

# Atari 815 DUAL DISK CONTROLLER by Bob Warne with additional comments, lables & equates by Dave Staugas#

Thanks to Curt Vendel and his Atari museum [atarimuseum.com](http://atarimuseum.com) for publishing the source code at [AtariAge](http://AtariAge) and kheller2 from [AtariAge](http://AtariAge) for post production of the source code. Thank you very much to both of you from the Atari community.

## Source Code files#

- [DUAL 815 DISK CONTROLLER source code - original](#) ; original source code from Curt Vendel and his Atari museum [atarimuseum.com](http://atarimuseum.com)
- [DUAL 815 DISK CONTROLLER - post processed original source code](#) ; same as above, but with post production for better reading from kheller2 from [AtariAge](http://AtariAge)

## Source Code#

```
.TITLE DUAL 815 DISK CONTROLLER
;
;
; *****
; *
; *   815 DUAL DISK CONTROLLER   *
; *
; *****
;
;
; PROGRAMMER:   BOB WARNE
;
; ADDITIONAL COMMENTS, LABELS & EQUATES BY DAVE STAUGAS
;
;
; .ENABLE AMA
; .RADIX 10
; .ASECT
;
; MEMORY MAP:
;
; 0000-00FF  RAM (USED FOR DATA BUFFER)
; 0100-0101  SYNCHRONOUS SERIAL DATA ADAPTER REGISTERS (MOTOROLA MC68B52)
; 0180-01FF  RAM (STACK AND PROGRAM SCRATCH)
; 0200-021F  MAPS TO PIA #2
; 0280-029F  MAPS TO PIA #3
; 0380-039F  MAPS TO PIA #1
; 0800-0FFF  CONTROLLER ROM (PART #1)
; 7000-77FF  CONTROLLER ROM (PART #2)
;
;
; EQUATES^H
;
;
BIT7   =   ^H080
BIT6   =   ^H040
BIT5   =   ^H020
BIT4   =   ^H010
BIT3   =   ^H008
BIT2   =   ^H004
```

```

BIT1    =      ^H002
BIT0    =      ^H001
;
; PAGE 0
;
ZPRL    =      ^H0000      ;START OF ZERO PAGE DATA BUFFER
ZP80    =      ^H0080
ZP91    =      ^H0091
FRMCRC  =      ^H00A0
ZPCRC   =      ^H00AE
ZPNTL   =      ^H00C0
ZPNTH   =      ^H00C1
SCTPT   =      ^H00F6
CRCLP   =      ^H00F7
IDAM    =      ^H00F8
TRKNO   =      ^H00F9
BYTE00  =      ^H00FA
SCTNO   =      ^H00FB
BYTE01  =      ^H00FC
CRC1    =      ^H00FD
CRC2    =      ^H00FE
BYTE4E  =      ^H00FF
;
      . =      ^H0100      ;START OF SSDA HARDWARE SPACE
;
; THE NEXT TWO ADDRESSES MAP TO THE MOTOROLA MC68B52 (SSDA)
;
; DATA BITS ARE REVERSED (0 -> 7, 1 -> 6, 2 -> 5, ETC)
; FROM THE BIT DEFINITIONS IN THE LITERATURE SO THAT
; SERIAL DATA IS SHIFTED MSB FIRST.
;
SSDA0:  . = .+1
;
; WHEN READ, THIS ADDR IS A STATUS REGISTER.
; BITS FOR THIS REGISTER ARE DEFINED:
;
; 7 - RECEIVER DATA AVAILABLE (RDA)
; 6 - TRANSMITTER DATA REGISTER AVAILABLE (TDRA)
; 5 - DATA CARRIER DETECT BAR (DCD)
; 4 - CLEAR TO SEND BAR (CTS)
; 3 - TRANSMITTER UNDERFLOW (TUF)
; 2 - RECEIVER OVERRUN (RX OVRN)
; 1 - RECEIVER PARITY ERROR (PE)
; 0 - INTERRUPT REQUEST (IRQ)
;
; BIT EQUATES FOR SSDA0 (READ):
;
RDA     =      BIT7      ;RECEIVER DATA AVAILABLE
TDRA    =      BIT6      ;TRANSMITTER DATA REGISTER AVAILABLE
DCD     =      BIT5      ;DATA CARRIER DETECT BAR
CTS     =      BIT4      ;CLEAR TO SEND BAR
TUF     =      BIT3      ;TRANSMITTER UNDERFLOW
RXOVRN  =      BIT2      ;RECEIVER OVERRUN
PE      =      BIT1      ;RECEIVER PARITY ERROR
IRQ     =      BIT0      ;INTERRUPT REQUEST
;
;
; WHEN WRITTEN, THIS ADDRESS IS CONTROL REGISTER #1
; BITS FOR THIS REGISTER ARE DEFINED:

```

```

;
; 7 - RECEIVER RESET (RX RS)
; 6 - TRANSMITTER RESET (TX RS)
; 5 - STRIP SYNCH CHARACTERS (STRIP SYNCH)
; 4 - CLEAR SYNCH
; 3 - TRANSMITTER INTERRUPT ENABLE (TIE)
; 2 - RECEIVER INTERRUPT ENABLE (RIE)
; 1 - ADDRESS CONTROL #1 (AC1)
; 0 - ADDRESS CONTROL #2 (AC2)
;
; BIT EQUATES FOR SSDA0 (WRITE):
;
RXRS      =      BIT7          ;RECEIVER RESET
TXRS      =      BIT6          ;TRANSMITTER RESET
STRIPS    =      BIT5          ;STRIP SYNCH CHARACTERS
CLEARS    =      BIT4          ;CLEAR SYNCH
TIE       =      BIT3          ;TRANSMITTER INTERRUPT ENABLE
RIE       =      BIT2          ;RECEIVER INTERRUPT ENABLE
AC1       =      BIT1          ;ADDRESS CONTROL #1
AC2       =      BIT0          ;ADDRESS CONTROL #2
SSDA1:    =      .+. +1
;
; A READ OF THIS ADDRESS ACCESSES DATA FROM THE SSDA RECEIVE
; DATA FIFO.
;
; A WRITE TO THIS ADDRESS ACCESSES ANY OF 4 INTERNAL REGISTERS
; WHICH ARE SELECTED BY WRITING APPROPRIATE DATA TO THE AC1, AC2
; DATA BITS IN CONTROL REGISTER #1.
;
; BIT DEFINITIONS FOR THE FOUR WRITE ONLY REGISTERS:
;
; AC1=0 & AC2=0 SELECT CONTROL REGISTER #2:
;
; 7 - PERIPHERAL CONTROL #1 (PC1)    ALWAYS 1
; 6 - PERIPHERAL CONTROL #2 (PC2)    ALWAYS 0
; 5 - 1 BYTE/2 BYTE TRANSFER        ALWAYS 0 (TWO BYTE)
; 4 - WORD LENGTH SELECT 1 (WS1)    ALWAYS 1
; 3 - WORD LENGTH SELECT 2 (WS2)    ALWAYS 1
; 2 - WORD LENGTH SELECT 3 (WS3)    ALWAYS 0
; 1 - TRANSMIT SYNCH CODE ON UNDERFLOW 1
; 0 - ERROR INTERRUPT ENABLE (EIE)   1
;
; BIT EQUATES FOR CONTROL REGISTER #2:
;
PC1       =      BIT7          ;PERIPHERAL CONTROL #1
PC2       =      BIT6          ;PERIPHERAL CONTROL #2
BYTE12    =      BIT5          ;1 OR 2 BYTE TRANSFER
WS1       =      BIT4          ;WORD LENGTH SELECT #1
WS2       =      BIT3          ;WORD LENGTH SELECT #2
WS3       =      BIT2          ;WORD LENGTH SELECT #3
TXSUND    =      BIT1          ;TRANSMIT SYNCH ON UNDERFLOW
EIE       =      BIT0          ;ERROR INTERRUPT ENABLE
;
;
; AC1=1 & AC2=0 SELECTS CONTROL REGISTER #3
;
; 7 - EXTERNAL/INTERNAL SYNCH MODE CONTROL (E/I SYNCH) =0
; 6 - 1 SYNCH CHAR/2 SYNCH CHAR MODE CONTROL =0
; 5 - CLEAR CTS BAR STATUS =0

```

```

; 4 - CLEAR TRANSMITTER UNDERFLOW STATUS (CTUF)           =0
; 3 - NOT USED
; 2 - NOT USED
; 1 - NOT USED
; 0 - NOT USED
;
; BIT EQUATES FOR CONTROL REGGISTER #3
;
EISYNC = BIT7           ;EXTERNAL/INTERNAL SYNCH MODE CONTROL
SYNCH12 = BIT6          ;SYNCH CHAR MODE CONTROL
CCTS = BIT5             ;CLEAR CTS BAR STATUS
CTUF = BIT4             ;CLEAT TUF STATUS
;
;
; AC1=0 & AC2=1 SELECTS SYNCH CODE REGISTER
;
; AC1=1 & AC2=1 SELECTS TRANSMIT DATA FIFO
      = ^H0180          ;START OF STACK & SCRATCH RAM
;
STPNT:  =.
RAM3:   =.+1
;
; NEXT FIVE BYTES ARE SAVE AREA FOR COMMAND FRAME INPUT
;
UNITNB: =.+1           ;DRIVE UNIT # (ASCII)
CTLCMD: =.+1           ;CONTROLLER COMMAND
AUX1:   =.+1           ;LSB OF 16-BIT LOGICAL SECTOR #
AUX2:   =.+1           ;MSB OF 16-BIT LOGICAL SECTOR #
RCDCSM: =.+1           ;COMMAND FRAME CHECKSUM
;
;
;
TMPOUT: =.+1           ;ALL SERIAL OUTPUT SHIFTED FROM THIS ADDR
DICKSM: =.+1           ;DATA FIELD CHECKSUM TEMP
      =.+8             ;SPARE
;
; NEXT 4 BYTES SENT TO COLLEEN FOR STATUS COMMAND ON BOTTOM DRV
;
STFLG2: =.+1           ;BOTTOM DRV STATUS FLAGS
DEVST2: =.+1           ;BOTTOM DRV XTRA STATUS
WCTLB2: =.+1           ;BOTTOM DRV WORST CASE TIMEOUT (LSB)
WCTMB2: =.+1           ;DELAY (MSB)
;
; NEXT 2 BYTES ARE CODES FOR PORTB2:
;
EGON:   =.+1           ;ERASE GATE TURN ON CODE
WGOFF:  =.+1           ;WRITE GATE TURN OFF CODE (LEAVES EG ON)
;
; NEXT 2 BYTES ARE USED FOR CRC GENERATION:
;
BAKER:  =.+1           ;CRC GENERATION SCRATCH 1
ABLE:   =.+1           ;CRC GENERATION SCRATCH 2
GAPBYT: =.+1           ;BYTE INIT'D TO $4E FOR CRC WRITE
;
;
DH80T:  =.+1           ;DOUBLE HEAD & 80 TRK TEST BYTE
;
; 7 - DOUBLE=1 / SINGLE=0 HEAD DRIVE
; 6 - 80 TRK=1 / 40 TRK=0 TRACK DENSITY

```

```

;
; BIT EQUATES:
;
DBLHED = BIT7 ;DOUBLE/SINGLE HEADED DRVS
TPI96 = BIT6 ;TRACK DENSITY BIT
;
; SPARE
;
STPDIR: .+.1
RDCRC2: .+.1
RDCRC1: .+.1
STPNB2: .+.1 ;# OF STEPS TO DESIRED TRACK, BOTTOM DRV
PRTRK2: .+.1 ;PRESENT TRACK #, BOTTOM DRV
SPSDC2: .+.1 ;LOOP COUNTER FOR BOTTOM DRV SHUTDOWN TIMER
SPMDC2: .+.1
SCRT1: .+.1
SCRT2: .+.1
;
SCTTBL: .+.11 ;START OF 11-BYTE ID FIELD
SPMDLC: .+.1 ;1 SEC LOOP COUNT FOR SPM SHUT-OFF (4 * 125 MS)
; SPARE
;
STPSCR: .+.1 ;STEP ROUTINE SCRATCH
PRSTRK: .+.1 ;PRESENT TRACK #, TOP DRV
STPNBR: .+.1 ;# OF STEPS TO DESIRED TRK #, TOP DRV
;
; THE NEXT 4 BYTES SENT TO COLLEEN FOR STATUS COMMAND, TOP DRV
;
STFLGS: .+.1 ;STATUS FLAGS (TOP DRV) SHIPPED TO COLLEEN
;
; BIT DEFINITIONS FOR CONTROLLER STATUS FLAGS BYTE:
;
; 7 - TRACK DENSITY (0=40 TRK, 1=80 TRK)
; 6 - SERIAL BAUD RATE (0=19.2KB, 1=38.4KBAUD)
; 5 - SECTOR SIZE (0=128B/S, 1=256 BYTES/SECTOR)
; 4 - MOTOR/SELECT LITE STATUS (0=OFF, 1=ON)
; 3 - WRITE PROTECT STATUS (1=WRITE PROTECTED)
; 2 - READ/WRITE ERROR (SET IF ERROR, LAST R/W)
; 1 - INVALID DATA FRAME " "
; 0 - INVALID COMMAND FRAME " "
;
; BIT EQUATES FOR STFLGS:
;
TRK80 = BIT7 ;TRACK DENSITY
BAUD38 = BIT6 ;SERIAL BAUD RATE
SEC256 = BIT5 ;SECTOR SIZE
MTR0 = BIT4 ;MOTOR ON STATUS
WPS0 = BIT3 ;WRITE PROTECT
RWE0 = BIT2 ;READ/WRITE ERROR
INVDF0 = BIT1 ;INVALID DATA FRAME ERROR
INVCF0 = BIT0 ;INVALID COMMAND FRAME ERROR
;
DEVSTA: .+.1 ;TOP DRV EXTRA STATUS
;
; BIT DEFINITIONS FOR EXTRA STATUS BYTE:
;
; 7 - N/A
; 6 - N/A
; 5 - N/A
; 4 - NO ID MARK?
; 3 - N/A

```

```

; 2 - N/A
; 1 - N/A
; 0 - N/A
;
; BIT EQUATES FOR DEVSTA:
;
NOID      =          BIT4          ;NO ID MARK FOUND?
DUM10    =          BIT3          ;???
;
WCTLSB:  .=.+1          ;WORST CASE TIMEOUT SENT TO COLLEEN (LSB)
WCTMSB:  .=.+1          ;(MSB)
;
PACNTR:  .=.+1          ;R/W ERROR RETRY COUNT
          .=.+2          ;SPARE
SSCTMS:  .=.+1          ;MSB,16-BIT LOGICAL SECTOR # (SECTOR BREAKDOWN)
SSCTLS:  .=.+1          ;LSB, SAVED BY SEC BRKDOWN BUT NOT USED
          .=.+1          ;SPARE
TRKNBR:  .=.+1          ;TRACK # COMPUTED IN SECTOR BREAK-DOWN
SCTNBR:  .=.+1          ;SECTOR # FROM SECTOR BREAK-DOWN
INMODE:  .=.+1          ;INTERNAL MODE REGISTER
;
; BIT DEFINITIONS FOR INMODE:
;
; 7 - 19.2/38.4 KBAUD SERIAL DATA RATE (19.2 KBAUD = 1)
; 6 - DRIVE SELECT (TOP DRIVE = 1, BOTTOM DRIVE = 0)
; 5 - SIDE SELECT (FOR DUAL HEADED DRIVES, SIDE0=0, SIDE1=1)
; 4 - N/A
; 3 - N/A
; 2 - SPM2 NOT ON >1 SEC IF SET
; 1 - N/A
; 0 - IF SET, WITHIN COMMAND INPUT ROUTINE (ALSO USED FOR DATA/CMD FRAME
;          CHECKSUM ROUTINE SELECT)
;
; BIT EQUATES FOR INMODE:
;
BAUD19   =          BIT7          ;SERIAL DATA RATE SELECT
TOPDRV   =          BIT6          ;DRIVE SELECT
SIDE1    =          BIT5          ;SIDE SELECT
DUM1     =          BIT4          ;DUMMY BIT NOT USED
DUM2     =          BIT3          ;DUMMY BIT NOT USED
S2NRDY   =          BIT2          ;BOTTOM DRV, SPM NOT READY
DUM3     =          BIT1          ;DUMMY BIT NOT USED
CMDFIN   =          BIT0          ;COMMAND FRAME INPUT
CRCTST   =          BIT0          ;CRC TESTING INDICATOR
DUM11    =          BIT0          ;DUMMY BIT USED
;
;
INSTAT:  .=.+1          ;INTERNAL STATUS REGISTER
;
; BIT DEFINITIONS FOR INTERNAL STATUS REGISTER:
;
; 7 - RECEIVING NEW COMMAND WORDS IF SET
; 6 - BUSY, SET WHENEVER GOING TO A ROUTINE THAT WILL KEEP US
;     AWAY FROM THE COMMAND FRAME MONITOR MORE THAN 400 MS
; 5 - SHUTDOWN IN PROGRESS, WHEN SET INDICATES CONTROL SHOULD RTS
;     FROM MAIN MONITOR--DOES NOT INDICATE 400 MS ABSENCE
; 4 - NO DISK
; 3 - FORMAT ERROR IF SET, AND READ AFTER WRITE FLAG!
; 2 - SPINDLE MOTOR HAS NOT BEEN ON 1 SEC IF SET

```

```

; 1 - COMMAND RECEIVE ERROR IF SET
; 0 - IF SET, COLLEEN HAS NOT BEEN WITH US 100 MS+
;
; BIT EQUATES FOR INSTAT:
;
RECCMD = BIT7 ;RECEIVING COMMAND
BUSY = BIT6 ;CONTROLLER BUSY
SHTDWN = BIT5 ;SHUTDOWN IN PROGRESS
NODISK = BIT4 ;NO DISK
FMTER = BIT3 ;FORMAT ERROR
RDAFWR = BIT3 ;READ AFTER WRITE INDICATOR
S1NRDY = BIT2 ;SPM TOP DRIVE NOT READY
CRECR = BIT1 ;COMMAND RECEIVE ERROR
NO800 = BIT0 ;COLLEEN HAS NOT BEEN WITH US
;
;
INSCRT: .+.1 ;LAST INDEX +1 TO TERMINATE INPUT OR CHECKSUM
TMRSCR: .+.1 ;TIMER SCRATCH
FSCTBP: .+.1 ;FORMAT VERIFY ERROR RETRY COUNT
BFRPNT: .+.1 ;FORMAT BAD SECTOR # BUFFER INDEX
        .+.1 ;SPARE
TMPSB: .+.1 ;*2 TEMP STORAGE FOR SECTOR BUILD-UP
MSBSB: .+.1 ;MSB FOR SECTOR BUILD-UP
CSSCRT: .+.1
YSAVE: .+.1
        .+.1 ;SPARE
CTABL: .+.1 ;JUMP INDIRECT ADDRESS FOR COMMAND INTERPRETER
CTABH: .+.1 ;MSB
CISCRT: .+.1 ;COMMAND INTERPRETER SCRATCH
SPMSDC: .+.1 ;LOOP COUNTER FOR TOP DRV SHUTDOWN TIMER
        = ^H0200 ;START OF PIA #3 ADDR SPACE
;
PORTA3: .+.1 ;PORT A, ONBOARD PIA #3
;
; BIT ASSIGNMENTS FOR PORT A, PIA #3:
;
; 7 - DRIVE ADDR DECODE INPUT
; 6 - N/A
; 5 - N/A
; 4 - WRITE LIGHT, TOP DRIVE OUTPUT
; 3 - READ LIGHT, TOP DRIVE OUTPUT
; 2 - 96/48 TPI SENSE (96 = 0) INPUT
; 1 - MOTOR ON/DRV SELECT, BOTTOM DRV OUTPUT
; 0 - DRIVE ADDR DECODE 2 INPUT
;
; BIT EQUATES FOR PORTA3:
;
DRVAD1 = BIT7 ;DRIVE ADDRESS DECODE #1
DUM4 = BIT6 ;DUMMY BIT NOT USED
DUM5 = BIT5 ;DUMMY BIT NOT USED
WLTOP = BIT4 ;WRITE LIGHT, TOP DRIVE
RLTOP = BIT3 ;READ LIGHT, TOP DRIVE
TPISNS = BIT2 ;TPI SENSE
SELBOT = BIT1 ;MOTOR ON/DRV SEL BOTTOM DRIVE
DRVAD2 = BIT0 ;DRIVE ADDRESS DECODE #2
;
PADDR3: .+.1 ;PORT A, PIA #3 DATA DIRECTION CONTROL (1=OUTPUT TO PIA)
PORTB3: .+.1 ;PORT B, PIA #3
;

```

```

; BIT DEFINITIONS FOR PORT B, PIA #3:
;
; 7 - DRIVE MANUFACTURER (=0 IF NOT MPI)          INPUT
; 6 - # OF HEAD(S) (DOUBLE HEADED IF =0)         INPUT
; 5 - BOTTOM DRV STEPPER MOTOR PHASE 4           OUTPUT
; 4 - BOTTOM DRV STEPPER MOTOR PHASE 3           OUTPUT
; 3 - BOTTOM DRV STEPPER MOTOR PHASE 2           OUTPUT
; 2 - BOTTOM DRV STEPPER MOTOR PHASE 1           OUTPUT
; 1 - EXTENDED PROM SPACE INSTALLED IF =0       INPUT
; 0 - INNER TRKS AMPLIFIER ADJUST (0=TRK>20)    OUTPUT
;
; BIT EQUATES FOR PORTB3:
;
DMANUF  =          BIT7          ;DRIVE MANUFACTURER
HEADNO  =          BIT6          ;# OF HEADS/DRV
SPBOT4  =          BIT5          ;BOTTOM STEPPER PHASE 4
SPBOT3  =          BIT4          ;BOTTOM STEPPER PHASE 3
SPBOT2  =          BIT3          ;BOTTOM STEPPER PHASE 2
SPBOT1  =          BIT2          ;BOTTOM STEPPER PHASE 1
XTPROM  =          BIT1          ;EXTENDED PROM SPACE
AMPADJ  =          BIT0          ;INNER TRACK AMPLIFIER ADJUST
;
;
PBDDR3:  .=.+1                ;DATA DIRECTION CONTROL, PORT B PIA #3
PIA3EC:  .=.+1
TIMR3R  =          ^H0215
TIMR23  =          ^H021F
          .=          ^H0280          ;START OF PIA #2 ADDR SPACE
PORTA2:  .=.+1                ;PORT A, PIA #2
;
; BIT DEFINITIONS FOR PORT A, PIA #2:
;
; 7 - WRITE LIGHT, BOTTOM DRIVE (0=ON)           OUTPUT
; 6 - READ LIGHT, BOTTOM DRIVE (0=ON)           OUTPUT
; 5 - N/A
; 4 - BOTTOM DRIVE, BOTTOM HEAD READ (0=ON)      OUTPUT
; 3 - BOTTOM DRIVE, TOP HEAD READ (0=ON)        OUTPUT
; 2 - N/A
; 1 - TOP DRIVE, BOTTOM HEAD READ (1=ON)        OUTPUT
; 0 - TOP DRIVE, TOP HEAD READ (1=ON)          OUTPUT
;
; BIT EQUATES FOR PORTA2:
;
WLBOT   =          BIT7          ;WRITE LIGHT, BOTTOM DRV
RLBOT   =          BIT6          ;READ LIGHT, BOTTOM DRV
DUM7    =          BIT5          ;DUMMY BIT NOT USED
BDBHRD  =          BIT4          ;BOTTOM DRV, BOTTOM HEAD, READ
BDTHRD  =          BIT3          ;BOTTOM DRV, TOP HEAD, READ
DUM8    =          BIT2          ;DUMMY BIT NOT USED
TDBHRD  =          BIT1          ;TOP DRIVE, BOTTOM HEAD, READ
TDTHRD  =          BIT0          ;TOP DRIVE, TOP HEAD, READ
;
PADDR2:  .=.+1                ;DATA DIRECTION CONTROL REG, PORT A, PIA #2
PORTB2:  .=.+1                ;PORT B, PIA #2
;
; BIT DEFINITIONS FOR PORT B, PIA #2:
; (ALL BITS ENABLED WHEN 0)
;
; 7 - BOTTOM DRIVE, BOTTOM HEAD ERASE           OUTPUT

```



```

; 6 - BOTTOM DRIVE, BOTTOM HEAD WRITE          OUTPUT
; 5 - BOTTOM DRIVE, TOP HEAD ERASE             OUTPUT
; 4 - BOTTOM DRIVE, TOP HEAD WRITE             OUTPUT
; 3 - TOP DRIVE, BOTTOM HEAD ERASE             OUTPUT
; 2 - TOP DRIVE, BOTTOM HEAD WRITE             OUTPUT
; 1 - TOP DRIVE, TOP HEAD ERASE                OUTPUT
; 0 - TOP DRIVE, TOP HEAD WRITE                OUTPUT
;
; BIT EQUATES FOR PORTB2:
;
BDBHER = BIT7          ;BOTTOM DRIVE, BOTTOM HEAD ERASE
BDBHWR = BIT6          ;BOTTOM DRV, BOTTOM HEAD WRITE
BDTHER = BIT5          ;BOT DRV, TOP HEAD ERASE
BDTHWR = BIT4          ;BOT DRV, TOP HEAD WRITE
TDBHER = BIT3          ;TOP DRV, BOTTOM HEAD ERASE
TDBHWR = BIT2          ;TOP DRV, BOTTOM HEAD WRITE
TDTHER = BIT1          ;TOP DRV, TOP HEAD ERASE
TDTHWR = BIT0          ;TOP DRV, TOP HEAD WRITE
;
PBDDR2:  .=.+1          ;DATA DIRECTION CONTROL, PORT B, PIA #2
PIA2EC:  .=.+1
TIMR2R = ^H0295
TIMR22 = ^H029F
        . = ^H0380      ;START OF PIA #1 ADDR SPACE
;
PORTA1:  .=.+1          ;PORT A, PIA #1
;
; BIT DEFINITIONS FOR PORT A, PIA #1:
;
; 7 - SYNCH MATCH          INPUT
; 6 - CLK RESET            OUTPUT
; 5 - CTS BAR RESET        OUTPUT
; 4 - WRITE PROTECT, TOP DRIVE INPUT
; 3 - MASTER RESET SSSDA  OUTPUT
; 2 - N/A
; 1 - MOTOR ON, DRIVE SELECT, TOP DRIVE OUTPUT
; 0 - WRITE PROTECT, BOTTOM DRIVE INPUT
;
; BIT EQUATES FOR PORTA1:
;
SYNCMT = BIT7          ;SYNCH MATCH
CLKRST = BIT6          ;CLK RESET
CTSRST = BIT5          ;CTS BAR RESET
WPTOP  = BIT4          ;WRITE PROTECT, TOP DRV
MRSSDA = BIT3          ;MASTER RESET SSSDA
DUM9   = BIT2          ;DUMMY BIT USED IN PROGRAM
SELTOP = BIT1          ;MOTOR ON, DRV SELECT TOP DRV
WPBOT  = BIT0          ;WRITE PROTECT, BOTTOM DRV
;
PADDR1:  .=.+1          ;DATA DIRECTION CONTROL, PORT A, PIA #1
PORTB1:  .=.+1          ;PORT B, PIA #1
;
; BIT DEFINITIONS FOR PORT B, PIA #1:
;
; 7 - SERIAL DATA INPUT    INPUT
; 6 - COMMAND FRAME          INPUT (1=RECEIVING CMD FRAME)
; 5 - TOP DRIVE STEPPER MOTOR PHASE 4 OUTPUT
; 4 - TOP DRIVE STEPPER MOTOR PHASE 3 OUTPUT
; 3 - TOP DRIVE STEPPER MOTOR PHASE 2 OUTPUT

```

```

; 2 - TOP DRIVE STEPPER MOTOR PHASE 1          OUTPUT
; 1 - READY/VCC                                INPUT
; 0 - SERIAL DATA OUTPUT                      OUTPUT
;
;
; BIT EQUATES FOR PORTB1:
;
SERIN      =      BIT7          ;SERIAL DATA INPUT
CMDFRM    =      BIT6          ;COMMAND FRAME
SPTOP4    =      BIT5          ;TOP DRV, STEPPER 4
SPTOP3    =      BIT4          ;TOP DRV, STEPPER 3
SPTOP2    =      BIT3          ;TOP DRV, STEPPER 2
SPTOP1    =      BIT2          ;TOP DRV, STEPPER 1
RDY       =      BIT1          ;READY/VCC
SEROUT    =      BIT0          ;SERIAL DATA OUTPUT
;
PBDDR1:   .=.+1                ;DATA DIRECTION CONTROL FOR PORT B, PIA #1
PIA1EC:   .=.+1
TIMR1R   =      ^H0395
TIMR3    =      ^H039E
TIMR2    =      ^H039F
;
; PAGE 8
;
; THE FOLLOWING ARE ADDRESSES INTO A JUMP TABLE FOR 38 KBAUD SERIAL
; AND DOUBLE HEADED DRIVES ROUTINES. THE TABLE WILL ULTIMATELY
; RESIDE AT 800 HEX.
;
XTND0    =      ^H0800
XTND1    =      ^H0803
XTND2    =      ^H0806
XTND3    =      ^H0809
XTND4    =      ^H080C
XTND5    =      ^H080F          ;FORMAT VERIFY FOR 38KBAUD, DOUBLE HEAD, OR 96 TPI
XTND6    =      ^H0812          ;SECTOR # VALIDITY TEST, > 720
XTND7    =      ^H0815
XTND8    =      ^H0818          ;DATA FIELD INPUT, 38K BAUD
XTND9    =      ^H081B
XTND10   =      ^H081E          ;SIDE 2 CLRS?
XTND11   =      ^H0821
XTND12   =      ^H0824
XTND13   =      ^H0827
XTND14   =      ^H082A
;
; MISCELLANEOUS EQUATES
;
TIMOUT    =      224            ;WORST CASE TIME OUT VALUE TO SHIP TO COLLEEN
MAXTRK   =      39             ;MAXIMUM TRACK #, THIS TIME AROUND
ACK       =      'A            ;ACKNOWLEDGE CODE
NAK       =      'N            ;NEGATIVE ACKNOWLEDGE
CMPLT    =      'C            ;COMMAND COMPLETE CODE
ERROR    =      'E            ;COMMAND EXECUTION ERROR
;
; COMMANDS SUPPORTED BY CONTROLLER
;
DOWNL    =      ^H020          ;DOWN LOAD
FORMT    =      ^H021          ;FORMAT DISK
WRINOV   =      ^H050          ;WRITE SECTOR WITHOUT VERIFY
RDSPIN   =      ^H051          ;CONTINUOUSLY READ A SECTOR (FOR TESTING ALIGNMENT)

```

```

READSC = ^H052 ;READ A SECTOR
STATCD = ^H053 ;STATUS COMMAND
READAD = ^H054 ;READ FROM ADDRESS
MTRDLY = ^H055 ;MOTOR ON DELAY 30 SEC
VERIFY = ^H056 ;VERIFY A SECTOR AGAINST INPUT DATA
WRIVER = ^H057 ;WRITE A SECTOR WITH VERIFY
;
; DATA PATTERNS USED FOR FORMATTING:
;
DAM = ^H0FB ;DATA ADDRESS MARK
;
; START OF ROM BASED PROGRAM
;
. = ^H00800
;
;
; *****
; POWER-ON INTIALIZATION
; *****
;
INIT: CLD ;BINARY ARITHMETIC
LDA #BDBHRD+BDTHRD+TDBHRD+TDTHRD
LDX #255
TXS ;STACK POINTER -> $1FF
STA PORTA2
STX PORTB2 ;INIT TO READ, ALL HEADS
STX PADDR2 ;ALL PORTA2 BITS ARE OUTPUT
STX PBDDR2 ;AND PORTB2 TOO
LDA #0
STA PORTA1 ;TURN OFF TOP DRV SPM
STA PORTA3 ;TURN OFF BOTTOM DRV TOO
STA PORTB3 ;CLEAR BOTTOM DRV STEPPER BITS
LDA #WLTOP+RLTOP+SELBOT
STA PADDR3 ;BITS SET FOR OUTPUT
LDX #CLKRST+CTSRST+MRSSDA+SELTOP
STX PADDR1 ;BITS SET FOR OUTPUT
LDX #SPTOP1+SPTOP2+SPTOP3+SPTOP4
STX PBDDR1 ;TOP DRIVE STEPPERS SET AS OUTPUT
INX
STX PBDDR3 ;BOTTOM DRIVE STEPPERS (& BIT0) SET AS OUT
LDX #SPTOP4+SPTOP3
STX PORTB1 ;INIT STP PHASE,BOTH DRVS
STX PORTB3
STX PIA1EC
STX PIA3EC
STX PIA2EC
;
; *****
; INITIALIZATION CONTINUED
; *****
;
INITC: LDX #CLKRST+CTSRST+MRSSDA
STX PORTA1 ;NABL RD CLK SWITCH,TUF LATCH
;AND SSDA CHIP
LDA #RXRS+TXRS+CLEAR
STA SSDA0 ;REST REC(X)MIT. NABL C2 WRITE
LDA #PC1+WS1+WS2+TXSUND+EIE
STA SSDA1 ;INIT C2 REG
LDA #RXRS+TXRS+CLEAR+AC1

```

```

        STA     SSSDA0           ;NABL C3 WRITE
        LDA     #0
        STA     SSSDA1           ;INIT C3 REG
;
; POWER ON RAM TEST
;
RAMTST: LDX     #127             ;128-BYTE BLOCKS TO TEST
RT1:    STA     ZPRL(X)
        STA     ZP80(X)
        STA     RAM3(X)         ;CLR & CHK 3 128 BYTE 6532'S
        CMP     ZPRL(X)
        BNE     RTERR
        CMP     ZP80(X)
        BNE     RTERR
        CMP     RAM3(X)
        BNE     RTERR
        DEX
        BPL     RT1
        BMI     PASRAM
RTERR:  LDA     #RLTOP           ;TOP READ LITE
RTERR1: STA     PORTA3          ;FLASH DRV1 RD LAMP FOR RAM
RTERR2: DEX
        BNE     RTERR2
        INY
        BNE     RTERR2
        EOR     #RLTOP
        BPL     RTERR1
;
; WE PASSED
;
PASRAM: LDA     #NO800+S1NRDY    ;SET INSTAT FOR COLEN ABSNT
        STA     INSTAT          ;AND SPM UNSTABL
        LDA     #BAUD19+TOPDRV+S2NRDY ;SET INMODE FOR 19K I/O ROUTS
        STA     INMODE          ;SPM2 UNSTBL,&DRIVE1 SEL
        LDA     #MAXTRK
        STA     PRTRK2          ;ASSUME DRIVE2 ON TRK 39
        STA     PRSTRK          ;ASSUME DRV1 ON TRK 39
        LDA     #4
        STA     SPMDLC          ;SET FOR 4 DELAYS OF 125MS EACH
                                ;FOR SPM STABL
        LDA     #TIMOUT&255
        STA     WCTLSB
        STA     WCTLB2
        LDA     #TIMOUT/256
        STA     WCTMSB          ;SET WORS CASE TIMOUT STAT BYTES
        STA     WCTMB2          ;FOR BOTH DRIVE 1 AND 2
        STA     DEVSTA          ;CLR ALL CONTROLR STAT DRV1
        STA     DEVST2          ;DITTO FOR DRV2
        STA     SPMSDC          ;CLR SHUTDN TIME DRV1
        STA     SPSDC2          ;DITTO FOR DRV2
;
; COPY ID FIELD INTO RAM
;
        LDX     #10             ;11-BYTE ID FIELD
SCTSET: LDA     IDFLD(X)
        STA     SCTTBL(X)
        DEX
        BPL     SCTSET          ;PRLMNARY SETUP OF RD/WRTSET
                                ;SYNC DATA

```

```

    INX                ;X=0
    LDY                #BAUD38+SEC256
    LDA                PORTB3
    AND                #XTPROM          ;CHECK EXTRA PROM FLAG
    BNE                INIT50
    JSR                XTND14          ;GO TO XTRA ROM FOR MOR DH/96
                                ;TPI AND 38K TSTING
INIT50: STY                STFLGS          ;SET DRV1 STAT TO REFLCT DH/96
                                ;TPI AND 38K CAPABILITIES
    STY                STFLG2          ;DITTO FOR DRV2
    STX                DH80T          ;SET FOR SGL HEAD, 40 TRACK
    JSR                CLRST          ;CLEAR ERROR FLAGS ON TOP DRV
    JSR                CLRST2         ;BOTTOM TOO
;
;
; *****
; WARM START RE-ENTRY POINT
; *****
;
;
INIT2:  LDX                #255          ;REDO STACK POINTER
        TXS                ;TO TOP OF STACK
        LDA                #^CWPS0
        JSR                CLRST1       ;CLEAR TOP DRV WP STATUS
        LDA                #^CWPS0
        JSR                CLRS12      ;BOTTOM DRV TOO
        LDX                #WPS0       ;GET WRITE PROTECT STATUS BIT IN X
        LDA                PORTA1      ;CHECK HARDWARE FOR STATUS
        TAY                ;SAVE IN Y FOR BOTTOM DRV TEST
        AND                #WPTOP     ;CHECK TOP DRV WRITE PROTECT
        BEQ                INIT29      ;IF NOT PROTECTED, GOTO BOTTOM DRV CHECK
        TXA                ;ELSE, GET WP FLAG IN ACCUM
        JSR                ORSTF       ;SET IT IN TOP DRV STATUS FLAGS
INIT29: TYA                ;GET PORTA1 IN ACCUM AGAIN
        AND                #WPBOT     ;CHECK BOTTOM DRV WRITE PROTECT
        BEQ                INIT21      ;IF NOT PROTECTED, GET ON WITH INIT
        TXA                ;ELSE, GET WP FLAG IN ACCUM
        JSR                ORSTF2     ;SET IT IN BOTTOM DRV STATUS FLAGS
;
;
;
INIT21: LDX                #BDBHRD+BDTHRD+TDBHRD+TDTHRD
        STX                PORTA2      ;TURN OFF ALL LTS AND RD NABLS
        LDA                PORTA3
        AND                #^C<WLTOP+RLTOP>
        STA                PORTA3
        BIT                DH80T
        BPL                INIT22
        JSR                XTND10      ;GO TO XTRA ROM FOR SIDE2 CLRS
INIT22: LDA                #TOPDRV
        JSR                ORINMD     ;SET FOR DRV1 SRVICE
        LDA                #MAXTRK
        JSR                CSTEP      ;CALC STPS NEEDED FOR TRK 39 DRV
        STY                STPNBR     ;1 AND SAVE TMP
        LDA                #^CTOPDRV
        JSR                ANDNMD     ;SET FOR DRIV2 SRVICE
        LDA                #MAXTRK
        JSR                CSTEP      ;CALC STPS NEEDED FOR TRK 39,DRV

```

```

        STY      STPNB2          ;2 AND SAVE TMP
        LDA      #SHTDWN
        JSR      ORINST          ;SET SHUTDN BIT IN INTRNL STAT
        LDA      #^CRDAFWR
        JSR      ANINS
INIT3:   LDX      SPMSDC
        BEQ      INIT4          ;BRNCH IF DRV1 TIMED OUT
        LDA      #255
        STA      TIMR23         ;SET 125MS DELA
        DEX
        STX      SPMSDC         ;UPDATE AND SAV DRV1 TIMR SCRT
INIT4:   LDX      SPMSDC2
        BEQ      INIT5          ;BRNCH IF DRV2 TIMED OUT
        LDA      #255
        STA      TIMR23         ;SET 125MS DELA
        DEX
        STX      SPMSDC2        ;UPDATE AND SAV DRV2 TIMR SCRT
INIT5:   LDX      SPMSDC2
        BNE      INIT6          ;BRNCH IF DRV1 NOT TIMED OUT
        LDX      #^H080
        LDY      STPNBR         ;GET SET TO STP DRV1
        BEQ      INIT11         ;BRNCH IF NO STPS NEEDED ON DRV1
        JSR      STEP1          ;STEP DRV1 AND SAV STP CNTR
        BEQ      INIT11         ;BRNCH IF LAST STP JUST DONE
INIT51:  STY      STPNBR
INIT6:   LDX      SPMSDC2
        BNE      INIT7          ;BRNCH IF DRV2 NOT TIMED OUT
        LDX      #^H080
        LDY      STPNB2         ;GET SET TO STP DRV2
        BEQ      INIT12         ;BRNCH IF NO STPS NEEDED ON DRV2
        JSR      STP2          ;STP DRV2 AND SAV STP CNTR
        BEQ      INIT12         ;BRNCH IF LAST STP JUST DONE
INIT61:  STY      STPNB2
INIT7:   LDX      SPMSDC2
        BNE      INIT8          ;BRNCH TO POLL TIMR IF DRV1 NOT
                                     ;TIMED OUT
        LDX      SPMSDC2
        BNE      INIT8          ;RNCH TO POLL TIMR IF DRV2 NOT
                                     ;TIMED OUT
        LDX      STPNBR
        BNE      INIT9          ;BRNCH TO POLL 5MS TIMR IF DRV1
                                     ;NOT ON TRK 39
        LDX      STPNB2
        BNE      INIT9          ;BRNCH TO POLL 5MS TIMR IF DRV2
                                     ;NOT ON TRK 39
INIT71:  LDA      #^CSHTDWN
        JSR      ANINS          ;CLR SHUTDN BIT IN INTRNL STAT
        JMP      MMON          ;LEAVE THIS ROUT--POLL FOR CMND
;
INIT11:  LDA      #CLKRST+CTSRST+MRSSDA
        STA      PORTA1         ;TURN OFF SPM1
        LDA      #S2NRDY
        JSR      ORINST          ;SET SPM1 UNSTBL BIT
        JMP      INIT51
;
INIT12:  LDA      #0
        STA      PORTA3         ;TURN OFF SPM2
        LDA      #S2NRDY
        JSR      ORINMD         ;SET SPM2 UNSTBL BIT

```

```

        JMP      INIT61
;
INIT8:  BIT      TIMR3R
        BMI      INIT10      ;BRNCH IF 125MS UP
        JSR      MMON
INIT9:  BIT      TIMR1R
        BMI      INIT5       ;BRNCH IF 5MS UP
        JSR      MMON
        JMP      INIT9
;
INIT10: BIT      TIMR1R
        BMI      INIT13      ;BRNCH IF 5 MS UP
        JSR      MMON
        JMP      INIT10
INIT13: JMP      INIT3
;
;
; *****
;   MAIN MONITOR
; *****
;
;           MONITORS THE READY PORT FOR
; COLLEEN PRESENCE. MONITORS THE CMND
; FRAME,BRINGS IN 5 BYTE COMMAND, CHE
; CKS FOR VALID CHECKSUM, CHECKS IF
; CMND ADDRESS IS FOR FLOPPY BEFORE
; GOING TO CMND INTERPRET ROUT
;
;
MMON:   LDA      PORTB1
        LSR
        LSR
        BCS      NCOLL      ;TST RDY PORT &BRNCH IF COLEN AB
;SENT
        LDA      INSTAT
        LSR
        BCS      JCOLL      ;TST IF COLEN PRSNT 100MS BRNCH
;IF NOT
        BIT      PORTB1
        BVC      NOCF       ;TST FOR CMND FRAME BRNCH IF NO
        BIT      INSTAT
        BVC      EDGE       ;BRNCH IF MISSED EDGE BY 400US
        LDA      #^H0A4
        JSR      TIMER
        BIT      PORTB1      ;WAIT 100MS,THEN TEST IF
        BVS      EDGE       ;CMND FRAME STILL PRESNT
NOCF:   LDA      INSTAT
        AND      #BUSY+SHTDWN
        BNE      MMX        ;TST TO JSR BACK TO BUSY OR
        JMP      MMON       ;SPIN IN MAIN MONITOR ROUT
MMX:    RTS
JCOLL:  LDA      #^H0A4
        JSR      TIMER
        LDA      #^CNO800
        JSR      ANINS      ;SET STAT TO REFLCT COLEN NOT
;PRESNT 100MS
        JMP      MMON
NCOLL:  LDA      #NO800
        JSR      ORINST     ;SET INTRNL STAT NOT RDY BIT

```

```

        JMP      NOCF
EDGE:   LDA      #RECCMD
        JSR      ORINST          ;SET INTRNL STAT "NEW CMND BIT"
        LDX      #0
        LDA      #5
        STA      INSCRT         ;SET UP TO BRING IN CMND
        JSR      CINPURT
;
; *****
;   MAIN MONITOR (CONTINUED)
; *****
;
MMON1:  LDX      #0
        DEC      INSCRT         ;SET UP TO GEN CS ON 4 BYTE
        JSR      CDCKSM         ;CMND
        BEQ      CCMD           ;BRNCH IF CS OK
        LDA      #CRECR         ;SET COMMAND RECEIVE ERROR
        JSR      ORINST         ;IN INSTAT
CCMD:   LDA      PORTA3         ;CHECK DRIVE ADDR DECODE
        LSR      #2             ;CHECK DRV ADDR #2
        BCC      ADRDC3         ;IF SET, GOTO ADRDC3
        LDX      #'1
        ASL      #2             ;RESTORE PORTA3
        BMI      ADRDC5         ;IF DRV ADDR #1 IS SET, BRANCH
        LDX      #'5
ADRDC5: CPX      UNITNB
        BEQ      ADRDC7         ;BRNCH IF DRIV ADRESED IS ODD
        INX
        CPX      UNITNB
        BEQ      ADRDC6         ;BRNCH IF EVEN DRIV ADRESED IS U
CCF:   BIT      PORTB1
        BVS      CCF           ;SPIN TIL COMND FRAM GONE
        LDA      #^C<RECCMD+CRECR> ;CLR REC NEW CMND AND
        JSR      ANINS          ;CS ERR BITS
        JMP      MMON          ;RETURN TO MON SER BUS
;
;
ADRDC6: LDA      #^C<TOPDRV>
        JSR      ANDNMD         ;UPDATE INMODE FOR DRIV 2 SEL
;
ADRDC8: JSR      CMDI           ;GO COMMAND INTERP---FOR US
        JMP      MMON
;
ADRDC7: LDA      #TOPDRV
        JSR      ORINMD         ;UPDATE INMODE FOR DRIV 1 SEL
        BNE      ADRDC8
ADRDC3: LDX      #'3
        ASL      #2
        BMI      ADRDC5
        LDX      #'7
        BNE      ADRDC5
;
; *****
;   COMMAND INTERPRETER
; *****
;
;
CMDI:  LDA      INSTAT         ;CHECK RECEIVE ERROR FLAG
        AND      #CRECR         ;MASK OFF OTHERS

```



```

BNE      CMD2          ;IF CHECKSUM ERROR, GOTO INVALID CMD NAK
LDA      CTLCMD        ;ELSE, FETCH CMD CODE FROM CMD FRAME
LDX      #TBBT-TABB    ;MAXIMUM INDEX INTO LEGAL CMD CODE TABLE
XTAB:    CMP      TABB(X) ;COMPARE REC'D CMD W/LEGAL CMD LIST
BEQ      GTCMD        ;GOT A MATCH, GO EXECUTE
DEX      ;ELSE, DECREMENT INDEX
BPL      XTAB         ;LOOP IF ^AANY LEGAL ENTRIES LEFT
BMI      CMD2         ;NO, UNCONDITIONAL SEND INVALID CMD FRAME
;
GTCMD:   LDA      TABL(X) ;YES, FETCH LO ADDR OF ROUT
STA      CTABL        ;SAVE IN INDIRECT JUMP VECTOR
LDA      TABH(X)     ;FETCH HI ADDR TOO
STA      CTABH       ;SAVE IN HI ADDR OF VECTOR
JMP      @CTABL      ;JUMP TO SELECTED COMMAND ROUTINE
;
CMD2:    LDA      #INVCF0 ;SET INVALID CMD FRAME FLAG
JSR      OR0STX      ;IN PRESENT DRIVE STATUS BYTE
JSR      MFUTST      ;SET SPM ON BIT IN STATUS TOO
;
CMD3:    BIT      PORTB1
BVS      CMD3        ;WAIT TIL CMD FRAME GONE
LDA      #NAK        ;LOAD NAK CODE
JSR      SAA2        ;SEND IT TO COLLEEN
LDA      #^CCRECR    ;CLEAR CMD RECEIVE ERROR FLAG
JSR      ANINS       ; IN INTERNAL STATUS REGISTER
RTS      ;RETURN TO MAIN MONITOR
;
; *****
;   COMMAND INTERPRETER LOOK-UP TABLES
; *****
;
;
TABB:    .BYTE     DOWNL, FORMT, WRINOV, RDSPIN, READSC, STATCD, READAD, MTRDLY, VERIFY, WRIVER
;
TBBT     =.-1
;
TABL:    .BYTE     DOWNLD&255 ;DOWN LOAD
          .BYTE     FRMT&255  ;FORMAT
          .BYTE     WRITE1&255 ;WRITE, NO VERIFY
          .BYTE     RDSPN&255  ;READ SPIN
          .BYTE     READ&255   ;READ SECTOR
          .BYTE     STCMD&255  ;STATUS COMMAND
          .BYTE     RDADR&255  ;READ FROM ADDRESS
          .BYTE     MOTRON&255 ;MOTOR ON DELAY
          .BYTE     WRITE&255  ;VERIFY ENTERS AT WRITE
          .BYTE     WRITE&255  ;WRITE/VERIFY
;
TABH:    .BYTE     DOWNLD/256
          .BYTE     FRMT/256
          .BYTE     WRITE1/256
          .BYTE     RDSPN/256
          .BYTE     READ/256
          .BYTE     STCMD/256
          .BYTE     RDADR/256
          .BYTE     MOTRON/256
          .BYTE     WRITE/256
          .BYTE     WRITE/256
;
; *****

```

```

;      WRITE ROUTINE
;      *****
;
WRITE:  LDA      #RDAFWR
        JSR      ORINST          ;SET INSTAT FOR READ AFTR WRITE
;
;  WRITE, NO VERIFY ENTRY POINT:
;
WRITE1: JSR      TSN              ;GO INSUR RECVD SECTR NUMBR VAL
        BCC      WR10            ;BRNCH IF SCTOR VALID
        JMP      WR91
WR10:   LDA      #^H04E
        STA      GAPBYT
        JSR      SETSUP
        BEQ      WR92            ;BRNCH IF NO STEPS NEEDE
        JSR      STEP
WR92:   STY      STPNBR          ;SAV TEMP NUMBR STEPS NEEDED
        STX      STPDIR          ;SAV STEP DIRECT TEMP
        JSR      INPUT           ;GO INPUT DATA FIELD,COMP,CALC-
        ;-ULATE CS AND COMPAR CS'S
        BEQ      WR20            ;BRNCH IF NO CS ERR
WR5:    LDA      #MTR0+INVDF0
WR51:   JSR      OR0STX          ;SET SPM AND DATA FRAME ERROR FLGS
        JMP      INIT2
;
WR20:   JSR      SAA1            ;GO ACK DATA FIELD
        LDX      STPDIR
        LDY      STPNBR
        JSR      STPTST          ;GO EXECUT STEPS IF NEEDED
        JSR      SPMDY           ;INSURE SPM STABL
        LDA      CTLCMD
        CMP      #VERIFY         ;CHECK FOR VERIFY ONLY
        BNE      WR201           ;BRNCH IF HERE FOR WRITE
        JSR      CRCSUP
        JMP      WR30            ;GO DO VERFY
WR201:  JSR      WPTST
        BEQ      WR2             ;BRNCH IF NOT WP
        JMP      WR7
WR2:    JSR      CRCSUP
WRITE2: LDY      #0              ;Y SET TO INDICAT PASS
        JSR      SYNC            ;THRU SYNC LOOP FOR IDAM
        BNE      WR21            ;BRNCH IF HAV SYNC
        LDA      #NOID
        JSR      ORST1           ;UPDATE DEVICE STAT FOR NO ID
        JMP      WR8
WR21:   JSR      FWSL
        LDX      #12
        JSR      FSL
        LDX      EGON
        STX      PORTB2
        LDX      #5
        JSR      FSL
        JSR      FWSL1           ;GO XMIT REMAINDR OF DATA FIELD
        ;GAP AND ID
        SEC                      ;CARRY MUST REFLECT LST BIT XMIT
        ;ED.
        LDX      #0              ;CLR BUFR POINTR
        JSR      WSL             ;GO XMIT 128 BYTE BUFR THRU
        ;CLK/DTA LOOKUP TBL

```

```

;
; *****
; WRITE CRC'S
; *****
;
; (CALLED ONLY ONCE BY WRITE ROUTINE, COULD BE "IN-LINE")
;
; WCRC WRITES 2 CRC'S FOR DATA FIELD
; DURING WRITE ROUT. NOT USED B
; BY FORMAT ROUT
;
WCRC: LDX #128-3 ;INDEX FOR 2 BYTES
WCRC1: BIT SSSA0
      BVC WCRC1 ;POLL SSSA FOR XMIT REG AVAIL
      LDA BAKER-<128-3>(X)
      ROR
      LSR
      LSR
      LSR
      TAY
      LDA CDLT(Y)
      STA SSSA1
      LDA BAKER-<128-3>(X)
      AND #^H01F
      TAY
      LSR
      LDA CDLT(Y)
      STA SSSA1
      INX
      BPL WCRC1
;
;
WSPIN: LDA SSSA0
      AND #TUF
      BEQ WSPIN ;BRNCH TIL TUF SET IN STATREG
      LDA #RDA+TDRA+CTS+PE+IRQ
      STA SSSA0 ;REST XMIT
      LDA PORTA1
      AND #^CCTSRST
      STA PORTA1 ;REST CTS FF
      ORA #CCTSRST
      STA PORTA1 ;SET CTS FF
      LDA WGOFF
      STA PORTB2
      JSR EGOFF
      LDA INSTAT
      AND #RDAFWR
      BEQ WR36 ;BRNCH IF NO READ CHK REQUIRED
WR30: LDA #READSC
      STA CTLCMD
      JSR RDWRLT
      JSR WFRD ;GO RD SCTR TO VRIFY
      BCS WRITE9 ;BRNCH IF BAD WRITE VRIFY
WR36: LDA #MTR0
      JSR OR0STX ;UPDTE STAT FOR GOOD WRITE
      JSR RDSUB
WR360: LDA #CMPLT
WR6: JSR SAA2 ;OUTPUT TO COLLEEN
      JMP INIT2 ;GOTO SHUTDOWN

```

```

;
WR91:  LDA    #^CRDAFWR
        JSR    ANINS
        JMP    CMD2

;
WR7:    LDA    #MTR0+WPS0+RWE0 ;SET STATUS FOR WP
WR70:   JSR    OR0STX
        LDA    #ERROR          ;SET A FOR ERR
        JMP    WR6

;
WR8:    LDX    PACNTR
        BEQ    WRITE9          ;BRNCH TO GIV UP WRITE ATMPT
        JSR    ERRTRY          ;REPOSITON HEAD
        JMP    WRITE2          ;REATMPT WRITE
WRITE9: LDA    #MTR0+RWE0
        JMP    WR70            ;GO SET STATUS FOR ERR
WFRD:   LDA    #1
        STA    PACNTR          ;SET FOR 2 READ ATTEMPTS
WR3:    LDY    #0
        JSR    SYNC            ;LOOK FOR IDAM
        BNE    WR300           ;BRNCH IF HAV SYNC
        LDA    #NOID
        JMP    WR44
WR300:  JSR    RDSUB
        LDA    #3
        LDY    #1
        JSR    SYNC01          ;LOOK FOR DAM
        BEQ    WR34            ;BRNCH IF NO SYNC
        LDX    #0
WR31:   BIT    SSSA0
        BPL    WR31
        LDA    SSSA1
        CMP    #^H0FB
        BNE    WR34
        LDA    INMODE
        LSR
        BCS    WR40            ;BRNCH IF HERE ON FORMAT CHK
        LDA    SSSA1
        CMP    ZPRL(X)
        BNE    WR34            ;BRNCH IF ONE DATA FIELD WORD
                                   ;ERR
WR32:   INX
WR33:   BIT    SSSA0
        BPL    WR33            ;BRNCH TIL RDA
        LDA    SSSA1
        CMP    ZPRL(X)
        BNE    WR34
        INX
        BEQ    WR37            ;BRNCH IF ALL DATA FIELD OK
        LDA    SSSA1
        CMP    ZPRL(X)
        BEQ    WR32
WR34:   DEC    PACNTR
        BPL    WR3              ;BRNCH TO RETRY READ CHK
        SEC
        RTS

;
WR37:   LDA    SSSA1
        CMP    BAKER

```

```

WR35:   BNE      WR43           ;BRNCH IF CRC1 NO MATCH
        BIT      SSDA0
        BPL      WR35
        LDA      SSDA1
        CMP      ABLE
        BNE      WR43
        JSR      RDSUB
        CLC
        RTS           ;CRY=NO ERR ON RETRN

;
WR40:   LDA      SSDA1
        CMP      #^H0B6
        BNE      WR34           ;BRNCH IF ERR
WR41:   INX
WR42:   BIT      SSDA0
        BPL      WR42
        LDA      SSDA1
        CMP      #^H0B6
        BNE      WR34           ;BRNCH IF ERR
        INX
        BEQ      WR37           ;BRNCH IF 256 BYTES CHKD
        LDA      SSDA1
        CMP      #^H0B6
        BEQ      WR41           ;BRNCH IF NO ERR
        JMP      WR34
WR43:   LDA      #DUM10
WR44:   JSR      ORST1           ;SET BIT IN DEVICE STATUS
        JMP      WR34

;
; *****
; READ ROUTINE
; *****
;
READ:   JSR      TSN             ;TEST FOR VALID SCT NUMBR
        BCC      READ01
        JMP      CMD2
READ01: JSR      SETSUP
        JSR      STPTST
        JSR      SPMDY           ;GO INSUR SPM STABL
READ0:  LDY      #0             ;Y SET TO INDICAT PASS THRU
        JSR      SYNC           ;SYNC LOOP FOR IDAM
        BNE      READ00
        LDA      #NOID          ;BIT IS IN DEVSTA
        JMP      RD35           ;UPDATE CNTRLR STAT FOR ERR
READ00: JSR      RDSUB
        LDY      #1             ;Y SET TO INDICATE PASS THRU
        LDA      #3
        JSR      SYNC01         ;SYNC LOOP FOR DAM
        BEQ      READ6         ;BRNCH IF NO SYNC
        LDX      #0             ;CLR BUFR POINTR
READ1:  BIT      SSDA0           ;POLL STATREG FOR RDA
        BPL      READ1
        LDA      SSDA1         ;SRVICE SSSA
        CMP      #DAM           ;FIRST WORD MUST BE DAM
        BNE      READ6         ;GO TO ERR IF NOT DAM
        LDA      SSDA1
        STA      ZPRL(X)       ;STOR IN XFER BUFR
READ2:  INX
READ3:  BIT      SSDA0           ;POLL STATREG FOR RDA

```

```

        BPL      READ3
        LDA      SSSA1
        STA      ZPRL(X)          ;STOR IN XFER BUFR
        INX
        BEQ      READ4          ;BRNCH IF BUFERS FUL
        LDA      SSSA1
        STA      ZPRL(X)
        JMP      READ2
READ4:  LDA      SSSA1
        STA      RDCRC1          ;SAVE CRC1
READ5:  BIT      SSSA0
        BPL      READ5
        LDA      SSSA1
        STA      RDCRC2          ;SAVE CRC2
        JSR      CRCSUP
        LDA      RDCRC1
        CMP      BAKER
        BNE      RD34
        LDA      RDCRC2
        CMP      ABLE
        BNE      RD34
RD31:   LDA      #MTR0
        JSR      OR0STX
RD32:   LDA      #CMPLT
RD33:   STA      TMPOUT
        JSR      SETOP
        JSR      BFROUT
        JMP      INIT2
RD34:   LDA      #DUM10
RD35:   JSR      ORST1          ;UPDATE CONTRLR STAT FOR ERR
READ6:  LDX      PACNTR
        BEQ      READ7          ;BRNCH TO GIV UP READ ATTEMPT
        JSR      ERRTRY          ;GO REPOSITION HEAD
        JMP      READ0          ;TRY TO READ AGAIN
READ7:  LDA      #MTR0+RWE0
        JSR      OR0STX          ;UPDATE STATUS TO REFLECT ERR
        LDA      #ERROR
        JMP      RD33          ;OUTPUT "E" TO COLN
RDSUB:  LDA      #^C<NOID+DUM10>
        JSR      ANDST1
        RTS

;
; *****
;  DOWNLOAD
; *****
;
;           INPUTS 256 BYTES FROM COLN,
;  XECUTES PROGRAM THEN,STRNG AT ZP00.
;
DOWNLD: JSR      SAA
        LDX      #1
        STX      AUX2
        JSR      INPUT
        BNE      DNLD6          ;BRNCH IF DATAFRAM ERR
        PHA
        JSR      ZPRL          ;GO XECUTE PROGRM
        BIT      INMODE
        BVC      DNLD2          ;BRNCH TO SET DRV2 STAT BYTES
        STA      STFLGS          ;SAV A(Y)(X),PS IN DRV1 STAT BYTES

```

```

        STX      STFLGS+1
        STY      STFLGS+2
        PLA
        STA      STFLGS+3
        JMP      DNLD4
DNLD2:  STA      STFLG2      ;SAV A@X(Y),PS IN DRV2 STAT BYTEZ
        STX      STFLG2+1
        STY      STFLG2+2
        PLA
        STA      STFLG2+3
DNLD4:  LDA      #CMPLT
        JSR      SAA2
        JMP      INIT21
DNLD6:  JSR      MFUTST      ;UPDATuE SPM FLG BIT
        LDA      #2
        JMP      WR51      ;GO UPDATE STAT FOR DATA FRAM ER
;
; *****
; READ-SPIN (ALIGNMENT TEST)
; *****
;
; RDSPIN---TURNS ON SPM,RDLITE,&HD
; NABL. SPINS TIL NXT CMD RECVD.
; USED FOR TSTIN ONLY
;
RDSPN:  LDX      #READSC
        STX      CTLCMD
        JSR      SETSUP
        JSR      STPTST
        JMP      INIT71
;
; *****
; READ FROM ADDRESS (128-BYTE)
; *****
;
RDADR:  CLC
        LDA      AUX1
        TAY
        ORA      AUX2
        BEQ      EXTST
        LDA      AUX2
        JMP      RDADR1
EXTST:  LDY      #XSTAT&255
        LDA      PORTB3
        AND      #XTPROM
        BNE      RDADR0
        SEC      ;SET CRY IF XTNDed ROM INSTALD
RDADR0: LDA      #XSTAT/256
RDADR1: STY      ZPNTL
        STA      ZPNTH      ;SET 0-PGE POINTR
        LDY      #0
RDADRL: LDA      @ZPNTL(Y)
        STA      ZPRL(Y)      ;MOV 128 BYTES TO BUFERS
        INY
        BPL      RDADRL
        BCC      RDADR3
        JSR      XTND1      ;GO TO UPDATE IF X ROM INSTLD
RDADR3: LDX      #1

```

```

        STX     AUX1
        DEX
        STX     AUX2
        JMP     RD32
;
; *****
;   STATUS CMND
; *****
;
;           TESTS TO DETERMINE WHICH
; DRIVE TO XMIT STATUS FOR, SENDS
; 4 BYTE STATUS AND CHKSM TO COL-
; -LEEN WHICH RELECTS STATE OF
; DRIVE DURING LAST COMMAND. UP-
; DATES DRIVE STATUS UPON COMPLET-
; -ION OF STATUS CMND
;
STCMD:  JSR     SAA             ;ACK CMND
        LDX     #4
        BIT     INMODE
        BVC     SCMD20        ;BRNCH IF CMND FOR DRV2
SCMD10: LDA     WCTMSB-4(X)    ;MOVE 4 BYTE STAT FOR DRV1
        STA     STPNT(X)      ;INTO MORE CONVENIENT LOC.
        DEX
        BNE     SCMD10
STCMD1: LDY     #4
        STY     INSCRT
        JSR     CDCKSM        ;GEN CS ON 4 BYTE STAT
        STA     RDCDSM
        LDA     #CMPLT
        STA     TMPOUT        ;SET TO XMIT "C"
        JSR     SETOP        ;OPEN SERIAL OUTPUT PORT
        LDX     #128-5        ;5 BYTES TO SEND
        JSR     OUTPUT
STCMD2: LDY     #15
STCMD3: DEY
        BNE     STCMD3        ;SET DELA OF
        LDY     UNITNB-<128-5>(X) ;74 MACH CYC
        STY     TMPOUT
        JSR     OUTPUT        ;OUTPUT 1 BYTE
        INX
        BPL     STCMD2
        JSR     RSTOUT        ;CLOSE OUTPUT PORT
        BIT     INMODE
        BVC     SCMD21        ;BRNCH IF CMND FOR DRV2
        JSR     CLRST        ;CLR ALL STAT BUT WP
        JSR     SPMFU        ;UPDATE DRV1 SPM FLG
        JMP     INIT2
SCMD20: LDA     WCTMB2-4(X)    ;MOVE 4 BYTE STAT FOR DRV2
        STA     STPNT(X)      ;INTO MORE CONVEINIENIENT LOC
        DEX
        BNE     SCMD20
        BEQ     STCMD1        ;FWR BUYTES THAN JMP
SCMD21: JSR     CLRST2        ;CLR ALL BUT BASIC STATUS FOR DR
        ;VE 2
        JSR     SPM2FU        ;UPDATE DRV2 SPM FLG
        JMP     INIT2
;
;

```



```

; *****
;   FORMAT ROUTINE
; *****
;
FRMT:  JSR      SAA
       JSR      SPMON          ;TURN ON SPM IF NOT ALREADY
       JSR      RDWRLT
       JSR      SPMDY          ;INSURE MOTOR UP TO SPEED
       JSR      WPTST
       BEQ      CONFRM          ;TEST DISC NOT WP,IF NOT,BRNCH.
       LDA      #MTR0+WPS0+RWE0
       JSR      OR0STX
       LDA      #^H0FF
       STA      ZPRL          ;DS TRMS TO FRNT
       STA      ZPRL+1
       JMP      READ7
CONFRM: LDY      #MAXTRK+5
       JSR      REHME
       LDX      #^H0FE
       STX      IDAM          ;INIT ID FIELD LIST
       LDX      #^H04E
       STX      BYTE4E
       LDX      #0
       STX      BYTE00
       STX      TRKNO
       STX      SCTTBL+4
       INX
       STX      BYTE01
TRKLP:  JSR      STSUP2          ;GO TO TST FOR <20 TRK SWITCH
       LDX      #0
       STX      SCTPT
       STX      CRCLP
       INX
       STX      SCTTBL+6
FCRCLP: LDX      #^H0F8
       JSR      CRCGEN          ;GO GEN 2 BYTE CRC ON 5 BYTE ID
                                   ;FIELD FOR THIS TRK,SCT NUMBER.
       LDX      CRCLP
       LDA      BAKER
       STA      FRMCRC(X)
       INX
       LDA      ABLE
       STA      FRMCRC(X)
       INX
       STX      CRCLP
       LDY      SCTPT
       LDX      SCTR(Y)
       BIT      STFLGS
       BPL      FORM0
       JSR      XTND4          ;GO TO XTRA ROM IF 38K CAPABL
FORM0:  STX      SCTTBL+6
       INC      SCTPT
       CPY      #18
       BNE      FCRCLP          ;BRNCH IF MOR CRCS TO DO
       LDX      #0
       STX      SCTPT
       LDA      FRMCRC(X)
       STA      CRC1          ;STORE FIRST CRC IN ID LIST
       INX

```

```

STX      SCTNO      ;REST SCT NUMBER TO 01
LDA      FRMCRC(X)  ;STOR SECOND CRC IN ID LIST
STA      CRC2
STX      CRCLP      ;SAVE CRC LIST POINTER
FWT:    LDA      TIMR1R
BPL      FWT        ;WAIT FOR LAST STEP TO SETTLE.
JSR      STST1      ;WAIT 10MS FOR LAST STP SETL
JSR      FWSL
LDX      EGON
STX      PORTB2     ;TURN ON ERASE GATE
LDX      #255       ;SET X AS 255 BYTE COUNTER.
JSR      FSL        ;GO XMIT 255 BYTES OF "4E"
DEX      ;SETS X TO FF AS X EQUALS 00 ON
          ;RETURN FROM PREVIOUS JSR.
          ;GO XMIT 255 BYTES OF "4E"
JSR      FSL
DEX
SCTLP:  JSR      FSL      ;XMIT 255 BYTES OF "4E"
LDA      #^H0AA
TAY
LDX      #8
JSR      FSL        ;XMIT 8 BYTES OF "00". THIS IS S
          ;ECOND HALF OF GAP3.
LDA      #^H044
LDY      #^H089
LDX      #3
JSR      FSL        ;XMIT 3 BYTES OF "A1". THIS IS
          ;ID AM SYNC LEADER.
LDA      #^H055
LDY      #^H054
LDX      #1
JSR      FSL        ;XMIT 1 BYTE "FE". THIS IS ID AM
LDX      #^H0F9
CLC
JSR      WSL        ;XMIT 7 BYTES OF ID FIELD THRU
          ;ID FIELD TABLE.
LDA      #^H092
FONE:  BIT      SSDA0
BVC      FONE
STA      SSDA1
LDY      #^H054
STY      SSDA1      ;XMIT 1 BYTE "4G" FAST-LATE
LDX      #18
JSR      FSL        ;XMIT 20 MORE BYTES OF "4E". THI
          ;S IS FIRST HALF GAP 2.
INC      CRCLP
LDX      CRCLP
LDA      FRMCRC(X)  ;MOVE CRC1 FOR NXT SCTOR INTO
STA      CRC1      ;ID FIELD LIST
LDA      #^H0AA
TAY
LDX      #12
JSR      FSL        ;XMIT 12 BYTES OF "00". THIS IS
          ;LAST HALF OF GAP 2.
INC      CRCLP
LDX      CRCLP
LDA      FRMCRC(X)  ;MOVE NXT CRC2 INTO ID
STA      CRC2      ;FIELD LIST
LDA      #^H044
LDY      #^H089

```

```

LDX      #3
JSR      FSL          ;XMIT 3 BYTES "A1" THIS IS DAM
                        ;SYNC LEADER

LDA      #^H055
LDY      #^H045
INX

JSR      FSL          ;XMIT 1 BYTE "FB" THIS IS DAM.
LDA      #^H045
LDY      #^H014
INX

JSR      FSL          ;XMIT 1 BYTE "00"
LDA      #^H045
LDX      #^H0FF      ;XMIT 255 BYTES "00". THIS,PLUS
JSR      FSL          ;PREVIOS XMIT=128 BYTE DATA FLD
LDA      #^H045
LDY      #^H029
INX

                        ;XMIT 1 BYTE "48" THIS IS
JSR      FSL          ;CRC1 FOR DATA FIELD OF ALL "00"
LDA      #^H014
LDY      #^H0AA
INX

                        ;XMIT 1 BYTE "29" THIS IS
JSR      FSL          ;CRC2 FOR DATA FIELD ALL "00"
LDA      #^H092
LDY      #^H054
INX

                        ;XMIT 1 BYTE "4E" THIS IS
JSR      FSL          ;1 BYTE OF GAP 3 FIRST HALF
LDA      #^H092
LDX      #23          ;XMIT 23 BYTES "4E" THIS
JSR      FSL          ;COMPLETES FIRST HALF GAP3
LDY      SCTPT
LDX      SCTRS(Y)
STX      SCTNO        ;MOV NXT SCTR IN
INC      SCTPT
CPY      #17          ;CHK FOR 18 SCTRS FRMTD
BNE      JSCTLP       ;IF NOT, GO DO ANOTHER
LDA      #RXRS+TXRS+CLEAR+AC1+AC2
STA      SSSDA0       ;REST XMIT
LDA      WGOFF
STA      PORTB2       ;TURN OFF WG
INC      TRKNO
INC      SCTTBL+4
JSR      EGOFF
LDX      #MAXTRK+1
LDA      DH80T
BEQ      FONE1
JSR      XTND3         ;GO TEST FOR MAX # OF TRKS TO FORMAT
FONE1:  CPX      TRKNO   ;CHK FOR ALL TRKS FORMATED
BEQ      FTST
LDX      #^H080       ;X MUST BE NEG VALUE BEFOR ENTRY
                        ;TO STEP ROUT TO STEP IN

JSR      STEP
JMP      TRKLP        ;GO FORMAT ANOTHR TRK
JSCTLP: JMP      SCTLP ;MUST GO TO SCTLP FROM HERE BRNCH
                        ;OUT OF RANGE

```

```

;
; *****

```

```

;      FORMAT TEST
;      *****
;
;      TESTS ALL SCTRS
;              AFTR ACTUAL
;      FORMAT AND COMPILES LISTING OF
;      BAD SCTRS TO SEND TO COLEN AT
;      END. (63 BAD SCTRS MAX)
;
FTST:  LDA      #READSC
      STA      CTLCMD
      JSR      RDWRLT
      LDX      #0
      STX      BFRPNT
      LDY      #^H080
FT1:   LDA      SCTRS(X)      ;ALTRNATLY MOVE SCTR #'S
      STA      ZPRL(Y)      ;FROM ROM INTO BUFERS
      INY      ;AT HEX LOC.0080 TO 0091
      INX      ;INCLUSIV
      INX
      CPX      #18
      BNE      FT2
      LDX      #1
      STX      ZP91
FT2:   CPX      #17
      BNE      FT1
      DEC      SCTTBL+4
FT21:  LDX      #128
FT3:   LDA      ZPRL(X)
      INX
      STX      FSCTBP      ;SAV SCTR PNTR
      STA      SCTTBL+6
      JSR      STSUP1      ;DO CRC ON SCTR ID FIELD
      LDA      #^H060
      STA      ABLE
      LDA      #^H0B1
      STA      BAKER
      LDA      #DUM11
      JSR      ORINMD      ;SET LSB INMODE FOR WFRD
      ;ROUT TSTING
      JSR      WFRD      ;READ 1 SCTR
      DEC      INMODE
      BCS      FT5      ;BRNCH IF SCTR READ ERR
FT4:   LDX      FSCTBP
      CPX      #128+18
      BNE      FT3      ;BRNCH IF MOR SCTRS TO READ
      LDY      SCTTBL+4
      BNE      FT40      ;BRANCH IF ALL TRKS NOT READ
      LDA      DH80T
      BEQ      FT6      ;BRANCH IF NOT DH OR 96 TPI DRVS
      JSR      XTND5
      JMP      FT6
;
FT40:  DEY
      STY      SCTTBL+4
      LDX      #0
      LDY      #1
      JSR      STPTST      ;STP TO NXT TRK
      JMP      FT21      ;CONT TST

```

```

;
FT5:   LDA     #FMTER
       JSR     ORINST           ;SET FORMAT ERR BIT INSTAT
;
;     TRACK/SECTOR CONVERSION TO
;     2 BYTE SECTOR NUMBER
;
SCTBU: LDY     #0
       STY     MSBSB
       LDX     BFRPNT
       LDA     SCTTBL+4       ;GET TRK NMBR
       ASL
       STA     TMPSB         ;SAVE TIMES TWO
       ASL
       ROL     MSBSB
       ASL
       ROL     MSBSB
       ASL
       ROL     MSBSB
       ADC     TMPSB
       TAY
       LDA     #0
       ADC     MSBSB
       STA     MSBSB
       TYA
       CLC
       ADC     SCTTBL+6
       STA     ZPRL(X)
       LDA     #0
       ADC     MSBSB
       INX
       STA     ZPRL(X)
       LDA     DH80T
       BEQ     FT51
       JSR     XTND11         ;GOTO INCREASE SEC # FOR SIDE 2
; ;
FT51:  INX
       STX     BFRPNT
       CPX     #128-2
       BNE     FT4           ;BRNCH IF BAD SCTR BUFR NOT FULL
FT6:   LDA     #^H0FF
       LDX     BFRPNT
       STA     ZPRL(X)
       INX
       STA     ZPRL(X)
       STA     SCTTBL+4
;
       LDA     #FORMT
       STA     CTLCMD
;
       LDA     INSTAT
       AND     #FMTER
       BNE     FT7           ;BRNCH IF ANY BAD SCTRS
       JMP     RD31
FT7:   LDA     #^CFMTER
       JSR     ANINS         ;CLR FRMTERR BIT INSTAT
       JMP     READ7
;
;

```

```

; *****
;   FORMAT SERVICE LOOP
; *****
;
FSL:   BIT       SSDA0           ;POLL STATRG FOR XMITRA
       BVC       FSL
       STA       SSDA1           ;DBL LOAD TO SSDA--16 BITS
       STY       SSDA1           ;0F CLK/DTA=8 BIT WORD TO DISC
       DEX
       BNE       FSL             ;CONT SRVCE TIL X=00
       RTS

;
; *****
;   MOTOR DELAY 30 SECS
; *****
;
;   TURN ON SPM AND SETS FOR 30
;   SEC DELAY BEFORE SHUTOFF
;
MOTRON: JSR      SAA
        JSR      SPMON
        LDX      #255
        BIT      INMODE
        BVC      MON2
        STX      SPMSDC           ;SET DRV1 MOTR FOR 30 SEC DLA
        BVS      MON4
MON2:   STX      SPSDC2           ;SET DRV2 MOTR FOR 30 SEC DLA
MON4:   JMP      STCMD

;
; *****
;   STEP
; *****
;
;           TESTS TO DETERMINE WHICH
;   DRIVE TO PERFORM 1 STEP ON. STEPS,
;   SETS UP 5MS TIMER BUT DOES NOT WAIT
;
STEP:   BIT      INMODE
        BVC      STP2             ;BRNCH IF HERE TO STEP DRIV2
;
;   REHOME HEAD
;   STEP ROUTINE
;
STEP1:  LDA      #^H0A0
        STA      TIMR3           ;SET FOR 5.25MS STEP WIDTH
        LDA      PORTB1
        AND      #SPTOP1+SPTOP2+SPTOP3+SPTOP4
        INX
        BMI      STPIN
        DEC      PRSTRK           ;STP TO OUTR TRK
        LSR
        STA      STPSCR
        AND      #BIT1
        BEQ      STEP2
        LDA      #SPTOP4
        ORA      STPSCR
        BPL      STEP3
STPIN:  INC      PRSTRK           ;STP TO INNR TRK

```

```

ASL
STA     STPSCR
AND     #BIT6
BEQ     STEP2
LDA     #SPTOP1
ORA     STPSCR
BPL     STEP3
STEP2:  LDA     STPSCR
STEP3:  STA     PORTB1           ;THIS IS PHYSICAL STEP
        DEY     ;Y=NUMBER OF STEPS TO GO
        RTS

;
;
STP2:   LDA     #^H0A0
        STA     TIMR3           ;SET FOR 5.25MS DELA
        LDA     PORTB3
        AND     #SPBOT1+SPBOT2+SPBOT3+SPBOT4
        INX
        BMI     STPIN2
        DEC     PRTRK2         ;STP TO OUTR TRK
        LSR
        STA     STPSCR
        AND     #BIT1
        BEQ     STEP22
        LDA     #SPBOT4
        ORA     STPSCR
        BPL     STEP32
STPIN2: INC     PRTRK2         ;STP TP INER TRK
        ASL
        STA     STPSCR
        AND     #BIT6
        BEQ     STEP22
        LDA     #SPBOT1
        ORA     STPSCR
        BPL     STEP32
STEP22: LDA     STPSCR
STEP32: STA     PORTB3         ;THIS IS PHYSICAL STEP
        DEY     ;Y=NUMBER OF STEPS TO GO
        RTS

;
; *****
;     SECTOR TEST
; *****
;
;     TESTS IF PRESENT CMND
;     SECTOR NUMR IS 1,2,OR3. SETS
;     CARRY IF SO FOR TESTING ON
;     RETURN TO BUFRS OUT OR INPUT.
;
SCTTST: LDX     #0
        CPX     SCTTBL+4
        CLC
        BNE     SCTST1         ;BRNCH IF PRESENT TRK# NOT 00
        LDA     SCTTBL+5         ;BRANCH IF ON SIDE 2
        BNE     SCTST1
        LDA     #3
        CMP     SCTTBL+6         ;TST IF PRSNT SCT 1,2,OR3
SCTST1: RTS
;

```

```

;
; *****
;   MOTOR STATUS UPDATE
; *****
;
;           TSTS TO DETRMIN WHICH DRV
;   TO UPDATE MOTR STAT ON. THEN,
;   UPDATES ONLY THAT STAT BIT
;
MFUTST: BIT      INMODE
        BVC      SPM2FU          ;BRNCH IF HERE TO UPDATE DRV2
                                   ;MOTR STAT
;
;
SPMFU:  LDA      PORTA1
        LSR
        LSR
        LDA      #MTR0
        BCS      MFU1
        LDA      #0
MFU1:   JMP      ORSTF
;
;
SPM2FU: LDA      PORTA3          ;TST IF DRV 2 MOTER
        LSR              ;ON
        LSR
        LDA      #MTR0
        BCS      MFU12        ;BRNCH IF DRV2 MOTR ON
        LDA      #0
MFU12:  JMP      ORSTF2        ;GO UPDATE DRV2 STATUS
;
;
; *****
;   ERROR HANDLING
; *****
;
ERRTRY  TSTS TO DETRMIN WHICH DRV
;           TO REPOSITION HEAD ON FOR
;   RD OR WRIT RETRY. EITHR REHOMES AND
;   REPOSITIONS OR STPS ONCE AND BACK.
;
ERRTRY: DEX
        STX      PACNTR
        TYA
        BNE      ERR2          ;BRNCH IF DAM NOT FOUND
        CPX      #2
        BNE      ERR2          ;BRNCH IF ONLY ONE PASS MADE
        LDY      #MAXTRK+5
        JSR      REHME          ;REHOME HEAD
        BMI      ERR6          ;ALWAYS BRNCH
ERR2:   CPX      #1
        BNE      ERR8          ;BRNCH IF THIS NOT THIRD ATMPPT
        LDY      STPDIR
        BMI      ERR4          ;BRNCH TO STP IN AND BACK
        DEC      TRKNBR
        JSR      ERRSUB        ;STP OUT
        INC      TRKNBR
        BPL      ERR6          ;ALWAYS BRNCH TO STP BACK
ERR4:   INC      TRKNBR

```



```

        JSR      ERRSUB      ;STEP IN
        DEC      TRKNBR
ERR6:   JSR      ERRSUB      ;STP BACK
ERR8:   RTS

ERRSUB: LDA      TRKNBR
        JSR      CSTEP
        JSR      STPTST
        RTS

;
;
; *****
;   READ/WRITE LITE SELECT
; *****
;
; RDWRLT      TSTS TO DETRMIN WHICH
; READ HD TO NABL AND TRNS ON EITHR RD
; OR WRIT LITE FOR PROPER DRV
;
RDWRLT: LDA      INMODE
        ASL
        BPL      RWLT4      ;BRNCH IF CMND FOR DRV2
        LDX      #BDBHRD+BDTHRD+TDBHRD
        ASL
        BMI      RWLT1      ;BRNCH IF CMND FOR SIDE2
        LDX      #BDBHRD+BDTHRD+TDTHRD
RWLT1:  STX      PORTA2
        LDA      PORTA3
        AND      #^C<WLTOP+RLTOP>
        STA      PORTA3
        LDX      CTLCMD
        CPX      #READSC
        BEQ      RWLT2
        ORA      #WLTOP
RWLT3:  STA      PORTA3
        RTS

;
RWLT2:  ORA      #RLTOP
        BNE      RWLT3      ;UNCONDITIONAL BRANCH (SAVE A BYTE)
;
RWLT4:  LDX      #RLBOT+BDBHRD+TDBHRD+TDTHRD
        ASL
        BMI      RWLT5      ;BRNCH IF CMND FOR SIDE2
        LDX      #RLBOT+BDTHRD+TDBHRD+TDTHRD
RWLT5:  TXA
        LDX      CTLCMD
        CPX      #READSC
        BEQ      RWLT6
        CLC
        ADC      #RLBOT      ;ADC?
RWLT6:  STA      PORTA2
        RTS

;
; *****
;   COMMANDS INPUT
; *****
;
;           INPUTS FROM COLLEEN TO
;   COMMANDS BUFRS. COMPLIMENTS DATA

```

```

; RUNS AT 19.2K EXCEPT -- WHEN
; INPUTTING DATA FIELD CS FOR INPUT
; ROUTINE. MAY RUN AT 19.2K OR 38.4K
; THEN
;
CHKIN: LDA     PORTB1      ;NTR HERE FOR DATA FIELD CHKSM
      BPL     CHKIN
      BMI     CIN0        ;HAV CS STRT BIT,BRNCH
CINPUT: BIT     PORTB1
      BVC     CIN5        ;BRNCH IF CMD FRAM GONE--ERR
      BPL     CINPUT      ;BRNCH IF NO STRT BIT YET
CIN0:  LDA     INMODE
      AND     #BAUD19+CMDFIN
      CMP     #CMDFIN
      BEQ     CIN1        ;BRNCH IF RUNNING AT 38K SPEED
      LDY     #11
CNDLA1: DEY
      BNE     CNDLA1      ;DELA OF 54 M CYC
CIN2:  LDY     #128-8      ;SET BIT CNTR
CNDLA2: TYA
      LDY     #14
CNDLA3: DEY
      BNE     CNDLA3      ;DELA OF 69 M CYC AT 19K, 9
                        ;M CYC AT 38K FOR BIT SPACE,
                        ;44 M CYC AT 38K FOR STRT BIT
      TAY
      LDA     PORTB1      ;RETREIV 1 BIT
      ROL
                        ;SHIFT OUT CRY
      ROR     UNITNB(X)   ;ASSEMBL BIT TO PRSNT WORD
      INY
      BPL     CIN3        ;BRNCH TIL 8 BITS DONE
      LDA     UNITNB(X)
      EOR     #255
      STA     UNITNB(X)   ;COMPLMNT LAST BYTE IN
      INX
      CPX     INSCRT
      BNE     CIN4        ;BRNCH IF MORE BYTES TO BRING IN
      RTS
;
CIN1:  LDY     #128-8
      NOP
      LDY     #9
      BNE     CNDLA3      ;ALWAYS BRNCH
CIN3:  LDA     INMODE
      AND     #BAUD19+CMDFIN
      CMP     #CMDFIN
      BNE     CNDLA2      ;BRNCH IF RUNNING AT 19K SPEED
      TYA
      LDY     #2
      NOP
      NOP
      NOP
      BNE     CNDLA3      ;ALWAYS BRNCH
CIN4:  LDY     #13
CNDLA4: DEY
      BNE     CNDLA4      ;DELA 64 M CYC
      BEQ     CINPUT      ;ALWAYS BRNCH
CIN5:  JMP     CMD2        ;HAVE CMND FRAM ERR
;

```

```

; *****
;   CRC SETUP FOR DATA FIELD
; *****
;
CRCSUP: LDA      #CRCTST
        JSR      ORINMD          ;SET LSB IN INMODE FOR CRC TSTIN
        LDX      #0
        LDY      #^H0E2
        STY      BAKER
        LDY      #^H095
        STY      ABLE
        JSR      CRCETR
        LDA      #^CCRCTST
        JSR      ANDNMD          ;CLR LSB IN INMODE
        RTS

;
; *****
;   ERASE GATE OFF
; *****
;
;
;
EGOFF   TIMESOUT 832US BEFOR
;        TURNING OFF PROPR EG
;
EGOFF:  LDA      #^H01A
        STA      TIMR3          ;SET TIMR FOR 832US
EGOFF1: LDA      TIMR1R
        BPL      EGOFF1        ;WAIT FOR TIMR INTRUPuT
        LDA      #BDBHER+BDBHWR+BDTHER+BDTHWR+TDBHER+TDBHWR+TDTHER+TDTHWR
        STA      PORTB2        ;TURN OFF EG
        RTS

;
; *****
;   SETSUP
; *****
;
;        PRFORMS COMMON HOUSE
;        KEEPING FOR READ AND WRITE ROUTS.
;
SETSUP: JSR      SAA            ;ACK CMND
        JSR      SPMON         ;TURN ON SPM IF NOT ALREADY
        JSR      RDWRLT
        LDA      #3            ;ERROR RETRY COUNT
        STA      PACNTR
        JSR      SCTBD         ;CONVRT RECEIVD SEC TO TRK/SEC
STSUP0: LDA      SCTNBR
        STA      SCTTBL+6      ;INIT SYNCTBL
        LDA      TRKNBR
        STA      SCTTBL+4
STSUP1: LDX      #^H0F8
        JSR      CRCGEN
        LDX      BAKER
        STX      SCTTBL+8      ;CRC1
        STA      SCTTBL+9      ;CRC2
STSUP2: LDA      PORTB3
        LDX      SCTTBL+4      ;GET TRK NMBR GOING TO
        CPX      #20
        BIT      DH80T
        BVC      STUP20

```

```

        JSR      XTND13          ;GO XTRZ ROM IF 96TPI
STUP20: BCC      STSUP3
        AND      #^CAMPADJ          ;CLR RD/WRITE >20 TRK GATE
        BNE      STSUP4
STSUP3: ORA      #AMPADJ          ;SET RD/WRITE <21 TRK GATE
STSUP4: STA      PORTB3
        LDA      TRKNBR
        JSR      CSTEP
        RTS

;
; *****
;   BUFFERS OUT
; *****
;
;           OUTPUTS ASCII "C"/"E",
;           256 BYTE BUFFER AND CHECK
;   SUM. COMPUTES CHECKSUM
;
BFR0UT: JSR      SCTTST          ;TST IF SCTR 1,2,OR 3
        STX      CSSCRT          ;CLR FOR FIRST CS ADD
        BCC      BOUT01          ;BRNCH IF TST RESLT NOT 1,2,OR 3
        LDA      #FORMAT
        CMP      CTLCMD
        BEQ      BOUT01          ;BRNCH IF PRSNT CMND A FORMAT
BOUT0:  LDA      ZPRL(X)          ;MOVE LOWR HALF
        STA      ZP80(X)          ;OF BFRS TO
        INX
        BPL      BOUT0
BOUT01: CLC
        BIT      INMODE
        BMI      BOUT02          ;BRNCH IF 19.2K
        JMP      XTND9
BOUT02: PHP
        JSR      OUTPUT          ;OUTPUT "C" OR "E"
        LDY      #8
BOUT1:  DEY
        BNE      BOUT1          ;DELA OF 39 MACH CYC
BOUT2:  LDY      #8
BOUT3:  DEY
        BNE      BOUT3          ;DELA OF 39 MACH CYC
        LDY      ZPRL(X)
        STY      TMP0UT          ;MOVE 1 BYTE INTO OUTPUT SCRATCH
        JSR      OUTPUT          ;OUTPUT 1 BYTE
        PLP
        JSR      DFCKSM          ;RETRIV CRY
        PHP
        JSR      DFCKSM          ;ADD LAST BYTE OUT TO CS
        INX
        PHP
        INX
        BNE      BOUT2          ;SAV CRY FROM LAST CS ADD
        LDA      CSSCRT          ;BRNCH IF 256 BYTES NOT DONE
        PLP
        ADC      #0              ;ADD LAST CS ADD CRY TO CS
        STA      TMP0UT
        LDY      #7
BOUT4:  DEY
        BNE      BOUT4          ;DELA OF 34 MACH CYC
        JSR      OUTPUT          ;OUTPUT CHEKSUM
RSTOUT: LDX      PBDDR1
        DEX
        STX      PBDDR1          ;REST OUTPUT PORT

```

```

        RTS
;
; *****
;   WRITE SERVICE LOOP
; *****
;
WSL:   BIT        SSDA0           ;POLL STATREG FOR XMITRA
        BVC       WSL
        LDA       ZPRL(X)        ;GET DATA WORD FROM BUFR
        ROR                          ;SHIFT OUT LOWR 4 BITS,BRING
        LSR                          ;IN LSB OF LAST BYTE XMITED
        LSR                          ;THRU CARRY
        LSR                          ;NOW HAV 5 BIT POINTER IN LSB'S
        TAY                          ;Y WILL BE POINTR
        LDA       CDLT(Y)        ;GET 1 BYTE C/D FROM TABLE
        STA       SSDA1
        LDA       ZPRL(X)        ;GET DATA WORD FROM BUFR AGAIN
        AND       #^H01F        ;MASK OFF TOP 3 BITS
        TAY                          ;Y WILL BE POINTER
        LSR                          ;SAV LSB IN CRY FOR NXT WORD
                                ;TO XMIT
        LDA       CDLT(Y)        ;GET 1 BYTE C/D FROM TABLE
        STA       SSDA1         ;XMIT
        INX                          ;CONT SERVICE ROUTINE TIL X=00
        BNE       WSL
        RTS
;
; *****
;   REHOME
; *****
;
;           TESTS TO DETERMINE TO
;   REHOME DRIVE 1 OR 2. REHOMES,
;   WAITING 5MS AFTR EACH STEP.
;   WAITS ONLY 5MS AFTR LST STP
;   SETS RESPECTIV DRIV SCRATH LOC
;   TO INDICAT TRK 00
;
REHME:  LDX       #SPTOP1+SPTOP4
        BIT       DH80T
        BVC       RHM10
        JSR       XTND7         ;GO TO XTRA ROM IF 96TPI
RHM10:  BIT       INMODE        ;TST TO REHME DRV1 OR 2
        BVS       REHME1        ;BRNCH TO REHME DRV1
        STX       PORTB3        ;INIT STP PHASE TO REHME DRIV 2
RHM12:  JSR       STEP
        BEQ       RHM42
RHM22:  LDA       TIMR1R
        BPL       RHM22         ;SPIN TIL 5MS UP
        JMP       RHM12
RHM32:  JMP       REHME1        ;GO REHOME DRIV 1
RHM42:  STY       PRTRK2        ;SET PRTRK2 TO INDICATE DRIV 2
                                ;ON TRK 00
RHM52:  LDA       #128
        STA       TIMR3         ;SET TIMR FOR 50MS
RHM62:  LDA       TIMR1R
        BPL       RHM62
        RTS
REHME1: STX       PORTB1        ;INIT STP PHASE

```

```

REHM4: JSR     STEP
        BEQ     REHM3
REHM2: LDA     TIMR1R
        BPL     REHM2           ;WAIT 5MS FOR STP SETL
        BMI     REHM4
REHM3: STY     PRSTRK           ;SET FOR TRK 00
        BEQ     RHM52
;
; *****
;   OUTPUT
; *****
;
;           OUTPUTS 1 BYTE FROM TMPOUT
;   TO COLLEEN AT 19.2K
;
OUTPUT: LDA     PORTB1
        AND     #^CSEROUT
        STA     PORTB1           ;START BIT OUT
        LDY     ZPRL             ;DELA 3 MACH CYC
        LDY     #128-8           ;SET BIT COUNTR
OUT1:   NOP
        NOP
        NOP                       ;DELA 6 MACH CYC
        STY     YSAVE            ;SAV BIT COUNTR
        LDY     #14
OUT2:   DEY
        BNE     OUT2             ;DELA 69 MACH CYC
        LDY     YSAVE
        LSR
        ROR     TMPOUT
        ROL
        STA     PORTB1
        INY
        BPL     OUT1             ;BRNCH TIL 8 BITS OUT
        LSR
        SEC
        ROL
        LDY     #17
OUT3:   DEY
        BNE     OUT3             ;DELA 84 MACH CYC
        NOP
        NOP                       ;DELA 4 MACH CYC
        STA     PORTB1           ;STOP BIT OUT
        RTS
;
; *****
;   SPINDLE MOTOR ON
; *****
;
;           TESTS TO DETERMINE WHICH
;   DRIVE TO TURN ON SPM. TURNS ON SPM I
;   F NOT ALREADY ON AND SETS TO INDICAT
;   500MS DELA REQUIRED
;SETS FOR 6 SEC DELA ON SHUTDOWN
;
SPMON:  BIT     INMODE
        BVC     SPO02           ;BRNCH IF HERE TO TURN ON SPM 2
;
;

```

```

; TEST IF SPM IS ON. IF NOT,TURN ON AND
; SET TO INDICATE 500MS DELAY REQ.
;
SPMON1: LDA PORTA1
        LSR
        LSR ;TEST IF SPM IS ON. IF OM,LEAVE
        BCS SPO1 ;ROUTINE
        LDA #CLKRST+CTSRST+MRSSDA+SELTOP
        STA PORTA1 ;TURN ON SPM
        LDA #255
        STA TIMR2 ;SET FOR 125MS DELAY
        LDA #4
        STA SPMDLC ;SET FOR 4 DELAYS
SPO1: LDA #24
      STA SPMSDC ;SET FOR 3 SEC DLA ON SHTDWN
      RTS
;
;
SPO02: LDA PORTA3
        LSR
        LSR
        BCS SPO12 ;BRNCH IF SPM2 ALREADY ON
        LDA #SELBOT
        STA PORTA3 ;TURN ON SPM2
        LDA #255
        STA TIMR23 ;SET FOR 125MS DELA
        LDA #4
        STA SPMDC2 ;SET FOR 4 DELAS
SPO12: LDA #24
      STA SPSPDC2 ;SET FOR 3 SEC DELA ON SHTDN
      RTS
;
; *****
; SPINDLE MOTOR READY
; *****
;
; TESTS TO DETERMINE WHICH
; DRIVE TO PERFORM DELAY ON. WAITS
; FOR 500 MS IF NEEDED. UPDATES STUTUS
; TO REFLECT THIS SPM ON AND STABL
;
SPMDY: BIT INMODE
        BVC SPDY12 ;BRNCH IF HERE FOR SPM2 DELA
;
;
;
; TEST IF SPM SPEED STABLE/WAIT IF NOT.
;
SPMDY1: LDA INSTAT
        LSR
        LSR
        LSR
        BCC SPMDY4 ;IF SPM STABL,LEAVE ROUT
SPMDY2: LDA TIMR1R
        BPL SPMDY2 ;WAIT 125MS
        DEC SPMDLC
        BEQ SPMDY3 ;CONT DELAY LOOPS TIL SPMDLC=00
        LDA #255

```

```

        STA     TIMR2           ;RESET TIMER FOR 125MS DELAY
        JMP     SPMDY2
SPMDY3: LDA     #^CS1NRDY
        JSR     ANINS           ;CLEAR MOTOR NOT READY BIT IN IN
SPMDY4: LDA     #NODISK         ;TERNL STAT REG
        JSR     OR0ST          ;SET SPM ON FLG IN XFER STAT
        RTS                     ;REG
;
SPDY12: LDA     INMODE
        AND     #S2NRDY
        BEQ     SPDY42         ;BRNCH IF SPM2 STABL
SPDY22: LDA     TIMR3R
        BPL     SPDY22         ;WAIT 125MS
        DEC     SPMDC2
        BEQ     SPDY32         ;CONT DELA LOOPS TIL SPMDC2=00
        LDA     #255
        STA     TIMR23         ;RESET TIMR FOR 125MS DELA
        JMP     SPDY22
SPDY32: LDA     #^CS2NRDY
        JSR     ANDNMD         ;CLR MOTOR UNSTABL BIT IN INMODE
SPDY42: RTS
;
; *****
;     CRC GENERATION LOOP
; *****
;
CRCGEN: LDA     #^H0FF
        STA     ABLE
        STA     BAKER
CRCETR: LDA     INMODE
        LSR
        .BYTE   ^H0BD         ;ABSOLUTE ADDR. SO INDEX DOESN'T WRAP INTO
                               ;0 PAGE
        .WORD   ZPCRC         ;"LDA ZPCRC(X) ABSOLUTE"
        BCC     CRCT1
        LDA     ZPRL(X)
CRCT1:  EOR     BAKER
        STA     SCRT1
        LSR
        LSR
        LSR
        LSR
        EOR     BAKER
        TAY
        LDA     INMODE
        LSR
        TYA
        BCC     CRCT3
        EOR     ZPRL(X)
CRCT2:  AND     #^H00F
        STA     SCRT2
        LDA     SCRT1
        AND     #^H0F0
        EOR     SCRT2
        STA     SCRT2
        LSR
        LSR
        LSR
        EOR     ABLE

```



```

    STA     ABLE
    LDA     SCRT2
    ASL
    ASL
    ASL
    ASL
    STA     SCRT1
    EOR     ABLE
    STA     BAKER
    LDA     SCRT1
    ASL
    EOR     SCRT2
    STA     ABLE
    INX
    BNE     CRCETR
    RTS
CRCT3:  .BYTE   ^H05D           ;FORCE ABS. ADDR. SO INDEX DOESN'T WRAP
        .WORD   ZPCRC           ;"EOR ZPCRC(X) ABSOLUTE"
        JMP    CRCT2
;
; *****
;     SYNCRONIZATION LOOP
; *****
;
SYNC00: JMP     SYNC7
SYNC:   LDA     #6               ;SET TIMR LOOP CNTR FOR 5 LOOPS
        STA     SPMDC2
SYNC0:  DEC     SPMDC2
        BEQ     SYNC00          ;BRNCH IF 5 TIMR LOOPS DONE
        CPY     #1
        BEQ     SYNC00          ;BRNCH IF HERE FOR DAM AND 1 TIM
                                   ;R LOOP DONE
        LDA     #255            ;SET TIMR FOR 130 MS
SYNC01: STA     TIMR23
NUTS:   LDA     TIMR3R
        BMI     SYNC0           ;BRNCH IF TIMR DONE
        LDA     #RXRS+TXRS+CLEAR
        STA     SSSA0
        LDA     #^H09B
        STA     SSSA1
SYNCCLP: LDA    #RXRS+TXRS+CLEAR+AC2
        STA     SSSA0           ;REST REC(X)MIT. NABL SR WRITE
        LDA     #^H044         ;LOAD SYNC REG WITH HEX 44 (FIRS
        STA     SSSA1         ;T HALF SYNC CHAR CLK/DATA PATRN
        LDA     PORTA1
        AND     #^CCLKRST
        STA     PORTA1        ;REST PA6(PORTA1) TO RESET READ
        ORA     #CLKRST       ;CLOCK FAST.
        STA     PORTA1        ;NABL READ CLOCK SWITCH
        LDA     #TXRS+AC2
        STA     SSSA0         ;NABL REC
SYNC1:  LDA     TIMR3R
        BMI     SYNC0         ;BRNCH IF TIMR DONE
        BIT     PORTA1
        BPL     SYNC1         ;BRNCH IF NO SYNC MATCH FROM
                                   ;SSDA CHIP
        LDA     #^H089        ;HAV FIRST SYNC MATCH. LOAD SYNC
        STA     SSSA1         ;REG WITH SECND HALF CLK/DATA
        LDX     #4           ;PATRN OF SYNC CHAR AND

```

```

SYNC2:  DEX                                ;WAIT APPROX. 13US MORE BEFORE
        BNE      SYNC2                    ;VERIFICATION OF SECOND SM
        BIT      PORTA1                   ;MUST HAV SECOND SM NOW. IF NOT,
        BMI      NUTS                     ;FALSE STRT RESTRT SYNCLOOP
                                                ;P.

        LDA      #TXRS
        STA      SSSA0                    ;NABL C2 WRITE,CONT REC
        LDA      #PC1+PC2+WS1+WS2+TXSUND+EIE
        STA      SSSA1                    ;NOW HAV SYNC. INHIBIT
                                                ;FURTHER SM OCCURENCES.

        CPY      #2
        BEQ      SYNC6
        LDX      #2
        LDA      #3
        STA      TIMR3
SYNC3:  BIT      TIMR1R
        BMI      NUTS
        BIT      SSSA0
        BPL      SYNC3                    ;POLL SSSA STAT REG FOR RDA
SYNC4:  LDA      SSSA1                    ;CMPRE NXT 2
        CMP      #^H0A1                   ;BYTES IN TO "A1"
        BNE      NUTS                     ;NOT EQUAL=FALSE STRT
        DEX
        BNE      SYNC4
        TYA
        BNE      SYNC6
        LDX      #128-8
SYNC5:  BIT      SSSA0
        BPL      SYNC5
        LDA      SSSA1                    ;CMPARE NXT 8 WORDS IN
        CMP      SCTTBL+3-<128-8>(X)      ;WITH THIS TRK,SCT DATA
        BNE      NUTS                     ;IN TBLE. IF ANY ERRS
        INX                                ;HAV FLSE STRT--RESTRT LOOP
        LDA      SSSA1
        CMP      SCTTBL+3-<128-8>(X)
        BEQ      SYNC8
        JMP      NUTS
SYNC8:  INX
        BPL      SYNC5
SYNC6:  LDA      #1
SYNC7:  RTS                                ;Z=0 ON EXIT MEANS HAV SYNC
;
; *****
;      TIMEOUT WG BEFORE STEP
; *****
;
TWGBS:  LDA      #8
        JSR      TIMER                    ;DELAY
        JSR      STEP
        RTS
;
; *****
;      TIMER
; *****
;
TIMER:  STA      TMRSCR
TMR1:  DEY
        BNE      TMR1
        DEC      TMRSCR

```

```

        BNE      TMR1
        RTS
;
; *****
;   STATUS CLEAR AND UPDATE
; *****
;
;   TSTS TO DETRMIN WHICH DRV TO
;   UPDATE STAT ON.  CLEARS ALL BUT BASIC
;   STATUS BITS, THEN UPDATES STAT BY
;   CONTNTS OF ACCUMULATR ON ENTRY TO
;   THIS ROUT
;
OR0STX: BIT      INMODE
        BVC      OR0ST2      ;BRNCH IF HERE TO UPDATE DRV2
;
OR0ST:  PHA
        LDA      #^C<MTR0+RWE0+INVDF0+INVCF0>
        AND      STFLGS
        STA      STFLGS
        PLA
ORSTF:  ORA      STFLGS
        STA      STFLGS
        RTS
;
ORINST: ORA      INSTAT
        STA      INSTAT
        RTS
;
OR0ST2: PHA
        LDA      #^C<MTR0+RWE0+INVDF0+INVCF0>
        AND      STFLG2
        STA      STFLG2
        PLA
ORSTF2: ORA      STFLG2
        STA      STFLG2
        RTS
;
CLRST:  LDA      #^C<MTR0+RWE0+INVDF0+INVCF0>
CLRST1: AND      STFLGS
        STA      STFLGS
        RTS
;
CLRST2: LDA      #^C<MTR0+RWE0+INVDF0+INVCF0>
CLRS12: AND      STFLG2
        STA      STFLG2
        RTS
;
; *****
;   WRITE PROTECT TEST
; *****
;
;   WPTST      TESTS TO DETRMIN WENICH
;   DRV TO PERFORM WP TEST ON.  Z FLG
;   RESET IF DRV WP
;
WPTST:  LDA      PORTA1
        BIT      INMODE
        BVC      WPTST1      ;BRNCH IF HERE TO CHK WP DRV2

```

```

WPTST0: AND      #WPTOP
        RTS
;
WPTST1: AND      #WPBOT
        RTS
;
ANINS:  AND      INSTAT
        STA      INSTAT
        RTS
;
; *****
; UPDATE INMODE BITS
; *****
;
ORINMD: ORA      INMODE
        STA      INMODE
        RTS
;
ANDNMD: AND      INMODE
        STA      INMODE
        RTS
;
; *****
; UPDATE DEVICE STATUS FLAGS
; *****
;
ORST1:  BIT      INMODE
        BVC      ORST10
        ORA      DEVSTA
        STA      DEVSTA
        RTS
;
ORST10: ORA      DEVST2
        STA      DEVST2
        RTS
;
ANDST1: BIT      INMODE
        BVC      ANST10
        AND      DEVSTA
        STA      DEVSTA
        RTS
;
ANST10: AND      DEVST2
        STA      DEVST2
        RTS
;
; *****
; TEST RECEIVED SECTOR NUMBR
; *****
;
TSN:    LDX      AUX2
        LDA      AUX1
        BEQ      TSN2          ;BRNCH IF LS HALF 00
TSN1:   CPX      #3
        BCS      TSN3          ;BRNCH IF MS HALF 03 OR MORE
        CMP      #^H0D1
        BCS      TSN4          ;BRNCH IF LS HALF D1 OR MORE
TSN11:  CLC
        RTS
        ;SET CRY FOR NO ERR

```

```

TSN2:  CPX      #0
        BNE     TSN1          ;BRNCH IF MS HALF NOT 00
TSN3:  SEC
        LDA     DH80T
        AND     #DBLHED+TPI96
        BEQ     TSN31
        JSR     XTND6          ;GO XTRA ROM IF 96TPI 0R DH
TSN31:  RTS
TSN4:  CPX      #2
        BEQ     TSN3          ;BRNCH IF MS HALF 02
        JMP     TSN11

;
;
; *****
;   DATA FIELD CHECKSUM COMPUTE
; *****
;
;           SMALL ROUT USED TO
; GENERAT CS ON DATA BUFRS FOR BOTH
; I/O ROUTS
;
DFCKSM: LDA     CSSCRT          ;GET CURENT CS WORD
        ADC     ZPRL(X)        ;ADD LAST WORD TO CS
        STA     CSSCRT          ;SAV TEMP
        RTS

;
;
; *****
;   COMMANDS CHECKSUM
; *****
;
;           GENERATES CS ON 4 BYTE CMND
; AND COMPARES RESULT AGAINST REC CS
; ON EXIT,Z=1 IF NO ERR
;
CDCKSM: LDA     #0
        CLC
        ;CLR A & CRY FOR FIRST ADD
CS1:   ADC     UNITNB(X)        ;ADD ONE WORD TO CS
        PHP
        ;SAV CRY FOR NXT ADD
        INX
        CPX     INSCRT          ;CHK FOR ALL BYTES ADDED
        BEQ     CS2            ;BRNCH IF ALL ADDED
        PLP
        ;RETREIV CRY
        JMP     CS1            ;GO DO ANOTHR ADD
CS2:   PLP
        ;RETREIV CRY
        ADC     #0              ;ADD LST CRY TO CS
        CMP     UNITNB(X)        ;COMPAR CS TO RECVD CS
        RTS

;
; *****
;   FORMAT/WRITE SETUP
; *****
;
;           PERFORMS BASIC
; SETUP OPRATIOMS FOR BOTH WRITE
; AND FORMAT ROUTS
;
;
FWSL:  LDA     PORTA2

```

```

ORA      #BDBHRD+BDTHRD+TDBHRD+TDTHRD
STA      PORTA2          ;DISABLE ALL READ HEADS
;
WGEGSU: LDA      INMODE
        ASL
        BMI      WGEG2          ;BRNCH IF DRV2 ADDRSD
;
        ASL
        ASL
        LDA      #^H0EF
        LDX      #^H0CF
        LDY      #^H0DF
        BCS      WGEG1
        LDA      #^H0BF
        LDX      #^H03F
        LDY      #^H07F
        BCC      WGEG1
;
WGEG2:  ASL
        ASL          ;LOAD CRY WITH "SIDE" BIT
        LDA      #^H0FE
        LDX      #^H0FC          ;PORTB2 VALUES (IN X, Y)
        LDY      #^H0FD
        BCS      WGEG1          ;BRNCH IF SIDE2 ADRSD
        LDA      #^H0FB
        LDX      #^H0F3
        LDY      #^H0F7
;
WGEG1:  STX      EGON
        STY      WGOFF          ;SAV TMP EG AND WG CODES
;
;
        STA      PORTB2
        LDA      #^H0D3
        STA      SSSA0          ;RST REC AND XMIT NABL
                                ;XMIT FIFO WRITE
        LDA      #^H092          ;PRELOAD XMIT FIFO
        LDY      #^H054          ;CLK/DTA PATRN
        STA      SSSA1          ;FOR 1 BYTE OF
        STY      SSSA1          ;"4E"
        LDX      #^H093          ;HOLD REC,
        STX      SSSA0          ;NABL XMIT,NABL XMIT FIFO WRITE
        RTS
;
; *****
;   FORMAT/WRITE SETUP PART 2
; *****
;
;           XMITSTHE SECOND HALF OF
; THE GAP AND ALL OF THE ID FOR
; THE DATA FIELD IN BOTH THE WRITE
; AND FORMAT ROUTS
;
;
FWSL1:  LDA      #^H0AA
        TAY
        LDX      #^H00C
        JSR      FSL          ;XMIT 12 BYTES "00"
        LDA      #^H044

```

```

LDY    #^H089
LDX    #^H003
JSR    FSL            ;XMIT 3 BYTES MODIFIED "A1"
LDA    #^H055
LDY    #^H045
INX
JSR    FSL            ;XMIT 1 BYTE "FB" THIS IS DAM
RTS

;
; *****
;   DATA FIELD INPUT
; *****
;
;           INPUTS FROM COLLEEN TO
; DATA BUFERS. COMPLIMENTS DATA. GEN-
; -ERATES CS.INPUTS COLLEEN'S CS
; THRU COMMANDS CHECKSUM ROUT AND
; COMPARES IT TO IT'S OWN CS. SETS
; Z FLG IF THEY MATCH.
;           ALSO, TESTS SCRATCH LOC
; INMODE TO DETERMIN WHAT SPEED TO
; INPUT AT (19.2K OR 38.4K)
;
INPUT:  JSR    SCTTST      ;TST IF PRSNT CMND SCTR NUMBR
;           ;1,2,OR 3
;           STX    CSSCRT      ;CLR FOR FIRST CS ADD
;           BCC    IN0        ;BRNCH IF TST RESLT NOT 1,2OR3
;           LDX    #128       ;SET TO BRING IN ONLY 128 BYTS
IN0:    LDA    #255
;           STA    TIMR22      ;SET TIMR FOR 125MS
;           LDY    #7
;           STY    INSCRT      ;SET POINTR FOR CS IN
;           INC    INMODE      ;SET BIT 0 IN INMODE FOR INPUT
;           CLC
;           BIT    INMODE
;           BMI    IN00       ;BRNCH FOR 19.2K RATE
;           JMP    XTND8      ;GO TO XTRA ROM FOR 38K
IN00:   PHP
;           ;ADD
IN01:   LDA    TIMR2R
;           BMI    IN9        ;BRNCH IF TIMD OUT--DTA FLD ERR
IN1:    LDA    PORTB1
;           BPL    IN01       ;BRNCH IF NO STRT BIT
;           LDY    #12
INDLA1: DEY
;           BNE    INDLA1      ;INDLA1=54 MACHIN CYCLS
;           LDY    #128-8      ;SET BIT COUNTR
INDLA2: TYA
;           LDY    #15
INDLA3: DEY
;           BNE    INDLA3      ;DELAY 74 MACH CYC AT 19K, 14
;           ;CYC AT 38K FOR BIT CNTRS AND
;           ;49 CYC AT 38K FOR STRT BIT DLA
;           TAY
;           LDA    PORTB1      ;RETRIEVE 1 BIT
;           ROL
;           ;SHIFT TO CRY
;           ROR    ZPRL(X)     ;ASSEMBL TO PRSNT WORD
;           INY
;           BPL    IN4        ;BRNCH UNTIL 8 BITS ASSEMBLED

```

```

LDA      ZPRL(X)
EOR      #255
STA      ZPRL(X)          ;COMPLIMENT LAST BYTE ASSMBLD
PLP
JSR      DFCKSM           ;RETRIVE LAST CS ADD CRY
PHP
INX
BNE      IN5              ;BRNCH IF BUFERS NOT FULL
LDX      #4
IN6:    DEX
BNE      IN6              ;SET DELA OF 21 MACH CYC
LDX      #6
JSR      CHKIN            ;GET CS FROM COLEN
STA      DICKSM           ;SAV RECDV CS TEMP
JSR      SCTTST           ;TST IF SCTR 1,2,OR 3
BCC      IN8              ;BRNCH IF NOT SCTR 1,2,OR 3
IN7:    LDA      ZP80(X)
STA      ZPRL(X)          ;MOVE TOP HALF OF BUFERS TO
INX
BPL      IN7
IN8:    LDA      CSSCRT           ;GET COMPUTED CS
PLP
ADC      #0                ; ADD LAST CRY TO CS
DEC      INMODE           ;REST INMODE "INPUT BIT" FOR
                          ;EXIT FROM INPUT ROUT
CMP      DICKSM           ;COMPARE TWO CS'S
                          ;T FROM INPUT ROUT
;
;
;  DELAY OF 225 USECS ADDED BY DAVE:
;
      PHP                ;SAVE Z FLAG FROM COMPARE
      LDY      #90        ;LOAD TIMER
INDLY:  DEY
BNE      INDLY           ;LOOP TIL TIME OUT
PLP
;
      BEQ      IN10        ;IF GOOD CHECKSUM, RETURN Z SET
      LDA      #NAK        ;ELSE, GET NAK BYTE
      JSR      SAA2        ;SEND BACK TO COLLEEN
      JMP      WR5         ;UPDATE FLGS & GOTO INIT2
IN10:  RTS
;
;
;
IN4:   LDA      INMODE
      BMI      INDLA2
IN5:   LDY      #6
      STY      TIMR22      ;SET TIMR FOR 2.5MS DELA
INDLA4: DEY
      BNE      INDLA4      ;INDLA4=29 MACH CYC
      BEQ      IN1         ;ALWAYS BRNCH
IN9:   DEC      INMODE
      PLP
      LDA      #255
      RTS
;
; *****
;  SECTOR BREAKDOWN ROUTINE

```



```

; *****
;
;           BREAKS THE TWO BYTE
;  SECTOR NUMBER RECEIVED OVR THE
;  BUS DOWN TO A TRK AND SCTOR NUMBR
;
;
SCTBD:  LDA    AUX2
        STA    SSCTMS
        LDA    AUX1
        STA    SSCTLS
        LDX    #^H004
SCTBD1: ROR    AUX2
        ROR
        DEX
        BNE    SCTBD1
        ROR    AUX2
        LDX    #^H004
SCTBD2: LSR    AUX2
        DEX
        BNE    SCTBD2
SCTBD3: ASL
        CMP    AUX2
        BCS    SCTBD4
        STA    AUX1
        LSR
        STA    TRKNBR
        SEC
        LDA    AUX2
        SBC    AUX1
        STA    SCTNBR
        RTS
SCTBD4: LSR
        TAX
        DEX
        LDA    #^H010
        CLC
        ADC    AUX2
        STA    AUX2
        TXA
        JMP    SCTBD3
;
;
; *****
;  COMPUTE STEP/DIRECTION
; *****
;
;  COMPUTE # OF STEPS AND DIRECTION
;  TO DESIRED TRK
;
CSTEP:  LDX    #^H080           ;ASSUME POS RESLT "STP IN"
        LDY    #0              ;ASSUME NO STEP NEEDED
        SEC
        BIT    INMODE
        BVS    CST10
        SBC    PRTRK2
        JMP    CST3
CST10:  SBC    PRSTRK           ;SUB PRSNT TRK FROM DESIRED TRK
CST3:   BEQ    CST2            ;RESULT=00 MEANS NO STP,BRNCH

```

```

        BPL      CST1          ;IF RESULT POS ,BRNCH
        LDX      #0           ;SET X TO INDICATE "STP OUT"
        EOR      #^H0FF      ;2'S COMP RESLT
        CLC
        ADC      #1
CST1:   TAY          ;Y= NUMBR OF STPS TO EXECUTE
CST2:   RTS
;
;
; *****
;   STEP TEST
; *****
;
;           TESTS FOR FURTHER STEPS
;   TO EXECUTE,STEPz,WAITS 5 OR 10 MS
;
STPTST: BEQ      STST2          ;EXIT IF NO STPS TO DO
GOSTP:  JSR      STEP
        BEQ      STST1
WAIT5:  LDA      TIMR1R        ;WAIT 5 MS BFOR NXT STP
        BPL      WAIT5
        JMP      GOSTP
STST1:  LDA      #20
        STA      TIMR2          ;SET FOR 10 MS DELAY
WAIT10: LDA      TIMR1R        ;WAIT 10 MS FOR HEAD SETL
        BPL      WAIT10
        STX      STPDIR
STST2:  RTS
;
; *****
;   SEND ACKNOWLEDGE/ACCUM
; *****
;
;           XMIT AN ACKNOWLEDGE OR THE
;   CONTENTS OF THE ACCUMULATR TO COLLE
;
SAA:    BIT      PORTB1
        BVS      SAA          ;WAIT TIL COMND FRAME GONE
SAA1:   LDA      #ACK
SAA2:   STA      TMOPT        ;SET TO XMIT CONTNTS OF ACUM
        JSR      SETOP
        JSR      OUTPUT
        JSR      RSTOUT       ;RESET OUTPUT PORT
        RTS
;
;
; *****
;   SET OUTPUT PORT
; *****
;
;           NABLS OUTPUT PORT AND PUTS
;   A "1" ON IT (STOP BIT)
;
SETOP:  LDX      PORTB1
        STX      PORTB1       ;INSURE A "1" ON PORT
        LDX      PBDDR1
        INX
        STX      PBDDR1       ;NABL OUTPUT PORT
        RTS

```

```

;
; *****
; DATA TABLES
; *****
;
; ID FIELD TABLE
;
IDFLD:  .BYTE  ^H0A1,^H0A1,^H0A1,^H0FE,^H0FF,^H000
        .BYTE  ^H0FF,^H001,^H0FF,^H0FF,^H04E
;-----
; SCTRS---TABL USED BY FORMAT ROUT
; FOR SECTOR STAGR
;
SCTRS:  .BYTE  10,9,18,8,17,7,16,6,15
        .BYTE  5,14,4,13,3,12,2,11
;-----
; CLK/DATA LOOK-UP TABLE
;
;
CDLT:   .BYTE  ^H0AA,^H0A9,^H0A4,^H0A5,^H092,^H091,^H094,^H095,^H04A,^H049,^H044
        .BYTE  ^H045,^H052,^H051,^H054,^H055
        .BYTE  ^H02A,^H029,^H024,^H025,^H012,^H011,^H014,^H015,^H04A,^H049,^H044
        .BYTE  ^H045,^H052,^H051,^H054,^H055
;-----
;
; EXTENDED STATUS BYTES:
;
XSTAT:  .BYTE  ^H004,^H0D0,^H002,^H000,^H001
        .BYTE  ^H007,^H00C,^H000,^H000,^H000
        .BYTE  ^H0AA,^H055
;
ENDCOD  =      .
AVAIL   =      VEC-ENDCOD          ;# OF FREE BYTES IN THIS ROM
;
; *****
; POWER-UP VECTORS
; *****
;
        . =      ^H017FC
;
VEC:    .WORD  INIT,INIT
;
;
        .END

```