BCS PHA JSR LDA STA DEY BPL BMI LDY JSR BCS LDY LDA	BAS2H WNDWDTH #\$01 WNDBTM SCRL3 VTABZ (BASL),Y (BAS2L),Y (BAS2L),Y (BAS2L),Y SCRL2 SCRL1 #\$00 CLEOLZ VTAB CH #\$A0	OF SCROLLING WINDOW INCR LINE NUMBER DONE? YES, FINISH FORM BASL,H (BASE ADDR) MOVE A CHR UP ON LINE NEXT CHAR OF LINE NEXT CHAR OF LINE CLEAR BOTTOM LINE GET BASE ADDR FOR BOTTOM LINE CARRY IS SET CURSOR H INDEX
FC80: FC81: FC82: FC82: FC86: FC88: FC89: FC80: FC90: FC91: FC91: FC97: FC97: FC92: FC92: FC92: FC94:	A4 88 69 C5 B0 48 20 B1 30 88 10 30 A0 20 B0 A4 A9 91	21 01 23 0D 24 28 2A F9 E1 00 9E 24 A0 28	FC	SCRL2 SCRL3 CLREOL CLEOLZ	STA LDY DEY PLA ADC CMP BCS PHA JSR LDA STA DEY BPL BMI LDY JSR BCS LDY LDA	BAS2H WNDWDTH #\$01 WNDBTM SCRL3 VTABZ (BASL),Y (BAS2L),Y (BAS2L),Y (BAS2L),Y SCRL2 SCRL1 #\$00 CLEOLZ VTAB CH #\$A0	OF SCROLLING WINDOW INCR LINE NUMBER DONE? YES, FINISH FORM BASL,H (BASE ADDR) MOVE A CHR UP ON LINE NEXT CHAR OF LINE NEXT LINE (ALWAYS TAKEN) CLEAR BOTTOM LINE GET BASE ADDR FOR BOTTOM LINE CARRY IS SET
FC80: FC81: FC82: FC84: FC86: FC86: FC85: FC85: FC90: FC91: FC91: FC95: FC97: FC92: FC92:	A4 88 69 C5 B0 48 20 B1 30 88 10 30 A0 20 B0 A4 A9 91	21 01 23 0D 24 28 2A F9 E1 00 9E 24 A0 28	FC	SCRL2 SCRL3 CLREOL CLEOLZ CLEOL2	STA LDY DEY PLA ADC CMP BCS PHA JSR LDA STA DEY BPL LDY LDY LDY LDY STA INY	BAS2H WNDWDTH #\$01 WNDBTM SCRL3 VTABZ (BASL),Y (BAS2L),Y (BAS2L),Y SCRL2 SCRL1 #\$00 CLEOLZ VTAB CH #\$40 (BASL),Y	OF SCROLLING WINDOW INCR LINE NUMBER DONE? YES, FINISH FORM BASL,H (BASE ADDR) MOVE A CHR UP ON LINE NEXT CHAR OF LINE NEXT CHAR OF LINE CLEAR BOTTOM LINE GET BASE ADDR FOR BOTTOM LINE CARRY IS SET CURSOR H INDEX
FC80: FC81: FC82: FC84: FC86: FC89: FC85: FC85: FC90: FC91: FC93: FC97: FC94: FC95: FC92: FC92: FC92: FC93:	A4 88 69 C5 B0 48 20 B1 91 88 10 30 A0 20 B0 A4 A9 91 C8 C4	21 01 23 0D 24 28 2A F9 E1 00 9E 24 A0 28 28 21	FC	SCRL2 SCRL3 CLREOL CLEOLZ CLEOL2	STA LDY DEY PLA ADC CMP BCS PHA JSR LDA STA DEY BPL LDY LDY LDY LDY STA INY	BAS2H WNDWDTH #\$01 WNDBTM SCRL3 VTABZ (BASL),Y (BAS2L),Y (BAS2L),Y SCRL2 SCRL1 #\$00 CLEOLZ VTAB CH #\$40 (BASL),Y	OF SCROLLING WINDOW INCR LINE NUMBER DONE? YES, FINISH FORM BASL,H (BASE ADDR) MOVE A CHR UP ON LINE NEXT CHAR OF LINE NEXT CHAR OF LINE NEXT LINE (ALWAYS TAKEN) CLEAR BOTTOM LINE GET BASE ADDR FOR BOTTOM LINE CARRY IS SET CURSOR H INDEX
FC80: FC81: FC82: FC82: FC82: FC85: FC85: FC90: FC91: FC91: FC91: FC93: FC95: FC97: FC94: FC92: FC92: FC92: FC92: FC92: FC93: FC92: FC93:	A4 88 69 C5 B0 48 20 B1 91 88 10 30 A0 20 B0 A0 20 80 A0 20 C8 C8 C3 20 C9 20 C9 20 20 20 20 20 20 20 20 20 20 20 20 20	21 01 23 0D 24 28 2A F9 E1 00 9E 24 A0 28 24 28 21 F9	FC	SCRL2 SCRL3 CLREOL CLEOLZ CLEOL2	STA LDY DEY PLA ADC CMP BCS PHA JSR LDA STA DEY BPL BML LDY JSR BCS LDY LDA STA INY CPY BCC	BAS2H WNDWDTH #\$01 WNDBTM SCRL3 VTABZ (BASL),Y (BAS2L),Y (BAS2L),Y (BAS2L),Y SCRL2 SCRL1 #\$00 CLEOLZ VTAB CH #\$A0	OF SCROLLING WINDOW INCR LINE NUMBER DONE? YES, FINISH FORM BASL,H (BASE ADDR) MOVE A CHR UP ON LINE NEXT CHAR OF LINE NEXT CHAR OF LINE NEXT LINE (ALWAYS TAKEN) CLEAR BOTTOM LINE GET BASE ADDR FOR BOTTOM LINE CARRY IS SET CURSOR H INDEX
FC80: FC81: FC82: FC82: FC82: FC82: FC82: FC82: FC90: FC91: FC91: FC92: FC92: FC92: FC92: FC92: FC92: FC92: FC92: FC92: FC93:	A4 88 69 C5 B0 48 20 B1 30 20 B1 30 20 B0 A4 A9 91 C8 C4 90 60	21 01 23 0D 24 28 2A F9 E1 00 9E 86 24 A0 28 21 F9	FC	SCRL2 SCRL3 CLREOL CLEOLZ CLEOL2	STA LDY DEY PLA ADC CMP BCS PHA JSR BCS LDA STA BPL BMI LDY JSR BCS LDY LDA STA INY CPY BCC RTS	BAS2H WNDWDTH #\$01 WNDBTM SCRL3 VTABZ (BASL),Y (BAS2L),Y (BAS2L),Y SCRL2 SCRL1 #\$00 CLEOLZ VTAB CH #\$40 (BASL),Y	OF SCROLLING WINDOW INCR LINE NUMBER DONE? YES, FINISH FORM BASL,H (BASE ADDR) MOVE A CHR UP ON LINE NEXT CHAR OF LINE NEXT CHAR OF LINE NEXT LINE (ALWAYS TAKEN) CLEAR BOTTOM LINE GET BASE ADDR FOR BOTTOM LINE CARRY IS SET CURSOR H INDEX
FC80: FC81: FC82: FC82: FC82: FC82: FC82: FC82: FC90: FC91: FC91: FC91: FC92: FC92: FC92: FC92: FC92: FC92: FC92: FC92: FC93:	A4 88 69 C5 B0 48 20 B1 30 20 B1 30 20 B0 A4 A9 91 C8 C4 90 60 38	21 01 23 0D 24 28 2A F9 E1 00 9E 86 24 A0 28 21 F9	FC	SCRL2 SCRL3 CLREOL CLEOLZ CLEOL2 WAIT	STA LDY DEY PLA ADC CMP BCS PHA JSR BCY BPL BMI LDY LDA STA BCS LDY LDA STA INY CPY BCC RTS SEC	BAS2H WNDWDTH #\$01 WNDBTM SCRL3 VTABZ (BASL),Y (BAS2L),Y (BAS2L),Y SCRL2 SCRL1 #\$00 CLEOLZ VTAB CH #\$40 (BASL),Y	OF SCROLLING WINDOW INCR LINE NUMBER DONE? YES, FINISH FORM BASL,H (BASE ADDR) MOVE A CHR UP ON LINE NEXT CHAR OF LINE NEXT CHAR OF LINE NEXT LINE (ALWAYS TAKEN) CLEAR BOTTOM LINE GET BASE ADDR FOR BOTTOM LINE CARRY IS SET CURSOR H INDEX
FC80: FC81: FC82: FC82: FC82: FC82: FC82: FC82: FC90: FC91: FC91: FC92: FC92: FC92: FC92: FC92: FC93:	A4 88 69 C5 B0 48 20 B1 91 88 80 20 B0 A0 20 B0 A4 A9 91 C8 C4 90 60 38 48	21 01 23 0D 24 28 2A F9 E1 00 9E 86 24 A0 28 21 F9	FC	SCRL2 SCRL3 CLREOL CLEOLZ CLEOL2 WAIT WAIT2	STA LDY DEY PLA ADC CMP BCS PHA STA DEY BDL LDA STA LDY LDA STA LDY LDY LDA STA INY CPY BCS STA INY CPY BCS SEC PHA	BAS2H WNDWDTH #\$01 WNDBTM SCRL3 VTABZ (BASL),Y (BAS2L),Y (BAS2L),Y SCRL2 SCRL1 #\$00 CLEOLZ VTAB CH #\$40 (BASL),Y WNDWDTH CLEOL2	OF SCROLLING WINDOW INCR LINE NUMBER DONE? YES, FINISH FORM BASL,H (BASE ADDR) MOVE A CHR UP ON LINE NEXT CHAR OF LINE NEXT CHAR OF LINE NEXT LINE (ALWAYS TAKEN) CLEAR BOTTOM LINE GET BASE ADDR FOR BOTTOM LINE CARRY IS SET CURSOR H INDEX
FC80: FC81: FC82: FC82: FC82: FC82: FC82: FC82: FC90: FC91: FC91: FC92: FC92: FC92: FC92: FC92: FC93:	A4 88 69 C5 B0 48 20 B1 91 88 80 20 B0 A0 20 B0 A4 A9 91 C8 C4 90 60 38 48	21 01 23 0D 24 28 2A F9 E1 00 9E 86 24 A0 28 21 F9	FC	SCRL2 SCRL3 CLREOL CLEOLZ CLEOL2 WAIT WAIT2	STA LDY DEY PLA ADC CMP BCS PHA STA DEY BDL LDA STA LDY LDA STA LDY LDY LDA STA INY CPY BCS STA INY CPY BCS SEC PHA	BAS2H WNDWDTH #\$01 WNDBTM SCRL3 VTABZ (BASL),Y (BAS2L),Y (BAS2L),Y SCRL2 SCRL1 #\$00 CLEOLZ VTAB CH #\$40 (BASL),Y WNDWDTH CLEOL2	OF SCROLLING WINDOW INCR LINE NUMBER DONE? YES, FINISH FORM BASL,H (BASE ADDR) MOVE A CHR UP ON LINE NEXT CHAR OF LINE NEXT CHAR OF LINE NEXT LINE (ALWAYS TAKEN) CLEAR BOTTOM LINE GET BASE ADDR FOR BOTTOM LINE CARRY IS SET CURSOR H INDEX STORE BLANKS FROM 'HERE' TO END OF LINES (WNDWDTH)
FC80: FC81: FC82: FC84: FC86: FC85: FC85: FC85: FC90: FC91: FC91: FC92: FC92: FC92: FC92: FC40: FCA2: FCA3: FCA5: FCA3: FCA3:	A4 88 69 C5 B0 48 20 B1 91 88 10 30 A0 20 B0 40 A9 91 C8 C4 90 60 38 48 E9 D0	21 01 23 0D 24 28 2A F9 E1 00 9E 86 2A 28 22 F9 28 21 F9 01 F0	FC	SCRL2 SCRL3 CLREOL CLEOLZ CLEOL2 WAIT WAIT2	STA LDY DEY PLA ADC CMP BCS PHA JSR LDA STA DEY BPL BML LDY JSR BCS LDY LDA STA INY CPY BCC RTS SEC PHA SBC BNE	BAS2H WNDWDTH #\$01 WNDBTM SCRL3 VTABZ (BASL),Y (BAS2L),Y (BAS2L),Y SCRL2 SCRL1 #\$00 CLEOLZ VTAB CH #\$00 (BASL),Y WNDWDTH CLEOL2 #\$01	OF SCROLLING WINDOW INCR LINE NUMBER DONE? YES, FINISH FORM BASL,H (BASE ADDR) MOVE A CHR UP ON LINE NEXT CHAR OF LINE NEXT CHAR OF LINE CLEAR BOTTOM LINE GET BASE ADDR FOR BOTTOM LINE CARRY IS SET CURSOR H INDEX STORE BLANKS FROM 'HERE' TO END OF LINES (WNDWDTH)
FC80: FC81: FC82: FC82: FC82: FC82: FC85: FC90: FC90: FC91: FC91: FC92: FC92: FC92: FC92: FC92: FC42: FCA3: FCA3: FCA3: FCA3: FCA3: FCA3: FCA3: FCA3: FCA3:	A4 88 69 C5 B0 48 20 B1 30 A0 20 B0 A4 91 C8 C4 90 C8 C4 90 60 88 E9 D0 68	21 01 23 0D 24 28 2A F9 E1 00 9E 86 24 A0 28 21 F9 01 FC	FC	SCRL2 SCRL3 CLREOL CLEOLZ CLEOL2 CLEOL2 WAIT WAIT2 WAIT3	STA LDY DEY PLA ADC CMP BCS PHA JSR BDL BMI LDY BPL BMI LDY LDA STA DSR BCS LDY LDA STA STA STA STA BCS LDY LDY BCS STA STA BCS STA STA BCS BCS STA BCS STA DEY BCS STA DEY BPLA STA STA DEY BPLA STA DEY BCS STA DEY BPLA STA DEY BPLA STA DEY BPLA STA STA DEY BPLA STA DEY BPLA STA DEY BPLA STA DEY BPLA STA DEY BPLA STA DEY BPLA STA DEY BPLA STA DEY BPLA STA DEY BPLA STA DEY BPLA STA DEY BPLA STA DEY BPLA STA DEY BPLA STA DEY BPLA STA DEY BPLA STA DEY BPLA STA DEY BPLA STA DEY BDA STA DEY BDA STA DEY BDA STA DEY BDA STA DEY BCS STA DEY BPLA STA DEY BCS STA DEY BCS STA DEY BCS STA DEY BCS STA BCS STA STA STA STA STA STA STA STA STA ST	BAS2H WNDWDTH #\$01 WNDBTM SCRL3 VTABZ (BASL),Y (BAS2L),Y (BAS2L),Y SCRL2 SCRL1 #\$00 CLEOLZ VTAB CH #\$A0 (BASL),Y WNDWDTH CLEOL2 WNDWDTH CLEOL2	OF SCROLLING WINDOW INCR LINE NUMBER DONE? YES, FINISH FORM BASL,H (BASE ADDR) MOVE A CHR UP ON LINE NEXT CHAR OF LINE NEXT CHAR OF LINE NEXT LINE (ALWAYS TAKEN) CLEAR BOTTOM LINE GET BASE ADDR FOR BOTTOM LINE CARRY IS SET CURSOR H INDEX STORE BLANKS FROM 'HERE' TO END OF LINES (WNDWDTH)
FC80: FC81: FC82: FC82: FC82: FC82: FC82: FC82: FC90: FC91: FC91: FC91: FC91: FC92: FC92: FC92: FC92: FC92: FC22: FC23:	A4 88 69 C5 B0 48 20 B1 91 80 20 B1 30 20 B0 A0 20 B0 A4 91 C8 C4 90 60 88 E9 D0 68 E9	21 01 23 0D 24 28 2A F9 E1 00 9E 86 24 A0 28 21 F9 01 FC 01	FC	SCRL2 SCRL3 CLREOL CLEOLZ CLEOL2 CLEOL2 WAIT WAIT2 WAIT3	STA LDY DEY PLA ADC CMP BCS PHA JSR BDL BMI LDY JSR BCS LDY LDA STA LDY LDA STA CPY BCC RTS SEC PHA SBC BNE PLA SBC	BAS2H WNDWDTH #\$01 WNDBTM SCRL3 VTABZ (BASL),Y (BAS2L),Y (BAS2L),Y SCRL2 SCRL1 #\$00 CLEOLZ VTAB CH #\$40 (BASL),Y WNDWDTH CLEOL2 #\$01 WAIT3 #\$01	OF SCROLLING WINDOW INCR LINE NUMBER DONE? YES, FINISH FORM BASL,H (BASE ADDR) MOVE A CHR UP ON LINE NEXT CHAR OF LINE NEXT CHAR OF LINE CLEAR BOTTOM LINE GET BASE ADDR FOR BOTTOM LINE CARRY IS SET CURSOR H INDEX STORE BLANKS FROM 'HERE' TO END OF LINES (WNDWDTH)
FC80: FC81: FC82: FC82: FC82: FC85: FC85: FC90: FC90: FC91: FC97: FC97: FC97: FC97: FC97: FC92: FC92: FC92: FC42: FC43:	A4 88 69 C5 B0 48 20 B1 30 20 B0 48 91 88 10 30 20 B0 40 91 C8 40 90 60 38 48 91 C8 20 D0 60 20 D1 20 2 2 D1 20 D1 20 D1 20 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	21 01 23 0D 24 28 2A F9 E1 00 9E 86 24 20 28 21 F9 01 F0 F0 F0	FC	SCRL2 SCRL3 CLREOL CLEOLZ CLEOL2 CLEOL2 WAIT WAIT2 WAIT3	STA LDY DEY PLA ADC CMP BCS PHA JSR LDA STA DEY BPL JSR LDY JSR BMI LDY LDA STA LDY LDA STA INY CPY BCC RTS SEC PHA SBC BNE BNE	BAS2H WNDWDTH #\$01 WNDBTM SCRL3 VTABZ (BASL),Y (BAS2L),Y (BAS2L),Y SCRL2 SCRL1 #\$00 CLEOLZ VTAB CH #\$A0 (BASL),Y WNDWDTH CLEOL2 WNDWDTH CLEOL2	OF SCROLLING WINDOW INCR LINE NUMBER DONE? YES, FINISH FORM BASL,H (BASE ADDR) MOVE A CHR UP ON LINE NEXT CHAR OF LINE NEXT CHAR OF LINE CLEAR BOTTOM LINE GET BASE ADDR FOR BOTTOM LINE CARRY IS SET CURSOR H INDEX STORE BLANKS FROM 'HERE' TO END OF LINES (WNDWDTH)
FC80: FC81: FC82: FC82: FC82: FC86: FC85: FC90: FC91: FC91: FC97: FC97: FC92: FC92: FC92: FC42: FC43:	A4 88 69 C5 B0 48 20 B1 88 10 30 A0 20 B0 48 91 88 20 B0 49 20 B0 48 91 C8 48 90 60 38 848 90 C8 60 60 80 60 91 80 80 80 80 80 80 80 80 80 80 80 80 80	21 01 23 0D 24 28 2A F9 E1 00 9E 86 24 A0 28 21 F9 01 FC 01 F6	FC	SCRL2 SCRL3 CLREOL CLEOLZ CLEOL2 CLEOL2 WAIT WAIT2 WAIT3	STA LDY DEY PLA ADC CMP BCS PHA STA DEY BPL BMI LDA STA DEY BPL LDY JSR BPL LDY LDY STA STA CPY BCC RTS SEC PHA SBC BNE BNE RTS	BAS2H WNDWDTH #\$01 WNDBTM SCRL3 VTABZ (BASL),Y (BAS2L),Y (BAS2L),Y SCRL2 SCRL1 #\$00 (CLEOLZ VTAB CH #\$00 (BASL),Y WNDWDTH CLEOL2 #\$01 WAIT3 #\$01 WAIT2	OF SCROLLING WINDOW INCR LINE NUMBER DONE? YES, FINISH FORM BASL,H (BASE ADDR) MOVE A CHR UP ON LINE NEXT CHAR OF LINE NEXT CHAR OF LINE CLEAR BOTTOM LINE GET BASE ADDR FOR BOTTOM LINE CARRY IS SET CURSOR H INDEX STORE BLANKS FROM 'HERE' TO END OF LINES (WNDWDTH) 1.0204 USEC (13+27/2*A+5/2*A*A)
FC80: FC81: FC82: FC82: FC82: FC86: FC85: FC90: FC91: FC91: FC97: FC97: FC92: FC92: FC92: FC42: FC43:	A4 88 69 C5 B0 81 91 88 10 30 20 B1 40 20 B1 20 B1 20 B0 A4 91 C8 90 38 82 C9 0 60 38 82 C5 C5 B1 91 80 C5 C5 C5 C5 C5 C5 C5 C5 C5 C5 C5 C5 C5	21 01 23 0D 24 28 2A F9 E1 00 9E 86 24 A0 28 21 F9 01 FC 01 F6 42	FC	SCRL2 SCRL3 CLREOL CLEOLZ CLEOL2 CLEOL2 WAIT2 WAIT2 WAIT3	STA LDY DEY PLA ADC CMP BCS PHA LDA STA DEY BPL BMI LDY JSR BPL LDY JSR ECS LDY LDA STA INY CPY BCC RTS SEC PHA SBC BNE BNE BNE RTS INC	BAS2H WNDWDTH #\$01 WNDBTM SCRL3 VTABZ (BASL),Y (BAS2L),Y (BAS2L),Y SCRL2 SCRL1 #\$00 (CLEOLZ VTAB CH #\$40 (BASL),Y WNDWDTH CLEOL2 #\$01 WAIT3 #\$01 WAIT2	OF SCROLLING WINDOW INCR LINE NUMBER DONE? YES, FINISH FORM BASL,H (BASE ADDR) MOVE A CHR UP ON LINE NEXT CHAR OF LINE NEXT CHAR OF LINE CLEAR BOTTOM LINE GET BASE ADDR FOR BOTTOM LINE CARRY IS SET CURSOR H INDEX STORE BLANKS FROM 'HERE' TO END OF LINES (WNDWDTH)
FC80: FC81: FC82: FC84: FC82: FC82: FC82: FC82: FC90: FC91: FC91: FC92: FC92: FC92: FC92: FC92: FC92: FC92: FC92: FC92: FC92: FC93: FC94: FC43: FC43: FC44:	A4 88 69 C5 B0 81 91 88 10 30 20 A4 91 C8 91 C8 91 C8 91 C8 91 C8 91 C8 91 C8 20 C5 D0 60 50 C5 D0 C5 D0 C5 D1 C5 C5 C5 C5 C5 C5 C5 C5 C5 C5 C5 C5 C5	21 01 23 0D 24 28 2A F9 E1 00 28 24 A0 28 21 F9 01 FC 01 FC 01 26 42 02	FC	SCRL2 SCRL3 CLREOL CLEOL2 CLEOL2 CLEOL2 WAIT WAIT2 WAIT3	STA LDY DEY PLA ADC CMP BCS PHA JSR STA DEY BPL BMI LDY JSR BCS LDY LDA STA DEY BPL BCS LDY LDA STA SCC RTS SEC PHA SEC RTS SEC PHA SEC RTS SEC PHA INY CPY BDL BNE RTS SEC PHA INY CPY BCS SEC PHA INY CPY BCS SC SEC PHA INY CPY BCS SC SC SC SC SC SC SC SC SC SC SC SC S	BAS2H WNDWDTH #\$01 WNDBTM SCRL3 VTABZ (BASL),Y (BAS2L),Y (BAS2L),Y VTAB CLEOLZ VTAB CH #\$00 (BASL),Y WNDWDTH CLEOL2 #\$01 WAIT3 #\$01 WAIT3 #\$01 WAIT3 A4L NXTA1 A4H	OF SCROLLING WINDOW INCR LINE NUMBER DONE? YES, FINISH FORM BASL,H (BASE ADDR) MOVE A CHR UP ON LINE NEXT CHAR OF LINE NEXT CHAR OF LINE NEXT LINE (ALWAYS TAKEN) CLEAR BOTTOM LINE GET BASE ADDR FOR BOTTOM LINE CARRY IS SET CURSOR H INDEX STORE BLANKS FROM 'HERE' TO END OF LINES (WNDWDTH) 1.0204 USEC (13+27/2*A+5/2*A*A) INCR 2-BYTE A4 AND A1
FC80: FC81: FC82: FC82: FC82: FC82: FC82: FC82: FC91: FC91: FC91: FC91: FC92: FC92: FC92: FC92: FC92: FC92: FC42: FC42: FC43:	A4 88 69 C5 B0 48 20 B1 30 A0 20 BA4 A9 S1 88 80 C8 20 BA4 A9 C8 C8 C9 0 60 38 88 C9 D C8 C9 C8 C9 C8 C9 C9 C9 C9 C9 C9 C9 C9 C9 C9 C9 C9 C9	21 01 23 0D 24 28 2A F9 E1 00 9E 624 A0 28 21 F9 01 FC 01 F6 42 23 3C	FC	SCRL2 SCRL3 CLREOL CLEOL2 CLEOL2 CLEOL2 WAIT WAIT2 WAIT3 NXTA4 NXTA1	STA LDY DEY PLA ADC CMP BCS PHA STA DEY BPL STA DEY BMI LDY JSR BMI LDY LDA STA INY CPY BCC RTS SEC PHA SEC PHA SEC PHA SEC BNE ENE SEC BNE ENE SEC BNE ENE SEC BNE ENE SEC BNE ENE SEC BNE ENE SEC BNE ENE SEC BNE ENE SEC BNE ENE SEC BNE ENE SEC BNE ENE SEC BNE ENE SEC BNE SEC PHA SEC SEC PHA SEC SEC PHA SEC SEC PHA SEC SEC PHA STA STA STA STA STA STA STA STA STA ST	BAS2H WNDWDTH #\$01 WNDBTM SCRL3 VTABZ (BASL),Y (BAS2L),Y (BAS2L),Y SCRL2 SCRL1 #\$00 CLEOLZ VTAB CH #\$A0 (BASL),Y WNDWDTH CLEOL2 #\$01 WAIT3 #\$01 WAIT3 #\$01 WAIT2 A4L NXTA1 A4H A1L	OF SCROLLING WINDOW INCR LINE NUMBER DONE? YES, FINISH FORM BASL,H (BASE ADDR) MOVE A CHR UP ON LINE NEXT CHAR OF LINE NEXT CHAR OF LINE MEXT LINE (ALWAYS TAKEN) CLEAR BOTTOM LINE GET BASE ADDR FOR BOTTOM LINE CARRY IS SET CURSOR H INDEX STORE BLANKS FROM 'HERE' TO END OF LINES (WNDWDTH) 1.0204 USEC (13+27/2*A+5/2*A*A)
FC80: FC81: FC82: FC82: FC82: FC82: FC82: FC82: FC91: FC91: FC91: FC91: FC92: FC92: FC92: FC92: FC92: FC92: FC42: FC42: FC43: FC43: FC43: FC44:	A4 88 69 C5 80 48 20 B1 30 20 B4 20 B1 30 20 B4 20 C5 80 48 20 C5 80 48 20 C5 80 20 C5 C5 C5 C5 C5 C5 C5 C5 C5 C5 C5 C5 C5	21 01 23 0D 24 28 2A F9 E1 00 9E 864 20 28 21 F9 01 FC 01 F6 42 023 3C	FC	SCRL2 SCRL3 CLREOL CLEOL2 CLEOL2 CLEOL2 WAIT WAIT2 WAIT3 NXTA4 NXTA1	STA LDY DEY PLA ADC CMP BCS PHA JSR STA DEY BPL BMI LDY JSR BCS LDY LDA STA DEY BPL BCS LDY LDA STA SCC RTS SEC PHA SEC RTS SEC PHA SEC RTS SEC PHA INY CPY BDL BNE RTS SEC PHA INY CPY BCS SEC PHA INY CPY BCS SC SEC PHA INY CPY BCS SC SC SC SC SC SC SC SC SC SC SC SC S	BAS2H WNDWDTH #\$01 WNDBTM SCRL3 VTABZ (BASL),Y (BAS2L),Y (BAS2L),Y SCRL2 SCRL1 #\$00 CLEOLZ VTAB CH #\$00 (BASL),Y WNDWDTH CLEOL2 #\$01 WAIT3 #\$01 WAIT3 #\$01 WAIT2 A4L NXTA1 A4H A1L A2L	OF SCROLLING WINDOW INCR LINE NUMBER DONE? YES, FINISH FORM BASL,H (BASE ADDR) MOVE A CHR UP ON LINE NEXT CHAR OF LINE NEXT CHAR OF LINE NEXT LINE (ALWAYS TAKEN) CLEAR BOTTOM LINE GET BASE ADDR FOR BOTTOM LINE CARRY IS SET CURSOR H INDEX STORE BLANKS FROM 'HERE' TO END OF LINES (WNDWDTH) 1.0204 USEC (13+27/2*A+5/2*A*A) INCR 2-BYTE A4 AND A1

FCC0: E5 3F FCC2: E6 3C	SE	BC A2H	(CARRY SET IF >=)
FCC2: E6 3C	II	NC A1L	(CARRY SET IF >=)
FCC4: D0 02	BI	NE RTS4B	
FCC6: E6 3D	IN RTS4B RT	NC A1H	
FCC8: 60	RTS4B RT	TS	
FCC9: A0 4B	HEADR LI	DY #\$4B	WRITE A*256 'LONG 1' HALF CYCLES (650 USEC EACH) THEN A 'SHORT 0' (400 USEC) WRITE TWO HALF CYCLES OF 250 USEC (101)
FCCB: 20 DB FC	JS	SR ZERDLY	HALF CYCLES
FCCE: DO F9	BN	NE HEADR	(650 USEC EACH)
FCDO. 60 FF	21	DC #¢FF	(050 Oble Enery
FCDU: 69 FE	AL	DC #ŞFE	
FCDZ: BU F5	BC	CS HEADR	THEN A SHORT U
FCD4: A0 21	الل	DY #\$21	(400 USEC)
FCD6: 20 DB FC	WRBIT JS	SR ZERDLY	WRITE TWO HALF CYCLES
FCD9: C8	11	NY	OF 250 USEC ('0')
FCD9: C8 FCDA: C8 FCDB: 88 FCDC: D0 FD FCDE: 90 05	II	NY	OR 500 USEC ('0')
FCDB: 88	ZERDLY DE	EY	
FCDC: D0 FD	BI	NE ZERDLY	
FCDE: 90 05	BC	CC WRTAPE	Y IS COUNT FOR
FCE0: A0 32	T.T	DY #\$32	TIMING LOOP
ECE2. 00		EV #452	
FCE2: 88 FCE3: D0 FD		NE ONEDIV	
FCES: DU FD		NE UNEDLI	
FCE5: AC 20 C0 FCE8: A0 2C	WRIAPE LI	DY IAPEOUI	
FCE8: A0 2C	LI	DY #\$2C	
FCEA: CA	DE	EX	
FCEB: 60	R	TS	
FCEC: A2 08	RDBYTE LI	DX #\$08	8 BITS TO READ READ TWO TRANSITIONS (FIND EDGE)
FCEE: 48	RDBYT2 PH	HA	READ TWO TRANSITIONS
FCEF: 20 FA FC	JS	SR RD2BIT	(FIND EDGE)
FCEF: 20 FA FC FCF2: 68	זס	LA	/
FCF3: 2A	F1	LA OL	NEXT BIT
	RC		NEAT BIT
FCF4: A0 3A	للنا 	DY #\$3A 	COUNT FOR SAMPLES
FCF6: CA		EX	
FCF7: D0 F5	BI	NE RDBYT2	
FCF9: 60 FCFA: 20 FD FC	R	TS	
FCFA: 20 FD FC	RD2BIT JS	SR RDBIT	
FCFD: 88	RDBIT DE	EY	DECR Y UNTIL
FCFD: 88 FCFE: AD 60 C0	LI	DA TAPEIN	TAPE TRANSITION
FD01: 45 2F	F(OP LAGTIN	
FD03: 10 F8	BI	PI. RDRTT	
FD05: 45 2F	F	OP LASTIN	
	EC	TA LASIIN	
FD07: 85 2F	51 01	IA LASIIN	SET CARRY ON Y
FD09: C0 80	CI	PY #\$80	SET CARRY ON Y
FDUB: 60	R	TS	
FD0C: A4 24	RDKEY LI	DY CH	
FD0E: B1 28	LI	DA (BASL),Y	SET SCREEN TO FLASH
FD10: 48	PH	HA	
FD11: 29 3F	A	ND #\$3F	
FD13: 09 40	OF	RA #\$40	
FD15: 91 28		TA (BASL),Y	
1910. 91 20	5.		
FD17, 68			
FD17: 68	PI	MD (KGWI.)	CO TO LIGER KEY-IN
FD17: 68 FD18: 6C 38 00	JN VEVIN IN	MP (KSWL)	GO TO USER KEY-IN
FD17: 68 FD18: 6C 38 00 FD1B: E6 4E	JI KEYIN IN	MP (KSWL) NC RNDL	GO TO USER KEY-IN
FD1D: D0 02	BI	NE KEYIN2	GO TO USER KEY-IN INCR RND NUMBER
FD1D: D0 02	BI	NE KEYIN2	INCR RND NUMBER
FD1D: D0 02 FD1F: E6 4F FD21: 2C 00 C0	BN IN KEYIN2 BI	NE KEYIN2 NC RNDH IT KBD	INCR RND NUMBER
FD1D: D0 02 FD1F: E6 4F FD21: 2C 00 C0 FD24: 10 F5	BN IN KEYIN2 BI	NE KEYIN2 NC RNDH IT KBD	INCR RND NUMBER
FD1D: D0 02 FD1F: E6 4F FD21: 2C 00 C0 FD24: 10 F5 FD26: 91 28	BN IN KEYIN2 BI	NE KEYIN2 NC RNDH IT KBD	INCR RND NUMBER
FD1D: D0 02 FD1F: E6 4F FD21: 2C 00 C0 FD24: 10 F5	Bî Iî KEYIN2 BI BI SI LI	NE KEYIN2 NC RNDH IT KBD PL KEYIN TA (BASL),Y DA KBD	INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE
FD1D: D0 02 FD1F: E6 4F FD21: 2C 00 C0 FD24: 10 F5 FD26: 91 28	Bî Iî KEYIN2 BI BI SI LI	NE KEYIN2 NC RNDH IT KBD PL KEYIN TA (BASL),Y DA KBD	INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE
FD1D: D0 02 FD1F: E6 4F FD21: 2C 00 C0 FD24: 10 F5 FD26: 91 28 FD28: AD 00 C0 FD2B: 2C 10 C0 FD2E: 60	Bî Iî KEYIN2 BI Sî LI BI RÎ	NE KEYIN2 NC RNDH IT KBD PL KEYIN TA (BASL),Y DA KBD IT KBDSTRB TS	INCR RND NUMBER
FD1D: D0 02 FD1F: E6 4F FD21: 2C 00 C0 FD24: 10 F5 FD26: 91 28 FD28: AD 00 C0 FD2B: 2C 10 C0 FD2E: 60	Bî Iî KEYIN2 BI Sî LI BI RÎ	NE KEYIN2 NC RNDH IT KBD PL KEYIN TA (BASL),Y DA KBD IT KBDSTRB TS	INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE CLR KEY STROBE GET KEYCODE
FD1D: D0 02 FD1F: E6 4F FD21: 2C 00 C0 FD24: 10 F5 FD26: 91 28 FD28: AD 00 C0 FD2B: 2C 10 C0 FD2E: 60	Bî Iî KEYIN2 BI Sî LI BI RÎ	NE KEYIN2 NC RNDH IT KBD PL KEYIN TA (BASL),Y DA KBD IT KBDSTRB TS	INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE CLR KEY STROBE GET KEYCODE
FD1D: D0 02 FD1F: E6 4F FD21: 2C 00 C0 FD24: 10 F5 FD26: 91 28 FD28: AD 00 C0 FD2B: 2C 10 C0 FD2E: 60	Bî Iî KEYIN2 BI Sî LI BI RÎ	NE KEYIN2 NC RNDH IT KBD PL KEYIN TA (BASL),Y DA KBD IT KBDSTRB TS	INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE CLR KEY STROBE GET KEYCODE
FD1D: D0 02 FD1F: E6 4F FD21: 2C 00 C0 FD24: 10 F5 FD26: 91 28 FD28: AD 00 C0 FD2B: 2C 10 C0 FD2B: 2C 10 C0 FD2F: 20 0C FD FD2F: 20 0C FD FD35: 20 0C FD	EXC JS RDCHAR JS	NE KEYIN2 NC RNDH IT KBD PL KEYIN TA (BASL),Y DA KBD IT KBDSTRB TS SR RDKEY SR ESC1 SR RDKEY	INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE CLR KEY STROBE GET KEYCODE HANDLE ESC FUNC. READ KEY
FD1D: D0 02 FD1F: E6 4F FD21: 2C 00 C0 FD24: 10 F5 FD26: 91 28 FD28: AD 00 C0 FD2B: 2C 10 C0 FD2E: 60 FD2F: 20 0C FD FD32: 20 2C FC FD35: 20 0C FD FD38: C9 9B	EXC JS RDCHAR JS	NE KEYIN2 NC RNDH IT KBD PL KEYIN TA (BASL),Y DA KBD IT KBDSTRB TS SR RDKEY SR ESC1 SR RDKEY	INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE CLR KEY STROBE GET KEYCODE HANDLE ESC FUNC. READ KEY
FD1D: D0 02 FD1F: E6 4F FD21: 2C 00 C0 FD24: 10 F5 FD26: 91 28 FD28: AD 00 C0 FD2B: 2C 10 C0 FD2E: 60 FD2F: 20 0C FD FD32: 20 0C FD FD32: 20 0C FD FD38: C9 9B FD3A: F0 F3	ESC JS RDCHAR JS	NE KEYIN2 NC RNDH IT KBD PL KEYIN TA (BASL),Y DA KBD IT KBDSTRB TS SR RDKEY SR ESC1 SR RDKEY MP #\$9B EQ ESC	INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE CLR KEY STROBE GET KEYCODE HANDLE ESC FUNC. READ KEY
FD1D: D0 02 FD1F: E6 4F FD21: 2C 00 C0 FD24: 10 F5 FD26: 91 28 FD28: AD 00 C0 FD2E: 60 FD2F: 20 0C FD FD32: 20 2C FC FD35: 20 0C FD FD35: 20 0C FD FD38: C9 9B FD3A: F0 F3 FD3C: 60	BN IN KEYIN2 BJ BI ST LI BJ RT ESC JS JS RDCHAR JS CN BH RT	NE KEYIN2 NC RNDH IT KBD PL KEYIN TA (BASL),Y DA KBD IT KBDSTRB TS SR RDKEY SR SR RDKEY SR RDKEY MP #\$9B EQ ESC TS TS	INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE CLR KEY STROBE GET KEYCODE HANDLE ESC FUNC. READ KEY
FD1D: D0 02 FD1F: E6 4F FD21: 2C 00 C0 FD24: 10 F5 FD26: 91 28 FD28: AD 00 C0 FD2B: 2C 10 C0 FD2E: 60 FD2F: 20 0C FD FD32: 20 2C FC FD35: 20 0C FD FD38: C9 9B FD3A: F0 F3 FD3C: 60 FD3D: A5 32	ESC JS RDCHAR JS NOTCR LI	NE KEYIN2 NC RNDH IT KBD PL KEYIN TA (BASL),Y DA KBD IT KBDSTRB TS SR RDKEY SR RDKEY SR RDKEY MP #\$9B EQ ESC TS DA INVFLG	INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE CLR KEY STROBE GET KEYCODE HANDLE ESC FUNC. READ KEY
FD1D: D0 02 FD1F: E6 4F FD21: 2C 00 C0 FD24: 10 F5 FD26: 91 28 FD28: AD 00 C0 FD2B: 2C 10 C0 FD2E: 60 FD2F: 20 0C FD FD32: 20 2C FC FD35: 20 0C FD FD38: C9 9B FD3A: F0 F3 FD3C: 60 FD3D: A5 32 FD3F: 48	KEYIN2 BI KEYIN2 BI ST LI BI CI ESC JS RDCHAR JS CN BI RDCHAR LI NOTCR LI PH	NE KEYIN2 NC RNDH IT KBD PL KEYIN TA (BASL),Y DA KBD IT KBDSTRB TS SR RDKEY SR ESC1 SR RDKEY SR ESC1 SR RDKEY MP #\$9B EQ ESC TS DA INVFLG HA	INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE CLR KEY STROBE GET KEYCODE HANDLE ESC FUNC. READ KEY
FD1D: D0 02 FD1F: E6 4F FD21: 2C 00 C0 FD24: 10 F5 FD26: 91 28 FD28: AD 00 C0 FD28: 2C 10 C0 FD28: 2C 10 C0 FD27: 20 0C FD FD37: 20 0C FD FD35: 20 0C FD FD38: C9 9B FD3A: F0 F3 FD3A: F0 F3 FD3C: 60 FD3D: A5 32 FD3F: 48 FD40: A9 FF	KEYIN2 BI KEYIN2 BI ST LI BI RT ESC JS RDCHAR JS CN BE RT NOTCR LI LI	NE KEYIN2 NC RNDH IT KBD PL KEYIN TA (BASL),Y DA KBD IT KBDSTRB TS SR RDKEY SR ESC1 SR RDKEY MP #\$9B EQ ESC TS DA INVFLG HA DA #\$FF	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
FD1D: D0 02 FD1F: E6 4F FD21: 2C 00 C0 FD24: 10 F5 FD26: 91 28 FD28: AD 00 C0 FD2B: 2C 10 C0 FD2E: 60 FD2F: 20 0C FD FD32: 20 2C FC FD35: 20 0C FD FD38: C9 9B FD3A: F0 F3 FD3C: 60 FD3D: A5 32 FD3F: 48 FD40: A9 FF FD42: 85 32	KEYIN2 BI KEYIN2 BI ST LI BI RT ESC JS RDCHAR JS CN BE RT NOTCR LI LI	NE KEYIN2 NC RNDH IT KBD PL KEYIN TA (BASL),Y DA KBD IT KBDSTRB TS SR RDKEY SR ESC1 SR RDKEY MP #\$9B EQ ESC TS DA INVFLG HA DA #\$FF	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
FD1D: D0 02 FD1F: E6 4F FD21: 2C 00 C0 FD24: 10 F5 FD26: 91 28 FD28: AD 00 C0 FD28: 2C 10 C0 FD28: 2C 10 C0 FD27: 20 0C FD FD37: 20 0C FD FD35: 20 0C FD FD38: C9 9B FD3A: F0 F3 FD3A: F0 F3 FD3C: 60 FD3D: A5 32 FD3F: 48 FD40: A9 FF	KEYIN2 BI KEYIN2 BI ST LI BI RT ESC JS RDCHAR JS CN BE RT NOTCR LI LI	NE KEYIN2 NC RNDH IT KBD PL KEYIN TA (BASL),Y DA KBD IT KBDSTRB TS SR RDKEY SR ESC1 SR RDKEY MP #\$9B EQ ESC TS DA INVFLG HA DA #\$FF	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
FD1D: D0 02 FD1F: E6 4F FD21: 2C 00 C0 FD24: 10 F5 FD26: 91 28 FD28: AD 00 C0 FD2B: 2C 10 C0 FD2E: 60 FD2F: 20 0C FD FD32: 20 2C FC FD35: 20 0C FD FD38: C9 9B FD3A: F0 F3 FD3C: 60 FD3D: A5 32 FD3F: 48 FD40: A9 FF FD42: 85 32	KEYIN2 BI KEYIN2 BI ST LI BI RT ESC JS RDCHAR JS RDCHAR JS RDCHAR JS NOTCR LI PH LI	NE KEYIN2 NC RNDH IT KBD PL KEYIN TA (BASL),Y DA KBD IT KBDSTRB TS SR RDKEY SR ESC1 SR RDKEY MP #\$9B EQ ESC TS DA INVFLG HA DA #\$FF	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
FD1D: D0 02 FD1F: E6 4F FD21: 2C 00 C0 FD24: 10 F5 FD26: 91 28 FD28: AD 00 C0 FD28: 2C 10 C0 FD28: 20 0C FD FD32: 20 0C FD FD32: 20 0C FD FD38: C9 9B FD3A: F0 F3 FD3C: 60 FD3D: A5 32 FD3F: 48 FD40: A9 FF FD42: 85 32 FD44: BD 00 02	BR IN KEYIN2 BJ ST LI BJ CN ESC JS RDCHAR JS RDCHAR JS RDCHAR LI RT NOTCR LI ST LI ST LI ST PP	NE KEYIN2 NC RNDH IT KBD PL KEYIN TA (BASL),Y DA KBD IT KBDSTRB TS SR RDKEY SR RDKEY SR ESC1 SR RDKEY MP #\$9B EQ ESC TS DA INVFLG HA DA #\$FF TA INVFLG DA IN,X SR COUT LA	INCR RND NUMBER KEY DOWN? LOOP REPLACE FLASHING SCREEN GET KEYCODE CLR KEY STROBE GET KEYCODE HANDLE ESC FUNC. READ KEY
FD1D: D0 02 FD1F: E6 4F FD21: 2C 00 C0 FD24: 10 F5 FD26: 91 28 FD28: AD 00 C0 FD2B: 2C 10 C0 FD2F: 20 0C FD FD32: 20 2C FC FD35: 20 0C FD FD36: C9 9B FD33: F0 FD37: 60 FD FD32: 60 FD30: A5 32 FD3F: 48 FD40: A9 FF FD42: 85 32 FD44: BD 00 02 FD47: 20 ED FD	BR IN KEYIN2 BJ ST LI BJ CN ESC JS RDCHAR JS RDCHAR JS RDCHAR LI RT NOTCR LI ST LI ST LI ST PP	NE KEYIN2 NC RNDH IT KBD PL KEYIN TA (BASL),Y DA KBD IT KBDSTRB TS SR RDKEY SR RDKEY SR ESC1 SR RDKEY MP #\$9B EQ ESC TS DA INVFLG HA DA #\$FF TA INVFLG DA IN,X SR COUT LA	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
FD1D: D0 02 FD1F: E6 4F FD21: 2C 00 C0 FD24: 10 F5 FD26: 91 28 FD28: AD 00 C0 FD28: 2C 10 C0 FD28: 20 0C FD FD27: 20 0C FD FD32: 20 2C FC FD35: 20 0C FD FD36: C9 9B FD37: FD37: 70 C0 FD FD37: 70 C0 FD FD37: 48 FD40: A9 FD42: 85 32 FD42: 85 FD42: 85 32 FD44: BD 00 02 FD47: 20 ED FD FD FD FD44: 85 32 FD44: 85 32	BR IN KEYIN2 BJ ST LI BJ CN ESC JS RDCHAR JS RDCHAR JS RDCHAR LI RT NOTCR LI ST LI ST LI ST PP	NE KEYIN2 NC RNDH IT KBD PL KEYIN TA (BASL),Y DA KBD IT KBDSTRB TS SR RDKEY SR RDKEY SR ESC1 SR RDKEY MP #\$9B EQ ESC TS DA INVFLG HA DA #\$FF TA INVFLG DA IN,X SR COUT LA	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
FD1D: D0 02 FD1F: E6 4F FD21: 2C 00 C0 FD24: 10 F5 FD26: 91 28 FD28: AD 00 C0 FD28: 2C 10 C0 FD28: 20 0C FD FD28: 20 0C FD FD32: 20 0C FD FD35: 20 0C FD FD36: 69 9B FD37: FD37: 70 F3 FD37: FD37: 70 F3 FD37: FD37: 48 FD40: A9 FD42: 85 32 FD47: FD44: BD 00 02 FD44: B0 00 02 FD44: 68 FD FD FD48: 85 32 FD41: FD49: 85 32 FD41:	KEYIN2 BI KEYIN2 BI ST LI BI RT ESC JS RDCHAR JS RDCHAR JS RDCHAR JS NOTCR LI ST LI ST LI ST LI LI ST LI	NE KEYIN2 NC RNDH IT KBD PL KEYIN TA (BASL),Y DA KBD IT KBDSTRB TS SR RDKEY SR ESC1 SR RDKEY MP #\$9B EQ ESC TS DA INVFLG DA #\$FF TA INVFLG DA IN,X SR COUT LA TA INVFLG DA IN,X	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
FD1D: D0 02 FD1F: E6 4F FD21: 2C 00 C0 FD24: 10 F5 FD26: 91 28 FD28: AD 00 C0 FD28: 2C 10 C0 FD28: 20 2C FD FD32: 20 2C FD FD35: 20 2C FD FD36: 60 9B FD37: FD37: 48 FD40: A9 FD42: 85 32 FD44: FD42: 85 32 FD44: FD40: 85 32 FD41: FD40: 85 32 FD41: F	KEYIN2 BI KEYIN2 BI ST LI BI RT ESC JS RDCHAR JS RDCHAR JS RDCHAR JS NOTCR LI ST LI ST LI ST LI LI ST LI	NE KEYIN2 NC RNDH IT KBD PL KEYIN TA (BASL),Y DA KBD IT KBDSTRB TS SR RDKEY SR ESC1 SR RDKEY MP #\$9B EQ ESC TS DA INVFLG DA #\$FF TA INVFLG DA IN,X SR COUT LA TA INVFLG DA IN,X	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
FD1D: D0 02 FD1F: E6 4F FD21: 2C 00 C0 FD24: 10 F5 FD26: 91 28 FD28: AD 00 C0 FD2B: 2C 10 C0 FD2F: 20 0C FD FD32: 20 2C FC FD35: 20 0C FD FD36: C9 9B FD37: 80 FD37: A5 32 FD37: 48 FD40: A9 FF FD42: 85 32 FD44: BD 00 02 FD47: 20 ED FD FD44: BD 00 02 FD44: 68 FD44: 68 FD41: 85 32 FD44: 68 FD44: 68 FD44: 60 02 FD50: C9 88 FD52: F0 1D 70 70	KEYIN2 BI KEYIN2 BI ST LI BI RT ESC JS RDCHAR JS RDCHAR JS RDCHAR JS NOTCR LI ST LI ST LI ST LI LI ST LI	NE KEYIN2 NC RNDH IT KBD PL KEYIN TA (BASL),Y DA KBD IT KBDSTRB TS SR RDKEY SR ESC1 SR RDKEY MP #\$9B EQ ESC TS DA INVFLG DA #\$FF TA INVFLG DA IN,X SR COUT LA TA INVFLG DA IN,X	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
FD1D: D0 02 FD1F: E6 4F FD21: 2C 00 C0 FD24: 10 F5 FD26: 91 28 FD28: AD 00 C0 FD28: 2C 10 C0 FD28: 20 0C FD FD28: 20 0C FD FD28: 20 0C FD FD32: 20 2C FC FD35: 20 0C FD FD36: 60 9B FD37: 60 FD30: A5 32 FD40: A9 FF FD42: 85 32 FD44: B0 00 02 FD47: 20 ED FD FD44: 68 FD44: 68 FD40: 8D 00 02 FD42: 69 88 FD45: 69 88 FD52: F0 1D FD55: 59 88 50 50 50 50	KEYIN2 BI KEYIN2 BI ST LI BI CA ESC JS RDCHAR JS RDCHAR JS RDCHAR LI ST LI ST ST ST LI ST ST CA CA BI ST ST CA CA CA CA CA CA CA CA CA CA CA CA CA	NE KEYIN2 NC RNDH IT KBD PL KEYIN TA (BASL),Y DA KBD IT KBDSTRB TS SR SR RDKEY SR ESC1 SR RDKEY MP #\$9B EQ ESC TS INVFLG DA INVFLG DA INVFLG DA IN,X SR COUT LA INVFLG DA IN,X SR COUT LA EQ PA IN,X SR COUT LA EQ DA IN,X SR EQ ROW #\$98	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
FD1D: D0 02 FD1F: E6 4F FD21: 2C 00 C0 FD24: 10 F5 FD26: 91 28 FD28: AD 00 C0 FD28: 2C 10 C0 FD28: 20 0C FD FD27: 20 0C FD FD32: 20 2C FC FD35: 20 0C FD FD36: 60 FT FD37: FD37: 48 FD40: A9 FF FD42: 85 32 FD41: 8D 00 02 FD47: 20 ED FD FD FD FD FD FD48: 85 32 FD41: B0 00 02 FD54: 68 FD49: B0 00 02 FD50: C9 98 FD52: F0 1D FD54: C9 98 FD55: C9 98 FD54:	KEYIN2 BI KEYIN2 BI ST LI BI CA ESC JS RDCHAR JS RDCHAR JS CA NOTCR LI ST LI ST ST LI ST ST LI CA BI CA BI ST ST LI ST ST LI ST ST ST ST ST ST ST ST ST ST ST ST ST	NE KEYIN2 NC RNDH IT KBD PL KEYIN TA (BASL),Y DA KBD IT KBDSTRB TS SR RDKEY SR ESC1 SR RDKEY SR ESC1 SR RDKEY MP #\$9B EQ ESC TS DA INVFLG DA INVFLG DA IN,X SR COUT LA TA INVFLG DA IN,X SR COUT LA TA INVFLG DA IN,X SR COUT LA EQ BCKSPC MP #\$98 EQ CANCEL	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
FD1D: D0 02 FD1F: E6 4F FD21: 2C 00 C0 FD24: 10 F5 FD26: 91 28 FD28: AD 00 C0 FD28: 2C 10 C0 FD28: 20 0C FD FD27: 20 0C FD FD32: 20 2C FC FD35: 20 0C FD FD36: 60 - FD FD37: 48 - FD40: A9 FD42: 85 32 - FD42: 85 32 FD42: 85 32 - FD FD FD FD44: BD 00 02 - FD FD FD44: BD 00 02 - FD - FD FD44: BD 00 02 - - FD - - FD45: E0 88 -	KEYIN2 BI IN KEYIN2 BI ST LI BI CAN ESC JS RDCHAR JS RDCHAR JS RDCHAR LI ST LI ST LI ST LI ST LI ST LI ST LI ST CN BI ST CN CN ST CN CN ST ST CN CN ST CN CN ST ST CN ST ST CN ST ST CN ST ST ST ST ST ST ST ST ST ST ST ST ST	NE KEYIN2 NC RNDH IT KBD PL KEYIN TA (BASL),Y DA KBD IT KBDSTRB TS SR RDKEY SR RDKEY SR RDKEY SR ESC1 SR RDKEY MP #\$98 EQ ESC TS DA INVFLG DA IN,X SR COUT LA TA INVFLG DA IN,X MP #\$88 EQ BCKSPC MP #\$98 EQ CANCEL PX #\$F8	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
FD1D: D0 02 FD1F: E6 4F FD21: 2C 00 C0 FD24: 10 F5 FD26: 91 28 FD28: AD 00 C0 FD28: 2C 10 C0 FD28: 20 0C FD FD28: 20 0C FD FD32: 20 2C FC FD35: 20 0C FD FD37: 48 FD40: A9 FD42: 85 32 FD44: FD42: 85 32 FD44: FD42: 85 32 FD44: FD40: BD 00 02 FD41: BD 00 02 FD50: </td <td>KEYIN2 BR IN KEYIN2 BI ST LL BT ST ST ST RDCHAR JS RDCHAR JS RDCHAR LS NOTCR LL ST LL ST ST ST CN BH CN BH ST ST ST ST ST ST ST ST ST ST ST ST ST</td> <td>NE KEYIN2 NC RNDH IT KBD PL KEYIN TA (BASL),Y DA KBD IT KBDSTRB TS SR SR RDKEY MP #\$9B EQ ESC TS INVFLG HA #\$FF TA INVFLG DA IN,X SR COUT LA INVFLG DA IN,X SR COUT LA INVFLG DA IN,X FQ BCKSPC MP #\$98 EQ CANCEL PX #\$F8</td> <td>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?</td>	KEYIN2 BR IN KEYIN2 BI ST LL BT ST ST ST RDCHAR JS RDCHAR JS RDCHAR LS NOTCR LL ST LL ST ST ST CN BH CN BH ST ST ST ST ST ST ST ST ST ST ST ST ST	NE KEYIN2 NC RNDH IT KBD PL KEYIN TA (BASL),Y DA KBD IT KBDSTRB TS SR SR RDKEY MP #\$9B EQ ESC TS INVFLG HA #\$FF TA INVFLG DA IN,X SR COUT LA INVFLG DA IN,X SR COUT LA INVFLG DA IN,X FQ BCKSPC MP #\$98 EQ CANCEL PX #\$F8	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?
FD1D: D0 02 FD1F: E6 4F FD21: 2C 00 C0 FD24: 10 F5 FD26: 91 28 FD28: AD 00 C0 FD28: 2C 10 C0 FD29: 20 0C FD FD2F: 20 0C FD FD32: 20 2C FC FD35: 20 0C FD FD38: C9 9B FD34: F0 F3 FD37: 48 FD40: A9 FF FD42: 85 32 FD44: BD 00 02 FD44: BD 00 02 FD44: 68 FD45: C9 98 FD45: C9 98 FD55: C9 98 FD56: F0 0A FD58: E0 F8 FD54: 90 03 FD54: 20 20 27 FE	KEYIN2 BI IN KEYIN2 BI ST LI BI CA ESC JS RDCHAR JS CA RDCHAR JS CA NOTCR LI LI ST ST LI ST ST LI CA BI CA BI CA BI CA BI CA BI CA CA ST ST CA CA CA CA CA CA CA CA CA CA CA CA CA	NE KEYIN2 NC RNDH IT KBD PL KEYIN TA (BASL),Y DA KBD IT KBDSTRB TS RDKEY SR RDKEY SR RDKEY SR RDKEY SR RDKEY MP #\$98 EQ ESC TS DA INVFLG DA INVFLG DA IN,X SR COUT LA TA INVFLG DA IN,X SR COUT LA P #\$98 EQ CANCEL PX #\$F8 CC NOTCR1 SR PE I	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
FD1D: D0 02 FD1F: E6 4F FD21: 2C 00 C0 FD24: 10 F5 FD26: 91 28 FD28: AD 00 C0 FD28: 2C 10 C0 FD28: 20 0C FD FD28: 20 0C FD FD28: 20 0C FD FD32: 20 2C FC FD33: C9 9B FD33: FD34: F0 F3 FD37: FD35: 60 FD FD40: A9 FD40: A9 FF FD42: 85 32 FD44: BD 00 02 FD47: 20 ED FD FD44: BD 00 02 FD44: 68 FD44: B0 00 02 FD45: E0 F0 D FD FD 68 FD52: F0 10 FD50: C9	KEYIN2 BI KEYIN2 BI ST LI BI CT ESC JS JS RDCHAR JS CT NOTCR LI ST LI ST LI ST LI ST LI ST CT BI CT BI ST LI ST ST ST ST ST ST ST ST ST ST	NE KEYIN2 NC RNDH IT KBD PL KEYIN TA (BASL),Y DA KBD IT KBDSTRB TS SR SR RDKEY SR ESC1 SR RDKEY MP #\$9B EQ ESC TS INVFLG DA #\$FF TA INVFLG DA IN,X SR COUT LA IN,X PA #\$88 EQ ECKSPC MP #\$88 EQ CANCEL PX #\$F8 CC NOTCR1 SR BELL	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
FD1D: D0 02 FD1F: E6 4F FD21: 2C 00 C0 FD24: 10 F5 FD26: 91 28 FD28: AD 00 C0 FD28: 2C 10 C0 FD28: 20 0C FD FD28: 20 0C FD FD28: 20 0C FD FD32: 20 2C FC FD33: C9 9B FD33: FD34: F0 F3 FD37: FD35: 60 FD FD40: A9 FD40: A9 FF FD42: 85 32 FD44: BD 00 02 FD47: 20 ED FD FD44: BD 00 02 FD44: 68 FD44: B0 00 02 FD45: E0 F0 D FD FD 68 FD52: F0 10 FD50: C9	KEYIN2 BI KEYIN2 BI ST LI BI CT ESC JS JS RDCHAR JS CT NOTCR LI ST LI ST LI ST LI ST LI ST CT BI CT BI ST LI ST ST ST ST ST ST ST ST ST ST	NE KEYIN2 NC RNDH IT KBD PL KEYIN TA (BASL),Y DA KBD IT KBDSTRB TS SR SR RDKEY SR ESC1 SR RDKEY MP #\$9B EQ ESC TS INVFLG DA #\$FF TA INVFLG DA IN,X SR COUT LA IN,X PA #\$88 EQ ECKSPC MP #\$88 EQ CANCEL PX #\$F8 CC NOTCR1 SR BELL	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
FD1D: D0 02 FD1F: E6 4F FD21: 2C 00 C0 FD24: 10 F5 FD26: 91 28 FD28: AD 00 C0 FD28: 2C 10 C0 FD28: 20 0C FD FD28: 20 0C FD FD28: 20 0C FD FD32: 20 2C FC FD33: C9 9B FD33: FD34: F0 F3 FD37: FD35: 60 FD FD40: A9 FD40: A9 FF FD42: 85 32 FD44: BD 00 02 FD47: 20 ED FD FD44: BD 00 02 FD44: 68 FD44: B0 00 02 FD45: E0 F0 D FD FD 68 FD52: F0 10 FD50: C9	KEYIN2 BI KEYIN2 BI ST LI BI CT ESC JS JS RDCHAR JS CT NOTCR LI ST LI ST LI ST LI ST LI ST CT BI CT BI ST LI ST ST ST ST ST ST ST ST ST ST	NE KEYIN2 NC RNDH IT KBD PL KEYIN TA (BASL),Y DA KBD IT KBDSTRB TS SR SR RDKEY SR ESC1 SR RDKEY MP #\$9B EQ ESC TS INVFLG DA #\$FF TA INVFLG DA IN,X SR COUT LA IN,X PA #\$88 EQ ECKSPC MP #\$88 EQ CANCEL PX #\$F8 CC NOTCR1 SR BELL	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
FD1D: D0 02 FD1F: E6 4F FD21: 2C 00 C0 FD24: 10 F5 FD26: 91 28 FD28: AD 00 C0 FD28: 2C 10 C0 FD28: 20 0C FD FD28: 20 0C FD FD28: 20 0C FD FD32: 20 2C FC FD33: C9 9B FD33: FD34: F0 F3 FD37: FD35: 60 FD FD40: A9 FD40: A9 FF FD42: 85 32 FD44: BD 00 02 FD47: 20 ED FD FD44: BD 00 02 FD44: 68 FD44: B0 00 02 FD45: E0 F0 D FD FD 68 FD52: F0 10 FD50: C9	KEYIN2 BI KEYIN2 BI ST LI BI CT ESC JS JS RDCHAR JS CT NOTCR LI ST LI ST LI ST LI ST LI ST CT BI CT BI ST LI ST ST ST ST ST ST ST ST ST ST	NE KEYIN2 NC RNDH IT KBD PL KEYIN TA (BASL),Y DA KBD IT KBDSTRB TS SR SR RDKEY SR ESC1 SR RDKEY MP #\$9B EQ ESC TS INVFLG DA #\$FF TA INVFLG DA IN,X SR COUT LA IN,X PA #\$88 EQ ECKSPC MP #\$88 EQ CANCEL PX #\$F8 CC NOTCR1 SR BELL	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
FD1D: D0 02 FD1F: E6 4F FD21: 2C 00 C0 FD24: 10 F5 FD26: 91 28 FD28: AD 00 C0 FD28: 2C 10 C0 FD28: 20 0C FD FD28: 20 0C FD FD28: 20 0C FD FD32: 20 2C FC FD33: C9 9B FD33: FD34: F0 F3 FD37: FD35: 60 FD FD40: A9 FD40: A9 FF FD42: 85 32 FD44: BD 00 02 FD47: 20 ED FD FD44: BD 00 02 FD44: 68 FD44: B0 00 02 FD45: E0 F0 D FD FD 68 FD52: F0 10 FD50: C9	KEYIN2 BI KEYIN2 BI ST ESC JS RDCHAR JS RDCHAR JS NOTCR LI ST ST LI ST ST CANCEL LI ST ST ST ST ST ST ST ST ST ST ST ST ST	NE KEYIN2 NC RNDH IT KBD PL KEYIN TA (BASL),Y DA KBD IT KBDSTRB TS SR SR RDKEY SR ESC1 SR RDKEY MP #\$9B EQ ESC TS INVFLG DA #\$FF TA INVFLG DA IN,X SR COUT LA IN,X PA #\$88 EQ ECKSPC MP #\$88 EQ CANCEL PX #\$F8 CC NOTCR1 SR BELL	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

FD67:	20	8E	FD	GETLNZ	JSR	CROUT PROMPT COUT	OUTPUT CR
FD6A:	A5	33		GETLN	LDA	PROMPT	
FD6C:	20	ED	FD		JSR	COUT	OUTPUT PROMPT CHAR
FD6F:	A2	01			LDX	#\$01	INIT INPUT INDEX WILL BACKSPACE TO 0
FD71:				BCKSPC	TXA		WILL BACKSPACE TO 0
FD72:				201010	BEO	GETINZ	MILL District To 0
FD74:					DEX	001000	
						DDOULD	
				NXTCHAR	JSR	RDCHAR	
FD78:					CMP	#PICK	USE SCREEN CHAR
FD7A:	D0	02					FOR CTRL-U
FD7C:	В1	28			LDA	(BASL),Y	
FD7E:	C9	Ε0		CAPTST	CMP	#\$E0	
FD80:	90	02			BCC	ADDINP	CONVERT TO CAPS
FD82:	29	DF			AND	#SDF	CONVERT TO CAPS
FD84.	9D	0.0	02	ADDINP	STA	TNX	ADD TO INPUT BUF
FD87.	Ca	8D	02	1.001.00	CMD	#\$80	
EDOO.	00	00			DNE	#90D NOTOD	
FD09.	20	D2	па		TOD	GLDEOI	ALD WO DOL ID AD
FD8B:	20	90	FC	an or m	JSR	CLREOL	CLR IO EOL IF CR
FD8E:	A9	8D		CROUT	LDA	#\$8D	
FD90:	DO	5B			BNE	COUT	
FD92:	Α4	3D		PRA1	LDY	A1H	ADD TO INPUT BUF CLR TO EOL IF CR PRINT CR,A1 IN HEX
FD94:	A6	3C			LDX	A1L	
FD96:	20	8E	FD	PRYX2	JSR	CROUT	
FD99:	20	40	F9		JSR	PRNTYX	
FD9C:	A0	00			LDY	#\$00	
FD9E:	A9	AD			LDA	#\$AD	PRINT '-'
FDA0:	4C	ED	FD		JMP	COUT	
FDA3.	<u>م</u>	30	2	XAM8		A1T.	
FDA5:							SET TO FINISH AT
FDA5:					CTTA	#\$U7	MOD 8=7
					JIA		MOD 8=7
FDA9:					LDA	AIH	
FDAB:					STA		
				MODSCHK			
FDAF:					AND	#\$07	
FDB1:	D0	03			BNE	DATAOUT	
FDB3:	20	92	FD	XAM	JSR	PRA1	
FDB6:	A9	A0			TDA	4000	
FDB8:					JSR	COUT	OUTPUT BLANK
FDBB:					LDA	(A1L).Y	OUTPUT BLANK OUTPUT BYTE IN HEX CHECK IF TIME TO, PRINT ADDR
FDBD:					JSR	PRBYTE	OUTPUT BYTE IN HEX
FDC0:					TCD	NVTA1	
					DCC	MODECUV	CURCE IE TIME TO
FDC3:					BCC	MODSCHK	CHECK IF TIME IO,
	~ ~						
FDC5:	60			101010	1010		
FDC6:	4A			XAMPM	LSR	A	DETERMINE IF MON
FDC5: FDC6: FDC7:	4A			XAMPM	LSR BCC	A XAM	DETERMINE IF MON
FDC6:	4A 90	EA		XAMPM	LSR	A XAM	DETERMINE IF MON
FDC6: FDC7:	4A 90 4A	EA		XAMPM	LSR BCC	A XAM A	DETERMINE IF MON
FDC6: FDC7: FDC9:	4A 90 4A 4A	EA		XAMPM	LSR BCC LSR LSR	A XAM A A	DETERMINE IF MON
FDC6: FDC7: FDC9: FDCA: FDCB: FDCD:	4A 90 4A 4A A5 90	EA 3E 02		XAMPM	LSR BCC LSR LSR LDA BCC	A XAM A A2L ADD	DETERMINE IF MON MODE IS XAM ADD, OR SUB
FDC6: FDC7: FDC9: FDCA: FDCB: FDCD:	4A 90 4A 4A A5 90	EA 3E 02		XAMPM	LSR BCC LSR LSR LDA BCC	A XAM A A2L ADD	DETERMINE IF MON MODE IS XAM ADD, OR SUB
FDC6: FDC7: FDC9: FDCA: FDCB: FDCD:	4A 90 4A 4A A5 90	EA 3E 02		XAMPM	LSR BCC LSR LSR LDA BCC	A XAM A A2L ADD	DETERMINE IF MON MODE IS XAM ADD, OR SUB
FDC6: FDC7: FDC9: FDCA: FDCB: FDCD: FDCF: FDC1:	4A 90 4A 4A A5 90 49 65	EA 3E 02 FF 3C		ADD	LSR BCC LSR LDA BCC EOR ADC	A XAM A A2L ADD	DETERMINE IF MON
FDC6: FDC7: FDC9: FDCA: FDCB: FDCB: FDCF: FDCF: FDD1: FDD3:	4A 90 4A 45 90 49 65 48	EA 3E 02 FF 3C		ADD	LSR BCC LSR LSR LDA BCC EOR ADC PHA	A XAM A A A2L ADD #\$FF A1L	DETERMINE IF MON MODE IS XAM ADD, OR SUB
FDC6: FDC7: FDC9: FDCA: FDCB: FDCD: FDCF: FDD1: FDD3: FDD4:	4A 90 4A 45 90 49 65 48 A9	EA 3E 02 FF 3C BD		ADD	LSR BCC LSR LSR LDA BCC EOR ADC PHA LDA	A XAM A A A2L ADD #\$FF AlL #\$BD	DETERMINE IF MON MODE IS XAM ADD, OR SUB SUB: FORM 2'S COMPLEMENT
FDC6: FDC7: FDC9: FDCA: FDCB: FDCD: FDCF: FDC7: FDD1: FDD3: FDD4: FDD6:	4A 90 4A A5 90 49 65 48 A9 20	EA 3E 02 FF 3C BD ED		ADD	LSR BCC LSR LDA BCC EOR ADC PHA LDA JSR	A XAM A A A2L ADD #\$FF AlL #\$BD	DETERMINE IF MON MODE IS XAM ADD, OR SUB
FDC6 : FDC7 : FDC9 : FDCA : FDCB : FDCD : FDC7 : FDD1 : FDD3 : FDD4 : FDD6 : FDD9 :	4A 90 4A 45 90 49 65 48 A9 20 68	EA 3E 02 FF 3C BD ED	FD	ADD	LSR BCC LSR LSR LDA BCC EOR ADC PHA LDA JSR PLA	A XAM A A2L ADD #\$FF A1L #\$BD COUT	DETERMINE IF MON MODE IS XAM ADD, OR SUB SUB: FORM 2'S COMPLEMENT PRINT '=', THEN RESULT
FDC6: FDC7: FDC9: FDCA: FDCB: FDCD: FDCF: FDD1: FDD3: FDD4: FDD6: FDD9: FDDA:	4A 90 4A 45 90 49 65 48 20 68 48	EA 3E 02 FF 3C BD ED	FD	ADD PRBYTE	LSR BCC LSR LDA BCC EOR ADC PHA LDA JSR PLA PHA	A XAM A A2L ADD #\$FF A1L #\$BD COUT	DETERMINE IF MON MODE IS XAM ADD, OR SUB SUB: FORM 2'S COMPLEMENT PRINT '=', THEN RESULT PRINT BYTE AS 2 HEX
FDC6: FDC7: FDC9: FDCA: FDCB: FDCD: FDCF: FDC7: FDD3: FDD4: FDD4: FDD9: FDD4: FDD4: FDD4: FDD4:	4A 90 4A 4A A5 90 49 65 48 A9 20 68 48 4A	EA 3E 02 FF 3C BD ED	FD	ADD PRBYTE	LSR BCC LSR LSR LDA BCC EOR ADC PHA LDA JSR PLA PHA LSR	A XAM A A2L ADD #\$FF A1L #\$BD COUT	DETERMINE IF MON MODE IS XAM ADD, OR SUB SUB: FORM 2'S COMPLEMENT PRINT '=', THEN RESULT
FDC6: FDC7: FDC9: FDCA: FDCB: FDCD: FDCD1: FDD1: FDD1: FDD4: FDD4: FDD9: FDDA: FDD5: FDDC:	4A 90 4A 45 90 49 65 48 A9 20 68 48 4A 4A	EA 3E 02 FF 3C BD ED	FD	ADD PRBYTE	LSR BCC LSR LSR LDA BCC EOR ADC PHA LDA JSR PLA PLA LSR LSR	A XAM A A2L ADD #\$FF A1L #\$BD COUT	DETERMINE IF MON MODE IS XAM ADD, OR SUB SUB: FORM 2'S COMPLEMENT PRINT '=', THEN RESULT PRINT BYTE AS 2 HEX
FDC6: FDC7: FDC9: FDC8: FDC8: FDC7: FDC7: FDD1: FDD3: FDD4: FDD6: FDD9: FDD6: FDD8: FDD2: FDD0:	4A 90 4A 45 90 49 65 48 A9 20 68 48 4A 4A 4A	EA 3E 02 FF 3C BD ED	FD	ADD PRBYTE	LSR BCC LSR LSR LDA BCC EOR ADC EOR ADC PHA LDA JSR PLA PHA LSR LSR LSR	A XAM A A2L ADD #\$FF AlL #\$BD COUT	DETERMINE IF MON MODE IS XAM ADD, OR SUB SUB: FORM 2'S COMPLEMENT PRINT '=', THEN RESULT PRINT BYTE AS 2 HEX
FDC6: FDC7: FDC9: FDC8: FDC8: FDC7: FDD1: FDD3: FDD4: FDD6: FDD9: FDDA: FDD2: FDDD2: FDDD2:	4A 90 4A A5 90 65 48 A9 20 68 48 4A 4A 4A	EA 3E 02 FF 3C BD ED	FD	ADD PRBYTE	LSR BCC LSR LSR LDA BCC EOR ADC PHA LDA JSR PHA LSR LSR LSR LSR	A XAM A A2L ADD #\$FF A1L #\$BD COUT A A A A A A	DETERMINE IF MON MODE IS XAM ADD, OR SUB SUB: FORM 2'S COMPLEMENT PRINT '=', THEN RESULT PRINT BYTE AS 2 HEX
FDC6: FDC7: FDC7: FDC9: FDC8: FDC5: FDC1: FDD1: FDD3: FDD4: FDD6: FDD8: FDD8: FDD5: FDD5: FDD5:	4A 90 4A A5 90 49 65 48 A9 20 68 48 4A 4A 4A 20	EA 3E 02 FF 3C BD ED	FD	ADD PRBYTE	LSR BCC LSR LSR LSR EOR ADC EOR ADC PHA LDA JSR PHA LSR LSR LSR LSR LSR	A XAM A A2L ADD #\$FF AlL #\$BD COUT	DETERMINE IF MON MODE IS XAM ADD, OR SUB SUB: FORM 2'S COMPLEMENT PRINT '=', THEN RESULT PRINT BYTE AS 2 HEX
FDC6: FDC7: FDC9: FDC9: FDC2: FDC2: FDC1: FDD1: FDD3: FDD4: FDD9: FDD4: FDD2: FDD2: FDD2: FDD5: FDD5: FDD5: FD25:	4A 90 4A 4A A5 90 49 65 48 49 20 68 48 4A 4A 4A 4A 20 68	EA 3E 02 FF 3C BD ED ED	FD	ADD PRBYTE	LSR BCC LSR LSR LSR BCC EOR ADC PHA LDA JSR PLA LSR LSR LSR LSR LSR JSR PLA	A XAM A A2L ADD #\$FF A1L #\$BD COUT A A A A PRHEXZ	DETERMINE IF MON MODE IS XAM ADD, OR SUB SUB: FORM 2'S COMPLEMENT PRINT '=', THEN RESULT PRINT BYTE AS 2 HEX DIGITS, DESTROYS A-REG
FDC6: FDC7: FDC9: FDC9: FDC2: FDC2: FDC1: FDD1: FDD3: FDD4: FDD9: FDD4: FDD2: FDD2: FDD2: FDD5: FDD5: FDD5: FD25:	4A 90 4A A5 90 49 65 48 A9 20 68 48 4A 4A 4A 20 68 29	EA 3E 02 FF 3C BD ED ED	FD	ADD PRBYTE	LSR BCC LSR LSR LDA BCC EOR ADC PHA LDA JSR PLA LSR LSR LSR LSR LSR LSR AND	A XAM A A A2L ADD #\$FF A1L #\$BD COUT A A A PRHEXZ #\$0F	DETERMINE IF MON MODE IS XAM ADD, OR SUB SUB: FORM 2'S COMPLEMENT PRINT '=', THEN RESULT PRINT BYTE AS 2 HEX DIGITS, DESTROYS A-REG PRINT HEX DIG IN A-REG
FDC6: FDC7: FDC9: FDC9: FDC2: FDC2: FDC1: FDD1: FDD3: FDD4: FDD9: FDD4: FDD2: FDD2: FDD2: FDD5: FDD5: FDD5: FD25:	4A 90 4A A5 90 49 65 48 A9 20 68 48 4A 4A 4A 20 68 29	EA 3E 02 FF 3C BD ED ED	FD	ADD PRBYTE PRHEX PRHEXZ	LSR BCC LSR LDA BCC EOR ADC PHA LDA JSR PLA PHA LSR LSR LSR LSR LSR AND ORA	A XAM A A2L ADD #\$FF A1L #\$BD COUT A A A A PRHEXZ #\$0F #\$B0	DETERMINE IF MON MODE IS XAM ADD, OR SUB SUB: FORM 2'S COMPLEMENT PRINT '=', THEN RESULT PRINT BYTE AS 2 HEX DIGITS, DESTROYS A-REG
FDC6: FDC7: FDC9: FDC9: FDC2: FDC2: FDC1: FDD1: FDD3: FDD4: FDD9: FDD4: FDD2: FDD2: FDD2: FDD5: FDD5: FDD5: FD25:	4A 90 4A A5 90 49 65 48 A9 20 68 48 4A 4A 4A 20 68 29 09	EA 3E 02 FF 3C BD ED ED E5 0F B0	FD	ADD PRBYTE PRHEX PRHEXZ	LSR BCC LSR LDA BCC EOR ADC PHA LDA JSR PLA PHA LSR LSR LSR LSR LSR AND ORA	A XAM A A2L ADD #\$FF A1L #\$BD COUT A A A A PRHEXZ #\$0F #\$B0	DETERMINE IF MON MODE IS XAM ADD, OR SUB SUB: FORM 2'S COMPLEMENT PRINT '=', THEN RESULT PRINT BYTE AS 2 HEX DIGITS, DESTROYS A-REG PRINT HEX DIG IN A-REG
FDC6: FDC7: FDC7: FDC9: FDC4: FDC5: FDC1: FDD1: FDD4: FDD4: FDD9: FDD4: FDD9: FDD2: FDD5: FDD5: FDD5: FDE2:	4A 90 4A A5 90 65 48 49 20 68 48 4A 4A 4A 20 68 29 09 C9	EA 3E 02 FF 3C BD ED ED E5 0F B0 BA	FD	ADD PRBYTE PRHEX PRHEXZ	LSR BCC LSR LDA BCC EOR ADC PHA LDA JSR PLA PHA LSR LSR LSR LSR LSR AND ORA	A XAM A A A2L ADD #\$FF A1L #\$BD COUT A A A PRHEXZ #\$0F	DETERMINE IF MON MODE IS XAM ADD, OR SUB SUB: FORM 2'S COMPLEMENT PRINT '=', THEN RESULT PRINT BYTE AS 2 HEX DIGITS, DESTROYS A-REG PRINT HEX DIG IN A-REG
FDC6: FDC7: FDC9: FDC9: FDC4: FDC5: FDC7: FDD1: FDD3: FDD4: FDD9: FDD9: FDD9: FDD2: FDD2: FDD2: FDD2: FDD2: FDD2: FDE5: FDE5: FDE9: FDE9: FDE9: FDE9:	4A 90 4A 4A 5 90 49 65 48 49 20 68 48 4A 4A 4A 20 68 29 09 C9 90 69	EA 3E 02 FF 3C ED ED ED E5 0F B0 BA 02 06	FD	ADD PRBYTE PRHEX PRHEXZ	LSR BCC LSR LSR LDA BCC EOR ADC PHA LDA JSR PHA LSR LSR LSR LSR LSR LSR CMP BCC ADC	A XAM A A A2L ADD #\$FF A1L #\$BD COUT A A A PRHEXZ #\$0F #\$B0 #\$BA COUT #\$06	DETERMINE IF MON MODE IS XAM ADD, OR SUB SUB: FORM 2'S COMPLEMENT PRINT '=', THEN RESULT PRINT BYTE AS 2 HEX DIGITS, DESTROYS A-REG PRINT HEX DIG IN A-REG LSB'S
FDC6: FDC7: FDC9: FDC9: FDC4: FDC5: FDC7: FDD1: FDD3: FDD4: FDD9: FDD9: FDD9: FDD2: FDD2: FDD2: FDD2: FDD2: FDD2: FDE5: FDE5: FDE9: FDE9: FDE9: FDE9:	4A 90 4A 4A 5 90 49 65 48 49 20 68 48 4A 4A 4A 20 68 29 09 C9 90 69	EA 3E 02 FF 3C ED ED ED E5 0F B0 BA 02 06	FD	ADD PRBYTE PRHEX PRHEXZ	LSR BCC LSR LSR LDA BCC EOR ADC PHA LDA JSR PHA LSR LSR LSR LSR LSR LSR CMP BCC ADC	A XAM A A A2L ADD #\$FF A1L #\$BD COUT A A A PRHEXZ #\$0F #\$B0 #\$BA COUT #\$06	DETERMINE IF MON MODE IS XAM ADD, OR SUB SUB: FORM 2'S COMPLEMENT PRINT '=', THEN RESULT PRINT BYTE AS 2 HEX DIGITS, DESTROYS A-REG PRINT HEX DIG IN A-REG LSB'S
FDC6: FDC7: FDC7: FDC9: FDC8: FDC8: FDC9: FDD1: FDD4: FDD9: FDD9: FDD9: FDD9: FDD9: FDD2: FDD5: FDD5: FDD5: FDE5: FDE5: FDE9:	4A 90 4A 4A 5 90 49 65 48 49 20 68 48 4A 4A 4A 20 68 29 09 C9 90 69 60 60	EA 3E 02 FF 3C ED ED E5 0F B0 BA 02 06 36	FD FD	ADD PRBYTE PRHEX PRHEXZ	LSR BCC LSR LDA BCC EOR ADC PHA LDA JSR PLA LSR LSR LSR LSR LSR LSR AND ORA CMP BCC ADC JMP CMP	A XAM A A A2L ADD #\$FF A1L #\$BD COUT A A A A PRHEXZ #\$0F #\$B0 #\$BA COUT #\$06 (CSWL) #\$A0	DETERMINE IF MON MODE IS XAM ADD, OR SUB SUB: FORM 2'S COMPLEMENT PRINT '=', THEN RESULT PRINT BYTE AS 2 HEX DIGITS, DESTROYS A-REG PRINT HEX DIG IN A-REG LSB'S VECTOR TO USER OUTPUT ROUTINE
FDC6: FDC7: FDC7: FDC7: FDC8: FDC8: FDC9: FDD1: FDD4: FDD4: FDD9: FDD4: FDD9: FDD2: FDD5: FDD5: FDD5: FDE2: FDE5: FDE9:	4A 90 4A A5 90 49 65 48 49 20 68 48 4A 4A 4A 20 68 84 8 4A 20 68 29 09 09 67 90 67 20 90 67 20 90 67 20 90 90 67 20 90 90 67 20 90 90 90 90 90 90 90 90 90 90 90 90 90	EA 3E 02 FF 3C ED ED E5 0F B0 BA 02 06 36 A0	FD FD	ADD PRBYTE PRHEX PRHEXZ	LSR BCC LSR LDA BCC EOR ADC PHA LDA JSR PLA LSR LSR LSR LSR LSR LSR AND ORA CMP BCC ADC JMP CMP	A XAM A A A2L ADD #\$FF A1L #\$BD COUT A A A A PRHEXZ #\$0F #\$B0 #\$BA COUT #\$06 (CSWL) #\$A0	DETERMINE IF MON MODE IS XAM ADD, OR SUB SUB: FORM 2'S COMPLEMENT PRINT '=', THEN RESULT PRINT BYTE AS 2 HEX DIGITS, DESTROYS A-REG PRINT HEX DIG IN A-REG LSB'S VECTOR TO USER OUTPUT ROUTINE
FDC6: FDC7: FDC7: FDC7: FDC8: FDC8: FDC7: FDD1: FDD1: FDD4: FDD4: FDD4: FDD4: FDD6: FDD7: FDD5: FDD7: FDE5: FDE7: FDE9:	4A 90 4A A5 90 49 65 48 49 20 68 48 4A 4A 4A 20 68 84 4A 20 68 29 09 69 60 60 90 60 90 90	EA 3E 02 FF 3C BD ED ED E5 0F B0 BA 02 06 36 A0 02	FD FD	ADD PRBYTE PRHEX PRHEXZ	LSR BCC LSR LDA BCC EOR ADC PHA LDA JSR PLA LSR LSR LSR LSR LSR LSR AND ORA CMP BCC ADC JMP CMP	A XAM A A A2L ADD #\$FF A1L #\$BD COUT A A A A PRHEXZ #\$0F #\$B0 #\$BA COUT #\$06 (CSWL) #\$A0	DETERMINE IF MON MODE IS XAM ADD, OR SUB SUB: FORM 2'S COMPLEMENT PRINT '=', THEN RESULT PRINT BYTE AS 2 HEX DIGITS, DESTROYS A-REG PRINT HEX DIG IN A-REG LSB'S VECTOR TO USER OUTPUT ROUTINE DON'T OUTPUT CTRL'S INVERSE
FDC6: FDC7: FDC7: FDC9: FDC8: FDC5: FDC1: FDD1: FDD3: FDD4: FDD6: FDD6: FDD6: FDD7: FDD7: FDD2: FDE5: FDE5: FDE9:	4A 90 4A 4A 45 90 49 65 48 49 20 68 48 4A 4A 4A 20 68 29 09 29 90 60 25	EA 3E 02FF 3C BD ED ED E5 0F B0 B0 B0 206 36 A0 02 32	FD FD	ADD PRBYTE PRHEX PRHEXZ COUT COUT1	LSR BCC LSR LDA BCC EOR ADC PHA LDA JSR PHA LSR LSR LSR LSR LSR LSR CMP BCC ADC JMP BCC AND	A XAM A A ADD #\$FF AlL #\$BD COUT A A A A PRHEXZ #\$0F #\$B0 #\$B0 #\$BA COUT #\$06 (CSWL) #\$06 (CSWL) #\$06 (CSWL)	DETERMINE IF MON MODE IS XAM ADD, OR SUB SUB: FORM 2'S COMPLEMENT PRINT '=', THEN RESULT PRINT BYTE AS 2 HEX DIGITS, DESTROYS A-REG PRINT HEX DIG IN A-REG LSB'S VECTOR TO USER OUTPUT ROUTINE DON'T OUTPUT CTRL'S INVERSE MASK WITH INVERSE FLAG
FDC6: FDC7: FDC7: FDC7: FDC8: FDC8: FDC7: FDD1: FDD3: FDD4: FDD6: FDD6: FDD6: FDD7: FDD7: FDD2: FDE5: FDE5: FDE5: FDE5: FDE5: FDE5: FDE5: FDE7: FDE5: FDE5: FDE7: FDE5: FDE7: FDE5: FDE7: FDE5: FDE7:	4A 90 4A 4A 45 90 49 65 48 48 4A 4A 4A 4A 4A 20 68 29 09 90 69 60 25 84	EA 3E 02 FF 3C BD ED ED E5 0F B0 BA 02 06 6 36 02 32 35	FD FD	ADD PRBYTE PRHEX PRHEXZ COUT COUT1 COUTZ	LSR BCC LSR LDA BCC EOR ADC PHA LDA JSR PLA LSR LSR LSR LSR LSR LSR CMP BCC ADC JMP BCC ADC STY	A XAM A A2L ADD #\$FF A1L #\$BD COUT A A A A PRHEXZ #\$0F #\$B0 #\$BA COUT #\$06 (CSNL) #\$06 (CSNL) #\$A0 COUTZ INVFLG YSAV1	DETERMINE IF MON MODE IS XAM ADD, OR SUB SUB: FORM 2'S COMPLEMENT PRINT '=', THEN RESULT PRINT BYTE AS 2 HEX DIGITS, DESTROYS A-REG PRINT HEX DIG IN A-REG LSB'S VECTOR TO USER OUTPUT ROUTINE DON'T OUTPUT CTRL'S INVERSE MASK WITH INVERSE FLAG SAV Y-REG
FDC6: FDC7: FDC7: FDC7: FDC9: FDC4: FDC7: FDD1: FDD4: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD2: FDD2: FDD2: FDE9:	4A 90 4A 45 90 65 48 49 20 68 48 4A 4A 20 68 29 09 65 68 48 4A 20 68 29 09 65 68 48 4A 4A 20 68 29 09 25 84 48 49 49 49 49 49 49 49 49 49 49 49 49 49	EA 3E 02 FF 3C BD ED E5 0F B0 BA 02 06 36 002 32 35	FD FD	ADD PRBYTE PRHEX PRHEXZ COUT COUT1 COUT2	LSR BCC LSR LSR LDA BCC EOR ADC PHA LDA JSR PLA LSR LSR LSR LSR LSR LSR CMP BCC ADC JMP CMP BCC ADC STY PHA	A XAM A A ADD #\$FF AlL #\$BD COUT A A A PRHEXZ #\$0F #\$B0 #\$BA COUT #\$06 (CSWL) #\$A0 COUTZ INVFLG YSAV1	DETERMINE IF MON MODE IS XAM ADD, OR SUB SUB: FORM 2'S COMPLEMENT PRINT '=', THEN RESULT PRINT BYTE AS 2 HEX DIGITS, DESTROYS A-REG PRINT HEX DIG IN A-REG LSB'S VECTOR TO USER OUTPUT ROUTINE DON'T OUTPUT CTRL'S INVERSE MASK WITH INVERSE FLAG SAV Y-REG SAV A-REG
FDC6: FDC7: FDC7: FDC7: FDC9: FDC9: FDC1: FDD1: FDD4: FDD4: FDD9: FDD4: FDD9: FDD4: FDD5: FDD5: FDD5: FDD5: FDE7: FDE9:	4A 90 4A 45 90 49 65 48 49 20 68 48 4A 4A 4A 4A 20 68 29 09 65 69 60 69 60 69 90 62 25 84 48 20	EA 3E 02 FF 3C BD ED ED E5 0F B0 BA 02 06 36 A0 02 235 FD	FD FD	ADD PRBYTE PRHEX PRHEXZ COUT COUT1 COUT2	LSR BCC LSR LSR LSR ECC EOR ADC PHA LDA JSR PLA LSR LSR LSR LSR LSR LSR CMP BCC CMP BCC ADC JMP CMP BCC AND STY PHA JSR	A XAM A A A2L ADD #\$FF A1L #\$BD COUT A A A PRHEXZ #\$0F #\$B0 #\$BA COUT #\$06 (CSWL) #\$A0 COUTZ INVFLG YSAV1 VIDOUT	DETERMINE IF MON MODE IS XAM ADD, OR SUB SUE: FORM 2'S COMPLEMENT PRINT '=', THEN RESULT PRINT BYTE AS 2 HEX DIGITS, DESTROYS A-REG PRINT HEX DIG IN A-REG LSB'S VECTOR TO USER OUTPUT ROUTINE DON'T OUTPUT CTRL'S INVERSE MASK WITH INVERSE FLAG SAV Y-REG SAV A-REG OUTPUT A-REG AS ASCII
FDC6: FDC7: FDC7: FDC7: FDC9: FDC4: FDC1: FDD1: FDD4: FDD4: FDD4: FDD6: FDD9: FDD7: FDD7: FDD7: FDD7: FD25: FD29:	4A 90 4A 4D 49 90 49 65 48 49 20 68 48 4A 4A 4A 4A 20 68 29 90 69 69 69 25 4 8 20 90 69 69 69 60 25 8 8 8 8 8 90 69 69 69 69 69 69 69 69 69 69 69 69 69	EA 3E 3C FF 3C BD ED ED E5 0F B0 8A 02 36 36 36 36 36 36 75 FD	FD FD 00	ADD PRBYTE PRHEX PRHEXZ COUT COUT1 COUTZ	LSR BCC LSR LSR LDA BCC EOR ADC PHA LDA JSR PHA LSR LSR LSR LSR LSR LSR LSR AND ORA CMP BCC ADC JMP BCC ADC STY PHA	A XAM A A2L ADD #\$FF AlL #\$BD COUT A A A PRHEXZ #\$0F #\$B0 #\$B0 #\$B0 #\$B0 (CSWL) #\$A0 COUT Z INVFLG YSAV1 VIDOUT	DETERMINE IF MON MODE IS XAM ADD, OR SUB SUB: FORM 2'S COMPLEMENT PRINT '=', THEN RESULT PRINT BYTE AS 2 HEX DIGITS, DESTROYS A-REG PRINT HEX DIG IN A-REG LSB'S VECTOR TO USER OUTPUT ROUTINE DON'T OUTPUT CTRL'S INVERSE MASK WITH INVERSE FLAG SAV A-REG OUTPUT A-REG AS ASCII RESTORE A-REG
FDC6: FDC7: FDC7: FDC7: FDC9: FDC4: FDC5: FDD1: FDD3: FDD4: FDD4: FDD6: FDD6: FDD5: FDD5: FDD5: FDD5: FDE7: FDE9:	4A 90 4A 4A5 90 49 65 48 49 65 48 49 65 48 49 68 48 4A 4A 4A 4A 4A 20 68 29 09 60 69 90 25 84 48 20 68 20 90 25 84 48 20 20 20 20 20 20 20 20 20 20 20 20 20	EA 3E 02 FF 3C BD ED E5 0F B0 BA 02 32 35 FD 35	FD FD 00	ADD PRBYTE PRHEX PRHEXZ COUT COUT1 COUTZ	LSR BCC LSR LDA BCC EOR ADC PHA LDA JSR PLA LDA JSR PLA LSR LSR LSR LSR LSR LSR CMP BCC ADC DRA JMP BCC ADC STY PLA LDY RCC ADC LSR LSR LSR LSR LSR LDA BCC EOR ADC PHA LSR LSR LDA PLA LSR LSR LDA PLA LSR LSR LDA PLA LSR LSR LDA PLA LSR LSR LDA PLA LSR LDA PLA LSR LSR LDA PLA LSR LDA PLA LSR LSR LDA PLA LSR LDA PLA LSR LSR LDA PLA LSR LSR LDA PLA LSR LSR LDA PLA LSR LSR LSR LDA PLA LSR LSR LSR LSR LSR LSR LSR LDA PLA LSR LSR LSR LSR LSR LSR LSR LSR LSR LSR	A XAM A A A2L ADD #\$FF A1L #\$BD COUT A A A PRHEXZ #\$0F #\$B0 #\$BA COUT #\$06 (CSWL) #\$A0 COUTZ INVFLG YSAV1 VIDOUT	DETERMINE IF MON MODE IS XAM ADD, OR SUB SUB: FORM 2'S COMPLEMENT PRINT '=', THEN RESULT PRINT BYTE AS 2 HEX DIGITS, DESTROYS A-REG PRINT HEX DIG IN A-REG LSB'S VECTOR TO USER OUTPUT ROUTINE DON'T OUTPUT CTRL'S INVERSE MASK WITH INVERSE FLAG SAV Y-REG SAV A-REG OUTPUT A-REG AS ASCII RESTORE A-REG AND Y-REG
FDC6: FDC7: FDC7: FDC9: FDC9: FDC4: FDC9: FDD1: FDD4: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FD29:	4A 90 4A 4A 590 49 65 48 48 4A 4A 4A 4A 4A 4A 4A 4A 4A 20 68 29 09 25 84 48 20 69 25 84 48 20 69 69 60 25 84 48 60 69 60 25 84 60 60 60 60 60 60 60 60 60 60 60 60 60	EA 3E 02 FF 3C ED ED ED ED 0F BA 02 06 36 A00 232 35 FD 35	FD FD FB	ADD PRBYTE PRHEX PRHEXZ COUT COUT1 COUT2	LSR BCC LSR LSR LDA BCC EOR ADC PHA LDA PHA LSR LSR LSR LSR LSR LSR LSR LSR AND ORA AND ORA ADC PHA LSR LSR LSR LSR LSR LSR LSR LSR LSR LSR	A XAM A A ADD #\$FF AlL #\$BD COUT A A A PRHEXZ #\$0F #\$B0 #\$BA COUT #\$06 (CSWL) #\$A0 COUTZ INVFLG YSAV1 VIDOUT	DETERMINE IF MON MODE IS XAM ADD, OR SUB SUB: FORM 2'S COMPLEMENT PRINT '=', THEN RESULT PRINT BYTE AS 2 HEX DIGITS, DESTROYS A-REG PRINT HEX DIG IN A-REG LSB'S VECTOR TO USER OUTPUT ROUTINE DON'T OUTPUT CTRL'S INVERSE MASK WITH INVERSE FLAG SAV A-REG OUTPUT A-REG AS ASCII RESTORE A-REG
FDC6: FDC7: FDC7: FDC9: FDC9: FDC4: FDC9: FDD1: FDD4: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FD29:	4A 90 4A A5 90 49 65 48 20 68 48 4A 4A 20 68 29 09 90 67 C9 90 67 C9 90 25 84 48 20 68 20 69 C9 90 25 84 40 20 69 C9 20 5 84 20 69 60 5 20 60 60 60 60 60 60 60 60 60 60 60 60 60	EA 3E 02 FFF 3C BD ED ED E5 0F 0BA 02 06 36 A00 232 35 FD 35 34	FD FD FB	ADD PRBYTE PRHEX PRHEXZ COUT COUT1 COUT2 BL1	LSR BCC LSR LSR LSR LDA BCC EOR ADC PHA LDA JSR PLA LSR LSR LSR LSR LSR LSR LSR LSR AND ORA AND ORA SC MP BCC ADC JSR PLA LSR LSR LSR LSR LSR LSR LSR LSR LSR LSR	A XAM A A A2L ADD #\$FF A1L #\$BD COUT A A A PRHEXZ #\$0F #\$B0 #\$BA COUT #\$06 (CSWL) #\$06 (CSWL) #\$A0 COUTZ INVFLG YSAV1 YSAV1	DETERMINE IF MON MODE IS XAM ADD, OR SUB SUB: FORM 2'S COMPLEMENT PRINT '=', THEN RESULT PRINT BYTE AS 2 HEX DIGITS, DESTROYS A-REG PRINT HEX DIG IN A-REG LSB'S VECTOR TO USER OUTPUT ROUTINE DON'T OUTPUT CTRL'S INVERSE MASK WITH INVERSE FLAG SAV Y-REG SAV A-REG OUTPUT A-REG AS ASCII RESTORE A-REG AND Y-REG
FDC6: FDC7: FDC7: FDC9: FDC9: FDC4: FDC9: FDD1: FDD4: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FD29:	4A 90 4A A5 90 49 65 48 20 68 48 4A 4A 20 68 29 09 90 67 C9 90 67 C9 90 25 84 48 20 68 20 69 C9 90 25 84 40 20 69 C9 20 5 84 20 69 60 5 20 60 60 60 60 60 60 60 60 60 60 60 60 60	EA 3E2 FFF3C BD ED E5 0FF B0 BA 02 32 35 FD 35 34 9F	FD FD 00	ADD PRBYTE PRHEX PRHEXZ COUT COUTZ BL1	LSR BCC LSR LDA BCC EOR ADC PHA LDA PLA LDA PLA LDA LDA LDA LDA LDA LDA DFLA LSR LSR LSR LSR LSR LSR AND CMP BCC ADC ADC LSR LSR LSR LSR LDA BCC EOR ADC PLA BCC EOR ADC PLA BCC EOR ADC PLA BCC EOR ADC PLA BCC EOR DC PLA BCC CMP BC	A XAM A A ADD #\$FF AlL #\$BD COUT A A A PRHEXZ #\$0F #\$B0 #\$BA COUT #\$06 (CSWL) #\$A0 COUTZ INVFLG YSAV1 VIDOUT	DETERMINE IF MON MODE IS XAM ADD, OR SUB SUB: FORM 2'S COMPLEMENT PRINT '=', THEN RESULT PRINT BYTE AS 2 HEX DIGITS, DESTROYS A-REG PRINT HEX DIG IN A-REG LSB'S VECTOR TO USER OUTPUT ROUTINE DON'T OUTPUT CTRL'S INVERSE MASK WITH INVERSE FLAG SAV Y-REG SAV A-REG OUTPUT A-REG AS ASCII RESTORE A-REG AND Y-REG
FDC6: FDC7: FDC7: FDC9: FDC9: FDC4: FDC9: FDD1: FDD4: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FD29:	4A 90 4A A5 90 49 65 48 20 68 48 4A 4A 20 68 48 20 68 4A 4A 20 69 90 69 60 C9 90 53 48 20 69 69 60 C9 90 53 48 20 69 65 54 60 69 60 69 60 69 60 69 60 69 60 60 60 60 60 60 60 60 60 60 60 60 60	EA 3E2 FFF3C BD ED E5 0FF B0 BA 02 32 35 FD 35 34 9F	FD FD 00	ADD PRBYTE PRHEX PRHEXZ COUT COUT1 COUT2 BL1 BLANK	LSR LSR LSR LSR LSR LDA BCC EOR ADC PHA LDA JSR PHA LSR LSR LSR LSR LSR LSR LSR LSR AND ORA ADC DEC ADC PHA LSR LSR LSR LSR LSR LSR LSR LSR LSR LSR	A XAM A A A2L ADD #\$FF AlL #\$BD COUT A A A PRHEXZ #\$0F #\$B0 #\$B0 #\$B0 #\$B0 (CSWL) #\$A0 COUTZ INVFLG YSAV1 YSAV1 YSAV	DETERMINE IF MON MODE IS XAM ADD, OR SUB SUB: FORM 2'S COMPLEMENT PRINT '=', THEN RESULT PRINT BYTE AS 2 HEX DIGITS, DESTROYS A-REG PRINT HEX DIG IN A-REG LSB'S VECTOR TO USER OUTPUT ROUTINE DON'T OUTPUT CTRL'S INVERSE MASK WITH INVERSE FLAG SAV Y-REG SAV A-REG OUTPUT A-REG AS ASCII RESTORE A-REG AND Y-REG
FDC6: FDC7: FDC7: FDC7: FDC9: FDC9: FDC1: FDD1: FDD4: FDD4: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD9: FDD0: FDD0: FDD0: FDD0: FDD0: FDD0: FD00:	4A 90 4A A5 90 49 65 8 48 4A 4A 20 68 8 48 4A 4A 20 68 8 48 4A 4A 20 68 90 90 67 90 25 84 8 20 8 25 84 8 20 67 90 25 84 8 20 67 90 25 84 8 20 67 20 67 20 20 20 20 20 20 20 20 20 20 20 20 20	EA 3E2 FFF3C BD ED ED E5 0F B0 BA 02 06 36 32 35 FD 35 34 9F	FD FD FB	ADD PRBYTE PRHEX PRHEXZ COUT COUT1 COUT2 BL1 BLANK	LSR LSR LSR LSR LSR LDA BCC EOR ADC PHA LDA JSR PHA LSR LSR LSR LSR LSR LSR LSR LSR AND ORA ADC DEC ADC PHA LSR LSR LSR LSR LSR LSR LSR LSR LSR LSR	A XAM A A A2L ADD #\$FF A1L #\$BD COUT A A A A PRHEXZ #\$0F #\$B0 #\$BA COUT #\$06 (CSWL) #\$A0 COUTZ INVFLG YSAV1 YSAV1 YSAV	DETERMINE IF MON MODE IS XAM ADD, OR SUB SUB: FORM 2'S COMPLEMENT PRINT '=', THEN RESULT PRINT BYTE AS 2 HEX DIGITS, DESTROYS A-REG PRINT HEX DIG IN A-REG LSB'S VECTOR TO USER OUTPUT ROUTINE DON'T OUTPUT CTRL'S INVERSE MASK WITH INVERSE FLAG SAV A-REG OUTPUT A-REG AS ASCII RESTORE A-REG AND Y-REG THEN RETURN
FDC6: FDC7: FDC7: FDC7: FDC9: FDC4: FDC1: FDD1: FDD4: FDD4: FDD4: FDD4: FDD9: FDD4: FDD7: FDD7: FDD7: FDD7: FD29:	4A 900 4AA A5 900 499 658 489 200 688 484 4A 4A 200 688 4A 4A 200 689 090 690 690 690 255 844 480 689 209 900 690 690 255 844 600 C600 C600 C600 C600 C600 C600 C60	EA 3E2 FFF3C BD ED ED E5 0FF B0 BA 02 32 35 FD 35 34 9F 16	FD FD FB	ADD PRBYTE PRHEX PRHEXZ COUT COUT1 COUT2 BL1 BLANK	LSR BCC LSR LSR LDA BCC EOR ADC PHA LDA JSR PLA LSR LSR LSR LSR LSR LSR LSR LSR AND ORA CMP BCC ADC JMP BCC ADC DEC ADC PHA LSR LSR LSR LSR LSR LSR LSR LSR LSR LSR	A XAM A A A2L ADD #\$FF A1L #\$BD COUT A A A A PRHEXZ #\$0F #\$B0 (CSWL) #\$BA COUT #\$06 (CSWL) #\$BA COUTZ INVFLG YSAV1 YSAV1 YSAV XAM8 SETMDZ #\$BA	DETERMINE IF MON MODE IS XAM ADD, OR SUB SUB: FORM 2'S COMPLEMENT PRINT '=', THEN RESULT PRINT BYTE AS 2 HEX DIGITS, DESTROYS A-REG PRINT HEX DIG IN A-REG LSB'S VECTOR TO USER OUTPUT ROUTINE DON'T OUTPUT CTRL'S INVERSE MASK WITH INVERSE FLAG SAV A-REG OUTPUT A-REG AS ASCII RESTORE A-REG AND Y-REG THEN RETURN BLANK TO MON
FDC6: FDC7: FDC7: FDC7: FDC9: FDC4: FDC5: FDC4: FDD4: FDD4: FDD4: FDD4: FDD4: FDD5: FDD5: FDD5: FD25: FD25: FD25: FD25: FD27: FD29:	4A 900 4AA A5 900 499 655 489 200 688 484 4A 4A 200 688 209 609 609 609 2584 488 209 900 609 609 2584 488 209 609 609 609 609 609 2584 800 200 609 609 609 609 609 609 609 609 609 6	EA 3E2 FFF3C BD ED E5 0F0 BA 02 32 35 FD 35 34 9F 16 BA	FD FD 000 FB	ADD PRBYTE PRHEX PRHEXZ COUT COUT1 COUT2 BL1 BLANK	LSR BCC LSR LSR LDA BCC EOR ADC PHA LDA JSR PLA LSR LSR LSR LSR LSR LSR LSR LSR AND ORA CMP BCC ADC JMP BCC ADC DEC ADC PHA LSR LSR LSR LSR LSR LSR LSR LSR LSR LSR	A XAM A A A2L ADD #\$FF A1L #\$BD COUT A A A A PRHEXZ #\$0F #\$B0 #\$BA COUT #\$06 (CSWL) #\$A0 COUTZ INVFLG YSAV1 VIDOUT YSAV1 YSAV1 YSAV XAM8 SETMDZ #\$BA	DETERMINE IF MON MODE IS XAM ADD, OR SUB SUB: FORM 2'S COMPLEMENT PRINT '=', THEN RESULT PRINT BYTE AS 2 HEX DIGITS, DESTROYS A-REG PRINT HEX DIG IN A-REG LSB'S VECTOR TO USER OUTPUT ROUTINE DON'T OUTPUT CTRL'S INVERSE MASK WITH INVERSE FLAG SAV A-REG OUTPUT A-REG AS ASCII RESTORE A-REG AND Y-REG THEN RETURN BLANK TO MON AFTER BLANK DATA STORE MODE?
FDC6: FDC7: FDC7: FDC7: FDC9: FDC9: FDC7: FDD1: FDD4: FDD9: FD29:	4A 900 4AA A5 900 499 658 488 420 688 48A 4AA 4AA 200 688 48A 4AA 209 099 60 209 900 255 48 200 60 209 900 255 48 200 60 209 200 200 200 200 200 200 200 200 20	EA 3E202 FFF3C BD ED E5 0F0 B00 BA206 360 2235 FD 35 34 9F 16 BAB	FD FD FB	ADD PRBYTE PRHEX PRHEXZ COUT COUT1 COUT2 BL1 BLANK	LSR BCC LSR LSR LSR LSR BCC EOR ADC PHA LSR LSR LSR LSR LSR LSR LSR LSR LSR LSR	A XAM A A A2L ADD #\$FF A1L #\$BD COUT A A A PRHEXZ #\$0F #\$B0 #\$BA COUT #\$06 (CSWL) #\$06 (CSWL) #\$A0 COUTZ INVFLG YSAV1 VIDOUT YSAV1 YSAV XAM8 SETMDZ #\$BA XAMPM	DETERMINE IF MON MODE IS XAM ADD, OR SUB SUB: FORM 2'S COMPLEMENT PRINT '=', THEN RESULT PRINT BYTE AS 2 HEX DIGITS, DESTROYS A-REG PRINT HEX DIG IN A-REG LSB'S VECTOR TO USER OUTPUT ROUTINE DON'T OUTPUT CTRL'S INVERSE MASK WITH INVERSE FLAG SAV A-REG OUTPUT A-REG AS ASCII RESTORE A-REG AND Y-REG THEN RETURN BLANK TO MON AFTER BLANK DATA STORE MODE?
FDC6: FDC7: FDC7: FDC7: FDC9: FDC9: FDC1: FDD1: FDD4: FDD9: FD29:	4A 900 4AA A5 900 499 658 480 400 688 480 400 688 480 400 600 600 600 255 848 200 600 600 255 848 200 600 600 255 848 600 600 600 600 600 600 600 600 600 60	EA 3E2 FFF3C ED ED E5 0FFB0 BA 022 325 FD 35 34 9F 16 BBB 31	FD FD FB	ADD PRBYTE PRHEX PRHEXZ COUT COUT1 COUT2 BL1 BLANK STOR	LSR BCC LSR LSR LSR BCC EOR ADC PHA LSR PLA LSR LSR LSR LSR LSR LSR LSR LSR LSR LSR	A XAM A A A2L ADD #\$FF A1L #\$BD COUT A A A PRHEXZ #\$0F #\$B0 #\$BA COUT #\$06 (CSWL) #\$06 (CSWL) #\$A0 COUTZ INVFLG YSAV1 VIDOUT YSAV1 YSAV XAM8 SETMDZ #\$BA XAMPM MODE	DETERMINE IF MON MODE IS XAM ADD, OR SUB SUB: FORM 2'S COMPLEMENT PRINT '=', THEN RESULT PRINT BYTE AS 2 HEX DIGITS, DESTROYS A-REG PRINT HEX DIG IN A-REG LSB'S VECTOR TO USER OUTPUT ROUTINE DON'T OUTPUT CTRL'S INVERSE MASK WITH INVERSE FLAG SAV Y-REG SAV A-REG OUTPUT A-REG AS ASCII RESTORE A-REG AND Y-REG THEN RETURN BLANK TO MON AFTER BLANK DATA STORE MODE?
FDC6: FDC7: FDC7: FDC7: FDC9: FDC9: FDC7: FDD1: FDD4: FDD9: FD29:	4A 900 4AA A5 900 499 658 480 400 688 480 400 688 480 400 600 600 600 255 848 200 600 600 255 848 200 600 600 255 848 600 600 600 600 600 600 600 600 600 60	EA 3E2 FFF3C ED ED E5 0FFB0 BA 022 325 FD 35 34 9F 16 BBB 31	FD FD FB	ADD PRBYTE PRHEX PRHEXZ COUT COUT1 COUT2 BL1 BLANK STOR	LSR BCC LSR LSR LSR LSR BCC EOR ADC PHA LSR LSR LSR LSR LSR LSR LSR LSR LSR LSR	A XAM A A A2L ADD #\$FF A1L #\$BD COUT A A A PRHEXZ #\$0F #\$B0 #\$BA COUT #\$06 (CSWL) #\$06 (CSWL) #\$A0 COUTZ INVFLG YSAV1 YSAV1 YSAV1 YSAV1 YSAV XAM8 SETMDZ #\$BA XAMPM MODE A2L	DETERMINE IF MON MODE IS XAM ADD, OR SUB SUB: FORM 2'S COMPLEMENT PRINT '=', THEN RESULT PRINT BYTE AS 2 HEX DIGITS, DESTROYS A-REG PRINT HEX DIG IN A-REG LSB'S VECTOR TO USER OUTPUT ROUTINE DON'T OUTPUT CTRL'S INVERSE MASK WITH INVERSE FLAG SAV A-REG OUTPUT A-REG AS ASCII RESTORE A-REG AND Y-REG THEN RETURN BLANK TO MON AFTER BLANK DATA STORE MODE?

FEOF:							STORE AS LOW BYTE AS (A3)
FE11: FE13:						A3L RTS5	INCR A3, RETURN
FE15:					INC		·
FE17:				RTS5	RTS		SAVE CONVERTED ':', '+',
FE18: FE1A:	A4 B9	34 FF	01	SETMODE	LDY	YSAV TN-1.Y	SAVE CONVERTED ':', '+', '-', '.' AS MODE.
FE1D:	85	31		SETMDZ	STA	MODE	,
FE1F:	60				RTS		
FE20: FE22:				LT LT2	LDX LDA	#\$01 A2L,X	COPY A2 (2 BYTES) TO
FE24:					STA	A4L,X	A4 AND A5
FE26:						A5L,X	
FE28: FE29:					DEX BPL	ፒ.ሞ2	
FE2B:					RTS	512	
				MOVE	LDA	(A1L),Y	MOVE (A1 TO A2) TO (A4)
FE2E: FE30:					STA	(A4L),Y NXTA4	(A4)
FE33:					BCC		
FE35:					RTS		
FE36:	B1 D1	3C		VFY	LDA	(A1L),Y	VERIFY (A1 TO A2) WITH (A4)
FE3A:					BEQ	VFYOK	(44)
						PRA1 (A1L),Y	
FE3F:							
FE41: FE44:					LDA	PRBYTE #\$A0	
FE46:					JSR	COUT	
FE49:					LDA JSR	#\$A8	
FE4B: FE4E:					JSR	COUT	
FE50:					JSR	(A4L),Y PRBYTE	
FE53:	A9	Α9			LDA	#\$A9	
FE55:	20	ED	FD	VFYOK	JSR	COUT	
FE5B:			FC		BCC		
FE5D:					RTS		
							MOVE A1 (2 BYTES) TO
FE61: FE63:				LIST2		#\$14	PC IF SPEC'D AND DISEMBLE 20 INSTRS
FE64:					JSR	INSTDSP	
FE67:							ADJUST PC EACH INSTR
FE6A: FE6C:						PCL	
FE6E:					STY PLA	РСП	
FE6F:	38				SEC		
FE70:							NEXT OF 20 INSTRS
FE72: FE74:					RTS	LIST2	
FE75:	8A			A1PC	TXA		IF USER SPEC'D ADR
FE76:						A1PCRTS	COPY FROM A1 TO PC
FE78: FE7A:				A1PCLP	LDA STA	A1L,X PCL,X	
FE7C:					DEX	102/11	
FE7D:						A1PCLP	
FE7F: FE80:				A1PCRTS		#\$3F	SET FOR INVERSE VID
FE82:					BNE	SETIFLG	VIA COUT1
FE84:				SETNORM	LDY	#\$FF INVFLG	SET FOR NORMAL VID
FE86: FE88:		32			STY RTS	INVFLG	
FE89:		00				#\$00	SIMULATE PORT #0 INPUT
FE8B:	85	3E		INPORT	STA	A2L	SPECIFIED (KEYIN ROUTINE)
FE8D:				INPRT	LDX	#KSWL #KEYIN	
FE8F: FE91:						IOPRT	
FE93:				SETVID			SIMULATE PORT #0 OUTPUT
FE95:				OUTPORT			SPECIFIED (COUT1 ROUTINE)
FE97: FE99:					LDX	#CSWL #COUT1	
FE9B:					LDA		SET RAM IN/OUT VECTORS
FE9D:						#\$0F	
FE9F: FEA1:						IOPRT1 #IOADR/256	
FEA3:					LDY		
FEA5:					BEQ	IOPRT2	
FEA7: FEA9:						#COUT1/256 LOC0,X	
FEAB:				1011112		LOCU,X LOC1,X	
FEAD:					RTS		
FEAE: FEAF:					NOP NOP		
			ΕO	XBASIC		BASIC	TO BASIC WITH SCRATCH
FEB3:	4C	03	ΕO	BASCONT	JMP	BASIC2	CONTINUE BASIC

FEB6:							
	20	75	FE	GO	JSR	A1PC	ADR TO PC IF SPEC'D
FEB9:	20	3F	FF		JSR	RESTORE	RESTORE META REGS GO TO USER SUBR
FEBC:					TMD	(DCI)	CO TO LICER CURR
FEBC:	00	5A			UMP	(FCL)	GO IO USER SOBR
FEBF:	4C	D7	FΑ	REGZ	JMP	REGDSP	TO REG DISPLAY
FEC2:	C6	34		TRACE	DEC	YSAV	
FEC4:	20	75	FE	STEPZ	JSR	Alpc	ADR TO PC IF SPEC'D
FEC7:					TMP	STED	TAKE ONE STEP
				USR	TMD		TO USR SUBR AT USRADR
FECA:	4C	го	03	USR	UMP	USRADR	IO USK SUBR AI USRADR
FECD:	A9	40		WRITE	LDA	#\$40	WRITE 10-SEC HEADER
FECF:	20	C9	FC		JSR	HEADR	WRITE 10-SEC HEADER
FED2:	A0	27			LDY	#\$27	
				WR1	TUDX	#\$00	
FED6:	11	20			FOR	(A1L,X)	
						(AIL,A)	
FED8:					PHA		
FED9:	A1	3C			LDA	(A1L,X) WRBYTE	
FEDB:	20	ED	FE		JSR	WRBYTE	
FEDE:	20	ΒA	FC		JSR	NXTA1	
FEE1:					T'DA	NXTA1 #\$1D	
FEE3:						11422	
					PLA		
FEE4:					BCC		
FEE6:	A0	22			LDY		
FEE8:	20	ED	FE		JSR	WRBYTE	
FEEB:	F0	4D			BEQ		
FEED:	∆2	10					
FEEF:				WRBYTE WRBYT2	ACT	πφ±0 λ	
						A	
FEF0:						WRBIT	
FEF3:	D0	FA			BNE	WRBYT2	
FEF5:	60				RTS		
FEF6.	20	0.0	FE	CRMON	JISR	BL1	HANDLE A CR AS BLANK
FEF9:	60			0101011	PLA	221	THEN POP STACK
FEFA:					PLA		AND RTN TO MON
FEFB:					BNE	MONZ	
FEFD:	20	FA	FC	READ	JSR	RD2BIT	FIND TAPEIN EDGE
FF00:	A9	16			LDA	#\$16	
FF02:	20	C9	FC		JISR	HEADR	DELAY 3.5 SECONDS
FF05:					CTTA	CURCIM	INIT CHKSUM=\$FF
					SIA	CHKSUM	INII CHRSOM=SFF
FF07:					JSR	RD2B11	FIND TAPEIN EDGE
FF0A:	A0	24		RD2	LDY	#\$24	FIND TAPEIN EDGE LOOK FOR SYNC BIT
FF0C:	20	FD	FC		JSR	RDBIT RD2	(SHORT 0)
FFOF:	В0	F9			BCS	RD2	LOOP UNTIL FOUND
FF11:					TCP	PDBTT	SKIP SECOND SYNC H-CYCLE
					TDI	KDD11 Udan	
FF14:					цр х		INDEX FOR 0/1 TEST
FF16:	20	EC	FC	RD3	JSR	RDBYTE	READ A BYTE
FF19:	81	3C			STA	(A1L,X) CHKSUM	STORE AT (A1)
FF1B:	45	2E			EOR	CHKSUM	
FF1D:	85	2E					UPDATE RUNNING CHKSUM
FF1F:							INC A1, COMPARE TO A2
					TDV	NXTA1 #\$35	COMPENSATE 0/1 INDEX
		35			гдт	#\$35	
FF22:							
FF24:	90				BCC	RD3	LOOP UNTIL DONE
	90		FC		BCC JSR	RD3	LOOP UNTIL DONE READ CHKSUM BYTE
FF24:	90 20	EC			JSR	RD3 RDBYTE	LOOP UNTIL DONE READ CHKSUM BYTE
FF24: FF26: FF29:	90 20 C5	EC 2E			JSR CMP	RD3 RDBYTE CHKSUM	READ CHKSUM BYTE
FF24: FF26: FF29: FF2B:	90 20 C5 F0	EC 2E 0D			JSR CMP BEQ	RD3 RDBYTE CHKSUM BELL	LOOP UNTIL DONE READ CHKSUM BYTE GOOD, SOUND BELL AND RETURN
FF24: FF26: FF29: FF2B: FF2D:	90 20 C5 F0 A9	EC 2E 0D C5		PRERR	JSR CMP BEQ LDA	RD3 RDBYTE CHKSUM BELL #\$C5	READ CHKSUM BYTE GOOD, SOUND BELL AND RETURN
FF24: FF26: FF29: FF2B: FF2D: FF2D:	90 20 C5 F0 A9 20	EC 2E 0D C5 ED	FD	PRERR	JSR CMP BEQ LDA JSR	RD3 RDBYTE CHKSUM BELL #\$C5 COUT	READ CHKSUM BYTE
FF24: FF26: FF29: FF2B: FF2D:	90 20 C5 F0 A9 20	EC 2E 0D C5 ED	FD	PRERR	JSR CMP BEQ LDA JSR	RD3 RDBYTE CHKSUM BELL #\$C5	READ CHKSUM BYTE GOOD, SOUND BELL AND RETURN
FF24: FF26: FF29: FF2B: FF2D: FF2D:	90 20 C5 F0 A9 20 A9	EC 2E 0D C5 ED D2	FD	PRERR	JSR CMP BEQ LDA JSR LDA	RD3 RDBYTE CHKSUM BELL #\$C5 COUT	READ CHKSUM BYTE GOOD, SOUND BELL AND RETURN
FF24: FF26: FF29: FF2B: FF2D: FF2F: FF32: FF34:	90 20 C5 F0 A9 20 A9 20	EC 2E 0D C5 ED D2 ED	FD FD	PRERR	JSR CMP BEQ LDA JSR LDA JSR	RD3 RDBYTE CHKSUM BELL #\$C5 COUT #\$D2 COUT	READ CHKSUM BYTE GOOD, SOUND BELL AND RETURN
FF24: FF26: FF29: FF2B: FF2D: FF2F: FF32: FF34: FF37:	90 20 C5 F0 A9 20 A9 20 20	EC 2E 0D C5 ED D2 ED ED	FD FD FD	PRERR	JSR CMP BEQ LDA JSR LDA JSR JSR	RD3 RDBYTE CHKSUM BELL #\$C5 COUT #\$D2 COUT COUT	READ CHKSUM BYTE GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL
FF24: FF29: FF29: FF2D: FF2F: FF32: FF34: FF37: FF3A:	90 20 C5 F0 20 20 20 20 A9	EC 2E 0D C5 ED 22 ED 87	FD FD FD	PRERR BELL	JSR CMP BEQ LDA JSR LDA JSR JSR LDA	RD3 RDBYTE CHKSUM BELL #\$C5 COUT #\$D2 COUT COUT #\$87	READ CHKSUM BYTE GOOD, SOUND BELL AND RETURN
FF24: FF29: FF28: FF2D: FF27: FF32: FF34: FF37: FF3A: FF32:	90 20 C5 F0 A9 20 A9 20 20 A9 4C	EC 2E 0D C5 ED 22 ED ED 87 ED	FD FD FD FD	PRERR BELL	JSR CMP BEQ LDA JSR LDA JSR LDA LDA JMP	RD3 RDBYTE CHKSUM BELL #\$C5 COUT #\$D2 COUT COUT #\$87 COUT	READ CHKSUM BYTE GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN
FF24: FF29: FF28: FF2D: FF27: FF32: FF34: FF37: FF3A: FF37: FF3F:	90 20 5 70 20 20 20 20 20 40 40	EC 2E 0D C5 ED 22 ED 87 ED 48	FD FD FD FD	PRERR BELL RESTORE	JSR CMP BEQ LDA JSR JSR JSR LDA JMP LDA	RD3 RDBYTE CHKSUM BELL #\$C5 COUT #\$D2 COUT COUT #\$87 COUT	READ CHKSUM BYTE GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS
FF24: FF29: FF28: FF2D: FF27: FF32: FF34: FF37: FF3A: FF32:	90 20 5 70 20 20 20 20 20 40 40	EC 2E 0D C5 ED 22 ED 87 ED 48	FD FD FD FD	PRERR BELL RESTORE	JSR CMP BEQ LDA JSR LDA JSR LDA LDA JMP	RD3 RDBYTE CHKSUM BELL #\$C5 COUT #\$D2 COUT COUT #\$87 COUT	READ CHKSUM BYTE GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN
FF24: FF29: FF28: FF2D: FF27: FF32: FF34: FF37: FF3A: FF37: FF37: FF37: FF37:	90 20 5 F0 20 20 20 20 20 40 45 48	EC 2E 0D C5 ED 22 ED 87 ED 48	FD FD FD FD	PRERR BELL RESTORE	JSR CMP BEQ LDA JSR LDA JSR LDA JMP LDA PHA	RD3 RDBYTE CHKSUM BELL #\$C5 COUT #\$D2 COUT COUT #\$87 COUT STATUS	READ CHKSUM BYTE GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS
FF24: FF20: FF29: FF20: FF20: FF32: FF34: FF34: FF37: FF37: FF34: FF34: FF41: FF42:	90 20 5 F0 20 20 20 20 20 40 45 48 45	EC 2E 0D C5 ED 22 ED 87 ED 48 45	FD FD FD FD	PRERR BELL RESTORE	JSR CMP BEQ LDA JSR LDA JSR LDA JMP LDA PHA LDA	RD3 RDBYTE CHKSUM BELL #\$C5 COUT #\$D2 COUT COUT #\$87 COUT STATUS ACC	READ CHKSUM BYTE GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS
FF24: FF29: FF29: FF2B: FF2D: FF2F: FF37: FF37: FF37: FF37: FF37: FF3F: FF41: FF42: FF44:	90 20 C5 F0 A9 20 A9 20 20 A9 4C A5 48 A5 A6	EC 2E 0D C5 ED 22 ED 87 ED 48 45 46	FD FD FD	PRERR BELL RESTORE RESTR1	JSR CMP BEQ LDA JSR LDA JSR LDA JMP LDA PHA LDA LDA	RD3 RDBYTE CHKSUM BELL #\$C5 COUT #\$D2 COUT COUT COUT \$TATUS ACC XREG	READ CHKSUM BYTE GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS
FF24: FF26: FF29: FF2B: FF2D: FF2D: FF32: FF34: FF37: FF34: FF37: FF3F: FF42: FF44: FF46:	90 20 C5 F0 A9 20 20 20 20 20 40 A9 4C A5 48 A5 A6 A4	EC 2E 0D C5 ED 22 ED 87 ED 48 45 46 47	FD FD FD	PRERR BELL RESTORE RESTR1	JSR CMP BEQ LDA JSR LDA JSR LDA JMP LDA LDA LDA LDA LDX LDY	RD3 RDBYTE CHKSUM BELL #\$C5 COUT #\$D2 COUT COUT COUT \$TATUS ACC XREG	READ CHKSUM BYTE GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS
FF24: FF29: FF29: FF29: FF20: FF34: FF34: FF37: FF37: FF37: FF34: FF34: FF41: FF44: FF44: FF48:	90 20 C5 F0 A9 20 A9 20 A9 4C A5 48 A5 A6 A4 28	EC 2E 0D C5 ED 22 ED 22 ED 87 ED 48 45 46 47	FD FD FD	PRERR BELL RESTORE RESTR1	JSR CMP BEQ LDA JSR LDA JSR LDA JSR LDA JMP LDA LDA LDA LDX LDY PLP	RD3 RDBYTE CHKSUM BELL #\$C5 COUT #\$D2 COUT COUT COUT \$TATUS ACC XREG	READ CHKSUM BYTE GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS
FF24: FF29: FF29: FF20: FF20: FF37: FF34: FF37: FF37: FF37: FF37: FF41: FF42: FF44: FF46: FF49:	90 20 C5 F0 A9 20 A9 20 A9 4C A5 48 A5 A6 A4 28 60	EC 2E 0D C5 ED 22 ED 22 ED 87 ED 48 45 46 47	FD FD FD	PRERR BELL RESTORE RESTR1	JSR CMP BEQ LDA JSR LDA JSR LDA JSR LDA PHA LDA LDA LDA LDY PLP RTS	RD3 RDBYTE CHKSUM BELL #\$C5 COUT #\$D2 COUT COUT #\$87 COUT STATUS ACC XREG YREG	READ CHKSUM BYTE GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS USED BY DEBUG SOFTWARE
FF24: FF29: FF29: FF29: FF20: FF34: FF34: FF37: FF37: FF37: FF34: FF34: FF41: FF44: FF44: FF48:	90 20 C5 F0 A9 20 A9 20 A9 4C A5 48 A5 A6 A4 28 60	EC 2E 0D C5 ED 22 ED 22 ED 87 ED 48 45 46 47	FD FD FD	PRERR BELL RESTORE RESTR1	JSR CMP BEQ LDA JSR LDA JSR LDA JSR LDA PHA LDA LDA LDA LDY PLP RTS	RD3 RDBYTE CHKSUM BELL #\$C5 COUT #\$D2 COUT COUT #\$87 COUT STATUS ACC XREG YREG	READ CHKSUM BYTE GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS USED BY DEBUG SOFTWARE
FF24: FF29: FF29: FF20: FF20: FF37: FF34: FF37: FF37: FF37: FF37: FF41: FF42: FF44: FF46: FF49:	90 20 C5 F0 A9 20 A9 20 A9 40 A5 48 A5 A6 A4 28 60 85	EC 2E 0D C5 ED 22 ED 48 45 46 47 45	FD FD FD	PRERR BELL RESTORE RESTR1	JSR CMP BEQ LDA JSR LDA JSR LDA JSR LDA PHA LDA LDA LDA LDY PLP RTS	RD3 RDBYTE CHKSUM BELL #\$C5 COUT #\$D2 COUT COUT #\$87 COUT STATUS ACC XREG YREG	READ CHKSUM BYTE GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS USED BY DEBUG SOFTWARE
FF24: FF20: FF20: FF20: FF20: FF32: FF34: FF37: FF37: FF37: FF37: FF37: FF41: FF44: FF46: FF46: FF49: FF44:	90 20 C5 F0 A9 20 A9 20 A9 40 A5 48 A5 A6 A4 28 60 85 86	EC 2E 0D C5 ED ED 2 ED 48 45 46 47 45 46	FD FD FD	PRERR BELL RESTORE RESTR1 SAVE SAV1	JSR CMP BEQ LDA JSR LDA JSR LDA JSR LDA PHA LDA LDA LDA LDY PLP RTS	RD3 RDBYTE CHKSUM BELL #\$C5 COUT #\$D2 COUT COUT #\$87 COUT STATUS ACC XREG YREG ACC XREG	READ CHKSUM BYTE GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS
FF24: FF29: FF29: FF20: FF27: FF37: FF37: FF37: FF37: FF37: FF41: FF44: FF44: FF48: FF49: FF44: FF44: FF44: FF44: FF44:	90 20 C5 F0 20 20 20 20 40 45 48 48 48 28 60 85 86 84	EC 2E 0D C5 ED ED ED 48 45 46 47 45 46 47	FD FD FD	PRERR BELL RESTORE RESTR1 SAVE SAV1	JSR CMP BEQ LDA JSR JSR JSR JDA JMP LDA LDA LDA LDA LDA LDA STA STX STY	RD3 RDBYTE CHKSUM BELL #\$C5 COUT #\$D2 COUT COUT #\$87 COUT STATUS ACC XREG YREG ACC XREG	READ CHKSUM BYTE GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS USED BY DEBUG SOFTWARE
FF24: FF29: FF29: FF20: FF20: FF34: FF37: FF34: FF37: FF37: FF37: FF41: FF42: FF44: FF46: FF46: FF48: FF49: FF42: FF42: FF42: FF42: FF42: FF42: FF42: FF42: FF42: FF42: FF42: FF42: FF42: FF42: FF42: FF42: FF42: FF42: FF32: FF34: FF44: FF45: FF44: FF45: FF44: FF45: FF44: FF45:	90 20 C5 F0 20 20 20 20 40 42 48 48 48 48 60 85 86 84 08	EC 2E 0D C5 ED ED ED 87 ED 48 46 47 45 46 47	FD FD FD	PRERR BELL RESTORE RESTR1 SAVE SAVE	JSR CMP BEQ LDA JSR LDA JSR LDA JMP LDA LDA LDA LDA LDY PLP RTS STA STY PHP	RD3 RDBYTE CHKSUM BELL #\$C5 COUT #\$D2 COUT COUT #\$87 COUT STATUS ACC XREG YREG ACC XREG	READ CHKSUM BYTE GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS USED BY DEBUG SOFTWARE
FF24: FF29: FF29: FF20: FF27: FF34: FF37: FF34: FF37: FF37: FF37: FF41: FF42: FF44: FF46: FF48: FF42: FF42: FF42: FF42: FF42: FF42: FF42: FF42: FF42: FF42: FF42: FF42: FF44:	90 20 C5 F0 A9 20 A9 20 A9 4C A5 48 A5 A6 A4 28 60 85 86 84 08 68	EC 2E 0D C5 ED ED ED 87 ED 48 45 46 47 45 46 47	FD FD FD	PRERR BELL RESTORE RESTR1 SAVE SAVE	JSR CMP BEQ JDA JSR LDA JSR LDA JMP LDA LDA LDA LDA LDX LDY PLP RTS STA STX STY PHP PLA	RD3 RDBYTE CHKSUM BELL #\$C5 COUT COUT COUT STATUS ACC XREG YREG ACC XREG YREG	READ CHKSUM BYTE GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS USED BY DEBUG SOFTWARE
FF24: FF29: FF29: FF20: FF20: FF37: FF37: FF37: FF37: FF37: FF41: FF42: FF44: FF44: FF46: FF44: FF42: FF42: FF42: FF42: FF42: FF42: FF42: FF42: FF42: FF42: FF42: FF42: FF44: FF45: FF55:	90 20 C5 F0 A9 20 A9 20 A9 4C A5 48 A5 A6 A4 28 60 85 86 84 08 85	EC 2E 0D C5 ED 2D ED 48 45 46 47 45 46 47 48	FD FD FD	PRERR BELL RESTORE RESTR1 SAVE SAVE	JSR CMP BEQ LDA JSR LDA JSR LDA JMP LDA LDA LDA LDA LDA LDA LDA STA STX STA STX STA STA	RD3 RDBYTE CHKSUM BELL #\$C5 COUT #\$D2 COUT COUT #\$87 COUT STATUS ACC XREG YREG ACC XREG	READ CHKSUM BYTE GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS USED BY DEBUG SOFTWARE
FF24: FF29: FF29: FF20: FF20: FF34: FF37: FF37: FF37: FF37: FF37: FF37: FF44: FF44: FF46: FF46: FF49: FF44: FF42: FF42: FF42: FF51: FF52: FF54:	90 20 C5 F0 A9 20 A9 20 A9 4C A5 48 A6 A4 28 60 85 86 85 BA	EC 2E 0D C5 ED 2ED 87 ED 48 45 46 47 45 46 47 48	FD FD FD	PRERR BELL RESTORE RESTR1 SAVE SAV1	JSR CMP BEQ LDA JSR LDA JSR LDA JMP LDA LDA LDA LDA LDA LDA LDA STA STX STY PLP STA STA STA STA STA STA	RD3 RDBYTE CHKSUM BELL #\$C5 COUT #\$87 COUT STATUS ACC XREG YREG ACC XREG YREG STATUS	READ CHKSUM BYTE GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS USED BY DEBUG SOFTWARE
FF24: FF29: FF29: FF20: FF20: FF37: FF37: FF37: FF37: FF37: FF41: FF42: FF44: FF44: FF46: FF44: FF42: FF42: FF42: FF42: FF42: FF42: FF42: FF42: FF42: FF42: FF42: FF42: FF44: FF45: FF55:	90 20 C5 F0 A9 20 A9 20 A9 4C A5 48 A6 A4 28 60 85 86 85 BA	EC 2E 0D C5 ED 2ED 87 ED 48 45 46 47 45 46 47 48	FD FD FD	PRERR BELL RESTORE RESTR1 SAVE SAV1	JSR CMP BEQ LDA JSR LDA JSR LDA JMP LDA LDA LDA LDA LDA LDA LDA STA STX STA STX STA STA	RD3 RDBYTE CHKSUM BELL #\$C5 COUT #\$87 COUT STATUS ACC XREG YREG ACC XREG YREG STATUS	READ CHKSUM BYTE GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS USED BY DEBUG SOFTWARE
FF24: FF29: FF29: FF20: FF20: FF34: FF37: FF37: FF37: FF37: FF37: FF37: FF44: FF44: FF46: FF46: FF49: FF44: FF42: FF42: FF42: FF51: FF52: FF54:	90 20 C5 F0 A9 20 20 A9 20 A9 4C A5 48 A6 A4 85 86 84 08 85 BA 86 85 BA	EC 2E 0D C5 ED 2ED 87 ED 48 45 46 47 45 46 47 48	FD FD FD	PRERR BELL RESTORE RESTR1 SAVE SAV1	JSR CMP BEQ LDA JSR LDA JSR LDA JMP LDA LDA LDA LDA LDA LDA LDA STA STX STY PLP STA STA STA STA STA STA	RD3 RDBYTE CHKSUM BELL #\$C5 COUT #\$87 COUT STATUS ACC XREG YREG ACC XREG YREG STATUS	READ CHKSUM BYTE GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS USED BY DEBUG SOFTWARE
FF24: FF22: FF29: FF29: FF20: FF34: FF37: FF37: FF37: FF37: FF41: FF42: FF44: FF44: FF44: FF44: FF44: FF44: FF45: FF51: FF52: FF55: FF55: FF57:	90 20 C5 F0 20 20 20 20 20 20 20 20 20 20 20 20 20	EC 2E 0D C5 ED 2ED ED 87 ED 48 45 46 47 45 46 47 48 49	FD FD FD	PRERR BELL RESTORE RESTR1 SAVE SAVE	JSR CMP BEQ LDA JSR LDA JSR LDA JSR LDA PHA LDA LDA LDA LDY PLP RTS STA STY PHP PLA STX STY CLD	RD3 RDBYTE CHKSUM BELL #\$C5 COUT #\$87 COUT STATUS ACC XREG YREG ACC XREG YREG STATUS	READ CHKSUM BYTE GOOD, SOUND BELL AND RETURN PRINT "ERR", THEN BELL OUTPUT BELL AND RETURN RESTORE 6502 REG CONTENTS USED BY DEBUG SOFTWARE
FF24: FF20: FF20: FF20: FF27: FF34: FF37: FF37: FF37: FF37: FF41: FF42: FF44: FF44: FF46: FF46: FF46: FF50: FF51: FF55: FF55: FF55: FF58:	90 20 C5 F0 20 20 20 20 20 20 20 20 20 20 20 20 20	EC 2E 0D C5 ED 2ED ED 87 ED 48 45 46 47 45 46 47 48 49	FD FD FD	PRERR BELL RESTORE RESTR1 SAVE SAVE	JSR CMP BEQ JSR LDA JSR LDA JSR LDA PHA LDA LDA LDA LDA LDA LDA STA STA STA STA STA STA STA STA STA ST	RD3 RDBYTE CHKSUM BELL #\$C5 COUT COUT COUT STATUS ACC XREG YREG XREG YREG STATUS SPNT	READ CHKSUM BYTE 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
FF24: FF29: FF29: FF29: FF27: FF34: FF37: FF37: FF37: FF37: FF41: FF42: FF44: FF44: FF44: FF44: FF44: FF45: FF51: FF55: FF55: FF55: FF55: FF55: FF55: FF55:	90 20 C5 F0 A9 20 A9 40 A5 48 A5 A6 A2 85 86 85 86 85 BA 8 BA 8	EC 2E 0D C5 ED 22 ED 22 ED 48 45 46 47 45 46 47 48 49 84	FD FD FD FD	PRERR BELL RESTORE RESTR1 SAVE SAVE SAV1	JSR CMP BEQ JSR LDA JSR LDA JSR LDA JMP LDA LDA LDA LDA LDA LDA STA STA STA STA STA STA STA STA STA ST	RD3 RDBYTE CHKSUM BELL #\$C5 COUT #\$87 COUT COUT STATUS ACC XREG YREG ACC XREG YREG STATUS SPNT SETNORM	READ CHKSUM BYTE 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
FF24: FF29: FF29: FF20: FF20: FF34: FF37: FF37: FF37: FF37: FF37: FF34: FF44: FF44: FF44: FF44: FF44: FF44: FF44: FF51: FF55: FF55: FF55: FF55: FF55: FF55: FF55:	90 20 C5 F0 A9 20 A9 4C A5 48 A5 48 A5 A6 A2 86 85 86 85 BA 85 BA 85 BA 80 20 20 20 20 20 20 20 20 20 20 20 20 20	EC 2E 0D C5 ED 2E 0D 2E 0D 2E 0E 2E	FD FD FD FD	PRERR BELL RESTORE RESTR1 SAVE SAV1	JSR CMP BEQ JSR LDA JSR LDA JSR LDA LDA LDA LDA LDA LDA LDA LDA STA STA STA STA STA STA STA STA STA ST	RD3 RDBYTE CHKSUM BELL #\$C5 COUT #\$87 COUT STATUS ACC XREG YREG ACC XREG YREG STATUS SPNT SETNORM INIT	READ CHKSUM BYTE 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
FF24: FF22: FF22: FF22: FF34: FF37: FF37: FF37: FF37: FF37: FF41: FF44: FF44: FF44: FF44: FF45: FF45: FF55: FF55: FF55: FF55: FF55: FF55: FF55: FF55:	90 20 C5 F0 A9 20 20 A9 4C A5 48 A5 A6 A4 28 60 85 86 84 08 85 BA 86 D8 60 20 20 20 20 20 20 20 20 20 20 20 20 20	EC 2E 0D C5 ED D2 ED 87 ED 48 45 46 47 45 46 47 48 49 84 2F 33	FD FD FD FD FD FE	PRERR BELL RESTORE RESTR1 SAVE SAV1	JSR CMP BEQ LDA JSR LDA JSR LDA JMP LDA PHA LDA LDA LDA LDA LDA STA STX STY PHP PLA STA STX STY PLDA STA STX STY JSR JSR JSR JSR JSR JSR JSR JSR LDA	RD3 RDBYTE CHKSUM BELL #\$C5 COUT #\$D2 COUT COUT COUT STATUS ACC XREG YREG ACC XREG YREG STATUS SPNT SETNORM INIT SETVID	READ CHKSUM BYTE 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
FF24: FF29: FF29: FF20: FF20: FF34: FF37: FF37: FF37: FF37: FF37: FF34: FF44: FF44: FF44: FF44: FF44: FF44: FF44: FF51: FF55: FF55: FF55: FF55: FF55: FF55: FF55:	90 20 C5 F0 A9 20 20 A9 4C A5 48 A5 A6 A4 28 60 85 86 84 08 85 BA 86 D8 60 20 20 20 20 20 20 20 20 20 20 20 20 20	EC 2E 0D C5 ED D2 ED 87 ED 48 45 46 47 45 46 47 48 49 84 2F 33	FD FD FD FD FD FE FE FE	PRERR BELL RESTORE RESTR1 SAVE SAVE SAVE	JSR CMP BEQ LDA JSR LDA JSR LDA JSR LDA PHA LDA LDA LDA LDA LDA LDA STA STY PHP PLA STX STY PHP PLA STX STX CLD RTS JSR JSR JSR JSR JSR JSR JSR LDA	RD3 RDBYTE CHKSUM BELL #\$C5 COUT #\$D2 COUT COUT #\$87 COUT STATUS ACC XREG YREG ACC XREG YREG STATUS STATUS SPNT SETNORM INIT SETVID SETKBD	READ CHKSUM BYTE 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
FF24: FF22: FF22: FF22: FF34: FF37: FF37: FF37: FF37: FF37: FF41: FF44: FF44: FF44: FF44: FF45: FF45: FF55: FF55: FF55: FF55: FF55: FF55: FF55: FF55:	90 20 F0 A9 20 20 A9 20 A9 20 A9 20 A9 20 A9 20 A9 20 A9 20 85 86 85 86 85 86 85 86 85 86 20 20 20 20 20 20 20 20 20 20 20 20 20	EC 2E 0D 25 ED 22 ED 87 ED 48 45 46 47 45 46 47 48 49 82F 93 89	FD FD FD FD FD FD FD FD FE FE	PRERR BELL RESTORE RESTR1 SAVE SAVE SAVE	JSR CMP BEQ LDA JSR LDA JSR LDA JMP LDA PHA LDA LDA LDA LDA LDA STA STX STY PHP PLA STA STX STY PLDA STA STX STY JSR JSR JSR JSR JSR JSR JSR JSR LDA	RD3 RDBYTE CHKSUM BELL #\$C5 COUT #\$D2 COUT COUT #\$87 COUT STATUS ACC XREG YREG ACC XREG YREG STATUS STATUS SPNT SETNORM INIT SETVID SETKBD	READ CHKSUM BYTE 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
FF24: FF22: FF22: FF22: FF34: FF37: FF34: FF37: FF37: FF34: FF41: FF42: FF44: FF44: FF44: FF44: FF44: FF45: FF55:	90 20 20 20 20 20 20 20 20 20 20 20 20 20	EC 2E 0D C5 ED 2ED ED 87 ED 48 45 46 47 45 46 47 48 49 84 2F 93 89	FD FD FD FD FD FD FD FD	PRERR BELL RESTORE RESTR1 SAVE SAVE RESET MON	JSR CMP BEQ JSR LDA JSR LDA JSR LDA PHA LDA LDA LDA LDA LDA LDA STA STA STA STA STA STA STA STA STA ST	RD3 RDBYTE CHKSUM BELL #\$C5 COUT COUT #\$87 COUT STATUS ACC XREG YREG STATUS SPNT SETNORM INIT SETVID SETKBD	READ CHKSUM BYTE 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
FF24: FF29: FF29: FF29: FF27: FF34: FF37: FF37: FF37: FF37: FF41: FF42: FF44: FF44: FF44: FF44: FF44: FF45: FF51: FF55: FF56: FF56: FF56: FF56: FF65:	90 20 5 F0 20 20 20 20 20 20 20 20 20 20 20 20 20	EC 2E 0D C5 ED 2ED 87 ED 48 45 46 47 45 46 47 48 49 84 2F 38 38 3A	FD FD FD FD FD FE FE FE FE FE FE	PRERR BELL RESTORE RESTR1 SAVE SAVE RESET MON	JSR CMP BEQ JSR LDA JSR LDA JSR LDA LDA LDA LDA LDA LDA LDA LDA LDA STA STA STA STA STA STA STA STA JSR STA JSR JSR JSR LDA LDA LDA LDA LDA LDA LDA LDA LDA LDA	RD3 RDBYTE CHKSUM BELL #\$C5 COUT COUT COUT STATUS ACC XREG YREG STATUS SPNT SETNORM INIT SETVID SETKBD BELL	READ CHKSUM BYTE 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
FF24: FF22: FF22: FF22: FF32: FF34: FF37: FF37: FF37: FF34: FF44: FF44: FF44: FF44: FF44: FF44: FF45: FF51: FF55: FF55: FF55: FF55: FF55: FF55: FF55: FF55: FF55: FF55: FF55: FF65: FF65: FF65: FF66: FF69:	90 20 C5 F0 20 20 20 20 20 20 20 20 20 20 20 20 20	EC 2E 0D C5 ED 2 ED ED 2 ED 48 45 46 47 45 46 47 48 49 84 2F 93 89 3A AA	FD FD FD FD FD FD FD FD FD FD FD FD FD F	PRERR BELL RESTORE RESTR1 SAVE SAV1 RESET MON	JSR CMP BEQ JSR LDA JSR LDA JSR LDA LDA LDA LDA LDA LDA LDA LDA LDA STA STX STA STX STA STX STA STA STA STA JSR STA JSR LDA LDA LDA LDA LDA LDA LDA LDA LDA LDA	RD3 RDBYTE CHKSUM BELL #\$C5 COUT COUT COUT STATUS ACC XREG YREG STATUS SPNT SETNORM INIT SETVID SETKBD BELL #\$AA	READ CHKSUM BYTE 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
FF24: FF22: FF22: FF22: FF34: FF37: FF37: FF37: FF37: FF37: FF41: FF42: FF44: FF44: FF44: FF44: FF44: FF45: FF51: FF55: FF56: FF66:	90 20 C5 F0 20 20 20 20 20 20 20 20 20 20 85 86 85 86 85 86 85 86 85 86 85 86 85 80 20 20 20 20 20 20 20 20 20 20 20 20 20	EC 2E 0D C5 ED 2ED ED 2ED ED 48 45 46 47 45 46 47 48 49 82F 93 89 3AA 33	FD FD FD FD FD FE FE FE FE FE	PRERR BELL RESTORE RESTR1 SAVE SAV1	JSR CMP BEQ LDA JSR LDA JSR LDA JSR LDA LDA LDA LDA LDA LDA LDA LDA LDA LDA	RD3 RDBYTE CHKSUM BELL #\$C5 COUT #\$D2 COUT COUT #\$87 COUT STATUS ACC XREG YREG STATUS SPNT SETNORM INIT SETVID SETKBD BELL #\$AA PROMPT	READ CHKSUM BYTE 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
FF24: FF22: FF22: FF22: FF34: FF37: FF37: FF37: FF37: FF37: FF41: FF42: FF44: FF44: FF44: FF44: FF44: FF45: FF51: FF55: FF56: FF66:	90 20 C5 F0 20 20 20 20 20 20 20 20 20 20 85 86 85 86 85 86 85 86 85 86 85 86 85 80 20 20 20 20 20 20 20 20 20 20 20 20 20	EC 2E 0D C5 ED 2ED ED 2ED ED 48 45 46 47 45 46 47 48 49 82F 93 89 3AA 33	FD FD FD FD FD FE FE FE FE FE	PRERR BELL RESTORE RESTR1 SAVE SAV1	JSR CMP BEQ LDA JSR LDA JSR LDA JSR LDA LDA LDA LDA LDA LDA LDA LDA LDA LDA	RD3 RDBYTE CHKSUM BELL #\$C5 COUT #\$D2 COUT COUT #\$87 COUT STATUS ACC XREG YREG STATUS SPNT SETNORM INIT SETVID SETKBD BELL #\$AA PROMPT	READ CHKSUM BYTE 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

	~ ~	~ ~			TOD	ENODE	GLEND NON NODE GON TOY
FF70:					JSR	ZMODE	CLEAR MON MODE, SCAN IDX
FF73:	20	A'/	F.F.	NX.I.T.I.W	JSR	GETNUM	GET ITEM, NON-HEX
FF76:	84	34			STY	YSAV	GET ITEM, NON-HEX CHAR IN A-REG X-REG=0 IF NO HEX INPUT
FF78:	A0	17			LDY	#\$17	X-REG=0 IF NO HEX INPUT
				CHRSRCH	DEY		
FF7B:					BMI	MON	NOT FOUND, GO TO MON
FF7D:					CMP	CHRTBL,Y	FIND CMND CHAR IN TEL FOUND, CALL CORRESPONDING
FF80:					BNE	CHRSRCH	
FF82:					JSR	TOSUB	FOUND, CALL CORRESPONDING
FF85:	Α4	34			LDY	YSAV NXTITM	SUBROUTINE
FF87:	4C	73	FF		JMP	NXTITM	
FF8A:	A2	03		DIG	LDX	#\$03	
FF8C:	0A				ASL	A	
FF8D:	0A				ASL ASL	A	GOT HEX DIG,
FF8E:	0A				ASL	A	SHIFT INTO A2
FF8F:	0A				ASL	A	
FF90:	0A			NXTBIT	ASL	A	
FF91:	26	3E			ROL	A2L	
FF93:	26	3F			ROL	A2H	
FF95:	CA				DEX	NXTBIT	LEAVE X=\$FF IF DIG
FF96:	10	F8			BPL	NXTBIT	
				NXTBAS	LDA	MODE	
FF9A:					BNE	NXTBS2	IF MODE IS ZERO THEN COPY A2 TO A1 AND A3
FF9C:					LDA	A2H.X	THEN COPY A2 TO
FF9E:					STA	A1H.X	A1 AND A3
FFA0:					STA	ДЗН X	
FFA2:				NXTBS2	TNY	11511,11	
FFA2: FFA3:				NA1D52	DEO	NYTDAC	
FFA5: FFA5:					DNE	NXTBAS	
				annun.	BNE	NXTCHR	
FFA7:	A2	00		GETNUM	LDX	#\$00	CLEAR A2
					STX	A2L	
FFAB:	86	3F			STX	A2H	
FFAD:	В9	00	02	NXTCHR	LDA	IN,Y	GET CHAR
FFB0:					INY		
FFB1:	49	В0			EOR	#\$B0	
FFB3:	C9	0A			CMP		
FFB5:	90	D3			BCC	DIG	IF HEX DIG, THEN
FFB7:	69	88			ADC	#\$88	
FFB9:	C9	FA			CMP	#\$FA	
FFBB:	В0	CD			BCS	DIG	
FFBD:	60				RTS		
FFBE:	A9	FE		TOSUB	LDA	#GO/256	PUSH HIGH-ORDER
FFC0:					PHA		SUBR ADR ON STK
FFC1:	В9	E3	FF		LDA	SUBTBL,Y	PUSH LOW-ORDER
FFC4:	48				PHA		SUBR ADR ON STK
FFC4: FFC5:					PHA LDA	MODE	SUBR ADR ON STK
FFC5:	A5	31			LDA	MODE #\$00	
FFC5:	A5	31			LDA		CLR MODE, OLD MODE
FFC5: FFC7:	A5 A0	31 00		ZMODE	LDA LDY	#\$00	CLR MODE, OLD MODE
FFC5: FFC7: FFC9:	A5 A0 84	31 00 31		ZMODE	LDA LDY STY		CLR MODE, OLD MODE TO A-REG
FFC5: FFC7: FFC9: FFCB:	A5 A0 84 60	31 00 31		ZMODE	LDA LDY STY RTS	#\$00 MODE	CLR MODE, OLD MODE TO A-REG GO TO SUBR VIA RTS
FFC5: FFC7: FFC9: FFCB: FFCC:	A5 A0 84 60 BC	31 00 31		ZMODE CHRTBL	LDA LDY STY RTS DFB	#\$00 MODE \$BC	CLR MODE, OLD MODE TO A-REG GO TO SUBR VIA RTS F("CTRL-C")
FFC5: FFC7: FFC9: FFCB: FFCC: FFCC:	A5 A0 84 60 BC B2	31 00 31		ZMODE CHRTBL	LDA LDY STY RTS DFB	#\$00 MODE \$BC	CLR MODE, OLD MODE TO A-REG GO TO SUBR VIA RTS F("CTRL-C") F("CTRL-Y")
FFC5: FFC7: FFC9: FFCB: FFCC: FFCC: FFCC:	A5 A0 84 60 BC B2 BE	31 00 31		ZMODE CHRTBL	LDA LDY STY RTS DFB DFB DFB	#\$00 MODE \$BC \$B2 \$BE	CLR MODE, OLD MODE TO A-REG GO TO SUBR VIA RTS F("CTRL-C") F("CTRL-Y") F("CTRL-E")
FFC5: FFC7: FFC9: FFCB: FFCC: FFCC: FFCC: FFCC:	A5 A0 84 60 BC B2 BE ED	31 00 31		ZMODE CHRTBL	LDA LDY RTS DFB DFB DFB DFB	#\$00 MODE \$BC \$B2 \$BE \$ED	CLR MODE, OLD MODE TO A-REG GO TO SUBR VIA RTS F("CTRL-C") F("CTRL-Y") F("CTRL-E") F("TT")
FFC5: FFC9: FFCB: FFCC: FFCC: FFCC: FFCC: FFCC: FFCF: FFC0:	A5 A0 84 60 BC B2 BE ED EF	31 00 31		ZMODE CHRTBL	LDA LDY RTS DFB DFB DFB DFB DFB	#\$00 MODE \$BC \$B2 \$BE \$ED \$EF	CLR MODE, OLD MODE TO A-REG GO TO SUBR VIA RTS F("CTRL-C") F("CTRL-Y") F("CTRL-E") F("TT") F("V")
FFC5: FFC7: FFC8: FFCC: FFCC: FFCC: FFC5: FFC6: FFC0: FFD0: FFD1:	A5 A0 84 60 BC B2 BE ED EF C4	31 00 31		ZMODE CHRTBL	LDA LDY STY RTS DFB DFB DFB DFB DFB	#\$00 MODE \$BC \$B2 \$BE \$ED \$EF \$C4	CLR MODE, OLD MODE TO A-REG GO TO SUBR VIA RTS F("CTRL-C") F("CTRL-Y") F("CTRL-E") F("T") F("V") F("CTRL-K")
FFC5: FFC7: FFC8: FFCC: FFCC: FFCC: FFCC: FFC5: FFC0: FFD1: FFD2:	A5 A0 84 60 B2 BE ED EF C4 EC	31 00 31		ZMODE CHRTBL	LDA LDY STY RTS DFB DFB DFB DFB DFB DFB DFB	#\$00 MODE \$BC \$B2 \$BE \$EB \$EF \$C4 \$EC	CLR MODE, OLD MODE TO A-REG GO TO SUBR VIA RTS F("CTRL-C") F("CTRL-Y") F("CTRL-E") F("TT") F("V") F("CTRL-K") F("S")
FFC5: FFC7: FFC8: FFC2: FFC2: FFC2: FFC4: FFC5: FFD0: FFD1: FFD2: FFD3:	A5 A0 84 60 BC BE ED EF C4 EC A9	31 00 31		ZMODE CHRTBL	LDA LDY STY RTS DFB DFB DFB DFB DFB DFB DFB DFB	#\$00 MODE \$BC \$B2 \$BE \$ED \$EF \$C4 \$EC \$A9	CLR MODE, OLD MODE TO A-REG GO TO SUBR VIA RTS F("CTRL-C") F("CTRL-Y") F("CTRL-E") F("T") F("V") F("CTRL-K") F("S") F("CTRL-P")
FFC5: FFC7: FFC8: FFC2: FFC2: FFC2: FFC2: FFC2: FFD0: FFD1: FFD2: FFD3: FFD4:	A5 A0 84 60 B2 B2 BE ED EF C4 EC A9 BB	31 00 31		ZMODE CHRTBL	LDA LDY STY RTS DFB DFB DFB DFB DFB DFB DFB DFB DFB	#\$00 MODE \$BC \$B2 \$BE \$ED \$C4 \$EC \$A9 \$BB	CLR MODE, OLD MODE TO A-REG GO TO SUBR VIA RTS F("CTRL-C") F("CTRL-Y") F("CTRL-E") F("T") F("V") F("CTRL-K") F("S") F("CTRL-P") F("CTRL-B")
FFC5: FFC7: FFC9: FFC5: FFCC: FFC2: FFC2: FFC5: FFD1: FFD1: FFD2: FFD3: FFD4: FFD5:	A5 A0 84 60 B2 B2 ED EF C4 EC A9 BB A6	31 00 31		ZMODE CHRTBL	LDA LDY STY RTS DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	#\$00 MODE \$BC \$B2 \$BE \$ED \$EF \$C4 \$EC \$A9 \$BB \$A6	CLR MODE, OLD MODE TO A-REG GO TO SUBR VIA RTS F("CTRL-C") F("CTRL-Y") F("CTRL-E") F("T") F("V") F("CTRL-K") F("S") F("CTRL-P") F("CTRL-B") F("-")
FFC5: FFC7: FFC9: FFC5: FFCC: FFC2: FFC2: FFD1: FFD2: FFD2: FFD4: FFD4: FFD5: FFD6:	A5 A0 84 60 B2 BE ED EF C4 EC A9 BB A6 A4	31 00 31		ZMODE CHRTBL	LDA LDY STY RTS DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	#\$00 MODE \$BC \$B2 \$BE \$ED \$EF \$C4 \$EC \$A9 \$BB \$A6 \$A4	CLR MODE, OLD MODE TO A-REG GO TO SUBR VIA RTS F("CTRL-C") F("CTRL-Y") F("CTRL-E") F("T") F("V") F("CTRL-K") F("S") F("CTRL-F") F("CTRL-B") F("-") F("+")
FFC5: FFC7: FFC9: FFC5: FFC0: FFC0: FFC7: FFC7: FFD1: FFD2: FFD4: FFD4: FFD5: FFD6: FFD7:	A5 A0 84 60 B2 BE ED EF C4 EC A9 BB A6 A4 06	31 00 31		ZMODE CHRTBL	LDA LDY STY RTS DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	#\$00 MODE \$BC \$B2 \$B5 \$EF \$C4 \$EC \$A9 \$B8 \$A6 \$A4 \$06	CLR MODE, OLD MODE TO A-REG GO TO SUBR VIA RTS F("CTRL-C") F("CTRL-E") F("CTRL-E") F("T") F("CTRL-K") F("CTRL-K") F("CTRL-P") F("CTRL-P") F("-) F("+") F("M") (F=EX-OR \$B0+\$89)
FFC5: FFC7: FFC9: FFC5: FFC2: FFC7: FFC7: FFD1: FFD2: FFD2: FFD4: FFD5: FFD5: FFD7: FFD8:	A5 A0 84 60 BC ED EF C4 EC A9 BB A6 A4 06 95	31 00 31		ZMODE CHRTBL	LDA LDY STY RTS DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	#\$00 MODE \$BC \$B2 \$B5 \$E5 \$C4 \$C4 \$C4 \$C4 \$A9 \$B8 \$A6 \$A6 \$A4 \$C4 \$A6 \$A6 \$A6 \$A6 \$A6 \$A6 \$A6 \$A6 \$A6 \$A6	CLR MODE, OLD MODE TO A-REG GO TO SUBR VIA RTS F("CTRL-C") F("CTRL-Y") F("CTRL-E") F("T") F("T") F("CTRL-K") F("S") F("CTRL-P") F("CTRL-B") F("-) F("+") F("M") (F=EX-OR \$B0+\$89) F("<")
FFC5: FFC7: FFC8: FFC8: FFC6: FFC6: FFC6: FFD0: FFD1: FFD2: FFD2: FFD3: FFD4: FFD5: FFD6: FFD7: FFD8: FFD9:	A5 A0 84 60 B2 B2 ED EF C4 EC A9 BB A6 A4 06 95 07	31 00 31		ZMODE CHRTBL	LDA LDY STY RTS DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	#\$00 MODE \$BC \$B2 \$B5 \$ED \$C4 \$C4 \$C4 \$C4 \$A9 \$BB \$A6 \$A4 \$06 \$95 \$07	CLR MODE, OLD MODE TO A-REG GO TO SUBR VIA RTS F("CTRL-C") F("CTRL-E") F("T") F("V") F("TV") F("CTRL-K") F("S") F("CTRL-B") F("-") F("+") F("M") (F=EX-OR \$B0+\$89) F("<") F("N")
FFC5: FFC7: FFC9: FFC2: FFC2: FFC2: FFC2: FFC1: FFD1: FFD2: FFD3: FFD4: FFD5: FFD5: FFD7: FFD5: FFD5: FFD5: FFD5: FFD5: FFD7:	A5 A0 84 60 B2 B2 ED EF C4 EC A9 BB A6 A6 95 07 02	31 00 31		ZMODE CHRTBL	LDA LDY STY RTS DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	#\$00 MODE \$BC \$B2 \$BE \$ED \$EF \$C4 \$EC \$A9 \$BB \$A6 \$A4 \$06 \$95 \$07 \$02	CLR MODE, OLD MODE TO A-REG GO TO SUBR VIA RTS F("CTRL-C") F("CTRL-E") F("T") F("T") F("V") F("CTRL-K") F("S") F("CTRL-P") F("CTRL-B") F("-") F("-") F("+") F("M") (F=EX-OR \$B0+\$89) F("<") F("I")
FFC5: FFC7: FFC9: FFC5: FFC2: FFC2: FFC2: FFD1: FFD2: FFD3: FFD4: FFD5: FFD6: FFD7: FFD8: FFD4: FFD5: FFD4: FFD5: FFD5: FFD7:	A5 A0 84 60 B2 ED EF C4 EC A9 BB A6 A4 06 95 07 02 05	31 00 31		ZMODE CHRTBL	LDA LDY STY RTS DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	#\$00 MODE \$BC \$B2 \$BE \$ED \$EF \$C4 \$EC \$A9 \$BB \$A6 \$A4 \$06 \$95 \$07 \$02 \$05	CLR MODE, OLD MODE TO A-REG GO TO SUBR VIA RTS F("CTRL-C") F("CTRL-E") F("T") F("V") F("CTRL-K") F("S") F("CTRL-P") F("CTRL-B") F("-") F("+") F("+") F("#1") F("klt") F("I") F("I") F("I") F("I")
FFC5: FFC7: FFC9: FFC5: FFC2: FFC2: FFC5: FFD1: FFD2: FFD4: FFD4: FFD5: FFD5: FFD9: FFD9: FFD9: FFD4: FFD9: FFD4: FFD4: FFD4: FFD9: FFD4: FFD7:	A5 A0 84 60 B2 ED EF C4 EC A9 BB A6 A4 06 95 07 02 05 F0	31 00 31		ZMODE CHRTBL	LDA LDY STY RTS DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	#\$00 MODE \$BC \$B2 \$B5 \$EF \$C4 \$EC \$A9 \$B8 \$A6 \$A4 \$06 \$95 \$02 \$02 \$05 \$F0	CLR MODE, OLD MODE TO A-REG GO TO SUBR VIA RTS F("CTRL-C") F("CTRL-E") F("CTRL-E") F("T") F("CTRL-K") F("S") F("CTRL-F") F("CTRL-P") F("-TRL-P") F("+") F("+") F("M") (F=EX-OR \$B0+\$89) F("<") F("I") F("I") F("W")
FFC5: FFC7: FFC9: FFC5: FFC2: FFC7: FFC7: FFD1: FFD2: FFD4: FFD4: FFD5: FFD5: FFD5: FFD5: FFD7: FFD8: FFD9: FFD4: FFD2:	A5 A0 84 60 BC ED EF C4 EC A9 BB A6 A4 06 95 07 02 05 F0 00	31 00 31		ZMODE CHRTBL	LDA LDY STY RTS DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	#\$00 MODE \$BC \$B2 \$BB \$ED \$EF \$C4 \$EC \$A9 \$BB \$A6 \$A4 \$06 \$95 \$07 \$02 \$05 \$F0 \$00	CLR MODE, OLD MODE TO A-REG GO TO SUBR VIA RTS F("CTRL-C") F("CTRL-E") F("CTRL-E") F("T") F("CTRL-K") F("CTRL-K") F("CTRL-P") F("CTRL-P") F("CTRL-B") F("CTRL-B") F("T) F("T) F("T) F("T) F("T) F("T) F("N") F("N") F("N") F("N") F("N") F("W") F("W") F("G")
FFC5: FFC7: FFC8: FFC6: FFC7: FFC8: FFC9: FFC7: FF01: FFD2: FFD4: FFD5: FFD6: FFD6: FFD7: FFD8: FFD8: FFD01: FFD2: FFD8: FFD8: FFD01: FFD2: FFD2: FFD3: FFD4: FFD5: FFD4: FFD5: FFD5: FFD4: FFD5: FFD5: FFD4: FFD5:	A5 A0 84 60 BC ED EF C4 EC A9 BB A6 A4 06 95 07 02 05 F0 00 EB	31 00 31		ZMODE CHRTBL	LDA LDY STY RTS DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	#\$00 MODE \$BC \$B2 \$B5 \$ED \$ED \$C4 \$A9 \$A9 \$A9 \$A4 \$06 \$95 \$07 \$02 \$05 \$F0 \$C0 \$C2 \$C2 \$C2 \$C2 \$C2 \$C2 \$C4 \$EC \$EC \$EC \$EC \$EC \$EC \$EC \$EC \$EC \$EC	CLR MODE, OLD MODE TO A-REG GO TO SUBR VIA RTS F("CTRL-C") F("CTRL-E") F("T") F("CTRL-E") F("V") F("CTRL-K") F("S") F("CTRL-B") F("CTRL-B") F("-") F("+") F("M") (F=EX-OR \$B0+\$89) F("<") F("I") F("I") F("I") F("I") F("G") F("G") F("G") F("R")
FFC5: FFC9: FFC8: FFC6: FFC6: FFC7: FFD1: FFD2: FFD2: FFD4: FFD5: FFD6: FFD7: FFD8: FFD8: FFD2: FFD8: FFD2: FFD8: FFD2: FFD2: </td <td>A5 A0 84 60 B2 BE ED EF C4 EC A9 BB A6 06 95 07 02 05 F0 00 EB 93</td> <td>31 00 31</td> <td></td> <td>ZMODE CHRTBL</td> <td>LDA LDY STY RTS DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB</td> <td>#\$00 MODE \$BC \$B2 \$BE \$ED \$C4 \$A9 \$BB \$A6 \$A4 \$06 \$95 \$07 \$02 \$05 \$07 \$02 \$07 \$02 \$07 \$02 \$07 \$02 \$95</td> <td>CLR MODE, OLD MODE TO A-REG GO TO SUBR VIA RTS F("CTRL-C") F("CTRL-E") F("T") F("T") F("V") F("CTRL-K") F("S") F("CTRL-B") F("CTRL-B") F("-") F("+") F("M") (F=EX-OR \$B0+\$89) F("<") F("I") F("I") F("I") F("I") F("I") F("I") F("I") F("R") F("R") F(":")</td>	A5 A0 84 60 B2 BE ED EF C4 EC A9 BB A6 06 95 07 02 05 F0 00 EB 93	31 00 31		ZMODE CHRTBL	LDA LDY STY RTS DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	#\$00 MODE \$BC \$B2 \$BE \$ED \$C4 \$A9 \$BB \$A6 \$A4 \$06 \$95 \$07 \$02 \$05 \$07 \$02 \$07 \$02 \$07 \$02 \$07 \$02 \$95	CLR MODE, OLD MODE TO A-REG GO TO SUBR VIA RTS F("CTRL-C") F("CTRL-E") F("T") F("T") F("V") F("CTRL-K") F("S") F("CTRL-B") F("CTRL-B") F("-") F("+") F("M") (F=EX-OR \$B0+\$89) F("<") F("I") F("I") F("I") F("I") F("I") F("I") F("I") F("R") F("R") F(":")
FFC5: FFC9: FFC8: FFC5: FFC7: FFC7: FFC7: FFD1: FFD2: FFD2: FFD4: FFD5: FFD6: FFD7: FFD8: FFD9: FFD8: FFD01: FFD02: FFD1: FFD2: FFD2: FFD1: FFD2:	A5 A0 84 60 B2 BE ED EF C4 EC A9 BB A6 06 95 07 02 05 F0 00 EB 93 A7	31 00 31		ZMODE CHRTBL	LDA LDY STY RTS DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	#\$00 MODE \$BC \$B2 \$BE \$ED \$C4 \$C4 \$C4 \$C4 \$C4 \$A9 \$BB \$A6 \$A4 \$06 \$95 \$07 \$02 \$05 \$07 \$02 \$05 \$F0 \$00 \$EB \$27	CLR MODE, OLD MODE TO A-REG GO TO SUBR VIA RTS F("CTRL-C") F("CTRL-Y") F("CTRL-E") F("V") F("V") F("CTRL-K") F("S") F("CTRL-B") F("-") F("-") F("-") F("-") F("-") F("-") F("-") F("-") F("-") F("
FFC5: FFC7: FFC9: FFC5: FFC6: FFC1: FFD2: FFD1: FFD2: FFD3: FFD4: FFD5: FFD5: FFD6: FFD9: FFD8: FFD8: FFD9: FFD7: FFD8: FFD9: FFD0: FFE0: FFE1:	A5 A0 84 60 B2 BE EF C4 EC A9 BB A6 A4 06 95 07 02 05 00 EB 93 A7 C6	31 00 31		ZMODE CHRTBL	LDA LDY STY RTS DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	#\$00 MODE \$BC \$B2 \$B5 \$EF \$C4 \$EC \$A9 \$B8 \$A6 \$A4 \$06 \$95 \$00 \$02 \$05 \$02 \$05 \$F0 \$00 \$20 \$27 \$27 \$27 \$27 \$27 \$27 \$27 \$27 \$27 \$27	CLR MODE, OLD MODE TO A-REG GO TO SUBR VIA RTS F("CTRL-C") F("CTRL-E") F("CTRL-E") F("T") F("CTRL-K") F("S") F("CTRL-P") F("CTRL-P") F("+") F("+") F("+") F("klt")
FFC5: FFC9: FFC9: FFC0: FFC0: FFC0: FFC1: FFD2: FFD1: FFD2: FFD3: FFD4: FFD5: FFD5: FFD6: FFD2: FFD5: FFD2: FFD5: FFD5: FFD5: FFD5: FFD5: FFD5: FFE05: FFE1: FFE2:	A5 A0 84 60 B2 BE EF C4 EF A9 BB A6 A4 06 95 07 02 05 00 88 93 A7 C6 99	31 00 31		ZMODE	LDA LDY STY RTS DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	#\$00 MODE \$BC \$B2 \$B5 \$C4 \$C4 \$C4 \$C4 \$C4 \$C4 \$C4 \$C4 \$C4 \$C4	CLR MODE, OLD MODE TO A-REG GO TO SUBR VIA RTS F("CTRL-C") F("CTRL-Y") F("CTRL-E") F("V") F("V") F("CTRL-K") F("S") F("CTRL-B") F("-") F("-") F("-") F("-") F("-") F("-") F("-") F("-") F("-") F("
FFC5: FFC9: FFC8: FFC6: FFC6: FFC7: FFC8: FFD1: FFD2: FFD3: FFD4: FFD5: FFD4: FFD5: FFD7: FFD8: FFD0: FFE0: FFE0: FFE0: FFE2: FFE3:	A5 A0 B2 BED EF C4 EC A9 BB A64 05 500 E3 A7 00 B3 A7 02 99 B2	31 00 31		ZMODE CHRTBL	LDA LDY STY RTS DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	#\$00 MODE \$EC \$B2 \$B5 \$ED \$C4 \$A9 \$A6 \$A4 \$A6 \$A4 \$06 \$95 \$07 \$02 \$05 \$07 \$02 \$05 \$50 \$00 \$EB \$93 \$A7 \$C6 \$99 BASCONT-1	CLR MODE, OLD MODE TO A-REG GO TO SUBR VIA RTS F("CTRL-C") F("CTRL-E") F("CTRL-E") F("T") F("CTRL-K") F("S") F("CTRL-P") F("CTRL-P") F("+") F("+") F("+") F("klt")
FFC5: FFC9: FFC8: FFC6: FFC6: FFC7: FFD1: FFD2: FFD4: FFD5: FFD6: FFD7: FFD8: FFD7: FFD8: FFD7: FFD8: FFD0: FFD0: FFD2: FFE1: FFE2: FFE2: </td <td>A5 A0 B2 BED EF C4 EC A9 BB A6 A0 6 95 60 00 EB 93 A7 699 B2 C9</td> <td>31 00 31</td> <td></td> <td>ZMODE</td> <td>LDA LDY STY RTS DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB</td> <td><pre>#\$00 MODE \$BC \$B2 \$BE \$ED \$C4 \$A9 \$BB \$A6 \$A4 \$06 \$95 \$07 \$02 \$05 \$07 \$02 \$05 \$F0 \$00 \$EB \$93 \$A7 \$C2 \$93 \$A7 \$C2 \$93 \$A7 \$C2 \$00 \$BB \$A2 \$00 \$BB \$A2 \$C2 \$C2 \$C2 \$C2 \$C2 \$C2 \$C2 \$C2 \$C2 \$C</pre></td> <td>CLR MODE, OLD MODE TO A-REG GO TO SUBR VIA RTS F("CTRL-C") F("CTRL-E") F("CTRL-E") F("T") F("CTRL-K") F("S") F("CTRL-P") F("CTRL-P") F("+") F("+") F("+") F("klt")</td>	A5 A0 B2 BED EF C4 EC A9 BB A6 A0 6 95 60 00 EB 93 A7 699 B2 C9	31 00 31		ZMODE	LDA LDY STY RTS DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	<pre>#\$00 MODE \$BC \$B2 \$BE \$ED \$C4 \$A9 \$BB \$A6 \$A4 \$06 \$95 \$07 \$02 \$05 \$07 \$02 \$05 \$F0 \$00 \$EB \$93 \$A7 \$C2 \$93 \$A7 \$C2 \$93 \$A7 \$C2 \$00 \$BB \$A2 \$00 \$BB \$A2 \$C2 \$C2 \$C2 \$C2 \$C2 \$C2 \$C2 \$C2 \$C2 \$C</pre>	CLR MODE, OLD MODE TO A-REG GO TO SUBR VIA RTS F("CTRL-C") F("CTRL-E") F("CTRL-E") F("T") F("CTRL-K") F("S") F("CTRL-P") F("CTRL-P") F("+") F("+") F("+") F("klt")
FFC5: FFC9: FFC8: FFC6: FFC6: FFC7: FFD1: FFD2: FFD3: FFD4: FFD5: FFD6: FFD7: FFD8: FFD8: FFD7: FFD8: FFD7: FFD6: FFD7: FFD6: FFE1: FFE2: FFE4: FFE4: FFE4: FFE4: FFE4:	A5 A0 84 60 B2 ED EF C4 EC A9 BB A6 A4 06 95 07 02 500 EB 93 A7 C6 99 B2 C9 BE	31 00 31		ZMODE	LDA LDY STY RTS DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	<pre>#\$00 MODE \$BC \$B2 \$BE \$ED \$C4 \$C4 \$C4 \$C4 \$A9 \$BB \$A6 \$A4 \$06 \$95 \$07 \$02 \$05 \$07 \$02 \$05 \$07 \$02 \$05 \$F0 \$00 \$EB \$93 \$A7 \$C6 \$99 BASCONT-1 USR-1 REGZ-1</pre>	CLR MODE, OLD MODE TO A-REG GO TO SUBR VIA RTS F("CTRL-C") F("CTRL-E") F("CTRL-E") F("T") F("CTRL-K") F("S") F("CTRL-P") F("CTRL-P") F("+") F("+") F("+") F("klt")
FFC5: FFC7: FFC9: FFC0: FFC0: FFC1: FFC2: FFD1: FFD2: FFD3: FFD4: FFD5: FFD6: FFD7: FFD8: FFD9: FFD8: FFD2: FFD5: FFD2: FFD2: FFD2: FFE1: FFE2: FFE3: FFE4: FFE5: FFE6:	A5 A0 84 60C B2 BE ED EF C4C A9 BB A6 05 F0 02 05 F0 00E 93 A7 C6 99 20 EC 20 EC 20 EC 20 EC 20 EC 20 EC 20 EC 20 EC 20 EC 20 EC 20 EC 20 EC 20 20 EC 20 20 EC 20 20 EC 20 20 EC 20 20 EC 20 20 EC 20 20 EC 20 20 EC 20 20 EC 20 20 EC 20 20 20 20 20 20 20 20 20 20 20 20 20	31 00 31		ZMODE	LDA LDY STY RTS DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	<pre>#\$00 MODE \$EC \$B2 \$B2 \$EF \$ED \$EF \$C4 \$EC \$A9 \$A4 \$06 \$95 \$A4 \$06 \$95 \$07 \$02 \$05 \$F0 \$00 \$EB \$93 \$A7 \$C6 \$93 \$A7 \$C6 \$99 BASCONT-1 USR-1 TRACE-1 TRACE-1</pre>	CLR MODE, OLD MODE TO A-REG GO TO SUBR VIA RTS F("CTRL-C") F("CTRL-E") F("CTRL-E") F("T") F("CTRL-K") F("S") F("CTRL-P") F("CTRL-P") F("+") F("+") F("+") F("klt")
FFC5: FFC9: FFC8: FFC6: FFC6: FFC7: FFD1: FFD2: FFD3: FFD4: FFD5: FFD6: FFD7: FFD8: FFD8: FFD7: FFD8: FFD7: FFD6: FFD7: FFD6: FFE1: FFE2: FFE4: FFE4: FFE4: FFE4: FFE4:	A5 A0 84 60C B2 BE ED EF C4C A9 BB A6 05 F0 02 05 F0 00E 93 A7 C6 99 20 EC 20 EC 20 EC 20 EC 20 EC 20 EC 20 EC 20 EC 20 EC 20 EC 20 EC 20 EC 20 20 EC 20 20 EC EC EC EC EC EC EC EC EC EC EC EC EC	31 00 31		ZMODE	LDA LDY STY RTS DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	<pre>#\$00 MODE \$BC \$B2 \$BE \$ED \$C4 \$C4 \$C4 \$C4 \$A9 \$BB \$A6 \$A4 \$06 \$95 \$07 \$02 \$05 \$07 \$02 \$05 \$07 \$02 \$05 \$F0 \$00 \$EB \$93 \$A7 \$C6 \$99 BASCONT-1 USR-1 REGZ-1</pre>	CLR MODE, OLD MODE TO A-REG GO TO SUBR VIA RTS F("CTRL-C") F("CTRL-E") F("CTRL-E") F("T") F("CTRL-K") F("S") F("CTRL-P") F("CTRL-P") F("+") F("+") F("+") F("klt")
FFC5: FFC7: FFC9: FFC0: FFC0: FFC0: FFC1: FFD2: FFD1: FFD2: FFD3: FFD4: FFD5: FFD6: FFD7: FFD8: FFD7: FFD8: FFD7: FFD8: FFD7: FFD8: FFD7: FFD8: FFD0: FFD0: FFD0: FFE0: FFE0: FFE0: FFE1: FFE2: FFE3: FFE4: FFE5: FFE6:	A5 A0 84 60C B2 BE EF C4C A9 BB A6 A4 065 07 02 05 F0 02 B3 A7 C6 99 B2 C99 B2 C99 B2 C1 35	31 00 31		ZMODE	LDA LDY STY RTS DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	<pre>#\$00 MODE \$EC \$B2 \$B2 \$EF \$ED \$EF \$C4 \$EC \$A9 \$A4 \$06 \$95 \$A4 \$06 \$95 \$07 \$02 \$05 \$F0 \$00 \$EB \$93 \$A7 \$C6 \$93 \$A7 \$C6 \$99 BASCONT-1 USR-1 TRACE-1 TRACE-1</pre>	CLR MODE, OLD MODE TO A-REG GO TO SUBR VIA RTS F("CTRL-C") F("CTRL-E") F("CTRL-E") F("T") F("CTRL-K") F("S") F("CTRL-P") F("CTRL-P") F("+") F("+") F("+") F("klt")
FFC5: FFC9: FFC9: FFC0: FFC0: FFC7: FFD1: FFD2: FFD3: FFD4: FFD5: FFD5: FFD5: FFD2: FFD5: FFD6: FFD7: FFD5: FFD5: FFD5: FFD5: FFD5: FFE1: FFE2: FFE3: FFE4: FFE4: FFE5: FFE6: FFE6: FFE6: FFE6: FFE6: FFE6:	A5 A0 84 6B2 BED EC4 EC4 BB BA6 A06 95 700 500 EB3 AC6 99 BC9 BC1 S5 8C	31 00 31		ZMODE	LDA LDY STY RTS DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	<pre>#\$00 MODE \$EC \$B2 \$ED \$ED \$C4 \$A9 \$A4 \$06 \$A4 \$06 \$A4 \$06 \$A4 \$06 \$95 \$07 \$02 \$05 \$F0 \$00 \$EB \$93 \$A7 \$C6 \$99 BASCONT-1 USR-1 RECZ-1 TRACE-1 VFY-1 STEPZ-1</pre>	CLR MODE, OLD MODE TO A-REG GO TO SUBR VIA RTS F("CTRL-C") F("CTRL-E") F("CTRL-E") F("T") F("CTRL-K") F("S") F("CTRL-P") F("CTRL-P") F("+") F("+") F("+") F("klt")
FFC5: FFC7: FFC9: FFC5: FFC6: FFC6: FFC7: FFD1: FFD2: FFD2: FFD3: FFD4: FFD5: FFD5: FFD6: FFD6: FFD7: FFD6: FFD7: FFD7: FFD7: FFD7: FFD7: FF21: FF22: FF23: FF24: FF25:	A5 A0 84 60C BE BE DE CE A9 BB A64 95 702 050 00 EB 33 C9 BC BE C1 50 C9 BC BE C1 50 C C2 C2 C2 C2 C2 C2 C2 C2 C2 C2 C2 C2 C	31 00 31		ZMODE	LDA LDY STY RTS DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	<pre>#\$00 MODE SEC \$B2 \$BE \$ED \$EF \$C4 \$A9 \$A6 \$A4 \$06 \$95 \$07 \$02 \$05 \$07 \$02 \$05 \$570 \$00 \$EB \$93 \$A7 \$C6 \$99 BASCONT-1 USR-1 REGZ-1 TRACE-1 VFY-1 INPRT-1</pre>	CLR MODE, OLD MODE TO A-REG GO TO SUBR VIA RTS F("CTRL-C") F("CTRL-E") F("CTRL-E") F("T") F("CTRL-K") F("S") F("CTRL-P") F("CTRL-P") F("+") F("+") F("+") F("klt")
FFC5: FFC7: FFC8: FFC8: FFC7: FFC8: FFC7: FFC8: FF01: FF02: FF03: FF04: FF04: FF07: FF08: FF07: FF08: FF07: FF08: FF07: FF08: FF07: FF02: FF02: FF02: FF02: FF02: </td <td>A5 A0 84 60C BE BE BE CCC A9 BB A6 A0 60 507 02 050 0702 807 807 807 807 807 807 807 807 807 807</td> <td>31 00 31</td> <td></td> <td>ZMODE</td> <td>LDA LDY STY RTS DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB</td> <td><pre>#\$00 MODE \$EC \$B2 \$ED \$ED \$C4 \$A9 \$A4 \$06 \$A4 \$06 \$A4 \$06 \$A4 \$06 \$95 \$07 \$02 \$05 \$F0 \$00 \$EB \$93 \$A7 \$C6 \$99 BASCONT-1 USR-1 RECZ-1 TRACE-1 VFY-1 STEPZ-1</pre></td> <td>CLR MODE, OLD MODE TO A-REG GO TO SUBR VIA RTS F("CTRL-C") F("CTRL-E") F("CTRL-E") F("T") F("CTRL-K") F("S") F("CTRL-P") F("CTRL-P") F("+") F("+") F("+") F("klt")</td>	A5 A0 84 60C BE BE BE CCC A9 BB A6 A0 60 507 02 050 0702 807 807 807 807 807 807 807 807 807 807	31 00 31		ZMODE	LDA LDY STY RTS DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	<pre>#\$00 MODE \$EC \$B2 \$ED \$ED \$C4 \$A9 \$A4 \$06 \$A4 \$06 \$A4 \$06 \$A4 \$06 \$95 \$07 \$02 \$05 \$F0 \$00 \$EB \$93 \$A7 \$C6 \$99 BASCONT-1 USR-1 RECZ-1 TRACE-1 VFY-1 STEPZ-1</pre>	CLR MODE, OLD MODE TO A-REG GO TO SUBR VIA RTS F("CTRL-C") F("CTRL-E") F("CTRL-E") F("T") F("CTRL-K") F("S") F("CTRL-P") F("CTRL-P") F("+") F("+") F("+") F("klt")
FFC5: FFC9: FFC8: FFC8: FFC8: FFC8: FFC9: FFD1: FFD2: FFD3: FFD4: FFD5: FFD4: FFD5: FFD6: FFD7: FFD8: FFD7: FFD8: FFD7: FFE0: FFE1: FFE2: FFE4: FFE4: FFE5: FFE6: FFE9:	A5 A0 84 BC B2 BE EC4 EC4 EC4 BB6 A2 BB6 A2 BB6 C2 BC BC BC BC BC BC BC BC BC BC BC BC BC	31 00 31		ZMODE	LDA LDY STY RTS DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	<pre>#\$00 MODE \$EC \$B2 \$EB \$ED \$C4 \$C4 \$A9 \$A6 \$A4 \$06 \$A4 \$06 \$95 \$07 \$02 \$05 \$07 \$02 \$05 \$F0 \$00 \$EB \$93 \$A7 \$C6 \$99 BASCONT-1 USR-1 REGZ-1 TRACE-1 VFY-1 INPRT-1 STEPZ-1 OUTPRT-1 STEPZ-1 OUTPRT-1</pre>	CLR MODE, OLD MODE TO A-REG GO TO SUBR VIA RTS F("CTRL-C") F("CTRL-E") F("CTRL-E") F("T") F("CTRL-K") F("S") F("CTRL-P") F("CTRL-P") F("+") F("+") F("+") F("klt")
FFC5: FFC7: FFC8: FFC5: FFC6: FFC1: FFD2: FFD3: FFD4: FFD5: FFD6: FFD7: FFD8: FFD7: FFD8: FFD7: FFD8: FFD7: FFD8: FFD7: FFD8: FFD2: FFD1: FFE0: FFE0: FFE1: FFE2: FFE2: FFE3: FFE4: FFE5: FFE6: FFE7: FFE8: FFE8: FFE8: FFE4: FFE8: FFE8: FFE8: FFE8: FFE8: FFE8:	A5 A0 84 6B2 BE EC4 EC4 BB6 A2 BB6 A2 BB6 A2 BB6 A2 BB6 BC7 005 F0 005 B07 B2 BC7 B2 BC7 B2 BC7 C05 B2 B2 B2 B2 B2 B2 B2 B2 B2 B2 B2 B2 B2	31 00 31		ZMODE	LDA LDY STY RTS DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	<pre>#\$00 MODE SEC \$B2 \$BE \$ED \$EF \$C4 \$A9 \$BB \$A6 \$A4 \$06 \$95 \$07 \$02 \$05 \$07 \$02 \$05 \$570 \$00 \$EB \$93 \$A7 \$C6 \$99 BASCONT-1 USR-1 REGZ-1 TRACE-1 VFY-1 INPRT-1 STEPZ-1 OUTPRT-1 SETMODE-1 SETMODE-1 SETMODE-1 </pre>	CLR MODE, OLD MODE TO A-REG GO TO SUBR VIA RTS F("CTRL-C") F("CTRL-E") F("CTRL-E") F("T") F("CTRL-K") F("S") F("CTRL-P") F("CTRL-P") F("+") F("+") F("+") F("klt")
FFC5: FFC9: FFC9: FFC5: FFC6: FFC7: FFD1: FFD2: FFD3: FFD4: FFD5: FFD5: FFD5: FFD6: FFD7: FFD8: FFD7: FFD5: FFD6: FFD5: FFD6: FFE1: FFE2: FFE3: FFE4: FFE4: FFE5: FFE6: FFE6: FFE9: FFE8: FFE9: FFE8: FFE8: </td <td>A5 A0 84 6B2 BEDF4 EC4 BB A6 25 70 25 60 82 82 82 82 82 82 82 82 82 82 82 82 82</td> <td>31 00 31</td> <td></td> <td>ZMODE</td> <td>LDA LDY STY RTS DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB</td> <td><pre>#\$00 MODE \$EC \$B2 \$B2 \$EF \$EF \$C4 \$EC \$A9 \$A6 \$A4 \$06 \$95 \$07 \$02 \$05 \$F0 \$00 \$EB \$93 \$A7 \$C6 \$99 BASCONT-1 USR-1 REGZ-1 TRACE-1 VFY-1 INPRT-1 STEPZ-1 OUTPRT-1 SETMODE-1</pre></td> <td>CLR MODE, OLD MODE TO A-REG GO TO SUBR VIA RTS F("CTRL-C") F("CTRL-E") F("CTRL-E") F("T") F("CTRL-K") F("S") F("CTRL-P") F("CTRL-P") F("+") F("+") F("+") F("klt")</td>	A5 A0 84 6B2 BEDF4 EC4 BB A6 25 70 25 60 82 82 82 82 82 82 82 82 82 82 82 82 82	31 00 31		ZMODE	LDA LDY STY RTS DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	<pre>#\$00 MODE \$EC \$B2 \$B2 \$EF \$EF \$C4 \$EC \$A9 \$A6 \$A4 \$06 \$95 \$07 \$02 \$05 \$F0 \$00 \$EB \$93 \$A7 \$C6 \$99 BASCONT-1 USR-1 REGZ-1 TRACE-1 VFY-1 INPRT-1 STEPZ-1 OUTPRT-1 SETMODE-1</pre>	CLR MODE, OLD MODE TO A-REG GO TO SUBR VIA RTS F("CTRL-C") F("CTRL-E") F("CTRL-E") F("T") F("CTRL-K") F("S") F("CTRL-P") F("CTRL-P") F("+") F("+") F("+") F("klt")
FFC5: FFC9: FFC9: FFC0: FFC0: FFC0: FFC1: FFD2: FFD1: FFD2: FFD4: FFD5: FFD4: FFD5: FFD4: FFD5: FFD5: FFD4: FFD5: FFD5: FFD6: FFD6: FFD7: FFD6: FFD7: FFE4: FFE2: FFE4: FFE4: FFE5: FFE4: FFE5: FFE4: FFE5: FFE4: FFE4: </td <td>A5 A0 84 6B2 BEDF4 EC49 BBAA4 95 700 500 EB37 C99 BC9 EC4 C3 8C3 96 A17 2B</td> <td>31 00 31</td> <td></td> <td>ZMODE</td> <td>LDA LDY STY RTS DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB</td> <td><pre>#\$00 MODE SEC \$B2 \$BE \$ED \$EF \$C4 \$A9 \$BB \$A6 \$A4 \$06 \$95 \$07 \$02 \$05 \$07 \$02 \$05 \$570 \$00 \$EB \$93 \$A7 \$C6 \$99 BASCONT-1 USR-1 REGZ-1 TRACE-1 VFY-1 INPRT-1 STEPZ-1 OUTPRT-1 SETMODE-1 SETMODE-1 SETMODE-1 </pre></td> <td>CLR MODE, OLD MODE TO A-REG GO TO SUBR VIA RTS F("CTRL-C") F("CTRL-E") F("CTRL-E") F("T") F("CTRL-K") F("S") F("CTRL-P") F("CTRL-P") F("+") F("+") F("+") F("klt")</td>	A5 A0 84 6B2 BEDF4 EC49 BBAA4 95 700 500 EB37 C99 BC9 EC4 C3 8C3 96 A17 2B	31 00 31		ZMODE	LDA LDY STY RTS DFB DFB DFB DFB DFB DFB DFB DFB DFB DFB	<pre>#\$00 MODE SEC \$B2 \$BE \$ED \$EF \$C4 \$A9 \$BB \$A6 \$A4 \$06 \$95 \$07 \$02 \$05 \$07 \$02 \$05 \$570 \$00 \$EB \$93 \$A7 \$C6 \$99 BASCONT-1 USR-1 REGZ-1 TRACE-1 VFY-1 INPRT-1 STEPZ-1 OUTPRT-1 SETMODE-1 SETMODE-1 SETMODE-1 </pre>	CLR MODE, OLD MODE TO A-REG GO TO SUBR VIA RTS F("CTRL-C") F("CTRL-E") F("CTRL-E") F("T") F("CTRL-K") F("S") F("CTRL-P") F("CTRL-P") F("+") F("+") F("+") F("klt")

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

	******	*****	******	**	
	*			*	
	* A	PPLE-I	I	*	
	* MINI	-ASSEM	BLER	*	
	*			*	
	* COPYR				
	* APPLE	COMPUT	ER INC.		
	*	~~~~~		*	
	* ALL RI	GHTS R	ESERVED	*	
		WOZNI.	λV	*	
	υ.	. BAUM		*	
	*******			**	
	TITLE ".	APPLE-	II MINI.	-ASS	EMBLER"
	FORMAT	EQU	\$2E		
	LENGTH	EQU	\$2F		
		EQU	\$31		
	PROMPT	EQU	\$33		
	YSAV	EQU	\$34		
		EQU	\$35		
		EQU	\$3A		
		EQU	\$3B		
		EQU	\$3D		
		EQU	\$3E		
		EQU EQU	\$3F \$42		
		EQU	\$43		
			\$44		
		EQU	\$200		
	INSDS2	EQU	\$F88E		
	INSTDSP	EQU	\$F8D0		
		EQU	\$F94A		
			\$F953		
		EQU	\$F9B4		
		EQU	\$F9BA		
		EQU	\$F9C0		
	MNEMR CURSUP	EQU	\$FA00 \$FC1A		
		EQU	\$FD67		
		EQU	\$FDED		
		EQU	, \$FE00		
	A1PCLP	EQU	\$FE78		
	BELL	EQU	\$FF3A		
	GETNUM	EQU	\$FFA7		
		EQU	\$FFBE		
		EQU	\$FFC7		
	CHRTBL		\$FFCC		
F500: E9 81	REL		\$F500 #\$81		IS FMT COMPATIBLE
F502: 4A	KEL	SBC LSR	#\$01		WITH RELATIVE MODE?
F502: 4A F503: D0 14		BNE	EBB3		NO.
F505: A4 3F		LDY	A2H		110.
F507: A6 3E		LDX	A2L		DOUBLE DECREMENT
F509: D0 01		BNE	REL2		
F50B: 88		DEY			
F50C: CA	REL2	DEX			
F50D: 8A		TXA			
F50E: 18		CLC			
F50F: E5 3A		SBC	PCL		FORM ADDR-PC-2
F511: 85 3E		STA	A2L		
F513: 10 01 F515: C8		BPL INY	REL3		
F516: 98	REL3	TYA			
1010. 00	, LLLJ	110			

	E5	20			SBC	חכת	
F519:				ERR3			ERROR IF >1-BYTE BRANCH
				FINDOP			Biaton II /I BIID Biation
							MOVE INST TO (PC)
F51D.			00	FNDOP2			NOVE INST TO (TC)
F520:					DEY	(PCL),Y	
F523:						FNDOP2	
F525:			FC			CURSUP	
F525: F528:							RESTORE CURSOR
F528:							TYPE FORMATTED LINE
					JSK		
F52E: F531:			F9				UPDATE PC
						PCH	
F533:			-			PCL	
F535:							GET NEXT LINE
				FAKEMON3	JSR	TOSUB	GO TO DELIM HANDLER
F53B:					LDY		RESTORE Y-INDEX
			FF	FAKEMON			READ PARAM
F540:							SAVE Y-INDEX
F542:							INIT DELIMITER INDEX
F544:				FAKEMON2	DEY		CHECK NEXT DELIM ERR IF UNRECOGNIZED DELIM
F545:							
F547:			FF				COMPARE WITH DELIM TABLE
F54A:						FAKEMON2	
F54C:							MATCH, IS IT CR?
F54E:	D0	E8			BNE	FAKEMON3	NO, HANDLE IT IN MONITOR
F550:	Α5	31			LDA	MODE	
F552:	A0	00				#\$0	
F554:	C6	34			DEC	YSAV	
F556:	20	00	FE		JSR	BL1	HANDLE CR OUTSIDE MONITOR
F559:	4C	95	F5		JMP	NXTLINE	
F55C:	A5	3D		TRYNEXT	LDA	A1H	GET TRIAL OPCODE
F55E:	20	8E	F8		JSR	INSDS2	GET FMT+LENGTH FOR OPCODE
F561:	AA				TAX		
F562:	BD	00	FA		LDA	MNEMR,X	GET LOWER MNEMONIC BYTE
F565:					CMP		MATCH?
F567:							NO, TRY NEXT OPCODE.
F569:			F9			MNEML,X	GET UPPER MNEMONIC BYTE
F56C:					CMP		MATCH?
F56E:						NEXTOP	NO, TRY NEXT OPCODE
F570:						FMT	NO, INI NEMI OFCODE
F572:					LDY		GET TRIAL FORMAT
F574:						#\$9D	TRIAL FORMAT RELATIVE?
F574:						REL	YES.
F578:							SAME FORMAT?
F57A: F57C:				NEXTOP		FINDOP	YES.
						A1H	NO, TRY NEXT OPCODE
F57E:		DC.			BNE	TRYNEXT	
F580:		44				FMT	NO MORE, TRY WITH LEN=2
F582:	C6	44 35			DEC	L	WAS L=2 ALREADY?
F582: F584:	C6 F0	44 35 D6			DEC BEQ	L TRYNEXT	WAS L=2 ALREADY? NO.
F582: F584: F586:	C6 F0 A4	44 35 D6 34		ERR	DEC BEQ LDY	L TRYNEXT	WAS L=2 ALREADY?
F582: F584: F586: F588:	C6 F0 A4 98	44 35 D6 34			DEC BEQ LDY TYA	L TRYNEXT	WAS L=2 ALREADY? NO.
F582: F584: F586: F588: F589:	C6 F0 A4 98 AA	44 35 D6 34		ERR	DEC BEQ LDY TYA TAX	L TRYNEXT YSAV	WAS L=2 ALREADY? NO. YES, UNRECOGNIZED INST.
F582: F584: F586: F588: F589: F58A:	C6 F0 A4 98 AA 20	44 35 D6 34 4A	F9	ERR	DEC BEQ LDY TYA TAX JSR	L TRYNEXT YSAV PRBL2	WAS L=2 ALREADY? NO. YES, UNRECOGNIZED INST. PRINT ^ UNDER LAST READ
F582: F584: F586: F588: F589: F58A: F58A:	C6 F0 A4 98 AA 20 A9	44 35 D6 34 4A DE	F9	ERR ERR2	DEC BEQ LDY TYA TAX JSR LDA	L TRYNEXT YSAV PRBL2 #\$DE	WAS L=2 ALREADY? NO. YES, UNRECOGNIZED INST. PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR
F582: F584: F586: F588: F589: F58A: F58D: F58F:	C6 F0 A4 98 AA 20 A9 20	44 35 06 34 4A DE ED	F9 FD	ERR ERR2	DEC BEQ LDY TYA TAX JSR LDA JSR	L TRYNEXT YSAV PRBL2 #\$DE COUT	WAS L=2 ALREADY? NO. YES, UNRECOGNIZED INST. PRINT ^ UNDER LAST READ
F582: F584: F586: F588: F589: F58A: F58D: F58F: F592:	C6 F0 A4 98 AA 20 A9 20 20	44 35 34 4A DE ED 3A	F9 FD FF	ERR ERR2 RESETZ	DEC BEQ LDY TYA TAX JSR LDA JSR JSR	L TRYNEXT YSAV PRBL2 #\$DE COUT BELL	WAS L=2 ALREADY? NO. YES, UNRECOGNIZED INST. PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION.
F582: F584: F586: F588: F589: F58A: F58D: F58F: F58F: F592: F595:	C6 F0 A4 98 AA 20 A9 20 20 A9	44 35 D6 34 4A DE ED 3A A1	F9 FD FF	ERR ERR2	DEC BEQ LDY TYA TAX JSR LDA JSR JSR	L TRYNEXT YSAV PRBL2 #\$DE COUT BELL #\$A1	WAS L=2 ALREADY? NO. YES, UNRECOGNIZED INST. PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION.
F582: F584: F586: F588: F589: F58A: F58D: F58F: F592:	C6 F0 A4 98 AA 20 A9 20 20 A9	44 35 D6 34 4A DE ED 3A A1	F9 FD FF	ERR ERR2 RESETZ NXTLINE	DEC BEQ LDY TYA TAX JSR LDA JSR LDA STA	L TRYNEXT YSAV PRBL2 #\$DE COUT BELL #\$A1 PROMPT	WAS L=2 ALREADY? NO. YES, UNRECOGNIZED INST. PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '!' INITIALIZE PROMPT
F582: F584: F586: F588: F589: F58A: F58D: F58F: F592: F595: F597: F599:	C6 F0 A4 98 AA 20 A9 20 20 A9 85 20	44 35 D6 34 4A DE ED 3A A1 33 67	F9 FD FF FD	ERR ERR2 RESETZ NXTLINE	DEC BEQ LDY TYA TAX JSR LDA JSR LDA STA JSR	L TRYNEXT YSAV PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ	WAS L=2 ALREADY? NO. YES, UNRECOGNIZED INST. PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '!' INITIALIZE PROMPT GET LINE.
F582: F584: F586: F588: F589: F587: F587: F592: F597: F599: F597:	C6 F0 A4 98 AA 20 A9 20 20 A9 85 20 20	44 35 D6 34 4A DE ED 3A A1 33 67 C7	F9 FD FF FD FF	ERR ERR2 RESETZ NXTLINE	DEC BEQ LDY TYA TAX JSR LDA JSR LDA STA JSR JSR JSR	L TRYNEXT YSAV PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE	WAS L=2 ALREADY? NO. YES, UNRECOGNIZED INST. PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '.!' INITIALIZE PROMPT GET LINE. INIT SCREEN STUFF
F582: F584: F586: F588: F589: F58A: F58D: F58F: F592: F595: F597: F599:	C6 F0 A4 98 AA 20 20 20 20 85 20 20	44 35 D6 34 4A DE ED 3A A1 33 67 C7	F9 FD FF FD FF	ERR ERR2 RESETZ NXTLINE	DEC BEQ LDY TYA TAX JSR LDA JSR JSR JSR JSR LDA	L TRYNEXT YSAV PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN	WAS L=2 ALREADY? NO. YES, UNRECOGNIZED INST. PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '!' INITIALIZE PROMPT GET LINE.
F582: F584: F586: F588: F589: F587: F587: F592: F597: F599: F597:	C6 F0 A4 98 AA 20 20 20 A9 85 20 20 AD	44 35 D6 34 4A DE ED 3A A1 33 67 C7 00	F9 FD FF FD FF	ERR ERR2 RESETZ NXTLINE	DEC BEQ LDY TYA TAX JSR LDA JSR JSR JSR JSR LDA	L TRYNEXT YSAV PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN	WAS L=2 ALREADY? NO. YES, UNRECOGNIZED INST. PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '.!' INITIALIZE PROMPT GET LINE. INIT SCREEN STUFF
F582: F584: F588: F589: F587: F587: F587: F595: F597: F599: F599: F597:	C6 F0 A4 98 AA 20 A9 20 20 20 85 20 20 20 AD C9	44 35 D6 34 4A ED 3A A1 33 67 C7 00 A0	F9 FD FF FD FF	ERR ERR2 RESETZ NXTLINE	DEC BEQ LDY TYA TAX JSR LDA JSR LDA STA JSR LDA STA JSR LDA CMP	L TRYNEXT YSAV PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0	WAS L=2 ALREADY? NO. YES, UNRECOGNIZED INST. PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '!' INITIALIZE PROMPT GET LINE. INIT SCREEN STUFF GET CHAR
F582: F584: F586: F588: F587: F587: F587: F597: F597: F597: F597: F597: F597: F597:	C6 F0 A4 98 AA 20 20 20 20 20 20 20 20 20 20 20 C9 F0	44 35 D6 34 4A ED 3A A1 33 67 C7 00 A0 13	F9 FD FF FD FF	ERR ERR2 RESETZ NXTLINE	DEC BEQ LDY TYA TAX JSR LDA JSR LDA STA JSR LDA STA JSR LDA CMP	L TRYNEXT YSAV PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0	WAS L=2 ALREADY? NO. YES, UNRECOGNIZED INST. PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '!' INITIALIZE PROMPT GET LINE. INIT SCREEN STUFF GET CHAR ASCII BLANK?
F582: F584: F588: F588: F589: F587: F597: F597: F597: F597: F597: F597: F597: F597: F597: F597:	C6 F0 A4 98 A0 20 20 20 A9 85 20 20 AD C9 F0 C8	44 35 D6 34 4A DE ED 3A A1 33 67 C7 00 A0 13	F9 FD FF FD FF	ERR ERR2 RESETZ NXTLINE	DEC BEQ LDY TYA JSR JSR LDA JSR LDA STA JSR LDA STA JSR LDA CMP BEQ INY	L TRYNEXT YSAV PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0 SPACE	WAS L=2 ALREADY? NO. YES, UNRECOGNIZED INST. PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '!' INITIALIZE PROMPT GET LINE. INIT SCREEN STUFF GET CHAR ASCII BLANK?
F582: F584: F588: F588: F587: F587: F597: F597: F597: F597: F597: F597: F594: F544: F546:	C6 F0 A4 98 AA 20 20 20 A9 85 20 20 20 C9 F0 C8 C9	44 35 D6 34 4A DE ED 3A A1 33 67 C7 00 A0 13 A4	F9 FD FF FD FF	ERR ERR2 RESETZ NXTLINE	DEC BEQ LDY TYA JSR LDA JSR LDA JSR JSR JSR JSR LDA STA JSR LDA STA JSR LDA STA JSR LDA STA JSR LDA STA ZSR JSR CMP BEQ	L TRYNEXT YSAV PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0 SPACE #\$A4	WAS L=2 ALREADY? NO. YES, UNRECOGNIZED INST. PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '!' INITIALIZE PROMPT GET LINE. INIT SCREEN STUFF GET CHAR ASCII BLANK? YES
F582: F584: F586: F588: F589: F587: F597:	C6 F0 A4 98 AA 20 20 20 20 20 20 20 20 20 20 C9 F0 C8 C9 F0	44 35 26 34 4A 26 50 3A 41 33 67 C7 00 A0 13 A4 92	F9 FD FF FD FF	ERR ERR2 RESETZ NXTLINE	DEC BEQ LDY TYA JSR LDA JSR LDA JSR JSR JSR JSR LDA STA JSR LDA STA JSR LDA STA JSR LDA STA JSR LDA STA ZSR JSR CMP BEQ	L TRYNEXT YSAV PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0 SPACE #\$A4 FAKEMON	WAS L=2 ALREADY? NO. YES, UNRECOGNIZED INST. 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
F582: F586: F588: F588: F589: F587: F592: F597: F597: F597: F597: F597: F597: F594: F542: F542: F544: F547:	C6 F0 A4 98 AA 20 20 20 20 20 20 20 20 20 20 C9 F0 C8 C9 F0 88	44 35 06 34 4A DE ED 3A A1 33 67 C7 00 A0 13 A4 92	F9 FD FF FD FF 02	ERR ERR2 RESETZ NXTLINE	DEC BEQ LDY TYA TAX JSR LDA JSR LDA STA JSR JSR LDA CMP BEQ LDA CMP BEQ DEY	L TRYNEXT YSAV PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0 SPACE #\$A4 FAKEMON	WAS L=2 ALREADY? NO. YES, UNRECOGNIZED INST. 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?
F582: F584: F588: F588: F588: F587: F592: F597:	C6 F0 A4 98 AA 20 20 20 20 A9 85 20 20 AD C9 F0 C8 C9 F0 88 20	44 35 06 34 4A DE ED 3A A1 33 67 C7 00 A0 13 A4 92 A7	F9 FD FF FD FF 02	ERR ERR2 RESETZ NXTLINE	DEC BEQ LDY TYA TAX JSR LDA JSR LDA JSR JSR JSR LDA CMP BEQ INY CMP BEQ DEY JSR	L TRYNEXT YSAV PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0 SPACE #\$A4 FAKEMON GETNUM	WAS L=2 ALREADY? NO. YES, UNRECOGNIZED INST. PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '!' INITIALIZE PROMPT GET LINE. INIT SCREEN STUFF GET CHAR ASCII '\$' IN COL 1? YES ASCII '\$' IN COL 1? YES, SIMULATE MONITOR NO, BACKUP A CHAR GET A NUMBER
F582: F584: F588: F588: F587: F587: F597:	C6 F0 A4 98 AA 20 20 A9 20 20 A9 85 20 20 AD C9 F0 C8 E9 F0 88 20 C9	44 35 06 34 4A ED 3A 41 33 67 C7 00 A0 13 A4 92 A7 93	F9 FD FF 02 FF	ERR ERR2 RESETZ NXTLINE	DEC BEQ LDY TYA JSR LDA JSR LDA JSR JSR JSR JSR LDA STA JSR LDA STA JSR LDA STA JSR JSR CMP BEQ DEY JSR CMP	L TRYNEXT YSAV PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0 SPACE #\$A4 FAKEMON GETNUM	WAS L=2 ALREADY? NO. YES, UNRECOGNIZED INST. 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
F582: F584: F588: F588: F587: F587: F597:	C6 F0 A4 98 AA 20 20 A9 20 20 A9 85 20 20 AD C9 F0 C8 88 20 C9 D0	44 35 D6 34 DE ED 3A A1 33 67 C7 00 A0 13 A4 92 A7 93 D5	F9 FD FF 02 FF	ERR2 RESETZ NXTLINE ERR4	DEC BEQ LDY TYA JSR LDA JSR LDA JSR JSR JSR JSR JSR LDA CMP BEQ INY CMP BEQ DEY JSR CMP BNE TXA	L TRYNEXT YSAV PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0 SPACE #\$A4 FAKEMON GETNUM #\$93 ERR2	WAS L=2 ALREADY? NO. YES, UNRECOGNIZED INST. 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.
F582: F584: F588: F588: F589: F587: F592: F597:	C6 F0 A4 98 AA 20 20 A9 20 20 A9 20 20 A0 20 A0 C9 F0 C8 88 20 C9 F0 88 20 C9 F0 88	44 35 D6 34 DE ED 3A A1 33 67 C7 00 A0 13 A4 92 A7 93 D5	F9 FD FF 02 FF	ERR2 RESETZ NXTLINE ERR4	DEC BEQ LDY TYA JSR LDA JSR LDA JSR JSR JSR JSR JSR LDA CMP BEQ INY CMP BEQ DEY JSR CMP BNE TXA	L TRYNEXT YSAV PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0 SPACE #\$A4 FAKEMON GETNUM #\$93 ERR2	WAS L=2 ALREADY? NO. YES, UNRECOGNIZED INST. 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.
F582: F584: F588: F588: F589: F587: F592: F597:	C6 F0 A4 98 AA 20 20 A9 20 20 A9 20 20 A0 20 A0 C9 F0 C8 88 20 C9 F0 88 20 C9 F0 88 F0 F0 88 F0 F0 F0 F0 F0 F0 F0 F0 F0 F0 F0 F0 F0	44 35 D6 34 ED 3A A1 33 67 C7 00 A0 13 A1 33 67 C7 00 0 0 2 3 A2 4 92 D5 D2	F9 FD FF 02	ERR2 RESETZ NXTLINE ERR4	DEC BEQ LDY TYA TAX JSR LDA JSR LDA STA JSR LDA CMP BEQ LDA CMP BEQ DEY JSR CMP BEQ DEY JSR CMP BEQ DEY JSR CMP BEQ DEY JSR CMP BEQ	L TRYNEXT YSAV PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0 SPACE #\$A4 FAKEMON GETNUM #\$93 ERR2 ERR2	WAS L=2 ALREADY? NO. YES, UNRECOGNIZED INST. PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '!' INITIALIZE PROMPT GET LINE. INIT SCREEN STUFF GET CHAR ASCII '\$' IN COL 1? YES ASCII '\$' IN COL 1? YES, SIMULATE MONITOR NO, BACKUP A CHAR GET A NUMBER ':' TERMINATOR?
F582: F584: F588: F588: F587: F587: F592: F597:	C6 F0 A4 98 AA 20 20 20 20 20 20 20 20 20 20 20 C9 F0 88 20 C9 F0 88 20 C9 D0 88 A F0 20 20 20 20 20 20 20 20 20 20 20 20 20	44 35 D6 34 ED 3A A1 33 67 C7 00 A0 13 A2 92 A7 93 D5 D2 78	F9 FD FF 02 FF	ERR2 RESETZ NXTLINE ERR4	DEC BEQ LDY TYA TAX JSR LDA JSR LDA STA JSR LDA STA JSR LDA CMP BEQ DEY JSR CMP BEQ DEY JSR CMP BEQ DEY JSR CMP BEQ JSR LDA LDA LDA LDA JSR LDA STA JSR LDA JSR LDA JSR LDA JSR LDA STA JSR LDA JSR LDA JSR LDA JSR LDA STA JSR LDA JSR LDA JSR LDA STA JSR LDA JSR LDA JSR LDA STA JSR LDA JSR LDA JSR LDA JSR LDA STA JSR LDA LDA LDA LDA LDA LDA LDA LDA LDA LDA	L TRYNEXT YSAV PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0 SPACE #\$A4 FAKEMON GETNUM #\$93 ERR2 ERR2 A1PCLP #\$3	WAS L=2 ALREADY? NO. YES, UNRECOGNIZED INST. 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.
F582: F584: F588: F588: F587: F587: F592: F597:	C6 F0 A4 98 AA 20 20 20 20 20 20 20 20 20 20 20 C9 F0 88 20 C9 F0 88 20 C9 D0 8A F0 20 20 20 A9 20 20 20 20 20 20 20 20 20 20 20 20 20	44 35 D6 34 ED 3A 33 67 C7 00 A0 13 A1 33 67 C7 00 A0 13 A2 92 A7 93 D5 D2 78 03	F9 FD FF 02 FF	ERR2 RESETZ NXTLINE ERR4	DEC BEQ LDY TYA TAX JSR LDA JSR LDA STA JSR LDA STA JSR LDA CMP BEQ DEY JSR CMP BEQ DEY JSR CMP BEQ DEY JSR CMP BEQ JSR LDA LDA LDA LDA JSR LDA STA JSR LDA JSR LDA JSR LDA JSR LDA STA JSR LDA JSR LDA JSR LDA JSR LDA STA JSR LDA JSR LDA JSR LDA STA JSR LDA JSR LDA JSR LDA STA JSR LDA JSR LDA JSR LDA JSR LDA STA JSR LDA LDA LDA LDA LDA LDA LDA LDA LDA LDA	L TRYNEXT YSAV PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0 SPACE #\$A4 FAKEMON GETNUM #\$93 ERR2 ERR2 A1PCLP #\$3	WAS L=2 ALREADY? NO. YES, UNRECOGNIZED INST. PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '!' INITIALIZE PROMPT GET LINE. INIT SCREEN STUFF GET CHAR ASCII '\$' IN COL 1? YES ASCII '\$' IN COL 1? 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.
F582: F584: F588: F588: F589: F587: F592: F597: F597: F597: F597: F597: F597: F597: F597: F597: F597: F597: F584: F584: F583: F584: F589: F588:	C6 F0 A4 98 A20 A9 20 20 A0 20 A0 C9 F0 C8 20 C9 F0 C8 20 C9 F0 C8 20 C9 F0 C8 20 C9 F0 C8 5 C9 S5 C9 C0 C9 S5 C0 C0 S5 C0 C0 S5 C0 C0 S5 C0 C0 C0 S5 C0 C0 C0 S5 C0 C0 C0 C0 C0 C0 C0 C0 C0 C0 C0 C0 C0	44 35 D6 34 DE ED 3A A1 33 67 C7 00 A0 13 A2 92 A7 93 D5 D2 78 03 3D	F9 FD FF 02 FF	ERR2 RESETZ NXTLINE ERR4 SPACE	DEC BEQ LDY TYA TXA JSR LDA JSR LDA STA JSR LDA STA JSR LDA STA MP BEQ DEY JSR CMP BEQ DEY JSR CMP BEQ DEY JSR CMP BEQ JSR CMP BEQ STA	L TRYNEXT YSAV PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0 SPACE #\$A4 FAKEMON GETNUM #\$93 ERR2 ERR2 AlPCLP #\$3 AlH	WAS L=2 ALREADY? NO. YES, UNRECOGNIZED INST. PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '!' INITIALIZE PROMPT GET LINE. INIT SCREEN STUFF GET CHAR ASCII '\$' IN COL 1? YES ASCII '\$' IN COL 1? 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.
F582: F584: F588: F588: F589: F587: F592: F597: F597: F597: F597: F597: F597: F597: F597: F597: F597: F597: F584: F584: F583: F584: F589: F588:	C6 F0 A4 98 AA 20 20 A9 20 20 A0 20 A0 C9 F0 88 20 C9 F0 88 20 C9 D0 85 20 20 85 20 20 20 20 20 20 20 20 20 20 20 20 20	44 35 D6 34 4A DE B3A A1 33 67 C7 00 A0 13 A1 92 A7 93 D5 D2 78 03 3D 34	F9 FD FF 02 FF FE FE	ERR2 RESETZ NXTLINE ERR4 SPACE NXTMN	DEC BEQ LDY TYA JSR LDA JSR LDA JSR LDA STA JSR LDA STA JSR CMP BEQ DEY JSR CMP BEQ DEY JSR CMP BEQ DEY JSR CMP BEQ JSR CMP BEQ STA JSR JSR LDA JSR LDA JSR JSR JSR JSR JSR JSR JSR JSR JSR JSR	L TRYNEXT YSAV PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0 SPACE #\$A4 FAKEMON GETNUM #\$93 ERR2 ERR2 A1PCLP #\$3 A1H GETNSP	WAS L=2 ALREADY? NO. YES, UNRECOGNIZED INST. 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
F582: F584: F588: F588: F589: F587: F592: F597: F597: F597: F597: F597: F597: F597: F597: F597: F597: F597: F597: F597: F597: F597: F597: F597: F584: F581: F584: F586: F589: F580: F589:	C6 F0 A4 98 AA 20 A9 20 A9 20 A0 20 A0 20 A0 C9 F0 88 20 C9 D0 88 C9 D0 88 F0 88 20 C9 D0 80 C9 C9 C9 C9 C0 C9 C0 C9 C0 C9 C0 C0 C9 C0 C0 C0 C0 C0 C0 C0 C0 C0 C0 C0 C0 C0	44 35 D6 34 ADE ED 3A A1 33 67 C7 00 A0 13 A2 92 A7 93 D5 D2 78 03 3D 34	F9 FD FF 02 FF FE F6	ERR2 RESETZ NXTLINE ERR4 SPACE NXTMN	DEC BEQ LDY TYA JSR LDA JSR LDA JSR JSR JSR LDA STA JSR CMP BEQ DEY JSR CMP BEQ DEY JSR CMP BEQ DEY JSR LDA STA STA STA STA STA STA STA	L TRYNEXT YSAV PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0 SPACE #\$A4 FAKEMON GETNUM #\$93 ERR2 ERR2 ERR2 A1HPCLP #\$3 A1H GETNSP A	WAS L=2 ALREADY? NO. YES, UNRECOGNIZED INST. 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
F582: F584: F588: F588: F589: F587: F592: F597: F597: F597: F597: F597: F597: F597: F597: F597: F597: F597: F542: F543: F544: F584: F584: F584: F585:	C6 F0 A4 98 A0 20 20 A9 20 20 A9 85 20 20 A0 C9 F0 88 C9 F0 88 C9 D0 88 F0 20 88 F0 20 88 F0 20 C9 F0 88 F0 80 80 F0 80 80 F0 80 80 80 80 80 80 80 80 80 80 80 80 80	44 35 D6 34 DE ED 3A A1 33 67 C7 00 A0 13 A2 92 A7 93 D5 D2 78 3D 25 BE	F9 FD FF 02 FF FE F6	ERR2 RESETZ NXTLINE ERR4 SPACE NXTMN NXTM	DEC BEQ LDY TYA TAX JSR LDA JSR JSR JSR JSR JSR LDA STA JSR LDA CMP BEQ INY CMP BEQ DEY JSR CMP BEQ DEY JSR LDA STA JSR SEC	L TRYNEXT YSAV PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0 SPACE #\$A4 FAKEMON GETNUM #\$93 ERR2 ERR2 A1PCLP #\$3 A1H GETNSP A #\$BE	WAS L=2 ALREADY? NO. YES, UNRECOGNIZED INST. 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.
F582: F584: F588: F588: F587: F587: F597: F597: F597: F597: F597: F597: F597: F597: F597: F597: F542: F542: F542: F543: F543: F543: F544:	C6 F0 A4 98 AA 20 20 A9 20 20 A9 20 20 A9 520 C9 F0 88 C9 F0 88 C9 F0 88 F0 20 80 F0 20 C9 F0 C9 F0 C9 F0 C9 F0 C9 C9 C9 C9 C9 C0 C9 C9 C0 C9 C0 C9 C0 C9 C0 C9 C0 C9 C0 C9 C0 C0 C9 C0 C0 C9 C0 C0 C9 C0 C0 C0 C0 C0 C0 C0 C0 C0 C0 C0 C0 C0	44 35 D6 34 4DE ED 3A A1 33 67 C7 00 A0 13 A4 92 A7 93 D5 D2 78 03 30 4 BE C2	F9 FD FF 02 FF FE F6	ERR2 RESETZ NXTLINE ERR4 SPACE NXTMN NXTM	DEC BEQ LDY TYA TAX JSR LDA JSR JSR JSR JSR JSR LDA STA JSR LDA CMP BEQ INY CMP BEQ DEY JSR CMP BEQ DEY JSR LDA STA JSR SEC	L TRYNEXT YSAV PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0 SPACE #\$A4 FAKEMON GETNUM #\$93 ERR2 ERR2 A1PCLP #\$3 A1H GETNSP A #\$BE	WAS L=2 ALREADY? NO. YES, UNRECOGNIZED INST. 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.
F582: F584: F588: F588: F587: F587: F592: F597: F597: F597: F597: F597: F597: F597: F597: F597: F597: F597: F584: F584: F584: F584: F584: F584: F585:	C6 F0 A4 98 A20 20 20 A9 20 A0 20 A0 20 A0 C9 C9 B8 20 C9 B8 20 C9 B8 20 C9 B8 20 C9 B8 C9 C9 B8 C9 C9 B5 C0 C9 C9 C9 C0 C9 C9 C0 C9 C9 C0 C0 C0 C0 C0 C0 C0 C0 C0 C0 C0 C0 C0	44 35 D6 34 4A DE ED 3A A1 33 67 C7 00 A0 13 A2 93 D5 D2 78 3D 278 3D 278 3D 234 BE C2 C1	F9 FD FF 02 FF FE F6	ERR2 RESETZ NXTLINE ERR4 SPACE NXTMN NXTM	DEC BEQ LDY TYA JSR LDA JSR LDA JSR LDA STA JSR LDA CMP BEQ DEY JSR CMP BEQ DEY JSR CMP BEQ JSR CMP BEQ JSR LDA STA JSR CMP BEQ JSR CMP BEQ SSC CMP BEQ STA	L TRYNEXT YSAV PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0 SPACE #\$A4 FAKEMON GETNUM #\$93 ERR2 ERR2 AlPCLP #\$3 AlPCLP #\$3 AlH GETNSP A #\$BE #\$2 ERR2 ERR2 ERR2 AL	WAS L=2 ALREADY? NO. YES, UNRECOGNIZED INST. 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.
F582: F584: F588: F588: F589: F587: F592: F597: F507:	C6 F0 A4 98 A20 20 20 A9 20 A0 20 A0 20 A0 C9 C9 C0 88 20 C9 D0 85 20 A9 E9 00 85 20 00 A9 C9 C9 C9 C9 C9 C9 C9 C9 C9 C9 C9 C9 C9	44 35 D6 34 4A DE ED 3A 33 67 C7 00 A0 13 A1 92 A7 93 D5 D2 78 03 3D 34 BE C2 C1	F9 FD FF 02 FF FE F6	ERR2 RESETZ NXTLINE ERR4 SPACE NXTMN NXTM	DEC BEQ LDY TYA TYA JSR LDA JSR LDA JSR JSR JSR LDA STA JSR CMP BEQ DEY JSR CMP BEQ DEY JSR CMP BEQ JSR CMP BNE TXA BEQ JSR LDA STA STA STA STA STA STA STA STA STA	L TRYNEXT YSAV 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 A	WAS L=2 ALREADY? NO. YES, UNRECOGNIZED INST. PRINT ^ UNDER LAST READ CHAR TO INDICATE ERROR POSITION. '!' INITIALIZE PROMPT GET LINE. INIT SCREEN STUFF GET CHAR 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?
F582: F584: F586: F588: F589: F592: F597: F597: F597: F597: F597: F597: F597: F597: F597: F597: F597: F597: F597: F584: F584: F581: F584: F584: F584: F585:	C6 F0 A4 98 A0 20 20 A9 20 20 A9 20 A0 20 A0 50 C9 88 20 20 A0 50 C9 88 C9 88 C0 80 A0 20 A0 A0 20 A0 A0 20 A0 A0 C0 A0 A0 C0 A0 C0 A0 C0 A0 C0 A0 C0 A0 C0 C0 A0 C0 C0 A0 C0 A0 C0 A0 C0 A0 C0 A0 C0 C0 A0 C0 C0 A0 C0 C0 A0 C0 A0 C0 C0 A0 C0 C0 A0 C0 A0 C0 C0 C0 C0 C0 C0 C0 C0 C0 C0 C0 C0 C0	44 35 D6 34 ADE ED 3A 33 67 C7 00 A0 33 A1 33 67 C7 00 A0 33 A1 33 A2 93 D5 D2 78 30 32 BE C2 C1	F9 FD FF F02 FF FE	ERR2 RESETZ NXTLINE ERR4 SPACE NXTMN NXTM	DEC BEQ LDY TYA JSR LDA JSR LDA JSR LDA JSR JSR JSR LDA STA JSR CMP BEQ DEY JSR CMP BEQ DEY JSR CMP BEQ DEY JSR CMP BEQ DEX STA SEC CMP BNE TXA STA STA STA STA STA STA STA STA STA ST	L TRYNEXT YSAV PRBL2 #\$DE COUT BELL #\$A1 PROMPT GETLNZ ZMODE IN #\$A0 SPACE #\$A4 FAKEMON GETNUM #\$93 ERR2 ERR2 A1PCLP #\$3 A1H GETNSP A #\$BE #\$BE #\$C2 ERR2 A A	WAS L=2 ALREADY? NO. YES, UNRECOGNIZED INST. 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.
F582: F584: F586: F588: F587: F592: F597: F597: F597: F597: F597: F597: F597: F597: F597: F597: F597: F597: F597: F597: F584: F584: F584: F584: F585:	C6 F0 A4 98 A0 20 A9 20 A9 20 A0 20 A0 C9 F0 820 C9 F0 820 C9 F0 820 C9 C9 820 A9 520 C9 C9 C0 A9 C9 C0 A9 C0 A0 A9 C0 A0 C9 C0 A0 C9 C0 C0 C0 C0 C0 C0 C0 C0 C0 C0 C0 C0 C0	44 35 06 34 4A DE ED 3A 33 67 C70 A0 13 A4 92 A7 93 D5 D2 78 03 3D 4 EC2 C1 04	F9 FD FF O2 FF FE F6	ERR2 RESETZ NXTLINE ERR4 SPACE NXTMN NXTM	DEC BEQ LDY TYA TAX JSR LDA JSR LDA STA JSR LDA CMP BEQ INY BEQ INY BEQ DEY JSR CMP BEQ DEY JSR CMP BEQ JSR CMP BEQ JSR LDA STA STA LDA STA LDA STA LDA STA LDA STA LDA 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 STA LDA STA JSR LDA STA LDA STA JSR LDA STA LDA STA JSR LDA STA STA STA STA STA STA STA STA STA ST	L TRYNEXT YSAV 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 A H\$SE #\$C2 ERR2 A A #\$4	WAS L=2 ALREADY? NO. YES, UNRECOGNIZED INST. 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.

F5CC:					ROL	A4L	
F5CE:					ROL	A4H	
F5D0:	CA				DEX		
F5D1:	10	F8			BPL	NXTM2	
F5D3:	C6	3D			DEC	AlH	DONE WITH 3 CHARS?
F5D5:	FO	F4			BEQ	NXTM2	YES, BUT DO 1 MORE SHIFT
F5D7:	10	E4			BPL	NXTMN	NO
F5D9:	A2	05		FORM1	LDX	#\$5	5 CHARS IN ADDR MODE
F5DB:	20	34	F6	FORM2	JSR	GETNSP	GET FIRST CHAR OF ADDR
F5DE:						YSAV	
F5E0:			F9		CMP	CHAR1,X	FIRST CHAR MATCH PATTERN?
F5E3:						FORM3	NO
F5E5:			F6			GETNSP	YES, GET SECOND CHAR
F5E8:						CHAR2,X	MATCHES SECOND HALF?
F5EB:			гэ			FORM5	YES.
F5ED:			PO			CHAR2,X	
			гэ				NO, IS SECOND HALF ZERO?
F5F0:					~	FORM4	YES.
F5F2:						#\$A4	NO, SECOND HALF OPTIONAL?
F5F4:					~	FORM4	YES.
F5F6:					LDY	YSAV	
F5F8:				FORM3	CLC		CLEAR BIT-NO MATCH
F5F9:	88			FORM4	DEY		BACK UP 1 CHAR
F5FA:				FORM5	ROL	FMT	FORM FORMAT BYTE
F5FC:	Ε0	03			CPX	#\$3	TIME TO CHECK FOR ADDR.
F5FE:	D0	0D			BNE	FORM7	NO
F600:	20	A7	FF		JSR	GETNUM	YES
F603:	A5	3F			LDA	A2H	
F605:	FO	01			BEO	FORM6	HIGH-ORDER BYTE ZERO
F607:	E8				INX		NO, INCR FOR 2-BYTE
F608:				FORM6		L	STORE LENGTH
F60A:				1 01010			RELOAD FORMAT INDEX
F60C:					DEY	πçσ	BACKUP A CHAR
F60D:				FORM7	STX	AlH	SAVE INDEX
F60F:				FORM/		AIN	DONE WITH FORMAT CHECK?
					DEX	DODWO	
F610:						FORM2	NO.
F612:						FMT	YES, PUT LENGTH
F614:					ASL	A	IN LOW BITS
F615:						A	
F616:					ORA	L	
F618:					CMP	#\$20	
F61A:	В0	06			BCS	FORM8	ADD "\$" IF NONZERO LENGTH
F61C:	A6	35			LDX	L	AND DON'T ALREADY HAVE IT
F61E:	FO	02			BEQ	FORM8	
F620:	09	80			ORA	#\$80	
F622:	85	44		FORM8	STA	FMT	
F624:						YSAV	
F626:			~ ~		LDA	IN,Y	
F629:	B9	00	02				GET NEXT NONBLANK
			02				GET NEXT NONBLANK
	C9	BB	02		CMP	#\$BB	'' START OF COMMENT?
F62B:	C9 F0	BB 04	02		CMP BEQ	#\$BB FORM9	'' START OF COMMENT? YES
F62B: F62D:	C9 F0 C9	BB 04 8D	02		CMP BEQ CMP	#\$BB FORM9 #\$8D	'' START OF COMMENT? YES CARRIAGE RETURN?
F62B: F62D: F62F:	C9 F0 C9 D0	BB 04 8D 80		EODMO	CMP BEQ CMP BNE	#\$BB FORM9 #\$8D ERR4	'' START OF COMMENT? YES
F62B: F62D: F62F: F631:	C9 F0 C9 D0 4C	BB 04 8D 80 5C	F5		CMP BEQ CMP BNE JMP	#\$BB FORM9 #\$8D ERR4 TRYNEXT	'' START OF COMMENT? YES CARRIAGE RETURN?
F62B: F62D: F62F: F631: F634:	C9 F0 C9 D0 4C B9	BB 04 8D 80 5C	F5	FORM9 GETNSP	CMP BEQ CMP BNE JMP LDA	#\$BB FORM9 #\$8D ERR4	'' START OF COMMENT? YES CARRIAGE RETURN?
F62B: F62D: F62F: F631: F634: F637:	C9 F0 C9 D0 4C B9 C8	BB 04 8D 80 5C 00	F5		CMP BEQ CMP BNE JMP LDA INY	#\$BB FORM9 #\$8D ERR4 TRYNEXT IN,Y	'' START OF COMMENT? YES CARRIAGE RETURN? NO, ERR.
F62B: F62D: F62F: F631: F634: F637: F638:	C9 F0 C9 D0 4C B9 C8 C9	BB 04 8D 80 5C 00 A0	F5		CMP BEQ CMP BNE JMP LDA INY CMP	#\$BB FORM9 #\$8D ERR4 TRYNEXT IN,Y #\$A0	'' START OF COMMENT? YES CARRIAGE RETURN?
F62B: F62D: F62F: F631: F634: F637: F638: F63A:	C9 F0 C9 D0 4C B9 C8 C9 F0	BB 04 8D 80 5C 00 A0 F8	F5		CMP BEQ CMP BNE JMP LDA INY CMP BEQ	#\$BB FORM9 #\$8D ERR4 TRYNEXT IN,Y	'' START OF COMMENT? YES CARRIAGE RETURN? NO, ERR.
F62B: F62D: F62F: F631: F634: F637: F638:	C9 F0 C9 D0 4C B9 C8 C9 F0	BB 04 8D 80 5C 00 A0 F8	F5		CMP BEQ CMP BNE JMP LDA INY CMP BEQ RTS	#\$BB FORM9 #\$8D ERR4 TRYNEXT IN,Y #\$A0 GETNSP	'' START OF COMMENT? YES CARRIAGE RETURN? NO, ERR.
F62B: F62D: F62F: F631: F634: F637: F638: F63A: F63C:	C9 F0 D0 4C B9 C8 C9 F0 60	BB 04 8D 5C 00 F8	F5 02	GETNSP	CMP BEQ CMP BNE JMP LDA INY CMP BEQ RTS ORG	#\$BB FORM9 #\$8D ERR4 TRYNEXT IN,Y #\$A0 GETNSP \$F666	'' START OF COMMENT? YES CARRIAGE RETURN? NO, ERR.
F62B: F62D: F62F: F631: F634: F637: F638: F63A: F63C:	C9 F0 D0 4C B9 C8 C9 F0 60	BB 04 8D 5C 00 F8	F5 02		CMP BEQ CMP BNE JMP LDA INY CMP BEQ RTS ORG	#\$BB FORM9 #\$8D ERR4 TRYNEXT IN,Y #\$A0 GETNSP \$F666	'' START OF COMMENT? YES CARRIAGE RETURN? NO, ERR.

	*********	*****	* * * * * * * *	
	*		*	
	* APPLE-II	I FLOA	ATING *	
	* POINT H	ROUTI	NES *	
	*		*	
	* COPYRIGH	HT 19'	77 BY *	
	* APPLE CON	MPUTER	R INC. *	
	*		*	
	* ALL RIGHT	rs res	SERVED *	
	*		*	
	* S.WC	DZNIAI	K *	
	*		*	

	TITLE "FLO			ROUTINES"
		EPZ : EPZ :		
		EPZ S		
	OVLOC H			
		ORG S		
F425: 18		CLC		CLEAR CARRY
F426: A2 02			#\$2	INDEX FOR 3-BYTE ADD.
F428: B5 F9	ADD1 I	LDA N	M1,X	
F42A: 75 F5	1	ADC N	M2,X	ADD A BYTE OF MANT2 TO MANT1
F42C: 95 F9	S	STA I	M1,X	
F42E: CA	I	DEX		INDEX TO NEXT MORE SIGNIF. BYTE.
F42F: 10 F7	E	BPL A	ADD1	LOOP UNTIL DONE.
F431: 60		RTS		RETURN
	MD1 A	ASL S	SIGN	CLEAR LSB OF SIGN.
F434: 20 37 F4				ABS VAL OF M1, THEN SWAP WITH M2
F437: 24 F9	ABSWAP H	BIT N	41	MANT1 NEGATIVE?
F439: 10 05				NO, SWAP WITH MANT2 AND RETURN.
F43B: 20 A4 F4		JSR I	FCOMPL	YES, COMPLEMENT IT. INCR SIGN, COMPLEMENTING LSB.
F43E: E6 F3 F440: 38	ABSWAP1 S	INC S		SET CARRY FOR RETURN TO MUL/DIV.
F440: 38 F441: A2 04	SWAP I	ידר אטיי		INDEX FOR 4 BYTE SWAP.
	SWAP1 S	STV I	#\$4 E-1,X	INDER FOR 4 DITE SWAF.
F445: B5 F7				SWAP A BYTE OF EXP/MANT1 WITH
F447: B4 F3				EXP/MANT2 AND LEAVE A COPY OF
F449: 94 F7				MANT1 IN E (3 BYTES). E+3 USED
F44B: 95 F3			X2-1,X	
F44D: CA		DEX		ADVANCE INDEX TO NEXT BYTE
F44E: D0 F3	H	BNE S	SWAP1	LOOP UNTIL DONE.
F450: 60	F	RTS		RETURN
F451: A9 8E	FLOAT I	LDA ‡	#\$8E	INIT EXP1 TO 14,
F453: 85 F8	5	STA 2	X1	THEN NORMALIZE TO FLOAT.
F455: A5 F9				HIGH-ORDER MANT1 BYTE.
F457: C9 C0				UPPER TWO BITS UNEQUAL?
F459: 30 OC				YES, RETURN WITH MANT1 NORMALIZED
F45B: C6 F8		DEC 2		DECREMENT EXP1.
F45D: 06 FB		ASL N		
F45F: 26 FA			M1+1	SHIFT MANT1 (3 BYTES) LEFT.
F461: 26 F9		ROL I		EVD1 ZEDOD
F463: A5 F8 F465: D0 EE				EXP1 ZERO? NO, CONTINUE NORMALIZING.
F465: DU EE F467: 60		RTS		RETURN.
F467: 80 F468: 20 A4 F4				CMPL MANT1, CLEARS CARRY UNLESS 0
F468: 20 A4 F4 F46B: 20 7B F4				RIGHT SHIFT MANTI OR SWAP WITH
F46E: A5 F4			X2	
F470: C5 F8		CMP 2		COMPARE EXP1 WITH EXP2.
F472: D0 F7				IF #,SWAP ADDENDS OR ALIGN MANTS.
F474: 20 25 F4		JSR /		ADD ALIGNED MANTISSAS.
		BVC 1		NO OVERFLOW, NORMALIZE RESULT.
F479: 70 05	I	BVS I	RTLOG	OV: SHIFT M1 RIGHT, CARRY INTO SIGN

	NLONGUD DOG		
F47B: 90 C4	ALGNSWP BCC * ELSE		SWAP IF CARRY CLEAR, T ARITH.
F47D: A5 F9			SIGN OF MANT1 INTO CARRY FOR
F47F: 0A			RIGHT ARITH SHIFT.
F480: E6 F8			INCR X1 TO ADJUST FOR RIGHT SHIFT
F482: F0 75 F484: A2 FA	RTLOG1 LDX		EXP1 OUT OF RANGE. INDEX FOR 6:BYTE RIGHT SHIFT.
F486: 76 FF			INDER FOR U.DITE RIGHT SHIFT.
F488: E8	INX		NEXT BYTE OF SHIFT.
F489: D0 FB			LOOP UNTIL DONE.
F48B: 60	RTS		RETURN.
F48C: 20 32 F4	FMUL JSR	MD1	ABS VAL OF MANT1, MANT2
F48F: 65 F8 F491: 20 E2 F4	ADC	X1	ADD EXP1 TO EXP2 FOR PRODUCT EXP CHECK PROD. EXP AND PREP. FOR MUL
F494: 18	CLC		CLEAR CARRY FOR FIRST BIT.
			M1 AND E RIGHT (PROD AND MPLIER)
F498: 90 03	BCC	MUL2	IF CARRY CLEAR, SKIP PARTIAL PROD
F49A: 20 25 F4	JSR	ADD	ADD MULTIPLICAND TO PRODUCT.
F49D: 88			NEXT MUL ITERATION.
F49E: 10 F5 F4A0: 46 F3	MDEND I GE	MUL1	LOOP UNTIL DONE.
F4A2: 90 BF	NORMX BCC	NORM	TEST SIGN LSB. IF EVEN,NORMALIZE PROD,ELSE COMP
F4A4: 38	FCOMPL SEC		SET CARRY FOR SUBTRACT.
F4A5: A2 03	LDX	#\$3	INDEX FOR 3 BYTE SUBTRACT.
F4A7: A9 00		⊾ #\$O	CLEAR A.
F4A9: F5 F8		X1,X	SUBTRACT BYTE OF EXP1.
F4AB: 95 F8 F4AD: CA	DEX		RESTORE IT. NEXT MORE SIGNIFICANT BYTE.
F4AE: D0 F7			LOOP UNTIL DONE.
F4B0: F0 C5	BEC	ADDEND	NORMALIZE (OR SHIFT RT IF OVFL).
F4B2: 20 32 F4	FDIV JSF	MD1	TAKE ABS VAL OF MANT1, MANT2. SUBTRACT EXP1 FROM EXP2.
F4B5: E5 F8	SBC	X1	SUBTRACT EXP1 FROM EXP2.
F4B7: 20 E2 F4	JSR	MD2	SAVE AS QUOTIENT EXP.
F4BA: 38 F4BB: A2 02			SET CARRY FOR SUBTRACT. INDEX FOR 3-BYTE SUBTRACTION.
F4BB: A2 02 F4BD: B5 F5			INDEX FOR 3-BILE SUBTRACTION.
F4BF: F5 FC			SUBTRACT A BYTE OF E FROM MANT2.
F4C1: 48			SAVE ON STACK.
F4C2: CA	DEX		NEXT MORE SIGNIFICANT BYTE.
F4C3: 10 F8			LOOP UNTIL DONE.
F4C5: A2 FD			INDEX FOR 3-BYTE CONDITIONAL MOVE
F4C7: 68 F4C8: 90 02	DIV3 PLA		PULL BYTE OF DIFFERENCE OFF STACK IF M2 <e don't="" m2.<="" restore="" td="" then=""></e>
F4CA: 95 F8		M2+3,X	IF M2CE INEN DON I RESIONE M2.
F4CC: E8	DIV4 INX		NEXT LESS SIGNIFICANT BYTE.
F4CD: D0 F8	BNE	DIV3	LOOP UNTIL DONE.
F4CF: 26 FB		M1+2	
F4D1: 26 FA			ROLL QUOTIENT LEFT, CARRY INTO LSB
F4D3: 26 F9 F4D5: 06 F7		M1 M2+2	
F4D7: 26 F6			SHIFT DIVIDEND LEFT
F4D9: 26 F5		M2	
F4DB: B0 1C	BCS	OVFL	OVFL IS DUE TO UNNORMED DIVISOR
F4DD: 88	DEY		NEXT DIVIDE ITERATION.
F4DE: D0 DA			LOOP UNTIL DONE 23 ITERATIONS.
F4E0: F0 BE F4E2: 86 FB		MDEND M1+2	NORM. QUOTIENT AND CORRECT SIGN.
F4E4: 86 FA			CLEAR MANT1 (3 BYTES) FOR MUL/DIV.
F4E6: 86 F9		M1	
F4E8: B0 0D			IF CALC. SET CARRY, CHECK FOR OVFL
F4EA: 30 04			IF NEG THEN NO UNDERFLOW.
F4EC: 68 F4ED: 68	PLA PLA		POP ONE RETURN LEVEL.
F4EE: 90 B2			CLEAR X1 AND RETURN.
			COMPLEMENT SIGN BIT OF EXPONENT.
F4F2: 85 F8		X1	STORE IT.
F4F4: A0 17			COUNT 24 MUL/23 DIV ITERATIONS.
F4F6: 60	RTS		RETURN.
F4F7: 10 F7 F4F9: 4C F5 03		MD3 OVLOC	IF POSITIVE EXP THEN NO OVFL.
r4r9; 40 P5 U3		\$F63D	
F63D: 20 7D F4		RTAR	
		X1	
F642: 10 13		UNDFL	
F644: C9 8E		#\$8E	
F646: D0 F5 F648: 24 F9		FIX1 M1	
F648: 24 F9 F64A: 10 0A		FIXRTS	
F64C: A5 FB		M1+2	
F64E: F0 06		FIXRTS	
F650: E6 FA		M1+1	
F652: D0 02		FIXRTS	
F654: E6 F9 F656: 60		M1	
		⊈#\$0	
F659: 85 F9		. M1	
F65B: 85 FA		M1+1	
F65D: 60	RTS		
		95	

	*******	*****	****	
	*		*	
	* APPLE	-II PS	SEUDO *	
	* MACHINE	INTER	RPRETER *	
	*		*	
	* COPYR			
	* APPLE C	OMPUTE	ER INC *	
	*		*	
	* ALL RIG			
	° Б. *	WOZNIA	* .	
	*******	*****	*******	
	TITLE "SW	EET16	INTERPRETER	R."
		EQU		
	ROH	EQU	\$1	
	R14H	EQU	\$1D	
			\$1E	
	R15H			
	SW16PAG			
	SAVE			
	RESTORE			
F689: 20 4A FF		ORG	SAVE	PRESERVE 6502 REG CONTENTS
F68C: 68		PLA	DAVE	PRESERVE 0502 REG CONTENTS
F68D: 85 1E			R15L	INIT SWEET16 PC
F68F: 68		PLA		FROM RETURN
F690: 85 1F		STA	R15H	ADDRESS
F692: 20 98 F6	SW16B	JSR	SW16C	INTERPRET AND EXECUTE
F695: 4C 92 F6		JMP	SW16B	ONE SWEET16 INSTR.
F698: E6 1E	SW16C			
F69A: D0 02			SW16D	INCR SWEET16 PC FOR FETCH
F69C: E6 1F		INC		
F69E: A9 F7 F6A0: 48		dda PHA	#SW16PAG	PUSH ON STACK FOR RTS
F6A1: A0 00		ldy Ldy	#\$0	PUSH ON STACK FOR RIS
F6A3: B1 1E				FETCH INSTR
F6A5: 29 OF		AND		MASK REG SPECIFICATION
F6A7: 0A		ASL		DOUBLE FOR TWO BYTE REGISTERS
F6A8: AA		TAX		TO X REG FOR INDEXING
F6A9: 4A			A	
F6AA: 51 1E				NOW HAVE OPCODE
F6AC: F0 OB		~	TOBR	IF ZERO THEN NON-REG OP
F6AE: 86 1D			R14H	INDICATE'PRIOR RESULT REG'
F6B0: 4A F6B1: 4A		LSR LSR	A	OPCODE*2 TO LSB'S
F6B2: 4A		LSR		OFCODE"2 TO LISE 5
F6B3: A8		TAY		TO Y REG FOR INDEXING
F6B4: B9 E1 F6			OPTBL-2,Y	LOW ORDER ADR BYTE
F6B7: 48		PHA		ONTO STACK
F6B8: 60	:	RTS		GOTO REG-OP ROUTINE
F6B9: E6 1E	TOBR	INC	R15L	
F6BB: D0 02			TOBR2	INCR PC
F6BD: E6 1F			R15H	
F6BF: BD E4 F6			BRTBL,X	LOW ORDER ADR BYTE
F6C2: 48 F6C3: A5 1D		PHA LDA	R14H	ONTO STACK FOR NON-REG OP 'PRIOR RESULT REG' INDEX
F6C3: A5 1D F6C5: 4A		LDA LSR	A A	PREPARE CARRY FOR BC, BNC.
F6C6: 60		RTS		GOTO NON-REG OP ROUTINE
F6C7: 68		PLA		POP RETURN ADDRESS
F6C8: 68		PLA		
F6C9: 20 3F FF			RESTORE	RESTORE 6502 REG CONTENTS
F6CC: 6C 1E 00			(R15L)	RETURN TO 6502 CODE VIA PC
F6CF: B1 1E	SETZ	LDA	(R15L),Y	HIGH-ORDER BYTE OF CONSTANT

F6D1: 95 01				
		STA	ROH,X	
F6D3: 88 F6D4: B1 1E		DEY	(DIET) V	LOW-ORDER BYTE OF CONSTANT
F6D6: 95 00			ROL,X	LOW-ORDER BITE OF CONSTANT
F6D8: 98		TYA		Y-REG CONTAINS 1
F6D9: 38		SEC		
F6DA: 65 1E			R15L	ADD 2 TO PC
F6DC: 85 1E			R15L	
F6DE: 90 02		BCC	SET2	
F6E0: E6 1F		INC	R15H	
F6E2: 60		RTS		
F6E3: 02	OPTBL	DFB	SET-1	1X
F6E4: F9	BRTBL	DFB	RTN-1	0
F6E5: 04				2X
F6E6: 9D				1
F6E7: 0D			ST-1	3X
F6E8: 9E			BNC-1	2
F6E9: 25 F6EA: AF				4X
F6EB: 16		DFB		3 5X
F6EC: B2			BP-1	4
F6ED: 47			LDDAT-1	
F6EE: B9			BM-1	5
F6EF: 51			STDAT-1	
F6F0: C0			BZ-1	6
F6F1: 2F				8X
F6F2: C9	:	DFB	BNZ-1	7
F6F3: 5B	:	DFB	STPAT-1	9X
F6F4: D2	:	DFB	BM1-1	8
F6F5: 85	1	DFB	ADD-1	AX
F6F6: DD			BNM1-1	9
F6F7: 6E	:			BX
F6F8: 05				A
F6F9: 33				CX
F6FA: E8				B
F6FB: 70				DX
F6FC: 93				C
F6FD: 1E F6FE: E7				EX D
F6FF: 65				FX
F700: E7				E
F701: E7				UNUSED
F702: E7				F
				ALWAYS TAKEN
			ROL,X	
	BK	EQU	*-1	
F707: 85 00		STA	ROL	
F709: B5 01		LDA	ROH,X	MOVE RX TO RO
F70B: 85 01		STA	ROH	
F70D: 60		RTS		
F70E: A5 00		LDA		
F710: 95 00			ROL,X	MOVE RO TO RX
F712: A5 01		LDA	ROL,X ROH	MOVE R0 TO RX
F712: A5 01 F714: 95 01		LDA STA	ROL,X ROH	MOVE RO TO RX
F712: A5 01 F714: 95 01 F716: 60		LDA STA RTS	ROL,X ROH ROH,X	MOVE RO TO RX
F712: A5 01 F714: 95 01 F716: 60 F717: A5 00	STAT	LDA STA RTS LDA	ROL,X ROH ROH,X ROL	
F712: A5 01 F714: 95 01 F716: 60 F717: A5 00 F719: 81 00	STAT STAT2	LDA STA RTS LDA STA	ROL,X ROH ROH,X ROL (ROL,X)	MOVE RO TO RX STORE BYTE INDIRECT
F712: A5 01 F714: 95 01 F716: 60 F717: A5 00 F719: 81 00	STAT STAT2	LDA STA RTS LDA STA	ROL,X ROH ROH,X ROL (ROL,X)	STORE BYTE INDIRECT
F712: A5 01 F714: 95 01 F716: 60 F717: A5 00 F719: 81 00	STAT STAT2	LDA STA RTS LDA STA	ROL,X ROH ROH,X ROL (ROL,X)	
F712: A5 01 F714: 95 01 F716: 60 F717: A5 00 F719: 81 00	STAT STAT2 STAT3 INR	LDA STA RTS LDA STA LDY STY INC	ROL,X ROH ROH,X ROL (ROL,X) #\$0 R14H ROL,X	STORE BYTE INDIRECT
F712: A5 01 F714: 95 01 F716: 60 F717: A5 00 F719: 81 00 F71B: A0 00 F71B: A0 00 F71D: 84 1D F71F: F6 00 F721: D0 02 F723: E6 01	STAT STAT2 STAT3 INR	LDA STA RTS LDA STA LDY STY INC BNE	ROL,X ROH ROH,X ROL (ROL,X) #\$0 R14H ROL,X	STORE BYTE INDIRECT INDICATE R0 IS RESULT NEG
F712: A5 01 F714: 95 01 F716: 60 F717: A5 00 F719: 81 00 F719: 84 00 F7110: 84 1D F71F: F6 00 F721: D0 02 F723: F6 01 F725: 60	STAT STAT2 STAT3 INR INR2	LDA STA RTS LDA STA LDY STY INC BNE INC RTS	ROL,X ROH ROH,X ROL (ROL,X) #\$0 R14H ROL,X INR2 ROH,X	STORE BYTE INDIRECT INDICATE R0 IS RESULT NEG INCR RX
F712: A5 01 F714: 95 01 F716: 60 F717: A5 00 F719: 81 00 F719: 84 00 F7110: 84 1D F71F: F6 00 F721: D0 02 F723: F6 01 F725: 60	STAT STAT2 STAT3 INR INR2	LDA STA RTS LDA STA LDY STY INC BNE INC RTS	ROL,X ROH ROH,X ROL (ROL,X) #\$0 R14H ROL,X INR2 ROH,X	STORE BYTE INDIRECT INDICATE R0 IS RESULT NEG INCR RX
F712: A5 01 F714: 95 01 F716: 60 F717: A5 00 F719: 81 00 F718: A0 00 F71B: 84 1D F71F: F6 00 F721: D0 02 F721: D0 02 F722: F6 01 F725: 60 F726: A1 00 F728: 85 00	STAT STAT2 STAT3 INR INR2 LDAT	LDA STA RTS LDA STA LDY STY INC BNE INC RTS LDA	ROL,X ROH ROH,X ROL (ROL,X) #\$0 R14H ROL,X INR2 ROH,X (ROL,X)	STORE BYTE INDIRECT INDICATE R0 IS RESULT NEG
F712: A5 01 F714: 95 01 F716: 60 F717: A5 00 F719: 81 00 F71B: A0 00 F71B: 84 1D F71F: F6 00 F721: D0 02 F722: F6 01 F725: 60 F726: A1 00 F728: 85 00 F72A: A0 00	STAT STAT2 STAT3 INR INR2 LDAT	LDA STA RTS LDA STA LDY STY INC BNE INC RTS LDA STA LDY	ROL,X ROH ROH,X ROL (ROL,X) #\$0 R14H ROL,X INR2 ROH,X (ROL,X) ROL #\$0	STORE BYTE INDIRECT INDICATE RO IS RESULT NEG INCR RX LOAD INDIRECT (RX) TO RO
F712: A5 01 F714: 95 01 F716: 60 F717: A5 00 F719: 81 00 F718: A0 00 F71B: A4 1D F71F: F6 00 F721: D0 02 F723: F6 01 F725: 60 F726: A1 00 F726: A1 00 F724: A0 00 F724: A0 00	STAT STAT2 STAT3 INR INR2 LDAT	LDA STA RTS LDA STA LDY STY INC BNE INC RTS LDA STA LDY STY	ROL,X ROH ROH,X ROL (ROL,X) #\$0 R14H ROL,X INR2 ROH,X (ROL,X) ROL #\$0	STORE BYTE INDIRECT INDICATE RO IS RESULT NEG INCR RX LOAD INDIRECT (RX) TO RO
F712: A5 01 F714: 95 01 F716: 60 F717: A5 00 F719: 81 00 F718: A0 00 F718: A4 1D F71F: F6 00 F721: D0 02 F723: F6 01 F725: 60 F726: A1 00 F728: 85 00 F722: A0 00 F722: 84 01 F725: F0 ED	STAT STAT2 STAT3 INR INR2 LDAT	LDA STA RTS LDA STA LDY STY INC BNE INC RTS LDA STA LDY STY BEQ	ROL,X ROH ROH,X ROL (ROL,X) #\$0 R14H ROL,X INR2 ROH,X (ROL,X) ROL #\$0 ROH STAT3	STORE BYTE INDIRECT INDICATE RO IS RESULT NEG INCR RX LOAD INDIRECT (RX) TO RO ZERO HIGH-ORDER RO BYTE ALWAYS TAKEN
F712: A5 01 F714: 95 01 F716: 60 F717: A5 00 F719: 81 00 F711: A0 00 F711: A0 00 F711: A0 00 F721: D0 02 F723: F6 01 F725: 60 F726: A1 00 F726: A1 00 F728: 85 00 F728: 85 00 F722: A0 00 F722: F0 ED F725: F0 ED F723: F0 ED	STAT STAT2 STAT3 INR INR2 LDAT	LDA STA RTS LDA STA LDY STY INC BNE INC RTS LDA STA LDY STY BEQ	ROL,X ROH ROH,X ROL (ROL,X) #\$0 R14H ROL,X INR2 ROH,X (ROL,X) ROL #\$0 ROH STAT3	STORE BYTE INDIRECT INDICATE RO IS RESULT NEG INCR RX LOAD INDIRECT (RX) TO RO ZERO HIGH-ORDER RO BYTE ALWAYS TAKEN HIGH ORDER BYTE = 0
F712: A5 01 F714: 95 01 F716: 60 F717: A5 00 F719: 81 00 F71B: A0 00 F71B: A0 00 F71B: A0 00 F71B: A0 00 F721: D0 02 F723: F6 01 F725: 60 F726: A1 00 F728: 85 00 F726: A1 00 F728: A0 00 F722: 84 01 F725: F0 ED F730: A0 00 F732: F0 06	STAT STAT2 STAT3 INR INR2 LDAT	LDA STA RTS LDA STA LDY STY INC BNE INC RTS LDA STA LDY STY BEQ LDY BEQ	ROL,X ROH ROH,X ROL (ROL,X) #\$0 R14H ROL,X INR2 ROH,X (ROL,X) ROL #\$0 ROH STAT3 #\$0 POP2	STORE BYTE INDIRECT INDICATE RO IS RESULT NEG INCR RX LOAD INDIRECT (RX) TO RO ZERO HIGH-ORDER RO BYTE ALWAYS TAKEN HIGH ORDER BYTE = 0 ALWAYS TAKEN
F712: A5 01 F714: 95 01 F716: 60 F717: A5 00 F719: 81 00 F719: 84 1D F719: 84 1D F71F: F6 00 F721: D0 02 F722: D0 02 F723: F6 01 F725: 60 F726: A1 00 F726: A1 00 F726: A1 00 F726: A1 00 F727: 84 01 F726: F0 ED F730: A0 00 F722: F0 06 F734: 20 66 F7	STAT STAT2 STAT3 INR INR2 LDAT POP POPD	LDA STA RTS LDA STA LDY STY INC BNE INC RTS LDY STY STY BEQ LDY SEQ LDY SEQ JSR	ROL,X ROH ROH,X ROL (ROL,X) #\$0 R14H ROL,X INR2 ROH,X (ROL,X) ROL,X) ROL #\$0 ROH STAT3 #\$0 POP2 DCR	STORE BYTE INDIRECT INDICATE RO IS RESULT NEG INCR RX LOAD INDIRECT (RX) TO RO ZERO HIGH-ORDER RO BYTE ALWAYS TAKEN HIGH ORDER BYTE = 0 ALWAYS TAKEN DECR RX
F712: A5 01 F714: 95 01 F716: 60 F717: A5 00 F719: 81 00 F719: 84 1D F719: 84 1D F711: F6 00 F721: D0 02 F722: D0 02 F723: F6 01 F725: 60 F726: A1 00 F726: A1 00 F728: 85 00 F726: A1 00 F728: 85 00 F726: A1 00 F728: 85 00 F727: A0 00 F739: F0 06 F734: 20 66 F7 F737: A1 00	STAT STAT2 STAT3 INR INR2 LDAT POP POPD	LDA STA RTS LDA STA LDY STY INC BNE INC STA LDY STY STY LDA LDY LDA	ROL,X ROH ROH,X ROL (ROL,X) #\$0 R14H ROL,X INR2 ROH,X (ROL,X) ROL #\$0 ROH STAT3 #\$0 POP2 DCR (ROL,X)	STORE BYTE INDIRECT INDICATE RO IS RESULT NEG INCR RX LOAD INDIRECT (RX) TO RO ZERO HIGH-ORDER RO BYTE ALWAYS TAKEN HIGH ORDER BYTE = 0 ALWAYS TAKEN DECR RX POP HIGH ORDER BYTE @RX
F712: A5 01 F714: 95 01 F716: 60 F717: A5 00 F719: 81 00 F718: A0 00 F718: A0 00 F718: A0 00 F721: D0 02 F723: F6 01 F725: 60 F726: A1 00 F726: A1 00 F726: 85 00 F726: 85 00 F726: 84 01 F726: F0 ED F730: A0 00 F732: F0 66 F734: 20 66 F7 F737: A1 00 F739: A8	STAT STAT2 STAT3 INR INR2 LDAT POP POPD	LDA STA RTS LDA STA LDY STY INC RTS LDA STA LDA STA LDY BEQ LDY BEQ LDY JSR LDA TAY	ROL,X ROH ROH,X ROL (ROL,X) #\$0 R14H ROL,X INR2 ROL,X ROH,X (ROL,X) ROL STAT3 #\$0 POP2 DCR (ROL,X)	STORE BYTE INDIRECT INDICATE RO IS RESULT NEG INCR RX LOAD INDIRECT (RX) TO RO ZERO HIGH-ORDER RO BYTE ALWAYS TAKEN HIGH ORDER BYTE = 0 ALWAYS TAKEN DECR RX POP HIGH ORDER BYTE @RX SAVE IN Y-REG
F712: A5 01 F714: 95 01 F716: 60 F717: A5 00 F718: A0 00 F711: A4 10 F711: B4 10 F711: B6 00 F721: D0 02 F722: F0 01 F725: 60 1 F726: A1 00 F728: 85 00 F728: A0 00 F728: R0 ED F728: F0 ED F728: F0 ED F728: F0 ED F728: F0 ED F730: A0 00 F732: F0 ED F732: F0 ED F733: A1 00 F739: A8 F73A: F73A: 20 66 F7	STAT STAT2 STAT3 INR INR2 LDAT POP POPD POP2	LDA STA RTS LDA STA LDY STY INC BNE INC RTS LDA STA LDY BEQ LDY BEQ JSR	ROL,X ROH ROH,X ROL (ROL,X) #\$0 R14H ROL,X INR2 ROL,X ROH,X (ROL,X) ROL STAT3 #\$0 POP2 DCR (ROL,X)	STORE BYTE INDIRECT INDICATE RO IS RESULT NEG INCR RX LOAD INDIRECT (RX) TO RO ZERO HIGH-ORDER RO BYTE ALWAYS TAKEN HIGH ORDER BYTE = 0 ALWAYS TAKEN DECR RX POP HIGH ORDER BYTE @RX SAVE IN Y-REG
F712: A5 01 F714: 95 01 F716: 60 F717: A5 00 F719: 81 00 F718: A0 00 F718: A0 00 F718: A0 00 F721: D0 02 F723: F6 01 F725: 60 F726: A1 00 F726: A1 00 F726: 85 00 F726: 85 00 F726: 84 01 F726: F0 ED F730: A0 00 F732: F0 66 F734: 20 66 F7 F737: A1 00 F739: A8	STAT STAT2 STAT3 INR INR2 LDAT POP POPD POP2	LDA STA RTS LDA STA LDY STY INC BNE INC RTS LDA STA LDY SEQ LDY BEQ LDY BEQ LDY LDA LDA LDA LDA LDA LDA LDA LDA LDA LDA	ROL,X ROH 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)	STORE BYTE INDIRECT INDICATE RO IS RESULT NEG INCR RX LOAD INDIRECT (RX) TO RO ZERO HIGH-ORDER RO 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
F712: A5 01 F714: 95 01 F716: 60 F717: A5 00 F718: A0 00 F719: 81 00 F711: A0 00 F711: F6 00 F721: D0 02 F723: F6 01 F725: 60 F726: A1 00 F728: 85 00 F728: R5 00 F728: R5 00 F728: R5 00 F728: R0 00 F728: F0 ED F730: A0 00 F732: F0 66 F734: 20 66 F739: A8 F739: F730: A1 00 F733: 20 66 F735: A2 66	STAT STAT2 STAT3 INR INR2 LDAT POP POPD POP2	LDA STA RTS LDA STA LDY STY INC BNE INC RTS LDA STA LDY SEQ LDY BEQ LDY BEQ LDY LDA LDA LDA LDA LDA LDA LDA LDA LDA LDA	ROL,X ROH ROH,X ROL (ROL,X) #\$0 R14H ROL,X INR2 ROH,X (ROL,X) ROL STAT3 #\$0 FOP2 DCR (ROL,X) POP2 DCR (ROL,X) ROL ROH	STORE BYTE INDIRECT INDICATE RO IS RESULT NEG INCR RX LOAD INDIRECT (RX) TO RO ZERO HIGH-ORDER RO 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 RO
F712: A5 01 F714: 95 01 F716: 60 F717: A5 00 F719: 81 00 F719: 81 00 F711: A0 00 F711: F6 00 F721: D0 02 F723: F6 01 F725: 60 60 F726: A1 00 F728: 85 00 F728: A0 00 F722: F0 ED F732: F0 60 F734: 20 66 F7 F737: A1 00 F739: A8 F7 F734: 20 66 F7 F735: A1 00 F735: A1 00 F735:	STAT STAT2 STAT3 INR INR2 LDAT POP POPD POP2	LDA STA RTS LDA STA LDY STY INC BNE INC RTS LDY STA LDY STY BEQ LDA JSR LDA JSR LDA STA STA STA	ROL,X ROH ROH,X ROL (ROL,X) #\$0 R14H ROL,X INR2 ROH,X (ROL,X) ROL STAT3 #\$0 FOP2 DCR (ROL,X) POP2 DCR (ROL,X) ROL ROH	STORE BYTE INDIRECT INDICATE RO IS RESULT NEG INCR RX LOAD INDIRECT (RX) TO RO ZERO HIGH-ORDER RO 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 RO
F712: A5 01 F714: 95 01 F716: 60 F717: A5 00 F719: 81 00 F719: 81 00 F711: A0 00 F711: B4 1D F711: F6 01 F721: D0 02 F723: F6 01 F725: 60 F726: F726: A1 00 F728: 85 00 F722: F0 ED F730: A0 00 F732: F0 66 F734: 20 66 F737: A1 00 F739: A8 ************************************	STAT STAT2 STAT3 INR INR2 LDAT POP POPD POP2 POP3	LDA STA STA RTS LDA STY STY STY STY STY LDA LDA LDA LDA LDA LDA STY LDA STY LDA STY LDA	ROL,X ROH ROH,X ROL (ROL,X) #\$0 R14H ROL,X INR2 ROH,X (ROL,X) ROL STAT3 #\$0 FOP2 DCR (ROL,X) POP2 DCR (ROL,X) ROL ROH	STORE BYTE INDIRECT INDICATE RO IS RESULT NEG INCR RX LOAD INDIRECT (RX) TO RO ZERO HIGH-ORDER RO 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
F712: A5 01 F714: 95 01 F716: 60 F717: A5 00 F719: 81 00 F711: A0 00 F711: R0 00 F711: F0 00 F712: D0 02 F723: F0 01 F725: 60 1 F726: A1 00 F728: 85 00 F728: R0 00 F728: R0 00 F728: R0 00 F728: R0 00 F730: A0 00 F731: A0 00 F732: F0 66 F734: 20 66 F7 F730: A1 00 F737: A1 00 F737: A1 00 F737: A1 00 F741: 84 01 F743: A0 0	STAT STAT2 STAT3 INR INR2 LDAT POP POPD POP2 POP3	LDA STA RTS LDA STA LDY STY INC BNC RTS LDA STA LDY SEQ LDA JSR LDA JSR LDA STA STA STY STA STY RTS	ROL,X ROH ROH,X ROL (ROL,X) #\$0 R14H ROL,X INR2 ROH,X (ROL,X) ROL #\$0 ROH STAT3 #\$0 POP2 DCR (ROL,X) POP2 DCR (ROL,X) ROL ROL,X) ROL ROL ROL ROL ROL ROL ROL ROL ROL ROL	STORE BYTE INDIRECT INDICATE RO IS RESULT NEG INCR RX LOAD INDIRECT (RX) TO RO ZERO HIGH-ORDER RO 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 RO INDICATE RO AS LAST RESULT REG
F712: A5 01 F714: 95 01 F716: 60 F717: A5 00 F719: 81 00 F711: A0 00 F711: R0 00 F711: F6 00 F721: D0 02 F723: F6 01 F725: 60 1 F726: A1 00 F722: R0 00 F722: R0 00 F722: F0 ED F732: F0 60 F734: 20 66 F737: A1 00 F739: A8 1 F735: 20 66 F7 F735: A1 00 1 F734: 20 66 F7 F735: A1 00 1 F743: A0 00 1 F741: 84 01 1 F743: A0	STAT STAT2 STAT3 INR INR2 LDAT POP POPD POP2 POP3 LDDAT	LDA STA STA RTS LDA STY STY INC BNE INC STA LDY STY BEQ JSR LDA JSR LDA JSR STY JSR STY STY STS STY STS STS STS STS STS	ROL,X ROH ROH,X ROL (ROL,X) #\$0 R14H ROL,X INR2 ROH,X (ROL,X) ROL #\$0 ROH STAT3 #\$0 ROH STAT3 #\$0 POP2 DCR (ROL,X) POP2 DCR (ROL,X) ROL ROL (ROL,X) ROL ROL ROL ROL ROL ROL ROL ROL ROL ROL	STORE BYTE INDIRECT INDICATE RO IS RESULT NEG INCR RX LOAD INDIRECT (RX) TO RO ZERO HIGH-ORDER RO 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 RO INDICATE RO AS LAST RESULT REG LOW-ORDER BYTE TO RO, INCR RX
F712: A5 01 F714: 95 01 F716: 60 F717: A5 00 F719: 81 00 F719: 81 00 F711: A0 00 F711: F6 00 F721: D0 02 F723: F6 01 F725: 60 F726: F726: A1 00 F722: A0 00 F722: R0 00 F722: F0 ED F732: F0 06 F732: F0 06 F733: A0 00 F732: F0 06 F733: A1 00 F733: A1 00 F737: A1 00 F737: A1 00 F737: A1 00 F741: 84 01 F743: A0 00 F741: 84 01 <td>STAT STAT2 STAT3 INR INR2 LDAT POP POPD POP2 POP3 LDDAT</td> <td>LDA STA STA RTS LDA STY INC BNE INTS LDA STA LDA STA LDA STA LDA STA LDA STY LDA STY LDA STY LDA STY LDA STY LDA STY 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 STA LDA STA LDA STA LDA STA STA STA STA STA STA STA STA STA ST</td> <td>ROL,X ROH ROH,X ROL (ROL,X) #\$0 R14H ROL,X INR2 ROH,X (ROL,X) ROL #\$0 ROH STAT3 #\$0 POP2 DCR (ROL,X) ROL ROH #\$0 ROH #\$0 POP2 DCR (ROL,X) ROL ROL,X) ROL ROH #\$0 ROH ROH ROH ROH ROH ROH ROH ROH</td> <td>STORE BYTE INDIRECT INDICATE RO IS RESULT NEG INCR RX LOAD INDIRECT (RX) TO RO ZERO HIGH-ORDER RO 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 RO INDICATE RO AS LAST RESULT REG</td>	STAT STAT2 STAT3 INR INR2 LDAT POP POPD POP2 POP3 LDDAT	LDA STA STA RTS LDA STY INC BNE INTS LDA STA LDA STA LDA STA LDA STA LDA STY LDA STY LDA STY LDA STY LDA STY LDA STY 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 STA LDA STA LDA STA LDA STA STA STA STA STA STA STA STA STA ST	ROL,X ROH ROH,X ROL (ROL,X) #\$0 R14H ROL,X INR2 ROH,X (ROL,X) ROL #\$0 ROH STAT3 #\$0 POP2 DCR (ROL,X) ROL ROH #\$0 ROH #\$0 POP2 DCR (ROL,X) ROL ROL,X) ROL ROH #\$0 ROH ROH ROH ROH ROH ROH ROH ROH	STORE BYTE INDIRECT INDICATE RO IS RESULT NEG INCR RX LOAD INDIRECT (RX) TO RO ZERO HIGH-ORDER RO 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 RO INDICATE RO AS LAST RESULT REG
F712: A5 01 F714: 95 01 F716: 60 F717: A5 00 F719: 81 00 F719: 81 00 F711: A0 00 F711: R4 1D F711: F6 01 F721: D0 02 F723: F6 01 F725: 60 F726: F726: A1 00 F728: 85 00 F728: R5 00 F728: R5 00 F728: R5 00 F728: R5 00 F728: R0 00 F739: A0 00 F737: A1 00 F737: A1 00 F739: A8 01 F747: A1 00 F748: A0 00 F745: 84 01 F745: A0 00 <td>STAT STAT2 STAT3 INR INR2 LDAT POP POPD POP2 POP3 LDDAT</td> <td>LDA STA RTS LDA STA LDY STY INC BNC RTS LDA STA LDA STA LDA STA LDA STA LDA STAY STY LDA STY LDA STY LDA STY STY STY STY STY STY STA STA STA STA STA STA STA STA STA STA</td> <td>ROL, X ROH ROH, X ROL (ROL, X) #\$0 R14H ROL, X INR2 ROH, X (ROL, X) ROH STAT3 #\$0 POP2 DCR (ROL, X) ROL ROH #\$0 ROH ROH #\$0 ROH ROH #\$0 ROH #\$0 ROH ROH ROH ROH ROH ROH ROH ROH</td> <td>STORE BYTE INDIRECT INDICATE RO IS RESULT NEG INCR RX LOAD INDIRECT (RX) TO RO ZERO HIGH-ORDER RO 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 RO INDICATE RO AS LAST RESULT REG LOW-ORDER BYTE TO RO, INCR RX HIGH-ORDER BYTE TO RO</td>	STAT STAT2 STAT3 INR INR2 LDAT POP POPD POP2 POP3 LDDAT	LDA STA RTS LDA STA LDY STY INC BNC RTS LDA STA LDA STA LDA STA LDA STA LDA STAY STY LDA STY LDA STY LDA STY STY STY STY STY STY STA STA STA STA STA STA STA STA STA STA	ROL, X ROH ROH, X ROL (ROL, X) #\$0 R14H ROL, X INR2 ROH, X (ROL, X) ROH STAT3 #\$0 POP2 DCR (ROL, X) ROL ROH #\$0 ROH ROH #\$0 ROH ROH #\$0 ROH #\$0 ROH ROH ROH ROH ROH ROH ROH ROH	STORE BYTE INDIRECT INDICATE RO IS RESULT NEG INCR RX LOAD INDIRECT (RX) TO RO ZERO HIGH-ORDER RO 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 RO INDICATE RO AS LAST RESULT REG LOW-ORDER BYTE TO RO, INCR RX HIGH-ORDER BYTE TO RO
F712: A5 01 F714: 95 01 F716: 60 F717: A5 00 F719: 81 00 F719: 81 00 F711: A0 00 F711: F6 00 F721: D0 02 F723: F6 01 F725: 60 F726: F726: A1 00 F722: A0 00 F722: R0 00 F722: F0 ED F732: F0 06 F732: F0 06 F733: A0 00 F732: F0 06 F733: A1 00 F733: A1 00 F737: A1 00 F737: A1 00 F737: A1 00 F741: 84 01 F743: A0 00 F741: 84 01 <td>STAT STAT2 STAT3 INR INR2 LDAT POP POPD POP2 POP3 LDDAT</td> <td>LDA STA RTS LDA STA LDY STY INC BNC RTS LDA STA LDA STA LDA STA LDA STA LDA STAY STY LDA STY LDA STY LDA STY STY STY STY STY STY STA STA STA STA STA STA STA STA STA STA</td> <td>ROL, X ROH ROH, X ROL (ROL, X) #\$0 R14H ROL, X INR2 ROH, X (ROL, X) ROL #\$0 ROH STAT3 #\$0 POP2 DCR (ROL, X) ROL ROH STAT3 #\$0 POP2 DCR (ROL, X) ROL ROH H\$0 ROH STAT3 #\$0 POP2 DCR (ROL, X) ROL ROH ROH ROH ROH ROH ROH ROH ROH</td> <td>STORE BYTE INDIRECT INDICATE RO IS RESULT NEG INCR RX LOAD INDIRECT (RX) TO RO ZERO HIGH-ORDER RO 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 RO INDICATE RO AS LAST RESULT REG LOW-ORDER BYTE TO RO, INCR RX</td>	STAT STAT2 STAT3 INR INR2 LDAT POP POPD POP2 POP3 LDDAT	LDA STA RTS LDA STA LDY STY INC BNC RTS LDA STA LDA STA LDA STA LDA STA LDA STAY STY LDA STY LDA STY LDA STY STY STY STY STY STY STA STA STA STA STA STA STA STA STA STA	ROL, X ROH ROH, X ROL (ROL, X) #\$0 R14H ROL, X INR2 ROH, X (ROL, X) ROL #\$0 ROH STAT3 #\$0 POP2 DCR (ROL, X) ROL ROH STAT3 #\$0 POP2 DCR (ROL, X) ROL ROH H\$0 ROH STAT3 #\$0 POP2 DCR (ROL, X) ROL ROH ROH ROH ROH ROH ROH ROH ROH	STORE BYTE INDIRECT INDICATE RO IS RESULT NEG INCR RX LOAD INDIRECT (RX) TO RO ZERO HIGH-ORDER RO 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 RO INDICATE RO AS LAST RESULT REG LOW-ORDER BYTE TO RO, INCR RX

	A5	01		LDA	ROH	BYTE AND INCR RX. THEN
F757:						STORE HIGH-ORDER BYTE.
		1F F7 66 F7	STPAT	JMP JSR		INCR RX AND RETURN DECR RX
F75F:				LDA		
F761:						STORE RO LOW BYTE @RX
		43 F7 00			POP3 ROL,X	INDICATE RO AS LAST RSLT REG
F768:			DCK			DECR RX
F76A:					ROH,X	
F76C: F76E:		00	DCR2		ROL,X	
F76E:			SUB	RTS LDY	#\$0	RESULT TO RO
F771:	38		CPR	SEC		NOTE Y-REG = 13*2 FOR CPR
F772:				LDA		
F774: F776:		00 00			ROL,X ROL,Y	RO-RX TO RY
F779:				LDA		
F77B:					ROH,X	
F780:		01 00	SUB2	TYA	ROH,Y	LAST RESULT REG*2
F781:				ADC		CARRY TO LSB
F783:				STA	R14H	
F785: F786:				RTS LDA	ROL	
F788:				ADC		
F78A:				STA		R0+RX TO R0
F78C: F78E:				LDA ADC	ROH ROH,X	
F790:				LDY		R0 FOR RESULT
F792:						FINISH ADD
F794:		1E 19 F7				NOTE X-REG IS 12*2! PUSH LOW PC BYTE VIA R12
F799:					R15H	FOR HOW FE DITE VIA KIZ
		19 F7		JSR	STAT2	PUSH HIGH-ORDER PC BYTE
F79E:				CLC	DNCO	NO CARRY TEST
F79F: F7A1:		1E				DISPLACEMENT BYTE
F7A3:				BPL	BR2	
F7A5:				DEY	D1 - T	
F7A6: F7A8:		1E 1E	BR2	ADC STA		ADD TO PC
F7AA:				TYA		
F7AB:				ADC	R15H	
F7AD: F7AF:				STA RTS	R15H	
F7B0:				BCS	BR	
F7B2:	60			RTS		
					-	
F7B3:	0A			ASL	A	DOUBLE RESULT-REG INDEX TO X REG FOR INDEXING
	0A AA		BP	ASL TAX		DOUBLE RESULT-REG INDEX TO X REG FOR INDEXING TEST FOR PLUS
F7B3: F7B4: F7B5: F7B7:	0A AA B5 10	01	BP	ASL TAX LDA BPL	ROH,X	TO X REG FOR INDEXING
F7B3: F7B4: F7B5: F7B7: F7B9:	0A AA B5 10 60	01 E8	BP	ASL TAX LDA BPL RTS	ROH,X BR1	TO X REG FOR INDEXING TEST FOR PLUS BRANCH IF SO
F7B3: F7B4: F7B5: F7B7:	0A AA B5 10 60 0A	01 E8	BP	ASL TAX LDA BPL	ROH,X	TO X REG FOR INDEXING TEST FOR PLUS
F7B3: F7B4: F7B5: F7B7: F7B9: F7BA: F7BB: F7BC:	0A AA B5 10 60 0A AA B5	01 E8 01	BP BM	ASL TAX LDA BPL RTS ASL TAX LDA	ROH,X BR1 A ROH,X	TO X REG FOR INDEXING TEST FOR PLUS BRANCH IF SO
F7B3: F7B4: F7B5: F7B7: F7B9: F7BA: F7BB: F7BC: F7BE:	0A AA B5 10 60 0A AA B5 30	01 E8 01 E1	BP BM	ASL TAX LDA BPL RTS ASL TAX LDA BMI	ROH,X BR1 A	TO X REG FOR INDEXING TEST FOR PLUS BRANCH IF SO DOUBLE RESULT-REG INDEX
F7B3: F7B4: F7B5: F7B7: F7B9: F7BA: F7BB: F7BC:	0A AA B5 10 60 0A AA B5 30 60	01 E8 01 E1	BP BM	ASL TAX LDA BPL RTS ASL TAX LDA	ROH,X BR1 A ROH,X BR1	TO X REG FOR INDEXING TEST FOR PLUS BRANCH IF SO DOUBLE RESULT-REG INDEX
F7B3: F7B5: F7B5: F7B9: F7B8: F7B8: F7B8: F7BC: F7BE: F7C0: F7C1: F7C2:	0A AA B5 10 60 0A 85 30 60 0A AA	01 E8 01 E1	BP BM BZ	ASL TAX 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: F7B4: F7B5: F7B7: F7B9: F7B8: F7B8: F7BC: F7BE: F7C0: F7C1: F7C2: F7C3:	0A AA B5 10 60 0A 85 30 60 0A AA B5	01 E8 01 E1	BP BM BZ	ASL TAX LDA BPL RTS ASL TAX LDA BMI RTS ASL TAX LDA	ROH,X BR1 A 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
F7B3: F7B5: F7B5: F7B9: F7B8: F7B8: F7B8: F7BC: F7BE: F7C0: F7C1: F7C2:	0A AA B5 10 60 0A AA 60 0A AA B5 15	01 E8 01 E1 00 01	BP BM BZ	ASL TAX LDA BPL RTS ASL TAX LDA BMI RTS ASL TAX LDA	ROH,X BR1 A 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
F7B3: F7B5: F7B7: F7B7: F7B8: F7B8: F7B8: F7B8: F7C0: F7C1: F7C2: F7C2: F7C2: F7C3: F7C7: F7C7:	0A AA B5 10 60 0A AA B5 30 60 0A AA B5 55 50 60	01 E8 01 E1 00 01 D8	BP BM BZ	ASL TAX LDA BPL RTS ASL TAX LDA BMI RTS ASL TAX LDA ORA BEQ RTS	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
F7B3: F7B5: F7B7: F7B7: F7B8: F7B8: F7B8: F7B8: F7C1: F7C2: F7C2: F7C2: F7C5: F7C7: F7C7: F7C7: F7C7:	0A AA B5 10 60 0A AA B5 30 60 0A B5 5 60 60 0A	01 E8 01 E1 00 01 D8	BP BM BZ	ASL TAX 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
F7B3: F7B5: F7B7: F7B7: F7B8: F7B8: F7B8: F7B8: F7C0: F7C1: F7C2: F7C2: F7C2: F7C3: F7C7: F7C7:	0A AA B5 10 60 0A AA B5 0A AA B5 15 60 0A AA	01 E8 01 E1 00 01 D8	BP BM BZ BNZ	ASL TAX LDA BPL RTS ASL TAX LDA BMI RTS ASL TAX LDA ORA BEQ RTS	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
F7B3: F7B5: F7B7: F7B7: F7B8: F7B8: F7B6: F7B6: F7C0: F7C0: F7C2: F7C3: F7C5: F7C7: F7C7: F7C9: F7C2: F7C2: F7C2: F7C2:	0A B5 10 60 0A B5 30 60 0A B5 15 60 0A AA B5 15	01 E8 01 E1 00 01 D8	BP BM BZ BNZ	ASL TAX 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 (BOTH BYTES)
F7B3: F7B5: F7B7: F7B7: F7B8: F7B8: F7B8: F7B2: F7C2: F7C2: F7C2: F7C2: F7C7: F7C7: F7C9: F7C8: F7C8: F7C2: F7C3:	0A B5 10 0A B5 30 0A B5 15 60 0A AA B5 15 50 0A B5 15 D0	01 E8 01 E1 00 01 D8 00 01 CF	BP BM BZ BNZ	ASL TAX LDA BPL RTS ASL TAX LDA BMI RTS ASL TAX LDA ORA BEQ RTS ASL TAX LDA ORA BEQ RTS ASL TAX	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
F7B3: F7B5: F7B7: F7B7: F7B8: F7B8: F7B6: F7B6: F7C0: F7C0: F7C2: F7C3: F7C5: F7C7: F7C7: F7C9: F7C2: F7C2: F7C2: F7C2:	0A B5 10 60 0A B5 30 0A B5 5 60 0A B5 5 00 AA B5 15 5 00 60 00 60	01 E8 01 E1 00 01 D8 00 01 CF	BP BM BZ BNZ	ASL TAX LDA BPL RTS ASL TAX LDA BMI RTS ASL TAX LDA ORA BEQ RTS ASL TAX LDA ORA BEQ RTS ASL TAX	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)
F7B3: F7B5: F7B7: F7B7: F7B8: F7B8: F7B2: F7B2: F7C2: F7C2: F7C2: F7C3: F7C5: F7C7: F7C5: F7C7: F7C5: F7C7: F7C5: F7C5: F7C2: F7C3: F7C5: F7C5: F7C5: F7C5: F7C5: F7C7: F7C5: F7C7: F7C7: F7C7: F7C7: F7C2: F7C2: F7C2: F7C2: F7C2: F7C2: F7C2: F7C2: F7C2: F7C2: F7C2: F7C2: F7C2: F7C2: F7C2: F7C2: F7C2: F7C2: F7C3: F7C2: F7C3:	0A B5 10 0A B5 30 0A B5 50 0A B5 50 0A B5 15 00 AA B5 15 00 AA B5 50 0A AA B5 50 0A AA AA AA AA AA AA AA AA AA AA AA AA	01 E8 01 E1 00 01 D8 00 01 CF	BP BM BZ BNZ	ASL TAX LDA BPL RTS ASL TAX LDA BMI RTS ASL TAX LDA ORA BEQ RTS ASL TAX LDA ORA BRTS ASL TAX LDA	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
F7B3: F7B5: F7B7: F7B7: F7B8: F7B8: F7B2: F7B2: F7C0: F7C2: F7C2: F7C2: F7C7: F7C7: F7C7: F7C7: F7C2:	0A B5 10 0A B5 30 0A B5 50 0A B5 15 00 AA B5 15 00 AA B5 50 0A AA B5	01 E8 01 E1 00 01 D8 00 01 CF	BP BM BZ BNZ	ASL TAX LDA BPL RTS ASL TAX LDA BMI RTS ASL TAX LDA ORA BEQ RTS ASL TAX LDA ORA BIN RTS ASL TAX LDA	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
F7B3: F7B5: F7B7: F7B7: F7B8: F7B8: F7B2: F7B2: F7C2: F7C2: F7C2: F7C3: F7C5: F7C7: F7C5: F7C7: F7C5: F7C7: F7C5: F7C5: F7C2: F7C3: F7C5: F7C5: F7C5: F7C5: F7C5: F7C7: F7C5: F7C7: F7C7: F7C7: F7C7: F7C2: F7C2: F7C2: F7C2: F7C2: F7C2: F7C2: F7C2: F7C2: F7C2: F7C2: F7C2: F7C2: F7C2: F7C2: F7C2: F7C2: F7C2: F7C3: F7C2: F7C3:	0A B5 10 60 0A B5 30 60 0A B5 50 0A B5 15 00 AB5 15 00 AA B5 35	01 E8 01 E1 00 01 D8 00 01 CF	BP BM BZ BNZ	ASL TAX LDA BPL RTS ASL TAX LDA BMI RTS ASL TAX LDA ORA BEQ RTS ASL TAX LDA ORA BIN RTS ASL TAX LDA	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
F7B3: F7B4: F7B5: F7B7: F7B2: F7B8: F7B2: F7B2: F7C2: F7C2: F7C2: F7C3: F7C3: F7C5: F7C7: F7C7: F7C7: F7C7: F7C7: F7C7: F7C7: F7C7: F7C7: F7C7: F7C7: F7C7: F7D8: F7D8:	0A B5 10 60 AA B5 30 0A B5 15 60 0A B5 15 00 AB5 15 00 AB5 15 00 AB5 5 5 9 70 85 70 70 70 70 70 70 70 70 70 70 70 70 70	01 E8 01 E1 00 01 D8 00 01 CF 00 01 FF C4	BP BM BZ BNZ BM1	ASL TAX LDA BPL RTS ASL TAX LDA BMI RTS ASL TAX LDA ORA BEQ RTS ASL TAX LDA ORA BNE RTS ASL TAX LDA ORA BNE RTS ASL TAX CORA BNE RTS ASL CORA BASL CORA BASL 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 ROH,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
F7B3: F7B4: F7B5: F7B5: F7B5: F7B2: F7B2: F7B2: F7B2: F7C2: F7C2: F7C2: F7C2: F7C2: F7C5: F7C5: F7C5: F7C5: F7C5: F7C5: F7C5: F7C5: F7C5: F7C5: F7C5: F7C5: F7C5: F7D5:	0A AA B5 10 60 AA B5 50 60 AA B5 50 60 AA B5 15 D0 60 AA B5 52 D0 60 AA B5 55 C0 AA B5 50 C0 AA B5 C0 AA AB5 50 C0 AA AB5 C0 AB5 C0 AB5 C0 AB5 C0 AB5 C0 AB5 C0 AB5 C0 AB5 C0 AB5 C0 AB5 C0 AB5 C0 AB5 C0 AB5 C0 C0 AB5 C0 C0 AB5 C0 C0 C0 C0 C0 C0 C0 C0 C0 C0 C0 C0 C0	01 E8 01 E1 00 01 D8 00 01 CF 00 01 FF C4	BP BM BZ BNZ BM1	ASL TAX LDA BPL RTS ASL TAX LDA BML RTS ASL TAX LDA ORA BEQ RTS ASL TAX LDA ORA BNE RTS ASL TAX LDA BNE RTS ASL TAX LDA BNE RTS ASL TAX SA SA SA SA SA SA SA SA SA SA SA SA SA	ROH,X BR1 A ROH,X BR1 A ROL,X ROH,X BR1 A ROL,X ROH,X BR1 A ROL,X ROH,X H\$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
F7B3: F7B5: F7B7: F7B7: F7B8: F7B8: F7B2: F7B2: F7C2: F7C2: F7C2: F7C2: F7C2: F7C5: F7C7: F7C5: F7C7: F7C2:	0A AA B5 10 60 AA B5 30 0A AB5 15 60 AB5 15 D0 60 AB5 15 D0 60 AB5 53 9 60 AA B5 50 0A AB5 50 0A AB5 60 A AB5 60 A AA AB5 60 A AA AB5 60 A AA AB5 60 AA AB5 60 A A AB5 60 A A AB5 60 A A A AB5 60 A A A AB5 60 A A A A AB5 60 A A A AB5 60 A A A A A A A A A A A A A A A A A A	01 E8 01 E1 00 01 D8 00 01 CF 00 01 FF C4	BP BM BZ BNZ BM1	ASL TAX LDA BPL RTS ASL TAX LDA BMI RTS ASL TAX LDA ORA BNC RTS ASL TAX LDA ORA BNC RTS ASL TAX LDA ORA BNC RTS ASL TAX LDA BCR BEQ RTS ASL	ROH,X BR1 A ROH,X BR1 A ROL,X ROH,X BR1 A ROL,X ROH,X BR1 A ROL,X ROH,X H\$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)
F7B3: F7B4: F7B5: F7B7: F7B2: F7B2: F7B2: F7B2: F7C2: F7C2: F7C2: F7C3: F7C3: F7C3: F7C4: F7C7: F7C5: F7C7: F7C7: F7C7: F7C7: F7C7: F7C7: F7C7: F7D2: F7D2: F7D3: F7D4: F7D5:	0A AA B5 10 60 AA B5 30 60 AA B5 50 AA B5 50 AA B5 50 AA B5 549 60 AA B5 549 60 AA B5 55 60 AA B5 55 B5 AA B5 50 AA AB5 50 AB5 50 AA AB5 50 AB5 50 AB5 50 A AB5 50 AB5 50 AB5 50 AB5 AB5 50 A AB5 50 A AB5 50 A AB5 50 A AB5 50 A AB5 50 A AB5 50 A A AB5 50 A AB5 50 A A A AB5 50 A A AB5 50 A A A AB5 A A A AB5 A A A A A A A A A A	01 E8 01 E1 00 01 D8 00 01 CF 00 01 FF C4	BP BM BZ BNZ BM1	ASL TAX LDA BPL RTS ASL TAX LDA BMI RTS ASL TAX LDA ORA BEQ RTS ASL TAX LDA ORA BNE RTS ASL TAX LDA AND EOR BEQ RTS ASL TAX LDA	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
F7B3: F7B4: F7B5: F7B7: F7B8: F7B8: F7B2: F7B2: F7C0: F7C0: F7C2: F7C3: F7C3: F7C5: F7C5: F7C7: F7C7: F7C7: F7C7: F7C7: F7C7: F7C7: F7C7: F7C7: F7C7: F7D3: F7D4: F7D5: F7D5: F7D7:	0A AB5 10 60 AB5 30 60 AB5 15 60 AB5 15 00 AB5 15 00 AB5 15 00 AB5 15 00 AB5 15 00 AB5 15 00 AB5 35 35 35 35	01 E8 01 E1 00 01 D8 00 01 CF 00 01 FF C4	BP BM BZ BNZ BM1	ASL TAX LDA BPL RTS ASL TAX LDA BMI RTS ASL TAX LDA ORA BEQ RTS ASL TAX LDA ORA BNE RTS ASL TAX LDA ORA BNE RTS ASL TAX LDA ORA BNE RTS ASL TAX LDA ORA BNE RTS ASL TAX LDA ASL TAX LDA CRTS ASL TAX LDA ASL TAX LDA ASL TAX LDA ASL TAX LDA ASL TAX LDA ASL TAX LDA ASL TAX LDA BMI RTS ASL TAX LDA BMI RTS ASL TAX LDA BASL TAX LDA BASL TAX LDA BASL TAX LDA BASL TAX LDA BASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA CRTS ASL TAX LDA CRTS ASL TAX LDA CRTS ASL TAX LDA CRTS ASL TAX LDA CRTS ASL TAX LDA CRTS ASL TAX LDA CRTS ASL TAX LDA CRTS ASL TAX LDA CRTS ASL TAX LDA CRTS ASL TAX LDA CRTS ASL TAX LDA CRTS ASL TAX LDA CRTS ASL TAX LDA CRTS ASL LDA CRTS ASL LDA ASL TAX LDA ASL TAX LDA ASL LTAX LDA ASL LDA ASL TAX LDA ASL LTAX LDA ASL LTAX LDA ASL ASL LTAX LDA ASL LTAX LDA ASL LTAX LDA ASL LDA ASL LDA ASL LDA ASL LDA ASL ASL LDA ASL ASL 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 SR1 A ROL, X ROH, X SFF BR1 A ROL, X ROH, 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) BRANCH IF SO
F7B3: F7B4: F7B5: F7B7: F7B2: F7B2: F7B2: F7B2: F7C2: F7C2: F7C2: F7C3: F7C3: F7C3: F7C4: F7C7: F7C5: F7C7: F7C7: F7C7: F7C7: F7C7: F7C7: F7C7: F7D2: F7D2: F7D3: F7D4: F7D5:	0A AB5 10 60 AB5 30 60 AB5 50 0A AB5 15 00 AB5 15 00 AB5 15 00 AB5 549 F0 0A AB5 55 949	01 E8 01 E1 00 01 D8 00 01 CF 00 01 FF C4	BP BM BZ BNZ BM1 BNM1	ASL TAX LDA BPL RTS ASL TAX LDA BML RTS ASL TAX LDA ORA BNE RTS ASL TAX LDA ORA BNE RTS ASL TAX LDA BNE RTS ASL TAX LDA BNE RTS ASL TAX LDA CRA BNE RTS ASL TAX LDA CRA CRA CRA CRA CRA CRA CRA CRA CRA CR	ROH, X BR1 A ROH, X BR1 A ROL, X ROH, X BR1 A ROL, X ROH, X #SFF BR1 A ROL, X ROH, X #SFF	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
F7B3: F7B4: F7B5: F7B7: F7B8: F7B8: F7B8: F7B8: F7B8: F7B2: F7C2: F7C2: F7C2: F7C2: F7C3: F7C2: F7C3: F7C4: F7C5: F7C7: F7C6: F7C7: F7C7: F7C7: F7C7: F7D2: F7D2: F7D2: F7D2: F7D5: F7D5: F7D5: F7D5: F7D5: F7D5: F726: F726: F726: F727: F727: F727: F728:	0A B5 100 AB5 300 AB5 150 0A B55 100 AB5 300 AB5 150 0A B55 49 60A AB5 349 60A AB5 35 900 AB5 500 AB5 70 AB5 70 AB5 70 AB5 70 70 AB5 70 70 70 70 70 70 70 70 70 70 70 70 70	01 E8 01 E1 00 01 D8 00 01 CF 00 01 FF C4 00 01 FF E9	BP BM BZ BNZ BM1 BNM1	ASL TAX LDA BPL RTS ASL TAX LDA BMI RTS ASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA BNE RTS ASL TAX LDA BNE RTS ASL TAX LDA BNE RTS ASL TAX CORA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA ASL TAX ASL TAX LDA ASL TAX ASL TAX LDA ASL TAX ASL TAX ASL TAX ASL TAX ASL TAX ASL TAX ASL TAX ASL TAX ASL TAX ASL TAX ASL TAX ASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX ASL TAX ASL TAX ASL TAX ASL TAX ASL TAX ASL TAX ASL TAX ASL TAX ASL RTS ASL TAX ASL ASL ASL TAX ASL RTS ASL TAX ASL RTS ASL RTS ASL TAX ASL RTS ASL RTS ASL RTS ASL RTS ASL TAX ASL RTS ASL RTS ASL TAX ASL RTS RTS ASL R	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
F7B3: F7B4: F7B5: F7B7: F7B8: F7B8: F7B8: F7B8: F7B8: F7B2: F7C2: F7C2: F7C2: F7C2: F7C3: F7C2: F7C3: F7C4: F7C5: F7C7: F7C6: F7C7: F7C7: F7C7: F7C7: F7D2: F7D2: F7D2: F7D2: F7D5: F7D5: F7D5: F7D5: F7D5: F7D5: F726: F726: F726: F727: F727: F727: F728:	0A B5 100 AB5 300 AB5 150 0A B55 100 AB5 300 AB5 150 0A B55 49 60A AB5 349 60A AB5 35 900 AB5 500 AB5 70 AB5 70 AB5 70 AB5 70 70 AB5 70 70 70 70 70 70 70 70 70 70 70 70 70	01 E8 01 E1 00 01 D8 00 01 CF 00 01 FF C4 00 01 FF B9	BP BM BZ BNZ BM1 BNM1	ASL TAX LDA BPL RTS ASL TAX LDA BMI RTS ASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA BNE RTS ASL TAX LDA BNE RTS ASL TAX LDA BNE RTS ASL TAX CORA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA ASL TAX ASL TAX LDA ASL TAX ASL TAX LDA ASL TAX ASL TAX ASL TAX ASL TAX ASL TAX ASL TAX ASL TAX ASL TAX ASL TAX ASL TAX ASL TAX ASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX LDA BEQ RTS ASL TAX ASL TAX ASL TAX ASL TAX ASL TAX ASL TAX ASL TAX ASL TAX ASL TAX ASL RTS ASL TAX ASL ASL ASL TAX ASL RTS ASL TAX ASL RTS ASL RTS ASL TAX ASL RTS ASL RTS ASL RTS ASL RTS ASL TAX ASL RTS ASL RTS ASL TAX ASL RTS RTS ASL R	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

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

6502 MICROPROCESSOR INSTRUCTIONS

- AOC 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
- BED Branch on Result Zero
- BIT Test Bits in Memory with Accumulator
- BMI Branch on Result Minus
- ONE 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 `I
- DEC Decrement Memory by One
- DEX Decrement index X by One
- DEY Decrement Index Y by One FOR "Exclusive-Or" Memory with
- Accumulator
- INC Increment Memory by One
- INX Increment Index X by One
- INY Increment Index I by One
- JMP Jump to New Location
- JSA Jump to New Location Saving Return Address

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

THE FOLLOWING NOTATION APPLIES TO THIS SUMMARY:

Α	Accumulator
X,Y	Index Registers
М	Memory
С	Borrow
Р	Processor Status Register
S	Stack Pointer
1	Change
_	No Change
+	Add
٨	Logical AND
-	Subtract
¥	Logical Exclusive OR
≜	Transfer From Slack
¥	Transfer To Stack
↓ ↓ ↓	Transfer To
←	Transfer To
V	Logical OR
PC	Program Counter
PCH	Program Counter High
PCL	Program Counter low
OPER	Operand
#	Immediate Addressing Mode

FIGURE 1. ASL-SHIFT LEFT ONE BIT OPERATION

С	•	7	6	5	4	3	2	1	0	-	0	
---	---	---	---	---	---	---	---	---	---	---	---	--

FIGURE 2 ROTATE ONE BIT LEFT (MEMORY OR ACCUMULATOR)

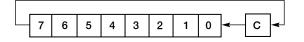


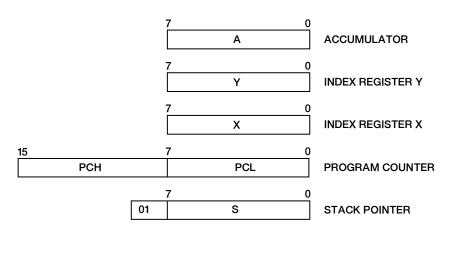
FIGURE 3.

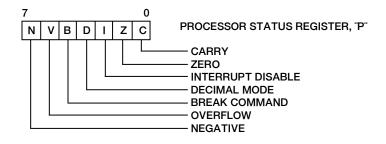
└ → C →	7	6	5	4	3	2	1	0	

NOTE 1: BIT — TEST BITS

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

PROGRAMMING MODEL





INSTRUCTION CODES

BVS BVS <th>to with carry A-M-C A.C with carry A-M-C A.C is with A M A but but (See Figure 1) Accumulator) (See Figure 1) arry clear Branch on C=0 arry set Branch on C=0</th> <th>ļ į</th> <th>A A Oper X Oper X Oper X Oper X Oper X Oper X Oper X Oper X Oper X Oper X</th> <th></th> <th></th> <th>BVS Branch on overllow CLC</th> <th>┟╌╍╍╎╌</th> <th>-<u></u></th> <th> </th> <th>, ,</th> <th></th> <th></th>	to with carry A-M-C A.C with carry A-M-C A.C is with A M A but but (See Figure 1) Accumulator) (See Figure 1) arry clear Branch on C=0 arry set Branch on C=0	ļ į	A A Oper X Oper X Oper X Oper X Oper X Oper X Oper X Oper X Oper X Oper X			BVS Branch on overllow CLC	┟╌╍╍╎╌	- <u></u>	 	, ,		
MC Operation Eig Z VVVV Parach on overflow set Eigendo not val Peration Vval Peration	to with carry A-M-CA.C with carry AA MA.C but but AA MA but (See Figure 1) Accumulator) (See Figure 1) Accumulator) and (See Figure 1)				<u>^∕^</u>	Branch on overllow			_	20	-	
Risk (R) (R) (R) (R) (R) (R) (R) (R) (R) (R)	with Carry in with AAMA but (See Figure 1) Accumulator) arry clear Branch on C=D arry set Branch on C=D					CLC	-		BVS Oper	5	2	
Ric ADC Oper Display Display <thdisplay< th=""> <thdisplay< td="" th<=""><td>ry with AAM A but A Accumulator) (See Figure 1) Accumulator) arry clear Branch on C=0 arry clear Branch on C=0</td><td></td><td></td><td></td><td></td><td>-</td><td></td><td></td><td></td><td></td><td></td><td></td></thdisplay<></thdisplay<>	ry with AAM A but A Accumulator) (See Figure 1) Accumulator) arry clear Branch on C=0 arry clear Branch on C=0					-						
R.K. ADD Open AD ADD Open AD ADD Open CL	ry with AAMA but Accumulator) (See Figure 1) Accumulator) and AAMA Accumulator) (See Figure 1) Accumulator) (See Figure 1) Accumulator) and AAMA					Clear carry flag	0 1 0	Implied	CLC	18	-	0
CLI D - 1 MD - 1	ry with AAMA bit bit (See Figure 1) Accumulator) (See Figure 1) arry clear Branch on C=0 arry set Branch on C=1					CLD Clear decimal mode		heilen	2	e.		
Bits alst AND ADD AND OFF(2012) Compare ADD Deriv ADD Call ADD Deri <	ry with AAM A bit Accumulator) fsee Figure 1) Accumulator fry clear Branch on C=0 sry set Branch on C=1									3	-	
Bits AND offer 23 2 VVVVV CU Molection 23 2 VVVVV Constraine CU Molection Current and constraints Current and const Curent and c	ry with AAMA but Accumulator) (See Figure 1) Accumulator) arry clear Branch on C=0 arry set Branch on C=1					5		implied	CLI	53	-	- 0
age Allo Oper 35 2 CHP Molect 35 2 CHP Molect 35 2 CHP Molect 35 2 CHP Molect 35 3	t bit Accumulator) (See Figure 1) Accumulator) (See Figure 1) arry clear Branch on C=0				~~~//	2 1					ſ	
R.K. Amo Oper 20 2 <t< td=""><td>left one bit lory or Accumulator) ch on carry clear Branch on C=0 ch on carry set Branch on C=1</td><td></td><td></td><td></td><td></td><td>Clear criedlow flag</td><td></td><td>Pollon</td><td>2</td><td>gg</td><td>-</td><td></td></t<>	left one bit lory or Accumulator) ch on carry clear Branch on C=0 ch on carry set Branch on C=1					Clear criedlow flag		Pollon	2	gg	-	
Riv Aim District Signation A Memory and construction Memory and constor constoreconstruction Memory and constructio	left one bit lory or Accumulator) sh on carry clear Branch on C=0 ch on carry set Branch on C=1									3	-	
CM AMD Derivity 38 3 Compare memory and entry A - M Immediate consister Compare memory and accumulator A - M Immediate constraints CMP eity AMD Derivy 31 2 2 Page x CMP eity AND Derivy 31 2 2 Page x CMP age ASI Oper Deri 1 V/V 2 Page x CMP age ASI Oper Deri 1 V/V 2 Page x CMP age ASI Oper E 2 1 Indirect/i Y CMP e BCC Oper SC 2 CPV 2 Page CPV e BSI Oper FO 2 CPV 2 Page Page CPV e BSI Oper FO 2 CPV 2 Page Page CPV e BSI Oper FO 2	left one bit ory or Accumulator) (See Figure 1) cry or Carry cry or Carry (See Figure 1) cry or Carry cry or Carry (See Figure 1) cry or Carry cry or Carry (See Figure 1) cry or Carry (See Figure 1)					CMP						
Circle Allo Opper 2 Allo Compare Compare <thcompare< th=""> Compare Compare</thcompare<>	left one bit ory or Accumulator) (See Figure 1) the Accumulator) (See Figure 1) (See Figure 1) (See Figure 1) (See Figure 1) (See Figure 1) (See Figure 1)		1 4004			Compare memory ar	 <	Immediate	_	82	~~~	~~^//
Inder ASI Oper 0A 1 V/V/ ASSOURCE ASSOURCE <td>left one bit ory or Accumulator) (See Figure 1) the Accumulator) (See Figure 1) the Accumulator) (See Figure 1) the Accumulator) (See Figure 1) the Accumulator) (See Figure 1)</td> <td></td> <td>ASL A ASL Oper ASL Oper X</td> <td></td> <td></td> <td></td> <td></td> <td>Zero Page, X</td> <td></td> <td>88</td> <td>1 01</td> <td></td>	left one bit ory or Accumulator) (See Figure 1) the Accumulator) (See Figure 1) the Accumulator) (See Figure 1) the Accumulator) (See Figure 1) the Accumulator) (See Figure 1)		ASL A ASL Oper ASL Oper X					Zero Page, X		88	1 01	
ASI Ober ASI O	left one bit ory or Accumulator) (See Figure 1) the carry clear Branch on C=0 ch on carry set Branch on C=1		ASL A ASL Oper ASL Oper X		-			Absolute Absolute Y		88		
age ASL Oper 66 2 0 <th< td=""><td>ory or Accumulator) ch on carry clear Branch on C=0 ch on carry set Branch on C=1</td><td></td><td>ASL Oper ASL Oper X</td><td></td><td></td><td></td><td></td><td>Absolute.Y</td><td></td><td>38</td><td> ით</td><td></td></th<>	ory or Accumulator) ch on carry clear Branch on C=0 ch on carry set Branch on C=1		ASL Oper ASL Oper X					Absolute.Y		38	 ით	
Bit ASL Ober Bit Z iii.x Bit Z Z iii.x Der	ch on carry clear Branch on C=0 ch on carry set Branch on C=1		ASL Oper X					(Indirect,X)		5	~	
Risk ASL Oper Re BCC Oper Set Zeto CPX Re BCC Oper BO 2 Compare memory and index X A. Memory and index X M - 1M Memory and index X M - 1M Memory and index X M - 1M Memory and index M Memory and index M Memory and index M Memory and index X Memory and index<	ch on carry clear Branch on C°O ch on carry set Branch on C°I		1C Door					(Indirect).Y		5	~	
re BCC Oper So 2 Compare memory and index X M Immediate CPX re BCC Oper BO 2 CPY Compare memory and index Y X M Immediate CPY re BEO Oper BO 2 CPY Compare memory and index Y X M Immediate CPY re BEO Oper EO 2 DEC DEC M Zero Page CPY re BMI Oper 2d 30 2 DEC DEC Absolute CPY re BMI Oper 30 2 DEC DEC Absolute CPY re BMI Oper 30 2 DEC DEC Absolute CPY re BMI Oper 30 2 DEC DEC Absolute CPY re BMI Oper 30 2 DEC DEC Absolute CPY re BMI Oper 30 2 DEC DEC Absolute DEC re BMI Oper 30 2	ch on carry clear Branch on C°O ch on carry set Branch on C=1	ł	ASL Oper.X			CPX						
Re BCC Oper 90 2 CPY Absolute CPY Re BSS Oper B0 2 CPY Zero Page CPY Re BSG Oper B0 2 CPY Zero Page CPY Re BSG Oper B0 2 CPY Zero Page CPY Re BSG Oper F0 2 Decrement memory and Y – M Zero Page CPY Re BNI Oper 2d 2 My/M6 Decrement memory M – 1M Zero Page DEC Re BNI Oper 30 2 DEC Absolute CPY Re BNE Oper 30 2 DEC Absolute DEC Absolute DEC Re BNE Oper 30 2 DEC Absolute DEC Absolute DEC Absolute DEC Absolute DEC	ch on carry clear Branch on C=0 ch on carry set Branch on C=1			-		Compare memory at	×	: Immediate	CPX #Oper	83	~ ~	///
e BCS Ober B0 2 Compare memory and index Y Y – M Immediate CPY index Y e BEO Oper F0 2 Compare memory and index Y Y – M Immediate CPY index P e BEO Oper F0 2 Decrement memory M – 1 – M Zero Page Decrement memory Decrement approx D	ch on carry set Branch on C=1		SCC Oper		ł			Absolute	CPX Oper	18	• ~	
ice BCS Oper B0 2 Compare memory and index Y Y – M Immediate CPY index CPY index CPY index Page CPY index Y Compare memory index Y N – M Zero Page CPY isolute DE	ch on carry set Branch on C=1					CPY						
Index Y Zero Page CPV age BiT* Oper Za 2 M6 DEC age BiT* Oper Za 2 M1M Zero Page DEC age BiT* Oper Za 3 My/M6 Decrement memory M - 1M Zero Page DEC re BMI Oper 30 Z DEC Decrement memory M - 1M Zero Page DEC re BMI Oper 30 Z DEC Decrement index X X - 1X Implied DEC re BPL Oper D0 Z DEC Decrement index X X - 1X Implied DE d BRK* D0 Z DE Decrement index Y Y - 1Y Implied DE e BVC Oper S0 Z Decrement index Y Y - 1Y Implied DE	9E0		3CS Oper			Compare memory ar	7	Immediate	CPY #Oper	3	~	
ie BE0 Oper F0 2						index Y		Zero Page Absolute	CPY Oper CPY Oper	38	2	
age Bit' Oper 24 2 My/Me Decrement memory M - 1 -+ M Zero Page. DEC re Bit' Oper 26 3 My/Me DEC DEC DEC re Bit Oper 30 2 DEC Absolute. X DEC re Bit Oper 30 2 DEC DEC re Bit Oper 30 2 DEC re Bit Oper DO 2 DEC re Bit Oper DO 2 DEC d Bit Oper DO 2 DEC d Bit Oper 10 2 DEC e Bit Oper 10 2 DEC d Bit Oper 10 2 DEC d Bit Coper S0 3 Decrement index Y Y - 1Y Implied e Bit Coper S0 2 Decrement index Y Y - 1Y Implied DEY	on result zero Branch on Z+1		3EQ Oper		!	PCL				; 	·	
age Bit* Oper 24 2 $M_y \veeM_6$ by one Zero Fage.X Dic Assolute Dic Assolute Dic Assolute Dic Assolute re BMI Oper 30 2 DE X X - 1 -+ X Implied Dic Assolute re BNE Oper Do 2 DE X X - 1 -+ X Implied Dic Assolute d BRL Oper DO 2 Dic Y X - 1 -+ X Implied Dic X d BRL Oper DO 2 Dic Y Y - 1 -+ Y Implied Dic Y d BRC Oper SO 2 Dic Y Y - 1 -+ Y Implied Dic Y d BRC Oper SO 2 Dic Y Y - 1 -+ Y Implied Dic Y e BVC Oper SO 2 Dic Y Y - 1 -+ Y Implied Dic Y						Decrement memory	ī	Zero Page	0FC Oner	ප	ć	//
Image: Non-out-out-out-out-out-out-out-out-out-out	A M, M7 + N,	•	311* Oper		1			Zero Page,X	DEC Oper.X	85	1010	
ee BMI Oper 30 2 DEX re BNE Oper D0 2 Decrement index X X - 1 - + X Implied re BNE Oper D0 2 DEX X - 1 - + X Implied re BPL oper 10 2 DEX X - 1 - + Y Implied d BPL oper 10 2 Decrement index Y Y - 1 - + Y Implied d BPL oper 00 1 Decrement index Y Y - 1 - + Y Implied e BVC Oper S0 2 Decrement index Y Y - 1 - + Y Implied		+		+				Absolute,X	DEC Oper,X	ä	~	
Ref BNE Oper D0 2 Decrement index X X = 1 -+-X Implied re BNE Oper D0 2 DF one DY one DY one DC crement index Y Y = 1 -+-Y Implied d BPL oper 10 2 DC crement index Y Y = 1 -+-Y Implied d BPL oper 00 1 DY one DC crement index Y Y = 1 -+-Y Implied e BVC Oper S0 2 L	ch on résult minus Branch on N=1		3MI Oper			DEX						
Re BNE Oper D0 2 DF re BPL oper 10 2 Decrement index Y Y - 1 -+ Y Implied d BPK 00 1 1 Dy one Dy one Implied e BVC Oper S0 2 Decrement index Y Y - 1 -+ Y Implied					 	Decrement index X	ī	Implied	DEX	3	-	~ · · - ^ /
re BPL oper 10 2 Decrement index Y Y - 1 -+Y Implied d BRK* 00 3 1 Decrement index Y Y - 1 -+Y Implied i BVC Oper S0 3 1 Decrement index Y Y - 1 -+Y Implied i BVC Oper S0 2 Decrement index Y Y - 1 -+Y Implied	Branch on Z=0		BNE Oper	_		nev					T	
re BPL oper 10 2 d BRK* 00 3 1 e BVC Oper 50 2 wei BVC Oper 50 2	BPL					Decrement index Y	ī	Imnlied	DEY	8	-	//
d BRK* BVC Oper re BVC Oper	Branch on N=0		3PL oper	-		by one						•
d BRK* ke BVC Oper	BAK											
re BVC Oper	Forced Interrupt PC-2 + P +	•	3BK•									
re BVC Oper												
			BVC Oper									
	Nole 1 440 % and 2 are used integration to the statue register # the result of A V M is near 3 - 5 coherence 2 - 0	t	Hale 2 A BRI	K COMMING C	note be mushed by set							

INSTRUCTION CODES

Name Description	Operation	Addressing Mode	Assembly Language Form	HEI Code	No. Bytes	P" Status Reg. N 2 C I D V	Name Description	Operation	Addressing Made	Assembly Language Form	Щ. 6 2 2 2 2 3 2 3 2 3 3 3 3 3 3 3 3 3 3 3	By les	.P. Status Reg. N Z C I D V
EOR				 			LSR						
"Exclusive-Or" memory with accumulator	A V M A	Immediate Zero Page Zero Page, X Absolute Absolute, X	EOR #Oper EOR Oper EOR Oper X EOR Oper X EOR Oper X	& & X & & & &	~~~~	·	Shift right one bit (memory or accumulator)	(See Figure 1)	Accumulator Zero Page Zero Page, X Absolute, X Absolute, X	LSR A LSR Oper LSR Oper X LSR Oper X LSR Oper X	29845 29875		^0
		Absolute,Y (Indirect,X)		841			NOP No American					· ·	
INC				5	,		ORA		hand		5	-	
Increment memory by one	M • 1 • M	Zero Page Zero Page,X Absolute Absolute.X	INC Oper INC Oper.X INC Oper INC Oper.X	88 H H H	~~~~	· ^>	"OR" memory with accumulator	A V M A	Immediate Zero Page Zero Page,X Absolute	*	8858	~~~~	
INX Increment index X by one	× ×	Implied	XNI	5	-	>			Absolute,X Absolute,Y (indirect,X)	ORA Oper.X ORA Oper.Y ORA (Oper.X)	9¢5	~~~	
INY									(Indirect).Y	ORA (Oper).Y	; =	101	
Increment index Y by one	Y - 1 - Y	Implied	ΝΥ	ŝ	~		PHA						
JMP							on stack	A †	Implied	РНА	\$	-	
Jump to new location	(PC+1) + PCH	Absolute Indirect	JMP Oper JMP (Oper)	2 8	ო ო	!	РНР						
JSR							Push processor status on stack	-	Implied	dHd	8	-	
Jump to new location saving return address	PC+2+ (PC+1) + PCL (PC+2) + PCH	Absolute	JSR Oper	ន	ę		PLA Pull accumulator	A +	Implied	PLA	8	-	^
LDA							from stack						
toad accumulator with memory	M M	Immediale Zero Page Zero Page,X		85 A§	~~~~	^^	Pull processor status from stack	+. 4	Implied	٥٦d	58		From Stack
		Absolute Absolute.X Absolute.Y (Indirect.X) (Indirect).Y	LDA Oper LDA Oper,X LDA Oper,Y LDA (Oper,X) LDA (Oper,Y	99875	~~~~~	. <u></u>	ROL Rotate one bit left (memory or accumulator)	(See Figure 2)	Accumulator Zero Page Zero Page X	ROL A ROL A ROL Oper ROL Oper X	288	- ~~	^^^
LDX	;								Absolute Absolute X	ROL Oper ROL Oper X	38		
Load Index X with memory	X + W	Immediale Zero Page Zero Page. Absolute Absolute	LDX #Cper LDX Oper LDX Oper.Y LDX Oper.Y	AS A6 A6 A6 B6 B6			ROR Rotate one bit right (memory or accumulator)	(See Figure 3)	Accumulator Zero Page Zero Page	ROR A ROR Oper ROR Oper X	388	-~~	///
LOY	;								Absolute Absolute.X	ROR Oper ROR Oper.X	۶ ۲		
Load Index Y with memory	} † ≥	immediale Zero Page Zero Page.X Absolute	LDY Oper.X	844S	2000					:			

DES
<u></u>
z
0
<u>C</u>
RC
ST
<u>Z</u>

,

Name Description	Operation	Addressing Mode	Assembly Language Form	H A BOO	No. Bytes	P Status Reg. N Z C I D V
RTI Return from interrupt	P+PC+	Implied	RT	9	-	From Stack
RTS					,	
Return from subroutine	PC1. PC+1 - PC	PC implied	HIS		-	
SBC Subtract memory from accumulator with borrow	A . M . Č A	Immediate Zero Page Zero Page.X Absolute Absolute.X Absolute.Y (Indirect.X)	SBC #Oper SBC 0per SBC 0per SBC 0per SBC 0per SBC 0per SBC 0per SBC 10per SBC 10per SBC 10per SBC 10per SBC 10per SBC 10per	88585522 2	~~~~~~~~~	~~~^^^
SEC				÷		
Set carry 11ag	۔ 	Implied	DEC	8	-	
SED Set decimal mode	1 -+-0	Implied	SED	F8	•	
SEI						
Set interrupt disable status	1+1	Implied	SEI	82	-	1
STA						
Store accumulator In memory	A M	Zero Page Zero Page, X Absolute Absolute, Y Absolute, Y (Indirect, X) (indirect), Y	STA Oper STA Oper,X STA Oper,X STA Oper,Y STA Oper,Y STA (Oper,Y) STA (Oper,Y)	****	~~~~~	
STX Store index X in memory	₩ + X	Zero Page Zero Page, Y Absolute	STX Oper STX Oper,Y STX Oper,Y	888	30.5	
STY Store index Y in memory	₩+	Zero Page Zero Page,X Absolute	STY Oper STY Oper STY Oper	2.2.2	355	
TAX						
Transfer accumulator to index X	X X	Implied	TAX	AA	1	^^
TAY Transfer accumulator to index Y	A + Y	Implied	TAY	AB	-	^/
TSX			104		•	1
Iransier slack pointer to index X	x to	Implied	Ye I	40	-	/

Name Description	Operation	Addressing Mode	Åssembly Language Form	Code 분	Bytes	No. "P" Status Rep. Byles N 2 C I D V
TXA						
Transfer index X to accumulator	X X	Implied	TXA	84	-	· ·//
TXS						
Transfer index X to stack pointer	× + ×	Implied	TXS	8	÷-	
TYA						
Transfer index Y to accumulator	∀ + ↓	Implied	TYA	8	-	//

HEX OPERATION CODES

00 — BRK	2F — NOP	5E —LSR — 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
- NOP	I		NOP	– LOX –	DO
03 — NOR	- NOP	I	1	87 — NOP	DE — DEC — Absolute, X
04 — NOR	33 — NOP		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 — РНР	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 — MOP	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 NOR	4B	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	4FMOP	7E — 808 — Absolute, X NOP	AD —Absolute	D4 — NOP	FB — NOP
21 — AND —(Indirect, X)	50 — BVC	7F — NOP	AE — LDX — Absolute		FC - NOP
22 — NOR	51 — EOR Indirect, Y	80 — NOR	AFNOR	D6 — DEC — Zero Page, X	FD — SBC — Absolute.X
Τ	52 — NOP	81 — STA — (Indirect, Xi	B0 BCS	07	FE — INC — Absolute, X
24 — BIT — Zero Page	53 — NOP	82 — NOP	81 — LDA — (Indirect), Y	08 — CLD	FF — NOP
— and —	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"tm Higher channel (7-13) modulators generally provide better system performance than lower channel modulators (2-6).
- h. The following power supplies (NOTE: current ratings do not include any capacity for peripheral boards.):
 - 1. +12 Volts with the following current capacity!

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

- b. For 8K, 20K or 32K 550mA.
- c. For 12K, 24K, 36K or 48K 850mA.
- 2. +5 Volts at 1.6 amps
- 3. -5 Volts at WmA.
- 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.

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 "(0" (shift-P) to clear screen.. You may now try "Monitor" commands if you wish. See details in "Ionitor" 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.

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

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

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

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

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

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

> Under no circumstances, should more than 140 VAC be applied to the input of the power supply. Permanent damage will result.

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

Nominal load current ratios are: The +12V supply load is ½ 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 ± 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

NOTES ON INTERFACING WITH THE HOME TV

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

Users report success with the following RF modulators:

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

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

the "Sup-r-Mod" by (assembled & tested) M&R Enterprises P.O. Box 1011 Sunnyvale, CA94088

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 300Q, is presented to the VHF input of the TV set, then stray RFI getting into the TV must be less than 500μ 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 closed in a tight metal box (an unpainted die cast aluminum box such as Pomona #2428). Even with these precautions, some trouble may be encountered with worms, and can be greatly helped by threading the coax cable connecting the modulator to the TV set repeatedly through a Ferrite toroid core Apple Computer supplies these cores in a kit:along with a 4 circuit connector/cable assembly to match the auxilliary video connector found on the Apple II board. This kit has order number A2MØ1ØX. The M&R "Sup-r-Mod is supplied with a coax cable and toroids.

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

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

The ASR - 33 Teletype

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

- a) The yellow wire and purple wire should both go to terminal 9 of Terminal Strip X. If the purple wire is going to terminal 8, then remove it and relocate it at terminal 9. This is necessary to change from the 60mA current loop to the 20mA current loop.
- b) Above Terminal Strip X is a connector socket identified as "2". Pin 8 is the input line + or high; Pin 7 is the input line or low. This connector mates with a Molex receptacle model 1375 #Ø3-Ø9-2151 or #03-09-2153. Recommended terminals are Molex #Ø2-Ø9-2136. An alternate connection method is via spade lugs to Terminal Strip X, terminal 7 (the + input line) and 6 (the input line).
- c) The following circuit can be built on a 16 pin DIP component carrier and then plugged into the Apple's 16 pin socket found at J-14: (The junction of the 3.3k resistor and the transistor base lead is floating). Pins 16 and 9 are used as tie points as they are unconnected on the Apple board. (Figure 1a).

The "RS - 232 Interface"

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

A Serial Out Machine Center Language Program

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

Entering Machine Language Program

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

Executing this Program

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

- 2. PR#Ø will inactivate the printer transfering control back to the Video monitor as the primary output device.
- 3. In Monitor mode \$3700 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

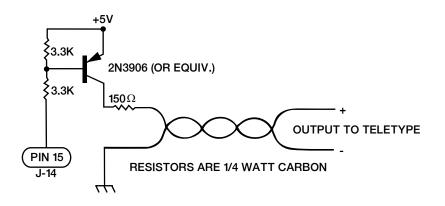
After entering, checking and saving the program perform the following procedure to get a feeling of how the program is used:

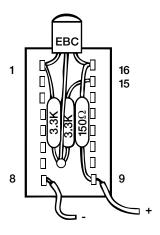
- 1. B^C (control B) into BASIC
- 2. Turn the teletype (printer on)
- 3. Type in the following
 - 10 CALL 880
 - 15 PRINT "ABCD...XYZØ1123456789"
 - 2Ø PR#Ø
 - 25 END
- Type in RUN and hit the "RETURN" key. The text in line 15 should be printed on the teletype and control is returned to the keyboard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







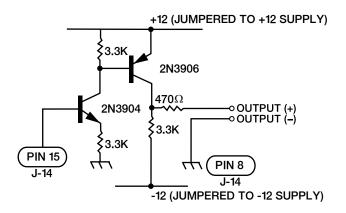


FIGURE 2 RS-232

			TELET	TYPE D	RIVER ROUTI		AOD.
3:42	2 P.M., 2	11/18/19					
		1	TITLE TE	ELETYP	E DRIVER ROU	UTINES'	
		2	******	*****	*****	*	
		3	*			*	
		4	*	TTY	DRIVER:	*	
		5	*]	FELETY	PE OUTPUT	*	
		6	* F	ROUTIN	IE FOR 72	*	
		7	* (COLUMN	I PRINT WITH	*	
		8	* E	BASIC	LIST	*	
		9	*			*	
		10	* COI	PYRIGH	IT 1977 BY:	*	
		11	* API	PLE CC	MPUTER INC.	*	
		12	*	11	/18/77	*	
		13	*			*	
		14	*		IGGINTON	*	
		15	*	S.W	02IV1/IIC	*	
		16	*			*	
		_ <i>i</i>	WNDWDTH		\$21	* ;FOR APPLE-II	
		±0	CH		1	;CURSOR HORIZ.	
		12	CSWL		\$36	; CHAR. OUT SWITCH	
		20	YSAVE		\$778		
			COLCNT		\$7F8	;COLUMN COUNT LOC.	
			MARK		\$C058		
			SPACE	EQU	\$C059		
		25	WAIT	EQU	\$FCA8		
		26		ORG	\$370		
***WAR	NING: OF	ERAND OV	/ERFLOW I	IN LIN	IE 27		
0370:	A9 82	27	TTINIT:	LDA	#TTOUT		
0372:	85 36	28		STA	CSWL	; POINT TO TTY ROUT	INES
0374:	A9 03	29			#TTOUT/256	;HIGH BYTE	
0376:	85 37	30		STA	CSWL+1		
0378:	A9 48	31		LDA		;SET WINDOW WIDTH	0.1TT
037A:	85 21	32		STA		;TO NUMBER COLUMNS	ONT
037C:	A5 24	33		LDA	CH	NULLER ME ADE NOW	

STA

RTS

PHA

CMP

PLA

BCS

PHA

LDA

INC

PLA

PHA

BCC

36 TTOUT: PHA

38 TTOUT2: LDA

44 TESTCTRL:BIT

47 PRNTIT: JSR

COLCNT

COLCNT

TESTCTRL

CH

#\$A0

BEQ PRNTIT

RTS1

COLCNT

DOCHAR

TTOUT2

#\$OD

037E: 8D F8

0381: 60

0382: 48

0383: 48

0384: AD F8

0387: C5 24

038A: BO 03

038D: A9 AO

038F: 2C CO

0392: FO 03

0394: EE F8

0397: 20 C1

039C: 90 E6

039E: 49 OD

03A1: DO OD

039A: 68

0393: 48

03A0: OA

0389: 68

038C: 48

34

35

37

39

40

41

42

45

46

48

49

50

51

52

53

43

FOR А ;ELIM PARITY ASL FINISH BNE

FIGURE 3a

;WHERE WE ARE NOW.

; CHECK FOR A TAB.

;IF C SET, NO TAB

; PRINT A SPACE.

; IF CONTROL CHAR.

;RESTORE CHAR

; RESTORE OUTPUT CHAR.

;TRICK TO DETERMINE

; IF NOT, ADD ONE TO CM

; PRINT THE CHAR ON TTY

; AND PUT BACK ON STAC

; CHECK FOR CAR RET.

; IF NOT CR, DONE.

;DO MORE SPACES FOR TA

;SAVE TWICE

; ON STACK.

TELETYPE DRIVER ROUTINES

2

3:42 P.M.,	, 11/13/1977				PAGE :
03A3:	8D F8 07 54		STA	COLCNT	CLEAR COLUMN COUNT
03A6:	A9 8A 55		LDA	#38A	;NOW DO LINE FEED
03A8:	20 C1 03 56		JSR	DOCHAR	
03AB:	A9 58 57		LDA	#153	
03AD:	20 A8 FC 58		JSR	7AIT	;200MSEC DELAY FOR LIB
0330:	AD F8 07 59	ETNICII.	LDA	COLCNT	;CHECK IF IN MARGIN
0333:	F0 08 60	FINISH:	3E0	SETCH	;FOR CR, RESET CH
0335:	E5 21 61		S3C	7VD7DTH	;IF SO, CARRY SET.
0337:	E9 F7 62		SSC	#SF7	
0339:	90 04 63		BCC	RETURN	
0393:	69 1F 64		ADC	#11F	;ADJUST CH
033D:	85 24 65	SETCH:	STA	CH	
033F:	68 66	RETURN:	PLA		
03C0:	60 67	RTS1:	RTS		;RETURN TO CALLER
03C1:	68	* HERE	STY	TELETYPE PRINT	A CHARACTER ROUTINE:
03C4:	8C 78 07 69	DOCHAR:	PHP	YSAVE	
03C5:	08 70	Docimit.	LDY		;SAVE STATUS.
03C7:	A0 08 71		CLC	#SOS	;11 BITS (1 START, 1 2
03C3:	18 72		PHA		;BEGIN 7ITH SPACE (ST2
03C9:	48 73	TTOUT3:	3CS		;SAVE A REG AND SET FOI
03C3:	80 05 74	110013.	LDA	MARKOUT	
03CE:	AD 59 CO 75		3CC	SPACE	;SEND A SPACE
0300:	90 03 76		LDA	TTOUT4	
0303:	AD 58 CO 77	MARKOUT:	LDA	MARK	;SEND A MARK
0305:	A9 D7 78	TTOUT4:	PHA	#%D7	;DELAY 9.091 MSEC FOR
0306:	48 79	DLY1:	LDA		
03D8:	A9 20 80	DLY2:	LSR	#\$20	
0309:	4A 81		BCC	A	
03D3:	90 FD 82		PLA	DLY2	
03DC:	68 83		SBC		
030E:	6A 84		3NE	#101	
03E0:	88 85		PLA	DLY1	
03E1:	D0 E3 86		ROR		;110 BAUD
03E2:	AC 78 07 87		DEY	A	;NEXT BIT (STOP BITS ?
03E3:	28 88		BNE		LOOP 11 3ITS.
03E5:	60 89		LDY	TTOUT3	
03E8:	90		PLP	YSAVE	;RESTORE Y-REG.
03E9:	91		RTS		;RESTORE STATUS
					;RETURN

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

FIGURE 3b

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

FIGURE 3c

INTERFACING THE APPLE

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

TABLE OF CONTENTS

- 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

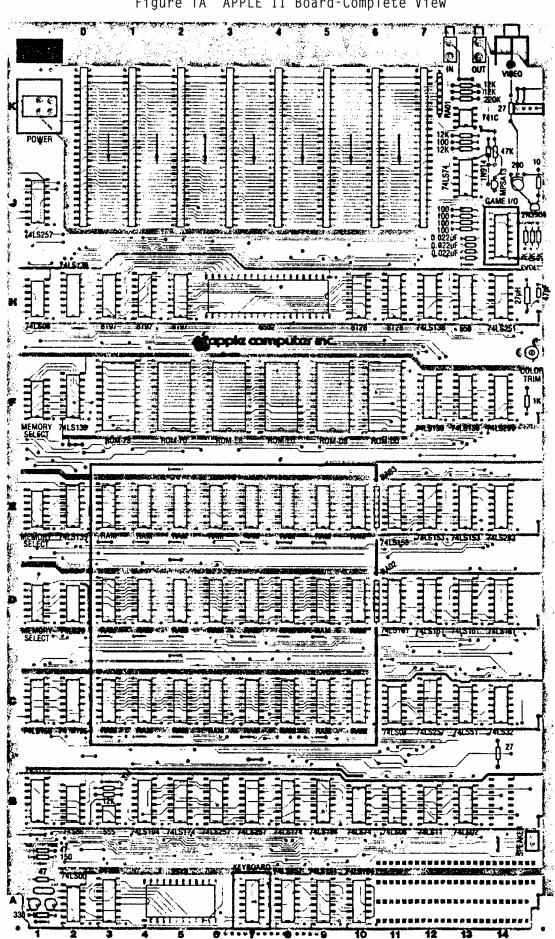
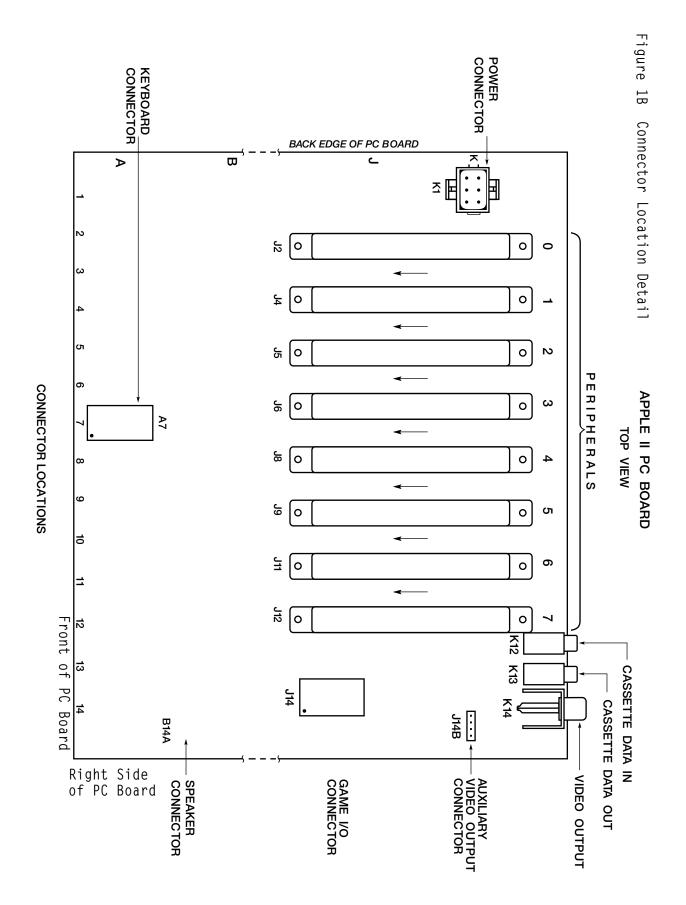


Figure 1A APPLE II Board-Complete View



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_{IN} =1Vpp (nominal), Z_{IN} =12K Ohms. Located at K12 as illustrated in Figure

<u>CASSETTE DATA OUT JACK</u>: Designed for connection to the "MIC" or "MICROPHONE" input found on most audio cassette tape recorders. V_{OUT} =25 mV into 17 Ohms, Z_{OUT} =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

- ANO-AN3: 8 addresses (CØ58-CØ5F) are assigned to selectively "SET" or "CLEAR" these four "ANNUNCIATOR" outputs. Envisioned to control indicator lights, each is a 74LSxx series TTL output and must be buffered if used to drive lamps.
- GND: System circuit ground. O Volt line from power supply.

NC: No connection.

- PDLØ-PDL3: Paddle control inputs. Requires a Ø-150K ohm variable resistance and +5V for each paddle. Internal 100 ohm resistors are provided in series with external pot to prevent excess current if pot goes completely to zero ohms.
- <u>SWØ-SW2:</u> Switch inputs. Testable by reading from addresses CØ61-CØ63 (or CØ69-CØ6B). These are uncommitted 74LSxx series inputs.
- +5V: Positive 5-Volt supply. To avoid burning out the connector pin, current drain MUST be less than 100mA.

KEYBOARD CONNECTOR

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

Figure 3

KEYBOARD CONNECTOR

TOP VIEW

(Front Edge of PC Board)

+5V 1 STROBE 2 RESET 3 N.C. 4 B6 5 B5 6 B7 7 GND 8	•	15 14 13 12 11 10	N.C. -12V N.C. B2 B1 B4 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.
- GND: System circuit ground. Ø Volt line from power supply.
- NC: No connection.
- RESET: System reset input. Requires switch closure to ground.
- STROBE: Strobe output from keyboard. The APPLE II recognizes the positive going edge of the incoming strobe.
- <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 #2HW25CØ-111 (or equivalent) 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

- AO-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 Ø₂ (5ØØnS). 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 6502 less than 300nS after the beginning of \emptyset_2 . During a read cycle the 6502 expects data to be ready no less than 100nS before the end of \emptyset_2 . These lines will drive up to a total of 8 total low power schottky TTL loads.

- DMA: Direct Memory Access control output. This line has a 3K Ohm pullup to +5V and should be driven with an open collector output.
- DMA IN: Direct Memory Access daisy chain input from higher priority peripheral devices. Will present no more than 4 standard TTL loads to the driving device.
- DMA OUT: Direct Memory Access daisy chain output to lower priority peripheral devices. This line will drive 4 standard TTL loads.
- GND: System circuit ground. Ø Volt line from power supply.
- INH: Inhibit Line.When a device pulls this line low, all ROM's on board are disabled (Hex addressed DØØØ through FFFF). This line has a 3K Ohm pullup to +5V and should be driven with an open collector output.
- INT IN: Interrupt daisy chain input from higher priority peripheral devices. Will present no more than 4 standard TTL loads to the driving device.
- INT OUT: Interrupt daisy chain output to lower priority peripheral devices. This line will drive 4 standard TTL loads.
- **I/O SELECT:** 256 addresses are set aside for each peripheral connector (see address map in "MEMORY" section). A read or write of such an address will send pin 1 on the selected connector low during Ø2 (50ØnS). This line will drive 4 standard TTL loads.
- <u>I/O STROBE</u>: Pin 20 on all peripheral connectors will go low during 0, of a read or write to any address C800-OFFF. This line will drive a total of 4 standard TTL loads.
- IRQ: Interrupt request line to the 6502. This line has a 3K Ohm pullup to +5V and should be driven with an open collector output. It is active low.
- NC: No connection.
- <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>₃ 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 6502. This line should change only during 01, 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</u>₀: Microprocessor phase V clock. Will drive up to a total of 16 standard TTL loads.
- 7M: Seven MHz high frequency clock. Will drive up to a total of 16 standard TTL loads.
- +12V: Positive 12-Volt supply.
- +5V: Positive 5-Volt supply
- -5V: Negative 5-Volt supply.
- -12V: Negative 12-Volt supply.

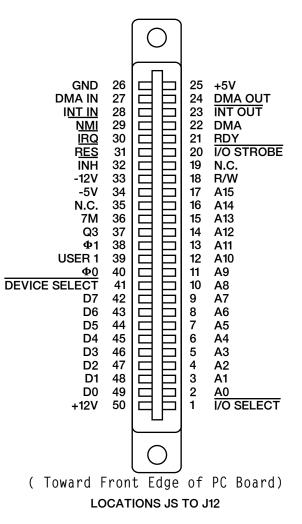
POWER CONNECTOR

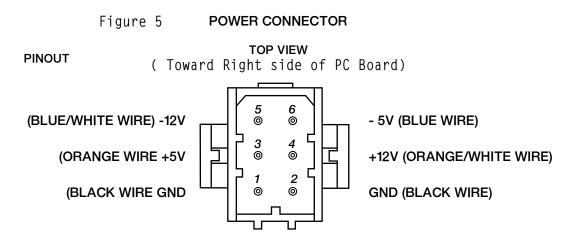
The four voltages required by the APPLE II are supplied via this AMP #9-35028-1,6 pin connector. See location and pin out in Figures 1 and 5.

PIN DESCRIPTION

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







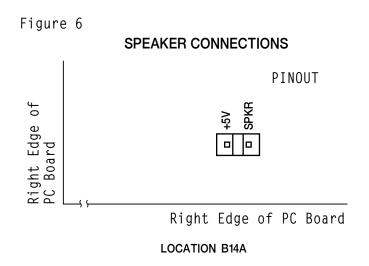
LOCATION K1

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.

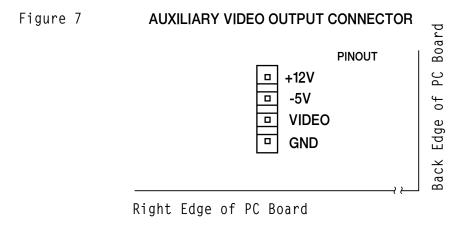
See locations illustrated in Figure 1.

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



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 C
(1000-1FFF) 4K "1" BLOCK 2		13	RAM ROW D
(2000-2FFF) 4K "2" BLOCK 3		12	RAM ROW E
(3000-3FFF) 4K "3" BLOCK 4	1	11	N.C.
(4000-4FFF) 4K "4" BLOCK 5		10	16K "0" BLOCK (0000-3FFF)
(5000-5FFF) 4K "5" BLOCK 6		9	16K "4" BLOCK (4000-7FFF)
(6000-EFFF) 4K "6" BLOCK7		8	16K "8" BLOCK (8000-BFFF)

LOCATIONS D1, E1, F1

MEMORY

TABLE OF CONTENTS

- 1. INTRODUCTION
- 2. INSTALLING YOUR OWN RAM
- 3. MEMORY SELECT SOCKETS
- 4. MEMORY MAP BY 4K BLOCKS5.
- 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, 2CØK 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 nlanual.

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, 8502 stack,	8000	
1000	pointers, etc.	9000	1
2000	high res graphics display primary page	A000	-
3000		B000	-
4000	high res. graphics display secondary page	C000	addresses dedicated to hardware functions
		D000	ROM socket DO: spare
5000			ROM socket D8: spare
	"	E000	ROM socket EO: BASIC
6000			ROM socket E8: BASIC
7000	-	F000	ROM socket FO: BASIC utility
1000			ROM socket F8: monitor

Memory Map Pages Ø to BFF

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

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

HEX ADDRESS	ASSIGNED FUNC	TION	COMMENTS		
C060/8	Cassette input		State of "Cassette Data In" appears in bit 7.		
C061/9	"SW1"		input on State of Switch 1 \land 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 "	u.	State of timer output for Paddle 3 appears in bit 7.		
C07X	"PDL STB"		Triggers paddle timers during ϕ_2 .		
C08X	DEVICE SELECT	0	Pin 41 on the selected		
C09X	"	1	Peripheral Connector goes low during ϕ_2 .		
COAX		2			
COBX		3			
COCX		4			
CODX		5			
COEX		6			
COFX		7			
C10X		8	Expansion connectors.		
C11X		9			
C12X		A			

HEX ADDRESS	ASSIGNED FUNCTION			ON	COMMENTS	
C13X	DEVICE SELECT B					
C14X			С			
C15X			D		Π.	
C16X	"		Е		u.	
C17X			F			
CIXX	I/O SELEC	T	1		Pin 1 on the selected Peripheral Connector goes low during φ_2 .	
C2XX			2			
СЗХХ	"		3		NOTES:	
C4XX	"		4		1. Peripheral Connector 0 does not get this	
C5XX			5		signal. 2. $\overline{I/O \text{ SELECT}}$ 1 uses the	
C6XX			6		same addresses as DEVICE SELECT 8-F.	
C7XX			7			
C8XX			8,	I/O STROBE	Expansion connectors.	
сэхх			9,	"		
CAXX			A,	ų		
CBXX			В,	п.		
CCXX	· 11		C,	n		
CDXX	11		D,	"		
CEXX	"		Е,	н		
CFXX	"		F,			
D000-D7FF	ROM socket	5 DO			Spare.	
D800-DFFF		D8		-4	Spare.	
E000-E7FF	н н	EO			BASIC.	
E800-EFFF		E8			BASIC.	
F000-F7FF		FO			1K of BASIC, 1K of utility.	
F800-FFFF		F8			Monitor.	

SYSTEM TIMING

SIGNAL DESCRIPTIONS

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

COLOR REF: Color reference frequency used by video circuitry, 3.530 MHz.

- $Ø_0$: Phase Ø clock to microprocessor, 1.023 MHz nominal.
- $\underbrace{ \emptyset_1 : }_{ nominal.}$ Microprocessor phase l clock, complement of \emptyset_0 , l.023 Mhz
- $\underbrace{ \emptyset_2 } \\ \text{Same as } \emptyset_0. \text{ Included here because the 6502 hardware and} \\ \text{programming manuals use the designation } \emptyset_2 \text{ instead of } \emptyset_0.$
- <u>Q3:</u> A general purpose timing signal which occurs at the same rate as the microprocessor clocks but is nonsymmetrical.

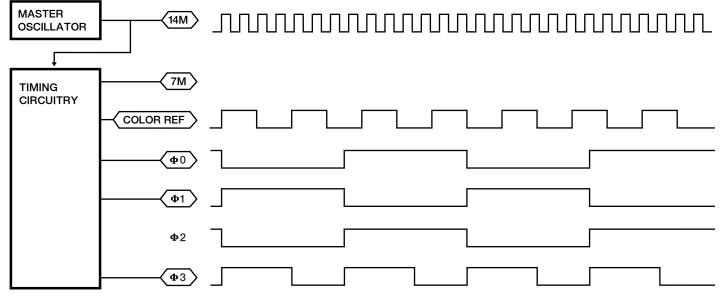
MICROPROCESSOR OPERATIONS

- <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 300nS 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



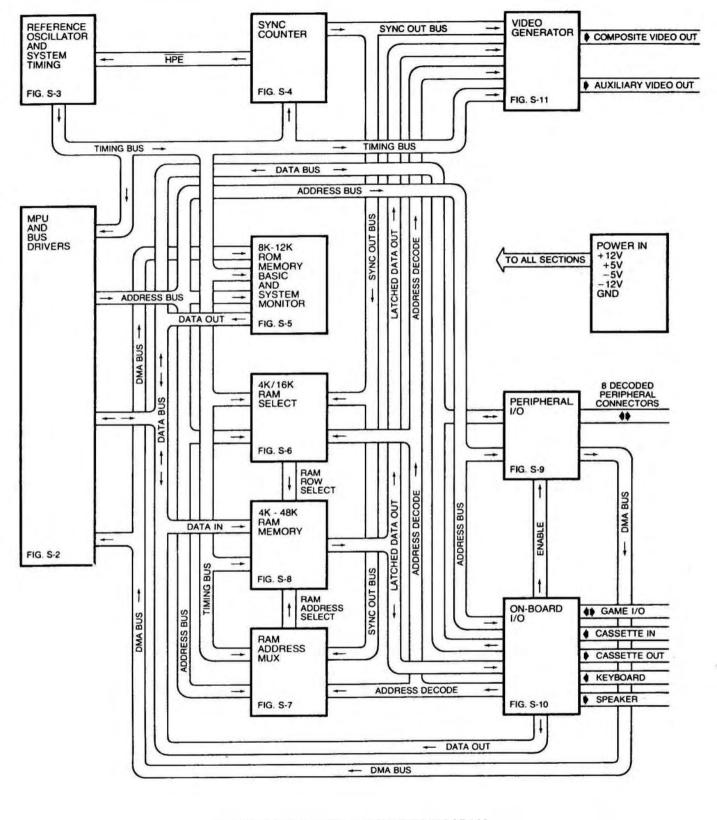
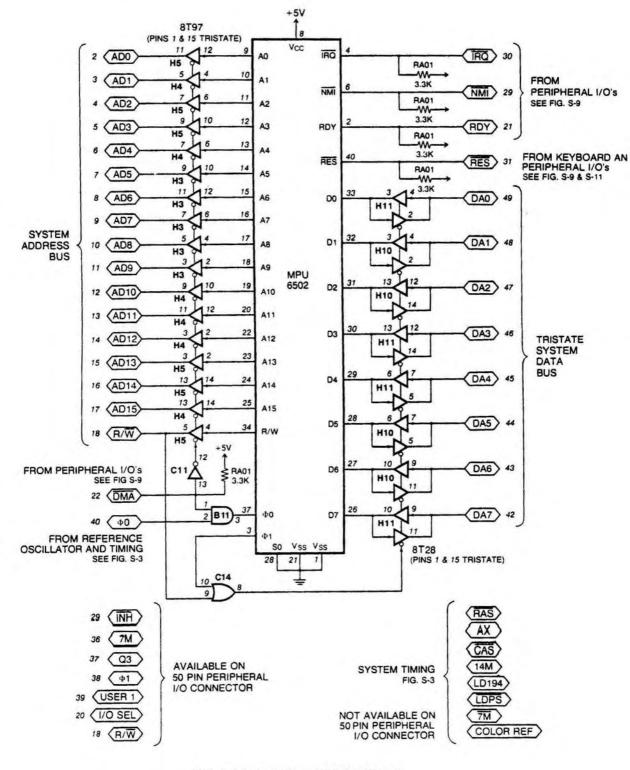


FIGURE S-1 APPLE II SYSTEM DIAGRAM





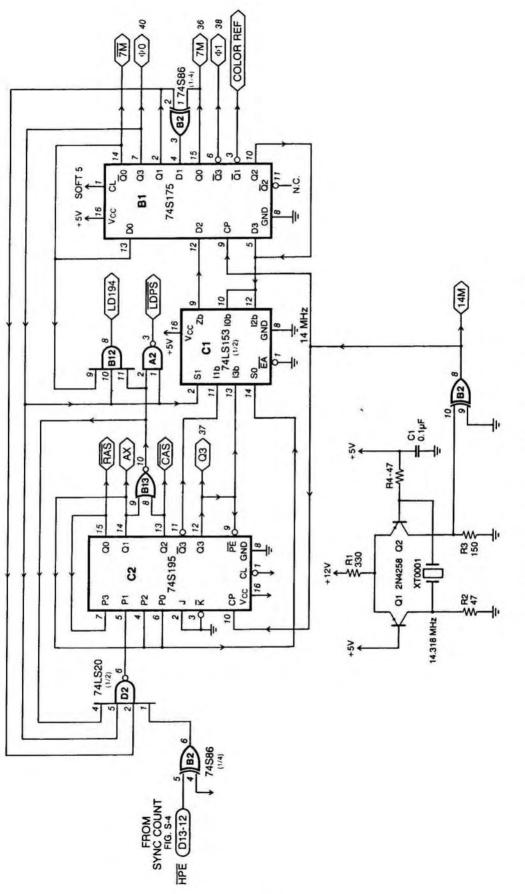
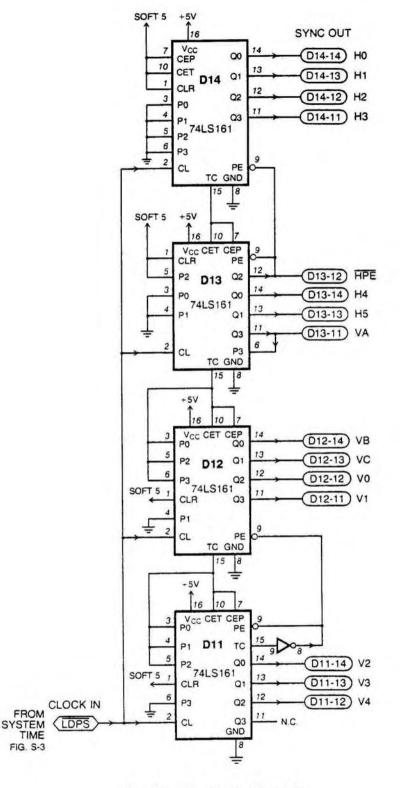


FIGURE S-3 REFERENCE OSCILLATOR AND SYSTEM TIMING

143





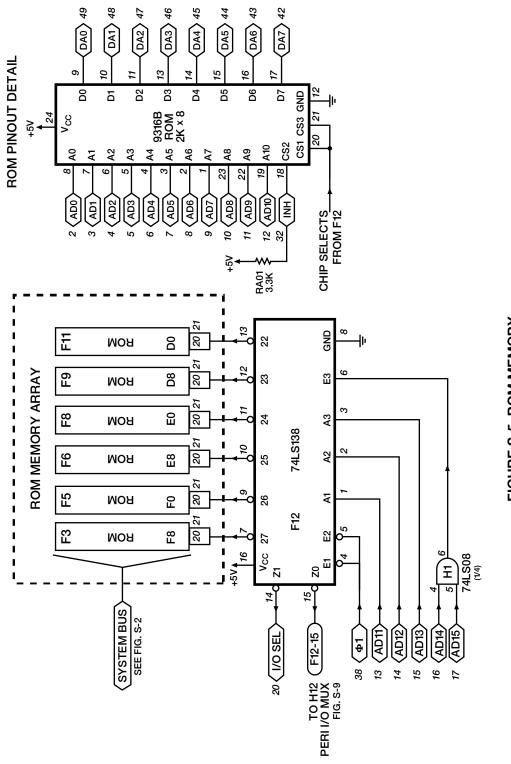


FIGURE S-5 ROM MEMORY

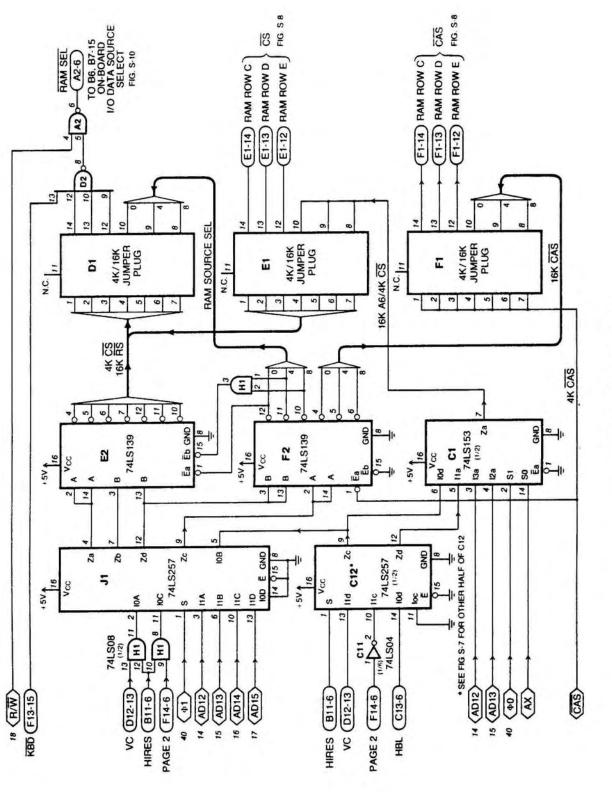
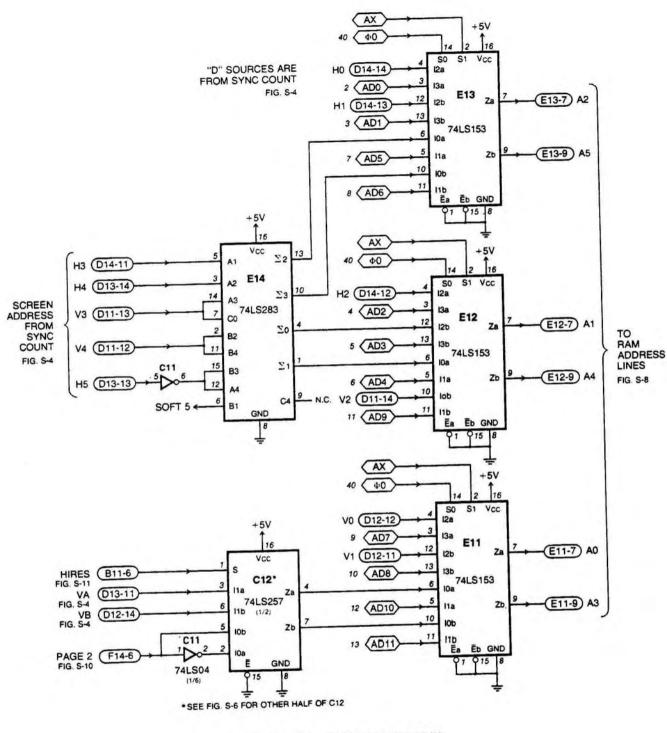


FIGURE S-6 4K/16K RAM SELECT





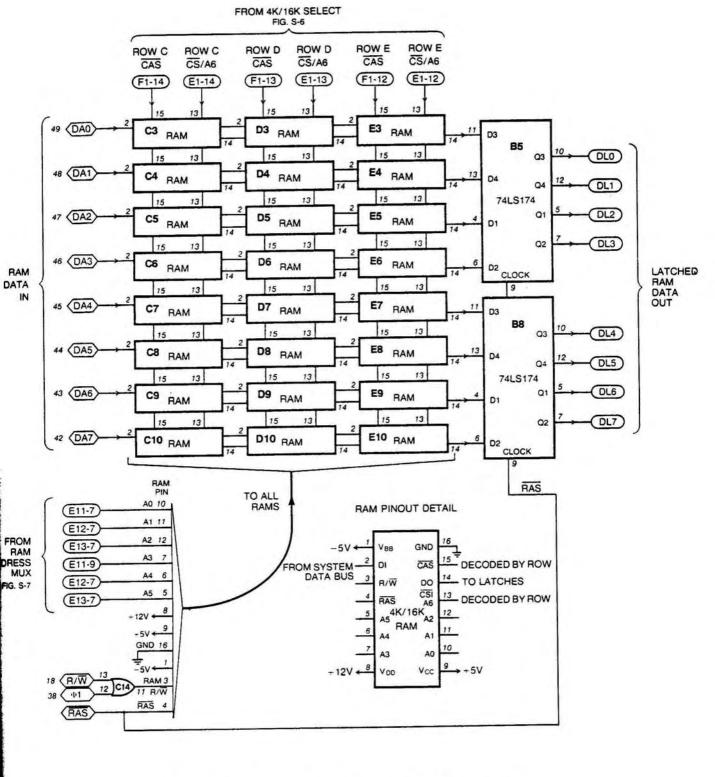
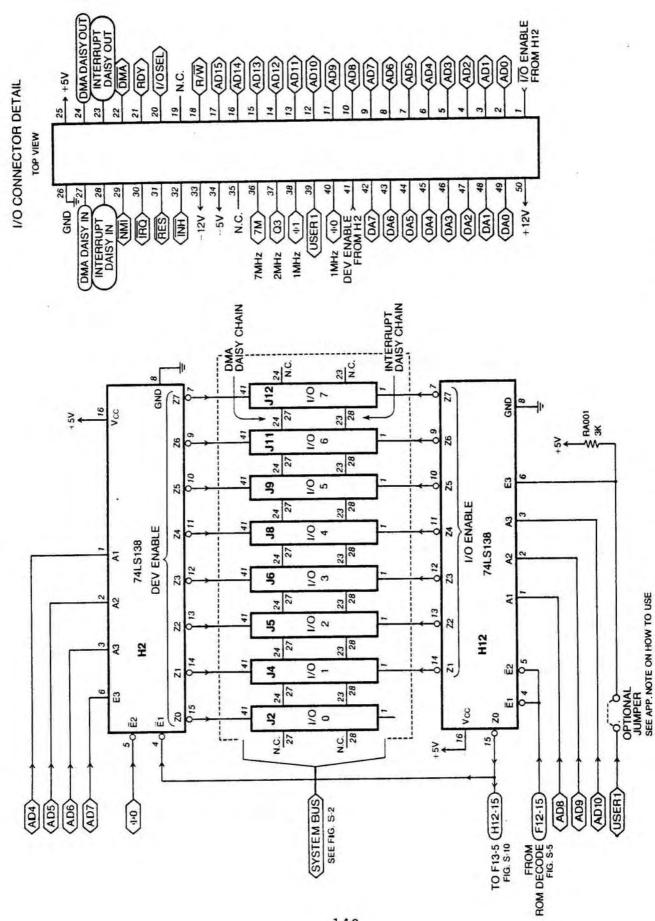


FIGURE S-8 4K TO 48K RAM MEMORY WITH DATA LATCH



149

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

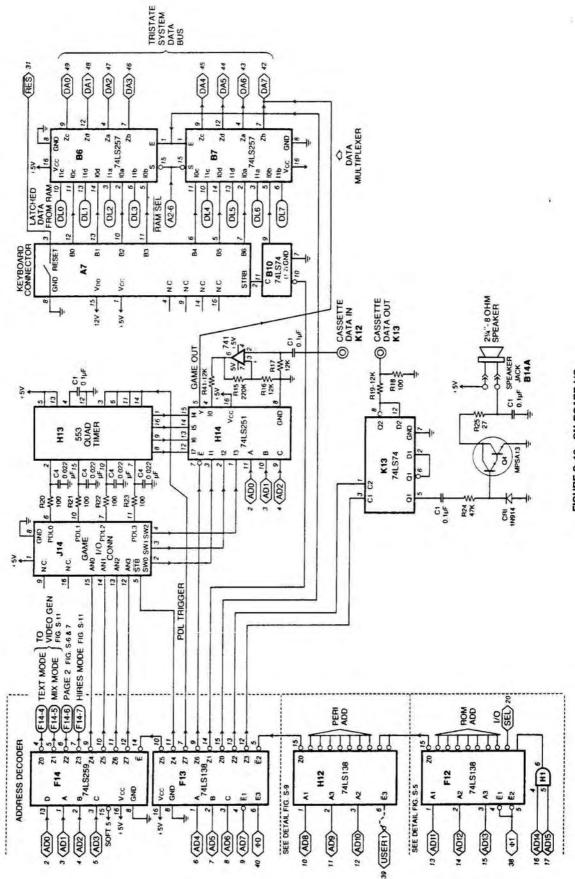


FIGURE S-10 ON-BOARD I/O

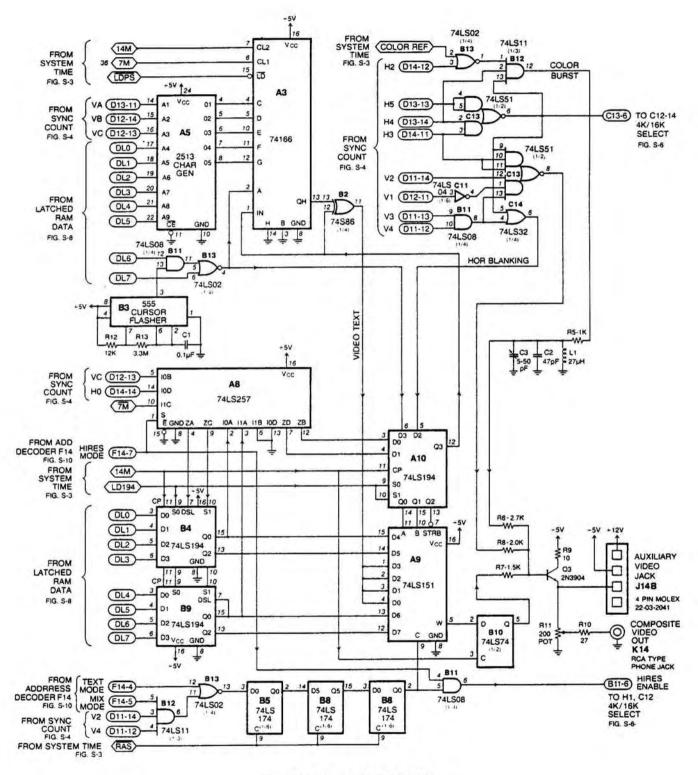


FIGURE S-11 VIDEO GENERATOR



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



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