```
        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
                F000    ROM     EQU    $F000     $E400 after SWTBUG
                0000    RAM     EQU    $0000
                E004    ACIA    EQU    $E004     6800=$8004, 6809=$E004
                E018    FDC     EQU    $E018     6800=$8018, 6809=$E018
                F800    SWTBUG  EQU    $F800     6800=$E0D0, 6809=$F800

                E014    DRVREG  EQU    FDC-4     FDC DRIVE SELECT REG
                E014    DRQREG  EQU    FDC-4     FDC DRQ IRQ
                E018    COMREG  EQU    FDC       FDC COMMAND REG
                E019    TRKREG  EQU    FDC+1     FDC TRACK REG
                E01A    SECREG  EQU    FDC+2     FDC SECTOR REG
                E01B    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
```

```
  F000                      ORG    ROM+$0000 BEGIN MONITOR
  F000 10CE 0070     START  LDS    #STACK

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

                0013 RESETA EQU    %00010011
                0011 CTLREG EQU    %00010001

  F004 86   13       INITA  LDAA   #RESETA   RESET ACIA
  F006 B7   E004            STAA   ACIA
  F009 86   11              LDAA   #CTLREG   SET 8 BITS AND 2 STOP
  F00B B7   E004            STAA   ACIA

  F00E 7E   F0FE            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

  F011 B6   E004     INCH   LDAA   ACIA      GET STATUS
  F014 47                   ASRA             SHIFT RDRF FLAG INTO CARRY
  F015 24   FA              BCC    INCH      RECIEVE NOT READY
  F017 B6   E005            LDAA   ACIA+1    GET CHAR
  F01A 84   7F              ANDA   #$7F      MASK PARITY
 >F01C 7E   F07D            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

  F01F 8D   F0       INHEX  BSR    INCH      GET A CHAR
  F021 81   30              CMPA   #'0       ZERO
  F023 2B   11              BMI    HEXERR    NOT HEX
  F025 81   39              CMPA   #'9       NINE
  F027 2F   0A              BLE    HEXRTS    GOOD HEX
  F029 81   41              CMPA   #'A
  F02B 2B   09              BMI    HEXERR    NOT HEX
  F02D 81   46              CMPA   #'F
  F02F 2E   05              BGT    HEXERR
  F031 80   07              SUBA   #7        FIX A-F
```

```
 F033 84   0F       HEXRTS  ANDA   #$0F      CONVERT ASCII TO DIGIT
 F035 39                    RTS

>F036 7E   F0B6     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

 F039 8D   09       BADDR   BSR    BYTE      GET HIGH ORDER ADDRESS
 F03B 97   79               STAA   XHI       STORE IT
 F03D 8D   05               BSR    BYTE      GET LOW ORDER ADDRESS
 F03F 97   7A               STAA   XLOW      STORE IT
 F041 9E   79               LDX    XHI       LOAD ADDRESS INTO INDEX
 F043 39                    RTS

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

 F044 8D   D9       BYTE    BSR    INHEX     GET FIRST DIGIT
 F046 48                    ASLA             SHIFT TO HIGH ORDER 4 BITS
 F047 48                    ASLA
 F048 48                    ASLA
 F049 48                    ASLA
 F04A 97   73               STAA   TEMP1     SAVE FIRST DIGIT
 F04C 8D   D1               BSR    INHEX     GET SECOND DIGIT
 F04E 9B   73               ADDA   TEMP1     COMBINE 2 DIGITS INTO BYTE
 F050 39                    RTS

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

 F051 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

  F053 8D   4E       CHANGE  BSR    OUTS      SPACE
  F055 A6   84               LDAA   0,X
  F057 8D   33               BSR    OUT2H     PRINT BYTE
  F059 8D   48               BSR    OUTS      SPACE
  F05B 8D   E7               BSR    BYTE      GET NEW BYTE
  F05D A7   84               STAA   0,X
  F05F 20   55               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

  F061 9E   79       VIEW    LDX    XHI
  F063 C1   56               CMPB   #'V       VIEW SAME LOCATION
  F065 27   0A               BEQ    VIEW1
  F067 30   1F               DEX
  F069 C1   42               CMPB   #'B       BACK ONE LOCATION
  F06B 27   04               BEQ    VIEW1
  F06D 30   01               INX               UNDO BACK
  F06F 30   01               INX               INCREMENT MEMORY POINTER
  F071 9F   79       VIEW1   STX    XHI       PUT IT BACK
  F073 96   79               LDAA   XHI
  F075 8D   15               BSR    OUT2H     PRINT ADDRESS
  F077 96   7A               LDAA   XLOW
  F079 8D   11               BSR    OUT2H
  F07B 20   D6               BRA    CHANGE    ENTER CHANGE FUNCTION

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

  F07D 34   04       OUTCH   PSH    B         SAVE ACC B
  F07F F6   E004     OUTCH1  LDAB   ACIA      GET STATUS
  F082 57                    ASRB
  F083 57                    ASRB              SHIFT TDR TO CARRY
  F084 24   F9               BCC    OUTCH1
  F086 B7   E005             STAA   ACIA+1    SEND CHAR
  F089 35   04               PUL    B         RESTORE ACC B
  F08B 39                    RTS

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

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

  F08C 97   73       OUT2H   STAA   TEMP1
  F08E 44                    LSRA             SHIFT HIGH DIGIT OVER
  F08F 44                    LSRA
  F090 44                    LSRA
  F091 44                    LSRA
  F092 8D   00               BSR    OUTHR     OUTPUT FIRST DIGIT
  F094 84   0F       OUTHR   ANDA   #$0F      MASK LEFT DIGIT
  F096 8B   30               ADDA   #$30      MAKE ASCII
  F098 81   39               CMPA   #'9       IS IT 0-9
  F09A 23   02               BLS    OUT21
  F09C 8B   07               ADDA   #$7       MAKE IT A-F
  F09E 8D   DD       OUT21   BSR    OUTCH     PRINT IT
  F0A0 96   73               LDAA   TEMP1     GET BYTE BACK
  F0A2 39                    RTS

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

  F0A3 86   20       OUTS    LDAA   #$20      SPACE
  F0A5 20   D6               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.

  F0A7 8D   D4       PDATA2  BSR    OUTCH
  F0A9 30   01               INX              VIEW CHAR
  F0AB A6   84       PDATA   LDAA   X
  F0AD 81   04               CMPA   #4        END OF TEXT
  F0AF 26   F6               BNE    PDATA2
  F0B1 39                    RTS

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

```
F0B2 8D   85       JUMP   BSR    BADDR     GET ADDRESS
F0B4 6E   84              JMP    0,X       JUMP TO IT

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

F0B6 10CE 0070    CTRL   LDS    #STACK
F0BA 8D   50              BSR    PCRLF
F0BC 86   2B              LDAA   #'+
F0BE 8D   BD              BSR    OUTCH
F0C0 BD   F011            JSR    INCH      GET COMMAND
F0C3 1F   894D            TAB              SAVE COMMAND
F0C6 8D   DB              BSR    OUTS      PRINT SPACE

F0C8 8E   F0DF            LDX    #CMMD     COMMAND TABLE
F0CB E1   84      CTRL1   CMPB   0,X       MATCH COMMAND
F0CD 27   0C              BEQ    CTRL3     FOUND MATCH
F0CF 30   01              INX
F0D1 30   01              INX
F0D3 30   01              INX              SKIP OVER COMMAND
F0D5 6D   84              TST    0,X       TABLE ENDS WITH $00
F0D7 26   F2              BNE    CTRL1
F0D9 20   DB              BRA    CTRL

F0DB AE   01      CTRL3   LDX    1,X       LOAD XFER ADDRESS
F0DD 6E   84              JMP    0,X

F0DF 4A           CMMD    FCC    /J/
F0E0 F0B2                 FDB    JUMP
F0E2 4D                   FCC    /M/
F0E3 F051                 FDB    MEMORY
F0E5 4E                   FCC    /N/
F0E6 F061                 FDB    VIEW
F0E8 56                   FCC    /V/
F0E9 F061                 FDB    VIEW
F0EB 42                   FCC    /B/
F0EC F061                 FDB    VIEW
F0EE 5A                   FCC    /Z/
F0EF F265                 FDB    TRK00
F0F1 54                   FCC    /T/
F0F2 F275                 FDB    SECTR
F0F4 4C                   FCC    /L/
F0F5 F2F4                 FDB    LOAD
F0F7 24                   FCC    /$/
F0F8 F800                 FDB    SWTBUG
F0FA 57                   FCC    /W/
F0FB F298                 FDB    WRTSEC
F0FD 00                   FCB    $00       END OF TABLE

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

```
                        * DESTROYS: acc A, INDEX

  F0FE 8D   0C       SIGNON  BSR    PCRLF
  F100 8E   F116             LDX    #HELLO
  F103 8D   A6               BSR    PDATA
  F105 8E   F136             LDX    #HELP
  F108 8D   A1               BSR    PDATA
  F10A 20   AA               BRA    CTRL


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

  F10C 9F   75       PCRLF   STX    XTEMP1
  F10E 8E   F130             LDX    #CRLF
  F111 8D   98               BSR    PDATA
  F113 9E   75               LDX    XTEMP1    RESTORE INDEX
  F115 39                    RTS


                        ****************************************
                        * STRINGS
  F116 20 4D 43 36   HELLO   FCC    / MC6800 DISK-BUG - VER 3.5/
  F11A 38 30 30 20
  F11E 44 49 53 4B
  F122 2D 42 55 47
  F126 20 2D 20 56
  F12A 45 52 20 33
  F12E 2E 35
  F130 0D 0A 00 00   CRLF    FCB    $0D,$0A,00,00,00,04
  F134 00 04

  F136 4A 20 78 78   HELP    FCC    /J xxxx Jump to xxxx/
  F13A 78 78 20 4A
  F13E 75 6D 70 20
  F142 74 6F 20 78
  F146 78 78 78
  F149 0D 0A 00              FCB    $0D,$0A,00
                        *
  F14C 4D 20 78 78   HELP01  FCC    /M xxxx Examine and alter memory/
  F150 78 78 20 45
  F154 78 61 6D 69
  F158 6E 65 20 61
  F15C 6E 64 20 61
  F160 6C 74 65 72
  F164 20 6D 65 6D
  F168 6F 72 79
  F16B 0D 0A 00              FCB    $0D,$0A,00
  F16E 4E 20 20 20   HELP02  FCC    /N      Next memory/
```

```
F172 20 20 20 4E
F176 65 78 74 20
F17A 6D 65 6D 6F
F17E 72 79
F180 0D 0A 00            FCB   $0D,$0A,00
F183 56 20 20 20  HELP03 FCC   /V      Same memory/
F187 20 20 20 53
F18B 61 6D 65 20
F18F 6D 65 6D 6F
F193 72 79
F195 0D 0A 00            FCB   $0D,$0A,00
F198 42 20 20 20  HELP04 FCC   /B      Previous memory/
F19C 20 20 20 50
F1A0 72 65 76 69
F1A4 6F 75 73 20
F1A8 6D 65 6D 6F
F1AC 72 79
F1AE 0D 0A 00            FCB   $0D,$0A,00
                         *
F1B1 5A 20 78 78  HELP05 FCC   /Z xx   Select & restore drive xx/
F1B5 20 20 20 53
F1B9 65 6C 65 63
F1BD 74 20 26 20
F1C1 72 65 73 74
F1C5 6F 72 65 20
F1C9 64 72 69 76
F1CD 65 20 78 78
F1D1 0D 0A 00            FCB   $0D,$0A,00
                         *
F1D4 54 20 74 74  HELP06 FCC   /T ttss Read sector ss on track tt into buffer/
F1D8 73 73 20 52
F1DC 65 61 64 20
F1E0 73 65 63 74
F1E4 6F 72 20 73
F1E8 73 20 6F 6E
F1EC 20 74 72 61
F1F0 63 6B 20 74
F1F4 74 20 69 6E
F1F8 74 6F 20 62
F1FC 75 66 66 65
F200 72
F201 0D 0A 00            FCB   $0D,$0A,00
                         *
F204 4C 20 74 74  HELP07 FCC   /L ttss Load file. Show transfer address./
F208 73 73 20 4C
F20C 6F 61 64 20
F210 66 69 6C 65
F214 2E 20 53 68
F218 6F 77 20 74
F21C 72 61 6E 73
F220 66 65 72 20
F224 61 64 64 72
F228 65 73 73 2E
F22C 0D 0A 00            FCB   $0D,$0A,00
```

```
                        *
 F22F 57 20 74 74   HELP08  FCC    /W ttss Write buffer at $0100 /
 F233 73 73 20 57
 F237 72 69 74 65
 F23B 20 62 75 66
 F23F 66 65 72 20
 F243 61 74 20 24
 F247 30 31 30 30
 F24B 20
 F24C 0D 0A 00              FCB    $0D,$0A,00
                        *
 F24F 24 20 20 20   HELP09  FCC    /$      Go to SWTBUG/
 F253 20 20 20 47
 F257 6F 20 74 6F
 F25B 20 53 57 54
 F25F 42 55 47
 F262 0D 0A 04              FCB    $0D,$0A,04

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

 F265 BD   F044     TRK00   JSR    BYTE      GET DRIVE SELECT CODE
 F268 B7   E014             STAA   DRVREG
 F26B 97   7B               STAA   CURDRV
 F26D 86   0B               LDAA   #$0B
 F26F B7   E018             STAA   COMREG
 F272 7E   F0B6             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:
 F275 BD   F039     SECTR   JSR    BADDR     GET TRK & SEC
 F278 8E   0100             LDX    #BUFFER
 F27B 96   79               LDAA   XHI       GET TRACK
 F27D D6   7A               LDAB   XLOW      GET SECTOR
 F27F BD   F371             JSR    READ      READ SECTOR
 F282 27   33               BEQ    DUMP      NO ERROR

 F284 BD   F10C     ERROR   JSR    PCRLF
 F287 86   45               LDAA   #'E
 F289 BD   F07D             JSR    OUTCH
 F28C BD   F0A3             JSR    OUTS      PRINT E & SPACE
 F28F 1F   984D             TBA              GET ERROR NUMBER
 F292 BD   F08C             JSR    OUT2H
 F295 7E   F0B6             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:
F298 BD   F039     WRTSEC  JSR    BADDR     GET TRK & SEC
F29B 8E   0100             LDX    #BUFFER
F29E 96   79               LDAA   XHI       GET TRACK
F2A0 D6   7A               LDAB   XLOW      GET SECTOR
F2A2 BD   F3D4             JSR    WRITE     WRITE SECTOR
F2A5 26   DD               BNE    ERROR

F2A7 BD   F10C             JSR    PCRLF
F2AA 86   4F               LDAA   #'O
F2AC BD   F07D             JSR    OUTCH
F2AF 86   4B               LDAA   #'K
F2B1 BD   F07D             JSR    OUTCH
F2B4 7E   F0B6     WRTSC9  JMP    CTRL

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

F2B7 8E   0100     DUMP    LDX    #BUFFER
F2BA 86   10               LDAA   #16       NUMBER OF LINES
F2BC 34   02       DUMP1   PSH    A
F2BE BD   F10C             JSR    PCRLF
F2C1 C6   10               LDAB   #16       NUMBER OF BYTES
F2C3 9F   77               STX    XTEMP2
F2C5 A6   84       DUMP2   LDAA   0,X
F2C7 BD   F08C             JSR    OUT2H     PRINT 2 HEX
F2CA BD   F0A3             JSR    OUTS      PRINT SPACE
F2CD 30   01               INX
F2CF 5A                    DECB             DONE WITH LINE
F2D0 26   F3               BNE    DUMP2

F2D2 C6   10               LDAB   #16       NUMBER OF BYTES
F2D4 9E   77               LDX    XTEMP2
F2D6 A6   84       DUMP3   LDAA   0,X       GET CHARACTER
F2D8 84   7F               ANDA   #$7F      MASK MSB
F2DA 81   7D               CMPA   #$7D
F2DC 2E   04               BGT    DUMP31
F2DE 81   1F               CMPA   #$1F      IS IT CONTROL
F2E0 2E   02               BGT    DUMP32
F2E2 86   5F       DUMP31  LDAA   #'_       DUMMY
F2E4 BD   F07D     DUMP32  JSR    OUTCH
F2E7 30   01               INX
F2E9 5A                    DECB             DONE WITH LINE
F2EA 26   EA               BNE    DUMP3

F2EC 35   02               PUL    A
F2EE 4A                    DECA             DONE WITH DUMP
```

```
  F2EF 26   CB              BNE    DUMP1
  F2F1 7E   F0B6    DUMP9   JMP    CTRL

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

  F2F4 BD   F039    LOAD    JSR    BADDR     GET TRACK AND SECTOR
  F2F7 BF   0100            STX    BUFFER
 >F2FA BD   F30D            JSR    LOADER    LOAD BINARY FILE
  F2FD BD   F0A3            JSR    OUTS
  F300 96   79              LDAA   XHI
  F302 BD   F08C            JSR    OUT2H     PRINT XFER ADDRESS
  F305 96   7A              LDAA   XLOW
  F307 BD   F08C            JSR    OUT2H
  F30A 7E   F0B6            JMP    CTRL

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

  F30D 8E   0200    LOADER  LDX    #BUFFER+256
  F310 9F   75              STX    XTEMP1
  F312 10DF 71              STS    SAVSTK    SAVE STACK FOR RETURN

                    * DO UNTIL END OF FILE

  F315 8D   34      LOAD1   BSR    GETCH     GET A CHAR FROM BUFFER
  F317 81   02              CMPA   #$02      DATA RECORD HEADER
  F319 27   0E              BEQ    LOAD2     BR IF SO
  F31B 81   16              CMPA   #$16      XFR ADDRESS HEADER
  F31D 26   F6              BNE    LOAD1     LOOP IF NEITHER
  F31F 8D   2A              BSR    GETCH     TRANSFER ADDRESS
  F321 97   79              STAA   XHI
  F323 8D   26              BSR    GETCH
  F325 97   7A              STAA   XLOW
  F327 20   EC              BRA    LOAD1

  F329 8D   20      LOAD2   BSR    GETCH     GET LOAD ADDRESS
  F32B 97   77              STAA   XTEMP2
  F32D 8D   1C              BSR    GETCH
  F32F 97   78              STAA   XTEMP2+1
  F331 8D   18              BSR    GETCH     GET BYTE COUNT
  F333 1F   894D            TAB               SAVE
  F336 27   DD              BEQ    LOAD1     BR IF COUNT=0

  F338 34   04      LOAD3   PSH    B
  F33A 8D   0F              BSR    GETCH     GET NEXT BYTE
  F33C 35   04              PUL    B
  F33E 9E   77              LDX    XTEMP2    GET LOAD ADDRESS
```

```
  F340 A7   84              STAA   0,X
  F342 30   01              INX
  F344 9F   77              STX    XTEMP2
  F346 5A                   DECB             END OF DATA RECORD?
  F347 26   EF              BNE    LOAD3
  F349 20   CA              BRA    LOAD1     GET ANOTHER RECORD

                    * GET CHARACTER ROUTINE - READS SECTOR IF NECESSARY

  F34B 9E   75      GETCH   LDX    XTEMP1
  F34D 8C   0200            CPX    #BUFFER+256 OUT OF DATA
  F350 26   11              BNE    GETCH4    GO READ CHARACTER

  F352 8E   0100    GETCH2  LDX    #BUFFER
  F355 A6   84              LDAA   0,X       GET NEXT TRACK
  F357 E6   01              LDAB   1,X       GET NEXT SECTOR
  F359 27   12              BEQ    GETCH6    0 SECTOR ENDS LOAD
 >F35B BD   F371    GETCH3  JSR    READ      NEXT SECTOR
  F35E 26   0A              BNE    GETCH5    READ ERROR
  F360 8E   0104            LDX    #BUFFER+4 POINT PAST LINK
  F363 A6   84      GETCH4  LDAA   0,X       GET CHAR FROM BUFFER
  F365 30   01              INX
  F367 9F   75              STX    XTEMP1
  F369 39                   RTS

  F36A 7E   F0B6    GETCH5  JMP    CTRL      ERROR RETURN

  F36D 10DE 71      GETCH6  LDS    SAVSTK    CLEAR SUBROUTINES
  F370 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.

 F371 8D   30       READ    BSR     SEEK        SET TRACK AND SECTOR
 F373 86   8C               LDAA    #RDCMND
 F375 1A   10               SEI                 NO INTERRUPTS ALLOWED
 F377 B7   E018             STAA    COMREG      READ SECTOR 10mS DELAY
>F37A BD   F3CD             JSR     DELAY

 F37D B6   E014     READ2   LDAA    DRQREG      SET STATUS
 F380 2B   07               BMI     READ3       BRANCH IF DATA PRESENT
 F382 27   F9               BEQ     READ2       LOOP IF BUSY
 F384 F6   E018             LDAB    COMREG      GET ERROR
 F387 20   0C               BRA     READ6       REPORT ERROR

 F389 B6   E01B     READ3   LDAA    DATREG      READ SECTOR LOOP
 F38C A7   84               STAA    0,X
 F38E 30   01               INX
>F390 7E   F37D             JMP     READ2

 F393 8D   06               BSR     WAIT        WAIT UNTIL 1771 IS DONE
 F395 C5   1C       READ6   BITB    #RDMSK
 F397 12                    NOP
 F398 1C   EF               CLI                 ENABLE INTURRUPTS
 F39A 39                    RTS

 F39B F6   E018     WAIT    LDAB    COMREG      GET STATUS
 F39E C5   01               BITB    #BUSY
 F3A0 26   F9               BNE     WAIT
 F3A2 39                    RTS
```

```
                        * SEEK THE SPECIFIED TRACK
  F3A3 F7   E01A    SEEK    STAB   SECREG    SET SECTOR
  F3A6 C1   0A              CMPB   #10
  F3A8 22   04              BHI    SIDE1
  F3AA D6   7B              LDAB   CURDRV
  F3AC 20   04              BRA    SEEK2
  F3AE D6   7B      SIDE1   LDAB   CURDRV
  F3B0 CA   40              ORAB   #$40      SIDE 1
  F3B2 F7   E014    SEEK2   STAB   DRVREG
  F3B5 B1   E019            CMPA   TRKREG
  F3B8 27   10              BEQ    SEEK4     TRACK IS CORRECT
  F3BA B7   E01B            STAA   DATREG    CHANGE TRACK
 >F3BD BD   F3CD            JSR    DELAY
  F3C0 86   1B              LDAA   #SKCMND
  F3C2 B7   E018            STAA   COMREG    ISSUE SEEK COMMAND
 >F3C5 BD   F3CD            JSR    DELAY
  F3C8 8D   D1              BSR    WAIT
 >F3CA 7E   F3CD    SEEK4   JMP    DELAY     JUMP AND RETURN


 >F3CD BD   F3D0    DELAY   JSR    DELAY1
 >F3D0 BD   F3D3    DELAY1  JSR    DELAY2
  F3D3 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.

  F3D4 8D   CD      WRITE   BSR    SEEK      SET TRACK AND SECTOR
  F3D6 86   AC              LDAA   #WTCMND
  F3D8 1A   10              SEI              NO INTERRUPTS ALLOWED
  F3DA B7   E018            STAA   COMREG    ISSUE WRITE COMMAND
 >F3DD BD   F3CD            JSR    DELAY
  F3E0 E6   84              LDAB   0,X

  F3E2 B6   E014    WRITE2  LDAA   DRQREG    SET STATUS
  F3E5 2B   07              BMI    WRITE3    BRANCH IF READY
  F3E7 27   F9              BEQ    WRITE2    LOOP IF BUSY
  F3E9 F6   E018            LDAB   COMREG    GET STATUS
  F3EC 20   0A              BRA    WRITE6    DONE

  F3EE F7   E01B    WRITE3  STAB   DATREG    SEND TO DISK
  F3F1 30   01              INX
  F3F3 E6   84              LDAB   0,X       GET NEXT BYTE
 >F3F5 7E   F3E2            JMP    WRITE2

  F3F8 C5   5C      WRITE6  BITB   #WTMSK    MASK ERRORS
  F3FA 12                   NOP
```

```
  F3FB 1C   EF              CLI             ENABLE INTERRUPTS
  F3FD 39                   RTS

                            END    START

0 ERROR(S) DETECTED
```

SYMBOL TABLE:

ACIA	E004	BADDR	F039	BUFFER	0100	BUSY	0001	BYTE	F044		
CHANGE	F053	CMMD	F0DF	COMREG	E018	CRLF	F130	CTLREG	0011		
CTRL	F0B6	CTRL1	F0CB	CTRL3	F0DB	CURDRV	007B	DATREG	E01B		
DELAY	F3CD	DELAY1	F3D0	DELAY2	F3D3	DRQ	0002	DRQREG	E014		
DRVREG	E014	DUMP	F2B7	DUMP1	F2BC	DUMP2	F2C5	DUMP3	F2D6		
DUMP31	F2E2	DUMP32	F2E4	DUMP9	F2F1	ERROR	F284	FDC	E018		
GETCH	F34B	GETCH2	F352	GETCH3	F35B	GETCH4	F363	GETCH5	F36A		
GETCH6	F36D	HELLO	F116	HELP	F136	HELP01	F14C	HELP02	F16E		
HELP03	F183	HELP04	F198	HELP05	F1B1	HELP06	F1D4	HELP07	F204		
HELP08	F22F	HELP09	F24F	HEXERR	F036	HEXRTS	F033	INCH	F011		
INHEX	F01F	INITA	F004	JUMP	F0B2	LOAD	F2F4	LOAD1	F315		
LOAD2	F329	LOAD3	F338	LOADER	F30D	MEMORY	F051	OUT21	F09E		
OUT2H	F08C	OUTCH	F07D	OUTCH1	F07F	OUTHR	F094	OUTS	F0A3		
PCRLF	F10C	PDATA	F0AB	PDATA2	F0A7	RAM	0000	RDCMND	008C		
RDMSK	001C	READ	F371	READ2	F37D	READ3	F389	READ6	F395		
RESETA	0013	ROM	F000	RSCMND	0018	SAVSTK	0071	SECREG	E01A		
SECTR	F275	SEEK	F3A3	SEEK2	F3B2	SEEK4	F3CA	SIDE1	F3AE		
SIGNON	F0FE	SKCMND	001B	STACK	0070	START	F000	SWTBUG	F800		
TEMP1	0073	TEMP2	0074	TRK00	F265	TRKREG	E019	VERMSK	0018		
VIEW	F061	VIEW1	F071	WAIT	F39B	WRITE	F3D4	WRITE2	F3E2		
WRITE3	F3EE	WRITE6	F3F8	WRTSC9	F2B4	WRTSEC	F298	WTCMND	00AC		
WTMSK	005C	XHI	0079	XLOW	007A	XTEMP1	0075	XTEMP2	0077		

+++