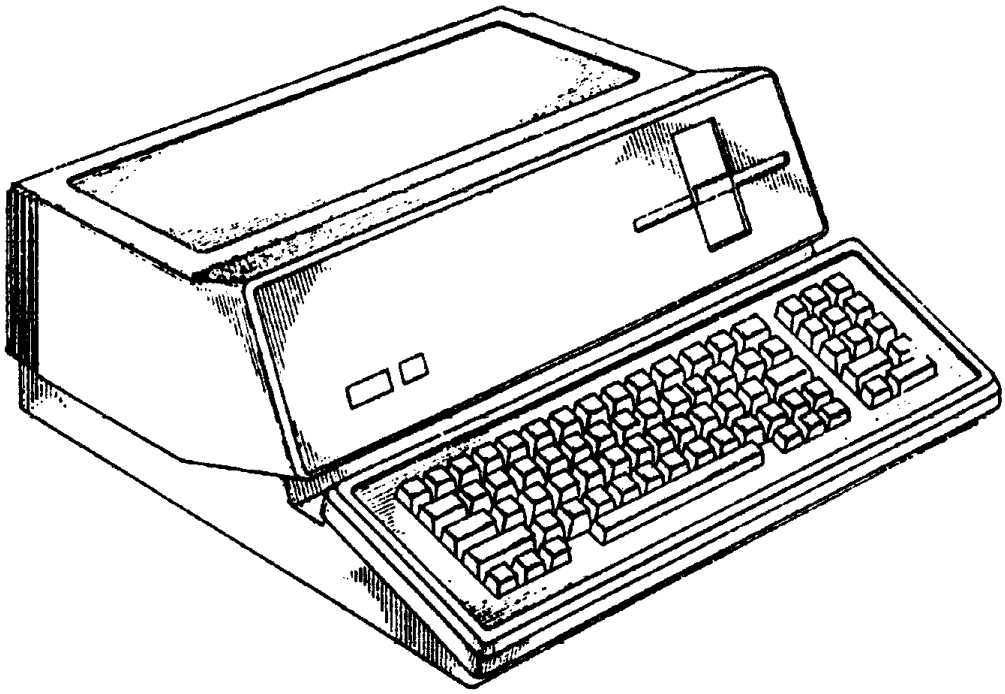
Apple /// Computer Information



APPLE /// ROM INFO

ADDED BY DAVID T CRAIG · 2006

Apple /// ROM Information

APPLE /// ROM INFORMATION

by
David Craig
736 Edgewater, Wichita, Kansas 67230

1986

This document describes the Apple /// microcomputer ROM organization. The ROM listing used was from Apple Computer's patent (# 4,383,296) of May 10, 1983 as assigned to Wendell B. Sander. The ROM listing appears to be from December 20, 1979.

The ROM occupies 4K bytes of memory in the address range $F000—$FFFF. This ROM is used by the Apple /// at system power-up to test various hardware components, initialize the character generator bitmap, and boot SOS (Sophisticated Operating System) from the Apple ///'s internal floppy diskette drive.

The ROM is organized as follows (routine names in lowercase were created by me since the source code did not contain a name at the particular location):

Addresses	Name	Description
F000-F124	REGRWTS	Read/Write a disk track and sector
F125-F12A	SETTRK	Set slot dependent track location
F12B-F13D	CHKDRV	Check if disk motor is stopped
F13E-F147	DRVINDX	Get index to drive number
F148-F1B9	READ16	Read disk sector
F1BA-F1BC	GOSERV	Interrupt service vector
F1BD-F218	RDADR16	Read disk sector address field
F219-F2B2	WRITE16	Write disk sector
F2B3-F2BB	SERVICE	Interrupt servicer
F2BC-F2C5	WNIBL9	Write 7-bit nibbles to disk
F2C6-F310	PRENIB16	Pre-nibblize disk sector data
F311-F354	POSTNIB16	Post-nibblize disk sector data
F355-F395	NIBL	6-bit to 7-bit nibble conversion table
F396-F3FF	DNIBL	7-bit to 6-bit denibbleize conversion table
F400-F455	SEEK	Disk track seeker
F456-F466	MSWAIT	100 microsecond delayer
F467-F46F	ONTABLE	Disk phase ON time table (in 100 microsecs)
F470-F478	OFFTABLE	Disk phase OFF time table (in 100 microsecs)
F479-F49F	BLOCKIO	Read/write a disk block (2 sectors)

Apple /// ROM Information

F4A0-F4A7	SECTABL	Block to sector conversion table
F4A8-F4C4	ANALOG	Joystick read routine
F4C5-F4CC	RAMTBL	RAM test bytes
F4CD-F4ED	CHPG	Hardware component phrases (eg "RAM", "ROM",...)
F4EE-F523	DIAGN	ROM system power-up entry (calls RECON [F689])
F524-F531	NXBYT	Test RAM page 0 (Zero Page)
F532-F545	CNTWR	Test RAM page 1 (Stack Page)
F546-F574	memsize	Size the RAM
F575-F5B9	ERRLP	Display screen error line ("DIAGNOSTICS")
F5BA-F5E6	zpgstktst	Test RAM zero page & stack page
F5E7-F60C	ROMTST	Test ROM hardware
F60D-F63D	VIATST	Test VIA hardware
F63E-F652	ACIA	Test ACIA hardware
F653-F67A	ATD	Test A/D hardware
F67B-F688	KEYPLUG	Test keyboard plugin
F689-F6C1	RECON	Reconfigure system (tests for Apple-1 key)
F6C2-F6E5	SEX	System exerciser
F5E6-F737	USRENTRY	Main RAM tester
F738-F747	STRWT	Error message string writer
F748-F77A	RAM	Determine size of RAM
F77B-F783	MESSERR	Display error message
F784-F7A0	RAMSET	Setup RAM
F7A1-F7C8	PTRINC	Increment extended addressing pointer
F7C9-F7F6	RAMERR	RAM error handler
F7F7-F7FF	RAMWT	RAM write
F800-F900	RET1	Nested RTS 'table' routine
F901-F92B	ENTRY	SARA Monitor entry point
F92C-F95D	GETNUM	Get number from user
F92E-F96B	TOSUB	Execute Monitor command
F96C-F97B	CMDTAB	Monitor command code table
F97C-F98B	CMDVEC	Monitor command vector table (byte-long entries)
F98C-F9AB	NXTA4	Increment 2 byte pointer
F9AC-F9C1	PRBYTE	Output a byte to screen
F9C2-F9C8	PRBYCOL	Output a byte followed by a colon
F9C9-F9D3	TST80WID	Test for 80-column screen width
F9D4-F9DE	A1PC	Test for new P.C.
F9DF-FA06	ASCII1	Store user ASCII string into memory
FA07-FA25	ASCII	Fetch ASCII character from keyboard
FA26-FA2B	CRMON	Dump line of hexadecimal bytes due to user CR
FA2C-FA3A	MOVE	Move bytes around in memory
FA3B-FA51	VRFY	Verify memory byte range
FA52-FA77	MISMATCH	Output verify mismatch data line
FA78-FA7A	USER	User control vector
FA7B-FA82	JUMP	Transfer control to user routine
FA83-FA90	RWERROR	Output error number
FA91-FA99	DEST	Copy source pointer to destination pointer
FA9A-FAB7	SEP	Test for seperator character in input line
FAB8-FABF	SETMODE	Setup user mode
FAC0-FAE8	READ	Handle Monitor READ disk block command
FAE9-FB20	DUMP8	Output line of memory bytes
FB21-FB48	DUMPASC	Output line of memory bytes as ASCII
FB49-FB4E	COL80	Setup 80-column display mode
FB4F-FB92	COL40	Setup 40-column display mode

ENTRY POINT

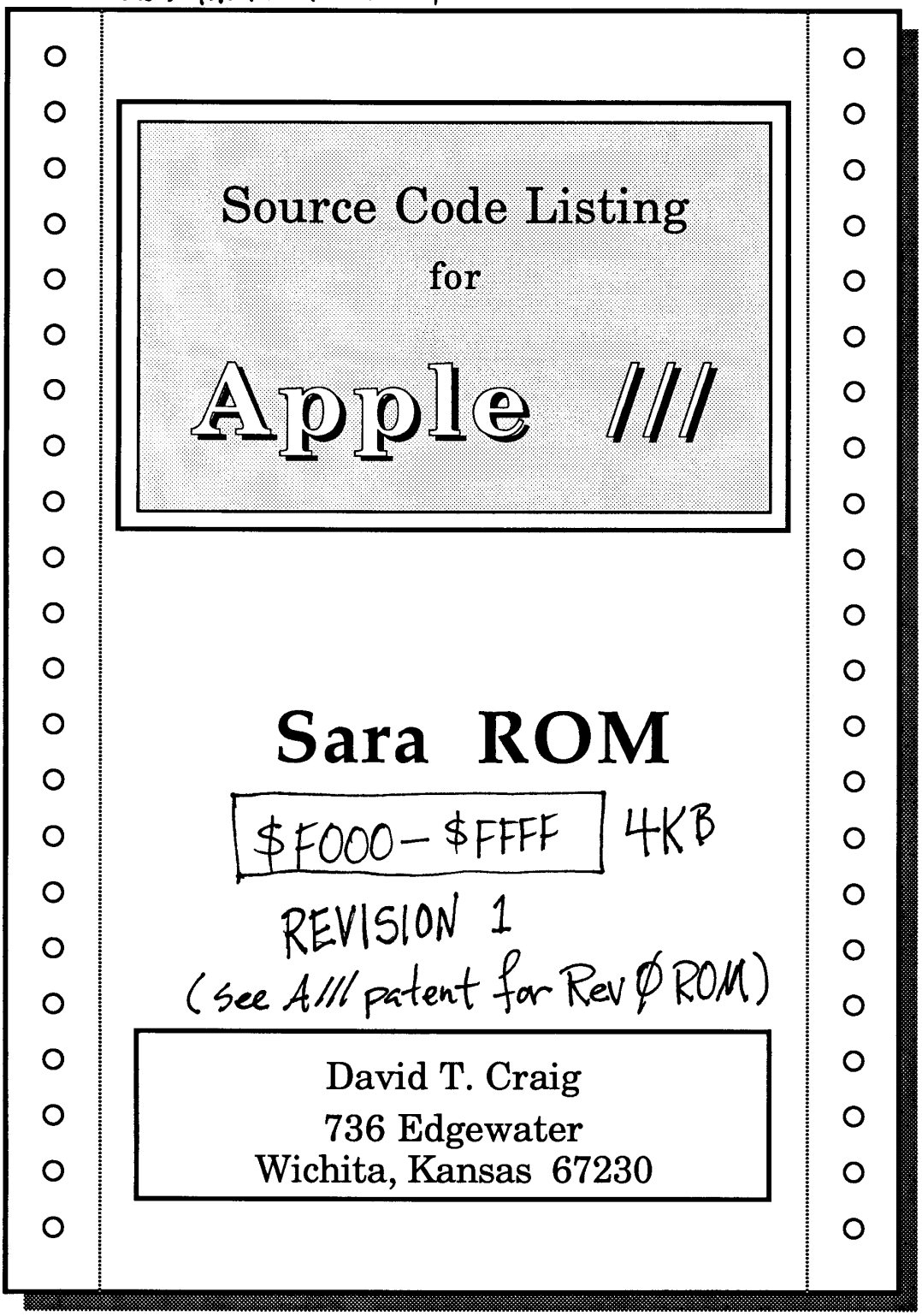
Apple /// ROM Information

FB93-FBA3	CONTROL	Handle user control character input
FBA4-FBB6	CURUP	Handle cursor up motion
FBB7-FBC8	CURIGHT	Handle cursor right motion
FBC9-FBD4	DURDOWN	Handle cursor down motion
FBD5-FBD8	LSTBACK	Handle backspace motion
FBD9-FBF1	CURLEFT	Handle cursor left motion
FBF2-FC04	COUT2	Output character to screen
FC05-FC24	BASCALC1	Compute character base address for screen output
FC25-FC32	COUT	Output character to current output device
FC33-FC35	COUT1	Character output vector
FC36-FC51	TSTBELL	Handle BELL character output (beep speaker)
FC52-FC5A	LNFD	Handle LINE FEED character output
FC5B-FC9C	SCROLL	Scroll screen lines
FC9D-FCAC	DISPLAY	Display character on 40-column screen
FCAD-FCBA	DSPL80	Display character on 80-column screen
FCBB-FCD4	NOTCR	Handle non-control character output
FCD5-FD0B	GETLNZ	Read user ASCII line from keyboard
FD0C-FD0E	RDKEY	Read keyboard key input vector
FD0F-FD47	KEYIN	Read raw keyboard key
FD48-FD5F	ESC3	Handle ESC character cursor motion
FD60-FD76	RDCHAR	Read keyboard character
FD77-FD7E	GOESC	ESC key cursor motion handler
FD7F-FD87	ESCVECT	ESC key editing command key code table
FD88-FD97	PICK	Read character from current cursor location
FD98-FDC5	CLDSTART	Cold boot system (initialize ROM globals)
FDC6-FEAD	GENENTR	Load character generator RAM with bitmap
FEAE-FEC4	VRETRCE	Wait/poll for CRT vertical retrace
FEC5-FFB3	CHRSET	Character generator character bitmap table
FFB4-FFB7	HOOKS	Output/Input vectors
FFB8-FFBB	VBOUNDS	Screen dimension bounds (0,80,0,24)
FFBC-FFBF	NMIIRQ	NMI request vector (JMP RECON [F689] RTI)
FFC0-FFEF	applecwrite	Apple Computer, Inc. 1980 copyright phrase
FFF0-FFF9	ESCTABL	ESC character table
FFFA-FFFB	NMI	NMI vector [FFCA]
FFFC-FFFD	RESET	RESET vector [F4EE] (Power-up Diagnostics)
FFFE-FFFF	IRG	IRG vector [FFCD]

--- The End ---

Assembler: Apple /// Pascal TLA 6502 Assembler
(converted to TLA format most likely by Scott Stinson)

41 pages



Source Code Listing

for

Apple ///

ROM - Disk I/O

David T. Craig
736 Edgewater
Wichita, Kansas 67230

```
ØØØØ|                ;
ØØØØ|                ; APPLE /// ROM - DISK I/O ROUTINES
ØØØØ|                ; COPYRIGHT 1979 BY APPLE COMPUTER, INC.
ØØØØ|                ;
ØØØØ|
ØØØØ|                           .ABSOLUTE
ØØØØ|                           .PROC   DISKIO
ØØØØ|                           .ORG    ØFØØØ
ØØØØ|
FØØØ|                ;*************************
FØØØ|                ;     CRITICAL TIMING     *
FØØØ|                ;   REQUIRES PAGE BOUND   *
FØØØ|                ;   CONSIDERATIONS FOR    *
FØØØ|                ;      CODE AND DATA      *
FØØØ|                ;     -----CODE-----      *
FØØØ|                ;   VIRTUALLY THE ENTIRE  *
FØØØ|                ;     'WRITE' ROUTINE     *
FØØØ|                ;      MUST NOT CROSS     *
FØØØ|                ;     PAGE BOUNDARIES     *
FØØØ|                ;  CRITICAL BRANCHES IN   *
FØØØ|                ;  THE 'WRITE', 'READ',   *
FØØØ|                ;  AND 'READ ADR' SUBRS   *
FØØØ|                ;  WHICH MUST NOT CROSS   *
FØØØ|                ;  PAGE BOUNDARIES ARE    *
FØØØ|                ;  NOTED IN COMMENTS      *
FØØØ|                ;                         *
FØØØ|                ;*************************
FØØØ|                ;                         *
FØØØ|                ;         EQUATES         *
FØØØ|                ;                         *
FØØØ| Ø2ØØ           NBUF1      .EQU    Ø2ØØ
FØØØ| Ø3Ø2           NBUF2      .EQU    Ø3Ø2          ; (ZERO PAGE AT $3ØØ)
FØØØ|                ;
FØØØ| ØØ8Ø           HRDERRS    .EQU    8Ø
FØØØ| ØØEØ           DVMOT      .EQU    ØEØ
FØØØ|                ;
FØØØ| ØØ81           IBSLOT     .EQU    81
FØØØ| ØØ82           IBDRVN     .EQU    IBSLOT+1
FØØØ| ØØ83           IBTRK      .EQU    IBSLOT+2
FØØØ| ØØ84           IBSECT     .EQU    IBSLOT+3
FØØØ| ØØ85           IBBUFP     .EQU    IBSLOT+4      ; & 5
FØØØ| ØØ87           IBCMD      .EQU    IBSLOT+6
FØØØ| ØØ88           IBSTAT     .EQU    IBSLOT+7
FØØØ| ØØ89           IBSMOD     .EQU    IBSLOT+8
FØØØ| ØØ89           CSUM       .EQU    IBSMOD        ; USED ALSO FOR ADDRESS HEADER CKSUM
FØØØ| ØØ8A           IOBPDN     .EQU    IBSLOT+9
FØØØ| ØØ8B           IMASK      .EQU    IBSLOT+ØA
FØØØ| ØØ8C           CURTRK     .EQU    IBSLOT+ØB
FØØØ| ØØ85           DRVOTRK    .EQU    CURTRK-7
FØØØ|                ; SLOT 4, DRIVE 1
FØØØ|                ; SLOT 4, DRIVE 2
FØØØ|                ; SLOT 5, DRIVE 1
FØØØ|                ; SLOT 5, DRIVE 2
FØØØ|                ; SLOT 6, DRIVE 1
FØØØ|                ; SLOT 6, DRIVE 2
FØØØ| ØØ93           RETRYCNT   .EQU    IBSLOT+12
FØØØ| ØØ94           SEEKCNT    .EQU    IBSLOT+13
FØØØ| ØØ9B           BUF        .EQU    IBSLOT+1A
FØØØ| ØØ9F           ENVTEMP    .EQU    IBSLOT+1E
FØØØ|                ; IBSLOT+$1F NOT USED
FØØØ|                ;
FØØØ|                ;*************************
FØØØ|                ;                         *
FØØØ|                ;     ----READADR----     *
FØØØ|                ;                         *
FØØØ|                ;*************************
FØØØ|                ;
FØØØ| ØØ95           COUNT      .EQU    IBSLOT+14     ; 'MUST FIND' COUNT.
FØØØ| ØØ95           LAST       .EQU    IBSLOT+14     ; 'ODD BIT' NIBLS.
FØØØ| ØØ96           CKSUM      .EQU    IBSLOT+15     ; CHECKSUM BYTE.
FØØØ| ØØ97           CSSTV      .EQU    IBSLOT+16     ; FOUR BYTES
FØØØ|                ;          CHECKSUM, SECTOR, TRACK, AND VOLUME.
FØØØ|                ;
FØØØ|                ;*************************
FØØØ|                ;                         *
FØØØ|                ;      ----WRITE----      *
FØØØ|                ;                         *
FØØØ|                ;      USES ALL NBUFS     *
FØØØ|                ;       AND 32-BYTE       *
FØØØ|                ;     DATA TABLE 'NIBL'   *
FØØØ|                ;                         *
FØØØ|                ;*************************
FØØØ|                ;
FØØØ|                ;*************************
FØØØ|                ;                         *
FØØØ|                ;       ----READ----      *
FØØØ|                ;                         *
FØØØ|                ;      USES ALL NBUFS     *
FØØØ|                ;   USES LAST 54 BYTES    *
FØØØ|                ;   OF A CODE PAGE FOR    *
```

```
F000|                       ;   SIGNIFICANT BYTES     *
F000|                       ;   OF DNIBL TABLE.       *
F000|                       ;                         *
F000|                       ;*************************
F000|                       ;
F000|                       ;*************************
F000|                       ;                         *
F000|                       ;     ----SEEK----        *
F000|                       ;                         *
F000|                       ;*************************
F000|                       ;
F000| 0095                  TRKCNT      .EQU    COUNT         ; HALFTRACKS MOVED COUNT.
F000| 009D                  PRIOR       .EQU    IBSLOT+1C
F000| 009E                  TRKN        .EQU    IBSLOT+1D
F000|                       ;
F000|                       ;*************************
F000|                       ;                         *
F000|                       ;    ----MSWAIT----       *
F000|                       ;                         *
F000|                       ;*************************
F000|                       ;
F000| 0099                  MONTIMEL    .EQU    CSSTV+2       ; MOTOR-ON TIME
F000| 009A                  MONTIMEH    .EQU    MONTIMEL+1    ; COUNTERS.
F000|                       ;
F000|                       ;*************************
F000|                       ;                         *
F000|                       ;     DEVICE ADDRESS      *
F000|                       ;      ASSIGNMENTS        *
F000|                       ;                         *
F000|                       ;*************************
F000|                       ;
F000| C080                  PHASEOFF    .EQU    0C080         ; STEPPER PHASE OFF.
F000| C081                  PHASEON     .EQU    0C081         ; STEPPER PHASE ON.
F000| C08C                  Q6L         .EQU    0C08C         ; Q7L,Q6L=READ
F000| C08D                  Q6H         .EQU    0C08D         ; Q7L,Q6H=SENSE WPROT
F000| C08E                  Q7L         .EQU    0C08E         ; Q7H,Q6L=WRITE
F000| C08F                  Q7H         .EQU    0C08F         ; Q7H,Q6H=WRITE STORE
F000| FFEF                  INTERUPT    .EQU    0FFEF
F000| FFDF                  ENVIRON     .EQU    0FFDF
F000| 0080                  ONEMEG      .EQU    80
F000| 007F                  TWOMEG      .EQU    7F
F000|                       ;
F000|                       ;*******************************
F000|                       ;
F000|                       ; EQUATES FOR RWTS AND BLOCK
F000|                       ;
F000|                       ;*******************************
F000|                       ;
F000| C088                  MOTOROFF    .EQU    0C088
F000| C089                  MOTORON     .EQU    0C089
F000| C08A                  DRVOEN      .EQU    0C08A
F000| C08B                  DRV1EN      .EQU    0C08B
F000| C081                  PHASON      .EQU    0C081
F000| C080                  PHSOFF      .EQU    0C080
F000| 0097                  TEMP        .EQU    CSSTV         ; PUT ADDRESS INFO HERE
F000| 0097                  CSUM1       .EQU    TEMP
F000| 0098                  SECT        .EQU    CSUM1+1
F000| 0099                  TRACK       .EQU    SECT+1
F000| 0099                  TRKN1       .EQU    TRACK
F000| 009A                  VOLUME      .EQU    TRACK+1
F000| 0083                  IBRERR      .EQU    HRDERRS+3
F000| 0082                  IBDERR      .EQU    HRDERRS+2
F000| 0081                  IBWPER      .EQU    HRDERRS+1
F000| 0080                  IBNODRV     .EQU    HRDERRS
F000|                       ;
F000|                       ;*******************************
F000|                       ;                               *
F000|                       ;     READ WRITE A              *
F000|                       ;   TRACK AND SECTOR            *
F000|                       ;                               *
F000|                       ;*******************************
F000|                       ;
F000| A0 01                 REGRWTS     LDY     #01           ; RETRY COUNT
F002| A6 81                             LDX     IBSLOT        ; GET SLOT # FOR THIS OPERATION
F004| 84 94                             STY     SEEKCNT       ; ONLY ONE RECALIBRATE PER CALL
F006| A9 05                             LDA     #0005
F008| 85 8F                             STA     08F
F00A| 08                                PHP                   ; DETERMINE INTERRUPT STATUS
F00B| 68                                PLA
F00C| 6A                                ROR     A
F00D| 6A                                ROR     A             ; GET INTERRUPT FLAG INTO BIT 7
F00E| 6A                                ROR     A
F00F| 6A                                ROR     A
F010| 85 8B                             STA     IMASK
F012| AD DFFF                           LDA     ENVIRON       ; PRESERVE ENVIRONMENT
F015| 85 9F                             STA     ENVTEMP
F017| 20 2BF1                          JSR     CHKDRV        ; SET ZERO FLAG IF MOTOR STOPPED
F01A| 08                                PHP                   ; SAVE TEST RESULTS
F01B| A5 85                             LDA     IBBUFP        ; MOVE OUT POINTER TO BUFFER INTO ZPAGE
F01D| 85 9B                             STA     BUF
```

```
FØ1F| A5 86                        LDA     IBBUFP+1
FØ21| 85 9C                        STA     BUF+1
FØ23| A9 EØ                        LDA     #DVMOT
FØ25| 85 9A                        STA     MONTIMEH
FØ27| A5 82                        LDA     IBDRVN          ; DETERMINE DRIVE ONE OR TWO
FØ29| C5 8A                        CMP     IOBPDN          ; SAME DRIVE USED BEFORE
FØ2B| 85 8A                        STA     IOBPDN          ; SAVE IT FOR NEXT TIME
FØ2D| Ø8                          PHP                     ; KEEP RESULTS OF COMPARE
FØ2E| 6A                          ROR     A               ; GET DRIVE NUMBER INTO CARRY
FØ2F| BD 89CØ                     LDA     MOTORON,X       ; TURN ON THE DRIVE
FØ32| 9ØØ1                       BCC     DRIVSEL         ; BRANCH IF DRIVE 1 SELECTED
FØ34| E8                          INX                     ; SELECT DRIVE 2
FØ35| BD 8ACØ              DRIVSEL LDA     DRVOEN,X
FØ38| 2Ø 4CF3                     JSR     SET1MEG         ; INSURE ONE MEGAHERTZ OPERATION
FØ3B| 28                          PLP                     ; WAS IT SAME DRIVE?
FØ3C| FØØA                       BEQ     OK
FØ3E| 28                          PLP                     ; MUST INDICATE DRIVE OFF BY SETTING ZERO FLAG
FØ3F| AØ Ø7                       LDY     #Ø7             ; DELAY 15Ø MS BEFORE STEPPING
FØ41| 2Ø 56F4              DRVWAIT JSR     MSWAIT          ; (ON RETURN A=Ø)
FØ44| 88                          DEY
FØ45| DØFA                       BNE     DRVWAIT
FØ47| Ø8                          PHP                     ; NOW ZERO FLAG SET
FØ48| A5 83                 OK      LDA     IBTRK           ; GET DESTINATION TRACK
FØ4A| A6 81                        LDX     IBSLOT          ; RESTORE PROPER X (SLOT*16)
FØ4C| 2Ø Ø4F1                     JSR     MYSEEK          ; AND GO TO IT
FØ4F|                       ; NOW AT THE DESIRED TRACK   WAS THE MOTOR ON TO START WITH?
FØ4F| 28                          PLP                     ; WAS MOTOR ON?
FØ5Ø| DØ17                       BNE     TRYTRK          ; IF SO, DON'T DELAY, GET IT TODAY!
FØ52|                       ;
FØ52|                       ; MOTOR WAS OFF, WAIT FOR IT TO SPEED UP
FØ52|                       ;
FØ52| AØ 12                 MOTOF   LDY     #12             ; WAIT EXACTLY 1ØØ US FOR EACH COUNT
FØ54| 88                    CONWAIT DEY                     ; IN MONTIME
FØ55| DØFD                       BNE     CONWAIT
FØ57| E6 99                       INC     MONTIMEL        ; COUNT UP TO ØØØØ
FØ59| DØF7                       BNE     MOTOF
FØ5B| E6 9A                       INC     MONTIMEH
FØ5D| 3ØF3                       BMI     MOTOF
FØ5F|                       ;
FØ5F|                       ;*******************************
FØ5F|                       ; MOTOR SHOULD BE UP TO SPEED
FØ5F|                       ; IF IT STILL LOOKS STOPPED THEN
FØ5F|                       ; THE DRIVE IS NOT PRESENT.
FØ5F|                       ;
FØ5F|                       ;*******************************
FØ5F|                       ;
FØ5F| 2Ø 2BF1                     JSR     CHKDRV          ; IS DRIVE PRESENT?
FØ62| DØØ5                       BNE     TRYTRK          ; YES, CONTINUE
FØ64| A9 8Ø              NODRIVERR LDA     #IBNODRV        ; NO, GET TELL EM NO DRIVE
FØ66| 4C EAFØ                    JMP     HNDLERR
FØ69|                       ;
FØ69|                       ; NOW CHECK IF IT IS NOT THE FORMAT DISK COMMAND,
FØ69|                       ;  LOCATE THE CORRECT SECTOR FOR THIS OPERATION
FØ69|                       ;
FØ69| A5 87                 TRYTRK  LDA     IBCMD           ; GET COMMAND CODE #
FØ6B| FØ76                       BEQ     ALLDONE         ; IF NULL COMMAND, GO HOME TO BED
FØ6D| C9 Ø3                       CMP     #Ø3             ; COMMAND IN RANGE?
FØ6F| BØ72                       BCS     ALLDONE         ; NO, DO NOTHING!
FØ71| 6A                          ROR     A               ; SET CARRY=1 FOR READ, Ø FOR WRITE
FØ72| BØØB                       BCS     TRYTRK2         ; MUST PRENIBBLIZE FOR WRITE
FØ74| AD DFFF                     LDA     ENVIRON
FØ77| 29 7F                       AND     #TWOMEG         ; SHIFT TO HIGH SPEED!
FØ79| 8D DFFF                     STA     ENVIRON
FØ7C| 2Ø C4F2                     JSR     PRENIB16
FØ7F| AØ 7F                 TRYTRK2 LDY     #7F             ; ONLY 127 RETRIES OF ANY KIND
FØ81| 84 93                       STY     RETRYCNT
FØ83| A6 81                 TRYADR  LDX     IBSLOT          ; GET SLOT NUM INTO X-REG
FØ85| 2Ø B9F1                     JSR     RDADR16         ; READ NEXT ADDRESS FIELD
FØ88| 9Ø22                       BCC     RDRIGHT         ; IF READ IS RIGHT, HURRAH!
FØ8A| 2Ø AAF1              TRYADR2 JSR     CHKINT          ; BRANCH TO CHECK FOR INTERRUPTS
FØ8D| C6 93                       DEC     RETRYCNT        ; ANOTHER MISTAKE!!
FØ8F| 1ØF2                       BPL     TRYADR          ; WELL, LET IT GO THIS TIME
FØ91| C6 94                       DEC     SEEKCNT         ; ONLY RECALIBRATE ONCE!
FØ93| DØ53                       BNE     DRVERR          ; TRIED TO RECALIBRATE A SECOND TIME, ERROR!
FØ95| A5 8F                       LDA     Ø8F             ; ANOTHER MISTAKE!!
FØ97| 3ØEA                       BMI     TRYADR          ; WELL, LET IT GO THIS TIME
FØ99| A5 8C                       LDA     CURTRK
FØ9B| 48                          PHA                     ; SAVE TRACK WE REALLY WANT
FØ9C| A9 6Ø                       LDA     #6Ø             ; RECALIBRATE ALL OVER AGAIN!   ERROR!
FØ9E| 2Ø 25F1                     JSR     SETTRK          ; PRETEND TO BE ON TRACK 8Ø
FØA1| A9 ØØ                       LDA     #ØØ
FØA3| 2Ø Ø4F1                     JSR     MYSEEK          ; MOVE TO TRACK ØØ
FØA6| 68                    GOCAL1  PLA
FØA7| 2Ø Ø4F1              GOCAL   JSR     MYSEEK          ; GO TO CORRECT TRACK THIS TIME!
FØAA| 9ØD7                       BCC     TRYADR          ; LOOP BACK, TRY AGAIN ON THIS TRACK
FØAC|                       ;
FØAC|                       ; HAVE NOW READ AN ADDRESS FIELD CORRECTLY.
FØAC|                       ; MAKE SURE THIS IS THE TRACK, SECTOR, AND VOLUME DESIRED.
FØAC|                       ;
FØAC| A4 99                 RDRIGHT LDY     TRACK           ; ON THE RIGHT TRACK?
```

CMD
1 -> Read
2 -> Write

```
FØAE| C4 8C                        CPY     CURTRK
FØBØ| FØØE                        BEQ     RTTRK        ; IF SO, GOOD
FØB2|                    ;
FØB2|                    ; RECALIBRATING FROM THIS TRACK
FØB2|                    ;
FØB2| A5 8C                        LDA     CURTRK       ; PRESERVE DESTINATION TRACK
FØB4| 48                          PHA
FØB5| 98                          TYA
FØB6| ØA                          ASL     A
FØB7| 2Ø 25F1                      JSR     SETTRK
FØBA| 68                          PLA
FØBB| 2Ø Ø4F1                      JSR     MYSEEK
FØBE| 9ØCA                        BCC     TRYADR2
FØCØ| A5 9A              RTTRK     LDA     VOLUME       ; GET ACTUAL VOLUME HERE
FØC2| 85 89                        STA     IBSMOD       ; TELL OPSYS WHAT VOLUME WAS THERE
FØC4| A5 98              CORRECTVOL LDA    SECT         ; CHECK IF THIS IS THE RIGHT SECTOR
FØC6| C5 84                        CMP     IBSECT
FØC8| DØCØ                        BNE     TRYADR2      ; NO, TRY ANOTHER SECTOR
FØCA| A5 87                        LDA     IBCMD        ; READ OR WRITE?
FØCC| 4A                          LSR     A            ; THE CARRY WILL TELL
FØCD| 9Ø2A                        BCC     WRIT         ; CARRY WAS SET FOR READ OPERATION,
FØCF| 2Ø 48F1                      JSR     READ16       ; CLEARED FOR WRITE
FØD2| BØB6                        BCS     TRYADR2      ; CARRY SET UPON RETURN IF BAD READ
FØD4| AD DFFF                      LDA     ENVIRON
FØD7| 29 7F                       AND     #TWOMEG
FØD9| 8D DFFF                      STA     ENVIRON      ; SET TWO MEGAHERTZ
FØDC| 2Ø ØFF3                      JSR     POSTNIB16    ; DO PARTIAL POSTNIBBLE CONVERSION
FØDF| A6 81                        LDX     IBSLOT       ; RESTORE SLOTNUM INTO X
FØE1| BØA7                        BCS     TRYADR2      ; CHECKSUM ERROR
FØE3| 18                  ALLDONE   CLC
FØE4| A9 ØØ                        LDA     #ØØ          ; NO ERROR
FØE6| 9ØØ3                        BCC     ALDONE1      ; SKIP OVER NEXT BYTE WITH BIT OPCODE
FØE8| A9 82              DRVERR    LDA     #IBDERR      ; BAD DRIVE
FØEA| 38                  HNDLERR   SEC                  ; INDICATE AN ERROR
FØEB| 85 88              ALDONE1   STA     IBSTAT       ; GIVE HIM ERROR
FØED| BD 88CØ                      LDA     MOTOROFF,X   ; TURN IT OFF
FØFØ| 2Ø AAF1                      JSR     CHKINT       ; BRANCH TO CHECK FOR INTERRUPTS
FØF3| A5 9F                       LDA     ENVTEMP      ; RESTORE ORIGINAL ENVIRONMENT
FØF5| 8D DFFF                      STA     ENVIRON
FØF8| 6Ø                          RTS
FØF9|
FØF9| 2Ø 16F2              WRIT      JSR     WRITE16      ; WRITE NYBBLES NOW
FØFC| 9ØE5                        BCC     ALLDONE      ; IF NO ERRORS
FØFE| A9 81                        LDA     #IBWPER      ; DISK IS WRITE PROTECTED!!
F1ØØ| 5ØE8                        BVC     HNDLERR      ; TAKEN IF TRUELY WRITE PROTECT ERROR
F1Ø2| DØ86                        BNE     TRYADR2      ; OTHERWISE ASSUME AN INTERRUPT MESSED THINGS UP
F1Ø4|                    ;
F1Ø4|                    ; THIS IS THE 'SEEK' ROUTINE
F1Ø4|                    ;  SEEKS TRACK 'N' IN SLOT #X/$1Ø
F1Ø4|                    ; IF DRIVENO IS NEGATIVE, ON DRIVE Ø
F1Ø4|                    ; IF DRIVENO IS POSITIVE, ON DRIVE 1
F1Ø4|                    ;
F1Ø4| ØA                  MYSEEK    ASL     A            ; ASSUME TWO PHASE STEPPER.
F1Ø5| 85 99               SEEK1     STA     TRKN1        ; SAVE DESTINATION TRACK(*2)
F1Ø7| 2Ø 18F1                      JSR     ALLOFF       ; TURN ALL PHASES OFF TO BE SURE.
F1ØA| 2Ø 3EF1                      JSR     DRVINDX      ; GET INDEX TO PREVIOUS TRACK FOR CURRENT DRIVE
F1ØD| B5 85                        LDA     DRVOTRK,X
F1ØF| 85 8C                       STA     CURTRK       ; THIS IS WHERE I AM
F111| A5 99                       LDA     TRKN1        ; AND WHERE I'M GOING TO
F113| 95 85                       STA     DRVOTRK,X
F115| 2Ø ØØF4              GOSEEK    JSR     SEEK         ; GO THERE!
F118| AØ Ø3                ALLOFF    LDY     #Ø3          ; TURN OFF ALL PHASES BEFORE RETURNING
F11A| 98                  NXOFF     TYA                  ; (SEND PHASE IN ACC.)
F11B| 2Ø 4AF4                      JSR     CLRPHASE     ; CARRY IS CLEAR, PHASES SHOULD BE TURNED OFF
F11E| 88                          DEY
F11F| 1ØF9                        BPL     NXOFF
F121| 46 8C                       LSR     CURTRK       ; DIVIDE BACK NOW
F123| 18                          CLC
F124| 6Ø                          RTS
F125|                    ;
F125|                    ; THIS SUBROUTINE SETS THE SLOT DEPENDENT TRACK
F125|                    ; LOCATION
F125|                    ;
F125| 2Ø 3EF1              SETTRK    JSR     DRVINDX      ; GET INDEX TO DRIVE NUMBER
F128| 95 85                       STA     DRVOTRK,X
F12A| 6Ø                          RTS
F12B|                    ;
F12B|                    ;*****************************
F12B|                    ;
F12B|                    ; SUBR TO TELL IF MOTOR IS STOPPED
F12B|                    ;
F12B|                    ; IF MOTOR IS STOPPED, CONTROLLER'S
F12B|                    ; SHIFT REG WILL NOT BE CHANGING.
F12B|                    ;
F12B|                    ; RETURN Y=Ø AND ZERO FLAG SET IF IT IS STOPPED.
F12B|                    ;
F12B|                    ;*****************************
F12B|                    ;
F12B| AØ ØØ              CHKDRV    LDY     #ØØ          ; INIT LOOP COUNTER
F12D| BD 8CCØ              CHKDRV1   LDA     Q6L,X        ; READ THE SHIFT REG
```

```
F13Ø| 2Ø 3DF1                    JSR     CKDRTS        ; DELAY
F133| 48                        PHA
F134| 68                        PLA
F135| DD 8CCØ                  CMP     Q6L,X         ; HAS SHIFT REG CHANGED?
F138| DØØ3                     BNE     CKDRTS        ; YES, MOTOR IS MOVING
F13A| 88                        DEY                   ; NO, DEC RETRY COUNTER
F13B| DØFØ                     BNE     CHKDRV1       ; AND TRY 256 TIMES
F13D| 6Ø                  CKDRTS  RTS                   ; THEN RETURN
F13E|                      ;
F13E| 48                  DRVINDX PHA                   ; PRESERVE ACC.
F13F| 8A                        TXA                   ; GET SLOT(*$1Ø)/8
F14Ø| 4A                        LSR     A
F141| 4A                        LSR     A
F142| 4A                        LSR     A
F143| Ø5 82                     ORA     IBDRVN        ; FOR DRIVE Ø OR 1
F145| AA                        TAX                   ; INTO X FOR INDEX TO TABLE
F146| 68                        PLA                   ; RESTORE ACC.
F147| 6Ø                        RTS
F148|                      ;
F148|                      ;*****************************
F148|                      ;
F148|                      ; NOTE: FORMATTING ROUTINES
F148|                      ;       NOTE INCLUDED FOR SOS
F148|                      ;
F148|                      ;*****************************
F148|                      ;
F148|                      ;*************************
F148|                      ;                       *
F148|                      ;     READ SUBROUTINE   *
F148|                      ;   (16-SECTOR FORMAT)  *
F148|                      ;                       *
F148|                      ;*************************
F148|                      ;                       *
F148|                      ;   READS ENCODED BYTES *
F148|                      ;  INTO NBUF1 AND NBUF2 *
F148|                      ;                       *
F148|                      ;  FIRST READS NBUF2    *
F148|                      ;           HIGH TO LOW,*
F148|                      ;  THEN READS NBUF1     *
F148|                      ;           LOW TO HIGH.*
F148|                      ;                       *
F148|                      ;   ---- ON ENTRY ----  *
F148|                      ;                       *
F148|                      ;  X-REG: SLOTNUM       *
F148|                      ;         TIMES $1Ø.    *
F148|                      ;                       *
F148|                      ;  READ MODE (Q6L, Q7L  *
F148|                      ;                       *
F148|                      ;   ---- ON EXIT ----   *
F148|                      ;                       *
F148|                      ;  CARRY SET IF ERROR   *
F148|                      ;                       *
F148|                      ;  IF NO ERROR:         *
F148|                      ;     A-REG HOLDS $AA.  *
F148|                      ;     X-REG UNCHANGED.  *
F148|                      ;     Y-REG HOLDS $ØØ.  *
F148|                      ;     CARRY CLEAR.      *
F148|                      ;   ---- CAUTION -----  *
F148|                      ;                       *
F148|                      ;        OBSERVE        *
F148|                      ;    'NO PAGE CROSS'    *
F148|                      ;      WARNINGS ON      *
F148|                      ;    SOME BRANCHES!!    *
F148|                      ;                       *
F148|                      ;   ---- ASSUMES ----   *
F148|                      ;                       *
F148|                      ;   1 USEC CYCLE TIME   *
F148|                      ;                       *
F148|                      ;*************************
F148|
F148| AØ 2Ø                READ16  LDY     #2Ø           ; 'MUST FIND' COUNT.
F14A| 88                   RSYNC   DEY                   ; IF CAN'T FIND MARKS.
F14B| FØ6A                        BEQ     RDERR         ; THEN EXIT WITH CARRY SET
F14D| BD 8CCØ                RD1     LDA     Q6L,X         ; READ NIBL.
F15Ø| 1ØFB                        BPL     RD1           ; *** NO PAGE CROSS! ***
F152| 49 D5                 RSYNC1  EOR     #ØD5          ; DATA MARK1?
F154| DØF4                        BNE     RSYNC         ; LOOP IF NOT.
F156| EA                          NOP                   ; DELAY BETWEEN NIBLS.
F157| BD 8CCØ                RD2     LDA     Q6L,X
F15A| 1ØFB                        BPL     RD2           ; *** NO PAGE CROSS! ***
F15C| C9 AA                      CMP     #ØAA          ; DATA MARK 2?
F15E| DØF2                      BNE     RSYNC1        ; (IF NOT, IS IT DM1?)
F16Ø| AØ 55                     LDY     #Ø55          ; INIT NBUF2 INDEX.
F162|                      ;                       ( ADDED NIBL DELAY)
F162| EA                          NOP                   ; DELAY BETWEEN NIBLS.
F163| BD 8CCØ                RD3     LDA     Q6L,X
F166| 1ØFB                        BPL     RD3           ; *** NO PAGE CROSS! ***
F168| C9 AD                      CMP     #ØAD          ; DATA MARK 3?
F16A| DØE6                      BNE     RSYNC1        ; (IF NOT, IS IT DM1?)
F16C|                      ;              (CARRY SET IF DM3!)
```

← Seems like "Note" should be "Not"

```
F16C| EA                        NOP                 ; DELAY BETWEEN NIBLS.
F16D| EA                        NOP                 ; DELAY BETWEEN NIBLS.
F16E| BD 8CCØ          RD4      LDA     Q6L,X
F171| 1ØFB                      BPL     RD4         ; *** NO PAGE CROSS! ***
F173| 99 Ø2Ø3                   STA     NBUF2,Y     ; STORE BYTES DIRECTLY
F176| AD EFFF                   LDA     INTERUPT    ; POLL INTERRUPT LINE
F179| Ø5 8B                     ORA     IMASK       ; (THIS MAY BE USED TO INVALIDATE POLL)
F17B| 1Ø37                      BPL     GOSERV
F17D| 88                        DEY                 ; INDEX TO NEXT
F17E| 1ØEE                      BPL     RD4
F18Ø| C8                RD5     INY                 ; (FIRST TIME Y=Ø)
F181| BD 8CCØ          RD5A     LDA     Q6L,X       ; GET ENCODED BYTES OF NBUF1
F184| 1ØFB                      BPL     RD5A
F186| 99 ØØØ2                   STA     NBUF1,Y
F189| AD EFFF                   LDA     INTERUPT    ; POLL INTERRUPT LINE
F18C| Ø5 8B                     ORA     IMASK       ; (THIS MAY BE USED TO INVALIDATE POLL)
F18E| 1Ø24                      BPL     GOSERV
F19Ø| CØ E4                     CPY     #ØE4        ; WITHIN 1 MS OF COMPLETION?
F192| DØEC                      BNE     RD5
F194| C8                        INY
F195| BD 8CCØ          RD6      LDA     Q6L,X       ; NO POLL FROM NOW ON
F198| 1ØFB                      BPL     RD6
F19A| 99 ØØØ2                   STA     NBUF1,Y
F19D| C8                        INY                 ; FINISH OUT NBUF1 PAGE
F19E| DØF5                      BNE     RD6
F1AØ| BD 8CCØ          RDCKSUM  LDA     Q6L,X       ; GET CHECKSUM BYTE.
F1A3| 1ØFB                      BPL     RDCKSUM
F1A5| 85 96                     STA     CKSUM
F1A7| 2Ø Ø1F2                   JSR     RDA6        ; CHECK BIT SLIP MARKS
F1AA|                   ;
F1AA|                   ; CHECK FOR INTERRUPTS
F1AA|                   ;
F1AA| 24 8B             CHKINT  BIT     IMASK       ; SHOULD INTERRUPTS BE ALLOWED?
F1AC| 1ØØ4                      BPL     $Ø1Ø        ; YES, ALLOW THEM.
F1AE| 24 8F                     BIT     Ø8F
F1BØ| 1ØØ1                      BPL     $Ø2Ø
F1B2| 58                $Ø1Ø    CLI
F1B3| 6Ø                $Ø2Ø    RTS
F1B4|
F1B4| 2Ø AAF2           GOSERV  JSR     SERVICE     ; GO TO SERVICE INTERRUPT
F1B7| 38                RDERR   SEC
F1B8| 6Ø                        RTS
F1B9|                   ;
F1B9|                   ;***************************
F1B9|                   ;                          *
F1B9|                   ;    READ ADDRESS FIELD    *
F1B9|                   ;        SUBROUTINE        *
F1B9|                   ;   (16-SECTOR FORMAT)     *
F1B9|                   ;                          *
F1B9|                   ;***************************
F1B9|                   ;                          *
F1B9|                   ;    READS VOLUME, TRACK   *
F1B9|                   ;        AND SECTOR        *
F1B9|                   ;                          *
F1B9|                   ;   ---- ON ENTRY ----     *
F1B9|                   ;                          *
F1B9|                   ;  XREG: SLOTNUM TIMES $1Ø *
F1B9|                   ;                          *
F1B9|                   ;  READ MODE (Q6L, Q7L)    *
F1B9|                   ;                          *
F1B9|                   ;   ---- ON EXIT ----      *
F1B9|                   ;                          *
F1B9|                   ;  CARRY SET IF ERROR      *
F1B9|                   ;                          *
F1B9|                   ;  IF NO ERROR:            *
F1B9|                   ;    A-REG HOLDS $AA.      *
F1B9|                   ;    Y-REG HOLDS $ØØ.      *
F1B9|                   ;    X-REG UNCHANGED.      *
F1B9|                   ;    CARRY CLEAR.          *
F1B9|                   ;                          *
F1B9|                   ;    CSSTV HOLDS CHKSUM,   *
F1B9|                   ;      SECTOR, TRACK, AND  *
F1B9|                   ;      VOLUME READ.        *
F1B9|                   ;                          *
F1B9|                   ;    USES TEMPS COUNT,     *
F1B9|                   ;      LAST, CSUM, AND     *
F1B9|                   ;      4 BYTES AT CSSTV.   *
F1B9|                   ;                          *
F1B9|                   ;    ---- EXPECTS ----     *
F1B9|                   ;                          *
F1B9|                   ;   ORIGINAL 1Ø-SECTOR     *
F1B9|                   ;  NORMAL DENSITY NIBLS    *
F1B9|                   ;   (4-BIT), ODD BITS,     *
F1B9|                   ;   THEN EVEN              *
F1B9|                   ;                          *
F1B9|                   ;    ---- CAUTION -----    *
F1B9|                   ;                          *
F1B9|                   ;        OBSERVE           *
F1B9|                   ;     'NO PAGE CROSS'      *
F1B9|                   ;       WARNINGS ON        *
```

```
F1B9|                     ;     SOME BRANCHES!!      *
F1B9|                     ;                          *
F1B9|                     ;     ---- ASSUMES ----    *
F1B9|                     ;                          *
F1B9|                     ;     1 USEC CYCLE TIME    *
F1B9|                     ;                          *
F1B9|                     ;**************************
F1B9|
F1B9| AØ FC               RDADR16    LDY     #ØFC
F1BB| 84 95                          STY     COUNT        ; 'MUST FIND' COUNT.
F1BD| C8                  RDASYN     INY
F1BE| DØØ4                          BNE     RDA1         ; LOW ORDER OF COUNT
F1CØ| E6 95                          INC     COUNT        ; (2K NIBLS TO FIND
F1C2| FØF3                          BEQ     RDERR        ; ADR MARK, ELSE ERR)
F1C4| BD 8CCØ              RDA1       LDA     Q6L,X        ; READ NIBL.
F1C7| 1ØFB                          BPL     RDA1         ; *** NO PAGE CROSS! ***
F1C9| C9 D5                RDASN1     CMP     #ØD5         ; ADR MARK 1?
F1CB| DØFØ                          BNE     RDASYN       ; (LOOP IF NOT)
F1CD| EA                            NOP                  ; ADDED NIBL DELAY
F1CE| BD 8CCØ              RDA2       LDA     Q6L,X
F1D1| 1ØFB                          BPL     RDA2         ; *** NO PAGE CROSS! ***
F1D3| C9 AA                          CMP     #ØAA         ; ADR MARK 2?
F1D5| DØF2                          BNE     RDASN1       ; (IF NOT, IS IT AM1?)
F1D7| AØ Ø3                         LDY     #Ø3          ; INDEX FOR 4-BYTE READ
F1D9|                     ;                    (ADDED NIBL DELAY)
F1D9| BD 8CCØ              RDA3       LDA     Q6L,X
F1DC| 1ØFB                          BPL     RDA3         ; *** NO PAGE CROSS! ***
F1DE| C9 96                          CMP     #96          ; ADR MARK 3?
F1EØ| DØE7                          BNE     RDASN1       ; (IF NOT IS IT AM1?)
F1E2|                     ;              (LEAVES CARRY SET!)
F1E2| 78                            SEI                  ; DISABLE INTERRUPT SYSTEM
F1E3| A9 ØØ                         LDA     #ØØ          ; INIT CHECKSUM
F1E5| 85 89                RDAFLD     STA     CSUM
F1E7| BD 8CCØ              RDA4       LDA     Q6L,X        ; READ 'ODD BIT' NIBBL
F1EA| 1ØFB                          BPL     RDA4         ; *** NO PAGE CROSS! ***
F1EC| 2A                            ROL     A            ; ALIGN ODD BITS,  1' INTO LSB
F1ED| 85 95                          STA     LAST         ; (SAVE THEM)
F1EF| BD 8CCØ              RDA5       LDA     Q6L,X        ; READ 'EVEN BIT' NIBL
F1F2| 1ØFB                          BPL     RDA5         ; *** NO PAGE CROSS ***
F1F4| 25 95                          AND     LAST         ; MERGE ODD AND EVEN BITS
F1F6| 99 97 ØØ                      STA     CSSTV,Y      ; STORE DATA BYTE
F1F9| 45 89                          EOR     CSUM
F1FB| 88                            DEY
F1FC| 1ØE7                          BPL     RDAFLD       ; LOOP ON 4 DATA BYTES.
F1FE| A8                            TAY                  ; IF FINAL CHECKSUM
F1FF| DØB6                          BNE     RDERR        ; NONZERO, THEN ERROR
F2Ø1| BD 8CCØ              RDA6       LDA     Q6L,X        ; FIRST BIT SLIP NIBBL
F2Ø4| 1ØFB                          BPL     RDA6         ; *** NO PAGE CROSS! ***
F2Ø6| C9 DE                          CMP     #ØDE
F2Ø8| DØAD                          BNE     RDERR        ; ERROR IF NONMATCH
F2ØA| EA                            NOP                  ; DELAY
F2ØB| BD 8CCØ              RDA7       LDA     Q6L,X        ; SECOND BIT-SLIP NIBL
F2ØE| 1ØFB                          BPL     RDA7         ; *** NO PAGE CROSS! ***
F21Ø| C9 AA                          CMP     #ØAA
F212| DØA3                          BNE     RDERR        ; ERROR IF NOMATCH
F214| 18                  RDEXIT     CLC                  ; CLEAR CARRY ON
F215| 6Ø                  WEXIT      RTS                  ; NORMAL READ EXITS.
F216|                     ;
F216|                     ;**********************
F216|                     ;                      *
F216|                     ;      WRITE SUBR      *
F216|                     ;  (16-SECTOR FORMAT)  *
F216|                     ;                      *
F216|                     ;**********************
F216|                     ;                      *
F216|                     ;   WRITES DATA FROM   *
F216|                     ;    NBUF1 AND NBUF2   *
F216|                     ;                      *
F216|                     ;  FIRST NBUF2,        *
F216|                     ;      HIGH TO LOW.    *
F216|                     ;  THEN NBUF1,         *
F216|                     ;      LOW TO HIGH     *
F216|                     ;                      *
F216|                     ;  ---- ON ENTRY ----  *
F216|                     ;                      *
F216|                     ;   X-REG  SLOTNUM     *
F216|                     ;        TIMES $1Ø     *
F216|                     ;                      *
F216|                     ;                      *
F216|                     ;  ---- ON EXIT ----   *
F216|                     ;                      *
F216|                     ;  CARRY SET IF ERROR. *
F216|                     ;   (W PROT VIOLATION) *
F216|                     ;                      *
F216|                     ;  IF NO ERROR:        *
F216|                     ;                      *
F216|                     ;    A-REG UNCERTAIN.  *
F216|                     ;    X-REG UNCHANGED.  *
F216|                     ;    Y-REG HOLDS $ØØ.  *
F216|                     ;    CARRY CLEAR.      *
```

```
F216|                    ;                       *
F216|                    ;  ---- ASSUMES ----    *
F216|                    ;                       *
F216|                    ;  1 USEC CYCLE TIME    *
F216|                    ;                       *
F216|                    ;************************
F216|                    ;
F216| 38                 WRITE16    SEC                 ; ANTICIPATE WPROT ERR.
F217| B8                            CLV                 ; TO INDICATE WRITE PROTECT ERROR INSTEAD OF
F218|                                                    ; INTERRUPT
F218| BD 8DCØ                       LDA     Q6H,X
F21B| BD 8ECØ                       LDA     Q7L,X       ; SENSE WPROT FLAG.
F21E| 30F5                         BMI     WEXIT       ; BRANCH IF WRITE PROTECTED
F22Ø| A9 FF              WRIT1      LDA     #ØFF        ; SYNC DATA.
F222| 9D 8FCØ                       STA     Q7H,X       ; (5) GOTO WRITE MODE
F225| 1D 8CCØ                       ORA     Q6L,X       ; (4)
F228| AØ Ø4                         LDY     #Ø4         ; (2) FOR FIVE NIBLS.
F22A| EA                           NOP                 ; (2)
F22B| 48                           PHA                 ; (4)
F22C| 68                           PLA                 ; (3)
F22D| 48                 WSYNC      PHA                 ; (4) EXACT TIMING
F22E| 68                           PLA                 ; (3)
F22F| 2Ø BBF2                      JSR     WNIBL7      ; (13,9,6) WRITE SYNC
F232| 88                           DEY                 ; (2)
F233| DØF8                         BNE     WSYNC       ; (2*)  MUST NOT CROSS PAGE!
F235| A9 D5                       LDA     #ØD5        ; (2)  1ST DATA MARK
F237| 2Ø BAF2                      JSR     WNIBL9      ; (15,9,6)
F23A| A9 AA                       LDA     #ØAA        ; (2)  2ND DATA MARK
F23C| 2Ø BAF2                      JSR     WNIBL9      ; (15,9,6)
F23F| A9 AD                       LDA     #ØAD        ; (2)  3RD DATA MARK
F241| 2Ø BAF2                      JSR     WNIBL9      ; (15,9,6)
F244| AØ 55                       LDY     #55         ; (2) NBUF2 INDEX
F246| EA                           NOP                 ; (2) FOR TIMING
F247| EA                           NOP                 ; (2)
F248| EA                           NOP                 ; (2)
F249| DØØ8                        BNE     VRYFRST     ; (3) BRANCH ALWAYS
F24B| AD EFFF            WINTRPT    LDA     INTERUPT    ; (4) POLL INTERRUPT LINE
F24E| Ø5 8B                       ORA     IMASK       ; (3)
F25Ø| 38                           SEC                 ; (2)
F251| 1Ø57                        BPL     SERVICE     ; (2) BRANCH IF INTERRUPT HAS OCCURED
F253| 3ØØØ              VRYFRST    BMI     WRTFRST     ; (3) FOR TIMING.
F255| B9 Ø2Ø3            WRTFRST    LDA     NBUF2,Y     ; (4)
F258| 9D 8DCØ                       STA     Q6H,X       ; (5) STORE ENCODED BYTE
F25B| BD 8CCØ                       LDA     Q6L,X       ; (4) TIME MUST = 32 US PER BYTE!
F25E| 88                           DEY                 ; (2)
F25F| 1ØEA                        BPL     WINTRPT     ; (3) (2 IF BRANCH NOT TAKEN)
F261| 98                           TYA                 ; (2) INSURE NO INTERRUPT THIS BYTE
F262| 3ØØ3                        BMI     WMIDLE      ; (3) BRANCH ALWAYS.
F264| AD EFFF            WNTRPT1    LDA     INTERUPT    ; (4) POLL INTERRUPT LINE
F267| Ø5 8B              WMIDLE     ORA     IMASK       ; (3)
F269| 38                           SEC                 ; (2)
F26A| 3ØØ2                        BMI     WDATA2      ; (3) BRANCH IF NO INTERRUPT
F26C| 1Ø3C                        BPL     SERVICE     ; GO SERVICE INTERRUPT.
F26E| C8                 WDATA2     INY                 ; (2)
F26F| B9 ØØØ2                     LDA     NBUF1,Y     ; (4)
F272| 9D 8DCØ                       STA     Q6H,X       ; (5) STORE ENCODED BYTE
F275| BD 8CCØ                       LDA     Q6L,X       ; (4)
F278| CØ E4                        CPY     #ØE4        ; (2) WITHIN 1 MS OF COMPLETION?
F27A| DØE8                        BNE     WNTRPT1     ; (3) (2) NO KEEP WRITTING AND POLLING.
F27C| EA                           NOP                 ; (2)
F27D| C8                           INY                 ; (2)
F27E| EA                 WDATA3     NOP                 ; (2)
F27F| EA                           NOP                 ; (2)
F28Ø| 48                           PHA                 ; (4)
F281| 68                           PLA                 ; (3)
F282| B9 ØØØ2                     LDA     NBUF1,Y     ; (4) WRITE LAST OF ENCODED BYTES
F285| 9D 8DCØ                       STA     Q6H,X       ; (5) WITHOUT POLLING INTERRUPTS.
F288| BD 8CCØ                       LDA     Q6L,X       ; (4)
F28B| A5 96                       LDA     CKSUM       ; (3) NORMALLY FOR TIMING
F28D| C8                           INY                 ; (2)
F28E| DØEE                        BNE     WDATA3      ; (3) (2)
F29Ø| FØØØ                        BEQ     WRCKSUM     ; (3) BRANCH ALWAYS
F292| 2Ø BBF2            WRCKSUM    JSR     WNIBL7      ; (13,9,6) GO WRITE CHECK SUM!!
F295| 48                           PHA                 ; (3)
F296| 68                           PLA                 ; (4)
F297| B9 CØF3            WRBITSLMK  LDA     BITSLIPMK,Y ; (4) LOAD BIT SLIP MARK
F29A| 2Ø BDF2                      JSR     WNIBL       ; (6,9,6)
F29D| C8                           INY                 ; (2)
F29E| CØ Ø4                        CPY     #Ø4         ; (2)
F2AØ| DØF5                        BNE     WRBITSLMK   ; (2) (3)
F2A2| 18                           CLC                 ; (2)
F2A3| BD 8ECØ            NOWRITE    LDA     Q7L,X       ; OUT OF WRITE MODE.
F2A6| BD 8CCØ                       LDA     Q6L,X       ; TO READ MODE.
F2A9| 6Ø                           RTS                 ; RETURN FROM WRITE.
F2AA|                    ;
F2AA| 2C 54F3            SERVICE    BIT     SEV         ; SET VFLAG TO INDICATE INTERRUPT
F2AD| 2Ø A3F2                      JSR     NOWRITE     ; TAKE IT OUT OF WRITE MODE!
F2BØ| A5 8F                       LDA     Ø8F
F2B2| 1ØØ2                        BPL     $Ø1Ø
F2B4| 85 8B                       STA     IMASK
```

```
F2B6| C6 8F                  $Ø1Ø       DEC     Ø8F
F2B8| 58                              CLI                     ; COULD NOT HAVE GOT HERE WITHOUT CLI OK
F2B9| 6Ø                              RTS
F2BA|                       ;
F2BA|                       ;*****************************
F2BA|                       ;                            *
F2BA|                       ;   7-BIT NIBL WRITE SUBRS   *
F2BA|                       ;                            *
F2BA|                       ;   A-REG OR'D PRIOR EXIT    *
F2BA|                       ;       CARRY CLEARED        *
F2BA|                       ;                            *
F2BA|                       ;*****************************
F2BA|                       ;
F2BA| 18                    WNIBL9     CLC                     ; (2) 9 CYCLES, THEN WRITE
F2BB| 48                    WNIBL7     PHA                     ; (3) 7 CYCLES, THEN WRITE
F2BC| 68                              PLA                     ; (4)
F2BD| 9D 8DCØ               WNIBL      STA     Q6H,X           ; (5) NIBL WRITE SUB
F2CØ| 1D 8CCØ                         ORA     Q6L,X           ; (4) CLOBBERS ACC. NOT CARRY
F2C3| 6Ø                              RTS
F2C4|                       ;
F2C4|                       ;*****************************
F2C4|                       ;                            *
F2C4|                       ;    PRENIBILIZE SUBR        *
F2C4|                       ;   (16-SECTOR FORMAT)       *
F2C4|                       ;                            *
F2C4|                       ;*****************************
F2C4|                       ;                            *
F2C4|                       ;  CONVERTS 256 BYTES OF     *
F2C4|                       ;  USER DATA IN (BUF) INTO   *
F2C4|                       ;  ENCODED BYTES TO BE       *
F2C4|                       ;  WRITTEN DIRECTLY TO DISK  *
F2C4|                       ;  ENCODED CHECK SUM IN      *
F2C4|                       ;  ZERO PAGE 'CKSUM'         *
F2C4|                       ;                            *
F2C4|                       ;   ---- ON ENTRY ----       *
F2C4|                       ;                            *
F2C4|                       ;  BUF IS 2-BYTE POINTER     *
F2C4|                       ;    TO 256 BYTES OF USER    *
F2C4|                       ;    DATA.                   *
F2C4|                       ;                            *
F2C4|                       ;  A-REG CHECK SUM.          *
F2C4|                       ;  X-REG UNCERTAIN           *
F2C4|                       ;  Y-REG HOLDS Ø.            *
F2C4|                       ;  CARRY SET.                *
F2C4|                       ;                            *
F2C4|                       ;*****************************
F2C4|                       ;
F2C4| A2 Ø2                 PRENIB16   LDX     #Ø2             ; START NBUF2 INDEX.
F2C6| AØ ØØ                           LDY     #ØØ             ; START USER BUF INDEX.
F2C8| 88                    PRENIB1    DEY                     ; NEXT USER BYTE
F2C9| B1 9B                           LDA     (BUF),Y
F2CB| 4A                              LSR     A               ; SHIFT TWO BITS OF
F2CC| 3E Ø1Ø3                         ROL     NBUF2-1,X       ; CURRENT USER BYTE
F2CF| 4A                              LSR     A               ; INTO CURRENT NBUF2
F2DØ| 3E Ø1Ø3                         ROL     NBUF2-1,X       ; BYTE.
F2D3| 99 Ø1Ø2                         STA     NBUF1+1,Y       ; (6 BITS LEFT).
F2D6| E8                              INX                     ; FROM Ø TO $55
F2D7| EØ 56                           CPX     #56
F2D9| 9ØED                           BCC     PRENIB1         ; BR IF NO WRAPAROUND
F2DB| A2 ØØ                           LDX     #ØØ             ; RESET NBUF2 INDEX
F2DD| 98                              TYA                     ; USER BUF INDEX
F2DE| DØE8                           BNE     PRENIB1         ; (DONE IF ZERO)
F2EØ| AØ 56                           LDY     #56             ; (ACC=Ø FOR CHECK SUM)
F2E2| 59 ØØØ3               PRENIB3    EOR     NBUF2-2,Y       ; COMBINE WITH PREVIOUS
F2E5| 29 3F                 PRENIB2    AND     #Ø3F            ; STRIP GARBAGE BITS
F2E7| AA                              TAX                     ; TO FORM RUNNING CHECK SUM
F2E8| BD 55F3                         LDA     NIBL,X          ; GET ENCODED EQUIV.
F2EB| 99 Ø1Ø3                         STA     NBUF2-1,Y       ; REPLACE PREVIOUS
F2EE| B9 ØØØ3                         LDA     NBUF2-2,Y       ; RESTORE ACTUAL PREVIOUS
F2F1| 88                              DEY
F2F2| DØEE                           BNE     PRENIB3         ; LOOP UNTIL ALL OF NBUF2 IS CONVERTED.
F2F4| 29 3F                          AND     #3F
F2F6| 59 Ø1Ø2                PRENIB4    EOR     NBUF1+1,Y       ; NOW DO THE SAME FOR
F2F9| AA                              TAX                     ; NIBBLE BUFFER 1
F2FA| BD 55F3                         LDA     NIBL,X          ; TO DO ANY BACK TRACKING (NBUF1-1)
F2FD| 99 ØØØ2                         STA     NBUF1,Y
F3ØØ| B9 Ø1Ø2                         LDA     NBUF1+1,Y       ; RECOVER THAT WHICH IS NOW 'PREVIOUS'
F3Ø3| C8                              INY
F3Ø4| DØFØ                           BNE     PRENIB4
F3Ø6| AA                              TAX                     ; USE LAST AS CHECK SUM
F3Ø7| BD 55F3                         LDA     NIBL,X
F3ØA| 85 96                          STA     CKSUM
F3ØC| 4C 4CF3                        JMP     SET1MEG         ; ALL DONE.
F3ØF|                       ;
F3ØF|                       ;*************************
F3ØF|                       ;                        *
F3ØF|                       ;    POSTNIBLIZE SUBR    *
F3ØF|                       ;    16-SECTOR FORMAT    *
F3ØF|                       ;                        *
F3ØF|                       ;*************************
```

```
F30F|                          ;
F30F| 38                       POSTNIB16  SEC
F310| A0 55                              LDY     #55          ; FIRST CONVERT TO 6 BIT NIBBLES
F312| A9 00                              LDA     #00          ; INIT CHECK SUM
F314| BE 0203                  PNIBL1     LDX     NBUF2,Y      ; GET ENCODED BYTE
F317| 5D 00F3                            EOR     DNIBL,X
F31A| 3030                              BMI     SET1MEG      ; SET 1 MHZ
F31C| 99 0203                            STA     NBUF2,Y      ; REPLACE WITH 6 BIT EQUIV.
F31F| 88                                DEY
F320| 10A6                              BPL     PRENIB1      ; LOOP UNTIL DONE WITH NIBBLE BUFFER 2
F322| C8                                INY                  ; NOW Y=0
F323| BE 0002                  PNIBL2     LDX     NBUF1,Y      ; DO THE SAME WITH
F326| 5D 00F3                            EOR     DNIBL,X
F329| 99 0002                            STA     NBUF1,Y      ; NIBBLE BUFFER 1
F32C| C8                                INY                  ; DO ALL 256 BYTES
F32D| D0F4                              BNE     PNIBL2
F32F| A6 96                             LDX     CKSUM        ; MAKE SURE CHECK SUM MATCHES
F331| 5D 00F3                            EOR     DNIBL,X      ; BETTER BE ZERO
F334| D016                              BNE     POSTERR      ; BRANCH IF IT IS
F336| A2 56                  POST1      LDX     #56          ; INIT NBUF2 INDEX
F338| CA                      POST2      DEX                  ; NBUF IDX $55 TO $00
F339| 30FB                              BMI     POST1        ; WRAPAROUND IF NEG
F33B| B9 0002                            LDA     NBUF1,Y
F33E| 5E 0203                            LSR     NBUF2,X      ; SHIFT 2 BITS FROM
F341| 2A                                ROL     A            ; CURRENT NBUF2 NIBL
F342| 5E 0203                            LSR     NBUF2,X      ; CURRENT NBUF1
F345| 2A                                ROL     A            ; NIBL.
F346| 91 9B                             STA     (BUF),Y      ; BYTE OF USER DATA
F348| C8                                INY                  ; NEXT USER BYTE
F349| D0ED                              BNE     POST2
F34B| 18                                CLC                  ; GOOD DATA
F34C| F34C                  POSTERR    .EQU    *
F34C| AD DFFF                SET1MEG    LDA     ENVIRON
F34F| 09 80                            ORA     #ONEMEG      ; SET TO ONE MEGAHERTZ CLOCK RATE
F351| 8D DFFF                            STA     ENVIRON
F354| 60                    SEV        RTS                  ; (SEV USED TO SET VFLAG)
F355|                          ;
F355|                          ;*************************
F355|                          ;                        *
F355|                          ;     6-BIT TO 7-BIT     *
F355|                          ;  NIBL CONVERSION TABLE  *
F355|                          ;                        *
F355|                          ;*************************
F355|                          ;                        *
F355|                          ;   CODES WITH MORE THAN  *
F355|                          ;   ONE PAIR OF ADJACENT  *
F355|                          ;    ZEROES OR WITH NO    *
F355|                          ;   ADJACENT ONES (EXCEPT *
F355|                          ;     B7) ARE EXCLUDED.  *
F355|                          ;                        *
F355|                          ;*************************
F355|                          ;
F355| 96 97 9A 9B 9D 9E 9F     NIBL   .BYTE 96,97,9A,9B,9D,9E,9F,0A6,0A7,0AB,0AC,0AD,0AE,0AF,0B2,0B3,0B4,0B5
F35C| A6 A7 AB AC AD AE AF
F363| B2 B3 B4 B5
F367| B6 B7 B9 BA BB BC BD            .BYTE 0B6,0B7,0B9,0BA,0BB,0BC,0BD,0BE,0BF,0CB,0CD,0CE,0CF,0D3,0D6,0D7
F36E| BE BF CB CD CE CF D3
F375| D6 D7
F377| D9 DA DB DC DD DE DF            .BYTE 0D9,0DA,0DB,0DC,0DD,0DE,0DF,0E5,0E6,0E7,0E9,0EA,0EB,0EC,0ED,0EE
F37E| E5 E6 E7 E9 EA EB EC
F385| ED EE
F387| EF F2 F3 F4 F5 F6 F7            .BYTE 0EF,0F2,0F3,0F4,0F5,0F6,0F7,0F9,0FA,0FB,0FC,0FD,0FE,0FF
F38E| F9 FA FB FC FD FE FF
F395|                          ;
F395|                          ;*************************
F395|                          ;                        *
F395|                          ;     7-BIT TO 6-BIT     *
F395|                          ;    'DENIBLIZE' TABL    *
F395|                          ;    (16-SECTOR FORMAT)  *
F395|                          ;                        *
F395|                          ;       VALID CODES      *
F395|                          ;     $96 TO $FF ONLY.   *
F395|                          ;                        *
F395|                          ;                        *
F395|                          ;   CODES WITH MORE THAN  *
F395|                          ;   ONE PAIR OF ADJACENT  *
F395|                          ;    ZEROES OR WITH NO    *
F395|                          ;   ADJACENT ONES (EXCEPT *
F395|                          ;    BIT 7) ARE EXCLUDED  *
F395|                          ;*************************
F395|                          ;
F395| F300                     DNIBL      .EQU    REGRWTS+300
F395| 01 00 01                            .BYTE   01,00,01
F398| 98 99 02 03 9C 04 05                .BYTE   98,99,02,03,9C,04,05,06,0A0,0A1,0A2,0A3,0A4,0A5,07,08,0A8
F39F| 06 A0 A1 A2 A3 A4 A5
F3A6| 07 08 A8
F3A9| A9 AA 09 0A 0B 0C 0D                .BYTE   0A9,0AA,09,0A,0B,0C,0D,0B0,0B1,0E,0F,10,11,12,13,0B8,14,15
F3B0| B0 B1 0E 0F 10 11 12
F3B7| 13 B8 14 15
F3BB| 16 17 18 19 1A                    .BYTE   16,17,18,19,1A
```

```
F3C0| DE AA EB FF C4 C5 C6  BITSLIPMK  .BYTE   ØDE,ØAA,ØEB,ØFF,ØC4,ØC5,ØC6,ØC7,ØC8,ØC9,ØCA,1B,ØCC,1C,1D,1E
F3C7| C7 C8 C9 CA 1B CC 1C
F3CE| 1D 1E
F3D0| D0 D1 D2 1F D4 D5 2Ø             .BYTE   ØD0,ØD1,ØD2,1F,ØD4,ØD5,2Ø,21,ØD8,22,23,24,25,26,27,28,ØE0,ØE1
F3D7| 21 D8 22 23 24 25 26
F3DE| 27 28 E0 E1
F3E2| E2 E3 E4 29 2A 2B E8             .BYTE   ØE2,ØE3,ØE4,29,2A,2B,ØE8,2C,2D,2E,2F,3Ø,31,32,ØF0,ØF1,33,34
F3E9| 2C 2D 2E 2F 3Ø 31 32
F3F0| F0 F1 33 34
F3F4| 35 36 37 38 F8 39 3A             .BYTE   35,36,37,38,ØF8,39,3A,3B,3C,3D,3E,3F
F3FB| 3B 3C 3D 3E 3F
F400|
F400|                          ;*************************
F400|                          ;                        *
F400|                          ;  FAST SEEK SUBROUTINE  *
F400|                          ;                        *
F400|                          ;*************************
F400|                          ;                        *
F400|                          ;   ---- ON ENTRY ----   *
F400|                          ;                        *
F400|                          ;  X-REG HOLDS SLOTNUM   *
F400|                          ;         TIMES $1Ø      *
F400|                          ;                        *
F400|                          ;  A-REG HOLDS DESIRED   *
F400|                          ;         HALFTRACK.     *
F400|                          ;                        *
F400|                          ;  CURTRK HOLDS DESIRED  *
F400|                          ;           HALFTRACK.   *
F400|                          ;                        *
F400|                          ;   ---- ON EXIT ----    *
F400|                          ;                        *
F400|                          ;  A-REG UNCERTAIN.      *
F400|                          ;  Y-REG UNCERTAIN.      *
F400|                          ;  X-REG UNDISTURBED.    *
F400|                          ;                        *
F400|                          ;  CURTRK AND TRKN HOLD  *
F400|                          ;      FINAL HALFTRACK.  *
F400|                          ;                        *
F400|                          ;  PRIOR HOLDS PRIOR     *
F400|                          ;    HALFTRACK IF SEEK   *
F400|                          ;    WAS REQUIRED.       *
F400|                          ;                        *
F400|                          ;  MONTIMEL AND MONTIMEH *
F400|                          ;    ARE INCREMENTED BY  *
F400|                          ;    THE NUMBER OF       *
F400|                          ;    1Ø0 USEC QUANTUMS   *
F400|                          ;    REQUIRED BY SEEK    *
F400|                          ;    FOR MOTOR ON TIME   *
F400|                          ;    OVERLAP.            *
F400|                          ;                        *
F400|                          ; --- VARIABLES USED --- *
F400|                          ;                        *
F400|                          ;  CURTRK, TRKN, COUNT,  *
F400|                          ;    PRIOR, SLOTTEMP     *
F400|                          ;    MONTIMEL, MONTIMEH  *
F400|                          ;                        *
F400|                          ;*************************
F400|                          ;
F400| 85 9E                    SEEK       STA     TRKN          ; SAVE TARGET TRACK
F402| C5 8C                               CMP     CURTRK        ; ON DESIRED TRACK?
F404| F042                               BEQ     SETPHASE      ; YES, ENERGIZE PHASE AND RETURN
F406| A9 ØØ                             LDA     #ØØ
F408| 85 95                             STA     TRKCNT        ; HALFTRACK COUNT.
F40A| A5 8C                    SEEK2      LDA     CURTRK        ; SAVE CURTRK FOR
F40C| 85 9D                             STA     PRIOR         ; DELAYED TURN OFF.
F40E| 38                               SEC
F40F| E5 9E                             SBC     TRKN          ; DELTA-TRACKS.
F411| F031                             BEQ     SEEKEND       ; BR IF CURTRK=DESTINATION
F413| B0Ø6                             BCS     OUT           ; (MOVE OUT, NOT IN)
F415| 49 FF                             EOR     #ØFF          ; CALC TRKS TO GO.
F417| E6 8C                             INC     CURTRK        ; DECR CURRENT TRACK (OUT)
F419| 90Ø4                             BCC     MINTST        ; (ALWAYS TAKEN).
F41B| 69 FE                    OUT        ADC     #ØFE          ; CALC TRACKS TO GO.
F41D| C6 8C                             DEC     CURTRK        ; DECR CURRENT TRACK (OUT)
F41F| C5 95                    MINTST     CMP     TRKCNT
F421| 90Ø2                             BCC     MAXTST        ; AND 'TRKS MOVED'
F423| A5 95                             LDA     TRKCNT
F425| C9 Ø9                    MAXTST     CMP     #Ø9
F427| B0Ø2                             BCS     STEP2         ; IF TRKCNT>$Ø8 LEAVE Y ALONE (Y=$Ø8)
F429| A8                       STEP       TAY                   ; ELSE SET ACCELERATION INDEX IN Y
F42A| 38                               SEC
F42B| 20 48F4                  STEP2      JSR     SETPHASE
F42E| B9 67F4                             LDA     ONTABLE,Y     ; FOR 'ONTIME'
F431| 20 56F4                             JSR     MSWAIT        ; (1Ø0 USEC INTERVALS)
F434| A5 9D                             LDA     PRIOR
F436| 18                               CLC                   ; FOR PHASE OFF
F437| 20 4AF4                          JSR     CLRPHASE      ; TURN OFF PRIOR PHASE
F43A| B9 70F4                          LDA     OFFTABLE,Y    ; THEN WAIT 'OFFTIME'
F43D| 20 56F4                          JSR     MSWAIT        ; (1Ø0 USEC INTERVALS)
F440| E6 95                            INC     TRKCNT        ; 'TRACKS MOVED' COUNT.
```

```
F442| DØC6                          BNE     SEEK2           ; (ALWAYS TAKEN)
F444| 2Ø 56F4                SEEKEND  JSR     MSWAIT          ; SETTLE 25 MSEC
F447| 18                            CLC                     ; SET FOR PHASE OFF
F448| A5 8C                  SETPHASE LDA     CURTRK          ; GET CURRENT TRACK
F44A| 29 Ø3                  CLRPHASE AND     #Ø3             ; MASK FOR 1 AND 4 PHASES
F44C| 2A                            ROL     A               ; DOUBLE FOR PHASE ON/OFF INDEX
F44D| Ø5 81                        ORA     IBSLOT
F44F| AA                            TAX
F45Ø| BD 8ØCØ                      LDA     PHASEOFF,X      ; TURN ON/OFF ONE PHASE
F453| A6 81                        LDX     IBSLOT          ; RESTORE X-REG
F455| 6Ø                  SEEKRTS  RTS                     ; AND RETURN
F456|                       ;
F456|                       ;***************************
F456|                       ;                          *
F456|                       ;    MSWAIT SUBROUTINE     *
F456|                       ;                          *
F456|                       ;***************************
F456|                       ;                          *
F456|                       ;   DELAYS A SPECIFIED     *
F456|                       ;    NUMBER OF 1ØØ USEC    *
F456|                       ;    INTERVALS FOR MOTOR   *
F456|                       ;    ON TIMING             *
F456|                       ;                          *
F456|                       ;     ---- ON EXIT -----   *
F456|                       ;                          *
F456|                       ;   A-REG HOLDS $ØØ        *
F456|                       ;   X-REG HOLDS $ØØ        *
F456|                       ;   Y-REG UNCHANGED        *
F456|                       ;   CARRY SET              *
F456|                       ;                          *
F456|                       ;   MONTIMEL, MONTIMEH     *
F456|                       ;    ARE INCREMENTED ONCE  *
F456|                       ;    PER 1ØØ USEC INTERVAL *
F456|                       ;    FOR MOTOR ON TIMING   *
F456|                       ;                          *
F456|                       ;     ---- ASSUMES -----   *
F456|                       ;                          *
F456|                       ;    1 USEC CYCLE TIME     *
F456|                       ;                          *
F456|                       ;***************************
F456|                       ;
F456| A2 11                  MSWAIT   LDX     #11
F458| CA                     MSW1     DEX                    ; DELAY 86 USEC
F459| DØFD                            BNE     MSW1
F45B| E6 99                          INC     MONTIMEL
F45D| DØØ2                          BNE     MSW2            ; DOUBLE BYTE INCREMENT
F45F| E6 9A                          INC     MONTIMEH
F461| 38                    MSW2     SEC
F462| E9 Ø1                          SBC     #Ø1             ; DONE IN INTERVALS
F464| DØFØ                          BNE     MSWAIT          ; (A-REG COUNTS)
F466| 6Ø                            RTS
F467|                       ;
F467|                       ;*************************
F467|                       ;                        *
F467|                       ;  PHASE ON-, OFF-TIME   *
F467|                       ;   TABLES IN 1ØØ-USEC   *
F467|                       ;   INTERVALS. (SEEK)    *
F467|                       ;                        *
F467|                       ;*************************
F467|                       ;
F467| Ø1 3Ø 28 24 2Ø 1E 1D  ONTABLE  .BYTE   Ø1,3Ø,28,24,2Ø,1E,1D,1C,1C
F46E| 1C 1C
F47Ø| 7Ø 2C 26 22 1F 1E 1D  OFFTABLE .BYTE   7Ø,2C,26,22,1F,1E,1D,1C,1C
F477| 1C 1C
F479|
F479| 86 83                  BLOCKIO  STX     IBTRK
F47B| AØ Ø5                          LDY     #Ø5
F47D| 48                            PHA
F47E| ØA                    TRKSEC   ASL     A
F47F| 26 83                          ROL     IBTRK
F481| 88                            DEY
F482| DØFA                          BNE     TRKSEC
F484| 68                            PLA
F485| 29 Ø7                          AND     #Ø7
F487| A8                            TAY
F488| B9 AØF4                       LDA     SECTABL,Y
F48B| 85 84                        STA     IBSECT
F48D| 2Ø ØØFØ                      JSR     REGRWTS
F49Ø| BØØB                         BCS     QUIT
F492| E6 86                        INC     IBBUFP+1
F494| E6 84                        INC     IBSECT
F496| E6 84                        INC     IBSECT
F498| 2Ø ØØFØ                      JSR     REGRWTS
F49B| C6 86                        DEC     IBBUFP+1
F49D| A5 88                  QUIT     LDA     IBSTAT
F49F| 6Ø                            RTS
F4AØ|                       ;
F4AØ| ØØ Ø4 Ø8 ØC Ø1 Ø5 Ø9  SECTABL  .BYTE   ØØ,Ø4,Ø8,ØC,Ø1,Ø5,Ø9,ØD
F4A7| ØD
F4A8|                       ;******************************
```

```
F4A8|                          ;                              *
F4A8|                          ;   JOYSTICK READ ROUTINE      *
F4A8|                          ;                              *
F4A8|                          ;******************************
F4A8|                          ;  ENTRY  ACC= COUNT DOWN HIGH *
F4A8|                          ;         X&Y= DON'T CARE      *
F4A8|                          ;                              *
F4A8|                          ;   EXIT  ACC= TIMER HIGH BYTE *
F4A8|                          ;           Y= TIMER LOW BYTE  *
F4A8|                          ;         CARRY CLEAR          *
F4A8|                          ;                              *
F4A8|                          ;     IF CARRY SET, ROUTINE    *
F4A8|                          ;       WAS INTERRUPTED &      *
F4A8|                          ;     ACC & Y ARE INVALID      *
F4A8|                          ;******************************
F4A8|                          ;
F4A8| FFD9                     TIMLATCH   .EQU    ØFFD9
F4A8| FFD8                     TIMER1L    .EQU    ØFFD8
F4A8| FFD9                     TIMER1H    .EQU    ØFFD9
F4A8| CØ66                     JOYRDY     .EQU    ØCØ66
F4A8|                          ;
F4A8| F4A8                     ANALOG     .EQU    *              ; CARRY SHOULD BE SET!
F4A8| 8D D9FF                               STA     TIMLATCH       ; START THE TIMER!
F4AB| AD EFFF                  ANLOG1     LDA     INTERUPT
F4AE| 2D 66CØ                               AND     JOYRDY         ; WAIT FOR ONE OR THE OTHER TO GO LOW
F4B1| 3ØF8                               BMI     ANLOG1
F4B3| AD 66CØ                               LDA     JOYRDY         ; WAS IT REALLY THE JOPYSTICK?
F4B6| 3ØØC                               BMI     GOODTIME       ; NOPE, WHAT TIME IS IT?
F4B8| 18                               CLC                    ; TIME'S A SLIP SLIDIN AWAY
F4B9| AD D9FF                            LDA     TIMER1H        ; NOW, WHAT TIME IS IT?
F4BC| AC D8FF                            LDY     TIMER1L
F4BF| 1ØØ3                            BPL     GOODTIME       ; TIME WAS VALID!
F4C1| AD D9FF                            LDA     TIMER1H        ; HI BYTE CHANGED
F4C4| 6Ø                       GOODTIME   RTS
F4C5|
F4C5|                                    .END
```

SYMBOL TABLE DUMP

AB - Absolute LB - Label UD - Undefined MC - Macro
RF - Ref DF - Def PR - Proc FC - Func
PB - Public PV - Private CS - Consts

ALDONE1	LB FØEB	ALLDONE	LB FØE3	ALLOFF	LB F118	ANALOG	LB F4A8	ANLOG1	LB F4AB
BITSLIPM	LB F3CØ	BLOCKIO	LB F479	BUF	AB ØØ9B	CHKDRV	LB F12B	CHKDRV1	LB F12D
CHKINT	LB F1AA	CKDRTS	LB F13D	CKSUM	AB ØØ96	CLRPHASE	LB F44A	CONWAIT	LB FØ54
CORRECTV	LB FØC4	COUNT	AB ØØ95	CSSTV	AB ØØ97	CSUM	AB ØØ89	CSUM1	AB ØØ97
CURTRK	AB ØØ8C	DISKIO	PR ----	DNIBL	LB F3ØØ	DRIVSEL	LB FØ35	DRV1EN	AB CØ8B
DRVERR	LB FØE8	DRVINDX	LB F13E	DRVOEN	AB CØ8A	DRVOTRK	AB ØØ85	DRVWAIT	LB FØ41
DVMOT	AB ØØEØ	ENVIRON	AB FFDF	ENVTEMP	AB ØØ9F	GOCAL	LB FØA7	GOCAL1	LB FØA6
GOODTIME	LB F4C4	GOSEEK	LB F115	GOSERV	LB F1B4	HNDLERR	LB FØEA	HRDERRS	AB ØØ8Ø
IBBUFP	AB ØØ85	IBCMD	AB ØØ87	IBDERR	AB ØØ82	IBDRVN	AB ØØ82	IBNODRV	AB ØØ8Ø
IBRERR	AB ØØ83	IBSECT	AB ØØ84	IBSLOT	AB ØØ81	IBSMOD	AB ØØ89	IBSTAT	AB ØØ88
IBTRK	AB ØØ83	IBWPER	AB ØØ81	IMASK	AB ØØ8B	INTERUPT	AB FFEF	IOBPDN	AB ØØ8A
JOYRDY	AB CØ66	LAST	AB ØØ95	MAXTST	LB F425	MINTST	LB F41F	MONTIMEH	AB ØØ9A
MONTIMEL	AB ØØ99	MOTOF	LB FØ52	MOTOROFF	AB CØ88	MOTORON	AB CØ89	MSW1	LB F458
MSW2	LB F461	MSWAIT	LB F456	MYSEEK	LB F1Ø4	NBUF1	AB Ø2ØØ	NBUF2	AB Ø3Ø2
NIBL	LB F355	NODRIVER	LB FØ64	NOWRITE	LB F2A3	NXOFF	LB F11A	OFFTABLE	LB F47Ø
OK	LB FØ48	ONEMEG	AB ØØ8Ø	ONTABLE	LB F467	OUT	LB F41B	PHASEOFF	AB CØ8Ø
PHASEON	AB CØ81	PHASON	AB CØ81	PHSOFF	AB CØ8Ø	PNIBL1	LB F314	PNIBL2	LB F323
POST1	LB F336	POST2	LB F338	POSTERR	LB F34C	POSTNIB1	LB F3ØF	PRENIB1	LB F2C8
PRENIB16	LB F2C4	PRENIB2	LB F2E5	PRENIB3	LB F2E2	PRENIB4	LB F2F6	PRIOR	AB ØØ9D
Q6H	AB CØ8D	Q6L	AB CØ8C	Q7H	AB CØ8F	Q7L	AB CØ8E	QUIT	LB F49D
RD1	LB F14D	RD2	LB F157	RD3	LB F163	RD4	LB F16E	RD5	LB F18Ø
RD5A	LB F181	RD6	LB F195	RDA1	LB F1C4	RDA2	LB F1CE	RDA3	LB F1D9
RDA4	LB F1E7	RDA5	LB F1EF	RDA6	LB F2Ø1	RDA7	LB F2ØB	RDADR16	LB F1B9
RDAFLD	LB F1E5	RDASN1	LB F1C9	RDASYN	LB F1BD	RDCKSUM	LB F1AØ	RDERR	LB F1B7
RDEXIT	LB F214	RDRIGHT	LB FØAC	READ16	LB F148	REGRWTS	LB FØØØ	RETRYCNT	AB ØØ93
RSYNC	LB F14A	RSYNC1	LB F152	RTTRK	LB FØCØ	SECT	AB ØØ98	SECTABL	LB F4AØ
SEEK	LB F4ØØ	SEEK1	LB F1Ø5	SEEK2	LB F4ØA	SEEKCNT	AB ØØ94	SEEKEND	LB F444
SEEKRTS	LB F455	SERVICE	LB F2AA	SET1MEG	LB F34C	SETPHASE	LB F448	SETTRK	LB F125
SEV	LB F354	STEP	LB F429	STEP2	LB F42B	TEMP	AB ØØ97	TIMER1H	AB FFD9
TIMER1L	AB FFD8	TIMLATCH	AB FFD9	TRACK	AB ØØ99	TRKCNT	AB ØØ95	TRKN	AB ØØ9E
TRKN1	AB ØØ99	TRKSEC	LB F47E	TRYADR	LB FØ83	TRYADR2	LB FØ8A	TRYTRK	LB FØ69
TRYTRK2	LB FØ7F	TWOMEG	AB ØØ7F	VOLUME	AB ØØ9A	VRYFRST	LB F253	WDATA2	LB F26E
WDATA3	LB F27E	WEXIT	LB F215	WINTRPT	LB F24B	WMIDLE	LB F267	WNIBL	LB F2BD
WNIBL7	LB F2BB	WNIBL9	LB F2BA	WNTRPT1	LB F264	WRBITSLM	LB F297	WRCKSUM	LB F292
WRIT	LB FØF9	WRIT1	LB F22Ø	WRITE16	LB F216	WRTFRST	LB F255	WSYNC	LB F22D

Assembly complete: 1Ø76 lines
Ø Errors flagged on this Assembly

65Ø2 OPCODE STATIC FREQUENCIES

```
   ADC :    1  m
   AND :    8  |  *******
```

```
    ASL :     3  |  **
    BCC :    1Ø  |  ********
    BCS :     7  |  ******
    BEQ :     8  |  *******
    BIT :     3  |  **
    BMI :    1Ø  |  ********
    BNE :    38  |  **********************************
    BPL :    28  |  ************************
    BVC :     1  m
    CLC :     9  |  *******
    CLI :     2  |  *
    CLV :     1  m
    CMP :    14  |  *************
    CPX :     1  m
    CPY :     4  |  ***
    DEC :     5  |  ****
    DEX :     2  |  *
    DEY :    13  |  ************
    EOR :     8  |  *******
    INC :    1Ø  |  ********
    INX :     2  |  *
    INY :    12  |  **********
    JMP :     2  |  *
    JSR :    39  |  ***********************************
    LDA :    86  M  ********************************************************************************
    LDX :    12  |  **********
    LDY :    18  |  ****************
    LSR :     9  |  *******
    NOP :    13  |  ************
    ORA :     9  |  *******
    PHA :    1Ø  |  ********
    PHP :     4  |  ***
    PLA :    11  |  *********
    PLP :     3  |  **
    ROL :     7  |  ******
    ROR :     6  |  *****
    RTS :    16  |  **************
    SBC :     2  |  *
    SEC :     9  |  *******
    SEI :     1  m
    STA :    42  |  ************************************
    STX :     1  m
    STY :     3  |  **
    TAX :     5  |  ****
    TAY :     3  |  **
    TXA :     1  m
    TYA :     4  |  ***

    Minimum frequency =     1
    Maximum frequency =    86

    Average frequency =    1Ø

    Unused opcodes:

    BRK  BVS  CLD  RTI  SED  TSX  TXS

    Program opcode usage:  87 %

--------------------------------------------------------------------------------------------------
(1.ØØ) That's all, Folks ...
--------------------------------------------------------------------------------------------------
```

Source Code Listing

for

Apple ///

ROM Diagnostics

David T. Craig
736 Edgewater
Wichita, Kansas 67230

```
0000|                       ;¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢
0000|                       ;¢ APPLE /// ROM - DIAGNOSTIC ROUTINES
0000|                       ;¢ COPYRIGHT 1979 BY APPLE COMPUTER, INC.
0000|                       ;¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢
0000|
0000|                                 .ABSOLUTE
0000|                                 .PROC   SARATESTS
0000|
0000|                       ;*********************************************************************
0000|                       ;
0000|                       ; SARA DIAGNOSTIC TEST ROUTINES
0000|                       ;
0000|                       ; DECEMBER 18,1979
0000|                       ;  BY
0000|                       ; W. BROEDNER & R. LASHLEY
0000|                       ;
0000|                       ; COPYRIGHT 1979 BY APPLE COMPUTER, INC.
0000|                       ;
0000|                       ;*********************************************************************
0000|
0000| 0001                  ROM         .EQU    01
0000| 0000                  ZRPG        .EQU    00
0000| 0010                  ZRPG1       .EQU    10
0000| 0018                  PTRLO       .EQU    ZRPG1+08
0000| 0019                  PTRHI       .EQU    ZRPG1+09
0000| 001A                  BNK         .EQU    ZRPG1+0A
0000| 0087                  IBCMD       .EQU    87
0000| 0085                  IBBUFP      .EQU    85
0000| 0091                  PREVTRK     .EQU    91
0000| F479                  BLOCKIO     .EQU    0F479
0000| 005D                  CV          .EQU    5D
0000| 00FF                  STK0        .EQU    0FF
0000| 1419                  IBNK        .EQU    1400+PTRHI
0000| 1810                  PHPR        .EQU    1800+ZRPG1
0000| C000                  KYBD        .EQU    0C000
0000| C008                  KEYBD       .EQU    0C008
0000| C010                  KBDSTRB     .EQU    0C010
0000| C058                  PDLEN       .EQU    0C058
0000| C047                  ADRS        .EQU    0C047
0000| C050                  GRMD        .EQU    0C050
0000| C051                  TXTMD       .EQU    0C051
0000| C066                  ADTO        .EQU    0C066
0000| C0D0                  DISKOFF     .EQU    0C0D0
0000| C0F1                  ACIAST      .EQU    0C0F1
0000| C0F2                  ACIACM      .EQU    0C0F2
0000| C0F3                  ACIACN      .EQU    0C0F3
0000| C100                  SLT1        .EQU    0C100
0000| C200                  SLT2        .EQU    0C200
0000| C300                  SLT3        .EQU    0C300
0000| C400                  SLT4        .EQU    0C400
0000| CFFF                  EXPROM      .EQU    0CFFF
0000| FFD0                  ZPREG       .EQU    0FFD0
0000| FFDF                  SYSD1       .EQU    0FFDF
0000| FFD2                  SYSD2       .EQU    0FFD2
0000| FFD3                  SYSD3       .EQU    0FFD3
0000| FFE0                  SYSE0       .EQU    0FFE0
0000| FFEF                  BNKSW       .EQU    0FFEF
0000| FFE2                  SYSE2       .EQU    0FFE2
0000| FFE3                  SYSE3       .EQU    0FFE3
0000| FC25                  COUT        .EQU    0FC25
0000| FD07                  CROUT1      .EQU    0FD07
0000| FD0F                  KEYIN       .EQU    0FD0F
0000| FBC7                  SETCVH      .EQU    0FBC7
0000| FD98                  CLDSTRT     .EQU    0FD98
0000| FD9D                  SETUP       .EQU    0FD9D
0000| F901                  MONITOR     .EQU    0F901
0000|                       ;
0000|                                 .ORG    0F4C5
F4C5| 00 B1 B2 BA B9 10 00  RAMTBL      .BYTE   00,0B1,0B2,0BA,0B9,10,00,13
F4CC| 13
F4CD| F4CD                  CHPG        .EQU    *
F4CD| 52 41                             .ASCII  "RA"
F4CF| CD                               .BYTE   0CD          ; M
F4D0| 52 4F                            .ASCII  "RO"
F4D2| CD                               .BYTE   0CD          ; M
F4D3| 56 49                            .ASCII  "VI"
F4D5| C1                               .BYTE   0C1          ; A
F4D6| 41 43 49                         .ASCII  "ACI"
F4D9| C1                               .BYTE   0C1          ; A
F4DA| 41 2F                            .ASCII  "A/"
F4DC| C4                               .BYTE   0C4          ; D
F4DD| 44 49 41 47 4E 4F 53             .ASCII  "DIAGNOSTI"
F4E4| 54 49
F4E6| C3                               .BYTE   0C3          ; C
F4E7| 5A                               .ASCII  "Z"
F4E8| D0                               .BYTE   0D0          ; P
F4E9| 52 45 54 52                      .ASCII  "RETR"
F4ED| D9                               .BYTE   0D9          ; Y
F4EE|                       ;
F4EE|                       ; SETUP SYSTEM
```

W=Walt

Broedner later designed the hardware for the Apple IIe computer which was released in January 1983

```
F4EE|                    ;
F4EE|                    ;
F4EE| A9 53                        LDA     #52+ROM         ; TURN OFF SCREEN, SET 2MHZ SPEED
F4F0| 8D DFFF                      STA     SYSD1           ; AND RUN OFF ROM
F4F3| A2 00                        LDX     #00             ; SET BANK SWITCH TO ZERO
F4F5| 8E E0FF                      STX     SYSE0
F4F8| 8E EFFF                      STX     BNKSW
F4FB| 8E D0FF                      STX     ZPREG           ; AND SET ZERO PAGE SAME
F4FE| CA                          DEX
F4FF| 8E D2FF                      STX     SYSD2           ; PROGRAM DDR'S
F502| 8E D3FF                      STX     SYSD3
F505| 9A                          TXS
F506| E8                          INX
F507| A9 0F                       LDA     #0F
F509| 8D E3FF                      STA     SYSE3
F50C| A9 3F                       LDA     #3F
F50E| 8D E2FF                      STA     SYSE2
F511| A0 0E                       LDY     #0E
F513| B9 D0C0              DISK1   LDA     DISKOFF,Y
F516| 88                          DEY
F517| 88                          DEY
F518| 10F9                        BPL     DISK1
F51A| AD 08C0                     LDA     KEYBD
F51D| 29 04                       AND     #04
F51F| D003                        BNE     NXBYT
F521| 4C 86F6                     JMP     RECON
F524|                    ;
F524|                    ; VERIFY ZERO PAGE
F524|                    ;
F524| A9 01              NXBYT   LDA     #01             ; ROTATE A 1 THROUGH
F526| 95 00              NXBIT   STA     ZRPG,X          ; EACH BIT IN THE 0 PG
F528| D5 00                      CMP     ZRPG,X          ; TO COMPLETELY TEST
F52A| D0FE              NOGOOD  BNE     NOGOOD          ; THE PAGE. HANG IF NOGOOD.
F52C| 0A                        ASL     A               ; TRY NEXT BIT OF BYTE
F52D| D0F7                      BNE     NXBIT           ; UNTIL BYTE IS ZERO.
F52F| E8                        INX                     ; CONTINUE UNTIL PAGE
F530| D0F2                      BNE     NXBYT           ; IS DONE.
F532| 8A              CNTWR   TXA                     ; PUSH A DIFFERENT
F533| 48                        PHA                     ; BYTE ONTO THE
F534| E8                        INX                     ; STACK UNTIL ALL
F535| D0FB                      BNE     CNTWR           ; STCK BYTES ARE FULL.
F537| CA                        DEX                     ; THEN PULL THEM
F538| 86 18                     STX     PTRLO           ; OFF AND COMPARE TO
F53A| 68              PULBT   PLA                     ; THE COUNTER GOING
F53B| C5 18                     CMP     PTRLO           ; BACKWARDS. HANG IF
F53D| D0EB                      BNE     NOGOOD          ; THEY DON'T AGREE.
F53F| C6 18                     DEC     PTRLO           ; GET NEXT COUNTER BYTE
F541| D0F7                      BNE     PULBT           ; CONTINUE UNTIL STACK
F543| 68                        PLA                     ; IS DONE. TEST LAST BYTE
F544| D0E4                      BNE     NOGOOD          ; AGAINST ZERO.
F546|                    ;
F546|                    ; SIZE IN MEMORY
F546|                    ;
F546| A2 08                     LDX     #08             ; ZERO THE BYTES USED TO DISPLAY
F548| 95 10              NOMEM   STA     ZRPG1,X         ; THE BAD RAM LOCATIONS
F54A| CA                        DEX                     ; EACH BYTE= A CAS LINE
F54B| 10FB                      BPL     NOMEM           ; ON THE SARA BOARD.
F54D| A2 02                     LDX     #02             ; STARTING AT PAGE 2
F54F| 86 19              NMEM1   STX     PTRHI           ; TEST THE LAST BYTE
F551| A9 00                     LDA     #00             ; IN EACH MEM PAGE TO
F553| A0 FF                     LDY     #0FF            ; SEE IF THE CHIPS ARE
F555| 91 18                     STA     (PTRLO),Y       ; THERE..(AVOID 0 & STK PAGES)
F557| D1 18                     CMP     (PTRLO),Y       ; CAN THE BYTE BE O'D?
F559| F007                      BEQ     NMEM2
F55B| 20 48F7                   JSR     RAM             ; NO, FIND WHICH CAS IT IS.
F55E| 94 10                     STY     ZRPG1,X         ; SET CORRES. BYTE TO $FF
F560| A6 19                     LDX     PTRHI           ; RESTORE X REGISTER
F562| E8              NMEM2   INX                     ; AND INCREMENT TO NEXT
F563| E0 C0                     CPX     #0C0            ; PAGE UNTIL I/O IS REACHED.
F565| D0E8                      BNE     NMEM1
F567| A2 20                     LDX     #20             ; THEN RESET TO PAGE 20
F569| EE EFFF                   INC     BNKSW           ; AND GOTO NEXT BANK TO
F56C| AD EFFF                   LDA     BNKSW           ; CONTINUE. (MASK INPUTS
F56F| 29 0F                     AND     #0F             ; FROM BANKSWITCH TO SEE
F571| C9 03                     CMP     #03             ; WHAT SWITCH IS SET TO)
F573| D0DA                      BNE     NMEM1           ; CONTINUE UNTIL BANK '3'
F575|                    ;
F575|                    ; SETUP SCREEN
F575|                    ;
F575| 20 9DFD              ERRLP   JSR     SETUP           ; CALL SCRN SETUP ROUTINE
F578| A2 00                     LDX     #00             ; SETUP I/O AGAIN
F57A| 8E E0FF                   STX     SYSE0           ; FOR VIA TEST
F57D| CA                        DEX                     ; PROGRAM DATA DIR
F57E| 8E D2FF                   STX     SYSD2           ; REGISTERS
F581| 8E D3FF                   STX     SYSD3
F584| A9 3F                     LDA     #3F
F586| 8D E2FF                   STA     SYSE2
F589| A9 0F                     LDA     #0F
F58B| 8D E3FF                   STA     SYSE3
F58E| A2 10                     LDX     #10             ; HEADING OF 'DIAGNOSTICS' WITH
```

```
F590| 20 38F7                      JSR     STRWT         ; THIS SUBROUTINE
F593| A2 00              ERRLP1    LDX     #00           ; PRINT 'RAM'
F595| 86 5D                        STX     CV            ; SET CURSOR TO 2ND LINE
F597| A9 04                        LDA     #04           ; SPACE CURSOR OUT 3
F599| 20 C7FB                      JSR     SETCVH        ; (X STILL=0 ON RETURN)
F59C| 20 38F7                      JSR     STRWT         ; THE SAME SUBROUTINE
F59F| A2 07                        LDX     #07           ; FOR BYTES 7 - 0 IN
F5A1| F5A1              RAMWT1    .EQU    *
F5A1| B5 10                        LDA     ZRPG1,X       ; OUT EACH BIT AS A
F5A3| A0 08                        LDY     #08           ; ' ' OR '1' FOR INDICATE BAD OR MISSING RAM
F5A5| 0A                RAMWT2    ASL     A             ; CHIPS SUBROUTINE 'RAM'        RAM
F5A6| 48                          PHA                   ; SETS UP THESE BYTES
F5A7| A9 AE                        LDA     #0AE          ; LOAD A '.' TO ACC.
F5A9| 9002                        BCC     RAMWT4
F5AB| A9 31                        LDA     #31           ; LOAD A '1' TO ACC.
F5AD| 20 25FC            RAMWT4    JSR     COUT          ; AND PRINT IT
F5B0| 68                          PLA                   ; RESTORE BYTE
F5B1| 88                          DEY                   ; AND ROTATE ALL 8
F5B2| D0F1                        BNE     RAMWT2        ; TIMES
F5B4| 20 07FD                      JSR     CROUT1        ; CLEAR TO END OF LINE.
F5B7| CA                          DEX
F5B8| 10E7                        BPL     RAMWT1
F5BA|                   ;
F5BA|                   ; ZPG & STK TEST
F5BA|                   ;
F5BA| 9A                          TXS
F5BB| 8C EFFF                     STY     BNKSW
F5BE| 98                ZP1       TYA
F5BF| 8D D0FF                     STA     ZPREG
F5C2| 85 FF                       STA     STK0
F5C4| C8                          INY
F5C5| 98                          TYA
F5C6| 48                          PHA
F5C7| 68                          PLA
F5C8| C8                          INY
F5C9| C0 20                       CPY     #20
F5CB| D0F1                        BNE     ZP1
F5CD| A0 00                       LDY     #00
F5CF| 8C D0FF                     STY     ZPREG
F5D2| 86 18                       STX     PTRLO
F5D4| E8                ZP2       INX
F5D5| 86 19                       STX     PTRHI
F5D7| 8A                          TXA
F5D8| D1 18                       CMP     (PTRLO),Y
F5DA| D006                        BNE     ZP3
F5DC| E0 1F                       CPX     #1F
F5DE| D0F4                        BNE     ZP2
F5E0| F005                        BEQ     ROMTST
F5E2| F5E2              ZP3       .EQU    *             ; CHIP IS THERE, BAD ZERO AND STACK
F5E2| A2 1A                       LDX     #1A           ; SO PRINT 'ZP' MESSAGE
F5E4| 20 7BF7                     JSR     MESSERR       ; & SET FLAG (2MHZ MODE)
F5E7|                   ;
F5E7|                   ; ROM TEST ROUTINE
F5E7|                   ;
F5E7| A9 00              ROMTST    LDA     #00           ; SET POINTERS TO
F5E9| A8                          TAY                   ; $F000
F5EA| A2 F0                       LDX     #0F0
F5EC| 85 18                       STA     PTRLO
F5EE| 86 19                       STX     PTRHI         ; SET X TO $FF
F5F0| A2 FF                       LDX     #0FF          ; FOR WINDOWING I/O
F5F2| 51 18              ROMTST1   EOR     (PTRLO),Y     ; COMPUTE CHKSUM ON
F5F4| E4 19                       CPX     PTRHI         ; EACH ROM BYTE,
F5F6| D006                        BNE     ROMTST2       ; WINDOW OUT
F5F8| C0 BF                       CPY     #0BF          ; RANGES FFC0-FFEF
F5FA| D002                        BNE     ROMTST2
F5FC| A0 EF                       LDY     #0EF
F5FE| C8                ROMTST2   INY
F5FF| D0F1                        BNE     ROMTST1
F601| E6 19                       INC     PTRHI
F603| D0ED                        BNE     ROMTST1
F605| A8                          TAY                   ; TEST ACC. FOR 0
F606| F005                        BEQ     VIATST        ; YES, NEXT TEST
F608| A2 03                       LDX     #03           ; PRINT 'ROM' AND
F60A| 20 7BF7                     JSR     MESSERR       ; SET ERROR
F60D|                   ;
F60D|                   ; VIA TEST ROUTINE
F60D|                   ;
F60D| 18                VIATST    CLC                   ; SET UP FOR ADDING BYTES
F60E| D8                          CLD
F60F| AD E0FF                     LDA     SYSE0         ; MASK OFF INPUT BITS
F612| 29 3F                       AND     #3F           ; AND STORE BYTE IN
F614| 85 18                       STA     PTRLO         ; TEMPOR. LOCATION
F616| AD EFFF                     LDA     BNKSW         ; MASK OFF INPUT BITS
F619| 29 4F                       AND     #4F           ; AND ADD TO STORED
F61B| 65 18                       ADC     PTRLO         ; BYTE IN TEMP. LOC.
F61D| 6D D0FF                     ADC     ZPREG         ; ADD REMAINING
F620| 85 18                       STA     PTRLO         ; REGISTERS OF THE
F622| AD DFFF                     LDA     SYSD1         ; VIA'S
F625| 29 5F                       AND     #5F           ; (MASK THIS ONE)
F627| 65 18                       ADC     PTRLO         ; AND TEST
```

```
F629| 6D D2FF                    ADC     SYSD2         ; TO SEE
F62C| 6D D3FF                    ADC     SYSD3         ; IF THEY AGREE
F62F| 6D E2FF                    ADC     SYSE2         ; WITH THE RESET
F632| 6D E3FF                    ADC     SYSE3         ; CONDITION.
F635| C9 E1                      CMP     #ØEØ+ROM      ; =E1?
F637| FØØ5                      BEQ     ACIA          ; YES, NEXT TEST
F639| A2 Ø6                      LDX     #Ø6           ; NO, PRINT 'VIA' MESS
F63B| 2Ø 7BF7                    JSR     MESSERR       ; AND SET ERROR FLAG
F63E|                    ;
F63E|                    ; ACIA TEST
F63E|                    ;
F63E| 18                 ACIA    CLC                   ; SET UP FOR ADDITION
F63F| A9 9F                      LDA     #9F           ; MASK INPUT BITS
F641| 2D F1CØ                    AND     ACIAST        ; FROM STATUS REG
F644| 6D F2CØ                    ADC     ACIACM        ; AND ADD DEFAULT STATES
F647| 6D F3CØ                    ADC     ACIACN        ; OIF CONTROL AND COMMAND
F64A| C9 1Ø                      CMP     #1Ø           ; REGS.  =1Ø?
F64C| FØØ5                      BEQ     ATD           ; YES, NEXT TEST
F64E| A2 Ø9                      LDX     #Ø9           ; NO, 'ACIA' MESSAGE AND
F65Ø| 2Ø 7BF7                    JSR     MESSERR       ; THEN SET ERROR FLAG
F653|                    ;
F653|                    ; A/D TEST ROUTINE
F653|                    ;
F653| A9 CØ              ATD     LDA     #ØCØ
F655| 8D DCFF                    STA     ØFFDC
F658| AD 5ACØ                    LDA     PDLEN+2
F65B| AD 5ECØ                    LDA     PDLEN+6
F65E| AD 5CCØ                    LDA     PDLEN+4
F661| AØ 2Ø                      LDY     #2Ø
F663| 88              ADCTST1 DEY                   ; WAIT FOR 4Ø USEC
F664| DØFD                      BNE     ADCTST1
F666| AD 5DCØ                    LDA     PDLEN+5       ; SET A/D RAMP
F669| C8              ADCTST3 INY                   ; COUNT FOR CONVERSION
F66A| FØØA                      BEQ     ADCERR
F66C| AD 66CØ                    LDA     ADTO          ; IF BIT 7=1?
F66F| 3ØF8                      BMI     ADCTST3       ; YES, CONTINUE
F671| 98                        TYA                   ; NO, MOVE COUNT TO ACC
F672| 29 EØ                     AND     #ØEØ          ; ACC<32
F674| FØØ5                      BEQ     KEYPLUG
F676| F676              ADCERR  .EQU    *             ; NO,
F676| A2 ØD                     LDX     #ØD           ; PRINT 'A/D' MESS
F678| 2Ø 7BF7                    JSR     MESSERR       ; AND SET ERROR FLAG
F67B|                    ;
F67B|                    ; KEYBOARD PLUGIN TEST
F67B|                    ;
F67B| AD Ø8CØ              KEYPLUG LDA     KEYBD         ; IS KYBD PLUGGED IN?
F67E| ØA                        ASL     A             ; (IS LIGHT CURRENT
F67F| 1Ø41                      BPL     SEX           ;  PRESENT?) NO, BRANCH
F681| AD DFFF                    LDA     SYSD1         ; IS ERROR FLAG SET?
F684| 3Ø3C                      BMI     SEX           ; ERROR HANG
F686|                    ;
F686|                    ; RECONFIGURE THE SYSTEM
F686|                    ;
F686| A9 77              RECON   LDA     #77           ; TURN ON SCREEN
F688| 8D DFFF                    STA     SYSD1
F68B| 2Ø 98FD                    JSR     CLDSTRT       ; INITIALIZE MONITOR AND DEFAULT CHARACTER SET
F68E| 2C 1ØCØ                    BIT     KBDSTRB       ; CLEAR KEYBOARD
F691| AD FFCF                    LDA     EXPROM        ; DISABLE ALL SLOTS
F694| AD 2ØCØ                    LDA     ØCØ2Ø
F697| A9 1Ø                      LDA     #1Ø           ; TEST FOR "APPLE 1"
F699| 2D Ø8CØ                    AND     KEYBD
F69C| DØØ3                      BNE     BOOT          ; NO, DO REGULAR BOOT
F69E| 2Ø Ø1F9                    JSR     MONITOR       ; AND NEVER COME BACK
F6A1| A2 Ø1              BOOT    LDX     #Ø1           ; READ BLOCK Ø
F6A3| 86 87                     STX     IBCMD
F6A5| CA                        DEX
F6A6| 86 85                     STX     IBBUFP        ; INTO RAM AT $AØØØ
F6A8| A9 AØ                     LDA     #ØAØ
F6AA| 85 86                     STA     IBBUFP+1
F6AC| 4A                        LSR     A             ; FOR TRACK 8Ø
F6AD| 85 91                     STA     PREVTRK       ; MAKE IT RECALIBRATE TOO!
F6AF| 8A                        TXA
F6BØ| 2Ø 79F4                    JSR     BLOCKIO
F6B3| 9ØØA                      BCC     GOBOOT        ; IF WE'VE SUCCEEDED. DO IT UP
F6B5| A2 1C                     LDX     #1C
F6B7| 2Ø 38F7                    JSR     STRWT         ; 'RETRY'
F6BA| 2Ø ØFFD                    JSR     KEYIN
F6BD| BØE2                      BCS     BOOT
F6BF| 4C ØØAØ            GOBOOT  JMP     ØAØØØ         ; GO TO IT FOOL...
F6C2|                    ;
F6C2|                    ; SYSTEM EXCERCISER
F6C2|                    ;
F6C2| AØ 7F              SEX     LDY     #7F           ; TRY FROM
F6C4| 98                SEX1    TYA                   ; $7F TO Ø
F6C5| 29 FE                     AND     #ØFE          ; ADD.=
F6C7| 49 4E                     EOR     #4E           ; $4E OR $4F
F6C9| FØØ3                      BEQ     SEX2          ; YES, SKP
F6CB| B9 ØØCØ                    LDA     KYBD,Y        ; NO, CONT
F6CE| 88                SEX2    DEY                   ; NEXT ADD
F6CF| DØF3                      BNE     SEX1
```

```
F6D1| AD 51CØ                        LDA     TXTMD           ; SET TXT
F6D4| B9 ØØC1              SEX3      LDA     SLT1,Y          ; EXCERCISE
F6D7| B9 ØØC2                        LDA     SLT2,Y          ; ALL
F6DA| B9 ØØC3                        LDA     SLT3,Y          ; SLOTS
F6DD| B9 ØØC4                        LDA     SLT4,Y
F6EØ| AD FFCF                        LDA     EXPROM          ; DISABLE EXPANSION ROM AREA
F6E3| C8                             INY
F6E4| DØEE                           BNE     SEX3
F6E6|                      ;
F6E6|                      ; RAM TEST ROUTINE
F6E6|                      ;
F6E6| A9 73                USRENTRY  LDA     #72+ROM
F6E8| 8D DFFF                        STA     SYSD1
F6EB| A9 18                          LDA     #18
F6ED| 8D DØFF                        STA     ZPREG
F6FØ| A9 ØØ                          LDA     #ØØ
F6F2| A2 Ø7                          LDX     #Ø7
F6F4| 95 1Ø                RAMTSTØ   STA     ZRPG1,X
F6F6| CA                             DEX
F6F7| 1ØFB                          BPL     RAMTSTØ
F6F9| 2Ø 84F7                       JSR     RAMSET
F6FC| Ø8                           PHP
F6FD| 2Ø F6F7              RAMTST1   JSR     RAMWT
F7ØØ| 2Ø F6F7                       JSR     RAMWT
F7Ø3| 28                           PLP
F7Ø4| 6A                           ROR     A
F7Ø5| Ø8                           PHP
F7Ø6| 2Ø A1F7                       JSR     PTRINC
F7Ø9| DØF2                          BNE     RAMTST1
F7ØB| 2Ø 84F7                       JSR     RAMSET
F7ØE| Ø8                           PHP
F7ØF| 2Ø FAF7              RAMTST4   JSR     RAMRD
F712| 48                           PHA
F713| A9 ØØ                        LDA     #ØØ
F715| 91 18                        STA     (PTRLO),Y
F717| 68                           PLA
F718| 28                           PLP
F719| 6A                           ROR     A
F71A| Ø8                           PHP
F71B| 2Ø A1F7                       JSR     PTRINC
F71E| DØEF                          BNE     RAMTST4
F72Ø|                      ;
F72Ø|                      ; RETURN TO START
F72Ø|                      ;
F72Ø| A9 ØØ                        LDA     #ØØ
F722| 8D EFFF                        STA     BNKSW
F725| 8D DØFF                        STA     ZPREG
F728| A2 Ø7                          LDX     #Ø7
F72A| BD 1Ø18              RAMTST6   LDA     PHPR,X
F72D| 95 1Ø                        STA     ZRPG1,X
F72F| CA                           DEX
F73Ø| 1ØF8                          BPL     RAMTST6
F732| 2Ø 7EF7                       JSR     ERROR
F735| 4C 75F5                       JMP     ERRLP
F738|                      ;
F738|                      ;*****************************
F738|                      ; SARA TEST SUBROUTINES
F738|                      ;*****************************
F738|                      ;
F738| BD CDF4              STRWT     LDA     CHPG,X
F73B| 48                           PHA
F73C| Ø9 8Ø                        ORA     #8Ø             ; NORMAL VIDEO
F73E| 2Ø 25FC                       JSR     COUT            ; & PRINT
F741| E8                           INX                     ; NXT
F742| 68                           PLA                     ; CHR
F743| 1ØF3                          BPL     STRWT
F745| 4C Ø7FD                       JMP     CROUT1          ; CLR TO END OF LINE
F748|                      ;
F748|                      ; SUBROUTINE RAM
F748|                      ;
F748| 48                   RAM       PHA                     ; SV ACC
F749| 8A                           TXA                     ; CONVRT
F74A| 4A                           LSR     A               ; ADD TO
F74B| 4A                           LSR     A               ; USE FOR
F74C| 4A                           LSR     A               ; 8 ENTRY
F74D| 4A                           LSR     A
F74E| Ø8                           PHP
F74F| 4A                           LSR     A
F75Ø| 28                           PLP
F751| AA                           TAX                     ; LOOKUP
F752| BD C5F4                       LDA     RAMTBL,X        ; IF VAL
F755| 1Ø14                          BPL     RAMØ            ; <Ø, GET
F757| 48                           PHA                     ; WHICH
F758| AD EFFF                       LDA     BNKSW
F75B| 29 ØF                        AND     #ØF
F75D| AA                           TAX
F75E| 68                           PLA
F75F| EØ ØØ                        CPX     #ØØ
F761| FØ13                          BEQ     RAM1            ; BANK?
F763| 4A                           LSR     A               ; SET
```

```
F764| 4A                          LSR     A           ; PROPER
F765| 4A                          LSR     A           ; RAM
F766| CA                          DEX                 ; VAL
F767| D00D                        BNE     RAM1
F769| 29 05                       AND     #05         ; CONVERT
F76B| D009              RAM0      BNE     RAM1        ; TO VAL
F76D| 8A                          TXA
F76E| F002                        BEQ     RAM00
F770| A9 03                       LDA     #03
F772| 9002              RAM00     BCC     RAM1
F774| 49 03                       EOR     #03
F776| 29 07              RAM1      AND     #07         ; BANKSW
F778| AA                          TAX
F779| 68                          PLA
F77A| 60                          RTS
F77B|                   ;
F77B|                   ; SUBROUTINE ERROR
F77B|                   ;
F77B| 20 38F7           MESSERR   JSR     STRWT       ; PRINT MESSAGE FIRST
F77E| A9 F3             ERROR     LDA     #0F2+ROM    ; SET 1
F780| 8D DFFF                     STA     SYSD1       ; MHZ MO
F783| 60                          RTS
F784|                   ;
F784|                   ; SUBROUTINE RAMSET
F784|                   ;
F784| A2 01              RAMSET    LDX     #01
F786| 86 1A                       STX     BNK
F788| A0 00                       LDY     #00
F78A| A9 AA                       LDA     #0AA
F78C| 38                          SEC
F78D| 48                 RAMSET1   PHA
F78E| 08                          PHP
F78F| A5 1A                       LDA     BNK
F791| 09 80                       ORA     #80
F793| 8D 1914                     STA     IBNK
F796| A9 02                       LDA     #02
F798| 85 19                       STA     PTRHI
F79A| A2 00                       LDX     #00
F79C| 86 18                       STX     PTRLO
F79E| 28                          PLP
F79F| 68                          PLA
F7A0| 60                          RTS
F7A1|                   ;
F7A1|                   ; SUBROUTINE PTRINC
F7A1|                   ;
F7A1| 48                 PTRINC    PHA
F7A2| E6 18                       INC     PTRLO
F7A4| D01D                       BNE     RETS
F7A6| A5 1A                       LDA     BNK
F7A8| 100E                       BPL     PINC1
F7AA| A5 19                       LDA     PTRHI
F7AC| C9 13                       CMP     #13
F7AE| F006                       BEQ     PINC2
F7B0| C9 17                       CMP     #17
F7B2| D004                       BNE     PINC1
F7B4| E6 19                       INC     PTRHI
F7B6| E6 19              PINC2     INC     PTRHI
F7B8| E6 19              PINC1     INC     PTRHI
F7BA| D007                       BNE     RETS
F7BC| C6 1A                       DEC     BNK
F7BE| C6 1A                       DEC     BNK
F7C0| 20 8DF7                     JSR     RAMSET1
F7C3| 68                 RETS      PLA
F7C4| A6 1A                       LDX     BNK
F7C6| E0 FD                       CPX     #0FD
F7C8| 60                          RTS
F7C9|                   ;
F7C9|                   ; SUBROUTINE RAMERR
F7C9|                   ;
F7C9| 48                 RAMERR    PHA
F7CA| A6 19                       LDX     PTRHI
F7CC| A4 1A                       LDY     BNK
F7CE| 3019                       BMI     RAMERR4
F7D0| 8A                          TXA
F7D1| 301D                       BMI     RAMERR5
F7D3| 18                          CLC
F7D4| 69 20                       ADC     #20
F7D6| 8C EFFF           RAMERR2   STY     BNKSW
F7D9| AA                          TAX
F7DA| 20 48F7           RAMERR3   JSR     RAM
F7DD| 68                          PLA
F7DE| 48                          PHA
F7DF| A0 00                       LDY     #00
F7E1| 51 18                       EOR     (PTRLO),Y
F7E3| 15 10                       ORA     ZRPG1,X
F7E5| 95 10                       STA     ZRPG1,X
F7E7| 68                          PLA
F7E8| 60                          RTS
F7E9| A9 00             RAMERR4   LDA     #00
F7EB| 8D EFFF                     STA     BNKSW
```

```
F7EE| FØEA                          BEQ     RAMERR3
F7FØ| 38                    RAMERR5 SEC
F7F1| E9 6Ø                        SBC     #6Ø
F7F3| C8                          INY
F7F4| DØEØ                        BNE     RAMERR2
F7F6|                       ;
F7F6|                       ; SUBROUTINE RAMWT
F7F6|                       ;
F7F6| 49 FF                 RAMWT   EOR     #ØFF
F7F8| 91 18                        STA     (PTRLO),Y
F7FA| D1 18                 RAMRD   CMP     (PTRLO),Y
F7FC| DØCB                        BNE     RAMERR
F7FE| 6Ø                    RET1    RTS
F7FF|
F7FF|                             .END
```

SYMBOL TABLE DUMP

AB - Absolute LB - Label UD - Undefined MC - Macro
RF - Ref DF - Def PR - Proc FC - Func
PB - Public PV - Private CS - Consts

ACIA	LB F63E	ACIACM	AB CØF2	ACIACN	AB CØF3	ACIAST	AB CØF1	ADCERR	LB F676
ADCTST1	LB F663	ADCTST3	LB F669	ADRS	AB CØ47	ADTO	AB CØ66	ATD	LB F653
BLOCKIO	AB F479	BNK	AB ØØ1A	BNKSW	AB FFEF	BOOT	LB F6A1	CHPG	LB F4CD
CLDSTRT	AB FD98	CNTWR	LB F532	COUT	AB FC25	CROUT1	AB FDØ7	CV	AB ØØ5D
DISK1	LB F513	DISKOFF	AB CØDØ	ERRLP	LB F575	ERRLP1	LB F593	ERROR	LB F77E
EXPROM	AB CFFF	GOBOOT	LB F6BF	GRMD	AB CØ5Ø	IBBUFP	AB ØØ85	IBCMD	AB ØØ87
IBNK	AB 1419	KBDSTRB	AB CØ1Ø	KEYBD	AB CØØ8	KEYIN	AB FDØF	KEYPLUG	LB F67B
KYBD	AB CØØØ	MESSERR	LB F77B	MONITOR	AB F9Ø1	NMEM1	LB F54F	NMEM2	LB F562
NOGOOD	LB F52A	NOMEM	LB F548	NXBIT	LB F526	NXBYT	LB F524	PDLEN	AB CØ58
PHPR	AB 181Ø	PINC1	LB F7B8	PINC2	LB F7B6	PREVTRK	AB ØØ91	PTRHI	AB ØØ19
PTRINC	LB F7A1	PTRLO	AB ØØ18	PULBT	LB F53A	RAM	LB F748	RAMØ	LB F76B
RAMØØ	LB F772	RAM1	LB F776	RAMERR	LB F7C9	RAMERR2	LB F7D6	RAMERR3	LB F7DA
RAMERR4	LB F7E9	RAMERR5	LB F7FØ	RAMRD	LB F7FA	RAMSET	LB F784	RAMSET1	LB F78D
RAMTBL	LB F4C5	RAMTSTØ	LB F6F4	RAMTST1	LB F6FD	RAMTST4	LB F7ØF	RAMTST6	LB F72A
RAMWT	LB F7F6	RAMWT1	LB F5A1	RAMWT2	LB F5A5	RAMWT4	LB F5AD	RECON	LB F686
RET1	LB F7FE	RETS	LB F7C3	ROM	AB ØØØ1	ROMTST	LB F5E7	ROMTST1	LB F5F2
ROMTST2	LB F5FE	SARATEST	PR ----	SETCVH	AB FBC7	SETUP	AB FD9D	SEX	LB F6C2
SEX1	LB F6C4	SEX2	LB F6CE	SEX3	LB F6D4	SLT1	AB C1ØØ	SLT2	AB C2ØØ
SLT3	AB C3ØØ	SLT4	AB C4ØØ	STKØ	AB ØØFF	STRWT	LB F738	SYSD1	AB FFDF
SYSD2	AB FFD2	SYSD3	AB FFD3	SYSEØ	AB FFEØ	SYSE2	AB FFE2	SYSE3	AB FFE3
TXTMD	AB CØ51	USRENTRY	LB F6E6	VIATST	LB F6ØD	ZP1	LB F5BE	ZP2	LB F5D4
ZP3	LB F5E2	ZPREG	AB FFDØ	ZRPG	AB ØØØØ	ZRPG1	AB ØØ1Ø		

```
Assembly complete:       545 lines
Ø   Errors flagged on this Assembly
```

6502 OPCODE STATIC FREQUENCIES

```
    ADC :   1Ø  |  **********
    AND :   12  |  ************
    ASL :    3  |  ***
    BCC :    3  |  ***
    BCS :    1  m  *
    BEQ :   12  |  ************
    BIT :    1  m  *
    BMI :    4  |  ****
    BNE :   31  |  *******************************
    BPL :    9  |  *********
    CLC :    3  |  ***
    CLD :    1  m  *
    CMP :   1Ø  |  **********
    CPX :    5  |  *****
    CPY :    2  |  **
    DEC :    3  |  ***
    DEX :    9  |  *********
    DEY :    5  |  *****
    EOR :    5  |  *****
    INC :    6  |  ******
    INX :    6  |  ******
    INY :    6  |  ******
    JMP :    4  |  ****
    JSR :   29  |  *****************************
    LDA :   56  M  *******************************************************
    LDX :   24  |  ************************
    LDY :   1Ø  |  **********
    LSR :    9  |  *********
    ORA :    3  |  ***
    PHA :   11  |  ***********
    PHP :    6  |  ******
    PLA :   12  |  ************
    PLP :    4  |  ****
    ROR :    2  |  **
    RTS :    6  |  ******
    SBC :    1  m  *
```

```
    SEC :     2  |  **
    STA :    30  |  ******************************
    STX :    18  |  ******************
    STY :     4  |  ****
    TAX :     4  |  ****
    TAY :     2  |  **
    TXA :     6  |  ******
    TXS :     2  |  **
    TYA :     4  |  ****

    Minimum frequency =     1
    Maximum frequency =    56

    Average frequency =     8

    Unused opcodes:

    BRK  BVC  BVS  CLI  CLV  NOP  ROL  RTI  SED  SEI  TSX

    Program opcode usage:  80 %

-------------------------------------------------------------------------------------------------
(1.00) That's all, Folks ...
-------------------------------------------------------------------------------------------------
```

Source Code Listing

for

Apple ///

ROM - Monitor

David T. Craig
736 Edgewater
Wichita, Kansas 67230

```
0000|                    ;ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
0000|                    ;Ä APPLE /// ROM - MONITOR
0000|                    ;Ä COPYRIGHT 1979 BY APPLE COMPUTER, INC.
0000|                    ;ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
0000|
0000|                              .ABSOLUTE
0000|                              .PROC   MONITOR
0000|                              .ORG    ØF7FE
F7FE|                    ;
F7FE|                    ;
F7FE| 6Ø                 RET1       RTS
F7FF| 3F                           .BYTE   Ø3F
F8ØØ| E9 Ø1                        SBC     #Ø1
F8Ø2| FØFA                        BEQ     RET1
F8Ø4| E9 Ø1                        SBC     #Ø1
F8Ø6| FØF6                        BEQ     RET1
F8Ø8| E9 Ø1                        SBC     #Ø1
F8ØA| FØF2                        BEQ     RET1
F8ØC| E9 Ø1                        SBC     #Ø1
F8ØE| FØEE                        BEQ     RET1
F81Ø| E9 Ø1                        SBC     #Ø1
F812| FØEA                        BEQ     RET1
F814| E9 Ø1                        SBC     #Ø1
F816| FØE6                        BEQ     RET1
F818| E9 Ø1                        SBC     #Ø1
F81A| FØE2                        BEQ     RET1
F81C| E9 Ø1                        SBC     #Ø1
F81E| FØDE                        BEQ     RET1
F82Ø| E9 Ø1                        SBC     #Ø1
F822| FØDA                        BEQ     RET1
F824| E9 Ø1                        SBC     #Ø1
F826| FØD6                        BEQ     RET1
F828| E9 Ø1                        SBC     #Ø1
F82A| FØD2                        BEQ     RET1
F82C| E9 Ø1                        SBC     #Ø1
F82E| FØCE                        BEQ     RET1
F83Ø| E9 Ø1                        SBC     #Ø1
F832| FØCA                        BEQ     RET1
F834| E9 Ø1                        SBC     #Ø1
F836| FØC6                        BEQ     RET1
F838| E9 Ø1                        SBC     #Ø1
F83A| FØC2                        BEQ     RET1
F83C| E9 Ø1                        SBC     #Ø1
F83E| FØBE                        BEQ     RET1
F84Ø| E9 Ø1                        SBC     #Ø1
F842| FØBA                        BEQ     RET1
F844| E9 Ø1                        SBC     #Ø1
F846| FØB6                        BEQ     RET1
F848| E9 Ø1                        SBC     #Ø1
F84A| FØB2                        BEQ     RET1
F84C| E9 Ø1                        SBC     #Ø1
F84E| FØAE                        BEQ     RET1
F85Ø| E9 Ø1                        SBC     #Ø1
F852| FØAA                        BEQ     RET1
F854| E9 Ø1                        SBC     #Ø1
F856| FØA6                        BEQ     RET1
F858| E9 Ø1                        SBC     #Ø1
F85A| FØA2                        BEQ     RET1
F85C| E9 Ø1                        SBC     #Ø1
F85E| FØ9E                        BEQ     RET1
F86Ø| E9 Ø1                        SBC     #Ø1
F862| FØ9A                        BEQ     RET1
F864| E9 Ø1                        SBC     #Ø1
F866| FØ96                        BEQ     RET1
F868| E9 Ø1                        SBC     #Ø1
F86A| FØ92                        BEQ     RET1
F86C| E9 Ø1                        SBC     #Ø1
F86E| FØ8E                        BEQ     RET1
F87Ø| E9 Ø1                        SBC     #Ø1
F872| FØ8A                        BEQ     RET1
F874| E9 Ø1                        SBC     #Ø1
F876| FØ86                        BEQ     RET1
F878| E9 Ø1                        SBC     #Ø1
F87A| FØ82                        BEQ     RET1
F87C| E9 Ø1                        SBC     #Ø1
F87E| FØØ2                        BEQ     RET3
F88Ø| E9 Ø1                        SBC     #Ø1
F882| FØ7C                 RET3       BEQ     RET2
F884| E9 Ø1                        SBC     #Ø1
F886| FØ78                        BEQ     RET2
F888| E9 Ø1                        SBC     #Ø1
F88A| FØ74                        BEQ     RET2
F88C| E9 Ø1                        SBC     #Ø1
F88E| FØ7Ø                        BEQ     RET2
F89Ø| E9 Ø1                        SBC     #Ø1
F892| FØ6C                        BEQ     RET2
F894| E9 Ø1                        SBC     #Ø1
F896| FØ68                        BEQ     RET2
F898| E9 Ø1                        SBC     #Ø1
F89A| FØ64                        BEQ     RET2
```

```
F89C| E9 Ø1                        SBC     #Ø1
F89E| FØ6Ø                        BEQ     RET2
F8AØ| E9 Ø1                        SBC     #Ø1
F8A2| FØ5C                        BEQ     RET2
F8A4| E9 Ø1                        SBC     #Ø1
F8A6| FØ58                        BEQ     RET2
F8A8| E9 Ø1                        SBC     #Ø1
F8AA| FØ54                        BEQ     RET2
F8AC| E9 Ø1                        SBC     #Ø1
F8AE| FØ5Ø                        BEQ     RET2
F8BØ| E9 Ø1                        SBC     #Ø1
F8B2| FØ4C                        BEQ     RET2
F8B4| E9 Ø1                        SBC     #Ø1
F8B6| FØ48                        BEQ     RET2
F8B8| E9 Ø1                        SBC     #Ø1
F8BA| FØ44                        BEQ     RET2
F8BC| E9 Ø1                        SBC     #Ø1
F8BE| FØ4Ø                        BEQ     RET2
F8CØ| E9 Ø1                        SBC     #Ø1
F8C2| FØ3C                        BEQ     RET2
F8C4| E9 Ø1                        SBC     #Ø1
F8C6| FØ38                        BEQ     RET2
F8C8| E9 Ø1                        SBC     #Ø1
F8CA| FØ34                        BEQ     RET2
F8CC| E9 Ø1                        SBC     #Ø1
F8CE| FØ3Ø                        BEQ     RET2
F8DØ| E9 Ø1                        SBC     #Ø1
F8D2| FØ2C                        BEQ     RET2
F8D4| E9 Ø1                        SBC     #Ø1
F8D6| FØ28                        BEQ     RET2
F8D8| E9 Ø1                        SBC     #Ø1
F8DA| FØ24                        BEQ     RET2
F8DC| E9 Ø1                        SBC     #Ø1
F8DE| FØ2Ø                        BEQ     RET2
F8EØ| E9 Ø1                        SBC     #Ø1
F8E2| FØ1C                        BEQ     RET2
F8E4| E9 Ø1                        SBC     #Ø1
F8E6| FØ18                        BEQ     RET2
F8E8| E9 Ø1                        SBC     #Ø1
F8EA| FØ14                        BEQ     RET2
F8EC| E9 Ø1                        SBC     #Ø1
F8EE| FØ1Ø                        BEQ     RET2
F8FØ| E9 Ø1                        SBC     #Ø1
F8F2| FØØC                        BEQ     RET2
F8F4| E9 Ø1                        SBC     #Ø1
F8F6| FØØ8                        BEQ     RET2
F8F8| E9 Ø1                        SBC     #Ø1
F8FA| FØØ4                        BEQ     RET2
F8FC| E9 Ø1                        SBC     #Ø1
F8FE| FØØØ                        BEQ     RET2
F9ØØ| 6Ø                  RET2    RTS
F9Ø1|                     ;
F9Ø1|                     ;
F9Ø1| Ø058                SCRNLOC .EQU    58
F9Ø1|                     ;
F9Ø1| Ø058                LMARGIN .EQU    SCRNLOC
F9Ø1| Ø059                RMARGIN .EQU    SCRNLOC+1
F9Ø1| Ø05A                WINTOP  .EQU    SCRNLOC+2
F9Ø1| Ø05B                WINBTM  .EQU    SCRNLOC+3
F9Ø1| Ø05C                CH      .EQU    SCRNLOC+4
F9Ø1| Ø05D                CV      .EQU    SCRNLOC+5
F9Ø1| Ø05E                BAS4L   .EQU    SCRNLOC+6
F9Ø1| Ø05F                BAS4H   .EQU    SCRNLOC+7
F9Ø1| Ø06Ø                BAS8L   .EQU    SCRNLOC+8
F9Ø1| Ø061                BAS8H   .EQU    SCRNLOC+9
F9Ø1| Ø058                TBAS4L  .EQU    SCRNLOC+A
F9Ø1| Ø063                TBAS4H  .EQU    SCRNLOC+ØB
F9Ø1| Ø064                TBAS8L  .EQU    SCRNLOC+ØC
F9Ø1| Ø065                TBAS8H  .EQU    SCRNLOC+ØD
F9Ø1| Ø066                FORGND  .EQU    SCRNLOC+ØE
F9Ø1| Ø067                BKGND   .EQU    SCRNLOC+ØF
F9Ø1| Ø068                MODES   .EQU    SCRNLOC+1Ø
F9Ø1| Ø069                CURSOR  .EQU    SCRNLOC+11
F9Ø1| Ø06A                STACK   .EQU    SCRNLOC+12
F9Ø1| Ø06B                PROMPT  .EQU    SCRNLOC+13
F9Ø1| Ø06C                TEMPX   .EQU    SCRNLOC+14
F9Ø1| Ø06D                TEMPY   .EQU    SCRNLOC+15
F9Ø1| Ø06E                CSWL    .EQU    SCRNLOC+16
F9Ø1| Ø06F                CSWH    .EQU    SCRNLOC+17
F9Ø1| Ø07Ø                KSWL    .EQU    SCRNLOC+18
F9Ø1| Ø071                KSWH    .EQU    SCRNLOC+19
F9Ø1| Ø072                PCL     .EQU    SCRNLOC+1A
F9Ø1| Ø073                PCH     .EQU    SCRNLOC+1B
F9Ø1| Ø074                A1L     .EQU    SCRNLOC+1C
F9Ø1| Ø075                A1H     .EQU    A1L+1
F9Ø1| Ø076                A2L     .EQU    A1L+2
F9Ø1| Ø077                A2H     .EQU    A1L+3
F9Ø1| Ø078                A3L     .EQU    A1L+4
F9Ø1| Ø079                A3H     .EQU    A1L+5
F9Ø1| Ø07A                A4L     .EQU    A1L+6
```

```
F901| 007B                A4H         .EQU    A1L+7
F901| 007C                STATE       .EQU    A1L+8
F901| 007D                YSAV        .EQU    A1L+9
F901| 007E                INBUF       .EQU    A1L+0A
F901| 0080                TEMP        .EQU    A1L+0C
F901| 0069                MASK        .EQU    CURSOR
F901|                     ;
F901| C000                KBD         .EQU    0C000
F901| C010                KBDSTRB     .EQU    0C010
F901|                     ;
F901| 0358                USERADR     .EQU    358
F901| F479                BLOCKIO     .EQU    0F479
F901| F686                RECON       .EQU    0F686       ; AS OF 12/20/1979
F901| F4EE                DIAGN       .EQU    0F4EE
F901| 0050                INBUFLEN    .EQU    50          ; ONLY 80 BYTES ($3A0-$3EF)
F901| 0081                IBSLOT      .EQU    81
F901| 0082                IBDRVN      .EQU    IBSLOT+1
F901| 0085                IBBUFP      .EQU    IBSLOT+4
F901| 0087                IBCMD       .EQU    IBSLOT+6
F901|                     ;
F901| F901                ENTRY       .EQU    *
F901| BA                              TSX
F902| 86 6A                           STX     STACK
F904| D8                  MON         CLD                 ; MUST BE HEX MODE
F905| 20 4EFC                         JSR     BELL
F908| A6 6A               MONZ        LDX     STACK       ; RESTORE STACK TO ORIGINAL LOCATION
F90A| 9A                              TXS
F90B| A9 DF                           LDA     #0DF        ; PROMPT (APPLE) FOR SARA MONITOR
F90D| 85 6B                           STA     PROMPT
F90F| 20 D5FC                        JSR     GETLNZ      ; GET A LINE OF INPUT
F912| 20 67F9               SCAN        JSR     ZSTATE      ; SET REGULAR SCAN
F915| 20 2CF9               NXTINP      JSR     GETNUM      ; ATTEMPT TO READ HEX BYTE
F918| 84 7D                           STY     YSAV        ; STORE CURRENT INPUT POINTER
F91A| A0 12                           LDY     #12         ; 18 COMMANDS
F91C| 88                  CMDSRCH     DEY
F91D| 30E5                           BMI     MON         ; GIVE UP IF UNRECOGNIZABLE
F91F| D9 6CF9                        CMP     CMDTAB,Y    ; FOUND?
F922| D0F8                           BNE     CMDSRCH     ; NO KEEP LOOKING
F924| 20 5EF9                        JSR     TOSUB       ; PERFORM FUNCTION
F927| A4 7D                           LDY     YSAV        ; GET NEXT POINTER
F929| 4C 15F9                        JMP     NXTINP      ; DO NEXT COMMAND
F92C|                     ;
F92C| A2 00               GETNUM      LDX     #00         ; CLEAR A2
F92E| 86 76                           STX     A2L
F930| 86 77                           STX     A2H
F932| B1 7E               NXTCHR      LDA     (INBUF),Y
F934| C8                              INY                 ; BUMP INDEX FOR NEXT TIME
F935| 49 B0                           EOR     #0B0
F937| C9 0A                           CMP     #0A         ; TEST FOR DIGIT
F939| 9006                           BCC     DIGIT       ; SAVE IT IF 1-9
F93B| 69 88                           ADC     #88         ; TEST FOR HEX A-F
F93D| C9 FA                           CMP     #0FA
F93F| 902A                           BCC     DIGRET
F941| A2 03               DIGIT       LDX     #03
F943| 0A                              ASL     A
F944| 0A                              ASL     A
F945| 0A                              ASL     A
F946| 0A                              ASL     A
F947| 0A                  NXTBIT      ASL     A           ; SHIFT HEX DIGITS INTO A2
F948| 26 76                           ROL     A2L
F94A| 26 77                           ROL     A2H
F94C| CA                              DEX
F94D| 10F8                           BPL     NXTBIT      ; SHIFTED ALL YET?
F94F| A5 7C               NXTBAS      LDA     STATE
F951| D006                           BNE     NXTBS2      ; IF ZERO THEN COPY TO A1,3
F953| B5 77                           LDA     A2H,X
F955| 95 75                           STA     A1H,X
F957| 95 79                           STA     A3H,X
F959| E8                  NXTBS2      INX
F95A| F0F3                           BEQ     NXTBAS
F95C| D0D4                           BNE     NXTCHR
F95E|
F95E|                     ; SWITCH ROUTINE FOR CHARACTER
F95E|
F95E| A9 FA               TOSUB       LDA     #0FA        ; PUSH ADDRESS OR FUNCTION
F960| 48                              PHA                 ; AND RETURN IT
F961| B9 7DF9                        LDA     CMDVEC,Y
F964| 48                              PHA
F965| A5 7C                           LDA     STATE       ; PASS MODE VIA ACC.
F967| A0 00               ZSTATE      LDY     #00
F969| 84 7C                           STY     STATE       ; RESET STATE OF SCAN
F96B| 60                  DIGRET      RTS
F96C| F96C                CMDTAB      .EQU    *
F96C| 00                              .BYTE   00          ; G    =GP (CALL) SUBROUTINE
F96D| 03                              .BYTE   03          ; J    =JUMP (CONT) PROGRAM
F96E| 06                              .BYTE   06          ; M    =MOVE MEMORY
F96F| EB                              .BYTE   0EB         ; R    =READ DISK BLOCK
F970| EC                              .BYTE   0EC         ; S    =MEMORY SEARCH
F971| EE                              .BYTE   0EE         ; U    =USER FUNCTION
F972| EF                              .BYTE   0EF         ; V    =VERIFY MEMORY BLOCKS
```

```
F973| FØ                          .BYTE   ØFØ         ; W    =WRITE DISK BLOCK
F974| F1                          .BYTE   ØF1         ; X    =REPEAT COMMAND LINE
F975| 99                          .BYTE   99          ; SP   =SPACE (BYTE SEPARATOR)
F976| 9B                          .BYTE   9B          ; "    =ASCII (HI BIT ON)
F977| AØ                          .BYTE   ØAØ         ; '    =ASCII (HI BIT OFF)
F978| 93                          .BYTE   93          ; :    =SET STORE MODE
F979| A7                          .BYTE   ØA7         ; .    =RANGE SEPARATOR
F97A| A8                          .BYTE   ØA8         ; /    =COMMAND SEPARATOR
F97B| 95                          .BYTE   95          ; <    =DEST/SOURCE SEPARATOR
F97C| C6                          .BYTE   ØC6         ; CR   =CARRIAGE RETURN
F97D|                     ;
F97D| F97D                CMDVEC  .EQU    *
F97D| 9Ø                          .BYTE   9Ø          ; GO-1
F97E| 8E                          .BYTE   8E          ; JUMP-1
F97F| 3F                          .BYTE   3F          ; MOVE-1
F98Ø| D3                          .BYTE   ØD3         ; READ-1
F981| Ø8                          .BYTE   Ø8          ; SEARCH-1
F982| 8B                          .BYTE   8B          ; USER-1
F983| 4E                          .BYTE   4E          ; VRFY-1
F984| D6                          .BYTE   ØD6         ; WRTE-1
F985| 2C                          .BYTE   2C          ; REPEAT-1
F986| B7                          .BYTE   ØB7         ; SPCE-1
F987| 1A                          .BYTE   1A          ; ASCII-1
F988| 1C                          .BYTE   1C          ; ASCIIØ-1
F989| CB                          .BYTE   ØCB         ; SETMODE-1
F98A| CB                          .BYTE   ØCB         ; SETMODE-1
F98B| AD                          .BYTE   ØAD         ; SEP-1
F98C| A4                          .BYTE   ØA4         ; DEST-1
F98D| 39                          .BYTE   39          ; CRMON-1
F98E|                     ;
F98E|                     ;
F98E| E6 7A               NXTA4   INC     A4L         ; BUMP 16 BIT POINTERS
F99Ø| DØØ2                        BNE     NXTA1
F992| E6 7B                       INC     A4H
F994| E6 74               NXTA1   INC     A1L         ; BUMP A1
F996| DØØ5                        BNE     TSTA1
F998| E6 75                       INC     A1H
F99A| 38                          SEC                 ; IN CASE OF ROLL OVER
F99B| FØ1Ø                        BEQ     RETA1
F99D| A5 74               TSTA1   LDA     A1L
F99F| 38                          SEC
F9AØ| E5 76                       SBC     A2L
F9A2| 85 8Ø                       STA     TEMP
F9A4| A5 75                       LDA     A1H
F9A6| E5 77                       SBC     A2H
F9A8| Ø5 8Ø                       ORA     TEMP
F9AA| DØØ1                        BNE     RETA1       ; IF A1 LESS THAN OR EQUAL TO A2
F9AC| 18                          CLC                 ; THEN CARRY CLEAR ON RETURN
F9AD| 6Ø                 RETA1   RTS
F9AE|                     ;
F9AE|                     ;
F9AE| 48                 PRBYTE  PHA                 ; SAVE LOW NIBBLE
F9AF| 4A                          LSR     A
F9BØ| 4A                          LSR     A           ; SHIFT HI NIBBLE TO PRINT.
F9B1| 4A                          LSR     A
F9B2| 4A                          LSR     A
F9B3| 2Ø B9F9                     JSR     PRHEXZ
F9B6| 68                          PLA
F9B7| 29 ØF               PRHEX   AND     #ØF         ; STRIP HI NIBBLE
F9B9| Ø9 BØ               PRHEXZ  ORA     #ØBØ        ; MAKE IT NUMERIC
F9BB| C9 BA                       CMP     #ØBA        ; IS IT >'9'
F9BD| 9ØØ2                       BCC     PRHEX2
F9BF| 69 Ø6                       ADC     #Ø6         ; MAKE IT 'A'-'F'
F9C1| 4C 39FC            PRHEX2  JMP     COUT
F9C4|                     ;
F9C4| 2Ø AEF9            PRBYCOL JSR     PRBYTE
F9C7|                     ;
F9C7| A9 BA               PRCOLON LDA     #ØBA        ; PRINT A COLON
F9C9| DØF6                        BNE     PRHEX2      ; BRANCH ALWAYS
F9CB|                     ;
F9CB| A9 Ø7               TST8ØWID LDA    #Ø7         ; ANTICIPATE
F9CD| 24 68                       BIT     MODES       ; TEST FOR 8Ø
F9CF| 5ØØ2                       BVC     SVMASK
F9D1| A9 ØF                       LDA     #ØF
F9D3| 85 69               SVMASK  STA     MASK
F9D5| 6Ø                          RTS
F9D6|                     ;
F9D6| 8A                 A1PC    TXA                 ; TEST FOR NEW PC
F9D7| FØØ7                        BEQ     OLDPC
F9D9| B5 74               A1PC1   LDA     A1L,X
F9DB| 95 72                       STA     PCL,X
F9DD| CA                          DEX
F9DE| 1ØF9                       BPL     A1PC1
F9EØ| 6Ø                 OLDPC   RTS
F9E1|                     ;
F9E1| 85 69               ASCII1  STA     MASK        ; SAVE HI BIT STATUS
F9E3| A4 7D               ASCII2  LDY     YSAV        ; MOVE ASCII TO MEMORY
F9E5| B1 7E                       LDA     (INBUF),Y
F9E7| E6 7D                       INC     YSAV        ; BUMP FOR NEXT THING.
F9E9| AØ ØØ                       LDY     #ØØ
```

```
F9EB| C9 A2                        CMP     #ØA2        ; ASCII " ?
F9ED| DØØ5                        BNE     ASCII3      ; NOPE, CONTINUE.
F9EF| A5 69                       LDA     MASK
F9F1| 1Ø32                        BPL     BITON       ; HE'S CHANGED MODES.
F9F3| 6Ø                          RTS
F9F4| C9 A7              ASCII3   CMP     #ØA7        ; ASCII ' ?
F9F6| DØØ5                        BNE     CRCHK       ; NO, TEST FOR EOL.
F9F8| A5 69                       LDA     MASK
F9FA| 3Ø2D                        BMI     BITOFF      ; CHANGE MODES.
F9FC| 6Ø                          RTS
F9FD|
F9FD| C9 8D              CRCHK    CMP     #8D         ; END OF LINE?
F9FF| FØØ7                        BEQ     ASCDONE     ; YES, FINISHED
FAØ1| 25 69                       AND     MASK
FAØ3| 2Ø C3FA                     JSR     STOR1       ; GO STORE IT!
FAØ6| DØDB                        BNE     ASCII2      ; DO NEXT.
FAØ8| 6Ø                ASCDONE  RTS
FAØ9|                   ;
FAØ9|                   ;
FAØ9| B1 74              SEARCH   LDA     (A1L),Y     ; LOAD SEARCH BYTE
FAØB| C5 7A                       CMP     A4L
FAØD| DØØ6                        BNE     SRCH1
FAØF| 2Ø 75FA                     JSR     PRINTA1     ; DUMP MEMORY
FA12| 2Ø EFFC                     JSR     CROUT
FA15| 2Ø 94F9            SRCH1    JSR     NXTA1       ; INCREMENT POINTER
FA18| 9ØEF                        BCC     SEARCH      ; CONTINUE SEARCH
FA1A| 6Ø                          RTS                 ; RETURN
FA1B|                   ;
FA1B|                   ;
FA1B| 38                ASCII    SEC                 ; INDICATE HI ON.
FA1C| 9Ø                         .BYTE   9Ø          ; (BCC - NEVER TAKEN)
FA1D| 18                ASCIIØ   CLC                 ; INDICATE HI OFF
FA1E| AA                CKMDE    TAX                 ; SAVE STATE
FA1F| 86 7C                      STX     STATE       ; RETAIN STATE
FA21| 49 BA                      EOR     #ØBA        ; ARE WE IN STORE MODE?
FA23| DØ7D                        BNE     ERROR
FA25| A9 FF              BITON    LDA     #ØFF        ; SET HI BIT UNMASKED
FA27| BØB8                        BCS     ASCII1
FA29| A9 7F              BITOFF   LDA     #7F         ; MASK HI BIT
FA2B| 10B4                        BPL     ASCII1      ; ALWAYS BRANCHES
FA2D| 2C ØØCØ            REPEAT   BIT     KBD         ; REPEAT UNTIL KEYPRESS
FA3Ø| 1ØØ3                        BPL     REPEAT1
FA32| 4C ØFFD                     JMP     KEYIN
FA35| 68                REPEAT1  PLA                 ; CLEAN UP STACK
FA36| 68                LFA36    PLA
FA37| 4C 12F9                     JMP     SCAN
FA3A|                   ;
FA3A|                   ;
FA3A| 2Ø B4FA            CRMON    JSR     BL1
FA3D| 4C Ø8F9                     JMP     MONZ
FA4Ø|                   ;
FA4Ø|                   ;
FA4Ø| 2Ø 9DF9            MOVE     JSR     TSTA1       ; TEST VALID RANGE
FA43| BØ5D                        BCS     ERROR
FA45| B1 74              MOVNXT   LDA     (A1L),Y     ; COMPARE BYTE FOR BYTE
FA47| 91 7A                      STA     (A4L),Y
FA49| 2Ø 8EF9                     JSR     NXTA4       ; BUMP BOTH A1 AND A4
FA4C| 9ØF7                        BCC     MOVNXT
FA4E| 6Ø                          RTS                 ; ALL DONE WITH MOVE
FA4F|                   ;
FA4F|                   ;
FA4F| 2Ø 9DF9            VRFY     JSR     TSTA1       ; TEST VALID RANGE
FA52| BØ4E                        BCS     ERROR
FA54| B1 74              VRFY1    LDA     (A1L),Y     ; COMPARE BYTE FOR BYTE
FA56| D1 7A                      CMP     (A4L),Y     ; MATCH?
FA58| FØØ6                        BEQ     VRFY2       ; YES, DO NEXT.
FA5A| 2Ø 66FA                     JSR     MISMATCH    ; PRINT BOTH BYTES
FA5D| 2Ø EFFC                     JSR     CROUT       ; GOTO NEWLINE
FA6Ø| 2Ø 8EF9            VRFY2    JSR     NXTA4       ; BUMP BOTH A1 AND A4
FA63| 9ØEF                        BCC     VRFY1
FA65| 6Ø                          RTS                 ; VERIFY DONE.
FA66|                   ;
FA66| A5 7B              MISMATCH LDA     A4H         ; PRINT ADDRESS OF A4
FA68| 2Ø AEF9                     JSR     PRBYTE
FA6B| A5 7A                       LDA     A4L
FA6D| 2Ø C4F9                     JSR     PRBYCOL     ; OUTPUT A COLON FOR SEPARATOR
FA7Ø| B1 7A                       LDA     (A4L),Y     ; AND THE DATA
FA72| 2Ø 84FA                     JSR     PRBYTSP     ; PRINT THE BYTE AND A SPACE
FA75| 2Ø 87FA            PRINTA1  JSR     PRSPC       ; LEAD WITH A SPACE
FA78| A5 75                       LDA     A1H         ; OUTPUT ADDRESS A1
FA7A| 2Ø AEF9                     JSR     PRBYTE
FA7D| A5 74                       LDA     A1L
FA7F| 2Ø C4F9                     JSR     PRBYCOL     ; SEPARATE WITH A COLON
FA82| B1 74              PRA1BYTE LDA     (A1L),Y     ; PRINT BYTE POINTED TO BY A1
FA84| 2Ø AEF9            PRBYTSP  JSR     PRBYTE
FA87| A9 AØ              PRSPC    LDA     #ØAØ        ; PRINT A SPACE
FA89| 4C 39FC                     JMP     COUT        ; END VIA OUTPUT ROUTINE.
FA8C|                   ;
FA8C| 4C 58Ø3            USER     JMP     USERADR
FA8F|                   ;
```

```
FA8F| 68                  JUMP      PLA
FA9Ø| 68                            PLA                 ; LEAVE STACK WITH NOTHIN' ON IT.
FA91| 2Ø D6F9              GO        JSR     A1PC        ; STUFF PROGRAM COUNTER
FA94| 6C 72ØØ                       JMP     @PCL        ; JUMP TO USER PROG.
FA97|                     ;
FA97| FA97                RWERROR   .EQU    *           ; PRINT ERROR NUMBER
FA97| 2Ø AEF9                       JSR     PRBYTE      ; PRINT THE OFFENDER
FA9A| A9 A1                         LDA     #ØA1        ; FOLLOWED BY A "!"
FA9C| 2Ø 39FC                       JSR     COUT
FA9F| 2Ø Ø7FD              ERROR2    JSR     NOSTOP      ; OUTPUT A CARRIAGE RETURN (NO STOPLST)
FAA2| 4C Ø4F9              ERROR     JMP     MON
FAA5|                     ;
FAA5| A5 76                DEST      LDA     A2L         ; COPY A2 TO A4 FOR DESTINATION OP
FAA7| 85 7A                         STA     A4L
FAA9| A5 77                         LDA     A2H
FAAB| 85 7B                         STA     A4H
FAAD| 6Ø                           RTS
FAAE|                     ;
FAAE| 2Ø B8FA              SEP       JSR     SPCE        ; SEPARATOR TEST STORE MODE OR DUMP.
FAB1| 98                           TYA                 ; ZERO MODE.
FAB2| FØ1D                         BEQ     SETMDZ      ; BRANCH ALWAYS
FAB4|                     ;
FAB4| C6 7D                BL1       DEC     YSAV        ; TEST FOR NO LINE
FAB6| FØ45                         BEQ     DUMP8       ; IF NO LINE, GIVEM A ROW OF BYTES
FAB8| CA                  SPCE      DEX                 ; TEST IF AFTER ANOTHER SPACE
FAB9| DØ16                         BNE     SETMDZ
FABB| C9 BA                         CMP     #ØBA        ; STORE MODE?
FABD| DØ4B                         BNE     TSTDUMP
FABF| 85 7C                STOR      STA     STATE       ; KEEP IT IN STORE STATE
FAC1| A5 76                         LDA     A2L         ; GET BYTE TO BE STORED
FAC3| 91 78                STOR1     STA     (A3L),Y     ; PUT IT IN MEMORY.
FAC5| E6 78                         INC     A3L         ; BUMP POINTER
FAC7| DØØ2                         BNE     DUMMY
FAC9| E6 79                         INC     A3H
FACB| 6Ø                    DUMMY     RTS                 ; ALSO USED FOR '/' TO CLEAR MODE
FACC|                     ;
FACC| A4 7D                SETMODE   LDY     YSAV        ; USE INPUT CHARACTER
FACE| 88                           DEY
FACF| B1 7E                         LDA     (INBUF),Y   ; TO SET MODE
FAD1| 85 7C                SETMDZ    STA     STATE
FAD3| 6Ø                           RTS
FAD4|                     ;
FAD4| A9 Ø1                READ      LDA     #Ø1         ; GET DISK COMMAND TO READ
FAD6| 2C                            .BYTE   2C          ; DUMMY BIT TO SKIP 2 BYTES
FAD7| A9 Ø2                WRTE      LDA     #Ø2         ; SET DISK COMMAND TO WRITE
FAD9| 85 87                SAVCMD    STA     IBCMD
FADB| A5 74                RWLOOP    LDA     A1L
FADD| 85 85                         STA     IBBUFP      ; COMMAND FORMAT IS
FADF| A5 75                         LDA     A1H         ; BLOCKNUMBER <ADDRESS END ADDRESS
FAE1| 85 86                         STA     IBBUFP+1
FAE3| A6 7B                         LDX     A4H         ; SEND BLOCK NUMBER VIA X & A
FAE5| A5 7A                         LDA     A4L
FAE7| 78                           SEI                 ; NO INTERRUPTS WHILE IN MONITOR
FAE8| 2Ø 79F4                       JSR     BLOCKIO     ; DO DISKO FEVER
FAEB| BØAA                         BCS     RWERROR     ; GIVE UP IF ERROR ENCOUNTERED
FAED| E6 7A                         INC     A4L         ; BUMP BLOCK NUMBER
FAEF| DØØ2                         BNE     NOVER
FAF1| E6 7B                         INC     A4H
FAF3| E6 75                NOVER     INC     A1H         ; BUMP RAM ADDRESS BY 512 BYTES
FAF5| E6 75                         INC     A1H
FAF7| 2Ø 9DF9                       JSR     TSTA1       ; TEST FOR FINISHED
FAFA| 9ØDF                         BCC     RWLOOP      ; NOT DONE, DO NEXT BLOCK
FAFC| 6Ø                           RTS
FAFD|                     ;
FAFD| A5 75                DUMP8     LDA     A1H
FAFF| 85 77                         STA     A2H
FBØ1| 2Ø CBF9                       JSR     TST8ØWID    ; GET WIDTH MASK INTO ACC
FBØ4| Ø5 74                         ORA     A1L
FBØ6| 85 76                         STA     A2L
FBØ8| DØØ6                         BNE     DUMPØ       ; BRANCH ALWAYS
FBØA|                     ;
FBØA| 4A                  TSTDUMP   LSR     A           ; DUMP?
FBØB| BØ95                ERROR1    BCS     ERROR
FBØD| 2Ø CBF9              DUMP      JSR     TST8ØWID    ; SET FOR EITHER 8Ø OR 4Ø COLUMNS
FB1Ø| A5 74                DUMPØ     LDA     A1L
FB12| 85 7A                         STA     A4L
FB14| A5 75                         LDA     A1H
FB16| 85 7B                         STA     A4H
FB18| 2Ø 9DF9                       JSR     TSTA1       ; TEST FOR VALID RANGE
FB1B| BØEE                         BCS     ERROR1
FB1D| 2Ø 75FA              DUMP1     JSR     PRINTA1     ; PRINT ADDRESS AND FIRST BYTE
FB2Ø| 2Ø 94F9              DUMP2     JSR     NXTA1
FB23| BØ1Ø                         BCS     DUMPASC     ; END WITH ASCII
FB25| A5 74                         LDA     A1L         ; TEST END OF LINE
FB27| 25 69                         AND     MASK        ; FOR 4Ø/8Ø COLUMN
FB29| DØØ5                         BNE     DUMP3
FB2B| 2Ø 35FB                       JSR     DUMPASC
FB2E| DØED                         BNE     DUMP1       ; BRANCH ALWAYS
FB3Ø| 2Ø 82FA              DUMP3     JSR     PRA1BYTE    ; GO PRINT NEXT BYTE AND A SPACE
FB33| DØEB                         BNE     DUMP2       ; ALWAYS (ACC JUST PULLED AS $AØ)
```

```
FB35|                           ;
FB35| A5 7A                     DUMPASC  LDA     A4L             ; RESET TO BEGINNING OF LINE
FB37| 85 74                              STA     A1L
FB39| A5 7B                              LDA     A4H
FB3B| 85 75                              STA     A1H
FB3D| 20 87FA                            JSR     PRSPC           ; PRINT AN EXTRA SPACE
FB40| A0 00                     ASC1     LDY     #00             ; TO INDEX MEMORY INDIRECT
FB42| B1 74                              LDA     (A1L),Y
FB44| 09 80                              ORA     #80             ; SET NORMAL VIDEO
FB46| C9 A0                              CMP     #0A0            ; TEST FOR CONTROL CHARACTERS
FB48| B002                              BCS     ASC2            ; OK TO PRINT NON CONTROLS
FB4A| A9 AE                              LDA     #0AE            ; OTHERWISE PRINT A SPACE
FB4C| 20 39FC                  ASC2     JSR     COUT            ; PUT IT OUT
FB4F| 20 8EF9                            JSR     NXTA4           ; BUMP BOTH A1 AND A4
FB52| B006                              BCS     ASC3            ; FINISHED
FB54| A5 74                              LDA     A1L             ; TEST END OF LINE
FB56| 25 69                              AND     MASK
FB58| D0E6                              BNE     ASC1            ; NOT DONE, PRINT NEXT
FB5A| 4C EFFC                  ASC3     JMP     CROUT
FB5D|                           ;
FB5D|                           ;
FB5D| 38                      COL80    SEC                     ; INDICATE 80 COLUMNS
FB5E| AD 53C0                            LDA     0C053           ; GOTO 80 COLUMN MODE
FB61| B004                              BCS     SET80           ; BRANCH ALWAYS
FB63|                           ;
FB63| 18                      COL40    CLC                     ; INDICATE 40 COLUMNS DESIRED
FB64| AD 52C0                            LDA     0C052           ; GOTO 40 COLUMN MODE
FB67| A5 68                     SET80    LDA     MODES
FB69| 09 40                              ORA     #40             ; ASSUME 80
FB6B| B002                              BCS     SET80A          ; AND BRANCH IF IT IS
FB6D| 29 BF                              AND     #0BF            ; BUT FIX FOR 40 IF NOT
FB6F| 85 68                     SET80A   STA     MODES
FB71| 09 7F                              ORA     #7F             ; ISOLATE BIT 7
FB73| 29 A0                              AND     #0A0            ; (BIT 7 SETS NORMAL/INVERSE)
FB75| 85 66                              STA     FORGND
FB77| B002                              BCS     SET80B          ; AGAIN ASSUMES 80 COLUMNS
FB79| A9 F0                              LDA     #0F0            ; IF NOT, SET FOR/BACKGROUND COLOR
FB7B| 85 67                     SET80B   STA     BKGND
FB7D|                           ;
FB7D| A5 58                     CLSCRN   LDA     LMARGIN         ; SET CURSOR TO TOP LEFT OF WINDOW
FB7F| 85 5C                              STA     CH
FB81| A5 5A                              LDA     WINTOP
FB83| 85 5D                              STA     CV              ; NOW DROP INTO CLEAR END OF PAGE
FB85|                           ;
FB85| A5 5C                     CLEOP    LDA     CH              ; SAVE CURRENT CURSOR POSITION
FB87| 48                                PHA
FB88| A5 5D                              LDA     CV
FB8A| 48                                PHA
FB8B| 20 C5FB                            JSR     SETCV
FB8E| 20 A2FB                  CLEOP1   JSR     CLEOL           ; CLEAR TO END OF FIRST LINE
FB91| A5 58                              LDA     LMARGIN
FB93| 85 5C                              STA     CH
FB95| 20 DDFB                            JSR     CURDOWN         ; GOTO NEXT LINE
FB98| 90F4                              BCC     CLEOP1
FB9A| 68                                PLA
FB9B| A8                                TAY
FB9C| 68                                PLA                     ; RESTORE CURSOR POSITION
FB9D| 85 5C                              STA     CH
FB9F| 98                                TYA                     ; GET OLD CV IN ACC AGAIN
FBA0| B023                              BCS     SETCV           ; BRANCH ALWAYS
FBA2|                           ;
FBA2| A5 5C                     CLEOL    LDA     CH              ; CLEAR TO END OF LINE FIRST
FBA4| 4C 89FC                            JMP     CLEOL1
FBA7|                           ;
FBA7| C9 80                     CONTROL  CMP     #80
FBA9| 9065                              BCC     DISPLAYX        ; IF INVERSE
FBAB| C9 8D                     TSTCR    CMP     #8D             ; IF CARRIAGE RETURN THEN NEW LINE
FBAD| D03A                              BNE     TSTBACK
FBAF| 20 A2FB                  CARRAGE  JSR     CLEOL           ; FIRST CLEAR TO THE END OF THIS LINE
FBB2| 20 D7FB                            JSR     SETCHZ          ; RESET CURSOR AND GOTO NEXT LINE (CARRY IS SET)
FBB5| 4C 16FC                            JMP     NXTLIN          ; THEN GOTO THE NEXT LINE.
FBB8|                           ;
FBB8|                           ;
FBB8| A5 5D                     CURUP    LDA     CV              ; TEST FOR TOP OF SCREEN
FBBA| C6 5D                              DEC     CV              ; ANTICIPATE 'NOT' TOP
FBBC| C5 5A                              CMP     WINTOP
FBBE| D002                              BNE     CURUP1          ; IT'S NOT TOP, CONTINUE
FBC0| A5 5B                              LDA     WINBTM          ; WRAP AROUND TO BOTTOM
FBC2| 38                      CURUP1   SEC                     ; DECREMENT BY ONE
FBC3| E9 01                              SBC     #01
FBC5| 85 5D                     SETCV    STA     CV              ; SAVE NEW VERTICAL LINE
FBC7| FBC7                     BASCALC  .EQU    *
FBC7| FBC7                     CURDN1   .EQU    *
FBC7| A5 5D                              LDA     CV              ; GET VALUES FOR FIRST PAGE ($400)
FBC9| 104E                              BPL     BASCALC1        ; ALWAYS
FBCB|                           ;
FBCB| 24 68                     CURIGHT  BIT     MODES           ; TEST FOR 80 OR 40
FBCD| 7002                              BVS     RIGHT1
FBCF| E6 5C                              INC     CH
FBD1| E6 5C                     RIGHT1   INC     CH              ; BUMP CURSOR HORIZONTAL
```

```
FBD3| A5 5C                        LDA     CH              ; TEST FOR NEW LINE
FBD5| C5 59                        CMP     RMARGIN
FBD7| A5 58              SETCHZ    LDA     LMARGIN         ; JUST IN CASE WE HAVE.
FBD9| 905D                        BCC     CTRLRET
FBDB| 85 5C              SETCVH    STA     CH              ; CURSOR AT START OF NEXT LINE
FBDD|                    ; DROP INTO CURDOWN FOR WRAP AROUND
FBDD|                    ;
FBDD| E6 5D              CURDOWN   INC     CV              ; MOVE CURSOR DOWN ONE LINE
FBDF| A5 5D                        LDA     CV              ; ANTICIPATE NOT BOTTOM
FBE1| C5 5B                        CMP     WINBTM          ; TEST FOR BOTTOM
FBE3| 90E2                        BCC     CURDN1
FBE5| A5 5A                        LDA     WINTOP
FBE7| B0DC                        BCS     SETCV           ; BRANCH ALWAYS
FBE9|                    ;
FBE9| C9 88              TSTBACK   CMP     #88             ; BACKSPACE?
FBEB| D05D                        BNE     TSTBELL
FBED| 24 68              CURLEFT   BIT     MODES           ; TEST FOR FOURTY OR EIGHTY MODE
FBEF| 7002                        BVS     LEFT80
FBF1| C6 5C                        DEC     CH
FBF3| C6 5C              LEFT80    DEC     CH
FBF5| 3006                        BMI     LEFTUP
FBF7| A5 5C                        LDA     CH              ; TEST FOR WRAP AROUND
FBF9| C5 58                        CMP     LMARGIN
FBFB| 103B                        BPL     CTRLRET
FBFD| 20 B8FB            LEFTUP    JSR     CURUP
FC00| A5 59                        LDA     RMARGIN
FC02| 85 5C                        STA     CH              ; SAVE NEW CURSOR POSITION
FC04| D0E7                        BNE     CURLEFT         ; BRANCH ALWAYS
FC06|                    ;
FC06| C9 A0              COUT2     CMP     #0A0            ; IS IT CONTROL CHARACTER
FC08| 909D                        BCC     CONTROL
FC0A| 24 68                       BIT     MODES           ; TEST FOR INVERSE
FC0C| 3002                        BMI     DISPLAYX        ; NO PUT IT OUT
FC0E| 29 7F                       AND     #7F             ; STRIP HI BIT
FC10| 20 9DFC            DISPLAYX  JSR     DISPLAY
FC13|                    ;
FC13| 20 CBFB            INCHORZ   JSR     CURIGHT         ; MOVE CURSOR RIGHT
FC16| B043              NXTLIN    BCS     SCROLL          ; IT'S BOTTOM, RESET CH=0 AND SCROLL
FC18| 60                          RTS                     ; RESET CH ONLY
FC19|                    ;
FC19| 08                BASCALC1  PHP                     ; CALC BASE ADR IN BAS4L,H
FC1A| 48                          PHA
FC1B| 4A                          LSR     A               ; FOR GIVEN LINE NO.
FC1C| 29 03                       AND     #03             ; 0<=LINE NO.<$17
FC1E| 09 04                       ORA     #04             ; ARG=000ABCDE, GENERATE
FC20| 85 5F                       STA     BAS4H           ; BAS4H=000001CD
FC22| 49 0C                       EOR     #0C
FC24| 85 61                       STA     BAS8H
FC26| 68                          PLA                     ; AND
FC27| 29 18                       AND     #18             ; BAS4L=EABAB000
FC29| 9002                        BCC     BSCLC2
FC2B| 69 7F                       ADC     #7F
FC2D| 85 5E              BSCLC2    STA     BAS4L
FC2F| 0A                          ASL     A
FC30| 0A                          ASL     A
FC31| 05 5E                       ORA     BAS4L
FC33| 85 5E                       STA     BAS4L
FC35| 85 60                       STA     BAS8L           ; SAME FOR PAGE 2
FC37| 28                          PLP
FC38| 60                CTRLRET   RTS
FC39|                    ;
FC39| 48                COUT      PHA                     ; SAVE CHARACTER
FC3A| 84 6D                       STY     TEMPY
FC3C| 86 6C                       STX     TEMPX
FC3E| 20 47FC                     JSR     COUT1
FC41| A4 6D                       LDY     TEMPY
FC43| A6 6C                       LDX     TEMPX
FC45| 68                          PLA
FC46| 60                          RTS
FC47| 6C 6E00            COUT1     JMP     @CSWL           ; NORMALLY COUT1
FC4A|                    ;
FC4A| C9 87              TSTBELL   CMP     #87             ; BELL?
FC4C| D004                        BNE     LNFD            ; NO TEST FOR FORM FEED
FC4E| AE 40C0            BELL      LDX     0C040           ; SOUND BELL
FC51| 60                          RTS
FC52| C9 8A              LNFD      CMP     #8A             ; LINE FEED?
FC54| D0E2                        BNE     CTRLRET
FC56| 20 DDFB                     JSR     CURDOWN         ; MOVE CURSOR DOWN A LINE
FC59| 90DD                        BCC     CTRLRET         ; BRANCH IF NO SCROLL NECESSARY.
FC5B|                    ;
FC5B| A5 5A              SCROLL    LDA     WINTOP          ; START WITH TOP LINE
FC5D| 48                          PHA                     ; SAVE IT FOR NOW
FC5E| 20 C5FB                     JSR     SETCV           ; GET BASCALC FOR THIS LINE
FC61| A2 03              SCRL1     LDX     #03             ; MOVE CURRENT BASCALC AS DESTINATION
FC63| B5 5E              SCRL2     LDA     BAS4L,X
FC65| 95 58                       STA     TBAS4L,X        ; (TEMPORARY BASE ADDR.)
FC67| CA                          DEX
FC68| 10F9                        BPL     SCRL2
FC6A| 68                          PLA                     ; GET DESTINATION LINE
FC6B| 18                          CLC
```

```
FC6C| 69 Ø1                     ADC     #Ø1           ; CALCULATE SOURCE LINE.
FC6E| C5 5B                     CMP     WINBTM        ; IS IT THE LAST LINE?
FC7Ø| BØ15                     BCS     LASTLN        ; YES, CLEAR IT
FC72| 48                       PHA                   ; SAVE AS NEXT DESTINATION LINE
FC73| 2Ø C5FB                   JSR     SETCV         ; GET BASE ADDR FOR SOURCE LINE
FC76| A5 59                     LDA     RMARGIN       ; MOVE SOURCE TO DESTINATION
FC78| 4A                       LSR     A             ; DIVIDE BY 2
FC79| A8                       TAY
FC7A| 88              SCRL3    DEY                   ; DONE YET
FC7B| 3ØE4                     BMI     SCRL1         ; YES, DO NEXT LINE
FC7D| B1 5E                     LDA     (BAS4L),Y
FC7F| 91 58                     STA     (TBAS4L),Y
FC81| B1 6Ø                     LDA     (BAS8L),Y
FC83| 91 64                     STA     (TBAS8L),Y
FC85| 9ØF3                     BCC     SCRL3         ; BRANCH ALWAYS
FC87| A5 58              LASTLN  LDA     LMARGIN       ; BLANK FILL THE LAST LINE
FC89| 4A                CLEOL1  LSR     A             ; DIVIDE BY 2
FC8A| A8                       TAY
FC8B| BØØ4                     BCS     CLEOL2
FC8D| A5 66                     LDA     FORGND        ; (NORMALLY A SPACE)
FC8F| 91 5E                     STA     (BAS4L),Y
FC91| A5 67              CLEOL2  LDA     BKGND         ; (IF 8Ø COLUMNS, ALSO A SPACE)
FC93| 91 6Ø                     STA     (BAS8L),Y
FC95| C8                       INY
FC96| 98                       TYA                   ; TEST FOR END OF LINE
FC97| ØA                       ASL     A             ; MULT BY 2 AGAIN
FC98| C5 59                     CMP     RMARGIN
FC9A| 9ØED                     BCC     CLEOL1        ; CONTINUE IF MORE TO DO.
FC9C| 6Ø                       RTS                   ; ALL DONE.
FC9D|                   ;
FC9D| 24 68             DISPLAY BIT     MODES         ; TEST FOR 4Ø OR 8Ø
FC9F| 7ØØC                     BVS     DSPL8Ø        ; STORE THE SINGLE CHARACTERS AND RETURN
FCA1| 46 5C                     LSR     CH            ; INSURE PROPER 4Ø COLUMN DISPLAY
FCA3| Ø6 5C                     ASL     CH            ; BY DROPPING BIT Ø
FCA5| 2Ø ADFC                   JSR     DSPL8Ø        ; DISPLAY IN $4ØØ PAGE.
FCA8| A5 67                     LDA     BKGND         ; ALSO SET BACKGROUND COLOR
FCAA| 91 6Ø             DSPBKGND STA    (BAS8L),Y
FCAC| 6Ø                       RTS
FCAD|                   ;
FCAD| 48                DSPL8Ø  PHA                   ; PRESERVE CHARACTER
FCAE| A5 5C                     LDA     CH            ; DETERMINE WHICH PAGE
FCBØ| 4A                       LSR     A
FCB1| A8                       TAY
FCB2| 68                       PLA
FCB3| BØF5                     BCS     DSPBKGND      ; BRANCH IF $9ØØ PAGE
FCB5| 91 5E                     STA     (BAS4L),Y
FCB7| 6Ø                       RTS
FCB8|                   ;
FCB8| B1 7E              NOTCR   LDA     (INBUF),Y     ; ECHO CHARACTER
FCBA| 2Ø 39FC                   JSR     COUT
FCBD| C9 88                     CMP     #88           ; BACKSPACE
FCBF| FØ1D                     BEQ     BKSPCE
FCC1| C9 98                     CMP     #98           ; CANCEL?
FCC3| FØØ8                     BEQ     CANCEL
FCC5| E6 8Ø                     INC     TEMP
FCC7| A5 8Ø                     LDA     TEMP
FCC9| C9 5Ø                     CMP     #INBUFLEN
FCCB| DØ17                     BNE     NXTCHAR       ; NO WRAP AROUND ALLOWED.
FCCD| A9 DC              CANCEL  LDA     #ØDC          ; OUTPUT BACKSLASH
FCCF| 2Ø 39FC                   JSR     COUT
FCD2| 2Ø EFFC                   JSR     CROUT
FCD5| FCD5              GETLNZ  .EQU    *
FCD5| A5 6B              GETLN   LDA     PROMPT
FCD7| 2Ø 39FC                   JSR     COUT
FCDA| AØ Ø1                     LDY     #Ø1
FCDC| 84 8Ø                     STY     TEMP          ; START AT BEGINNING OF INBUF
FCDE| A4 8Ø              BKSPCE  LDY     TEMP
FCEØ| FØF3                     BEQ     GETLN
FCE2| C6 8Ø                     DEC     TEMP          ; BACK UP INPUT BUFFER
FCE4| 2Ø 6ØFD           NXTCHAR JSR     RDCHAR        ; GET INPUT
FCE7| A4 8Ø                     LDY     TEMP
FCE9| 91 7E                     STA     (INBUF),Y
FCEB| C9 8D                     CMP     #8D
FCED| DØC9                     BNE     NOTCR
FCEF| FCEF              CROUT   .EQU    *
FCEF| 2C ØØCØ                   BIT     KBD           ; TEST FOR START/STOP
FCF2| 1Ø13                     BPL     NOSTOP
FCF4| 2Ø 2EFD                   JSR     KEYIN3        ; READ KBD
FCF7| C9 AØ                     CMP     #ØAØ          ; IS IT A SPACE?
FCF9| FØØ7                     BEQ     STOPLST       ; YES, PAUSE TIL NEXT KEYPRESS.
FCFB| C9 89                     CMP     #89           ; QUIT THIS OPERATION
FCFD| DØØ8                     BNE     NOSTOP        ; NO, IGNORE THIS KEY.
FCFF| 4C 9FFA                   JMP     ERROR2        ; YES, RESTART
FDØ2| AD ØØCØ           STOPLST LDA     KBD
FDØ5| 1ØFB                     BPL     STOPLST
FDØ7| A9 8D              NOSTOP  LDA     #8D
FDØ9| 4C 39FC                   JMP     COUT
FDØC|                   ;
FDØC| 6C 7ØØØ            RDKEY   JMP     @KSWL
FDØF|                   ;
```

```
FD0F| A9 7F              KEYIN    LDA   #7F          ; MAKE SURE FIRST IS CURSOR
FD11| 85 63                       STA   TBAS4H
FD13| 20 88FD                     JSR   PICK         ; GO READ SCREEN
FD16| 48                 KEYIN1   PHA                ; SAVE CHR AT CURSOR POSITION
FD17| 20 35FD                     JSR   KEYWAIT      ; TEST FOR KEYPRESS
FD1A| B008                       BCS   KEYIN2       ; GO GET IT
FD1C| A5 69                       LDA   CURSOR       ; GIVE THEM AN UNDERSCORE FOR A TIME
FD1E| 20 9DFC                     JSR   DISPLAY
FD21| 20 35FD                     JSR   KEYWAIT      ; GO SEE IF KEYPRESSED
FD24| 68                 KEYIN2   PLA
FD25| 08                         PHP                ; SAVE KEYPRESS STATUS
FD26| 48                         PHA
FD27| 20 9DFC                     JSR   DISPLAY
FD2A| 68                         PLA
FD2B| 28                         PLP
FD2C| 90E8                       BCC   KEYIN1
FD2E| AD 00C0             KEYIN3   LDA   KBD          ; READ KEYBOARD
FD31| 2C 10C0             KEYIN4   BIT   KBDSTRB     ; CLEAR KEYBOARD STROBE
FD34| 60                         RTS
FD35| E6 58              KEYWAIT  INC   TBAS4L       ; JUST KEEP COUNTING
FD37| D009                       BNE   KWAIT2
FD39| E6 63                       INC   TBAS4H
FD3B| A9 7F                       LDA   #7F          ; TEST FOR DONE
FD3D| 18                         CLC
FD3E| 25 63                       AND   TBAS4H
FD40| F005                       BEQ   KEYRET       ; RETURN IF TIMED OUT
FD42| 0E 00C0             KWAIT2   ASL   KBD
FD45| 90EE                       BCC   KEYWAIT
FD47| 60                 KEYRET   RTS
FD48|                    ;
FD48|                    ;
FD48| FD48               ESC3     .EQU  *
FD48| 20 77FD                     JSR   GOESC
FD4B| A5 68              ESCAPE   LDA   MODES        ; SET TO + SIGN FOR CURSOR MOVES
FD4D| 29 80                       AND   #80
FD4F| 49 AB                       EOR   #0AB
FD51| 85 69                       STA   CURSOR
FD53| 20 0CFD             ESC1     JSR   RDKEY        ; READ NEXT CHARACTER
FD56| A0 08                       LDY   #08          ; TEST FOR ESCAPE COMMAND
FD58| D9 F0FF             ESC2     CMP   ESCTABL,Y
FD5B| F0EB                       BEQ   ESC3
FD5D| 88                         DEY
FD5E| 10F8                       BPL   ESC2         ; LOOP TIL FOUND OR DONE
FD60|                    ;
FD60| A9 80              RDCHAR   LDA   #80          ; GO READ A CHARACTER
FD62| 25 68                       AND   MODES
FD64| 85 69                       STA   CURSOR       ; SAVE STANDARD CURSOR
FD66| 20 0CFD                     JSR   RDKEY
FD69| C9 9B                       CMP   #9B          ; ESCAPE CHARACTER?
FD6B| F0DE                       BEQ   ESCAPE
FD6D| C9 95                       CMP   #95          ; FORWARD COPY?
FD6F| D0D6                       BNE   KEYRET
FD71| 20 88FD                     JSR   PICK         ; GET CHARACTER FROM SCREEN
FD74| 09 80                       ORA   #80          ; SET TO NORMAL ASCII
FD76| 60                         RTS
FD77|                    ;
FD77| A9 FB              GOESC    LDA   #0FB
FD79| 48                         PHA
FD7A| B9 7FFD                     LDA   ESCVECT,Y
FD7D| 48                         PHA
FD7E| 60                         RTS
FD7F| A1                 ESCVECT  .BYTE  0A1         ; CLEOL-1
FD80| 84                         .BYTE  84          ; CLEOP-1
FD81| 7C                         .BYTE  7C          ; CLSCRN-1
FD82| 62                         .BYTE  62          ; COL40-1
FD83| 5C                         .BYTE  5C          ; COL80-1
FD84| EC                         .BYTE  0EC         ; CURLEFT-1
FD85| CA                         .BYTE  0CA         ; CURIGHT-1
FD86| DC                         .BYTE  0DC         ; CURDOWN-1
FD87| B7                         .BYTE  0B7         ; CURUP-1
FD88|                    ;
FD88| A5 5C              PICK     LDA   CH           ; GET A CHARACTER AT CURRENT CURSOR POSITION
FD8A| 4A                         LSR   A            ; DETERMINE WHICH PAGE.
FD8B| A8                         TAY
FD8C| 24 68                      BIT   MODES        ; AND IF 80 COLUMN MODE
FD8E| 5005                       BVC   PICK40       ; FORGET CARRY IF 40 COLUMNS
FD90| 9003                       BCC   PICK40       ; GET CHARACTER FROM $400
FD92| B1 60                      LDA   (BAS8L),Y
FD94| 60                         RTS
FD95| B1 5E              PICK40   LDA   (BAS4L),Y
FD97| 60                         RTS
FD98|                    ;
FD98| FD98               CLDSTRT  .EQU  *
FD98| A9 03                      LDA   #03
FD9A| 8D 0FF                     STA   0FFD0        ; ZERO PAGE IS ON 3!
FD9D| FD9D               SETUP    .EQU  *
FD9D| D8                         CLD                ; OF COURSE!
FD9E| A2 03                      LDX   #03
FDA0| 86 7F                      STX   INBUF+1
FDA2| BD BCFF            SETUP1   LDA   NMIRQ,X
```

```
FDA5| 9D CAFF                    STA     ØFFCA,X
FDA8| BD B4FF                    LDA     HOOKS,X
FDAB| 95 6E                      STA     CSWL,X
FDAD| BD B8FF                    LDA     VBOUNDS,X
FDBØ| 95 58                      STA     LMARGIN,X
FDB2| CA                         DEX
FDB3| 1ØED                      BPL     SETUP1
FDB5| 85 82                      STA     IBDRVN
FDB7| A9 AØ                      LDA     #ØAØ          ; INPUT BUFFER AT $3AØ
FDB9| 85 7E                      STA     INBUF
FDBB| A9 6Ø                      LDA     #6Ø
FDBD| 85 81                      STA     IBSLOT
FDBF| A9 FF                      LDA     #ØFF
FDC1| 85 68                      STA     MODES
FDC3| 2Ø 63FB                    JSR     COL4Ø         ; SET 4Ø COLUMNS, CLEAR SCREEN
FDC6|                    ;
FDC6| ØØAØ              ADR     .EQU    ØAØ
FDC6| ØØAØ              CPORTL  .EQU    ADR
FDC6| ØØA1              CPORTH  .EQU    ADR+1
FDC6| ØØA2              CTEMP   .EQU    ADR+2
FDC6| ØØA3              CTEMP1  .EQU    ADR+3
FDC6| ØØA4              YTEMP   .EQU    ADR+4
FDC6| ØØCØ              ROWTEMP .EQU    ADR+2Ø
FDC6| CØDB              CWRTON  .EQU    ØCØDB
FDC6| CØDA              CWRTOFF .EQU    ØCØDA
FDC6| FFEC              CB2CTRL .EQU    ØFFEC
FDC6| FFED              CB2INT  .EQU    ØFFED
FDC6|                    ;
FDC6|                    ;
FDC6| A9 78              GENENTR LDA     #78           ; INIT SCREEN INDX LOCATIONS
FDC8| 85 AØ                      STA     CPORTL
FDCA| A9 Ø8                      LDA     #Ø8
FDCC| 85 A1                      STA     CPORTH
FDCE| A9 FØ                      LDA     #ØFØ          ; SET UP INDEX TO CHRSET
FDDØ| 85 A4                      STA     YTEMP
FDD2| A9 ØØ                      LDA     #ØØ
FDD4| AA                         TAX
FDD5| 95 CØ              ZIPTEMPS STA    ROWTEMP,X
FDD7| E8                         INX
FDD8| EØ 2Ø                      CPX     #2Ø
FDDA| DØF9                      BNE     ZIPTEMPS
FDDC| A9 Ø5                      LDA     #Ø5           ; FAKE THE FIRST BIT PATTERN
FDDE| 18                         CLC                   ; (PHANTOM 9TH BIT SHIFTED AS BIT Ø)
FDDF| Ø8                         PHP
FDEØ| 48                         PHA
FDE1| 86 A2              GENASC  STX     CTEMP         ; GENERATE THE ASCII
FDE3| AØ Ø7              GASCI1  LDY     #Ø7           ; CODES FOR THE FIRST PASS
FDE5| A6 A2              GASCI2  LDX     CTEMP
FDE7| 8A                 GASCI3  TXA
FDE8| 91 AØ                      STA     (CPORTL),Y    ; $XXF=CHR Ø / 4
FDEA| E8                         INX                   ; $XXE=CHR 1 / 5
FDEB| 88                         DEY                   ; $XXD=CHR 2 / 6
FDEC| 3ØØ6                      BMI     GASCI4        ; $XXC=CHR 3 / 7
FDEE| CØ Ø3                      CPY     #Ø3           ; $XXB=CHR Ø / 4
FDFØ| DØF5                      BNE     GASCI3        ; $XXA=CHR 1 / 5
FDF2| FØF1                      BEQ     GASCI2        ; $XX9=CHR 2 / 6
FDF4| 2Ø 99FE            GASCI4  JSR     NXTPORT       ; $XX8=CHR 3 / 7
FDF7| BØØ8                      BCS     CBYTES        ; GO DECODE CHARACTER TABLE
FDF9| C9 ØA                      CMP     #ØA           ; SECOND SET OF 4?
FDFB| DØE6                      BNE     GASCI1
FDFD| A2 24                      LDX     #24
FDFF| DØEØ                      BNE     GENASC        ; BRANCH ALWAYS
FEØ1| 68                 CBYTES  PLA                   ; RESTORE BIT PATTERN
FEØ2| 28                         PLP
FEØ3| A2 17                      LDX     #17           ; (4 CHARACTERS OF 6 ROWS)
FEØ5| AØ Ø5              CCOLMS  LDY     #Ø5           ; (FIVE COLUMNS)
FEØ7| 36 C4              CSHFT   ROL     ROWTEMP+4,X   ; BREAK BYTE INTO
FEØ9| ØA                         ASL     A             ; 5 BIT GROUPS
FEØA| DØØE                      BNE     SHFTCNT       ; BRANCH IF MORE BITS IN THIS BYTE
FEØC| 84 A2                      STY     CTEMP
FEØE| C6 A4                      DEC     YTEMP         ; (NOTE. CARRY IS SET)
FE1Ø| FØ16                      BEQ     DONE          ; BRANCH IF ALL DONE
FE12| A4 A4                      LDY     YTEMP         ; GET CHARACTER TABLE INDEX
FE14| B9 C4FE                    LDA     CHRSET-1,Y
FE17| 2A                         ROL     A             ; (CARRY KEEPS BYTE NON-ZERO UNTIL ALL 8 ARE
FE18|                                                  ;  ARE SHIFTED)
FE18| A4 A2                      LDY     CTEMP         ; RESTORE COLUMN COUNT
FE1A| 88                 SHFTCNT DEY                   ; GOT ALL FIVE BITS?
FE1B| DØEA                      BNE     CSHFT         ; NO, DO NEXT
FE1D| CA                         DEX                   ; ALL ROWS DONE
FE1E| 1ØE5                      BPL     CCOLMS        ; NO, DO NEXT
FE2Ø| Ø8                         PHP                   ; SAVE REMAINING BIT PATTERN AND CARRY
FE21| 48                         PHA
FE22| 2Ø 28FE                    JSR     STORCHRS      ; MOVE EM TO NON DISPLAYED VIDEO AREA
FE25| 4C Ø1FE                    JMP     CBYTES
FE28|                    ;
FE28| FE28              DONE    .EQU    *
FE28|                    ;
FE28| A2 1F              STORCHRS LDX    #1F           ; MOVE CHARACTER PATTERNS TO VIDEO AREA
FE2A| AØ ØØ              STORSET LDY     #ØØ
```

```
FE2C| B5 CØ                  STOROW    LDA     ROWTEMP,X
FE2E| ØA                              ASL     A             ; SHIFT TO CENTER
FE2F| 29 3E                           AND     #3E           ; STRIP EXTRA GARBAGE
FE31| 91 AØ                           STA     (CPORTL),Y
FE33| CA                              DEX
FE34| C8                              INY
FE35| CØ Ø8                           CPY     #Ø8           ; THIS GROUP DONE
FE37| DØF3                           BNE     STOROW        ; NO, NEXT ROW
FE39| 2Ø 99FE                        JSR     NXTPORT
FE3C| C9 Ø8                           CMP     #Ø8
FE3E| FØØ4                          BEQ     GENDONE       ; ALL ROWS STORED?
FE4Ø| 8A                              TXA
FE41| 1ØE7                          BPL     STORSET
FE43| 6Ø                            RTS                   ; PARTIAL SET ($478-$5FF)
FE44|                         ;
FE44| A9 Ø1                  GENDONE   LDA     #Ø1           ; SET NORMAL MODE
FE46| 85 A2                           STA     CTEMP
FE48| A9 6Ø                  GEN1      LDA     #6Ø           ; PREPARE TO SEND BYTES TO CHARACTER
FE4A| 2C DBCØ                        BIT     CWRTON        ; GENERATOR RAM
FE4D| 2Ø AEFE                        JSR     VRETRCE       ; WAIT FOR NEXT VERTICAL RETRACE
FE5Ø| A9 2Ø                           LDA     #2Ø           ; WAIT AGAIN
FE52| 2Ø AEFE                        JSR     VRETRCE
FE55| 2C DACØ                        BIT     CWRTOFF       ; CHARACTERS ARE NOW LOADED
FE58| 2Ø 88FE                        JSR     ALTCHR        ; REPEAT THIS SET FOR OTHER 64 CHARACTERS
FE5B| C6 A2                           DEC     CTEMP         ; HAVE WE DONE ALTERNATES YET?
FE5D| 1Ø16                          BPL     GEN2          ; NO, DO IT!
FE5F| A9 Ø8                           LDA     #Ø8           ; BUMP ASCII VALUES FOR NEXT SET
FE61| 85 A1                           STA     CPORTH
FE63| AØ Ø7                  NXTASCI   LDY     #Ø7           ; THE USUAL COUNTDOWN
FE65| B1 AØ                  NXTASC2   LDA     (CPORTL),Y
FE67| 18                              CLC
FE68| 69 Ø8                           ADC     #Ø8
FE6A| 91 AØ                           STA     (CPORTL),Y
FE6C| 88                              DEY
FE6D| 1ØF6                          BPL     NXTASC2
FE6F| 2Ø 99FE                        JSR     NXTPORT
FE72| 9ØEF                          BCC     NXTASCI
FE74| 6Ø                            RTS
FE75| AØ Ø3                  GEN2      LDY     #Ø3           ; SETUP ALTERNATE WITH UNDERLINES
FE77| A9 7F                           LDA     #7F
FE79| 99 FCØ5                 UNDER     STA     Ø5FC,Y
FE7C| 99 FCØ7                         STA     Ø7FC,Y
FE7F| 88                              DEY
FE8Ø| 1ØF7                          BPL     UNDER
FE82| A9 Ø8                           LDA     #Ø8
FE84| 85 A1                           STA     CPORTH
FE86| DØCØ                          BNE     GEN1
FE88|                         ;
FE88| AØ Ø7                  ALTCHR    LDY     #Ø7           ; ADJUST ASCII FOR ALTERNATE SET
FE8A| B1 AØ                  ALTC1     LDA     (CPORTL),Y
FE8C| 49 2Ø                           EOR     #2Ø           ; $2Ø-->  $4Ø-->$6Ø
FE8E| 91 AØ                           STA     (CPORTL),Y
FE9Ø| 88                              DEY
FE91| 1ØF7                          BPL     ALTC1         ; ADJUST THEM ALL
FE93| 2Ø 99FE                        JSR     NXTPORT
FE96| 9ØFØ                          BCC     ALTCHR
FE98| 6Ø                            RTS
FE99|                         ;
FE99| A5 AØ                  NXTPORT   LDA     CPORTL        ; CONVERT $78->$F8 OR $F8-$78
FE9B| 49 8Ø                           EOR     #8Ø
FE9D| 85 AØ                           STA     CPORTL
FE9F| 3ØØ2                          BMI     NOHIGH
FEA1| E6 A1                           INC     CPORTH
FEA3| A5 A1                  NOHIGH    LDA     CPORTH
FEA5| C9 ØC                           CMP     #ØC
FEA7| DØØ4                          BNE     PORTDN
FEA9| A9 Ø4                           LDA     #Ø4
FEAB| 85 A1                           STA     CPORTH
FEAD| 6Ø                    PORTDN    RTS
FEAE|                         ;
FEAE| 85 A3                  VRETRCE   STA     CTEMP1        ; SAVE BITS TO BE STORED
FEBØ| AD ECFF                        LDA     CB2CTRL       ; CONTROL PORT FOR 'CB2'
FEB3| 29 3F                           AND     #3F           ; RESET HI BITS TO Ø
FEB5| Ø5 A3                           ORA     CTEMP1
FEB7| 8D ECFF                        STA     CB2CTRL
FEBA| A9 Ø8                           LDA     #Ø8           ; TEST VERTICAL RETRACE
FEBC| 8D EDFF                        STA     CB2INT
FEBF| 2C EDFF                  VWAIT     BIT     CB2INT        ; WAIT FOR RETRACE
FEC2| FØFB                          BEQ     VWAIT
FEC4| 6Ø                            RTS
FEC5|                         ;
FEC5| FEC5                    CHRSET    .EQU    *
FEC5|                         ;
FEC5| FØ Ø1 82 18 4Ø 84 81             .BYTE   ØFØ,Ø1,82,18,4Ø,84,81,2F,58,44,81,29,Ø2,1E,Ø1,91,7C,1F,49,3Ø
FECC| 2F 58 44 81 29 Ø2 1E
FED3| Ø1 91 7C 1F 49 3Ø
FED9| 8A Ø8 43 14 31 2A 22             .BYTE   8A,Ø8,43,14,31,2A,22,13,ØE3,ØF7,ØC4,91,48,ØA2,ØDA,24,ØC6,4A
FEEØ| 13 E3 F7 C4 91 48 A2
FEE7| DA 24 C6 4A
FEEB| 62 8C 24 C6 F8 63 8C             .BYTE   62,8C,24,ØC6,ØF8,63,8C,ØC1,46,17,52,8A,ØAF,16,14,ØE3,33,31
```

```
FEF2| C1 46 17 52 8A AF 16
FEF9| 14 E3 33 31
FEFD| C6 F8 DC 73 3F 46 17            .BYTE   ØC6,ØF8,ØDC,73,3F,46,17,62,8C,21,ØE6,18,6A,8D,61,ØCF,18,62
FFØ4| 62 8C 21 E6 18 6A 8D
FFØB| 61 CF 18 62
FFØF| 74 D1 B9 18 49 4C 91            .BYTE   74,ØD1,ØB9,18,49,4C,91,ØCØ,ØF3,Ø9,2C,91,ØCØ,14,1D,8C,ØEF,Ø7
FF16| CØ F3 Ø9 2C 91 CØ 14
FF1D| 1D 8C EF Ø7
FF21| 17 43 88 31 84 1E DF            .BYTE   17,43,88,31,84,1E,ØDF,ØB,31,84,ØF8,ØFE,77,3E,3E,17,62,8C,ØFD
FF28| ØB 31 84 F8 FE 77 3E
FF2F| 3E 17 62 8C FD
FF34| C7 5Ø E3 ØB 51 C5 E8            .BYTE   ØC7,5Ø,ØE3,ØB,51,ØC5,ØE8,ØC8,73,18,ØC,42,3E,Ø1,Ø2,2Ø,42,3E
FF3B| C8 73 18 ØC 42 3E Ø1
FF42| Ø2 2Ø 42 3E
FF46| 41 18 8C Ø8 ØØ 7Ø EE            .BYTE   41,18,8C,Ø8,ØØ,7Ø,ØEE,ØØ,11,11,21,11,Ø2,ØEØ,3C,21,31,Ø2,ØEØ
FF4D| ØØ 11 11 21 11 Ø2 EØ
FF54| 3C 21 31 Ø2 EØ
FF59| 1C ØØ C8 B9 8Ø 62 14            .BYTE   1C,ØØ,ØC8,ØB9,8Ø,62,14,1F,46,ØA2,ØDE,43,2C,Ø4,88,ØBE,ØFF,ØCE
FF6Ø| 1F 46 A2 DE 43 2C Ø4
FF67| 88 BE FF CE
FF6B| 7D 37 49 88 95 18 98            .BYTE   7D,37,49,88,95,18,98,Ø9,62,ØD1,44,ØE8,88,ØFB,Ø2,9Ø,4Ø,ØØ,1Ø
FF72| Ø9 62 D1 44 E8 88 FB
FF79| Ø2 9Ø 4Ø ØØ 1Ø
FF7E| EØ Ø3 Ø2 ØØ 4Ø ØØ ØØ            .BYTE   ØEØ,Ø3,Ø2,ØØ,4Ø,ØØ,ØØ,Ø8,ØØ,ØØ,28,1Ø,42,44,25,82,ØB8,2F,48
FF85| Ø8 ØØ ØØ 28 1Ø 42 44
FF8C| 25 82 B8 2F 48
FF91| 25 44 1Ø 82 Ø2 ØØ 2F            .BYTE   25,44,1Ø,82,Ø2,ØØ,2F,5A,4Ø,45,Ø2,8E,64,5Ø,9Ø,Ø1,3E,26,42,8Ø
FF98| 5A 4Ø 45 Ø2 8E 64 5Ø
FF9F| 9Ø Ø1 3E 26 42 8Ø
FFA5| 21 8Ø ØØ Ø5 ØØ F8 8Ø            .BYTE   21,8Ø,ØØ,Ø5,ØØ,ØF8,8Ø,ØØ,Ø5,Ø8,ØF8,8Ø,28,Ø5,88
FFAC| ØØ Ø5 Ø8 F8 8Ø 28 Ø5
FFB3| 88
FFB4|                       ;
FFB4| FFB4                  HOOKS       .EQU    *
FFB4| Ø6FC                              .WORD   COUT2
FFB6| ØFFD                              .WORD   KEYIN
FFB8| FFB8                  VBOUNDS     .EQU    *
FFB8| ØØ 5Ø ØØ 18                       .BYTE   ØØ,5Ø,ØØ,18
FFBC|                       ;
FFBC| 4C 86F6               NMIRQ       JMP     RECON       ; IN DIAGNOSTICS
FFBF| 4Ø                              RTI
FFCØ|                       ;
FFCØ| 43 4F 5Ø 59 52 49 47            .ASCII  "COPYRIGHT JANUARY, 198Ø  APPLE COMPUTER INC..JRH"
FFC7| 48 54 2Ø 4A 41 4E 55
FFCE| 41 52 59 2C 2Ø 31 39
FFD5| 38 3Ø 2Ø 2Ø 41 5Ø 5Ø
FFDC| 4C 45 2Ø 43 4F 4D 5Ø
FFE3| 55 54 45 52 2Ø 49 4E
FFEA| 43 2E 2E 4A 52 48
FFFØ|                       ;
FFFØ| CC DØ D3 B4 B8 88 95  ESCTABL     .BYTE   ØCC,ØDØ,ØD3,ØB4,ØB8,88,95,8A,8B,ØØ
FFF7| 8A 8B ØØ
FFFA|                       ;
FFFA| CAFF                  NMI         .WORD   ØFFCA
FFFC| EEF4                  RESET       .WORD   DIAGN       ; NOTHING
FFFE| CDFF                  IRQ         .WORD   ØFFCD
ØØØØ|
ØØØØ|                               .END
```

J. R. Huston (also worked on SOS)

```
--------------------------------------------------------------------------------------------------------------
SYMBOL TABLE DUMP
--------------------------------------------------------------------------------------------------------------

AB - Absolute     LB - Label     UD - Undefined     MC - Macro
RF - Ref          DF - Def       PR - Proc          FC - Func
PB - Public       PV - Private   CS - Consts

A1H      AB ØØ75 |  A1L      AB ØØ74 |  A1PC     LB F9D6 |  A1PC1    LB F9D9 |  A2H      AB ØØ77 |
A2L      AB ØØ76 |  A3H      AB ØØ79 |  A3L      AB ØØ78 |  A4H      AB ØØ7B |  A4L      AB ØØ7A |
ADR      AB ØØAØ |  ALTC1    LB FE8A |  ALTCHR   LB FE88 |  ASC1     LB FB4Ø |  ASC2     LB FB4C |
ASC3     LB FB5A |  ASCDONE  LB FAØ8 |  ASCII    LB FA1B |  ASCIIØ   LB FA1D |  ASCII1   LB F9E1 |
ASCII2   LB F9E3 |  ASCII3   LB F9F4 |  BAS4H    AB ØØ5F |  BAS4L    AB ØØ5E |  BAS8H    AB ØØ61 |
BAS8L    AB ØØ6Ø |  BASCALC  LB FBC7 |  BASCALC1 LB FC19 |  BELL     LB FC4E |  BITOFF   LB FA29 |
BITON    LB FA25 |  BKGND    AB ØØ67 |  BKSPCE   LB FCDE |  BL1      LB FAB4 |  BLOCKIO  AB F479 |
BSCLC2   LB FC2D |  CANCEL   LB FCCD |  CARRAGE  LB FBAF |  CB2CTRL  AB FFEC |  CB2INT   AB FFED |
CBYTES   LB FEØ1 |  CCOLMS   LB FEØ5 |  CH       AB ØØ5C |  CHRSET   LB FEC5 |  CKMDE    LB FA1E |
CLDSTRT  LB FD98 |  CLEOL    LB FBA2 |  CLEOL1   LB FC89 |  CLEOL2   LB FC91 |  CLEOP    LB FB85 |
CLEOP1   LB FB8E |  CLSCRN   LB FB7D |  CMDSRCH  LB F91C |  CMDTAB   LB F96C |  CMDVEC   LB F97D |
COL4Ø    LB FB63 |  COL8Ø    LB FB5D |  CONTROL  LB FBA7 |  COUT     LB FC39 |  COUT1    LB FC47 |
COUT2    LB FCØ6 |  CPORTH   AB ØØA1 |  CPORTL   AB ØØAØ |  CRCHK    LB F9FD |  CRMON    LB FA3A |
CROUT    LB FCEF |  CSHFT    LB FEØ7 |  CSWH     AB ØØ6F |  CSWL     AB ØØ6E |  CTEMP    AB ØØA2 |
CTEMP1   AB ØØA3 |  CTRLRET  LB FC38 |  CURDN1   LB FBC7 |  CURDOWN  LB FBDD |  CURIGHT  LB FBCB |
CURLEFT  LB FBED |  CURSOR   AB ØØ69 |  CURUP    LB FBB8 |  CURUP1   LB FBC2 |  CV       AB ØØ5D |
CWRTOFF  AB CØDA |  CWRTON   AB CØDB |  DEST     LB FAA5 |  DIAGN    AB F4EE |  DIGIT    LB F941 |
DIGRET   LB F96B |  DISPLAY  LB FC9D |  DISPLAYX LB FC1Ø |  DONE     LB FE28 |  DSPBKGND LB FCAA |
DSPL8Ø   LB FCAD |  DUMMY    LB FACB |  DUMP     LB FBØD |  DUMPØ    LB FB1Ø |  DUMP1    LB FB1D |
DUMP2    LB FB2Ø |  DUMP3    LB FB3Ø |  DUMP8    LB FAFD |  DUMPASC  LB FB35 |  ENTRY    LB F9Ø1 |
ERROR    LB FAA2 |  ERROR1   LB FBØB |  ERROR2   LB FA9F |  ESC1     LB FD53 |  ESC2     LB FD58 |
ESC3     LB FD48 |  ESCAPE   LB FD4B |  ESCTABL  LB FFFØ |  ESCVECT  LB FD7F |  FORGND   AB ØØ66 |
GASCI1   LB FDE3 |  GASCI2   LB FDE5 |  GASCI3   LB FDE7 |  GASCI4   LB FDF4 |  GEN1     LB FE48 |
```

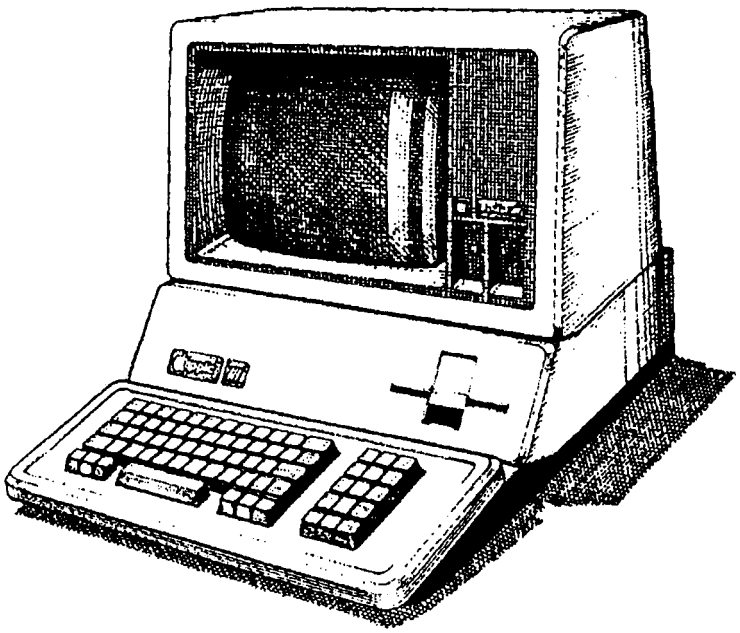
J=James
R=Richard
aka
Dick
Huston

```
GEN2     LB FE75 |  GENASC   LB FDE1 |  GENDONE  LB FE44 |  GENENTR  LB FDC6 |  GETLN    LB FCD5 |
GETLNZ   LB FCD5 |  GETNUM   LB F92C |  GO       LB FA91 |  GOESC    LB FD77 |  HOOKS    LB FFB4 |
IBBUFP   AB 0085 |  IBCMD    AB 0087 |  IBDRVN   AB 0082 |  IBSLOT   AB 0081 |  INBUF    AB 007E |
INBUFLEN AB 0050 |  INCHORZ  LB FC13 |  IRQ      LB FFFE |  JUMP     LB FA8F |  KBD      AB C000 |
KBDSTRB  AB C010 |  KEYIN    LB FD0F |  KEYIN1   LB FD16 |  KEYIN2   LB FD24 |  KEYIN3   LB FD2E |
KEYIN4   LB FD31 |  KEYRET   LB FD47 |  KEYWAIT  LB FD35 |  KSWH     AB 0071 |  KSWL     AB 0070 |
KWAIT2   LB FD42 |  LASTLN   LB FC87 |  LEFT80   LB FBF3 |  LEFTUP   LB FBFD |  LFA36    LB FA36 |
LMARGIN  AB 0058 |  LNFD     LB FC52 |  MASK     AB 0069 |  MISMATCH LB FA66 |  MODES    AB 0068 |
MON      LB F904 |  MONITOR  PR ---- |  MONZ     LB F908 |  MOVE     LB FA40 |  MOVNXT   LB FA45 |
NMI      LB FFFA |  NMIRQ    LB FFBC |  NOHIGH   LB FEA3 |  NOSTOP   LB FD07 |  NOTCR    LB FCB8 |
NOVER    LB FAF3 |  NXTA1    LB F994 |  NXTA4    LB F98E |  NXTASC2  LB FE65 |  NXTASCI  LB FE63 |
NXTBAS   LB F94F |  NXTBIT   LB F947 |  NXTBS2   LB F959 |  NXTCHAR  LB FCE4 |  NXTCHR   LB F932 |
NXTINP   LB F915 |  NXTLIN   LB FC16 |  NXTPORT  LB FE99 |  OLDPC    LB F9E0 |  PCH      AB 0073 |
PCL      AB 0072 |  PICK     LB FD88 |  PICK40   LB FD95 |  PORTDN   LB FEAD |  PRA1BYTE LB FA82 |
PRBYCOL  LB F9C4 |  PRBYTE   LB F9AE |  PRBYTSP  LB FA84 |  PRCOLON  LB F9C7 |  PRHEX    LB F9B7 |
PRHEX2   LB F9C1 |  PRHEXZ   LB F9B9 |  PRINTA1  LB FA75 |  PROMPT   AB 006B |  PRSPC    LB FA87 |
RDCHAR   LB FD60 |  RDKEY    LB FD0C |  READ     LB FAD4 |  RECON    AB F686 |  REPEAT   LB FA2D |
REPEAT1  LB FA35 |  RESET    LB FFFC |  RET1     LB F7FE |  RET2     LB F900 |  RET3     LB F882 |
RETA1    LB F9AD |  RIGHT1   LB FBD1 |  RMARGIN  AB 0059 |  ROWTEMP  AB 00C0 |  RWERROR  LB FA97 |
RWLOOP   LB FADB |  SAVCMD   LB FAD9 |  SCAN     LB F912 |  SCRL1    LB FC61 |  SCRL2    LB FC63 |
SCRL3    LB FC7A |  SCRNLOC  AB 0058 |  SCROLL   LB FC5B |  SEARCH   LB FA09 |  SEP      LB FAAE |
SET80    LB FB67 |  SET80A   LB FB6F |  SET80B   LB FB7B |  SETCHZ   LB FBD7 |  SETCV    LB FBC5 |
SETCVH   LB FBDB |  SETMDZ   LB FAD1 |  SETMODE  LB FACC |  SETUP    LB FD9D |  SETUP1   LB FDA2 |
SHFTCNT  LB FE1A |  SPCE     LB FAB8 |  SRCH1    LB FA15 |  STACK    AB 006A |  STATE    AB 007C |
STOPLST  LB FD02 |  STOR     LB FABF |  STOR1    LB FAC3 |  STORCHRS LB FE28 |  STOROW   LB FE2C |
STORSET  LB FE2A |  SVMASK   LB F9D3 |  TBAS4H   AB 0063 |  TBAS4L   AB 0058 |  TBAS8H   AB 0065 |
TBAS8L   AB 0064 |  TEMP     AB 0080 |  TEMPX    AB 006C |  TEMPY    AB 006D |  TOSUB    LB F95E |
TST80WID LB F9CB |  TSTA1    LB F99D |  TSTBACK  LB FBE9 |  TSTBELL  LB FC4A |  TSTCR    LB FBAB |
TSTDUMP  LB FB0A |  UNDER    LB FE79 |  USER     LB FA8C |  USERADR  AB 0358 |  VBOUNDS  LB FFB8 |
VRETRCE  LB FEAE |  VRFY     LB FA4F |  VRFY1    LB FA54 |  VRFY2    LB FA60 |  VWAIT    LB FEBF |
WINBTM   AB 005B |  WINTOP   AB 005A |  WRTE     LB FAD7 |  YSAV     AB 007D |  YTEMP    AB 00A4 |
ZIPTEMPS LB FDD5 |  ZSTATE   LB F967 |

Assembly complete:      1129 lines
0   Errors flagged on this Assembly

-------------------------------------------------------------------------------------------------------
6502 OPCODE STATIC FREQUENCIES
-------------------------------------------------------------------------------------------------------

    ADC :    5  |  ***
    AND :   14  |  ********
    ASL :   12  |  *******
    BCC :   21  |  *************
    BCS :   20  |  ************
    BEQ :   82  |  **************************************************
    BIT :   12  |  *******
    BMI :    7  |  ****
    BNE :   41  |  *************************
    BPL :   18  |  ***********
    BVC :    2  |  *
    BVS :    3  |  *
    CLC :    7  |  ****
    CLD :    2  |  *
    CMP :   35  |  **********************
    CPX :    1  m
    CPY :    2  |  *
    DEC :    7  |  ****
    DEX :    7  |  ****
    DEY :    9  |  *****
    EOR :    6  |  ***
    INC :   18  |  ***********
    INX :    3  |  *
    INY :    3  |  *
    JMP :   18  |  ***********
    JSR :   79  |  **************************************************
    LDA :  117  M  ********************************************************************************
    LDX :   12  |  *******
    LDY :   20  |  ************
    LSR :   11  |  *******
    ORA :   10  |  ******
    PHA :   16  |  **********
    PHP :    4  |  **
    PLA :   14  |  ********
    PLP :    3  |  *
    ROL :    4  |  **
    RTI :    1  m
    RTS :   34  |  *********************
    SBC :   67  |  ****************************************
    SEC :    5  |  ***
    SEI :    1  m
    STA :   72  |  **********************************************
    STX :    7  |  ****
    STY :    5  |  ***
    TAX :    2  |  *
    TAY :    5  |  ***
    TSX :    1  m
    TXA :    2  |  *
    TXS :    1  m
    TYA :    3  |  *
```

```
    Minimum frequency =    1
    Maximum frequency =  117

    Average frequency =   17

    Unused opcodes:

    BRK  CLI  CLV  NOP  ROR  SED

    Program opcode usage:  89 %

-------------------------------------------------------------------------------
(1.ØØ) That's all, Folks ...
-------------------------------------------------------------------------------
```

=FINIS=

Apple III Computer Information

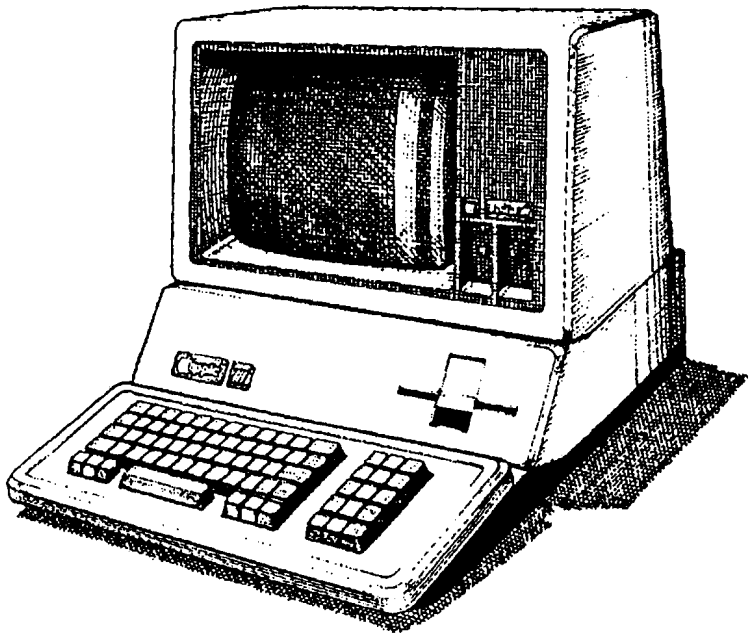


Inside the Apple III ROM

Document Table of Contents

Revision 2 • 04 Dec 1997
Revision 1 • 30 Nov 1997

Apple III Computer Information



Inside the Apple III ROM

Revision 2 • 04 Dec 1997

Inside the Apple /// Computer ROM

David T. Craig • 04 December 1997
71533.606@compuserve.com

TABLE OF CONTENTS

1 INTRODUCTION

This document provides a general overview of the contents of the Apple /// computer ROM revision 1. This information should be used in conjunction with a copy of the ROM source code listing. The audience of this document is anyone with an interest in the technology of the Apple /// computer's hardware and software.

NOTE

There were two revisions of the Apple /// ROM, revision 0 and revision 1. Revision 0 ROMs had at address F1B9 the value 60. Revision 1 ROMs had at address F1B9 the value A0.

This ROM contains 4 KB of 6502 programming and several data tables. The ROM occupies memory addresses F000-FFFF. The basic purpose of the ROM is to test the Apple /// computer hardware and boot an operating system from the ///'s built-in floppy disk drive. The ROM also contains a simple Monitor program whose purpose is to allow the user to interact with the /// at the hexadecimal level.

Apple planned from an architectural perspective to support two 4K ROMs. But only one ROM was ever created. The Environment Register let you control which ROM was active. Both ROMs shared the same address space so you could only have one ROM active at a time. This feature would have doubled the ROM's effective size providing Apple with more room for ROM-based features that higher-level /// software (e.g. SOS) could have used.

When the Apple /// computer is turned on the ROM's flow of execution is as follows:

1) The ROM starts execution at the address contained in FFFC-FFFD (RESET) which is address F4EE (DIAGN).

2) Diagnostics (DIAGN/F4EE) starts. The diagnostic first initializes some memory for the ROM's use. If the Open Apple and the Control keys are held down then enter the ROM Monitor. Otherwise run several diagnostic checks of the /// hardware (tests zero page, sizes memory, initializes screen text buffer, tests stack memory, tests ROM checksum, tests VIA chip, tests ACIA chip, tests A/D circuitry, tests keyboard connection). Any diagnostic failures display an error message and the user has to reset the computer.

3) Read block 0 (512 bytes) to address A000 from the floppy disk in the built-in disk drive (BOOT/F6A1). If no disk is found or block 0 cannot be read then display "RETRY" and wait for the user to reset the computer. If the block is successfully read then execute the block contents (this is called the SOS Bootstrap Loader: see section ROM USAGE BY SOS).

2 ROM SECTIONS

Section	Address	Purpose
Disk I/O	F000-F4C4	Read and write floppy disk blocks (512 bytes each)
Diagnostics	F4C5-F7FE	Diagnose the /// hardware
Monitor	F7FF-FFFF	Interacts with user so user can do simple things

3 IMPORTANT ROM ROUTINES

BLOCKIO / F479 — Reads or write a disk block (512 bytes), calls routine REGRWTS (F000) which reads a sector (256 bytes) from the disk.

BOOT / F6A1 — Read floppy disk block #0 into address A000, execute the block.

ENTRY / F901 — Monitor entry point.

DIAGN / F4EE — Diagnostic entry point.

USRENTRY/F6E6 — Tests RAM and displays a table showing chip failures (users may execute this routine from the Monitor). This test is aimed at Apple ///s with 128K of RAM that exists on the older 12-Volt RAM boards. Though this routine will work with the newer 5-Volt RAM boards (256K) this test shows wrong information when RAM errors occur since the two RAM boards contain a different number of RAM chips. You can identify the different RAM boards as follows: The 5V boards have a large gray ceramic resistor near the edge and the 12V boards have a small blue tubular capacitor. To test the ///'s RAM you really should use Apple's /// Diagnostics Disk which lets you specify which RAM board you have.

4 ROM TABLES

Here's a list of the important data tables in the ROM. This list does not include disk I/O tables.

Table Name / Address	Contents
CHRSET / FEC5-FFB3	Default character set (overridden when SOS loads the character set from SOS.DRIVER)
Copyright / FFC0-FFEF	Copyright message (contains the initials "JRH" for J. R. "Dick" Huston who was a key player behind the /// and SOS)
NMI / FFFA-FFFB	Jump address for the Non-Maskable Interrupt signal
RESET / FFFC-FFFD	Jump address when the /// is powered on
IRQ / FFFE-FFFF	Jump address for the Interrupt Request signal

5 ROM USAGE BY SOS

The Apple /// operating system (SOS = Sophisticated Operating System or Sara's OS) uses several ROM routines. These routines seem to all be related to disk block I/O. The following discussion is based on SOS version 1.3.

When the ROM loads block 0 from a SOS disk the ROM is loading the SOS Bootstrap Loader program. This program, which is at most 512 bytes in length, uses the ROM routine REGRWTS (F000) to read the SOS Loader into memory. This program does not test the ROM revision. It is interesting to note that ROM routine BLOCKIO is not used, instead a lower-level routine (REGRWTS) is used.

The SOS Loader determines if the ROM is revision 1 by comparing address F1B9's contents against A0 (reference: SOS source file SOSLDR.D.SRC). If this comparison fails then SOS displays on the screen the error "ROM ERROR: PLEASE NOTIFY YOUR DEALER." If the ROM revision is correct then the SOS loader uses the ROM's disk I/O routines to read more of SOS into memory.

The disk /// driver that is built into SOS also uses the ROM to perform disk block I/O (reference: DISK3.SRC). It is interesting to note that when the disk driver is initialized the driver checks if the ROM revision is 0 or 1. A revision of 0 is detected if address F1B9 contains 60. If neither revision is found then the disk driver returns an error to SOS (I don't think this will ever happen since the SOS loader has already determined that the ROM is revision 1). For a valid ROM revision the disk driver sets up several jump vectors which point to the appropriate addresses in the ROM for the various ROM routines needed by the disk driver. Therefore, the disk driver seems compatible with either ROM revision whereas the SOS loader likes only revision 1.

The .CONSOLE driver source listing appears to not use any ROM routines even though the ROM contains 40 and 80 column text routines and keyboard input routines. I assume the console driver was much more sophisticated than the ROM's text features and so using the ROM routines would not have worked well for this driver. I also assume that if the console driver used the ROM that when ROM

revision 1 was built the console driver would have had to be changed and Apple (smartly) did not want to do this.

6 MONITOR COMMANDS

Holding down the Open Apple and Control keys when the /// starts or when you press the Reset key activates the /// ROM Monitor. The screen will display in the upper left corner a small right-facing arrow with a blinking underscore character as the cursor. The Monitor's commands are based on the Apple][‘s Monitor commands but some commands have changed slightly and others are new for the (newer) ///.

The Monitor supports the following commands:

addr1.addr2	Dump memory data to screen from address 1 to address 2 and display ASCII character at the right of the screen.
CARRIAGE RETURN	Dump next line of addresses to the screen.
SPACE	Pause current memory dump. Press again to continue.
addr:byte_list	Store starting at the address the list of bytes.
addr:'text'	Store text starting at address with high bit clear.
addr:"text"	Store text starting at address with high bit set.
addr3<addr1.addr2M	Move data in addresses 1-2 to address 3.
addr3<addr1.addr2V	Verify data in addresses 1-2 is the same as data starting at address 3.
byte<addr1.addr2S	Search memory in address range 1-2 for the byte.
block<addr1.addr2W	Write address range to disk starting at the disk block.
block<addr1.addr2R	Read disk starting at block to the address range.
addrG	Call subroutine at the address.
addrJ	Jump to the address.
U	Call user routine starting at address $03F8.
X	Repeat last command line until you press the SPACE BAR.
ESC-8	Display 80 columns of text.
ESC-4	Display 40 columns of text.
/	Seperate multiple commands on the same line.
CTRL-I	Interrupt current operation, return to Monitor command line.

Note: See Wells' *Apple /// Entry Points* article for a great overview of the ROM Monitor, its commands (with some syntax errors), and the memory locations that need setting up for the key ROM routines to work. Apple's */// Service Reference Manual* (p. 13.57) has a list of Monitor commands. Anderson's *The Apple Nobody Knows* also has good Monitor command info.

To obtain a binary dump of the /// ROM you can do the following:

1. Initialize a disk on either the /// or an Apple][computer.
2. Insert the new disk in the ///.
3. Start the /// and hold down the Open Apple and Control keys.
4. You should be in the /// Monitor.
5. Type 0<F000.FFFW to write the ROM to disk blocks 0 to 7
6. Use a disk block reader on the /// or the][to read the ROM blocks and save them to a real file.

This disk writing is needed since the ROM does not provide a command for redirecting screen output to the ///'s serial port. But, I've read that you can output the ROM contents to the ///'s serial port but this involves using the Monitor to write a small program. If anyone has such a program please send a copy my way.

7 A FEW COMMENTS

I find it interesting, at least from a software engineering perspective, to note that in my opinion the /// ROM is missing several key features which I thought any system ROM would need. The ROM is missing two features which I think would have been useful to Apple and outside /// programmers:

1) The ROM does not have an explicit version number which exists at a specific ROM address. This version number could be used to validate the ROM in case there were several different ROMs (as there were). Apple uses a pseudo ROM version number (called the revision number) during the loading of SOS but this is somewhat lame in my opinion.

2) The ROM does not have a dispatch routine for use by the OS or applications that want to use ROM routines. This dispatch routine would reside at a specific address (e.g., F000) and it would take as input a command number and a set of parameters. These parameters could be passed via registers or on the stack. This routine would allow Apple to change the ROM and ROM "users" would not need to change their programming as long as they used the selector routine. The Apple][ROM did not have such a routine which caused Apple many headaches when it wanted to change the Apple][ROM and had to keep lots of routines in their same place.

3) The ROM source code is rather sparse concerning comments. It would be nice if the ROM contained detailed information about what each routine did and how to call the routines. Obviously, Apple did not expect anyone but Apple's own programmers to ever see the ROM source or use the ROM routines. (I've seen the Lisa computer's ROM listing which is much better documented than the ///'s and both are comparable in terms of age).

8 REFERENCES

Apple /// ROM Listing - Revision 0

This can be found in the Apple /// patent (#4,383,296) dated May 1983. Note that in places this ROM listing is not always readable.

Apple /// ROM Listing - Revision 1

I have a very readable listing of the revision 1 ROM that was printed on a laser printer.

Apple /// Service Reference Manual (Level 2)

This almost 500 page document by Apple has everything you would want to know about the ///'s hardware, low-level software, and how to service a broken ///. Includes descriptions of the System Monitor (a.k.a. Development Monitor) [page 17.3] and the built-in RAM test routine [page 13.51].

Apple /// SOS Bootstrap Loader Listing

Shows how 512 bytes of code are used to load SOS from disk into the ///'s memory.

The following articles provide good ROM information:

Apple /// Entry Points, Andy Wells, Call-APPLE, October 1981

Apple /// Dabbling, Rick Smith, Apple Orchard, Summer 1981

/// Bits: John Jeppson's Guided Tour of Highway ///, John Jeppson, Softalk, May 1983

The Apple Nobody Knows, Alan Anderson, Apple Orchard, Fall 1981

Unlocking the Apple /// - Part 3, Alan Anderson, Apple Orchard, September 1982

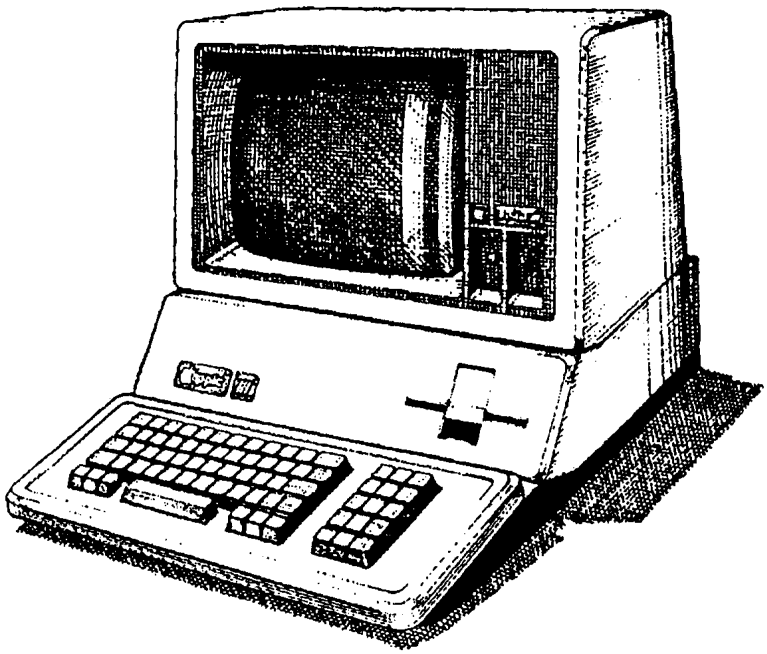
Apple ///: 12-Volt 128K Internal Diagnostics, Apple Technical Information Library

9 DOCUMENT MODIFICATION HISTORY

30 Nov 1997	Created this document.
04 Dec 1997	Corrected a few problems, extended the Reference section to include more /// articles pertaining to the /// ROM, added this section, added section MONITOR COMMANDS.

###

Apple III Computer Information



Inside the Apple III ROM

Revision 1 • 30 Nov 1997

Inside the Apple /// Computer ROM

David T. Craig • 30 November 1997
71533.606@compuserve.com

TABLE OF CONTENTS

1 INTRODUCTION

This document provides a general overview of the contents of the Apple /// computer ROM revision 1. This information should be used in conjunction with a copy of the ROM source code listing. The audience of this document is anyone with an interest in the technology of the Apple /// computer's hardware and software.

NOTE

There were two revisions of the Apple /// ROM, revision 0 and revision 1. Revision 0 ROMs had at address F1B9 the value 60. Revision 1 ROMs had at address F1B9 the value A0.

This ROM contains 4 KB of 6502 programming and several data tables. The ROM occupies memory addresses F000-FFFF. The basic purpose of the ROM is to test the Apple /// computer hardware and boot an operating system from the ///'s built-in floppy disk drive. The ROM also contains a simple Monitor program whose purpose is to allow the user to interact with the /// at the hexadecimal level.

When the Apple /// computer is turned on the ROM's flow of execution is as follows:

1) The ROM starts execution at the address contained in FFFC-FFFD (RESET) which is address F4EE (DIAGN).

2) Diagnostics (DIAGN/F4EE) starts. The diagnostic first initializes some memory for the ROM's use. If the Open Apple key is held down then enter the ROM Monitor. Otherwise run several diagnostic checks of the /// hardware (tests zero page, sizes memory, initializes screen text buffer,

tests stack memory, tests ROM checksum, tests VIA chip, tests ACIA chip, tests A/D circuitry, tests keyboard connection). Any diagnostic failures display an error message and the user has to reset the computer.

3) Read block 0 (512 bytes) to address A000 from the floppy disk in the built-in disk drive (BOOT/F6A1). If no disk is found or block 0 cannot be read then display "RETRY" and wait for the user to reset the computer. If the block is successfully read then execute the block contents (this is called the SOS Bootstrap Loader: see section ROM USAGE BY SOS).

2 ROM SECTIONS

Section	Address	Purpose
Disk I/O	F000-F4C4	Read and write floppy disk blocks (512 bytes each)
Diagnostics	F4C5-F7FE	Diagnose the /// hardware
Monitor	F7FF-FFFF	Interacts with user so user can do simple things

3 IMPORTANT ROM ROUTINES

BLOCKIO / F479	Reads or write a disk block (512 bytes), calls routine REGRWTS (F000) which reads a sector (256 bytes) from the disk
BOOT / F6A1	Read floppy disk block #0 into address A000, execute the block
ENTRY / F901	Monitor entry point
DIAGN / F4EE	Diagnostic entry point
USRENTRY/F6E6	Tests RAM and displays a table showing chip failures (users may execute this routine from the Monitor)

4 ROM TABLES

Here's a list of the important data tables in the ROM. This list does not include disk I/O tables.

Table Name / Address	Contents
CHRSET / FEC5-FFB3	Default character set (overridden when SOS loads the character set from SOS.DRIVER)
Copyright / FFC0-FFEF	Copyright message (contains the initials "JRH" for J. R. Huston who was a key player behind the /// and SOS)
NMI / FFFA-FFFB	Jump address for the Non-Maskable Interrupt signal
RESET / FFFC-FFFD	Jump address when the /// is powered on

IRQ / FFFE-FFFF Jump address for the Interrupt Request signal

5 ROM USAGE BY SOS

The Apple /// operating system (SOS) uses several ROM routines. These routines seem to all be related to disk block I/O. The following discussion is based on SOS version 1.3.

When the ROM loads block 0 from a SOS disk the ROM is loading the SOS Bootstrap Loader program. This program, which is at most 512 bytes in length, uses the ROM routine REGRWTS (F000) to read the SOS Loader into memory. This program does not test the ROM revision. It is interesting to note that ROM routine BLOCKIO is not used, instead a lower-level routine (REGRWTS) is used.

The SOS Loader determines if the ROM is revision 1 by comparing address F1B9's contents against A0 (reference: SOS source file SOSLDR.D.SRC). If this comparison fails then SOS displays on the screen the error "ROM ERROR: PLEASE NOTIFY YOUR DEALER." If the ROM revision is correct then the SOS loader uses the ROM's disk I/O routines to read more of SOS into memory.

The disk /// driver that is built into SOS also uses the ROM to perform disk block I/O (reference: DISK3.SRC). It is interesting to note that when the disk driver is initialized the driver checks if the ROM revision is 0 or 1. A revision of 0 is detected if address F1B9 contains 60. If neither revision is found then the disk driver returns an error to SOS (I don't think this will ever happen since the SOS loader has already determined that the ROM is revision 1). For a valid ROM revision the disk driver sets up several jump vectors which point to the appropriate addresses in the ROM for the various ROM routines needed by the disk driver. Therefore, the disk driver seems compatible with either ROM revision whereas the SOS loader likes only revision 1.

6 A FEW COMMENTS

I find it interesting, at least from a software engineering perspective, that the ROM is missing some key features which I thought any system ROM would need. The ROM is missing two features which I think would have been useful to Apple and outside /// programmers:

1) The ROM does not have an explicit version number which exists at a specific ROM address. This version number could be used to validate the ROM in case there were several different ROMs (as there were). Apple uses a pseudo ROM version number (called the revision number) during the loading of SOS but this is somewhat lame in my opinion.

2) The ROM does not have a selector routine for use by the OS or applications that want to use ROM routines. This selector would reside at a specific address (e.g., F000) and it would take as input a command number and a set of parameters. These parameters could be passed via registers or on the stack. This routine would allow Apple to change the ROM and ROM

"users" would not need to change their programming as long as they used the selector routine. The Apple][ROM did not have such a routine which caused Apple many headaches when it wanted to change the Apple][ROM and had to keep lots of routines in their same place.

7 REFERENCES

Apple /// ROM Listing

I have a very nice listing of revision 1 ROM. A listing (that is somewhat readable) for the earlier revision 0 ROM may be found in the Apple /// patent.

Apple /// Service Reference Manual (Level 2)

This almost 500 page book by Apple has everything you would want to know about the ///'s hardware, low-level software, and how to service a broken ///. Includes descriptions of the System Monitor (a.k.a. Development Monitor) [page 17.3] and the built-in RAM test routine [page 13.51].

Apple /// SOS Bootstrap Loader Listing

Shows how 512 bytes of code is used to load SOS from disk into the ///'s memory.

###

Apple /// Computer Technical Information

SOME COMMENTS ABOUT THE APPLE /// COMPUTER BOOT ROM

David T Craig -- 27 February 2004

BACKGROUND

The Apple /// computer was introduced by Apple Computer in 1980 and was discontinued in 1985.

This computer was a microcomputer with orginally 128 KB of RAM memory expandable to 256 KB of RAM. It featured a 4 KB ROM (addressed from $F000 to $FFFF hexadecimal) which housed the initial programming that executed when the user turned on the computer. This ROM contained programming for the following functions:

+ diagnose hardware circuitry and memory
+ load and run a disk operating system (i.e. "boot")
+ provide an interface to a simple monitor program

The author wrote these comments after looking at the Apple /// ROM listing as found in Apple Computer's patent number 4,383,296 dated 10 May 1983. This analysis occured during a scanning of the Apple /// patent.

ROM COMMENTS

The Apple /// patent's ROM program listing is terrible in terms of printed quality. Many parts are very faint and impossible to read. I assume this was done on purpose by Apple's legal department so that Apple's competitors would not be able to duplicate this ROM programming easily.

The ROM programming does not seem to have been built for expansion. By this I mean the programming seems to have been written to just make it work and no long term thought was given to the ROM programming's organization.

There were two versions of the ROM. The Apple /// operating system (OS) programming needed to differentiate between the ROM versions since the ROM contained several routines which the OS used. This version determination was not done in a logical way. A memory location was chosen at random (at least it seems this way to me) to serve as the ROM's "version number". The OS had to test this "version number" when it needed to use specific ROM services.

The ROM version also determined the location of several ROM routines which the Apple /// OS used.

The ROM's organization could have been improved greatly in my opinion if it was organized differently. At the beginning of the ROM address space ($F000) include a short header containing the following:

$F000 - ROM version number
$F001 - ROM size (K bytes)
$F002 - ROM checksum (2 bytes)
$F003 - ROM routine dispatch jump vector (3 bytes)
$F006 - ROM copyright notice (e.g. "(c) Apple Computer 1980")

The remainder of the ROM would have contained whatever programming and table data was needed.

The routine dispatch jump vector would be a standard jump instruction to a routine in the ROM whose purpose would be to let outside programs such as the operating system, device drivers, or even application programs access ROM routines in a ROM version independent manner. The dispatch routine would take as input a command number (in say the CPU's A register) and return result information in the CPU's X and Y registers. The A register on return would contain an error result with 0 meaning no error. Or, some fixed memory area could be use to handle ROM routine parameters. This dispatch mechanism could be seen as a BIOS (basic input output system).

Possible dispatch routines could be:

+ Restart or Cold start or Warm start the computer
+ Read a block from a disk drive
+ Write a block to a disk drive
+ Return size in blocks of a disk drive
+ Checksum the ROM for diagnostic purposes
+ Test computer's RAM memory for diagnostic purposes
+ Enter the Apple /// Monitor program

This dispatch mechanism would have simplified the Apple /// OS use of the ROM services since the ROM would always be accessed from just one address ($F003). If the OS requested a ROM service which was unavailable (e.g. an old ROM was installed) then the ROM would tell the OS that the service did not exist via a dispatch error result.

CONCLUSION

Hopefully this little commentary provides some useful information to its reader. If you are interested in the Apple /// computer you should see its patents (one is for the Apple ///, the other is for the Apple /// Plus). The first patent contains the full ROM listing, but the author has a real digital copy which is much more readable.

Enjoy.

###