Apple-I[™] Operation Manual



20863 Stevens Creek Blvd. B 3-C Cupertino, California 95014 (408) 996-1010

SPECIFICATIONS

MICROPROCESSOR:	MOS TECHNOLOGY 6502	
Microprocessor Clock Frequency:	1.023 MHz	
Effective Cycle Frequency: (Including Refresh Waits)	0.960 MHz	
VIDEO OUTPUT:	Composite positive video, 75 ohms, level adjustable between zero and +5Vpp.	
Line Rate:	15734 Hz	
Frame Rate:	60.05 Hz	
Format:	40 characters/line, 24 lines; with automatic scrolling	
Display Memory:	Dynamic shift registers (1K x 7)	
Character Matrix:	5 x 7	
RAM MEMORY:	16-pin, 4K Dynamic, type 4096 (2104)	
On-board RAM Capacity:	8K bytes (4K supplied)	
POWER SUPPLIES:	+5 Volts @ 3 amps, +/- 12 Volts @0.5 amps, and -5 Volts @ 0.5 amps	
Input Power Requirements:	8 to 10 Volts AC (RMS) @ 3 amps, 26 to 28 Volts AC (RMS) Center-Tapped, 1A.	
Recommended Transformers:	Stancor # P-8380 or Triad F31-X Stancor # P-8667 or Triad F40-X	



20863 Stevens Creek Blvd. B 3-C Cupertino, California 95014 (408) 996-1010

©1977 Apple Computer, Inc.

INTRODUCTION

The Apple Computer is a complete microprocessor system, consisting of a Mos Technology 6502 microprocessor and support hardware, integral video display electronics, dynamic memory and refresh hardware, and fully regulated power supplies. It contains resident system monitor software, enabling the user, via the keyboard and display, to write, examine, debug, and run programs efficiently; thus being an educational tool for the learning of microprocessor programming, and an aid in the development of software.

The integral video display section and the keyboard interface renders unnecessary the need for an external teletype. The display section contains its own memory, leaving all of RAM for user programs, and the output format is 40 characters/ line, 24 lines/page, with auto scrolling. Almost any ASCII encoded keyboard will interface directly with the Apple system.

The board has sockets for upto 8K bytes of the 16 pin, 4K type, RAM, and the system is fully expandable to 65K via the edge connector. The system uses dynamic memory (4K bytes supplied), although static memory may also be used. All refreshing of dynamic memory, including all "off-board" expansion memory, is done automatically. The entire system timing, including the microprocessor clock and all video signals, originates in a single crystal oscillator.

Further, the printed circuit board contains a "breadboard area", in which the user can add additional "on-board" hardware (for example, extra PIA's, ACIA's, EROM's, and so on).

This manual is divided into three Sections:

Section I GETTING THE SYSTEM RUNNING. Section II USING THE SYSTEM MONITOR. (listing included) Section III EXPANDING THE SYSTEM.

Please read Section I thoroughly, before attempting to "power-up" your system, and study Section III carefully before attempting to expand your system. In addition to this manual, Apple "Tech Notes" are available which contain examples of expansion hardware and techniques.

SECTION I GETTING THE SYSTEM RUNNING

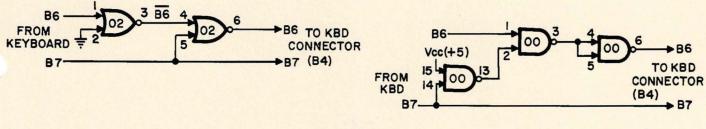
The Apple Computer is fully assembled, tested, and burned in. The only external devices necessary for operation of the system are: An ASCII encoded keyboard, a video display monitor, and AC power sources of 8 to 10 Volts (RMS) @3 amps and 28Volts (RMS) @1 amp. The following three articles describe the attachment of these devices in detail.

Keyboard:

Any ASCII encoded keyboard, with positive DATA outputs, interfaces directly with the Apple system via a "DIP" connector. If your keyboard has negative logic DATA outputs (rare), you can install inverters (7404) in the breadboard area. The strobe can be either positive or negative, of long or short duration. The "DIP" keyboard connector (B4) has inputs for seven DATA lines, one STROBE line, and two normally-open pushbutton switches, used for RESET (enter monitor), and CLEAR SCREEN (see schematic diagram, sheet 3 of 3, for exact circuitry). This keyboard connector also supplies three voltages, (+5V, +12V,and -12V) of which one or more may be necessary to operate the keyboard. Pin 15 of the keyboard connector (B4) must be tied to +5V (pin 16) for normal operation.

NOTE: The system monitor accepts only uppercase alpha (A-F, R).

It is therefore convenient, though it's not essential, to have a keyboard equipped with uppercase alpha lock (usually in the electronics). Either of the following suggested circuits may be used to provide alpha lock capability, if needed, and can be built in the breadboard area.



Display:

The Apple Computer outputs a composite video signal (composite of sync and video information) which can be applied to any standard raster-scan type video display monitor. The output level is adjustable with the potentiometer located near the video output Molex connector, J2. The additional two outside pins on the Molex connector supply +5 and +12 volts, to be used in future Apple accessories. The composite video signal can also be modulated at the proper RF frequency, with an inexpensive commercially available device, and applied to the antenna terminals of a home television receiver. Since the character format is 40 characters / line, all television receivers will have the necessary bandwidth to display the entire 40 characters. Two large manufacturers of video display monitors, which connect directly with the Apple Computer, are Motorola and Ball. The mating four-pin Molex connector is provided.

AC Power Sources:

Two incoming AC power sources are required for operation: 8 to 10 VAC (RMS) at 3 amps, and 28 VAC (RMS) Center-Tapped at lamp. These AC supplies enter the system at the Molex connector, J1. The 8 to 10 volts AC provides the raw AC for the +5 volt supply, while the 28 VCT supplies the raw AC for the +12 and -12 volt supplies, and the -5V supply is derived from the -12V regulated output.

The board, as supplied, requires no more than 1.5 amps DC from the +5V supply, while the regulator is capable of supplying 3 amps. The remaining 1.5 amps DC from the +5V supply is available for user hardware expansion (provided suitable transformer ratings are employed).

A suitable source of the raw AC voltages required, are two commercially available transformers; Stancor P/N P-8380 or equivalent (8 to 10 volts at 3 amps), and Stancor P/N P-8667 or equivalent (28VCT at 1 amp). Simply wire the secondaries to the mating six-pin Molex connector supplied, and wire the primaries in parallel, as shown in the schematic diagram (power supply section, Dwg. No. 00101, sheet 3 of 3.

TEST PROGRAM

After attaching the keyboard, display, and AC power sources, you can try a simple program to test if your system and the attachments are functioning together properly. While it does not test many possible areas of the microprocessor system, the test program will test for the correct attachment of the keyboard, display, and power supplies.

FIRST:

Hit the RESET button to enter the system monitor. A backslash should be displayed, and the cursor should drop to the next line.

SECOND:

Type- Ø: A9 bØ bAA b 2Ø bEF bFF b E8 b 8A b 4C b 2 bØ (RET) (Ø is a zero, NOT an alpha "O"; b means blank or space; and (RET) hit the "return" key on the keyboard)

THIRD:

Type- \emptyset . A (RET) (This should print out, on the display, the program you have just entered.)

FOURTH:

Type- R (RET) (R means run the program.)

THE PROGRAM SHOULD THEN PRINT OUT ON THE DISPLAY A CONTINUOUS STREAM OF ASCII CHARACTERS. TO STOP THE PRO-GRAM AND RETURN TO THE SYSTEM MONITOR, HIT THE "RESET" BUTTON. TO RUN AGAIN, TYPE : R (RET).

SECTION II USING THE SYSTEM MONITOR

The Hex Monitor is a PROM program in locations FFØØ to FFFF (hex) which uses the keyboard and display to perform the front panel functions of examining memory, and running programs. The monitor program is entered by hitting (RESET), which displays backslash - return. A backslash alone (cursor remains on same line as backslash) indicates bad page 0 RAM.

Commands are typed on a "line-at-a-time" basis with editing. Each line may consist of any number of commands (up to 128 characters). None are executed until (RETURN) is typed. The (SHIFT-0) (backarrow) backspaces and echos an underline. The (ESC) cnacels a line and echos backslash-return.

One or more hexadecimal digits (0-9, A-F)are used for address and data values. Addresses use the four least significant digits of a group, and data values, the two least significant digits. The following examples illustrate the variety of acceptable commands:

1.	Opening a location (examining the contents			
	of a single address).			
	USER TYPES/	4F (RET)		
	MONITOR TYPES/	ØØ4F: ØF (contents		
		of 4F)		

- 2. Examining a block; from the last examined location, to a specified one. USER TYPES/ .5A (RET) MONITOR TYPES/ ØØ5Ø: ØØ Ø1 Ø2 Ø3 Ø4 Ø5 Ø6 Ø7 ØØ58: Ø8 Ø9 ØA
- Note: 4F is still considered the most recently opened location.

 Combining examples 1 and 2 to print a block of memory in a single command. USER TYPES/ 4F.5A (RET) MONITOR TYPES/ ØØ5Ø: ØØ Ø1 Ø2 Ø3 Ø4 Ø5 Ø6 Ø7 ØØ58: Ø8 Ø9 ØA

- Note: Only the first location of the block (4F) is considered "opened".
- 4. Examining several individual locations at once.

USER TYPES/ 4F b 52 b 56 (RET) MONITOR TYPES/ ØØ4F: ØF ØØ52: Ø2 ØØ56: Ø6

- Note: 56 is considered the most recently "opened" location. The "b" is a blank or comma, and is a delimiter for separation purposes only. A string of delimiters has the same effect as a single one (bbb is as effective as b).
- 5. Examining several blocks of memory at once.

USER TYPES/	4F.52 b 56 b 58.5A
	(RET)
MONITOR TYPES/	ØØ4F: ØF
	ØØ5Ø: ØØ Ø1 Ø2
	ØØ56: Ø6
	ØØ58: Ø8 Ø9 ØA

- Note: 58 is considered the most recently "opened" location. Refer to example 2.
- 6. Examining successive blocks. 4F.52 (RET) USER TYPES/ MONITOR TYPES/ ØØ4F: ØF ØØ5Ø: ØØ Ø1 Ø2 USER TYPES/ .55 (RET) ØØ53: Ø3 Ø4 Ø5 MONITOR TYPES/ USER TYPES/ .5A (RET) MONITOR TYPES/ 0056: 06 07 ØØ58: Ø8 Ø9 ØA
- Depositing data in a single location. USER TYPES/ 3Ø: AØ (RET) MONITOR TYPES/ ØØ3Ø: FF (prior contents)
- Note: Location 30 is considered opened and now contains 30.
- Bepositing data in successive locations from that last used in a deposit command.
 USER TYPES/ : A1 b A2 b A3 b A4 b A5 (RET)
 (This deposits A1 in location 31, A2 in 32,

and so on.)

9. Combining examples 7 and 8 in a single command.
 USER TYPES/ 30: A0 b A1 b A2 b

A3 b A4 b A5 (RET) MONITOR TYPES/

ØØ3Ø: FF (prior contents of location 3Ø)

 Depositing data in successive locations with separate commands. USER TYPES/ 30: A0 b A1 (RET)

USER TYPES/	39: AV DAI (REI)
MONITOR TYPES/	ØØ30: FF
USER TYPES/	:A2 b A3 (RET)
USER TYPES/	:A4 b A5 (RET)

NOTE: Capital letters enclosed in parenthesis represent single keystrokes. Example: (RET) means hit the "return" key.

- Note: A colon in a command means "start depositing data from the most recently deposited location, or if none, then from the most recently opened one.
- 11. Examining a block, then depositing into it. USER TYPES/ 30.35 (RET) MONITOR TYPES/ ØØ3Ø: AØ A1 A2 A3 A4 A5 A6 USER TYPES/ :B0 b B1 b B2 b B3 b B4 b B5 (RET)
- Note: New data deposited beginning at most recently opened location (30)
- Run a program at a specified address.
 USER TYPES/ 1ØFØ R (RET)
 MONITOR TYPES/ 1ØFØ: A9 (contents)
- Note: The cursor is left immediately to the right of the "A9"; it is not returned to the next line.
- Run at the most recently examined location. USER TYPES/ 1ØFØ (RET) MONITOR TYPES/ 1ØFØ: A9 USER TYPES/ R (RET)
- 14. Enter a program into memory and run it in one line. USER TYPES/ 4Ø: A9 bØb 2Ø b EF b FF b 38 b 69 b Ø b 4C b 4Ø bØ R (RET) MONITOR TYPES/ 40: FF (prior contents of 4Ø)

MONITOR TYPES/ 40: FF (prior contents of 40)

- 15. An "on line" error correction. USER TYPES/ 4Ø: A1 b A2 b A3A4A5A6 b A7 (data A6 will be loaded in location 42) USER TYPES/ 4Ø5Ø6Ø7Ø: AA (data AA will be loaded in location 6Ø7Ø)
- 16. Useful routines in monitor which can be accessed by user programs.GETLINE: location FF1F:

monitor entry point (jumping to FF1F will enter monitor and echo carriage return. You can then examine memory locations with the monitor.)

- ECHO: location FFEF: prints one byte (ASCII) (data from "A" (accumulator), contents of "A" not disturbed. Example: 20 b EF b FF (JRS ECHO)).
- PRBYTE: location FFDC: prints one byte (HEX) (data from "A", contents of "A" disturbed.)

PRHEX: location FFE5: prints one hex digit (data from four least significant bits of "A", contents of "A" disturbed.)

NOTE: RAM locations $\emptyset\emptyset24$ to $\emptyset\emptyset2B$ are used as index pointers by the monitor, and are invalid for user use, when using monitor. Also, locations $\emptyset2\emptyset\emptyset$ to $\emptyset27F$ are used as input buffer storage, and are also invalid for user use when using the monitor.

-4-

6502 HEX MONITOR LISTING

FPF0D8RESETCLDClear desimal arithmetic mode.FPF056CLIFPF086 (12 D0)ST DSPSet iup.FPF080 (12 D0)STA DSP CAFab (12 D0)FPF080 11 D0STA MBD CAEnable interrupts, set CAI, CBI, forFPF080 11 D0STA DSP CApositive edge sense/output mode.FPF060 11 D0STA DSP CApositive edge sense/output mode.FPF0C0 PFNOTCRCMP 459BESC?FP11F0 35BEC DSCAPEYes.FP1170 35BEC DSCAPEYes.FP1270 36GETLINELDA 458GFP1360 47JSR ECHOOutput it.FP14A9 DCESCAPELDA 458GFP1226 EF FFJSR ECHOOutput it.FP1226 EF FFJSR ECHOOutput it.FP2340 f1Initialize text index.FP24A0 f1DY 8501Entitialize text index.FP2519 D0BEY MSTCHARLDA 4500FP2619 D4DA KDD CRKey ready?FP2719 D4DA KDD CRKey ready?FP23A0 11 D0NEXTCHARLDA 4500FP39A0 FFJSR ECHOROutput it.FP3179 D5STA NNAdot text index.FP2219 D4DA KBD CRKey ready?FP23A0 11 D0NEXTCHARLDA KBD CRFP33A0 FFJSR ECHODiplay character.FP34A0 FFJA KBD CR </th <th></th> <th></th> <th></th> <th></th> <th></th> <th></th>						
FF92A4 7FLDY 487FMask for DSP data direction register.FF9480 11 D4ST DSPSet tup.FF9680 11 D4ST A DSP CRpositive edge sense/output mode.FF9680 11 D4ST A DSP CRpositive edge sense/output mode.FF9760 DFNOTCRCMP 450BESC?FF11F4 13BC DSCAPEYes.FF1369 9BCMP 459BESC?FF1464 6FBPL NEXTCHARAuto ESC if > 127.FF1514 6FBPL NEXTCHARAuto ESC if > 127.FF1626 FFJA 450C''.'.'.FF1224 FFJS ECRIOOutput it.FF1224 FFJS ECRIOOutput it.FF278 BACKSPACEDY 4801Initialize text index.FF2730 F6EM GETLINEBey Med CRFF28A9 01NEXTCHARLook 4800FF2730 F6EM GETLINEFF28A9 01LOA KBDFF2930 09 09 02STA IN, YAddu to text buffer.Bey Med CRFF3329 09 04LDA 4580FF3426 FFJS KECRIOFF339 09 04LDA 4580FF3430 FFLDA KBDFF3430 FFFF359 09 04FF3426 FFJS RCHARLDA KBDFF3430 FFFF3330 FFFF3430 FFFF3330 FFFF3430 FFFF35A9 FFFF44				RESET		Clear decimal arithmetic mode.
FF04Set it up.Set it up.FF0789 A7LDA #\$A7KBD and DSP control register mask.FF068D 11 D0STA KBD CREnable interrupts, set CA1. CB1, forFF068D 12 D0STA DSP CRpositive edge sense/output mode.FF078D 11 D0STA DSP CRrew?FF078D 17CMP #\$DF"e*?FF11F61 3BEC BACKSPACEYes.FF11F61 3BEC ASCAPEYes.FF11F60 3BEC ESCAPEYes.FF1116 4FBPL NEXTCHARAuto ESC if > 127.FF1216 4FBEL NEXTCHARAuto ESC if > 127.FF14A9 DCESCAPELDA #\$A0CR.FF1236 FFISR ECHOOutput it.FF1247 41LDA #\$A1Initiallise text index.FF24A6 41LDY #\$41Initiallise text index.FF2730 F6BACKSPACEDEYBackup text index.FF2730 F6BMI GETLINEBeyond start of line, reinitialize.FF2710 10 0NEXTCHARBDE NOTCRNo.FF3990 49 49STA LDA KBD CRNo.FF3990 49 49STA EGNODig text index.FF3991 10 0NEXTCHARLDA KBD CRFF3990 49 49STA EGNODig text index.FF3440 FF FFJSR ECHODig text index.FF3450 46 FF FSTA EGNODig text index.FF3450 47 FFBEC SCAPESTA EGNOFF3450						
FF70A9 A7LDA $\$5A7$ KBD and DSP control register mask.FF708D 11 DØSTA KBD CRpositive edge sense/output mode.FF7060 13 DØSTA KBD CRpositive edge sense/output mode.FF7060 DFNOTCRGMP $\$5DF$ " $e^+?$ FF11F0 13BEO BACKSPACEYes.FF11F0 3BEO ESCAPEYes.FF11F0 3BEO ESCAPEYes.FF12C8INYAdvance text index.FF14A9 DCESCAPELDA $\$5D$ FF1520 EF FFJSR ECHOOutput it.FF1620 EF FFJSR ECHOOutput it.FF2120 EF FFJSR ECHOOutput it.FF268BACKSPACEDEYBACKSPACEDEYBackup text index.FF2730 76BMI GETLINEBeyond start of line, reinitialize.FF2730 76BMI GETLINEBeyond start of line, reinitialize.FF2730 90 0CMP $\$3D$ CR?FF3320 FF FJSR ECHODisplay character.FF3390 90 42STA KBO CRNo.FF3320 FF FLDY $\$30$ CR?FF3390 90 4BEN NOTCRNo.FF3320 FF FLDY $\$30$ CR?FF3420 FF FLDY $\$30$ CR?FF3320 FF FLDY $\$30$ CR?FF3420 FF FLDY $\$30$ CR?FF3390 90 42BEN NOTCRNo.FF3420 FF FLDY $\$30$ CR?<						
FF008D 11 D0STA KBD CREnable interrupts, set CA1, CB1, forFF008D 13 D0STA KBD CRPositive edge sense/output mode.FF008D 13 D0STA KBD CR" \bullet "?FF01F013BEO BACKSPACEYes.FF11F013BEO BACKSPACEYes.FF11F013BEO ASCAPEYes.FF11F013BEO ESCAPEYes.FF11A9 DCESCAPELDA #SDCFF12A9 DCESCAPELDA #SDCF714A9 DCESCAPELDA #SDCF715F403GETLINELDA #SDCF716F404GETLINELDA #SDCF71778GETLINELDA #SDCF718F400GETLINELDA #SDCF72120 EF FFJSR ECHOOutput it.F72288BACKSPACEDETBackup text index.F723900LDA KBD CRKey ready?F724A0 01DONEXTCHARLDA KBD CRF725A0 10 D0NEXTCHARLDA KBD CRF72616 FBBUN EXTCHARLDA KBD CRF72720 J6 D0CMP #\$ADCachestrater.F73190 00 4BNE NOTCRNo.F73220 FF FFJSR ECHODisplay character.F73390 00 4BNE NOTCRNo.F74185 ZETSTORASLF74380 FFLDA #\$00F74480 40 FFLDA #\$00F74490 40SEC SETSTORF74490 40 <td></td> <td></td> <td></td> <td></td> <td></td> <td></td>						
FFGCSD IS D0STA DSP CRpositive edge sense/output mode.FFGCSD DFNOTCRNOTCRNOTCRNOTCRFF11F0 13BEQ BACKSPACEYes.FF11F0 03BEQ ESCAPEYes.FF11F0 03BEQ ESCAPEYes.FF11C8INYAdvance text index.FF11A 9 DCESCAPELDA #\$DCFF14A 9 DCESCAPELDA #\$DCFF14A 9 DCESCAPELDA #\$DCFF14A 9 DCESCAPELDA #\$DCFF14A 9 DCESCAPELDA #\$DCFF24A 9 DCESCAPELDA #\$DCFF24A 9 DCESCAPELDA #\$DCFF24A 9 DCESCAPELDA #\$DCFF24A 9 DCESCAPEDA #\$DCFF24A 9 DCESCAPEDA #\$DCFF24A 9 DCESCAPEDA #\$DCFF24A 9 DCESCAPEDA #\$DCFF25A D1 DØNEXTCHARLDA #\$DCFF26A 9 DLDA KBD CRKey ready?FF27A 0 1 DØNEXTCHARLDA KBD CRFF319 0 Ø QEF FFISR ECHODisplay character.FF33A 9 FFLDA KBD CRCR?FF33A 9 FFLDY #\$FFReset text index.FF3420 FF FSE ESTAPDESTA MODEFF4464FS ESTAPDESTA MODEFF4489 Ø Ø C2NEXT ITEMLDA #\$90FF4489 Ø Ø G2NEXT ITEMLD	~	and the second				
FF0FC GMP #3DF" \bullet "?FP11F013BEG BACKSPACEYes.FP13G 9 95GMP #395ESC ?FP15F0 03BEG BACKSPACEYes.FP17C8INYAdvance text index.FP18If ØFBPL NEXTCHARAuto ESC if >127.FP18A9 DCESCAPELDA #3DC"N".FP1620 EF FFJSR ECHOOutput it.FP17A9 DCESCAPELDA #3DCCR.FP18A9 DCGETLINELDA #3DCCR.FP2120 EF FFJSR ECHOOutput it.FP2210 F6BACKSPACEDETBackup text index.FP2330 F6BACKSPACEDETBackup text index.FP24A0 01NEXTCHARLDA KSD CRKey ready?FP2510 F6BACKSPACEDETBackup text index.FP2610 F76MG ETLINELDA KSD CRKey ready?FP2730 F6GMF #58DCR?NoFP3390 ØLDA KSDCR?NoFP3420 EF FFJSR ECHODisplay character.F73729 SDGMP #58DCR?NoF73729 SDGMP #58DCR?F73729 SDGMP #58DCR?F738A9 GFLDA #590For XAM mode.F74930 FFCMP #58DCR?F74940 FFSETSTORASLF74940 FFBC CETLINEYes, setSTOR mode.F74185 BBL						
FFI1FV 13EC D BACKSPACEYes.FF13G 9 BCMP #\$9BESC?FF15FØ 03BCO ESCAPEYes.FF17G 8INYAdvance text index.FF18IØ ØFBPL NEXTCHARAuto ESC if > 127.FF14A 9 DCESCAPELDA #\$DC"\".FF17C 0 EF FFJSR ECHOOutput it.FF17A 9 8DGETLINELDA #\$8DCR.FF17A 9 01GETLINELDA #\$8DCR.FF2730 F6BMCKSPACEBeckup text index.FF2730 F6BMI GETLINEBeckup text index.FF2730 F6BMI GETLINEBeckup text index.FF2730 F6BMI GETLINEBeckup text index.FF27AD 10 ØNEXTCHARLoop until ready.FF27AD 40 ØLDA KBDLoad character. B7 should be 'l'.FF39A 90 ØCMP #\$40Display character.FF39A 90 ØLDA #\$40 ØFor XAM mode.FF39A 90 ØLDA #\$40 ØFor XAM mode.FF4485 2B SETMORASLLeaves 57B if setting 57OR mode.FF4485 2B SETMORSTA MODE\$40 \$40 \$40 \$XAM.FF4489 ØØ Ø2CMP #\$5A"'''FF4489 ØØ Ø2CMP #\$5A"'''FF4489 ØØ Ø2CMP #\$5A"'''FF4489 ØØ Ø2CMP #\$5A"'''FF4489 ØØ Ø2CMP #\$6ACR'FF4489 ØØ Ø2CMP #\$6A''''FF44		and the second second				
FF13G0 9BCMP #\$9BESC?FF15FØ 03BEQ ESCAPEYes.FF17C8INYAdvance text index.FF1814 ØFBPL NEXTCHARAuto ESC if > 127.FF16A9 DCESCAPEJSR ECHOFF17A9 BDGETLINELDA #\$DCF716A9 BDGETLINELDA #\$SDF72126 EF FFJSR ECHOOutput it.F72246 01LDY #\$91Initialize text index.F723A9 01D0NEXTCHARLDA #SDF724A9 01NEXTCHARLDA KBD CRKey ready?F725A9 11 D0NEXTCHARLDA KBDLoad character. B7 should be '1'.F72616 FBBPL NEXTCHARLoad character.F72739 90 02STA IN. YAdd to text buffer.F73129 80 FFLDY #\$90CR?F733A9 FFLDY #\$90For XAM mode.F733A9 FFLDY #\$90For XAM mode.F74440 SETSTORASLLeaves STB if setting STOR mode.F74498 00 62NEXT ITEMLDA #\$90F744B9 00 64BLSKIPINYAdvance text index.STG STOR SAE = BLOK XAM.F744B9 00 64SETSTORF745C8BLSKIPF74670 ASETSTORF747G8BLSKIPF744B9 00 62CMP #\$80F745F0 ACMP #\$80F746F0 ASEC COLF747G8BLSKIP <td></td> <td></td> <td></td> <td>NOTCR</td> <td></td> <td></td>				NOTCR		
FF15FØ 03BEQ ESCAPEYes.FF17C6INYAdvance text index.FF1849 0CESCAPELDA #\$CCAPEAuto ESC if > 127.FF1AA9 0DESCAPELDA #\$CCAPEOutput it.FF1120 EF FFJSR ECHOOutput it.FF2120 EF FFISR ECHOOutput it.FF2230 FFLDY #\$01Initialize text index.FF2586BACKSPACEDEYBackup text index.EscapeBackup text index.FF2730 F6BMI GETLINEBeyond start of line, reinitialize.FF2730 F6BMI GETLINEBeyond start of line, reinitialize.FF2730 F6BMI GETLINEBeyond start of line, reinitialize.FF2730 F6BMI GETLINELoad character.FF2730 F7Should be 'l'.Start.FF2730 F6BDL NEXT CHARLoad character.FF3799 00 02STA N.YAdd text buffer.FF3799 00 02STA NOECRPFF3930 04FFReset text index.FF39A0 FFLDA #\$00For XAM mode.FF4185 2BSETMODESTA MODEQ0 00 stant, STB STOR Mode.FF4185 2BSETMODESTA MODEQ0 00 stant, STB STOR, SAE = BLOK XAM.FF4485 00 GZCMP #\$5DCR?FF4485 00 GZCMP #\$5A''.'?FF4489 00 02NEXT ITEMDA KIN EFF4489 00 04CMP #\$5A''.'? <td></td> <td></td> <td></td> <td></td> <td></td> <td></td>						
FF17C6INYAdvance text index.FF1816 ØFBFL NEXTCHARAuto ESC if >127.FF1620 EF FFJSR ECHOOutput it.FF17A9 BDGETLINELDA #\$DC"\".".FF17A9 BDGETLINELDA #\$ADCR.FF24AØ 01DG VIDU it.Initialize text index.FF2686BACKSPACEDEYBackup text index.FF2730 F6BMI GETLINEBeyond start of line, reinitialize.FF2730 F6BMI GETLINEBeyond start of line, reinitialize.FF2710 F76BACKSPACEDEYBackup text index.FF2710 F76BDI NEXTCHARLDA KBDLoad character.FF2710 F76BPL NEXTCHARLoad character.Stanual text index.FF3990 Ø0CMP #\$sbDCR?FF39FF39D9 Ø4BEN NOTCRNo.FF39FF39A0 FFLDY #\$pfReset text index.FF40Ø4SETSTORASLLeaves STB if setting STOR mode.FF4485 06 Ø2NEXT ITEMLDA H\$ØGR?FF4459 Øf Ø4DGPLINEYes, done this line.FF4459 Øf Ø4BEQ GETLINEYes, done this line.FF4469 Å2NEXT ITEMLDA H\$ØFF44FØ Øf Ø4BEQ GETLINEYes, est STOR, SAE = BLOK XAM.FF4469 Å2CMP #\$AE"??FF45C9 BACMP #\$AE"??FF46FØ FØBEQ GETLINEYes, on						
FF1819USABPL NEXTCHARAuto ESC if >127.FF1AA9 DCESCAPELDA #58D"\","FF1C20 EF FFJSR ECHOOutput it.F71220 EF FFJSR ECHOOutput it.F72120 EF FFLDA #58DCR.F724A9 G1LDA #58DCR.F72686BACKSPACEDEYBackup text index.F727F72730 F6BMI GETLINEBeyond start of line, reinitialize.F72730 F6BMI GETLINEBeyond start of line, reinitialize.F72730 F6CRKey ready?F726A0 ID 00NEXTCHARLoa Kab CRF727A0 ID 00NEXTCHARLoa character.F73199 60 62STA IN. YAdd to text buffer.F737C9 8DCMP #58DCR?F739D0 D4BNE NOTCRNo.F739D0 D4BNE NOTCRNo.F74185 2BSETMODESTA MODEF74464 0SETSTORASLF4445 2BSETMODESTA MODEF74452 2BSETMODESTA MODEF74459 60 60Rey #59DCR?F74453 60 60BEQ GETLINEYes, done this line.F747C9 8DCMP #58DCR?F74459 60 60CMP #58DCR?F747C9 8DCMP #58DCR?F74859 60 60CMP #58DCR?F749F9 04BEQ GETLINEYes, done this line.						
FF1AA9 DCESCAPELDA #\$DC"N".FFIC20 EF FFJSR ECHOOutput it.F71C20 EF FFJSR ECHOOutput it.F72120 EF FFJSR ECHOOutput it.F72120 EF FFJSR ECHOOutput it.F72230 F6BMI GETLINEBeyond start of line, reinitialize.F72730 F6BMI GETLINEBeyond start of line, reinitialize.F72730 F6BMI GETLINEBeyond start of line, reinitialize.F72730 F76BMI GETLINELoog until ready.F728AD 11 D0NEXTCHARLDA KBDLoog until ready.F73130 60 62STA IN. YAdd to text buffer.F73157 63 6DCMP #\$SDCR?F73157 80 0CMP #\$SDCR?F73150 40 FFLDA #\$900For XAM mode.F73153 80 FFLDA #\$900For XAM mode.F74363BLSKIPINYAdvance text index.F74452 BSETMODESTA MODE\$90 ef 42F74450 40 62NEXTIPEYes, done this line.F74459 40 40 2NEXTIPEYes, done this line.F74459 40 62CMP #\$AE"."?F74459 40 7BEQ SETMODESet BLOK XAM mode.F74376 8DCMP #\$AE"."?F74459 40 62NEXTIPEYes, can user program.F74559 40 62STX Hand H.F74559 40 62STX Hand H.F74						
FFIC20 EF FFJSR ECHOOutput it.FF1C20 EF FFJSR ECHOOutput it.FF2120 EF FFJSR ECHOOutput it.FF24AV 01LDY #\$01Initiallize text index.FF2688BACKSPACEDEYFF2680BACKSPACEDEYFF273V F6BMI GETLINEBeyond start of line, reinitialize.FF273V F6BMI GETLINEBeyond start of line, reinitialize.FF27AV 01NEXTCHARLDA KBD CRFF26AD 10 00NEXTCHARLDA KBD CRFF26AD 10 00CMP #\$450CR?FF3199 00 02STA IN, YAdd to text buffer.FF3199 00 04DA #\$50CR?FF33AV FFJSR ECHODisplay character.FF39DV D4BNE NOTCRNo.FF31SA 00LDA #\$50CR?FF33AV GFLDA #\$50CR?FF4463SETSTORASLFF4464SETSTORASLFF4463 00 02NEXT ITEMLDA IN, YFF44B9 00 02NEXT ITEMLDA IN, YFF44C9 04 02MEXT ITEMLDA IN, YFF44B9 00 02NEXT ITEMLDA IN, YFF44B9 00 02NEXT ITEMLDA IN, YFF44B9 00 02<						
FFIA9 8DGETLINELDA #\$8DCR.FF21A9 61JSR ECHOOutput it.FF24A9 61LDY #\$01Initiallize text index.FF268KBACKSPACEDEYBack up text index.FF2730 F6BMI GETLINEBeyon distart of line, reinitialize.FF2730 F6BMI GETLINEBeyon distart of line, reinitialize.FF2730 F6DYDEXTCHARLDA KBDFF2719 FBBPL NEXTCHARLood character. B7 should be '1'.FF3199 69 02STA IN, YAdd to text buffer.FF3199 69 02STA IN, YAdd to text buffer.FF33D19 D9BNE NOTCRNo.FF39D4BNE NOTCRNo.FF39A9 69LDA #\$00For XAM mode.FF4163 2BSETMODESTA MODE\$00 = XAM, \$7B = STOR, SAE = BLOK XAM.FF4464SETSTORASLLeaves \$7B if setting STOR mode.FF4452 BSETMODESTA MODE\$00 = XAM, \$7B = STOR, SAE = BLOK XAM.FF4459 69 69 02CMP #\$8DCR?FF4459 40 69 02CMP #\$8DCR?FF4459 80 69 02CMP #\$8DCR?FF4559 80 69 02CMP #\$8DCR?FF4659 81BEQ ESTINODESet BLOCK XAM mode.FF4779 94BCC BLSKIPSkip delimiter.FF4859 84CMP #\$8D'''''FF4959 85BEQ RUNYee, set STOR mode.FF5759 8				ESCAPE		
FF122424EFFJSR ECHOOutput it.FF24A4 40 LD ¥\$01Initiallize text index.FF26A5BACKSPACEDEYBack up text index.FF2734F6BMI GETLINEBeyond start of line, reinitialize.FF27A5D11 D0NEXTCHARLDA KBD CRKey ready?FF26A5D19 D0LDA KBD CRLoa dcharacter.FF27A5J09 00LDA KBDLoa dcharacter.FF3197 00 02STA IN. YAdd text buffer.FF33C6 85CRP %85DCR?FF39D0 04BNE NOTCRNo.FF39A9 00LDA K\$00For XAM mode.FF31A9 00LDA \$\$00For XAM mode.FF4496 02NEXT ITEMLDA \$\$00For XAM mode.FF448528SETMODESTA MODE\$ 00 = xAm, STA = STOR, SAE = BLOK XAM.FF4489 00 02NEXT ITEMLDA IN. YGet character.FF4489 00 02NEXT ITEMLDA IN. YGet character.FF4590 00 02NEXT ITEMLDA IN. YGet character.						
F724AØLDY $\#$ ØIInitialize text index.F72586BACKSPACEDEYBackup text index.F72730 F6BMI GETLINEBeyond start of line, reinitialize.F72710 FBDNEXTCHARLDA KBD CRKey ready?F72710 FBBPL NEXTCHARLoou nuil ready.F72710 FBBPL NEXTCHARLoou nuil ready.F72710 FBSTA IN. YAdd to text buffer.F73190 G0STA IN. YAdd to text buffer.F737C9 8DCMP #\$8DCR?F739D0 D4BNE NOTCRNo.F739D4 04BNE NOTCRNo.F739D4 04EDA #\$00For XAM mode.F741D5 2BSETSTORASLLeaves 57B if setting STOR mode.F743C8BLSKIPINYAdvance text index.F744B9 00 02CMP #\$3DCR?F744D9 04BEQ GETLINEYes, done this line.F747C9 8DCMP #\$3DCR?F748P6 04BEQ GETLINEYes, ets STOR mode.F749P6 05BEQ SETTORSet bLOCK XAM mode.F749P6 04BEQ SETTINEYes, set STOR mode.F741D9 64 02CMP #\$5AE""F741B9 60 02CMP #\$5AE""F742P6 04BEQ SETINOPESet bLOCK XAM mode.F743C8CBSEXTSet of one canceler.F744P6 04CBLSKIPSkip delimiter.F745P				GETLINE		
FF2688BACKSPACEDEYBack up text index.FF2730F6BMI GETLINEBeyond start of line, reinitialize.FF27AD 11 D0NEXTCHARLDA KBD CRKey ready?FF27AD 10D0LDA KBDLoa Character. B7 should be '1'.FF31999002STA IN, YAdd to text buffer.FF31999002STA IN, YAdd to text buffer.FF31G9 8DCMP #\$8DCR?FF33A0FFLDY #\$FFReset text index.FF34A0FFLDY #\$FFReset text index.FF35A4FAFA 4500 FF449002NEXT ITEMLDA #\$90FF4480SETSTORASLLeaves \$7B if setting STOR mode.FF44852BSETMODESTA MODE\$90 = XAM, \$7B = STOR, \$AE = BLOK XAM.FF4368BLSKIPINYAdvance text index.FF448990 90 92NEXT ITEMLDA #\$20CR?FF449190 94BEQ GETLINEYes, done this line.FF4490 94BEQ GETLINEYes, done this line.FF4590 94BEQ SETSTORYes, set STOR mode.FF4690 46CMP #\$BA""?FF53F46BEQSETSTORFF55F46BEQSETSTORFF55F47BEQSETSTORFF5590 90CMP #\$20FF5694 90CMP #\$20FF57 </td <td></td> <td></td> <td></td> <td></td> <td></td> <td></td>						
FF2730 F6BMI GETLINEBeyond start of line, reinitialize.FF26AD 11 D0NEXTCHARLDA KBD CRKey ready?FF26I0 FBBPL NEXTCHARLoad character. B7 should be '1'.FF3190 002STA IN. YAdd to text buffr.Add to text buffr.FF3390 00CMP #\$8DCR?FF37D9 00DCMP #\$8DFF37D9 00DCMP #\$8DFF37D9 00DCMP #\$6DFF38A0 FFLDA #\$00For XAM mode.FF37A3 SETSTORASLLeaves \$7B if setting STOR mode.FF44B5 2BSETMODESTA MODE\$00 extra XAM.FF4485 2BSETMODESTA MODE\$00 extra XAM.FF4486 00 extra XAM.YB = STOR, \$AE = BLOK XAM.FF44S0 00 extra XAMCR?FF44S0 00 extra XAMSETSTOR mode.FF47C9 8DCMP #\$8DCR2P\$4BCQ GETLINEYe4Ye5Sto denthis line.FF45S0 94 f4BCC BLSKIPSkip delimiter.Skip delimiter.FF45S74BBAFF55S10 EST STORYe5Ye5, run user program.FF57F0 3BBEQ SETSTORYe5Ye5FF5886 28STX Hand H.FF5986 28STX Hand H.FF5986 28STX Hand H.FF5880 40FF6869 88 <td></td> <td></td> <td></td> <td></td> <td></td> <td></td>						
FF29AD 11 DØNEXTGHARLDA KBD CRKey ready?FF2C1Ø DØLDA KBDLoop until ready.FF3199 ØØ 02STA IN, YAdd to text buffer.FF3199 ØØ 02STA IN, YAdd to text buffer.FF3199 ØØ 02STA IN, YAdd to text buffer.FF3190 ØØ 12STA IN, YAdd to text buffer.FF3190 ØØ 12STA IN, YAdd to text buffer.FF31AØ FFJSR ECHODisplay character.FF33AØ FFLDY %SFReset text index.FF34AØ FFLDY %FFReset text index.FF35AATAXØ+X.FF44ØASETSTORASLLaves \$78 if setting STOR mode.FF44FF4482NEXT ITEMLDA #\$\$ØGP 2NACE text index.FF4489 ØØ Ø2NEXT ITEMLDA IN, YGet character.FF4489 ØØ Ø2NEXT ITEMLDA IN, YGet character.FF4779 8DCRP #\$80DCR?FF4869 AEC9 AECMP #\$8ADC72RR''?FF4990 F4BCC BLSKIPSkip delimiter.FF49FF49FF4090 F4BCC SETSTORYes, set STOR mode.FF53FØ EBEQ SETMODESet BLOCK XAM mode.FF53FØ EBBEQ RUNYes, run user program.FF55FF5862 9STX Hand H.FF59				BACKSPACE		
FF2C10BPLNEXTCHARLoop unit ready.FF2EAD 10D0LDA KBDLoad character. B7 should be '1'.FF31990002STA IN, YAdd to text buffer.FF3320EF FFJSR ECHODisplay character.FF37C9 8DCMP #88DCR?FF39D0D4BNE NOTCRNo.FF39D4LDA #\$00For XAM mode.FF37ATAX $G \rightarrow X$.FF4004SETSTORASLLeaves \$7B if setting STOR mode.FF4185 2BSETMODESTA MODE\$00 = x AM, \$7B = STOR, \$AE = BLOK XAM.FF4485 2BNEXT ITEMLDA IN, YGet character.FF4490 00 02NEXT ITEMLDA IN, YGet character.FF47C9 8DCMP #\$8DCR?FF48C9 AECMP #\$AE""?FF4990 00 F4BEQ GETLINEYes, done this line.FF45C9 AECMP #\$AE""?FF47C9 BACMP #\$AE""?FF48F0 F0BEQ SETMODESet BLOCK XAM mode.FF51C9 BACMP #\$BA""?FF4550 22CMP #\$BA""?FF5560 29STX Hand H.FF5886 29STX Hand H.FF5984 2ASTY YSAVSave Y for comparison.FF5886 29STX Hand H.FF5984 2ASTY YSAVSave Y for comparison.FF5886 29 <td></td> <td></td> <td></td> <td></td> <td></td> <td></td>						
F72EAD 10LDA KBDLoad character. B7 should be 'l'.F73199 00 02STA IN, YAdd to text buffer.F73120 EF FFJSR ECHODisplay character.F737C9 8DCMP #\$8DCR?F738D0 D4BNE NOTCRNo.F739D0 D4ENE NOTCRNo.F737F739D0 D4ENE NOTCRNo.F738A0 FFLDA #\$00For XAM mode.F739A0 00LDA #\$00For XAM mode.F741A5 2BSETMODESTA MODE\$00 00 + xx.FF42G8BLSKIPINYAdvance text index.FF43C8BLSKIPINYAdvance text index.FF44B9 00 02NEXT ITEMLDA IN, YGet character.FF47C9 8DCMP #\$8DCR?FF48C9 AECMP #\$8A""?FF49F0 D4BEQ GETLINEYes, done this line.FF45F6 9BBEQ SETMODESet BLOCK XAM mode.FF55C9 D2CMP #\$D2"R"?FF55G9 D2CMP #\$D2"R"?FF5866 29STX Hand H.FF5986 29STX Hand H.FF5084 2ASTY YSAVSave Y for comparison.FF5669 04CMP #\$90Map digits to \$0-9.FF57F64 99 04EOR #\$B0Map digits to \$0-9.FF5886 29STX Hand H.FF5989 00 02NEXTHEXLDA IN, YGet character for hex tes				NEXTCHAR		
FF3199 60 62STA IN, YAdd to text buffer.FF3420 EF FFJSR ECHODisplay character.FF37C9 8DCMP #\$8DGR?FF39D0 D4BNE NOTCRNo.FF38A0 FFLDY #\$FFReset text index.FF30A9 60LDA #\$60For XAM mode.FF41ATAX $\oint \star X$.FF446ASETSTORASLLeaves \$7B if setting STOR mode.\$00 ± XAM, \$7B = STOR, \$AE = BLOK XAM.FF4485 2BSETMODESTA MODE\$00 40 ± XAM, \$7B = STOR, \$AE = BLOK XAM.FF43FF44B9 60 62NEXT ITEMLDA IN, YGet character.FF47C9 8DCR?FF47C9 8DCR?FF48C9 AECMP #\$8DCR?FF49FF49F0 D4BEQ GETLINEYes, done this line."'.'?FF47F9 60 62NEXT ITEMLDA M\$80CMP #\$8DCR?FF49F0 D4BEQ GETLINEYes, done this line."'.'?FF45F9 60 F4BCC BLSKIPSkip delimiter.Stip delimiter.FF51C9 D2CMP #\$50CMP #\$50Yes, van user program.FF55F9 G2CMP #\$50FF55F9 62STX HA0 40CMP #\$50Map digits to \$0 -9.FF55B9 60 62STX HFF55B9 60 62STX HFF6696 64BCC DIGYes, VANSave Y for comparison. <td></td> <td></td> <td></td> <td></td> <td></td> <td></td>						
FF3420 EF FFJSR ECHODisplay character.FF37C9 8DCMP #\$8DCR?FF39D0 D4EN EN OTCRNo.FF31A0 FFLDY #\$FFReset text index.FF35A0 00 LDA #\$00For XAM mode.FF37AATAX $0 \rightarrow x$.FF40ØASETSTORASLLeaves 57B if setting STOR mode.For XAM mode.FF4185 2BSETMODESTA MODEFF43C8BLSKIPINYAdvance text index.FF44FF47C9 8DCMP #\$8DCR?FF47F0 90D4BEQ GETLINEFF49F0 D4BEQ GETLINEYes, done this line.FF49F0 D4BEQ GETLINEYes, done this line.FF41C9 8DCMP #\$8A"."."?FF45F0 F0BEQ SETSTORYes, set STOR mode.FF51C9 BACMP #\$8D".".??FF55F0 EBEEQ SETSTORYes, run user program.FF5986 28STX L\$00+L.FF5886 29STX Hand H.FF57F9 40EOC #\$80Map digits to \$0-9.FF5669 48ADC #\$64Digit?FF6469 88ADC #\$88Map digits to \$0-9.FF6469 88ADC #\$88Map digits to \$0-9.FF6469 88ADC #\$86Map digits to \$0-9.FF6469 88ADC #\$86Map digits to \$0-9.FF6469 88ADC #\$88Map leiter "A"="F" to \$FA-FF						
FF37C9 & BDCMP #\$ & BDCR?FF39DØ D4BNE NOTCRNo.FF38AØ ØFLDX #\$\$ FFReset text index.FF30A9 ØØLDA #\$ ØØFor XAM mode.FF37AATAXØ+X.FF40ØASETSTORASLLeaves \$7B if setting STOR mode.\$ ØØ = XAM, \$7B = STOR, \$AE = BLOK XAM.FF4185 2BSETMODESTA MODE\$ ØØ ØØ 2NEXT ITEMLDA IN, YAdvance text index.Get character.FF44B9 ØØ ØZNEXT ITEMLDA IN, YGet character.FF47C9 & BDCMP #\$ & BCQ GETLINEYes, done this line."."?FF45F9 ØØ ØABEQ GETLINEYes, done this line."."??FF45F9 ØØF4BCQ SETMODESet BLOCK XAM mode.FF51C9 BACMP #\$ BAFF55C9 D2CMP #\$ D2CMP #\$ D2"R"?FF55F0 2BBEQ SETSTORYes, run user program.FF55S9 00 02CMP #\$ D4FF5886 28STX Hand H.FF55B9 00 02NEXTHEXLDA IN, YGet character for hex test.FF5689 04C9 04CMP #\$ ØADigit?FF6690 66BCC DIGYes.FF6690 66BCC DIGYes.FF6690 66BCC NOTHEXNo, character not hex.FF6690 6FF67 </td <td></td> <td></td> <td></td> <td></td> <td></td> <td></td>						
FF39DØ D4BNE NOTCRNo.FF33AØ FFLDY #\$FFReset text index.FF36AATAX $\oint \times X$.FF40Ø ASETSTORASLLeaves \$7B if setting STOR mode.FF41B5 2BSETMODESTA MODE\$ØØ = XAM, \$7B = STOR, \$AE = BLOK XAM.FF43C8BLSKIPINYAdvance text index.FF44B5 0Ø 0Ø 2NEXT ITEMLDA IN, YGet character.FF47C9 8DCMP #\$8DCR?FF49FØ 4BEQ GETLINEYes, done this line.FF49FØ 4BCC BLSKIPSkip delimiter.FF47FØ FØBEQ SETMODESet BLOCK XAM mode.FF53FØ EBBEQ SETSTORYes, set STOR mode.FF55C9 D2CMP #\$D2"I"?FF57FØ 3BBEQ RUNYes, run user program.FF5886 28STX L\$ØØ L.FF5986 28STX L\$ØØ L.FF55B9 0Ø 02NEXTHEXLDA IN, YGet character for hex test.FF62FF64C9 0ACMP #\$ØADigit?FF64S0 88ADMap digits to \$0-9.FF64C9 FACMP #\$A8FF64C9 FACMP #\$A4FF64C9 FACMP #\$A4FF64S0 88ADE #\$A8ADMap digits to \$0-9.FF64C9 FACMP #\$A4FF64S0 88ADE #\$A8FF64S0 88ADE #\$A58FF66S0 88ADE #\$A58 </td <td></td> <td></td> <td></td> <td></td> <td></td> <td></td>						
FF3BAØFFLDY #\$FFReset text index.FF3DA9ØØLDA #\$ØØFor XAM mode.FF3FAATAX $\oint XA$ FF4ØØASETSTORASLLeaves \$7B if setting STOR mode.FF4185 2BSETMODESTA MODE\$ØØ = XAM, \$7B = STOR, \$AE = BLOK XAM.FF43C8BLSKIPINYAdvance text index.FF44B9ØØ Ø2NEXT ITEMLDA IN, YGet character.FF47C98DCMP #\$8DCR?FF48C9AE"."?FF49PØBEQ GETLINEYes, done this line.FF48C9AE"."?FF49FØBEQ SETMODESte BLOCK XAM mode.FF51C9BACMP #\$BA"."?FF55FØ EBBEQ SETSTORYes, set STOR mode.FF55S70EBBEQ RUNYes, run user program.FF55S6 28STX L\$Ø0+L.FF55S9ØØG2FF6249BGECC M#\$S0FF64C9GACMP #\$ØAFF65B9ØØGet character for hex test.FF56S9ØØGD #\$S8Map digits to \$0-9.FF64C9GACMP #\$ØADigit?FF669ØG6BCC DIGYes.FF6690G6BCC DIGYes.FF6690G6BCC DIGYes.FF6690GAASLFF67ØAD						
FF3DA9 00 LDA #\$00For XAM mode.FF3FAATAX $0 \rightarrow X$.FF40ØASETSTORASLLeaves \$7B if setting STOR mode.FF4185 2BSETMODESTA MODE\$00 = XAM, \$7B = STOR, \$AE = BLOK XAM.FF43C8BLSKIPINYAdvance text index.FF44B9Ø0 Ø0ØEXT ITEMLDA IN, YGet character.FF47C98DCMP #\$8DCR?FF48C9AECMP #\$AE"."?FF49FØ Ø0BEQ GETLINEYes, done this line.FF45F9FØBEQ GETMODESet BLOCK XAM mode.FF45F9FØBEQ SETSTORYes, set STOR mode.FF55F9 BABEQ SETSTORYes, run user program.FF55C9 D2CMP #\$D2"R"?FF55F57FØBEQ RUNYes, run user program.FF5886 29STX Hand H.FF5984 2ASTY YSAVSave Y for comparison.FF56B9Ø0EOR #\$D0Map digits to \$0 - 9.FF64C9 ØACMP #\$GADigit?FF64C9 FACMP #\$S8Map letter "A"-"F" to \$FA-FF.FF6669 88ADC #\$S8Map letter ?FF6669 88ADC #\$S8Map letter ?FF66Ø11BCC NOTHEXNo, character not hex.FF66Ø12ASLHex digit to MSD of A.FF70ØAASLHex digit to MSD of A.FF71Ø4A						
FF3FAATAX $\not (\rightarrow x.)$ FF40 $\emptyset A$ SETSTORASLLeaves \$7B if setting STOR mode.FF4185 2BSETMODESTA MODE\$ $\emptyset \emptyset = xAM$, \$7B = STOR, \$AE = BLOK XAM.FF43C8BLSKIPINYAdvance text index.FF44B9 $\emptyset \emptyset \emptyset \emptyset 2$ NEXT ITEMLDA IN, YGet character.FF47C9 8DCMP #\$8DCR?FF48C9 AECMP #\$AE"."?FF49P\ F4BCC BLSKIPSkip delimiter.FF45F\ Ø F\ ØBEQ SETMODESet BLOCK XAM mode.FF51C9 BACMP #\$BA"!"?FF53F\ Ø EBBEQ SETSTORYes, set STOR mode.FF55G9 D2CMP #\$D2"R"?FF5886 28STX L\$ $\emptyset \emptyset \rightarrow L$.FF5886 29STX Hand H.FF57B9 $\emptyset \emptyset \emptyset \emptyset 2$ NEXTHEXLDA IN, YGet character for hex test.FF6249 B\ ØFF64C9 $\emptyset A$ CMP #\$\$AFF64G9 $\emptyset \emptyset \emptyset$ BCC DIGFF6669 88ADC #\$88Map leiter "A"-"F" to \$FA-FF.FF66 $\emptyset A$ DIGFF67 $\emptyset A$ ASLFF67 $\emptyset A$ ASLFF67 $\emptyset A$ ASLFF71 $\emptyset A$ ASLFF71 $\emptyset A$ ASLFF72A2 $\emptyset 4$ LDX #\$ $\emptyset 4$ Shift count.						
FF40ØASETSTORASLLeaves \$7B if setting STOR mode.FF4185 2BSETMODESTA MODE\$00 \$00 \$00 \$00 \$00 \$00 \$00 \$00 \$00 \$00						
FF4185 2BSETMODESTA MODE $$ \emptyset \emptyset = xAM, \$7B = sTOR, \$AE = BLOK XAM.$ FF43C8BLSKIPINYAdvance text index.FF43C8BLSKIPINYAdvance text index.FF44B9 $\emptyset \emptyset \emptyset \emptyset \emptyset D$ NEXT ITEMLDA IN, YGet character.FF47C9 8DCMP #\$sDCR?FF49FØ D4BEQ GETLINEYes, done this line.FF45FØ AECMP #\$AE"."?FF40PØ F4BCC BLSKIPSkip delimiter.FF47FØ FØBEQ SETMODESet BLOCK XAM mode.FF51C9 BACMP #\$BA":"?FF55C9 D2CMP #\$D2"R"?FF55FF55G2 8STX L\$ØØ+ L.FF5986 28STX L\$ØØ+ L.FF55B9 ØØ Ø2NEXTHEXLDA IN, YGet character for hex test.FF55Sf Ø # DAFF55B9 ØØ Ø2NEXTHEXLDA IN, YGet character for hex test.FF6249 BØFF64C9 ØACMP #\$D4Digit?FF6690 Ø6ECC DIGYes.FF6690 Ø1BCC NOTHEXNo, character not hex.FF66Ø4DIGASLFF71ØAASLFF71ØAASLFF71ØAASLFF72A2 Ø4LDX #\$Ø4				SETSTOR		
FF43C8BLSKIPINYAdvance text index.FF44B9 ØØ Ø2NEXT ITEMLDA IN, YGet character.FF47C9 8DCMP #\$8DCR?FF49FØ D4BEQ GETLINEYes, done this line.FF48C9 AECMP #\$AE"."?FF4090 F4BCC BLSKIPSkip delimiter.FF45C9 BACMP #\$BA":"?FF55C9 BACMP #\$BA":"?FF55C9 D2CMP #\$D2"R"?FF55C9 D2CMP #\$D2"R"?FF55F0 3BBEQ RUNYes, run user program.FF5986 28STX L\$00 + L.FF55B9 ØØ Ø2NEXTHEXLDA IN, YGet character for hex test.FF6249 BØFF64C9 ØACMP #\$\$ØMap digits to \$\$0 -9.FF64C9 ØACMP #\$\$ADigit?FF6690 Ø6BCC DIGYes.FF6690 Ø6BCC DIGYes.FF6690 Ø11BCC NOTHEXNo, character not hex.FF66ØADIGASLFF70 ØAASLHex digit to MSD of A.FF71 ØAASLFF71FF72A2 Ø4LDX #\$Ø4Shift count.	_	-				
FF44B9 ØØ Ø2NEXT ITEMLDA IN, YGet character.FF47C9 8DCMP #\$8DCR?FF48FØ D4BEQ GETLINEYes, done this line.FF48G9 AECMP #\$AE"."?FF4090 F4BCC BLSKIPSkip delimiter.FF47FØ FØBEQ SETMODESet BLOCK XAM mode.FF51C9 BACMP #\$BA"."?FF55C9 D2CMP #\$D2"R"?FF55FØ 3BBEQ SETSTORYes, set STOR mode.FF55FØ 3BBEQ RUNYes, run user program.FF55F9 36 28STX L\$00 + L.FF55S9 00 02NEXTHEXLDA IN, YGet character for hex test.STY YSAVSave Y for comparison.FF55B9 00 02NEXTHEXLDA IN, YGet character for hex test.FF6249 B0FF64C90 4ACMP #\$D4Digit?FF6690 06FF6690 06BCC DIGYes.FF76FF66G9 FACMP #\$FAFF66G9 I1BCC NOTHEXFF66G4ASLFF67GAASLFF70GAASLFF71GAASLFF72A2 04LDX #\$04Shift count.						
FF47C9 8DCMP #\$8DCR?FF49FØ D4BEQ GETLINEYes, done this line.FF409Ø F4BCC BLSKIPSkip delimiter.FF4FFØ FØBEQ SETMODESet BLOCK XAM mode.FF51C9 BACMP #\$BA":"?FF53FØ EBBEQ SETSTORYes, set STOR mode.FF55C9 D2CMP #\$D2"R"?FF57FØ 3BBEQ RUNYes, run user program.FF5886 28STX L\$ØØ L.FF5986 28STX L\$ØØ L.FF55B9 ØØ Ø2NEXTHEXLDA IN, YGet character for hex test.FF6249 BØFF64C9 ØACMP #\$ØADigit?FF669Ø Ø6BCC DIGYes.FF66S8ADC #\$88Map digits to \$Ø -9.FF6690 11BCC NOTHEXNo, character not hex.FF6EØADIGASLFF77ØAASLFF71ØAASLFF72A2 Ø4LDX #\$Ø4Shift count.						
FF49FØ D4BEQ GETLINEYes, done this line.FF4BC9 AECMP #\$AE"."?FF4D90 F4BCC BLSKIPSkip delimiter.FF4FFØBEQ SETMODESet BLOCK XAM mode.FF51C9 BACMP #\$BA":"?FF53FØ EBBEQ SETSTORYes, set STOR mode.FF55C9 D2CMP #\$D2"R"?FF57FØ 3BBEQ RUNYes, run user program.FF5886 28STX L\$00 L.FF5986 28STX Hand H.FF55B9 ØØ Ø2NEXTHEXLDA IN, YFF5689 ØØGCMP #\$BØMap digits to \$Ø -9.FF6249 BØEOR #\$BØMap digit?FF6669 88ADC #\$88Map letter "A"-"F" to \$FA-FF.FF6669 88ADC #\$88Map letter?FF6696 11BCC NOTHEXNo, character not hex.FF66ØADIGASLFF71ØAASLHex digit to MSD of A.FF72A2 Ø4LDX #\$Ø4Shift count.						
FF4BC9 AECMP #\$AE"."?FF4D90 F4BCC BLSKIPSkip delimiter.FF4F $F0$ $f0$ BEQ SETMODESet BLOCK XAM mode.FF51C9 BACMP #\$BA"."?FF53 $F0$ EBBEQ SETSTORYes, set STOR mode.FF55C9 D2CMP #\$D2"R"?FF57 $F0$ 3BBEQ RUNYes, run user program.FF5986 28STX L\$0\$ 0 0$ 0$ 0$ 2$FF5584 2ASTY YSAVSave Y for comparison.FF5684 2ASTY YSAVSave Y for comparison.FF57F0 60BCCMP #$0AFF64C9 0 ACMP #$0ADigit?FF6690 06BCC DIGYes.FF6690 11BCC NOTHEXNo, character not hex.FF6690 11BCC NOTHEXNo, character not hex.FF6690 11BCC NOTHEXNo, character not hex.FF6670 AASLHex digit to MSD of A.FF700 AASLFF71FF710 AASLFF72A2 04LDX #$04Shift count.$						
FF4D9ØF4BCC BLSKIPSkip delimiter.FF4FFØFØBEQ SETMODESet BLOCK XAM mode.FF51C9 BACMP #\$BA":"?FF53FØ EBBEQ SETSTORYes, set STOR mode.FF55C9 D2CMP #\$D2"R"?FF57FØ 3BBEQ RUNYes, run user program.FF5986 28STX L\$ØØ \rightarrow L.FF5184 2ASTY YSAVSave Y for comparison.FF55B9 ØØ Ø2NEXTHEXLDA IN, YGet character for hex test.FF5649 ØØEOR #\$BØMap digits to \$Ø -9.FF6690 Ø6BCC DIGYes.FF6869 88ADC #\$88Map letter "A"-"F" to \$FA-FF.FF6690 11BCC NOTHEXNo, character not hex.FF67ØADIGASLFF70ØAASLHex digit to MSD of A.FF71ØAASLLDX #\$Ø4Shift count.			C9 AE			
FF4FFØBEQ SETMODESet BLOCK XAM mode.FF51C9 BACMP #\$BA":"?FF53FØ EBBEQ SETSTORYes, set STOR mode.FF55C9 D2CMP #\$D2"R"?FF57FØ 3BBEQ RUNYes, run user program.FF57S6 28STX L\$ØØ+L.FF5886 29STX Hand H.FF55B9 ØØ Ø2NEXTHEXLDA IN, YGet character for hex test.FF6249 BØFF64C9 ØACMP #\$ØAFF64C9 ØACMP #\$ØAFF6690 Ø6BCC DIGFF6669 88ADC #\$88Map letter "A"-"F" to \$FA-FF.FF67ØADIGFF66ØADIGFF66ØAASLFF71ØAASLFF72A2 Ø4LDX #\$Ø4Shift count.Shift count.		FF4D	9Ø F4			Skip delimiter.
FF51C9 BACMP #\$BA":"?FF53FØ EBBEQ SETSTORYes, set STOR mode.FF55C9 D2CMP #\$D2"R"?FF57FØ 3BBEQ RUNYes, run user program.FF5786 28STX L\$ØØ+L.FF5B86 29STX Hand H.FF5D84 2ASTY YSAVSave Y for comparison.FF5FB9 ØØ Ø2NEXTHEXLDA IN, YGet character for hex test.FF6249 BØEOR #\$BØMap digits to \$Ø-9.FF64C9 ØACMP #\$ØAFF6869 88ADC #\$88Map letter "A"-"F" to \$FA-FF.FF6AC9 FACMP #\$FAFF6EØADIGFF6FØADIGFF6FØAASLFF70ØAASLFF71ØAASLFF72A2 Ø4LDX #\$Ø4Shift count.		FF4F	FØFØ		BEQ SETMODE	
FF55C9 D2CMP $\#$ D2"R"?FF57FØ 3BBEQ RUNYes, run user program.FF5986 28STX L\$Ø\$ D2FF5B86 29STX Hand H.FF5D84 2ASTY YSAVSave Y for comparison.FF5FB9 ØØ Ø2NEXTHEXLDA IN, YGet character for hex test.FF6249 BØEOR $\#$ \$BØMap digits to \$Ø - 9.FF64C9 ØACMP $\#$ \$ØADigit?FF669Ø Ø6BCC DIGYes.FF6798ADC $\#$ \$88Map letter "A"-"F" to \$FA-FF.FF64C9 FACMP $\#$ \$FAHex letter?FF6590 11BCC NOTHEXNo, character not hex.FF66ØADIGASLFF67ØAASLHex digit to MSD of A.FF70ØAASLFF71FF72A2 Ø4LDX $\#$ \$Ø4Shift count.		FF51	C9 BA		CMP #\$BA	":"?
FF55C9 D2CMP $\#$ \$D2"R"?FF57FØ 3BBEQ RUNYes, run user program.FF5986 28STX L\$ØØ \rightarrow L.FF5B86 29STX Hand H.FF5D84 2ASTY YSAVSave Y for comparison.FF5FB9 ØØ Ø2NEXTHEXLDA IN, YGet character for hex test.FF6249 BØEOR $\#$ \$BØMap digits to \$Ø -9.FF64C9 ØACMP $\#$ \$ØADigit?FF669Ø Ø6BCC DIGYes.FF6869 88ADC $\#$ \$88Map letter "A"-"F" to \$FA-FF.FF64C9 FACMP $\#$ \$FAHex letter?FF6290 11BCC NOTHEXNo, character not hex.FF65ØADIGASLFF67ØAASLHex digit to MSD of A.FF70ØAASLFF71FF72A2 Ø4LDX $\#$ \$Ø4Shift count.		FF53	FØEB		BEQ SETSTOR	Yes, set STOR mode.
FF5986 28STX L $\$ \emptyset \emptyset \rightarrow L$.FF5B86 29STX Hand H.FF5D84 2ASTY YSAVSave Y for comparison.FF5FB9 $\emptyset \emptyset \emptyset 2$ NEXTHEXLDA IN, YGet character for hex test.FF6249 BØEOR #\$BØMap digits to \$Ø - 9.FF64C9ØACMP #\$ØADigit?FF669ØØ6BCC DIGYes.FF6869 88ADC #\$88Map letter "A"-"F" to \$FA-FF.FF6AC9 FACMP #\$FAHex letter?FF6C9Ø11BCC NOTHEXNo, character not hex.FF6EØADIGASLFF76FFF70ØAASLHex digit to MSD of A.FF71ØAASLShift count.		FF55	C9 D2		CMP #\$D2	"R"?
FF5B86 29STX Hand H.FF5D84 2ASTY YSAVSave Y for comparison.FF5D84 2ASTY YSAVSave Y for comparison.FF5FB9 ØØ Ø2NEXTHEXLDA IN, YGet character for hex test.FF6249 BØEOR #\$BØMap digits to \$Ø -9.FF64C9 ØACMP #\$ØADigit?FF6690 Ø6BCC DIGYes.FF6869 88ADC #\$88Map letter "A"-"F" to \$FA-FF.FF6AC9 FACMP #\$FAHex letter?FF6C90 11BCC NOTHEXNo, character not hex.FF6EØADIGASLFF76ØAASLHex digit to MSD of A.FF71ØAASLShift count.		FF57	FØ 3B		BEQ RUN	Yes, run user program.
FF5D84 2ASTY YSAVSave Y for comparison.FF5FB9 ØØ Ø2NEXTHEXLDA IN, YGet character for hex test.FF6249 BØEOR #\$BØMap digits to \$Ø -9.FF64C9 ØACMP #\$ØADigit?FF669Ø Ø6BCC DIGYes.FF6869 88ADC #\$88Map letter "A"-"F" to \$FA-FF.FF6AC9 FACMP #\$FAHex letter?FF6C9Ø 11BCC NOTHEXNo, character not hex.FF6EØADIGASLFF6FØAASLHex digit to MSD of A.FF70ØAASLFr71FF71ØAASLFF72A2 Ø4LDX #\$Ø4		FF59	86 28		STX L	\$ØØ→ L.
FF5FB9 ØØ Ø2NEXTHEXLDA IN, YGet character for hex test.FF6249 BØEOR #\$BØMap digits to \$Ø-9.FF64C9 ØACMP #\$ØADigit?FF669Ø Ø6BCC DIGYes.FF6869 88ADC #\$88Map letter "A"-"F" to \$FA-FF.FF6AC9 FACMP #\$FAHex letter?FF6C9Ø 11BCC NOTHEXNo, character not hex.FF6EØADIGASLFF7ØØAASLHex digit to MSD of A.FF71ØAASLFF72FF72A2 Ø4LDX #\$Ø4Shift count.					STX H	and H.
FF6249 BØEOR #\$BØMap digits to \$Ø-9.FF64C9 ØACMP #\$ØADigit?FF669Ø Ø6BCC DIGYes.FF6869 88ADC #\$88Map letter "A"-"F" to \$FA-FF.FF6AC9 FACMP #\$FAHex letter?FF6C9Ø 11BCC NOTHEXNo, character not hex.FF6EØADIGASLFF6FØAASLHex digit to MSD of A.FF7ØØAASLFF71FF71ØAASLFF72A2 Ø4LDX #\$Ø4					STY YSAV	Save Y for comparison.
FF64C9 ØACMP $\#$ \$ØADigit?FF669Ø Ø6BCC DIGYes.FF6869 88ADC $\#$ \$88Map letter "A"-"F" to \$FA-FF.FF6AC9 FACMP $\#$ \$FAHex letter?FF6C9Ø 11BCC NOTHEXNo, character not hex.FF6EØADIGASLFF7ØØAASLHex digit to MSD of A.FF71ØAASLFF72A2 Ø4LDX $\#$ \$Ø4Shift count.				NEXTHEX		Get character for hex test.
FF6690 06BCC DIGYes.FF6869 88ADC #\$88Map letter "A"-"F" to \$FA-FF.FF6AC9 FACMP #\$FAHex letter?FF6C90 11BCC NOTHEXNo, character not hex.FF6EØADIGASLFF6FØAASLHex digit to MSD of A.FF70ØAASLFF71ØAASLFF72A2 04LDX #\$04						Map digits to \$\$ -9.
FF 6869 88ADC #\$88Map letter "A"-"F" to \$FA-FF.FF 6AC9 FACMP #\$FAHex letter?FF 6C9Ø 11BCC NOTHEXNo, character not hex.FF 6EØADIGASLFF 6FØAASLHex digit to MSD of A.FF 7ØØAASLFF 71ØAASLFF 72A2 Ø4LDX #\$Ø4						Digit?
FF6AC9 FACMP #\$FAHex letter?FF6C9Ø 11BCC NOTHEXNo, character not hex.FF6EØADIGASLFF6FØAASLHex digit to MSD of A.FF7ØØAASLFF71ØAASLFF72A2 Ø4LDX #\$Ø4						
FF6C9Ø 11BCC NOTHEXNo, character not hex.FF6EØADIGASLFF6FØAASLHex digit to MSD of A.FF7ØØAASLFF71ØAASLFF72A2 Ø4LDX #\$Ø4						
FF6EØADIGASLFF6FØAASLHex digit to MSD of A.FF7ØØAASLFF71ØAASLFF72A2 Ø4LDX #\$Ø4Shift count.						
FF6FØAASLHex digit to MSD of A.FF7ØØAASLFF71ØAASLFF72A2 Ø4LDX #\$Ø4Shift count.				DIC		No, character not hex.
FF70ØAASLFF71ØAASLFF72A2Ø4LDX #\$Ø4Shift count.				DIG		
FF71 ØA ASL FF72 A2 Ø4 LDX #\$Ø4 Shift count.						Hex digit to MSD of A.
EFF72 A2 Ø4 LDX #\$Ø4 Shift count.						
						Chift count
nex digit leit, Mob to carry.	-			HEXSHIFT		
		1111	V 11	IIEAOHIF I	ADD ·	nex digit lett, Mob to carry.

6502 HEX MONITOR LISTING (continued)

FF7526.28ROL LRotate into LSD.FF77CADEXDone 4 shifts?FF77CADEXDone 4 shifts?FF76DF8ENE HEXSHIFTNo, loop.FF7CC3INYAdvence text index.FF7TD0 EØENEXTHEXAlways taken. Check next character for hex.FF77C4 2ANOTHEXCPY YSAVCheck if L. Hempty (no hex digita).FF78F97F97EGQ ESCAPEYes, generate ESC sequence.FF78J24 2BBIT MODETest MODE byte.FF78J24 2BLDA LLSD's of hex data.FF78J26STA (STL, X)Store at current 'store index'.FF88D9 B5ENE NEXTTEMGet next texm. (no carry).FF88D4 B5NOTSTORBM NEXTHEMFF91GC 44 67TONEXTITEMJMP NEXTITEMFF91GC 44 67TONEXTITEMJMP NEXTITEMFF91S5STA ALAL-I, XGet next teom index in the LSO's AdM.FF91S2SETADRLDX 4502FF91S2SETADRLDX 4502FF91S4S4StarrawaFF91S4S5STA ALAL-I, XFF91S2SETADRLDA 4580FF41C4FFSETADRLDA 458DCR.CR.FF42D0 F7ENE SETADRLDA 458DCR.FF43D14NXTPRNTFF44D14NXTPRNTFF45J26JSA 44FF45			5502	ment month on hiom	
FF76OADEXDone 4 shifts?FF76OABNX THEXSHIP?FF76C6INYAdvence text index.FF77OF 4BN EXTHEXAlways taken. Check next character for hex.FF77OF 4AD EXTHEXAlways taken. Check next character for hex.FF78F97SC 24BN CNTEXFF785 42BIT MODETest MODE byte.FF785 16BV CNOTSTORB6 4 for STOR, 1 for XAM and BLOCK XAMFF78J 26STA (STL.)Store at current 'store index'.FF88D 26INC STLIncrement store index.FF89J 26STA (STL.)Store at current 'store index'.FF89B 26INC STLGet next item. (no carry).FF87F52TONEXTITEM MP NEXTITEMGet next item. (no carry).FF87J 26NOTSTORBM XAMNEYFF979 27STA STL-1, X'store index'.FF979 28NOTSTORBM XAMLDRun at current XAM index.FF979 29S 27STA AXML-1, XAndt 'cXAM index'.FF985 27STA AXML-1, XAndt 'cXAM index'.FF999 20GTES XAML-1, XAndt 'cXAM index'.FF979 24LD 4 4\$SDCR.FF985 25STA STL-1, X'store index'.FF999 26STA STL-1, X'store index'.FF979 26STA STL-1, X'store index'.FF985 27EDARLDa 4\$SDFF14D0 14NYTPRY	FF75	26 28		ROL L	Rotate into LSD.
FFTADØ F8BNE HEXSHIFTNo. loop.FFTCC6INYAdvence text index.FFTCC4NOTHEXBNE NEXTHEXFFTFC4 2ANOTHEXCheck if L. Hempty (no hex digits).FF81F9 97BEQ ESCAPEYes, generate ESC sequence.FF83F24 2BBIT MODETest MODE byte.FF84F26LDA LLSO's of hex data.FF85F26STA (STL, X)Store at current 'store index'.FF86F26INC STLIncrement store index'.FF87F27NOCESTHAdi carry to 'store index'.FF88F26INN (XANNEXTGet next icone index' high order.FF89F42TONEXTITEMMP NEXTITEMGet next icone index' high order.FF9942 44 0FTONEXTITEMMP NEXTITEMGet next command item.FF9942 44 0FINN (XANNEXTB7 # 0 for XAM, 1 for BLOCK XAM.FF9992 25STA XAML-1, XGoy hex data toFF9092 25STA XAML-1, XAdi to 'XAM index'.FF44D4 1NXTPRNTBNE SETADRLoop unless X # 0.FF44D4 7SDE STADRLDA AMH'XAmine index'.FF4532 25JSR PRBYTEOutput it.FF4534 26JSR PRBYTEOutput it.FF4536 26 FJSR PRBYTEOutput it.FF4536 26 FFJSR PRBYTEOutput it.FF4536 26 FFJSR PRBYTEOutput it.FF4536 26 FFJSR PRBYTEOutpu	FF77	26 29		ROL H	Rotate into MSD's.
FFTCG8INYAdvence text index.FFTDD9 60BNE NEXTREXAlways taken. Check next character for hex.FFTSG4 2ANOTHEXCPY YSAVCheck if L. H empty (no hex digits).FF81F907BEQ ESCAPEYes, generate ESC sequence.FF8332 42BBIT MODETest MODE byte.FF84F97A5 28LDA LStore at current 'store index'.FF8550 46STA (STL, X)Store at current 'store index'.FF89D0 55BNE NEXTITEMGet next item. (no carry).FF81F62TONEXTITEMMP (XAMI.)FF81G4 49FTTONEXTITEMGet next item. (AD index.)FF9146 44 FFTONEXTITEMMP (XAMI.)Run at current 'store index' ligh order.FF9146 44 FFTONEXTITEMMP (XAMI.)Run at current 'AAM index.FF9194 28NOTSTORBMI XAMNEXTB7 $= 0$ for XAM, 1 for BLOCK XAM.FF9195 25STA ARAL-1, Xat current 'AAM index'.FF9195 25STA ARAL-1, XAd to 'XAM index'.FF9295 23STA XAML-1, XAnd to 'XAM index'.FF42D4 F7BNE ESTADRLoop unless X = 0.FF43D4 14NXTPRNTNE ERADATANE means no address to print.FF44D4 14NXTPRNTBNE ESTADRLoop unless X = 0.FF4536 26LDA XAMI"it warmine index' ligh-order tyte.FF45A 36 26LDA XAMIExamine index'.FF4636 26<	FF79	CA		DEX	Done 4 shifts?
FFTD DØ DØ DNE NEXTHEX Always taken. Check next. character for hex. FFRT C4 2A NOTHEX BEQ ESCAPE Yes, generate ESC sequence. FF83 24 2B BIT MODE Test MODE byte. FF85 54 10 BVC NOTSTOR B6 * Ø for STOR, 1 for XAM and BLOCK XAM FF85 54 10 LDA L LSD's of hex data. FF86 126 STA (STL, X) Store at current 'store index'. FF86 D4 5 BNE NEXTITEM Get next item, ion carry). FF87 D4 55 BNE NEXTITEM Get next item. FF88 D4 5 TONEXTITEM Met next item. FF87 D4 55 TONEXTITEM Get next item. FF87 D4 50 RUN JMP (XAML) Run at current XAM index. FF97 32 42 LDX #302 Byte cont. FF97 93 20 STA STAL-1, X Cop hex data to FF97 94 20 D4 L-1, X Cop hex data to FF97 95 23 STA STAL-1, X And to 'XAM index'. FF44 D4 F7 BNE PERDATA NE means to address to print. FF42 D4 F7 JSR PEBYTE Output it. FF84 32 45 LDA #\$80 CR.	FF7A	DØF8		BNE HEXSHIFT	No, loop.
FFTC4 2ANOTHEXCPY YSAVCheck if L, H empty (no hex digits).FF81F997BC ESC ESC ESC equence.FF8324 2BBIT MODETest MODE byte.FF8559 10BC NOTSTORB6 4 for STOR. I for XAM and BLOCK XAMFF87A5 28LDA LLSD's of hex data.FF8859 10BSBNC NOTSTORBC 4 for STOR. I for XAM and BLOCK XAMFF8856 26NC STLIncrement store index'.FF89BC 26BNC NTTTEMGet next item. (no carry).FF81F62 40RINJMP NEXTITEMFF9146 44 FFTONEXTITEMMP (XAML).Run at current XAM index.FF9146 24UNJMP NEXTITEMGet next icommand item.FF9192 28NOTSTORBMI XAMNEXTB7 $=$ for XAM, 1 for BLOCK XAM.FF9195 25STA ASTL-1, X'store index'.FF9195 25STA ASML-1, XAnd to 'XAM index'.FF9295 23STA XAML-1, XAnd to 'XAM index'.FF42D4 F7BNE ESTADRLoop unless X = 4.FF43D4 14NXTPRNTNE ESTADRLoop unless X = 4.FF44D4 14NXTPRNTNE ESTADRLoop unless X = 4.FF4538 25LDA XAMLVituamine index' high-order byte.FF4526 CF FJSR PRBYTEOutput it in hex format.FF4626 FFJSR PRBYTEOutput it in hex format.FF47JG CF FJSR PRBYTEOutput it.FF47A3 24LDA XAML	FF7C	C8		INY	
FFTFC4 2ANOTHEXCPY YSAVCheck if L, H empty (no hex digits).FF81F971BEQ ESCAPEYes, generate ESC sequence.FF8354 2BBIT MODETest MODE byte.FF8556 10BVC NOTSTORB6 e f for STOR, I for XAM and BLOCK XAMFF8781 26STA (STL, X)Store at current 'store index'.FF8850 26INC STLIncrement store index.FF8852 6INC STLIncrement istore index'.FF8752 7INC STHAdd carry to 'store index'.FF9844 FFTONSTORBM XAMEXTB7 e for XAM, i for BLOCK XAM.FF9942 40NOTSTORBM XAMEXTB7 e for XAM, i for BLOCK XAM.FF9992 80NOTSTORBM XAMEXTB7 e for XAM, i for BLOCK XAM.FF9992 80NOTSTORBM XAMEXTB7 e for XAM, i for BLOCK XAM.FF9995 25STA STALL, X'store index'.FF9142 42NOTSTORBM XAMEXTB7 e for XAM index'.FF9295 25STA STALL, X'store index'.FF9420 F7BS ESTADRLop unless X = 0.FF44D14NXTPRNTBNE ESTADRLoop unless X = 0.FF4520 FFJSR PRETEOutput it.FF4620 CFFJSR PRETEOutput it.FF4620 CFFJSR PRETEOutput it.FF4640 CFFJSA KAMHExamine index' high-order byte.FF4620 CFFJSR PRETEOutput it.FF4735 26LDA XA	FF7D	DØ EØ		BNE NEXTHEX	Always taken. Check next character for her
FF81F0F0BEC SSGAPEYes, generate ESC sequence.FF8324 2BBIT MODETest MODE byte.FF8559 14BVC NOTSTOR $B6 = 9$ for STOR, 1 for XAM and BLOCK XAMFF8081 26STA (STL, X)Store at current 'store index'.FF80BC 62INC STLIncrement store index'.FF80D0 B5BNE NEXTITEMGet next citem. (no carry).FF81C4 4 07TONEXTITEMMP NEXTITEMFF914 C4 4 07RUNJAP (XAML)Run at current 'store index' high order.FF979 28NTSTORBMI XAMNETT7 # for XAM, 1 for BLOCK XAM.FF999 52STA STL-1, X'store index'.FF999 52STA XAML-1, XAnd to 'XAM index'.FF41CADEXNext of 2 bytes.FF42D0 F7BNE SETADRLoop unless X = 0.FF4320 F7JSR ECHOOutput it.FF449 DC.R.SER PERPTEFF4540 FFJSR PRBYTEOutput it.FF4649 B0C.R.SER CHOFF48A5 25LDA XAMLCow-order 'seamine index' byte.FF4978 A3 40FRDATALDA 45A0FF4020 CFFJSR PRBYTEOutput it.FF4049 A4LDA 45A4Bian.FF4020 CFFJSR ECHOOutput it.FF4040 CFFJSR ECHOOutput it.FF4020 CFFJSR ECHOOutput it.FF4040 CFFJSR ECHOOutp	FF7F	C4 2A	NOTHEX	CPY YSAV	
FF8324 2BBIT MODETest MODE byte.FF8559 10BVC NOTSTORB6 4 for STOR, 1 for XAM and BLOCK XAMFF8550 10B126STA (STL, X)Store at current 'store index'.FF80B126INC STLIncrement store index.FF80D135BNE NEXITIEMGet next item. (no carry).FF87E6 24INC STLMA datary to 'store index' high order.FF8762 40 00INC STLJMP NEXTITEMFF9732 2BNOTSTORBMIX AMNEXTFF9732 2BNOTSTORBMIX AMNEXTFF9752 32STA STL-1, X'store index'.FF9855 7SETADRLDA L-1, XCopy hex data toFF9995 25STA STL-1, X'store index'.FF44D0 14NXTPRNTBNE SETADRLoop unless X = 0.FF44D0 14NXTPRNTBNE SETADRLoop unless X = 0.FF44D2 0C FFJSR ECHOOutput it.FF45A9 8DCR.CR.FF46A9 40PRDATANE means no address to print.FF47JSR ECHOOutput it.FF48A5 24LDA A/AMHCarmine index' high-order byte.FF4924 DC FFJSR PRBYTEOutput it.FF4024 DC FFJSR PRBYTEOutput it.FF4024 DC FFJSR PRBYTEOutput it.FF44D4 CFFJSR ECHOOutput it.FF45A9 40CFFJSR PRBYTE74D4 CFFJSR PRBYTEOutpu	FF81	FØ 97		BEQ ESCAPE	
FF8554 14BVC NOTSTOR $B6 = 4$ for STOR, 1 for XAM and BLOCK XAMFF87A5 28LDA LLSD's of hex data.FF8981 26STA (STL, X)Store at current 'store index'.FF80D4 25BNE NEXTITEMGet next item. (no carry).FF87FF87TONEXTIEMJMP NEXTITEMFF8744 4F FTONEXTIEMTonextrine index'.FF9734 28NOTSTORBMI XAMNEXTFF9734 28NOTSTORBMI XAMNEXTFF9855 27SETADRLDA 4.FF999 525STA STL-1, X'store index'.FF91GC 44 00RUNJMP (XAML)Run at current XAM index.FF979 23STA STL-1, X'store index'.FF98B5 27SETADRLDA 41, XAnd to 'XAM 1 for BLOCK XAM.FF999 525STA STL-1, X'store index'.FF41CADA L.1, XAnd to 'XAM index'.FF42D4 71BNE SETADRLoop unless X = 4.FF43D4 14NXTPRNTBNE SETADRLoop unless X = 4.FF44D4 14NXTPRNTBNE PRDATANE means no address to print.FF4526 EF FFJSR ECHOOutput it.FF4526 EF FFJSR ECHOOutput it.FF4526 EF FFJSR ECHOOutput it.FF4526 EF FFJSR ECHOOutput it.FF4626 EF FFJSR ECHOOutput it.FF4626 EF FFJSR ECHOOutput it.FF4626 EF F	FF83			BIT MODE	
FF87A5 28LDA LLSD's of hex data.FF88B1 26STA (STL, X)Store at current 'store index'.FF88DØ B5BNE NEXTITEMGet next item. (no carry).FF87F6 27INC STHAdd carry to 'store index' high order.FF914C 44 FFTONEXTITEMJMP NEXTITEMFF946C 24 00RUNJMP (XAML)Run at current XAM index.FF9732 28NOTSTORBMI XAMNEXTB7 7 6 for XAM, 1 for BLOCK XAM.FF9737 28NOTSTORDMI XAMNEXTB7 7 8 for XAM, 1 for BLOCK XAM.FF9752 5SETADRLDA L-1, X'store index'.FF9795 25STA STL-1, X'store index'.FF44DØ F7BNE SETADRLoo yunless X = Ø.FF44DØ F7BNE SETADRLoo yungets.FF44DØ F7BNE SETADRLoo yungets.FF44DØ F7JSR ECHOOutput it.FF4529 DC FFJSR PRBYTEOutput it.FF8429 DC FFJSR PRBYTEOutput it.FF85A9 8ALDA 4\$A0Blank.FF8629 DC FFJSR PRBYTEOutput it.FF8728 CFFJSR ECHOOutput it.FF8629 DC FFJSR PRBYTEOutput it.FF8728 CFFJSR PRBYTEOutput it.FF86A9 A0PRDATALDA 4\$A0FF8726 CFFJSR PRBYTE757JSR ECHOOutput it.FF8729 DC FFJSR PRBYTE758CA	FF85	50 10		BVC NOTSTOR	
FF8981.26STA (STL, X)Store at current 'store index'.FF80DØ B5BNE NEXTITEMIncrement store index.FF80DØ B5BNE NEXTITEMGet next item. (no carry).FF87E6 27INC STLIncrement store index.FF914C 44 FFTONEXTITEMGet next item. (no carry).FF9732 BNOTSTORBMI XAMNEXTFF9732 BNOTSTORBMI XAMNEXTFF9952 5STA STL-1, XStore index'.FF9952 5STA STL-1, X'store index'.FF9952 5STA STL-1, X'store index'.FF41DØ F7BNE SETADRLoop uless X = Ø.FF42DØ F7BNE SETADRLoop uless X = Ø.FF43DØ 14NXTPRNTBNE PRDATANE means on address to print.FF44DØ 14NXTPRNTBNE PRDATANE means on address to print.FF4520 FFJSR PRBYTEOutput it.FF4520 GFFJSR PRBYTEOutput it.FF4640 GFFJSR PRBYTEOutput it. <tr< td=""><td>FF87</td><td></td><td></td><td>LDA L</td><td></td></tr<>	FF87			LDA L	
FF8DDØDØDKNC STLIncrement store index.FF8DDØBKE NEXTITEMGet next item. (no carry).FF8FE627INC STHGet next item. (no carry).FF946C 44 FFTONEXTITEMJMP (XAML)Get next item. (no carry).FF9730 2BNOTSTORBMI XAMNEXTB7 2 Ø for XAM. 1 for BLOCK XAM.FF9932 20LDX #\$92Byte cont.FF9955STA STL-1, X'store index'.FF9795 25STA STL-1, X'store index'.FF41CADEXNext of 2 bytes.FF42DØ F7BKE SETADRLoop unless X = Ø.FF44DØ F7NTPRNTBKE SETADRLoop unless X = Ø.FF45A9 8DCR.Examine index' high-order byte.FF46A9 8DCR.GR.FF45A5 25LDA XAML'Examine index' high-order byte.FF46A9 8DLDA #\$4BDCR.FF47JØ CFFJSR PRBYTEOutput it.FF48A5 24LDA XAMLLow-order 'examine index' byte.FF49A9 40PRDATALDA #\$4ABlank.FF45A9 40PRDATALDA #\$4ABlank.FF46A9 40FFFJSR PRBYTEOutput it.FF47JØ CFFJSR PRBYTEOutput it.FF48A5 24LDA XAMLCet data byte at 'examine index'.FF48A74FFFJSR PRBYTEOutput it.FF49A74EFFJSR PRBYTEOutp					
FF8DDØ B5BNE NEXTITEMGet next item. (no carry).FP8FE6 27INC STHAdd carry to 'store index' high order.FP914C 44 FFTONEXTITEMJMP NEXTITEMGet next command item.FP946C 24 00RUNJMP (XAML)Run at current XAM index.FF9732 42LDX #\$02Byte count.FF9795 25STA STL-1, XByte count.FF9195 25STA STL-1, X'store index'.FF41CADEXNext of 2 bytes.FF42DØ F7BNE SETADRLoop unless X = Ø.FF44DØ F7BNE SETADRLoop unless X = Ø.FF4520 FFJSR PCHOOutput it.FF4649 8DCR.Carry.FF47DØ CFFJSR PCHOOutput it.FF4820 CFFJSR PCHOOutput it.FF499 52SET SECHOOutput it.FF4020 CFFJSR PRBYTEOutput it.FF41DO CFFJSR PRBYTEOutput it.FF4220 CFFJSR PRBYTEOutput it.FF4520 CFFJSR PRBYTEOutput it.FF4520 CFFJSR PRBYTEOutput it.FF4620 CFFJSR PRBYTEOutput it.FF4720 CFFJSR PRBYTEOutput it.<					
FF8FE6 27INC STHAdd carry to 'store index' high order.FF914C 44 FFTONEXTITEMIMP NEXTITEMGet next command item.FF9730 2BNOTSTORBMI XAMNEXTB7 = Ø for XAM, 1 for BLOCK XAM.FF9793 2BNOTSTORBMI XAMNEXTB7 = Ø for XAM, 1 for BLOCK XAM.FF9795 26STA STL-1, XCopy hex data toFF9795 25STA XAML-1, XAnd to 'XAM index'.FF9795 23STA XAML-1, XAnd to 'XAM index'.FF41CADEXNext of 2 bytes.FF4220 F7BNE SETADRLoop unless X = Ø.FF44740 14NXTPRNTBNE PRDATAFFA680CR.'Examine index' high-order byte.FF4649 8DCR.'Examine index' high-order byte.FF4649 8DLDA #\$4DCR.FF4652 26JSR PRBYTEOutput it.FF4652 24LDA XAMLCervamine index' high-order byte.FF47JG CFFJSR PRBYTEOutput it.FF48A5 24LDA #\$AØBlank.FF499 BAJSR ECHOOutput it.FF4974 Ø EF FFJSR ECHOOutput it.FF4026 EF FFJSR PRBYTEOutput it.FF4762 6F FFJSR PRBYTEOutput it.FF4662 5<					
FF914C 44 FFTONEXTITEMJMP NEXTITEMGet next command item.FF946C 24 00RUNJMP (XAML)Run at current XAM index.FF9730 28NOTSTORBMI XAMNEXTBT # 0 for XAM, 1 for BLOCK XAM.FF9952 57SETADRLDA #\$02Byte count.FF9952 57SETADRLDA L-1, X'store index'.FF9152 58STA XAML-1, X'store index'.FF41CADEXNatt of 2 bytes.FF42D0 F7BNE SETADRLoop unless X = 0.FF44D0 14NXTPRNTBNE PRDATAFFA620 EF FJSR ECHOOutput it.FFA620 EF FJSR PRBYTEOutput it in hex format.FFB645 24LDA XAMLLow-order 'examine index' byte.FFB520 DC FFJSR PRBYTEOutput it in hex format.FFB520 DC FFJSR PRBYTEOutput it.FFB720 EF FFJSR CHOOutput it.FFB720 EF FFJSR PRBYTEOutput it.FFB720 DC FF					
FF946C 24 ØØRUNJMP (XAML)Run at current XAM index.FF973Ø 28NOTSTORBMI XAMNEXTB7 € Ø for XAM, 1 for BLOCK XAM.FF9795 26SETADRLDX #§02Byte count.FF9995 25STA XTL-1, XCopy hex data toFF9795 25STA XAML-1, XAnd to 'XAM index'.FF41CADEXNext of 2 bytes.FF42DØ F7ENE SETADRLDA #\$3DFF44DØ 14NXTPRNTBNE PRDATAFF45A9 8DCR.FF48A5 25JSR ECHOOutput it.FF48A5 25JSR PRBYTEOutput it in hex format.FFB2JØ DC FFJSR PRBYTEOutput it in hex format.FFB3A9 AØFRDATALDA #\$AØFFB4Q GC FFJSR PRBYTEOutput it.FFB549 GALDA #\$AØBlank.FFB7Q EF FFJSR ECHOOutput it.FFB7G EF FFJSR ECHOOutput it.FFB424LDA XAMLLow-order 'examine index' high-order byte.FFB7Q EF FFJSR PRBYTEOutput it.FFB7Q EF FFJSR ECHOOutput it.FFG46 25CMOOutput it.FFG46 25CMPCompare 'examine index'.FFG46 25LDA XAMLCet data byte at 'examine index'.FFG46 25LDA XAMLCompare 'examine index'.FFG46 28XAMNEXTSto MODEFFC46 29SEC H			TONEXTITEM		
FF973Ø 2BNOTSTORBMI XAMNEXTB7 = Ø for XAM, 1 for BLOCK XAM.FF99A2 Ø2LDX #\$Ø2Byte count.FF9995 25STA XALL-1, XCopy hex data toFF9795 25STA XALL-1, X'store index'.FF9795 23STA XALL-1, X'store index'.FF41CADEXNext of 2 bytes.FF42DØ F7BNE SETADRLoop unless X = Ø.FF44DØ 14NXTPRNTBNE PRDATANE means no address to print.FF482Ø EF FFJSR ECHOOutput it.FFA82Ø EF FFJSR ECHOOutput it in hex format.FFB4DG CFFJSR PRBYTEOutput it in hex format.FFB5A9 8DLDA #\$A0Utput it in hex format.FFB6A9 Ø0PRDATADa #\$BAFFB7JSR ECHOOutput it.FFB7JSR ECHOOutput it.FFB7JSR ECHOOutput it.FFB7JSR ECHOOutput it.FFB7JSR ECHOOutput it.FFB7JSR PRBYTEOutput it.FFB7S2BC MPFFC7A5 24LDA XAMLFF					
FF99A2 Ø2LDX #\$Ø2Byte count.FF99B5 27SETADRLDA L-1, XCopy hex data toFF9095 25STA STL-L, X'store index'.FF9195 23STA STL-L, X'store index'.FF9195 23STA STL-L, XNext of 2 bytes.FF41CADEXNext of 2 bytes.FF42DØ F7BNE SETADRLoop unless X = Ø.FF44DØ 14NXTPRNTBNE PRDATANE means no address to print.FF46A9 8DCR.Output it.FFA8A5 25LDA XAMH'Examine index' high-order byte.FFA8A5 24LDA #\$SDCR.FFB42Ø DC FFJSR PRBYTEOutput it in hex format.FFB52Ø DC FFJSR PRBYTEOutput it in hex format.FFB52Ø EF FFJSR ECHOOutput it.FFB6GE FFJSR PRBYTEOutput it.FFB72Ø EF FFJSR PRBYTEOutput it.FFB72Ø DC FFJSR PRBYTEOutput it.FFB72Ø EF FFJSR PRBYTEOutput it.FFB72Ø DC FFJSR PRBYTEOutput it in hex format.FFB72Ø DC FFJSR PRBYTEOutput it.FFB640 G CFFJSR PRBYTEOutput it.FFB720 GC FFJSR PRBYTEOutput it.FFB720 GC FFJSR PRBYTEOutput it.FFC686 23XAMNEXTSTA XAMLFFC652 8CMP LCompare 'examine index'.FFC652 8 <td></td> <td></td> <td></td> <td></td> <td></td>					
FF9BB5 27SETADRLDA L-1, XCopy hex data toFF9D95 25STA STL-1, X'store index'.FF4DD6STA XAML-1, XAnd to 'XAM index'.FFA1 CADEXNext of 2 bytes.FFA1 CADØ F7BNE SETADRLoo punless X # 0.FFA4 DØ 14NXTPRNTBNE PRDATANE means no address to print.FFA6 20 GF FFJSR ECHOOutput it.FFA7 20 GC FFJSR PRBYTEOutput it in hex format.FFB4 20 GC FFJSR PRBYTEOutput it in hex format.FFB5 40 GC FFJSR PRBYTEOutput it in hex format.FFB5 40 GE FFFJSR ECHOOutput it.FFB7 20 GE FFJSR PRBYTEOutput it.FFB7 20 GE FFFJSR PRBYTEOutput it.FFB7 20 GE FFFJSR PRBYTEOutput it.FFB7 41 24LDA #\$40Blank.FFB7 53 28XAMNEXTSTX MODEFFC1 20 GC FFJSR PRBYTEOutput it.FFC6 52 61LDA XAMLCompare 'examine index'.FFC6 52 7JSR PRBYTEOutput it.FFC6 52 8KAMNEXTSTX MODEFFC6 52 9SBC HCompare 'examine index'.FFC6 52 9SBC HFor MODE(XAM mode).FFD6 42 4INC XAMLCheck low-order 'examine index'.FFD6 42 5INC XAMHFor MOD 8= ØFFD6 43 124NOT 8507For MOD 8= ØFFD6 44LSRFor MOD 8= ØFFD6 45PRBYTEJSR PRHEXFFD6 46PRBYTEHAFFD7 40 <td></td> <td></td> <td>NOISION</td> <td></td> <td></td>			NOISION		
FF9095 25STA STL-1, X'store index',FF91CADEXNext of 2 bytes.FFA1CADEXNext of 2 bytes.FFA2DØ F7BNE SETADRLoop unless X = Ø.FFA4DØ 14NXTPRNTBNE PRDATANE means no address to print.FFA6A9 8DLDA #\$8DCR.FFA82Ø EF FFJSR ECHOOutput it.FFA8S2 5LDA XAMH'Examine index' high-order byte.FFA6A5 25LDA XAMLLow-order 'examine index' byte.FFB6A5 24LDA XAMLLow-order 'examine index' byte.FFB22Ø DC FFJSR PRBYTEOutput it in hex format.FFB5A9 AØPRDATALDA #\$AØBlank.FFB7ZØ EF FFJSR ECHOOutput it.FFB6A9 AØPRDATALDA #\$AØBlank.FFC1ZØ EF FFJSR PRBYTEOutput it.FFC626 EF FFJSR PRBYTEOutput it.FFC7A5 24LDA XAMLCet data byte at 'examine index'.FFC6SC 8CMP LCompare 'examine index'.FFC6SC 8CMP LCompare 'examine index'.FFC7A5 24LDA XAMLFFFC6SC 70NEXTITEMNot less, so no more data to output.FFC6SC 8INC XAMLFFC7S2 9SC 1FFC8G7SN AMNLFFC9A9 00RE NONBCHKFFC9A5 24MOD8CHKFFC6S2 4MOD8CHK			CETADD	and the set of the set of the set	
FF9F95 23STA XAML-1, XAnd to 'xAM index'.FFA1CADEXNext of 2 bytes.FFA2DØ F7BNE SETADRLoop unless X = Ø.FFA4DØ 14NXTPRNTBNE PRDATANE means no address to print.FFA6A9 8DCR.CR.FFA82Ø EF FFJSR ECHOOutput it.FFA8ZØ EF FFJSR PRBYTEOutput it in hex format.FFB4ZØ CF FJSR PRBYTEOutput it in hex format.FFB5A9 8DCFFJSR PRBYTEØ C FFJSR PRBYTEOutput it.FFB5A9 8ALDA #\$AMLFFB6A9 4ØPRDATALDA #\$AØBlank.FFB72Ø EF FFJSR ECHOØ EF FFJSR ECHOOutput it.FFB72Ø EF FFJSR PRBYTEOutput it.SR ECHOOutput it.FFC7A54LDA XAMLFFC62Ø EF FFJSR PRBYTEOutput it in hex format.FFC7FFC7A52LDA XAMLFFC6SC 52LDA XAMLFFC7SC 28CMP LFFC8C5 28CMP LFFC6BC 70Not less, so no more data to output.FFD6A524MOD8CHKIncrement 'examine index'.FFD7A00BNE MOD8CHKIncrement 'examine index'.FFC6BNE MOD8CHKIncrement 'examine index'.FFD6A524MOD8CHKLDA XAMLFFD6A524MOD8CHKLDA XAMLFFD6A52			SEIADR		
FFA1 FFA2 DV F7DEX BNE SETADR BNE SETADR BNE SETADR BNE SETADR Loop unless X = Ø.FFA4 FFA4 FFA4 FFA6 A9 8DNXTPRNT LDA \$\$8DNE means no address to print.FFA6 FFA8 ZØ EF FFJSR ECHO JSR ECHO DOUPUt it in hex format. LDA XAMH Low-order 'examine index' high-order byte.FFA6 FFA0 ZØ DC FFJSR PRBYTE JSR PRBYTE Output it in hex format. LDA *\$52 SFA3 DA LDA XAML LOW-order 'examine index' byte.FFB2 FFB2 ZØ DC FF FFB2 ZØ DC FF FFB2 ZØ DC FFJSR PRBYTE JSR PRBYTE Output it in hex format. LDA *\$58A ''.'.FFB5 FFB2 ZØ DC FF FFB7 ZØ EF FF SSR ECHO DOutput it. Compare 'examine index'.FFC4 FFC7 A5 24 FFC7 FFC6 FFC7 A5 25 FFC6 A5 25 FFC6 FFC7 A5 24 FFC6 FFC6 FFC7 A5 24 FFC6 FFC6 FFC7 FFC6 A5 25 FFC7 FFC6 A5 25 FFC7 FFC6 A5 26 FFC6 FFC7 FFC6 FFC7 A5 24 FFC6 FFC6 FFC6 FFC7 A5 24 FFC6 FFC6 FFC6 FFC6 FFC7 A5 24 FFC6 FFC6 FFC6 FFC7 A5 24 FFC6 FFC6 FFC7 A5 25 FFC6 A5 26 FFC7 FFC7 A5 26 FFC6 A5 27 FFC6 A5 27 FFC7 A5 28 FFC6 A5 28 FFC6 A5 29 FFC7 A5 29 FFC7 A5 29 FFC7 A5 29 FFC7 A5 29 FFC7 A5 29 FFC7 A5 29 FFC7 A5 A5 A5 A5 FFC7 A5 A5 A5 A5 FFC7 A5 A					
FFA2DØF7BNE SETADRLoop unless X = Ø.FFA4DØ14NXTPRNTBNE PRDATANE means no address to print.FFA6A9 8DLDA $\#$ 8DCR.FFA82Ø EF FFJSR ECHOOutput it.FFA82Ø DC FFJSR PRBYTEOutput it in hex format.FFB0A5 24LDA XAMLLow-order 'examine index' byte.FFB22Ø DC FFJSR PRBYTEOutput it in hex format.FFB22Ø DC FFJSR PRBYTEOutput it in hex format.FFB5A9 BALDA $\#$ 8AØ"i".FFB5A9 AØPRDATALDA $\#$ 8AØFFB6A9 AØPRDATALDA $\#$ 8AØFFB7JØ DC FFJSR ECHOOutput it.FFB7JØ DC FFJSR PRBYTEOutput it.FFB6A1 24LDA (XAML, X)Get data byte at 'examine index'.FFC7A5 24LDA XAMLCompare 'examine index'.FFC686 2BXAMNEXTSTX MODEØ \clubsuit MODE (XAM mode).FFC7A5 25LDA XAMLFrece Se 29FFC6BC TONEXTITEMNot less, so no more data to output.FFDØ46 25INC XAMLIncrement 'examine index'.FFDØ45 24MOD8CHKIDA XAMLFFD6A5 24MOD8CHKIDA XAMLFFD6A5 24MOD8CHKIncrement 'examine index'.FFD616STONEXTITEMNot less, so no more data to output.FFD6A5 24MOD8CHKIDA XAMLFFD6A5 24	and the second	A State of the second sec			
FFA4DØ 14NXTPRNTBNE PRDATANE means no address to print.FFA6A9 8DLDA #\$8DCR.FFA8A5 25LDA XAMH'Examine index' high-order byte.FFAD2Ø DC FFJSR PRBYTEOutput it in hex format.FFB22Ø DC FFJSR PRBYTEOutput it in hex format.FFB5A5 24LDA XAMLLow-order 'examine index' byte.FFB5A5 24LDA #\$AA''.''.FFB5A9 A0PRDATALDA #\$AØFFB6G EF FFJSR ECHOOutput it.FFB7JSR ECHOOutput it.FFB6A1 24LDA #\$AØFFC12Ø DC FFJSR PRBYTEOUTput it.FFG7FFC646 2BXAMNEXTST MODEØ C FFJSR PRBYTEOutput it.FFC6A6 2BXAMNEXTST MODEØ C FFJSR PRBYTEOutput it in hex format.FFC7A5 24LDA XAMLFFC6A5 25LDA XAMLFFC6A5 25LDA XAMHFFC6BC 1BCS TONEXTITEMNot less, so no more data to output.FFD0E6 24FFD0AD #\$97FFD0AD #\$97FFD04ALSRFFD04AFFD04AFFD04AFFD04AFFD645FFD7JSR PRHEXOutput hex digit.FFD64AFFD7 <t< td=""><td></td><td></td><td></td><td></td><td></td></t<>					
FFA6A9 8DLDA #\$8DCR.FFA620 EF FFJSR ECHOOutput it.FFA020 DC FFJSR PRBYTEOutput it in hex format.FFB0A5 24LDA XAMLLow-order 'examine index' byte.FFB220 DC FFJSR PRBYTEOutput it in hex format.FFB220 DC FFJSR PRBYTEOutput it in hex format.FFB5A9 BALDA #\$BA''.''FFB720 EF FFJSR ECHOOutput it.FFB6A9 A0PRDATALDA #\$A0Blank.FFE720 EF FFJSR ECHOOutput it.FFB6A1 24LDA (XAML, X)Get data byte at 'examine index'.FFC120 DC FFJSR PRBYTEOutput it in hex format.FFC486 2BXAMNEXTSTX MODEØ MODE (XAM mode).FFC526CMP LCompare 'examine index' to hex data.FFC6A5 25LDA XAMLCompare 'examine index' to hex data.FFC6FFCSEC TONEXTITEMNot less, so no more data to output.FFD6E6 24INC XAMLFor MOD 8= ØFFD6A5 24MOD8CHKLDA XAMLCheck low-order 'examine index'.FFD6A5 24MOD8CHKLDA XAMLCheck low-order 'examine index'.FFD6A5 24MOD8CHKLDA XAMLCheck low-order 'examine index'.FFD74ALDA XAMLCheck low-order 'examine index'.FFD6A5 24MOD8CHKLDA XAMLCheck low-order 'examine index'.FFD7A5 24MOD8CHK			NUTER		
FFA82Ø EF FFJSR ECHOOutput it.FFA8A5 25LDA XAMH'Examine index' high-order byte.FFA02Ø DC FFJSR PRBYTEOutput it in hex format.FFB4A5 24LDA XAMLLow-order 'examine index' byte.FFB5A9 BALDA #\$\$BA''.'.FFB72Ø EF FFJSR PRBYTEOutput it.FFB72Ø EF FFJSR ECHOOutput it.FFB72Ø EF FFJSR ECHOOutput it.FFB72Ø EF FFJSR ECHOOutput it.FFB720 EF FFJSR PRBYTEOutput it.FFB720 CFFJSR PRBYTEOutput it.FFC120 DC FFJSR PRBYTEOutput it in hex format.FFC220 DC FFJSR PRBYTEOutput it.FFC486 2BXAMNEXTSTX MODEØ MODE (XAM mode).FFC7A5 24LDA XAMLCompare 'examine index' to hex data.FFC6SC 528CMP LCompare 'examine index' to hex data.FFC7A5 25LDA XAMLFree examine index' to hex data.FFC6BØ C1BCS TONEXTITEMNot less, so no more data to output.FFD6A5 24MOD8CHKIncrement 'examine index'.FFD4E6 25INC XAMLCheck low-order 'examine index'.FFD643BPL NXTPRNTAlways taken.FFD748PRBYTEPHASave A for LSD.FFD643LSRFFFFFFD744LSRFFD64ALSRFFD7<			NXTPRNT		
FFABA5 25LDA XAMH'Examine index' high-order byte.FFAD 20 DC FFJSR PRBYTEOutput it in hex format.FFB5A5 24LDA XAMLLow-order 'examine index' byte.FFB220 DC FFJSR PRBYTEOutput it in hex format.FFB520 EF FFJSR ECHOOutput it.FFB720 EF FFJSR ECHOOutput it.FFB720 EF FFJSR ECHOOutput it.FFB720 EF FFJSR ECHOOutput it.FFB720 EF FFJSR PRBYTEOutput it.FFE7124LDA (XAML, X)Get data byte at 'examine index'.FFC120 DC FFJSR PRBYTEOutput it in hex format.FFC220 DC FFJSR PRBYTEOutput it in hex format.FFC352 8CMP LCompare 'examine index'.FFC486 2BXAMNEXTSTX MODE#MODE (XAM mode).FFC552 8CMP LCompare 'examine index' to hex data.FFC652 8CMP LCompare 'examine index'.FFC652 9SBC HFFFD652 8INC XAMLFFD762 4INC XAMLFFD652 8INC XAMLFFD652 8INC XAMLFFD652 9INC XAMLFFD643BPL NXTPRNTFFD644ISRFFD748PRBYTEFFD648PRBYTEFFD748PRBYTEFFD648PRBYTEFFD744LSRFFD7<					
FFAD20DC FFJSR PRBYTEOutput it in hex format.FFB0A5 24LDA XAMLLow-order 'examine index' byte.FFB220DC FFJSR PRBYTEOutput it in hex format.FFB5A9 BALDA #\$BA''''.FFB720EF FFJSR ECHOOutput it.FFB6A9 AØPRDATALDA #\$AØBlank.FFB7JSR ECHOOutput it.FFB7JSR PRBYTEOutput it.FFB7JSR PRBYTEOutput it.FFB7JC FFJSR PRBYTEOutput it.FFC486 2BXAMNEXTSTX MODEFFC7A5 24LDA XAMLFFC6C5 28CMP LCompare 'examine index' to hex data.FFC6S5 2LDA XAMHFFC6B6 21BCS TONEXTITEMFFD6B0 621BCS TONEXTITEMFFD7A008CHKLDA XAMLFFD8E6 25INC XAMHFFD4E6 25INC XAMHFFD529 Ø7MD #\$407FFD4E6 25MOD8CHKFFD529 Ø7AD #\$407FFD4ALSRFFD54ALSRFFD54ALSRFFD64ALSRFFD74ALSRFFD74ALSRFFD64ALSRFFD74ALSRFFD64ALSRFFD64ALSRFFD64ALSRFFD74ALSRFFD6 <td></td> <td></td> <td></td> <td></td> <td></td>					
FFBØA5 24LDA XAMLLow-order 'examine index' byte.FFB220 DC FFJSR PRBYTEOutput it in hex format.FFB5A9 BALDA #\$BA".".FFB720 EF FFJSR ECHOOutput it.FFBAA9 AØPRDATALDA #\$AØBlank.FFB620 EF FFJSR ECHOOutput it.FFB720 CFFJSR PRBYTEOutput it.FF6120 DC FFJSR PRBYTEOutput it in hex format.FFC486 2BXAMNEXTSTX MODE $\emptyset \rightarrow MODE$ (XAM mode).FFC5A5 24LDA XAMLFFC6FFC6A5 25LDA XAMHFFC6A5 25LDA XAMHFFC6A5 25LDA XAMHFFC6A5 25LDA XAMLFFD2DØ Ø2BNE MOD8CHKFFD446 25INC XAMLFFD5A9A0FFD6A5 24MOD8CHKFFD7AND #\$Ø7For MOD 8 = ØFFD829 Ø7AND #\$Ø7FFD7ANFFD64ALSRFFD74ALSRFFD74ALSRFFD74ALSRFFD74ALSRFFF629 ØFPRHEXOutput hex digit.FFF29 ØFFFF468FFF5JSR PRHEXOutput hex digit.FFF2FFF4FFF468FFF5PLAFFF69 EØFFF69 EØFFF79 EØ <td></td> <td></td> <td></td> <td></td> <td></td>					
FFB220DC FFJSR PRBYTEOutput it in hex format.FFB5A9 BALDA #\$BA".",FFB720EF FFJSR ECHOOutput it.FFBAA9 A0PRDATALDA #\$A0Blank.FFBC20EF FFJSR ECHOOutput it.FFBFA1 24LDA (XAML, X)Get data byte at 'examine index'.FFC120DC FFJSR PRBYTEOutput it in hex format.FFC486 2BXAMNEXTSTX MODE $0 \rightarrow MODE$ (XAM mode).FFC3C5 28CMP LCompare 'examine index' to hex data.FFC485 25LDA XAMLFFC6FFC652 29SBC HFFD2DØ 92BNE MOD8CHKIncrement 'examine index'.FFD4E6 24INC XAMLCheck low-order 'examine index'.FFD4E6 25INC XAMLCheck low-order 'examine index' byteFFD829 Ø7AND #\$Ø7For MOD 8 = ØFFDD4ALSRISRFFD544LSRFFD54ALSRFFD54ALSRFFD64ALSRFFD74ALSRFFD74ALSRFFE468PLAFFE468FFE59FRHEXOutput hex digit.FFE468FFE59FFE468FFE59FFE59FFE468FFE59FFE59FF					
FFB5A9 BALDA #\$BA":".FFB720 EF FFJSR ECHOOutput it.FFBAA9 AØPRDATALDA #\$AØBlank.FFBCØ EF FFJSR ECHOOutput it.FFBFA1 24LDA (XAML, X)Get data byte at 'examine index'.FFC120 DC FFJSR PRBYTEOutput it in hex format.FFC486 2BXAMNEXTSTX MODEØ → MODE (XAM mode).FFC7A5 24LDA XAMLCompare 'examine index' to hex data.FFC8C5 28CMP LCompare 'examine index' to hex data.FFC686 C1BCS TONEXTITEMNot less, so no more data to output.FFD0E6 24INC XAMLFFD0E6 25INC XAMHFFD480 C3BPL NXTPRNTFFD6A5 24MOD8CHKFFD7AND #\$\$7For MOD 8= ØFFD829 Ø7AND #\$\$7FFD24ALSRFFD54ALSRFFD64ALSRFFD74ALSRFFD64ALSRFFD64ALSRFFD64ALSRFFD64ALSRFFD64ALSRFFD74ALSRFFD64ALSRFFD64ALSRFFD74ALSRFFD64ALSRFFD74ALSRFFD64ALSRFFD64ALSRFFF690 ØFFE790 Ø<		and the second second second			Low-order 'examine index' byte.
FFB7 $2\emptyset$ EF FFJSR ECHOOutput it.FFBA A9 AØPRDATALDA $\#$ \$AØBlank.FFBC 2Ø EF FFJSR ECHOOutput it.FFBF A1 24LDA (XAML, X)Get data byte at 'examine index'.FFC12Ø DC FFJSR PRBYTEOutput it in hex format.FFC486 2BXAMNEXTSTX MODE $\emptyset \rightarrow$ MODE (XAM mode).FFC352 28CMP LCompare 'examine index' to hex data.FFC6S5 28LDA XAMLFFC6 25 29FFC7SBC HCompare 'examine index' to hex data.FFC6E5 29SBC HFFD2DØ Ø2BNE MOD8CHKFFD4E6 25INC XAMLFFD5A5 24MOD8CHKFFD6A5 24MOD8CHKFFD7AND $\#$ \$Ø7FFD829 Ø7AND $\#$ \$Ø7FFD829 Ø7AND $\#$ \$Ø7FFD04ALSRFFD54ALSRFFD54ALSRFFD64ALSRFFD74ALSRFFD74ALSRFFF2JSR PRHEXOutput hex digit.FFE468PLAFFE468PLAFFE468FFE5PSR PRHEXOutput hex digit.FFE468FFE5Mask LSD for hex print.FFE7Ø9 BØORA $\#$ \$BØAdd "Ø".					
FFBAA9AØPRDATALDA $\#$AØ$ Blank.FFBC2Ø EF FFJSR ECHOOutput it.FFBF A124LDA (XAML, X)Get data byte at 'examine index'.FFC12Ø DC FFJSR PRBYTEOutput it in hex format.FFC4862BXAMNEXTSTX MODEØ \rightarrow MODE (XAM mode).FFC7A524LDA XAMLFFC8C528CMP LCompare 'examine index' to hex data.FFC6FFCBSC HCompare 'examine index' to hex data.FFC6E529SBC HFFC2E9SBC HFFC4BOD8CHKIncrement 'examine index'.FFD5DØ Ø2BNE MOD8CHKIncrement 'examine index'.FFD4E6E5INC XAMLCheck low-order 'examine index' byteFFD6A524MOD8CHKLDA XAMLFFD6A524MOD8CHKLDA XAMLFFD6A524MOD8CHKIncrement 'examine index'.FFD6A524MOD8CHKLDA XAMLFFD6A524MOD8CHKLDA XAMLFFD6A524MOD8CHKLDA XAMLFFD6A524MOD8CHKIncrement 'examine index'.FFD7B07For MOD 8= ØFFD648PRBYTEPHAFFD744LSRFFD744LSRFFD64ALSRFFE04ALSRFFE120 E5 FFJSR PRHEX <t< td=""><td></td><td></td><td></td><td></td><td>":"•</td></t<>					":"•
FFBC $2\emptyset$ EF FFJSR ECHOOutput it.FFBFA1 24LDA (XAML, X)Get data byte at 'examine index'.FFC12Ø DC FFJSR PRBYTEOutput it in hex format.FFC486 2BXAMNEXTSTX MODE $\emptyset + MODE$ (XAM mode).FFC7A5 24LDA XAML $\emptyset + MODE$ (XAM mode).FFC8C5 28CMP LCompare 'examine index' to hex data.FFC6F5 29SBC HFFC6FFC6B0 C1BCS TONEXTITEMNot less, so no more data to output.FFD0E6 24INC XAMLIncrement 'examine index'.FFD2DØ Ø2BNE MOD8CHKIncrement 'examine index'.FFD4E6 25INC XAMLFor MOD 8= ØFFD548PRBYTEPHASave A for LSD.Save A for LSD.FFDD 4ALSRLSRFFD54ALSRFFD64ALSRFFD64ALSRFFE468PLAFFE529 ØFPRHEXFFE468FFE7MSD to LSD position.FFE74AFFE64AFFE7Mask LSD for hex print.FFE7Ø 50FFE7Ø 50FFE7 <td>FFB7</td> <td>20 EF FF</td> <td></td> <td>JSR ECHO</td> <td>Output it.</td>	FFB7	20 EF FF		JSR ECHO	Output it.
FFBFA1 24LDA (XAML, X)Get data byte at 'examine index'.FFC12Ø DC FFJSR PRBYTEOutput it in hex format.FFC486 2BXAMNEXTSTX MODE $\emptyset $			PRDATA		Blank.
FFC1 $2\emptyset$ DC FFJSR PRBYTEOutput it in hex format.FFC486 2BXAMNEXTSTX MODE $\emptyset wedge MODE$ (XAM mode).FFC7A5 24LDA XAMLFFC8C5 28CMP LCompare 'examine index' to hex data.FFC6A5 25LDA XAMHFFC6E5 29SEC HFFC6E6 24INC XAMLFFD2DØ Ø2BNE MOD8CHKFFD4E6 25FFD4E6 25FFD829 Ø7AND #\$Ø7For MOD 8= ØFFDC 48PRBYTEFFDD 4ALSRFFD54AFFD54AFFD64AFFD64AFFD6FFD64AFFD74AFFD6FFD74AFFD6FFD6FFD7FFD74AFFF6FFD6FFF7FFF7FFF64AFFF6FFF64AFFF6FFF6FFF7999999999999999999999999999<	FFBC	20 EF FF		JSR ECHO	Output it.
FFC486 2BXAMNEXTSTX MODE $ \ensuremath{ \ensu$	FFBF	A1 24		LDA (XAML, X)	
FFC7A5 24LDA XAMLFFC8C5 28CMP LCompare 'examine index' to hex data.FFC4A5 25LDA XAMHFFCCE5 29SBC HFFC6EØ C1BCS TONEXTITEMNot less, so no more data to output.FFD0E6 24INC XAMLFFD4E6 25INC XAMHFFD6A5 24MOD8CHKIncrement 'examine index'.FFD6A5 24MOD8CHKLDA XAMLFFD7A00 #\$07For MOD 8= ØFFD829 Ø7AND #\$07For MOD 8= ØFFD04ABPL NXTPRNTAlways taken.FFD04ALSRFFDF4ALSRFFDF4ALSRFFDF4ALSRFF12Ø ØFPRHEXOutput hex digit.FFE468FF2468FF259 ØFFF269 ØFFF269 ØFFF27Ø 9 BØORA #\$BØAdd "Ø".	FFC1	20 DC FF		JSR PRBYTE	
FFC8C5 28CMP LCompare 'examine index' to hex data.FFCAA5 25LDA XAMHFFCCE5 29SBC HFFCEBØ C1BCS TONEXTITEMNot less, so no more data to output.FFDØE6 24INC XAMLFFD2DØ Ø2BNE MOD8CHKIncrement 'examine index'.FFD4E6 25INC XAMHFFD5A5 24MOD8CHKLDA XAMLFFD6A5 24MOD8CHKLDA XAMLFFD6A5 24MOD8CHKLDA XAMLFFD6A5 24MOD8CHKLDA XAMLFFD748PRBYTEPHASave A for LSD.FFD04ALSRFFDF4ALSRFFDF4ALSRFFFDF4ALSRFFE120 £5 FFJSR PRHEXFFE468PLAFFE529 ØFPRHEXAND #\$ØFMask LSD for hex print.FFE529 ØFPRHEXAND #\$ØFMask LSD for hex print.	FFC4	86 2B	XAMNEXT	STX MODE	Ø MODE (XAM mode).
FFCAA5 25LDA XAMHFFCCE5 29SBC HFFCEBØ C1BCS TONEXTITEMFFDØE6 24INC XAMLFFD2DØ Ø2BNE MOD8CHKFFD4E6 25INC XAMHFFD6A5 24MOD8CHKFFD829 Ø7AND #\$Ø7FFDAIØ C8BPL NXTPRNTFFDC48PRBYTEFFD24ALSRFFD54ALSRFFD64ALSRFFFØ4ALSRFFFØ4AASRFFEØ4AASRFFEØ4AASRFFF2JSR PRHEXOutput hex digit.FFE468PLAFF529 ØFPRHEXAND #\$ØFMask LSD for hex print.FFE7Ø9 BØORA #\$BØAdd "Ø".Orther for the print.				LDA XAML	
FFCCE5 29SBC HFFCEBØ C1BCS TONEXTITEMNot less, so no more data to output.FFDØE6 24INC XAMLFFD2DØ Ø2BNE MOD8CHKIncrement 'examine index'.FFD4E6 25INC XAMHFFD6A5 24MOD8CHKLDA XAMLFFD829 Ø7AND #\$Ø7For MOD 8= ØFFDAIØ C8BPL NXTPRNTAlways taken.FFDC48PRBYTEPHASave A for LSD.FFDD4ALSRFFDFFFDF4ALSRFFE468PLAMSD to LSD position.FFE468PLARestore A.FFE529 ØFPRHEXAND #\$ØFMask LSD for hex print.GRA #\$BØAdd "Ø".	FFC8	,C5 28		CMP L	Compare 'examine index' to hex data.
FFCEBØ C1BCS TONEXTITEM INC XAMLNot less, so no more data to output.FFDØE6 24INC XAMLIncrement 'examine index'.FFD2DØ Ø2BNE MOD8CHKIncrement 'examine index'.FFD4E6 25INC XAMHCheck low-order 'examine index' byteFFD6A5 24MOD8CHKLDA XAMLCheck low-order 'examine index' byteFFD829 Ø7AND #\$Ø7For MOD 8= ØFFDAIØ C8BPL NXTPRNTAlways taken.FFDC48PRBYTEPHASave A for LSD.FFDD4ALSRFFDEFFDF4ALSRFFDEFFEØ4ALSRFFEFFEØ4ALSRFFE12Ø E5 FFJSR PRHEXOutput hex digit.FFE468PLARestore A.FFE529 ØFPRHEXAND #\$ØFMask LSD for hex print.ORA #\$BØAdd "Ø".	FFCA	A5 25		LDA XAMH	
FFDØE6 24INC XAMLFFD2DØØ2BNE MOD8CHKIncrement 'examine index'.FFD4E6 25INC XAMHCheck low-order 'examine index' byteFFD6A5 24MOD8CHKLDA XAMLCheck low-order 'examine index' byteFFD829 Ø7AND #\$Ø7For MOD 8 = ØFFDA1Ø C8BPL NXTPRNTAlways taken.FFDC48PRBYTEPHASave A for LSD.FFDD4ALSRFFDEFFDF4ALSRFFEØ4ALSRFFEØ4ALSRFFEØ4ALSRFFEØ4ALSRFFEØ4ALSRFFEØ4ALSRFFEØ4ALSRFFEØ9 ØFPRHEXOutput hex digit.FFE468FFE529 ØFFFE7Ø BØORA #\$BØAdd "Ø".	FFCC	E5 29		SBC H	
FFD2DØ Ø2BNE MOD8CHKIncrement 'examine index'.FFD4E6 25INC XAMHFFD6A5 24MOD8CHKLDA XAMLCheck low-order 'examine index' byteFFD829 Ø7AND #\$Ø7For MOD 8= ØFFDA1Ø C8BPL NXTPRNTAlways taken.FFDC48PRBYTEPHASave A for LSD.FFDD4ALSRFFDEFFDF4ALSRFFDEFFEØ4ALSRFFEØFFEØ4ALSRFFEØFFEØ4ALSRFFEØFFEØ4ALSRFFEØFFEØ4ALSRFFEØFFEØ4ALSRFFEØFFEØ4ALSRFFEØFFEØ9 ØFPRHEXADD #\$ØFFFE529 ØFPRHEXAND #\$ØFFFE7Ø9 BØORA #\$BØAdd "Ø".	FFCE	BØ C1		BCS TONEXTITEM	Not less, so no more data to output.
FFD4E6 25INC XAMHFFD6A5 24MOD8CHKLDA XAMLCheck low-order 'examine index' byteFFD829 Ø7AND #\$Ø7For MOD 8= ØFFD810 C8BPL NXTPRNTAlways taken.FFDC48PRBYTEPHASave A for LSD.FFDD4ALSRFFDEFFDF4ALSRFFDF4ALSRFFDF4ALSRFFEØ4ALSRFFEØ4ALSRFFEØ4ALSRFFEØ4ALSRFFEØ4ALSRFFEØ4ALSRFFEØ9 ØFPRHEXAND #\$ØFMask LSD for hex print.FFE7Ø9 BØORA #\$BØAdd "Ø".Fd "Ø".	FFDØ	E6 24		INC XAML	
FFD6A5 24MOD8CHKLDA XAMLCheck low-order 'examine index' byteFFD829 Ø7AND #\$Ø7For MOD 8= ØFFDA1Ø C8BPL NXTPRNTAlways taken.FFDC48PRBYTEPHASave A for LSD.FFDD4ALSRFFDEFFDF4ALSRMSD to LSD position.FFEØ4ALSRFFEØFFEØ4ALSRFFEFFEØ4ALSRFFEØFFEØ4ALSRFFEØFFEØ4ALSRFFEØFFEØ4ALSRCutput hex digit.FFEØ2Ø E5 FFJSR PRHEXOutput hex digit.FFE468PLARestore A.FFE529 ØFPRHEXAND #\$ØFFFE7Ø9 BØORA #\$BØAdd "Ø".	FFD2	DØ Ø2		BNE MOD8CHK	Increment 'examine index'.
FFD829 Ø7AND #\$Ø7For MOD 8= ØFFDA1Ø C8BPL NXTPRNTAlways taken.FFDC48PRBYTEPHASave A for LSD.FFDD4ALSRLSRFFDF4ALSRMSD to LSD position.FFEØ4ALSRMSD to LSD position.FFEØ4ALSRFFEFFEØ4ALSRFFEØ4ALSRFFEØ4ALSRFFE12Ø E5 FFJSR PRHEXFFE468PLAFFE529 ØFFFE7Ø9 BØORA #\$BØAdd "Ø".	FFD4	E6 25		INC XAMH	
FFDA1Ø C8BPL NXTPRNTAlways taken.FFDC48PRBYTEPHASave A for LSD.FFDD4ALSRFFDEFFDF4ALSRMSD to LSD position.FFEØ4ALSRMSD to LSD position.FFEØ4ALSRFFEIFFEØ4ALSRFFEØ4ALSRFFEØ4ALSRFFEØ5FFJSR PRHEXOutput hex digit.FFE468PLAFFE529 ØFPRHEXAND #\$ØFMask LSD for hex print.FFE7Ø9 BØORA #\$BØ	FFD6	A5 24	MOD8CHK	LDA XAML	Check low-order 'examine index' byte
FFDA1Ø C8BPL NXTPRNTAlways taken.FFDC48PRBYTEPHASave A for LSD.FFDD4ALSRFFDEFFDF4ALSRMSD to LSD position.FFEØ4ALSRMSD to LSD position.FFEØ4ALSRFFEIFFEØ4ALSRFFEØ4ALSRFFEØ4ALSRFFEØ55JSR PRHEXOutput hex digit.PLAFFE529 ØFPRHEXFFE7Ø9 BØORA #\$BØAdd "Ø".Other print.	FFD8	29 Ø7		AND #\$Ø7	For MOD 8 = Ø
FFDC 48PRBYTEPHASave A for LSD.FFDD 4ALSRFFDE 4ALSRFFDF 4ALSRFFEØ 4ALSRFFEØ 4ALSRFFE1 2Ø E5 FFJSR PRHEXOutput hex digit.FFE4 68PLAFFE5 29 ØFPRHEXAND #\$ØFMask LSD for hex print.FFE7 Ø9 BØORA #\$BØ	FFDA	1Ø C8			Always taken.
FFDD 4ALSRFFDE 4ALSRFFDF 4ALSRFFEØ 4ALSRFFEØ 4ALSRFFE1 2Ø E5 FFJSR PRHEXFFE4 68PLAFFE5 29 ØFPRHEXAND #\$ØFMask LSD for hex print.FFE7 Ø9 BØORA #\$BØ	FFDC	48	PRBYTE	PHA	
FFDE4ALSRFFDF4ALSRMSD to LSD position.FFEØ4ALSRFFE12Ø E5 FFJSR PRHEXOutput hex digit.FFE468PLARestore A.FFE529 ØFPRHEXAND #\$ØFMask LSD for hex print.FFE7Ø9 BØORA #\$BØAdd "Ø".				LSR	
FFDF4ALSRMSD to LSD position.FFEØ4ALSRFFE12Ø E5 FFJSR PRHEXOutput hex digit.FFE468PLARestore A.FFE529 ØFPRHEXAND #\$ØFMask LSD for hex print.FFE7Ø9 BØORA #\$BØAdd ''Ø''.					
FFEØ4ALSRFFE12Ø E5 FFJSR PRHEXOutput hex digit.FFE468PLARestore A.FFE529 ØFPRHEXAND #\$ØFMask LSD for hex print.FFE7Ø9 BØORA #\$BØAdd "Ø".					MSD to LSD position.
FFE12Ø E5 FFJSR PRHEXOutput hex digit.FFE468PLARestore A.FFE529 ØFPRHEXAND #\$ØFMask LSD for hex print.FFE7Ø9 BØORA #\$BØAdd ''Ø''.					
FFE468PLARestore A.FFE529 ØFPRHEXAND #\$ØFMask LSD for hex print.FFE7Ø9 BØORA #\$BØAdd ''Ø''.					Output hex digit.
FFE529 ØFPRHEXAND #\$ØFMask LSD for hex print.FFE7Ø9 BØORA #\$BØAdd ''Ø''.					
FFE7 Ø9 BØ ORA #\$BØ Add "Ø".			PRHEX		
		5, 5		() 4	

6502 HEX MONITOR LISTING (continued)

FFEB	90 02	BCC ECHO	Yes, output it.
FFED	69 Ø6	ADC #\$Ø6	Add offset for letter.
FFEF	2C 12 DØ ECHO	BIT DSP	DA bit (B7) cleared yet?
FFF2	3Ø FB	BMI ECHO	No, wait for display.
FFF4	8D 12 DØ	STA DSP	Output character. Sets DA.
FFF7	60	RTS	Return.
FFF8	00 00 (unused)		

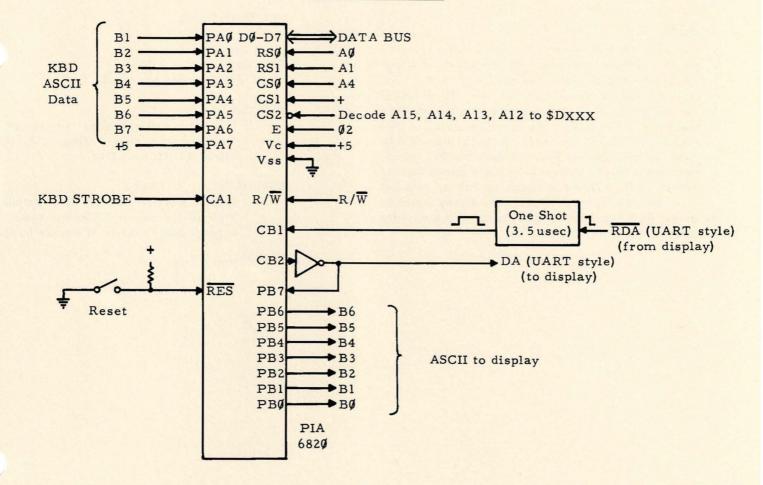
HARDWARE NOTES

Page Ø Vari	ables
XAML	24
XAMH	25
STL	26
STH	27
L	28
Н	29
YSAV	2A
MODE	2B

FFFA ØØ ØF (NMI) FFFC ØØ FF (RESET) FFFE ØØ ØØ (IRQ)

Other Var	lables	
IN	200-27F.	
KBD	DØ1Ø	
KBD CR	DØ11	PIA
DSP	DØ12	
DSP CR	DØ13)
)

KBD/DSP Interface



-7-

SECTION III HOW TO EXPAND THE APPLE SYSTEM

The Apple system can be expanded to include more memory and IO devices, via a 44-pin edge connector. The system is fully expandable to 65K, with the entire data and address busses, clocks, control signals (i. e. IRQ, NMI, DMA, RDY, etc.), and power sources available at the connector. All address lines are TTL buffered, and data lines can drive ten equivalent capacitive loads (one TTL load and 130pf) without external buffers. All clock signals are TTL. The Apple system runs at approximately 1 MHz (see spec sheet) and is fully compatible with 6800/6500 style timing.

Three power sources are available at the edge connector: +5 volts regulated, and raw DC (approximately +/- 14V) for the +12V, -12V, and -5V supplies. If +12V, -12V, or -5V supplies are required, EXTERNAL REGULATORS MUST BE USED. An excess of 1.5 amps from the "onboard" regulated +5V supply is available for expansion (assuming suitable transformer ratings are employed). Exercise great care in the handling of the raw DC, as no short-circuit protection is provided.

REFRESH:

Four out of every 65 clock cycles is dedicated to memory refresh. At the start of a refresh cycle (150 ns after leading edge of \emptyset 1), RF goes low, and remains low for one clock cycle. \emptyset 2 is inhibited during a refresh cycle, and the processor is held in \emptyset 1 (it's inactive state). Dynamic memories, which must clock during refresh cycles, should derive their clock from \emptyset 0, which is equivalent to \emptyset 2, except that it continues during a refresh cycle. Devices, such as PIA's, will not be affected by a refresh cycle, since they react to \emptyset 2 only. Refer to Apple "Tech Notes" for a variety of interfacing examples. DMA:

The Apple system has full DMA capability For DMA, the DMA control line tri-states the address buss, thus allowing external devices to control the buss. Consult MOS TECHNOLOGY 6502 Hardware Manual for details. (For DMA use, the solder jumper on the board, marked "DMA", must be broken.)

For the 6502 microprocessor, the RDY line is used to halt the processor for single stepping, or slow ROM applications. Refer to Apple "Tech Notes" for examples.

SOFTWARE CONSIDERATIONS:

The sequences listed below are the routines used to read the keyboard or output to the display.

Read Key from KBD:

(LDA KBD CR (DØ11) BPL LDA KBD DATA (DØ1Ø)

Output to Display: (DØ12) BPL STA DSP (DØ12) (DØ12)

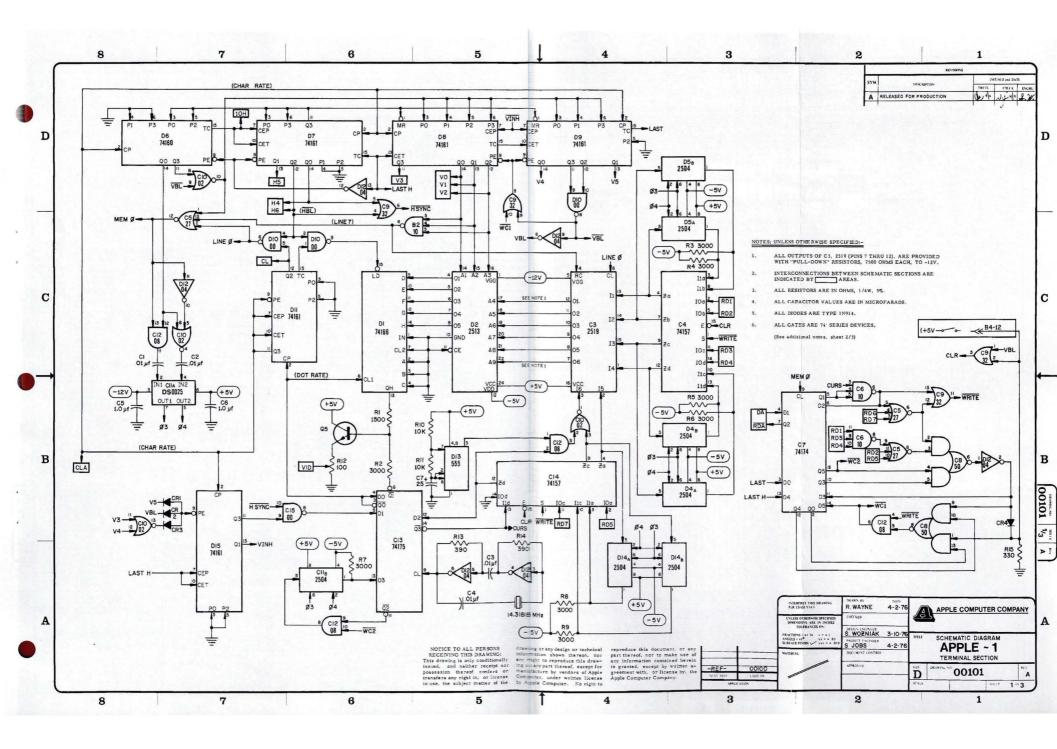
PIA Internal Registers: KBD Data DØ1Ø High order bit equals 1.

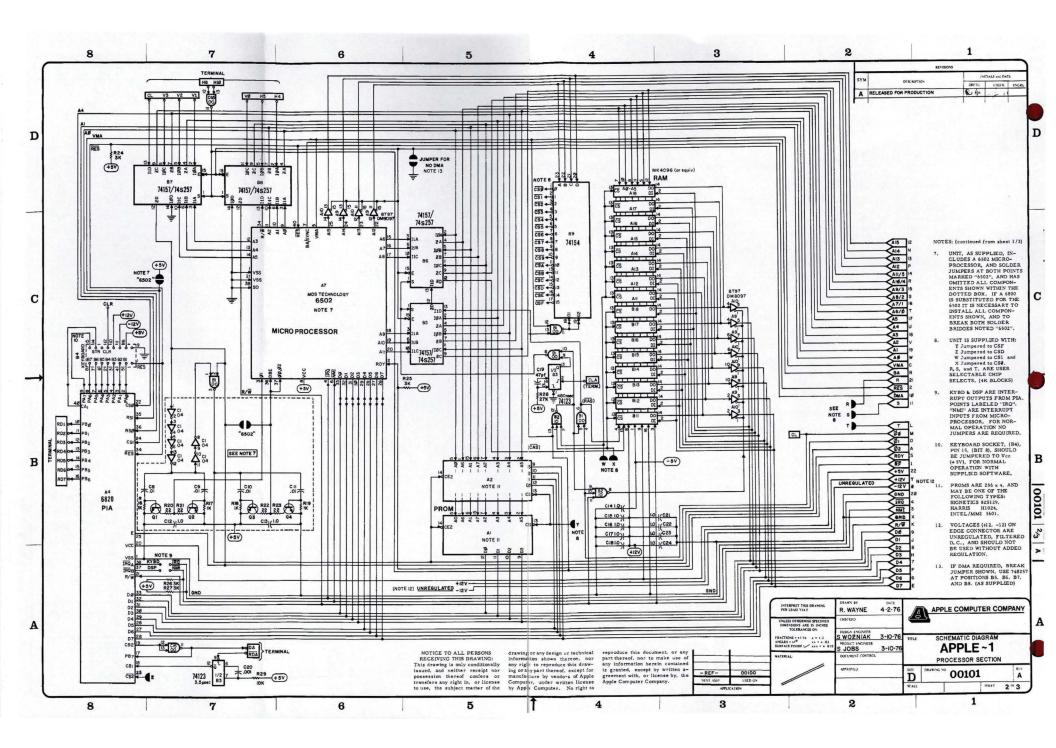
> KBD Control Reg. DØ11 High order bit indicates "key ready". Reading key clears flag. Rising edge of KBD sets flag.

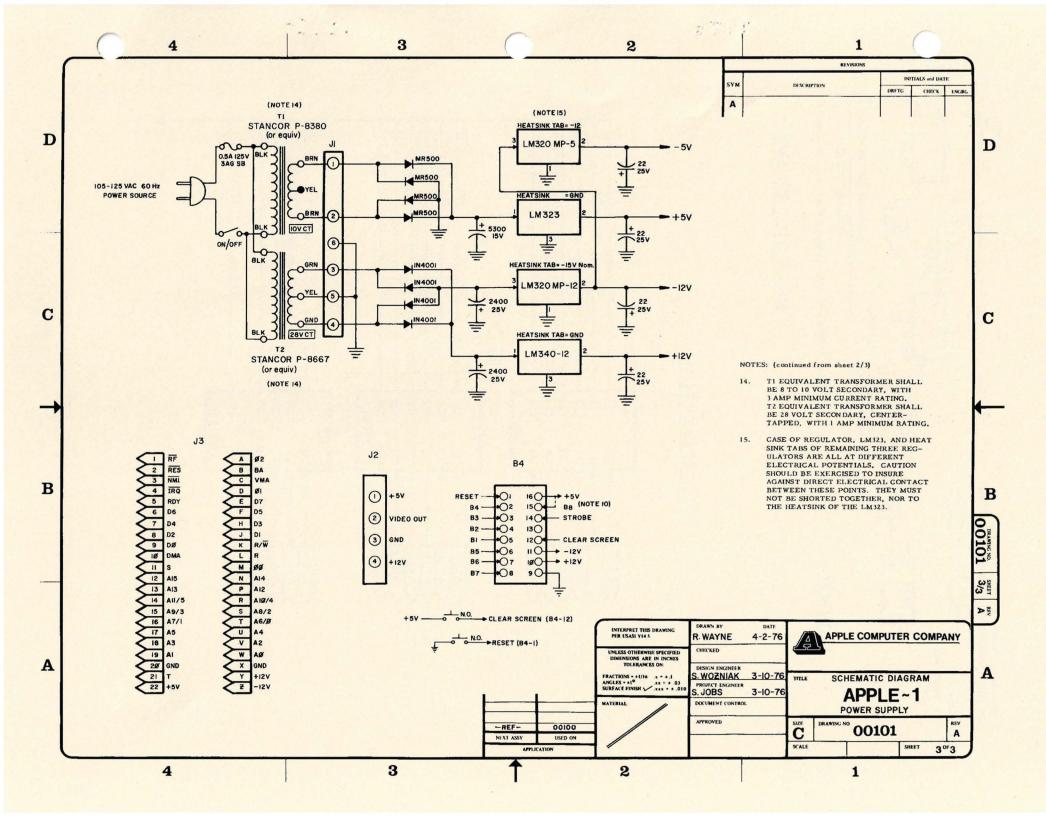
> DSP DATA DØ12 Lower seven bits are data output, high order bit is "display ready" input (lequals ready, Ø equals busy)

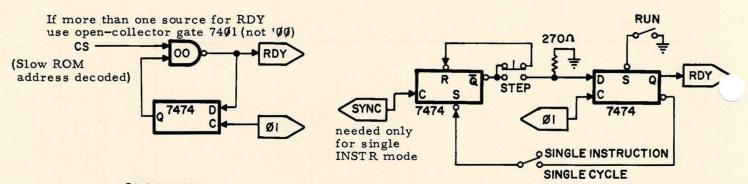
DSP Control Reg. DØ13

-8-







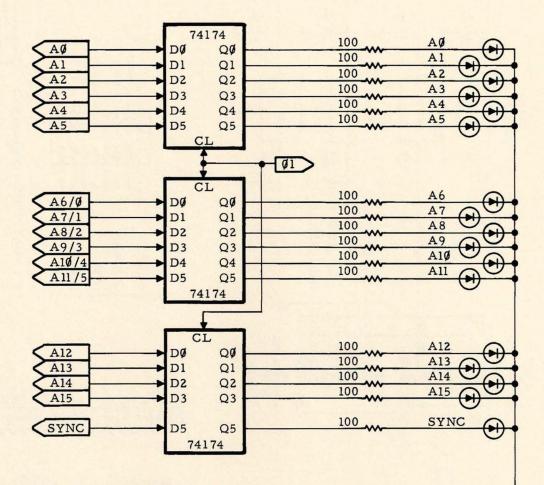


SLOW ROM

(NOTE: Features not needed may be omitted)

SINGLE STEP FOR 6502

ADDRESS DISPLAY



The Apple Computer Company hereby warrants each of its products, and all components therein contained, to be free from defects in materials and/or workmanship for a period of thirty (30) days from date of purchase. In the event of the occurrence of malfunction, or other indication of failure attributable directly to faulty workmanship and/or material, then, upon return of the product to the Apple Computer Company, at 770 Welch Road, Palo Alto, California, 94304 (postage prepaid), the Apple Computer Company will, at its option, repair or replace said products or components thereof, to what ever extent Apple Computer Company shall deem necessary, to restore said product to proper operating condition. All such repairs or replacements shall be rendered by the Apple Computer Company, without charge to the customer.

The responsibility for the failure of any Apple Computer product, or component thereof, which, at the discretion of the Apple Computer Company, shall have resulted either directly or indirectly from accident, abuse, or misapplication of the product, shall be assumed by the customer, and the Apple Computer Company shall assume no liability as a consequence of such events under the terms of this warranty.

While every effort, on the part of Apple Computer Company, is made to provide clear and accurate technical instruction on the use, implementation, and application of its products, the Apple Computer Company shall assume noliability in events which may arise from the application of such technical instruction, nor shall the Apple Computer Company be held liable for the quality, interconnection, or application of periferal products, which may have been recommended by Apple Computer Company, but which have not been supplied as part of the product.

This warranty contains and embodies the limits of responsibility of the Apple Computer Company, with regard to its products, and no other liability is expressed, implied, or should be assumed by the purchaser, and in no event shall the Apple Computer Company be held liable for the loss of time, effort, or transportation costs, nor for loss of potential profits or other consequential losses which might arise from the purchase, assembly, use, application, or subsequent sale of the products of Apple Computer Company, nor from any instructions and/or technical information thereto related.

