```
        NAM    M6800 DISK-BUG DS VER 3.5
        OPT    PAG

* Floppy Disk Controller Debug Monitor
* Written 27 Aug 1980
* Michael Holley
*
* Record of modifications
* 18 OCT 1981  Disk routines DC-1
* 23 JAN 1982  Command Table
*  8 MAY 1982  DDC-16 Disk Controller
* 27 JUN 2001  Double Sided Disks
*  3 JUN 2003  Help Text
*
* This version requires a Disk Controller with
* a DRQ/IRQ status register as found in the
* DC-3, DC-4 and DC5 cards
*
**************************************
* Commands
* J xxxx Jump to location xxxx
*
* M xxxx Examine and alter memory location
* N      Examine next memory location
* V      Examine same memory location
* B      Examine previous memory location
*
* Z xx   Select and restore drive xx to track 0
*
* T ttss Read sector ss on track tt into BUFFER
*        and display it
*
* L ttss Load binary file starting at track tt
*        and sector ss. Display transfer address.
*
* W ttss Write BUFFER into sector ss on track tt
*
* $      Return to SWTBUG or SBUG-E
*
```

```
                ************************************
                * VECTORS
                * For programming into ROM
                *
                * ORG $FFF8
                *IRQA   FDB START
                *SWI    FDB CTRL
                *NMI    FDB START
                *RESET  FDB START

                ************************************
                * System Locations
C000            ROM     EQU    $C000     $E400 after SWTBUG
0000            RAM     EQU    $0000
8004            ACIA    EQU    $8004     6800=$8004, 6809=$E004
8018            FDC     EQU    $8018     6800=$8018, 6809=$E018
E0D0            SWTBUG  EQU    $E0D0     6800=$E0D0, 6809=$F800

8014            DRVREG  EQU    FDC-4     FDC DRIVE SELECT REG
8014            DRQREG  EQU    FDC-4     FDC DRQ IRQ
8018            COMREG  EQU    FDC       FDC COMMAND REG
8019            TRKREG  EQU    FDC+1     FDC TRACK REG
801A            SECREG  EQU    FDC+2     FDC SECTOR REG
801B            DATREG  EQU    FDC+3     FDC DATA REG

                * RAM VARIABLES
0070            STACK   EQU    RAM+$0070
0071                    ORG    RAM+$0071
0071            SAVSTK  RMB    2         TEMP FOR STACK POINTER
0073            TEMP1   RMB    1         TEMP FOR INHEX AND OUTHEX
0074            TEMP2   RMB    1         GENERAL TEMP
0075            XTEMP1  RMB    2         INDEX REG TEMP
0077            XTEMP2  RMB    2         INDEX REG TEMP
0079            XHI     RMB    1         INDEX HIGH BYTE
007A            XLOW    RMB    1         INDEX LOW BYTE
007B            CURDRV  RMB    1         LAST USED DRIVE

                * Buffer location is shown in sign-on message
0100            BUFFER  EQU    RAM+$0100
```

```
C000                    ORG    ROM+$0000 BEGIN MONITOR
C000 8E 00 70  START    LDS    #STACK

                ************************************
                * FUNCTION: INITA - Initialize ACIA
                * INPUT: none
                * OUTPUT: none
                * CALLS: none
                * DESTROYS: acc A

0013            RESETA  EQU    %00010011
0011            CTLREG  EQU    %00010001

C003 86 13      INITA   LDA A  #RESETA   RESET ACIA
C005 B7 80 04           STA A  ACIA
C008 86 11              LDA A  #CTLREG   SET 8 BITS AND 2 STOP
C00A B7 80 04           STA A  ACIA

C00D 7E C0 F1           JMP    SIGNON    GO TO START OF MONITOR

                ************************************
                * FUNCTION: INCH - Input character
                * INPUT: none
                * OUTPUT: char in acc A
                * DESTROYS: acc A
                * CALLS: none
                * DESCRIPTION: Gets 1 character from terminal

C010 B6 80 04  INCH     LDA A  ACIA      GET STATUS
C013 47                 ASR A            SHIFT RDRF FLAG INTO CARRY
C014 24 FA              BCC    INCH      RECIEVE NOT READY
C016 B6 80 05           LDA A  ACIA+1    GET CHAR
C019 84 7F              AND A  #$7F      MASK PARITY
C01B 7E C0 79           JMP    OUTCH     ECHO & RTS

                ************************************
                * FUNCTION: INHEX - INPUT HEX DIGIT
                * INPUT: none
                * OUTPUT: Digit in acc A
                * CALLS: INCH
                * DESTROYS: acc A
                * Returns to monitor if not HEX input

C01E 8D F0      INHEX   BSR    INCH      GET A CHAR
C020 81 30              CMP A  #'0       ZERO
C022 2B 11              BMI    HEXERR    NOT HEX
C024 81 39              CMP A  #'9       NINE
C026 2F 0A              BLE    HEXRTS    GOOD HEX
C028 81 41              CMP A  #'A
C02A 2B 09              BMI    HEXERR    NOT HEX
C02C 81 46              CMP A  #'F
C02E 2E 05              BGT    HEXERR
C030 80 07              SUB A  #7        FIX A-F
```

```
 C032 84 0F     HEXRTS  AND A  #$0F      CONVERT ASCII TO DIGIT
 C034 39                RTS

 C035 7E C0 AF  HEXERR  JMP    CTRL      RETURN TO CONTROL LOOP

                ************************************
                * FUNCTION: BADDR - Build Address
                * INPUT: none
                * OUTPUT: Address in INDEX and XHI & XLOW
                * CALLS: BYTE
                * DESTROYS: INDEX, acc A
                * RAM: XHI, XLOW

 C038 8D 09     BADDR   BSR    BYTE      GET HIGH ORDER ADDRESS
 C03A 97 79             STA A  XHI       STORE IT
 C03C 8D 05             BSR    BYTE      GET LOW ORDER ADDRESS
 C03E 97 7A             STA A  XLOW      STORE IT
 C040 DE 79             LDX    XHI       LOAD ADDRESS INTO INDEX
 C042 39                RTS

                ************************************
                * FUNCTION: BYTE - Read BYTE  2 hex digits
                * INPUT: none
                * OUTPUT: BYTE in acc A
                * CALLS: INHEX
                * DESTROYS: acc A
                * RAM: TEMP1

 C043 8D D9     BYTE    BSR    INHEX     GET FIRST DIGIT
 C045 48                ASL A            SHIFT TO HIGH ORDER 4 BITS
 C046 48                ASL A
 C047 48                ASL A
 C048 48                ASL A
 C049 97 73             STA A  TEMP1     SAVE FIRST DIGIT
 C04B 8D D1             BSR    INHEX     GET SECOND DIGIT
 C04D 9B 73             ADD A  TEMP1     COMBINE 2 DIGITS INTO BYTE
 C04F 39                RTS

                ************************************
                * FUNCTION: MEMORY - Memory Examine and Change
                * INPUT: none
                * OUTPUT: none
                * CALLS: BADDR
                * DESTROYS: acc A,acc B, INDEX
                * RAM: XHI, XLOW

 C050 8D E6     MEMORY  BSR    BADDR     BUILD ADDRESS

                * BRA CHANGE
                ************************************
                * FUNCTION: CHANGE - Memory Examine and Change
                * INPUT: none
                * OUTPUT: none
                * CALLS: BADDR, OUTS, OUT2H, BYTE
```

```
                * DESTROYS: acc A,acc B, INDEX
                * RAM: XHI, XLOW

 C052 8D 49     CHANGE  BSR    OUTS      SPACE
 C054 A6 00             LDA A  0,X
 C056 8D 2E             BSR    OUT2H     PRINT BYTE
 C058 8D 43             BSR    OUTS      SPACE
 C05A 8D E7             BSR    BYTE      GET NEW BYTE
 C05C A7 00             STA A  0,X
 C05E 20 4F             BRA    CTRL      RETURN TO LOOP

                ************************************
                * FUNCTION: VIEW - Memory Examine and Change
                * INPUT:  acc B - N, B, or S
                * OUTPUT: none
                * CALLS: OUT2H, CHANGE
                * DESTROYS: acc A,acc B, INDEX
                * RAM: XHI, XLOW
                * S = VIEW SAME MEMORY, N = VIEW NEXT MEMORY
                * B = BACK ONE MEMORY

 C060 DE 79     VIEW    LDX    XHI
 C062 C1 56             CMP B  #'V       VIEW SAME LOCATION
 C064 27 07             BEQ    VIEW1
 C066 09                DEX
 C067 C1 42             CMP B  #'B       BACK ONE LOCATION
 C069 27 02             BEQ    VIEW1
 C06B 08                INX              UNDO BACK
 C06C 08                INX              INCREMENT MEMORY POINTER
 C06D DF 79     VIEW1   STX    XHI       PUT IT BACK
 C06F 96 79             LDA A  XHI
 C071 8D 13             BSR    OUT2H     PRINT ADDRESS
 C073 96 7A             LDA A  XLOW
 C075 8D 0F             BSR    OUT2H
 C077 20 D9             BRA    CHANGE    ENTER CHANGE FUNCTION

                ************************************
                * FUNCTION: OUTCH - Output a char to ACIA
                * INPUT: char in A
                * OUTPUT: none
                * CALLS: none
                * DESTROYS: none

 C079 37        OUTCH   PSH B            SAVE ACC B
 C07A F6 80 04  OUTCH1  LDA B  ACIA      GET STATUS
 C07D 57                ASR B
 C07E 57                ASR B            SHIFT TDR TO CARRY
 C07F 24 F9             BCC    OUTCH1
 C081 B7 80 05          STA A  ACIA+1    SEND CHAR
 C084 33                PUL B            RESTORE ACC B
 C085 39                RTS

                ************************************
                * FUNCTION: OUT2H - Output 2 hex digits
```

```
                * INPUT: BYTE in acc A
                * OUTPUT: none
                * CALLS: OUTCH
                * DESTROYS: none
                * RAM: TEMP1

 C086 97 73     OUT2H   STA A  TEMP1
 C088 44                LSR A           SHIFT HIGH DIGIT OVER
 C089 44                LSR A
 C08A 44                LSR A
 C08B 44                LSR A
 C08C 8D 00             BSR    OUTHR    OUTPUT FIRST DIGIT
 C08E 84 0F     OUTHR   AND A  #$0F     MASK LEFT DIGIT
 C090 8B 30             ADD A  #$30     MAKE ASCII
 C092 81 39             CMP A  #'9      IS IT 0-9
 C094 23 02             BLS    OUT21
 C096 8B 07             ADD A  #$7      MAKE IT A-F
 C098 8D DF     OUT21   BSR    OUTCH    PRINT IT
 C09A 96 73             LDA A  TEMP1    GET BYTE BACK
 C09C 39                RTS

                ****************************************
                * FUNCTION: OUTS - Output a space
                * INPUT: none
                * OUTPUT: none
                * CALLS: OUTCH
                * DESTROYS: acc A

 C09D 86 20     OUTS    LDA A  #$20     SPACE
 C09F 20 D8             BRA    OUTCH    (BSR & RTS)

                ****************************************
                * FUNCTION: PDATA - Print data string
                * INPUT: Start of string in INDEX
                * OUTPUT: none
                * CALLS: OUTCH
                * DESTROYS: acc A, INDEX
                * The character string must be terminated
                * with a End Of Text $04.

 C0A1 8D D6     PDATA2  BSR    OUTCH
 C0A3 08                INX             VIEW CHAR
 C0A4 A6 00     PDATA   LDA A  X
 C0A6 81 04             CMP A  #4       END OF TEXT
 C0A8 26 F7             BNE    PDATA2
 C0AA 39                RTS

                ****************************************
                * FUNCTION: JUMP
                * INPUT: none
                * OUTPUT: none
                * CALLS: BADDR
                * DESTROYS: INDEX
```

```
C0AB 8D 8B     JUMP    BSR    BADDR     GET ADDRESS
C0AD 6E 00             JMP    0,X       JUMP TO IT

               ************************************
               * MONITOR CONTROL LOOP

C0AF 8E 00 70  CTRL    LDS    #STACK
C0B2 8D 4B             BSR    PCRLF
C0B4 86 2B             LDA A  #'+
C0B6 8D C1             BSR    OUTCH
C0B8 BD C0 10          JSR    INCH      GET COMMAND
C0BB 16                TAB              SAVE COMMAND
C0BC 8D DF             BSR    OUTS      PRINT SPACE

C0BE CE C0 D2          LDX    #CMMD     COMMAND TABLE
C0C1 E1 00     CTRL1   CMP B  0,X       MATCH COMMAND
C0C3 27 09             BEQ    CTRL3     FOUND MATCH
C0C5 08                INX
C0C6 08                INX
C0C7 08                INX              SKIP OVER COMMAND
C0C8 6D 00             TST    0,X       TABLE ENDS WITH $00
C0CA 26 F5             BNE    CTRL1
C0CC 20 E1             BRA    CTRL

C0CE EE 01     CTRL3   LDX    1,X       LOAD XFER ADDRESS
C0D0 6E 00             JMP    0,X

C0D2 4A        CMMD    FCC    /J/
C0D3 C0 AB             FDB    JUMP
C0D5 4D                FCC    /M/
C0D6 C0 50             FDB    MEMORY
C0D8 4E                FCC    /N/
C0D9 C0 60             FDB    VIEW
C0DB 56                FCC    /V/
C0DC C0 60             FDB    VIEW
C0DE 42                FCC    /B/
C0DF C0 60             FDB    VIEW
C0E1 5A                FCC    /Z/
C0E2 C2 58             FDB    TRK00
C0E4 54                FCC    /T/
C0E5 C2 68             FDB    SECTR
C0E7 4C                FCC    /L/
C0E8 C2 E1             FDB    LOAD
C0EA 24                FCC    /$/
C0EB E0 D0             FDB    SWTBUG
C0ED 57                FCC    /W/
C0EE C2 89             FDB    WRTSEC
C0F0 00                FCB    $00       END OF TABLE

               ************************************
               * FUCTION: SIGNON
               * INPUT: none
               * OUTPUT: none
               * CALLS: PDATA
```

```
                * DESTROYS: acc A, INDEX

 C0F1 8D 0C     SIGNON  BSR    PCRLF
 C0F3 CE C1 09          LDX    #HELLO
 C0F6 8D AC             BSR    PDATA
 C0F8 CE C1 29          LDX    #HELP
 C0FB 8D A7             BSR    PDATA
 C0FD 20 B0             BRA    CTRL


                ************************************
                * FUCTION: PCRLF     Print CR and LF
                * INPUT: none
                * OUTPUT: none
                * CALLS: PDATA
                * DESTROYS: acc A
                * RAM: XTEMP1

 C0FF DF 75     PCRLF   STX    XTEMP1
 C101 CE C1 23          LDX    #CRLF
 C104 8D 9E             BSR    PDATA
 C106 DE 75             LDX    XTEMP1    RESTORE INDEX
 C108 39                RTS


                ************************************
                * STRINGS
 C109 20        HELLO   FCC    / MC6800 DISK-BUG - VER 3.5/
 C123 0D        CRLF    FCB    $0D,$0A,00,00,00,04

 C129 4A        HELP    FCC    /J xxxx Jump to xxxx/
 C13C 0D                FCB    $0D,$0A,00
                *
 C13F 4D        HELP01  FCC    /M xxxx Examine and alter memory/
 C15E 0D                FCB    $0D,$0A,00
 C161 4E        HELP02  FCC    /N      Next memory/
 C173 0D                FCB    $0D,$0A,00
 C176 56        HELP03  FCC    /V      Same memory/
 C188 0D                FCB    $0D,$0A,00
 C18B 42        HELP04  FCC    /B      Previous memory/
 C1A1 0D                FCB    $0D,$0A,00
                *
 C1A4 5A        HELP05  FCC    /Z xx   Select & restore drive xx/
 C1C4 0D                FCB    $0D,$0A,00
                *
 C1C7 54        HELP06  FCC    /T ttss Read sector ss on track tt into buffer/
 C1F4 0D                FCB    $0D,$0A,00
                *
 C1F7 4C        HELP07  FCC    /L ttss Load file. Show transfer address./
 C21F 0D                FCB    $0D,$0A,00
                *
 C222 57        HELP08  FCC    /W ttss Write buffer at $0100 /
 C23F 0D                FCB    $0D,$0A,00
                *
```

```
C242 24        HELP09  FCC    /$      Go to SWTBUG/
C255 0D                FCB    $0D,$0A,04

                       ************************************
                       * FUCTION: TRK00   RESTORE TO TRACK 00
                       * INPUT: none
                       * OUTPUT: none
                       * CALLS: BYTE
                       * DESTROYS:

C258 BD C0 43  TRK00   JSR    BYTE      GET DRIVE SELECT CODE
C25B B7 80 14          STA A  DRVREG
C25E 97 7B             STA A  CURDRV
C260 86 0B             LDA A  #$0B
C262 B7 80 18          STA A  COMREG
C265 7E C0 AF          JMP    CTRL

                       ************************************
                       * FUCTION: SECTR  SECTOR READ
                       * INPUT: none
                       * OUTPUT: none
                       * CALLS: BADDR, CTRL, PCRLF, OUTCH, OUTS, OUTHEX
                       * EXTERNAL: READ
                       * DESTROYS: acc A acc B INDEX
                       * RAM:
C268 BD C0 38  SECTR   JSR    BADDR     GET TRK & SEC
C26B CE 01 00          LDX    #BUFFER
C26E 96 79             LDA A  XHI       GET TRACK
C270 D6 7A             LDA B  XLOW      GET SECTOR
C272 BD C3 56          JSR    READ      READ SECTOR
C275 27 31             BEQ    DUMP      NO ERROR

C277 BD C0 FF  ERROR   JSR    PCRLF
C27A 86 45             LDA A  #'E
C27C BD C0 79          JSR    OUTCH
C27F BD C0 9D          JSR    OUTS      PRINT E & SPACE
C282 17                TBA              GET ERROR NUMBER
C283 BD C0 86          JSR    OUT2H
C286 7E C0 AF          JMP    CTRL

                       ************************************
                       * FUCTION: WRTSEC  SECTOR WRITE
                       * INPUT: none
                       * OUTPUT: none
                       * CALLS: BADDR, CTRL, PCRLF, OUTCH, OUTS, OUTHEX
                       * EXTERNAL: WRITE
                       * DESTROYS: acc A acc B INDEX
                       * RAM:
C289 BD C0 38  WRTSEC  JSR    BADDR     GET TRK & SEC
C28C CE 01 00          LDX    #BUFFER
C28F 96 79             LDA A  XHI       GET TRACK
C291 D6 7A             LDA B  XLOW      GET SECTOR
C293 BD C3 B6          JSR    WRITE     WRITE SECTOR
C296 26 DF             BNE    ERROR
```

```
C298 BD C0 FF          JSR    PCRLF
C29B 86 4F             LDA A  #'O
C29D BD C0 79          JSR    OUTCH
C2A0 86 4B             LDA A  #'K
C2A2 BD C0 79          JSR    OUTCH
C2A5 7E C0 AF  WRTSC9  JMP    CTRL

               ************************************

C2A8 CE 01 00  DUMP    LDX    #BUFFER
C2AB 86 10             LDA A  #16       NUMBER OF LINES
C2AD 36        DUMP1   PSH A
C2AE BD C0 FF          JSR    PCRLF
C2B1 C6 10             LDA B  #16       NUMBER OF BYTES
C2B3 DF 77             STX    XTEMP2
C2B5 A6 00     DUMP2   LDA A  0,X
C2B7 BD C0 86          JSR    OUT2H     PRINT 2 HEX
C2BA BD C0 9D          JSR    OUTS      PRINT SPACE
C2BD 08                INX
C2BE 5A                DEC B             DONE WITH LINE
C2BF 26 F4             BNE    DUMP2

C2C1 C6 10             LDA B  #16       NUMBER OF BYTES
C2C3 DE 77             LDX    XTEMP2
C2C5 A6 00     DUMP3   LDA A  0,X       GET CHARACTER
C2C7 84 7F             AND A  #$7F       MASK MSB
C2C9 81 7D             CMP A  #$7D
C2CB 2E 04             BGT    DUMP31
C2CD 81 1F             CMP A  #$1F       IS IT CONTROL
C2CF 2E 02             BGT    DUMP32
C2D1 86 5F     DUMP31  LDA A  #'_        DUMMY
C2D3 BD C0 79  DUMP32  JSR    OUTCH
C2D6 08                INX
C2D7 5A                DEC B             DONE WITH LINE
C2D8 26 EB             BNE    DUMP3

C2DA 32                PUL A
C2DB 4A                DEC A             DONE WITH DUMP
C2DC 26 CF             BNE    DUMP1
C2DE 7E C0 AF  DUMP9   JMP    CTRL

               ************************************
               * FUCTION: LOAD
               * INPUT: none
               * OUTPUT: none
               * CALLS: LOADER

C2E1 BD C0 38  LOAD    JSR    BADDR     GET TRACK AND SECTOR
C2E4 FF 01 00          STX    BUFFER
C2E7 BD C2 FA          JSR    LOADER    LOAD BINARY FILE
C2EA BD C0 9D          JSR    OUTS
C2ED 96 79             LDA A  XHI
C2EF BD C0 86          JSR    OUT2H     PRINT XFER ADDRESS
```

```
C2F2 96 7A            LDA A  XLOW
C2F4 BD C0 86         JSR    OUT2H
C2F7 7E C0 AF         JMP    CTRL

                      ************************************
                      * FUNCTION: LOADER - Binary file loader
                      * INPUT: Track and sector in BUFFER 0&1
                      * OUTPUT: Transfer Address in XHI & XLOW
                      * CALLS: READ

C2FA CE 02 00  LOADER  LDX    #BUFFER+256
C2FD DF 75            STX    XTEMP1
C2FF 9F 71            STS    SAVSTK    SAVE STACK FOR RETURN

                      * DO UNTIL END OF FILE

C301 8D 2F     LOAD1   BSR    GETCH     GET A CHAR FROM BUFFER
C303 81 02            CMP A  #$02      DATA RECORD HEADER
C305 27 0E            BEQ    LOAD2     BR IF SO
C307 81 16            CMP A  #$16      XFR ADDRESS HEADER
C309 26 F6            BNE    LOAD1     LOOP IF NEITHER
C30B 8D 25            BSR    GETCH     TRANSFER ADDRESS
C30D 97 79            STA A  XHI
C30F 8D 21            BSR    GETCH
C311 97 7A            STA A  XLOW
C313 20 EC            BRA    LOAD1

C315 8D 1B     LOAD2   BSR    GETCH     GET LOAD ADDRESS
C317 97 77            STA A  XTEMP2
C319 8D 17            BSR    GETCH
C31B 97 78            STA A  XTEMP2+1
C31D 8D 13            BSR    GETCH     GET BYTE COUNT
C31F 16               TAB              SAVE
C320 27 DF            BEQ    LOAD1     BR IF COUNT=0

C322 37        LOAD3   PSH B
C323 8D 0D            BSR    GETCH     GET NEXT BYTE
C325 33               PUL B
C326 DE 77            LDX    XTEMP2    GET LOAD ADDRESS
C328 A7 00            STA A  0,X
C32A 08               INX
C32B DF 77            STX    XTEMP2
C32D 5A               DEC B            END OF DATA RECORD?
C32E 26 F2            BNE    LOAD3
C330 20 CF            BRA    LOAD1     GET ANOTHER RECORD

                      * GET CHARACTER ROUTINE - READS SECTOR IF NECESSARY

C332 DE 75     GETCH   LDX    XTEMP1
C334 8C 02 00         CPX    #BUFFER+256 OUT OF DATA
C337 26 11            BNE    GETCH4    GO READ CHARACTER

C339 CE 01 00  GETCH2  LDX    #BUFFER
C33C A6 00            LDA A  0,X       GET NEXT TRACK
```

```
 C33E E6 01             LDA B  1,X       GET NEXT SECTOR
 C340 27 11             BEQ    GETCH6    0 SECTOR ENDS LOAD
 C342 BD C3 56  GETCH3  JSR    READ      NEXT SECTOR
 C345 26 09             BNE    GETCH5    READ ERROR
 C347 CE 01 04          LDX    #BUFFER+4 POINT PAST LINK
 C34A A6 00     GETCH4  LDA A  0,X       GET CHAR FROM BUFFER
 C34C 08                INX
 C34D DF 75             STX    XTEMP1
 C34F 39                RTS

 C350 7E C0 AF  GETCH5  JMP    CTRL      ERROR RETURN

 C353 9E 71     GETCH6  LDS    SAVSTK    CLEAR SUBROUTINES
 C355 39                RTS
```

```
                ****************************************
                * FLEX 2 I/O DRIVERS FOR 5/8 CONTROLLER
                * COMMENTS FROM FLEX 6809 GUIDE
                * 02 DEC 1981   MICHAEL HOLLEY

                * EQUATES
0002            DRQ     EQU    $02       DRQ BIT MASK
0001            BUSY    EQU    $01       BUSY MASK
001C            RDMSK   EQU    $1C       READ ERROR MASK
0018            VERMSK  EQU    $18       VERIFY ERROR MASK
005C            WTMSK   EQU    $5C       WRITE ERROR MASK
008C            RDCMND  EQU    $8C       READ COMMAND
00AC            WTCMND  EQU    $AC       WRITE COMMAND
0018            RSCMND  EQU    $18       RESTORE COMMAND
001B            SKCMND  EQU    $1B       SEEK COMMAND


                ****************************************
                * READ
                * Entry - (X) = FCB Sector Buffer Address
                *         (A) = Track Number
                *         (B) = Sector Number
                * The sector referenced by the track and sector
                * number is to be read into the Sector Buffer
                * area of the indicated FCB.

C356 8D 2D      READ    BSR    SEEK      SET TRACK AND SECTOR
C358 86 8C              LDA A  #RDCMND
C35A 0F                 SEI               NO INTERRUPTS ALLOWED
C35B B7 80 18           STA A  COMREG    READ SECTOR 10mS DELAY
C35E BD C3 AF           JSR    DELAY

C361 B6 80 14   READ2   LDA A  DRQREG    SET STATUS
C364 2B 07              BMI    READ3     BRANCH IF DATA PRESENT
C366 27 F9              BEQ    READ2     LOOP IF BUSY
C368 F6 80 18           LDA B  COMREG    GET ERROR
C36B 20 0B              BRA    READ6     REPORT ERROR

C36D B6 80 1B   READ3   LDA A  DATREG    READ SECTOR LOOP
C370 A7 00              STA A  0,X
C372 08                 INX
C373 7E C3 61           JMP    READ2

C376 8D 05              BSR    WAIT      WAIT UNTIL 1771 IS DONE
C378 C5 1C      READ6   BIT B  #RDMSK
C37A 01                 NOP
C37B 0E                 CLI               ENABLE INTURRUPTS
C37C 39                 RTS

C37D F6 80 18   WAIT    LDA B  COMREG    GET STATUS
C380 C5 01              BIT B  #BUSY
C382 26 F9              BNE    WAIT
C384 39                 RTS
```

```
                * SEEK THE SPECIFIED TRACK
C385 F7 80 1A   SEEK    STA B  SECREG    SET SECTOR
C388 C1 0A              CMP B  #10
C38A 22 04              BHI    SIDE1
C38C D6 7B              LDA B  CURDRV
C38E 20 04              BRA    SEEK2
C390 D6 7B      SIDE1   LDA B  CURDRV
C392 CA 40              ORA B  #$40      SIDE 1
C394 F7 80 14   SEEK2   STA B  DRVREG
C397 B1 80 19           CMP A  TRKREG
C39A 27 10              BEQ    SEEK4     TRACK IS CORRECT
C39C B7 80 1B           STA A  DATREG    CHANGE TRACK
C39F BD C3 AF           JSR    DELAY
C3A2 86 1B              LDA A  #SKCMND
C3A4 B7 80 18           STA A  COMREG    ISSUE SEEK COMMAND
C3A7 BD C3 AF           JSR    DELAY
C3AA 8D D1              BSR    WAIT
C3AC 7E C3 AF   SEEK4   JMP    DELAY     JUMP AND RETURN


C3AF BD C3 B2   DELAY   JSR    DELAY1
C3B2 BD C3 B5   DELAY1  JSR    DELAY2
C3B5 39         DELAY2  RTS

                ****************************************
                * WRITE
                * Entry - (X) = FCB Sector Buffer Address
                *         (A) = Track Number
                *         (B) = Sector Number
                * The sector referenced by the track and sector
                * number is to be written from the Sector Buffer
                * area of the indicated FCB.

C3B6 8D CD      WRITE   BSR    SEEK      SET TRACK AND SECTOR
C3B8 86 AC              LDA A  #WTCMND
C3BA 0F                 SEI              NO INTERRUPTS ALLOWED
C3BB B7 80 18           STA A  COMREG    ISSUE WRITE COMMAND
C3BE BD C3 AF           JSR    DELAY
C3C1 E6 00              LDA B  0,X

C3C3 B6 80 14   WRITE2  LDA A  DRQREG    SET STATUS
C3C6 2B 07              BMI    WRITE3    BRANCH IF READY
C3C8 27 F9              BEQ    WRITE2    LOOP IF BUSY
C3CA F6 80 18           LDA B  COMREG    GET STATUS
C3CD 20 09              BRA    WRITE6    DONE

C3CF F7 80 1B   WRITE3  STA B  DATREG    SEND TO DISK
C3D2 08                 INX
C3D3 E6 00              LDA B  0,X       GET NEXT BYTE
C3D5 7E C3 C3           JMP    WRITE2

C3D8 C5 5C      WRITE6  BIT B  #WTMSK    MASK ERRORS
C3DA 01                 NOP
```

```
 C3DB 0E                CLI             ENABLE INTERRUPTS
 C3DC 39                RTS

                        END    START
```

NO ERROR(S) DETECTED

SYMBOL TABLE:

ACIA	8004	BADDR	C038	BUFFER	0100	BUSY	0001	BYTE	C043
CHANGE	C052	CMMD	C0D2	COMREG	8018	CRLF	C123	CTLREG	0011
CTRL	C0AF	CTRL1	C0C1	CTRL3	C0CE	CURDRV	007B	DATREG	801B
DELAY	C3AF	DELAY1	C3B2	DELAY2	C3B5	DRQ	0002	DRQREG	8014
DRVREG	8014	DUMP	C2A8	DUMP1	C2AD	DUMP2	C2B5	DUMP3	C2C5
DUMP31	C2D1	DUMP32	C2D3	DUMP9	C2DE	ERROR	C277	FDC	8018
GETCH	C332	GETCH2	C339	GETCH3	C342	GETCH4	C34A	GETCH5	C350
GETCH6	C353	HELLO	C109	HELP	C129	HELP01	C13F	HELP02	C161
HELP03	C176	HELP04	C18B	HELP05	C1A4	HELP06	C1C7	HELP07	C1F7
HELP08	C222	HELP09	C242	HEXERR	C035	HEXRTS	C032	INCH	C010
INHEX	C01E	INITA	C003	JUMP	C0AB	LOAD	C2E1	LOAD1	C301
LOAD2	C315	LOAD3	C322	LOADER	C2FA	MEMORY	C050	OUT21	C098
OUT2H	C086	OUTCH	C079	OUTCH1	C07A	OUTHR	C08E	OUTS	C09D
PCRLF	C0FF	PDATA	C0A4	PDATA2	C0A1	RAM	0000	RDCMND	008C
RDMSK	001C	READ	C356	READ2	C361	READ3	C36D	READ6	C378
RESETA	0013	ROM	C000	RSCMND	0018	SAVSTK	0071	SECREG	801A
SECTR	C268	SEEK	C385	SEEK2	C394	SEEK4	C3AC	SIDE1	C390
SIGNON	C0F1	SKCMND	001B	STACK	0070	START	C000	SWTBUG	E0D0
TEMP1	0073	TEMP2	0074	TRK00	C258	TRKREG	8019	VERMSK	0018
VIEW	C060	VIEW1	C06D	WAIT	C37D	WRITE	C3B6	WRITE2	C3C3
WRITE3	C3CF	WRITE6	C3D8	WRTSC9	C2A5	WRTSEC	C289	WTCMND	00AC
WTMSK	005C	XHI	0079	XLOW	007A	XTEMP1	0075	XTEMP2	0077

+++