OS-65D V3.2 DISASSEMBLY MANUAL

SOFTWARE CONSULTANTS
7053 ROSE TRAIL / MEMPHIS, TENN. 38134 / (901) 377-3503

OS-65D VERSION 3.2 DISASSEMBLY

by

Software Consultants
7Ø53 Rose Trail
Memphis, TN 38134
(9Ø1) 377-35Ø3

This document is not from any official source, but was done using the "brute force method". That is, starting with the small amount of data released by OSI, each routine was painstakingly traced and decoded by hand. Great care was taken to insure accuracy throughout, however, if you do find any errors or ommisions, please let us know. We will then forward all such corrections to all purchasers.

In several places within this listing you will find comments which are less than complimentary to OSI. This was not done with the intent of belittling the original authors of OS-65D, but strictly to inform all readers of the shortcomings as well as the virtues of this operating system. If anyone feels we have been overly critical, we apologize.

If any of your friends asks you to allow them to make a copy of this document, please ask them to first read the following.

Software Consultants is a professional software house specializing in OSI compatible products. We are in business to make a profit, just as all businesses are. The OS-65D disassembly represents over 5ØØ manhours of research, compilation, and editing. The price was set as low as was possible while still allowing us a reasonable profit. If we are denied this reasonable profit by large numbers of people making pirate copies then we will not be able to continue working on other products for OSI equipment. You may save yourself a few dollars, but you will also be jeopardizing one of the very limited number of sources of high quality products for the OSI community.

GENERAL INFORMATION

One of the most frustrating features of using Ohio Scientific equipment is the almost total lack of useable documentation. OS-65D is supposed to be a "developmental" operating system, which implies that the user can develop his own machine language programs and tie them into the OS. Obviously this is not the case or this document would never have come into existance.

We originally broke the OS not as a money making project, but to enable us to tie our own machine language programs into the OS, and to give us the information necessary to make modifications that suit our needs. Once completed, we felt others attempting to use this OS could use this information to the same advantage that we have. Of course, the profit motive was also a deciding factor.

We assume that anyone using this document is throughly familiar with the workings of OS-65D V3.2 and is also a competent 65Ø2 assembler programmer. Every effort has been made to make each routine within the OS as clear as possible. However, this is a reference manual, not a textbook.

We suggest that upon first reading this document you simply scan through and read all comments rather than attempt to absorb the entire thing at one reading. Then you may go back and read the actual code after first getting a feel for the contents and flow of the OS.

This manual was intentionally printed on just one side of the paper to allow you to put your own notes on the facing pages. In particular, if you make changes to the OS, note each change in the listing along with it's purpose and the date made. If you will do your documentation as if you were going to be struck with amnesia tomorrow, it will truly make your life easier.

Following the listing of the OS itself is a complete cross reference showing the locations where each label is used. The location where the label is defined is marked with an asterisk. This should prove invaluable in both tracing logic and in assuring yourself that any changes made will not have any undesired side effects.

Our intention in the preparation of this manual was to make it as useful as possible to you, the purchaser. If after careful study of the listing, you still have unanswered questions about the workings of OS-65D, write us and we will attempt to answer your questions.

Happy computing!

DOS from Track ZERO

```
; INITIALIZATION ROUTINE
;
; THIS IS THE ENTRY POINT WHEN THE SYSTEM IS BOOTED.
; THE CODE FROM $2200 TO $22FF IS OVERLAYED BY BASIC WHEN IT IS CALLED.
;
2200 A9 01          LDA #1
2202 20 B6 22       JSR PATCH0      SET SECTOR # AND STEP RATE
2205 20 BC 26       JSR SETTK       MOVE TO TRACK 1
2208 A9 2A          LDA #$2A
220A 85 FF          STA MEMHI       SET HI MEM ADDR
220C 20 54 27       JSR LDHEAD      LOAD HEAD
220F 86 FE          STX MEMLO       SET LOW MEM ADDR TO 0
2211 20 67 29       JSR READDK      READ TK 1 INTO $2A00
2214 20 61 27       JSR UNLDHD      UNLOAD HEAD
2217 8E 01 F4       STX PTRPIA+1
221A 8E 00 F4       STX PTRPIA      CLEAR PRINTER PIA (X=0)
221D 8E 03 F4       STX PTRPIA+3
2220 CA             DEX             X=FF
2221 8E 01 DF       STX KPORT+1     SET KEYBOARD SOUND GENERATOR TO
;                                   LOWEST FREQUENCY (192.753 HZ)
;                                   THEN TURN IT OFF @ $228F!!!
2224 8E 02 F4       STX PTRPIA+2    SET PRINTER PIA
2227 AD 06 FB       LDA UART+6
222A 8E 05 FB       STX UART+5      SET SERIAL PORT
222D A9 04          LDA #4
222F 8D 01 F4       STA PTRPIA+1    PRINTER AGAIN
2232 8D 03 F4       STA PTRPIA+3
2235 8C 01 C0       STY FLOPIN+1    RESET DISK (Y=0)
2238 A0 40          LDY #$40
223A 8C 00 C0       STY FLOPIN
223D 8D 01 C0       STA FLOPIN+1
2240 A9 01          LDA #1
2242 20 C6 29       JSR SETDRV      SET TO DRIVE 1
2245 A9 03          LDA #3
2247 8D 00 FC       STA TERMAC      RESET TERMINAL ACIA
224A A0 11          LDY #$11
224C 8C 00 FC       STY TERMAC      SET TERMINAL ACIA
224F A2 1E          LDX #$1E
2251 9D 00 CF CLRX16 STA X16ACI,X   SET CA-10X 16 WAY SERIAL BOARD
2254 98             TYA             (IF ADDRESSED @ $CF00)
2255 9D 00 CF       STA X16ACI,X
2258 A9 03          LDA #3
225A CA             DEX
225B CA             DEX
225C 10 F3          BPL CLRX16
225E A2 08          LDX #8          CLEAR VIDEO SCREEN
2260 A9 D0          LDA #$D0
2262 85 FF          STA MEMHI
2264 A0 00          LDY #0
2266 84 FE          STY MEMLO
2268 A9 20          LDA #$20
226A 91 FE    CLRVID STA (MEMLO),Y
226C C8             INY
```

```
226D D0 FB          BNE CLRVID
226F E6 FF          INC MEMHI
2271 CA             DEX
2272 DO F6          BNE CLRVID
2274 86 00          STX PAGE0       X = 0
;
; WE ORIGINALLY THOUGHT THE ABOVE INSTRUCTION WAS USED FOR A
; FOR A PURPOSE WE HAVE NEVER SEEN DOCUMENTED. WHEN BASIC IS RUN
; IT PUTS A JUMP AT $0000 TO $0474 (4C 74 04).THIS JUMP WILL TAKE
; YOU TO THE COMMAND MODE. IF YOU RESET THE SYSTEM WHILE BASIC
; IS RUNNING AND DO NOT WISH TO LOSE THE PROGAM IN MEMORY, ALL YOU
; HAVE TO DO IS TO JUMP TO $0000 FROM THE MONITOR.  I.E. TYPE M THEN
; L012E0000RG FOR A SERIAL SYSTEM OR .0000G FOR A VIDEO SYSTEM.
; ANOTHER TIME WHEN THIS IS USEFUL IS WHEN BASIC IS AT AN INPUT
; STATEMENT AND YOU DO NOT WANT TO CONTINUE THE PROGRAM.  SINCE YOU
; CANNOT USE CONTROL C AT AN INPUT STATEMENT, JUST RESET AND DO THE
; ABOVE. WE SUSPECTED THE PURPOSE OF THIS INSTRUCTION WAS TO
; PREVENT DOING JUST THIS IF THE RESET IS HIT, THEN D, THEN RESET
; AGAIN BEFORE BASIC HAS BOOTED. ACTUALLY THAT IS NOT THE REASON,
; BUT SINCE THIS IS A USEFUL PIECE OF INFORMATION, WE PUT IT IN
; ANYWAY. THE REASON $00 IS SET TO 0 IS AS A FLAG FOR BASIC TO
; KNOW WHETHER OR NOT TO SWAP PAGE 0 AND 1 (SEE $2D50).
;
;
; MEMTST : HIGHEST MEMORY TEST ROUTINE
;
; THIS ROUTINE CHECKS FOR THE HIGHEST AVAILABLE MEMORY PAGE.
; IT STARTS WITH THE PAGE @ $BF00 AND MOVES DOWN IN STEPS OF ONE
; PAGE UNTIL IT FINDS MEMORY.  A WORD OF CAUTION.  IF YOU HAVE LESS
; THAN 48K AND INTEND TO USE SOME OF THE UPPER ADDRESS SPACE FOR
; HARDWARE, THEN THE STARTING PAGE ADDRESS @ $2277 SHOULD BE MODIFIED
; OR THE MEMORY TEST MAY DO STRANGE THINGS TO YOUR DEVICE.
;
2276 A0 BF    MEMTST LDY #$BF        START TEST @ $BF00
2278 20 EC 22        JSR MEMCHK      TEST THIS PAGE
227B F0 03           BEQ HMFND       IF SO, FOUND MEMORY
227D 88              DEY             TRY NEXT PAGE
227E D0 F8           BNE MEMTST+2    ALWAYS JUMP BACK
2280 8C 00 23 HMFND  STY HIMEM       STORE HIGHEST MEMORY PAGE
2283 A2 01           LDX #1          CHECK FOR SERIAL OR VIDEO
2285 AD 01 FE        LDA $FE01       (EITHER 65-A OR 65-V PROM)
2288 F0 01           BEQ *+1
228A E8              INX             IF VIDEO SET X=2
228B 8E C6 2A        STX DEFDEV      STORE DEFAULT DEVICE
; THE DEFAULT DEVICE ABOVE IS PICKED UP BY BEXEC* AND PUT INTO THE
; INPUT & OUTPUT DISTRIBUTOR BYTES.  THIS IS THE REASON THAT THE
; BASIC STARTUP MESSAGE IS NOT PRINTED ON BOOTING THE SYSTEM, SINCE
; THE OUTPUT DISTRIBUTOR ON DISK IS $00, WHICH DOES NOT OUTPUT TO
; ANYTHING.
228E EA              NOP
228F A2 01           LDX #1          SET VIDEO TO 64 CHAR/LINE
;                                    TURN OFF SOUND GENERATOR, COLOR
2291 8E 00 DE        STX VIDSIZ
```

```
2294 4C B3 22          JMP GOBAS        SKIP OVER UNUSED CODE!
;
;$2297-$22B2 IS UNUSED CODE
;
2297 EC 22 F0          CPX $F022
229A 18                CLC
229B A0 D7             LDY #$D7
229D 20 EC 22          JSR MEMCHK
22A0 D0 11             BNE $22B3
22A2 A0 00             LDY #0
22A4 BE C7 22          LDX $22C7,Y
22A7 F0 0A             BEQ $22B3
22A9 C8                INY
22AA B9 C7 22          LDA $22C7,Y
22AD 9D 99 25          STA VIDOUT,X
22B0 C8                INY
22B1 D0 F1             BNE $22A4
;
22B3 4C E6 2A GOBAS    JMP BASIC        LOAD AND EXECUTE BASIC
;
; THE JUMP ABOVE IS TO THE SAME ROUTINE USED WITH THE 'BA'
; COMMAND. YET 'BEXEC*' IS RUN ONLY WHEN THE SYSTEM IS BOOTED.
; THE METHOD USED TO DO THIS IS REALLY QUITE ELEGANT. THE INPUT
; DISTRIBUTOR ON DISK IS SET FOR MEMORY INPUT, WHILE THE MEMORY
; INPUT POINTER ON DISK POINTS TO $2E25. THIS IS WITHIN THE OS
; INPUT BUFFER. AND WHAT IS AT $2E25 ON DISK? WHY, 'RUN BEXEC* (CR)',
; OF COURSE. THEN WHEN 'BEXEC*' RUNS, IT SETS THE INPUT AND OUTPUT
; DISTRIBUTORS FROM THE DEFAULT DEVICE (SEE NOTE @ $228B), SO THE
; NEXT TIME THE 'BA' COMMAND IS EXECUTED, 'BEXEC*' IS NOT RUN.
;
; THIS PATCH ADDED FOR ADAPTIVE STEP RATE
;
22B6 8D 5E 26 PATCH0   STA SECTNM       SET SECTOR TO 1
22B9 A2 08             LDX #$08
22BB 86 EF             STX STEPRT       STEP RATE
22BD 60                RTS
;
; $22C7 THRU $22EB IS A TABLE USED BY THE UNUSED ROUTINE @$2297.
;
; MEMCHK : MEMORY CHECK SUBROUTINE.  CALLED @ $2278
;
; THERE MUST BE SOME REASON TO ONLY CHECK THE LOWEST SIX BITS OF
; THE BYTE UNDER TEST, BUT WE SURE CAN'T THINK OF ONE!
;
22EC 84 FF    MEMCHK   STY MEMHI       POINT TO PAGE UNDER TEST
22EE B1 FE             LDA (MEMLO),Y    GET A BYTE FROM THAT PAGE
22F0 29 3F             AND #$3F         KILL HIGHEST 2 BITS (?!)
22F2 49 3F             EOR #$3F         INVERT ALL REMAINING BITS
22F4 91 FE             STA (MEMLO),Y    PUT HASHED BYTE BACK
22F6 85 FD             STA TS2          AND SAVE IT
22F8 B1 FE             LDA (MEMLO),Y    GET BYTE BACK FROM MEMORY
22FA 29 3F             AND #$3F         KILL HIGHEST 2 BITS
22FC C5 FD             CMP TS2          IS IT THE SAME?
```

2294 A0 DB LDY #$DB
2296 20 EC 22 JSR 22EC
2299 F0 19 BEQ 22B3

```
22FE 60          RTS          EXIT WITH EQL FLAG SET
```

```
; OS-65D V3.2 (NMHZ)
;
; ZERO PAGE LOCATIONS USED BY OS
;
0000            PAGE0  = $0000          BASE OF PAGE ZERO
00E0            TS1    = $00E0          TEMPORARY STORAGE
00E1            OSIBAD = $00E1          OS INPUT BUFFER ADDRESS
00E3            STROAD = $00E3          ADDRESS USED BY STROUT ROUTINE
00E5            HSTTK  = $00E5          HIGHEST TRACK NUMBER OF FILE
00EE            TKNHLD = $00EE          TRACK NUMBER HOLD
00EF            STEPRT = $00EF          STEP RATE FOR DISK
00F5            SCTRTY = $00F5          SECTOR RETRY COUNT
00F6            WRTRTY = $00F6          WRITE RETRY COUNT
00F7            RDRTYM = $00F7          READ VERIFICATION RETRY COUNT
;                                       AFTER MOVING HEAD (3)
;                        NOT USED ON    VERIFY AFTER DISK WRITE
00F8            RDRTYN = $00F8          READ VERIFICATION RETRY COUNT
;                                       WITHOUT MOVING HEAD (7)
; ON READ, TOTAL RETRIES BEFORE AN ERROR = RDRTYN * RDRTYM (21)
00F9            SCTBYP = $00F9          SECTORS BYPASSED COUNTER
00FA            SCTLEN = $00FA          SECTOR LENGTH IN PAGES
00FB            SCTNUM = $00FB          SECTOR NUMBER
00FC            STKADR = $00FC          STACK ADDRESS
00FD            TS2    = $00FD          TEMPORARY STORAGE
00FE            MEMLO  = $00FE          INDIRECT MEMORY ADDRESS, LOW
00FF            MEMHI  = $00FF          "        "      "    , HI
;
; OTHER MEMORY ADDRESSES REFERRED TO BY THE OS
; ALL EXCEPT THOSE MARKED WITH AN ASTERISK ARE PART OF AN INSTRUCTION
; AND THERFORE ARE CASES OF SELF-MODIFING CODE.
; DURING THE LISTING, ANY ADDRESS WHICH IS MODIFIED IS DENOTED BY
; TWO ASTERISKS (**) IN THE PLACE OF AN ADDRESS.
;
0100            STACK  = $0100          * BASE OF STACK (PAGE 1)
0213            SWAP4A = $0213          * 4 BYTES SWAPPED DURING POLLED
;                                       KEYBOARD ROUTINE
1300            STASM  = $1300          * COLD START FOR ASSEMBLER
1303            RTASM  = $1303          * RESTART ASSEMBLER ENTRY POINT
1700            STEM   = $1700          * COLD START FOR EXTENDED MONITOR
;                                       THERE IS NO RESTART POINT
20C4            RTBAS  = $20C4          * RESTART BASIC ENTRY POINT
20E4            STBAS  = $20E4          * COLD START FOR BASIC
235F            X.HOLD = $235F          X REGISTER HOLD
2361            Y.HOLD = $2361          Y REGISTER HOLD
2363            A.HOLD = $2363          ACCUMULATOR HOLD
2378            IOOFS  = $2378          VECTORED I/O OFFSET
238A            MINADR = $238A          MEMORY INPUT ADDRESS
2391            MOTADR = $2391          "     OUTPUT   "
23AC            D1IADR = $23AC          DISK 1 BUFFER INPUT ADDRESS
23C3            D1OADR = $23C3          "    "    "     OUTPUT    "
23FD            D2IADR = $23FD          "    2    "     INPUT     "
2416            D2OADR = $2416          "    "    "     OUTPUT    "
25A4            VOTOFS = $25A4          VIDEO OUTPUT LINE OFFSET
```

```
262B            VLP1    = $262B         VIDEO LINE POINTER DURING SCROLL
262E            VLP2    = $262E         "     "      "      "      "
2639            VLOSAV  = $2639         "     "   OFFSET SAVE
267B            NMHZ    = $267B         NMHZ VARIABLE
;                                       $31=1MHZ, $62=2MHZ
;                                       WE CAN NOT FIND WHERE THIS IS
;                                       SET, PROBABLY IN BASIC.
2AC6            DEFDEV  = $2AC6         DEFAULT I/O DEVICE (SET @ $228B)
2CE5            BUFOFS  = $2CE5         OS BUFFER OFFSET
2CED            MAXBUF  = $2CED         MAXIMUM SIZE OF OS BUFFER
D700            PLINE   = $D700         * PRINT LINE FOR 540 VIDEO
;
; FLOPPY DISK PIA (MC6821)
;
C000            FLOPIN  = $C000         FLOPPY DISK STATUS PORT
;
; BIT  FUNCTION
; ---  --------
;  0   DRIVE 0 READY (0 IF READY)
;  1   TRACK 0 (0 IF AT TRACK 0)
;  2   FAULT (0 IF FAULT)
;  4   DRIVE 1 READY (0 IF READY)
;  5   WRITE PROTECT (0 IF WRITE PROTECT)
;  6   DRIVE SELECT (1 = A OR C, 0 = B OR D)
;  7   INDEX (0 IF AT INDEX HOLE)
;
C002            FLOPOT  = $C002         FLOPPY DISK CONTROL PORT
;
; BIT  FUNCTION
; ---  --------
;  0   WRITE ENABLE (0 ALLOWS WRITING)
;  1   ERASE ENABLE (0 ALLOWS ERASING)
;      ERASE ENABLE IS ON 200us AFTER WRITE IS ON
;      ERASE ENABLE IS OFF 530us AFTER WRITE IS OFF
;  2   STEP BIT : INDICATES DIRECTION OF STEP (WAIT 10 us FIRST)
;      0 INDICATES STEP TOWARD 76
;      1 INDICATES STEP TOWARD 0
;  3   STEP (TRANSITION FROM 1 TO 0)
;      MUST HOLD AT LEAST 10 us, MIN 8us BETWEEN
;  4   FAULT RESET (0 RESETS)
;  5   SIDE SELECT (1 = A OR B, 0 = C OR D)
;  6   LOW CURRENT (0 FOR TRKS 43-76, 1 FOR TRKS 0-42)
;  7   HEAD LOAD (0 TO LOAD : MUST WAIT 40ms AFTER)
;
; FLOPPY DISK ACIA (MC6850)
;
C010            ACIA    = $C010         DISK CONTROLLER ACIA STATUS PORT
C011            ACIAIO  = $C011         "      "        "    I/O    "
;
; OTHER HARDWARE ADDRESSES
;
CF00            X16ACI  = $CF00         NORMAL BASE ADDRESS OF CA10X BOARD
DE00            VIDSIZ  = $DE00         VIDEO SIZE (1 = 64 CHAR, 0 = 32)
```

```
DF00            KPORT   = $DF00         POLLED KEYBOARD PORT
F400            PTRPIA  = $F400         PARALLEL PRINTER PIA (MC6821)
FB00            UART    = $FB00         430 BOARD SERIAL PORT (S1883)
FC00            TERMAC  = $FC00         SERIAL TERMINAL ACIA STATUS PORT
FC01            TERMIO  = $FC01         "       "       "   I/O    "
FD00            KPOLL   = $FD00         POLLED KEYBOARD ROUTINE (ROM)
;
; THE ACIAS AT $CFXX AND $FC00 ARE ALL MC6850's
```

```
; START OF RESIDENT OS MEMORY AREA
;
2300            HIMEM   = $2300         HIGHEST MEMORY PAGE ADDRESS
;                                       SET @ $2280
;
; I/O DISPATCH TABLE (ADDRESS = ACTUAL ADDRESS - 1)
; ROUTINES ARE CALLED BY PUSHING THE ADDRESS ON
; THE STACK AND DOING AN RTS.
;
; INPUT DISPATCH TABLE
;
2301 F5 24      IOTABL .WORD TERMIN-1     TERMINAL (ACIA): BASIC DEVICE 1
2303 2A 25             .WORD KBINP-1      POLLED KEYBOARD: BASIC DEVICE 2
2305 17 25             .WORD SERINP-1     SERIAL (UART): BASIC DEVICE 3
2307 85 23             .WORD NULLIN-1     NULL: BASIC DEVICE 4
2309 88 23             .WORD MEMIN-1      MEMORY: BASIC DEVICE 5
230B A0 23             .WORD DK1IN-1      DISK1: BASIC DEVICE 6
230D EF 23             .WORD DK2IN-1      DISK2: BASIC DEVICE 7
230F AF 24             .WORD X16INP-1     CA10X: BASIC DEVICE 8
;
; OUTPUT DISPATCH TABLE
;
2311 CC 24             .WORD TERMOT-1     TERMINAL (ACIA): BASIC DEVICE 1
2313 98 25             .WORD VIDOUT-1     540 VIDEO: BASIC DEVICE 2
2315 0C 25             .WORD SEROUT-1     SERIAL (UART): BASIC DEVICE 3
2317 9E 24             .WORD PTROUT-1     PARALLEL PRINTER: BASIC DEVICE 4
2319 8F 23             .WORD MEMOT-1      MEMORY: BASIC DEVICE 5
231B B1 23             .WORD DK1OUT-1     DISK1: BASIC DEVICE 6
231D 02 24             .WORD DK2OUT-1     DISK2: BASIC DEVICE 7
231F BC 24             .WORD X16OUT-1     CA10X: BASIC DEVICE 8
;
;GENERAL STORAGE AREA
;
2321 01         INDST   = *             INPUT DISTRIBUTOR
2322 01         OUTDST  = *             OUTPUT DISTRIBUTOR
2323 00         X16DEV  = *             CA10X DEVICE # * 2 (0-1E)
2324 D4         RNDSED  = *             RANDOM NUMBER SEED
2325 00         KPDO    = *             KEY PRESSED DURING OUTPUT
2326 7E         D1BFLO  = *+2           DISK1 BUFFER LOW ADDRESS
2327 31
2328 7E         D1BFHI  = *+2           DISK1 BUFFER HI ADDRESS
2329 3D
232A 50         D1FRST  = *             DISK1 FIRST TRACK
232B 51         D1LAST  = *             DISK1 LAST TRACK
232C 50         D1CRTK  = *             DISK1 CURRENT TRACK
232D 00         D1BFDR  = *             DISK1 BUFFER 'DIRTY' FLAG
232E 7E         D2BFLO  = *+2           DISK2 BUFFER LOW ADDRESS
232F 3D
2330 7E         D2BFHI  = *+2           DISK2 BUFFER HI ADDRESS
2331 49
2332 50         D2FRST  = *             DISK2 FIRST TRACK
2333 51         D2LAST  = *             DISK2 LAST TRACK
2334 50         D2CRTK  = *             DISK2 CURRENT TRACK
```

```
2335 00       D2BFDR = *              DISK2 BUFFER 'DIRTY' FLAG
;
; START OF ACTUAL CODE
;
; INPUT/OUTPUT ROUTINES
;
2336 4C D6 2C IN1    JMP INPUT       USED BY INECHO  @$2340
;
; DOINP : DO VECTORED INPUT BASED ON VALUE IN INDST
;
; (SEE NOTE AT $2CD6)
; THE OS-65D MANUAL SAYS THAT INPUT IS DONE FROM THE LOWEST SET
; DEVICE & ALL OTHERS ARE IGNORED. WRONG!!! IF MORE THAN ONE BIT
; IS SET IN INDST, THINGS REALLY GO CRAZY. TRY ENTERING "IO 11"
; (OR "IO 12" FOR A VIDEO SYSTEM) AT "A*" AND WATCH THE RESULTS.
;
2339 A0 00    DOINP  LDY #$00        SET FOR INPUT
233B AD 21 23        LDA INDST       GET INPUT DISTRIBUTOR
233E D0 0B           BNE DOIO        GO DO INPUT
;
; INECHO : INPUT & ECHO. ALSO CHECKS FOR CONTROL CHARACTERS.
;
2340 20 36 23 INECHO JSR IN1         INPUT AND ECHO
;                                    THIS SHOULD HAVE BEEN
;                                    JSR INPUT. WHY THEY DID IT
;                                    THIS WAY WE DON'T KNOW.
;
; PRINT ROUTINE : OUTPUT TO ALL ACTIVE DEVICES
; OUTPUT CHARACTER IN A
;
2343 20 67 23 PRINT  JSR SAVAXY      SAVE ALL REGISTERS
2346 AD 22 23        LDA OUTDST      GET OUTPUT DISTRIBUTOR
2349 A0 10           LDY #$10        DENOTES OUTPUT
;
; DO I/O, EITHER INPUT OR OUTPUT BASED ON VALUE IN Y
;
234B A2 FF    DOIO   LDX #$FF        SET INDEX TO DETERMINE DEVICE
234D D0 22           BNE PATCH1 (#2371)GO TO PATCH FOR I/O OFFSET
234F E8              INX
2350 4A              LSR A           CHECK FOR I/O BIT ON
2351 90 09           BCC DONXIO      ($235C) BRANCH IF NOT
2353 48              PHA             SAVE REST OF I/O DIST BYTE
2354 8A              TXA             AND DEVICE NUMBER FOUND
2355 48              PHA             I/O DEVICE NOW IN A
2356 20 76 23        JSR IODISP      GO DO I/O
2359 68              PLA             RESTORE A AND X
235A AA              TAX
235B 68              PLA
235C D0 F1    DONXIO BNE DOIO+4      ($234F) IF ANY BITS STILL ON
;
; RSTAXY : RESTORE A,X,Y (USED AFTER SAVAXY)
; WARNING! THIS ROUTINE MASKS OUT THE UPPER BIT IN A
;
```

```
235E A2 00    RSTAXY LDX **          RESET X
2360 A0 00           LDY **          RESET Y
2362 A9 00           LDA **          RESET A
2364 29 7F           AND #$7F        KILL UPPER BIT IN A
2366 60              RTS             BACK WE GO
;
; SAVAXY : SAVE A,X,Y FOR LATER
;
2367 8D 63 23 SAVAXY STA A.HOLD      SAVE A
236A 8C 61 23        STY Y.HOLD      SAVE Y
236D 8E 5F 23        STX X.HOLD      SAVE X
2370 60              RTS
;
; PATCH TO SET I/O OFFSET
;
2371 8C 78 23 PATCH1 STY IOOFS       STORE I/O OFFSET
2374 D0 D9           BNE DOIO+4      ($234F) GO BACK
;
; IODISP : I/O DISPATCH ROUTINE
;
2376 0A       IODISP ASL A           MULTIPLY I/O DEVICE BY 2
2377 69 00           ADC **          I/O OFFSET (0=INPUT $10=OUTPUT)
2379 AA              TAX             GET SET TO GET I/O ADDRESS
237A BD 02 23        LDA IOTABL+1,X  GET HI BYTE
237D 48              PHA             PUSH ON STACK
237E BD 01 23        LDA IOTABL,X    GET THE LOW BYTE
2381 48              PHA             PUSH ON STACK
2382 AD 63 23        LDA A.HOLD      RESTORE A FOR OUTPUT
2385 60              RTS             JUMP TO ROUTINE
;
; NULLIN : NULL INPUT ROUTINE (BASIC DEVICE 4)
;
; WHILE THE NULL INPUT ROUTINE IN ITSELF IS NOT THAT USEFUL,
; SINCE IT IS 3 BYTES LONG IT COULD BE USED AS A JUMP TO A
; USER DEFINED INPUT ROUTINE.
;
2386 A9 00    NULLIN LDA #$00
2388 60              RTS
;
; MEMIN : INPUT FROM MEMORY ROUTINE (BASIC DEVICE 5)
; THIS ROUTINE IS ALSO USED FOR THE INDIRECT FILE FUNCTION
;
2389 AD 00 00 MEMIN  LDA **          GET BYTE FROM MEMORY
;                                    MODIFIED BY COMINC
238C A2 00           LDX #$00        SET OFFSET
238E F0 05           BEQ COMINC      GO TO COMMON INCREMENT ROUTINE
;
; MEMOT : MEMORY OUTPUT ROUTINE (BASIC DEVICE 5)
; THIS ROUTINE IS ALSO USED BY THE INDIRECT FILE FUNCTION.
;
2390 8D 00 00 MEMOT  STA **          PUT BYTE IN MEMORY
;                                    MODIFIED BY COMINC
2393 A2 07           LDX #$07        SET OFFSET
```

```
;
; COMINC : COMMON INCREMENT ROUTINE
;
; THE FOLLOWING ROUTINE IS USED BY THE DISK 1 AND 2 INPUT
; AND OUTPUT ROUTINES AS WELL AS THE MEMORY INPUT AND OUTPUT
; ROUTINES. THIS IS AN EXTREME CASE OF SELF MODIFING CODE WHICH
; SHOULD NORMALLY BE AVOIDED. X IS USED AS THE INDEX
; AND IS SET BY EACH INDIVIDUAL ROUTINE BEFORE CALLING THIS ROUTINE.
;
2395 8D 63 23 COMINC STA A.HOLD      SAVE A
2398 FE 8A 23        INC MINADR,X    INCREMENT MEMORY ADDRESS
239B D0 03           BNE *+5         ($23A0)
239D FE 8B 23        INC MINADR+1,X
23A0 60              RTS
;
; DK1IN : DISK 1 INPUT ROUTINE (BASIC DEVICE 6)
;
23A1 A0 00    DK1IN  LDY #$00        SET Y OFFSET
23A3 20 66 24        JSR CKBFEN-2    CHECK FOR END OF BUFFER
23A6 D0 03           BNE *+5         ($23AB) IF NOT END OF BUFFER, CONT
23A8 20 CC 23        JSR DK1NXT      READ NEXT TRACK
23AB AD 7E 31        LDA **          LOAD BYTE (MODIFIED BY COMINC)
23AE A2 22           LDX #$22        SET THE OFFSET
23B0 D0 E3           BNE COMINC      GO USE COMMON INCREMENT ROUTINE
;
; DK1OUT : DISK 1 OUTPUT ROUTINE (BASIC DEVICE 6)
;
; THIS ROUTINE WILL ALLOW YOU TO PRINT ANY CHARACTERS TO DISK EXCEPT
; A LINE FEED ($0A). SOMETIMES IT IS USEFUL TO BE ABLE TO WRITE A
; LINE FEED TO DISK, I.E. CREATING A WORD PROCESSOR OR ASSEMBLER
; FILE WITH BASIC. IF YOU WISH TO DO SO, YOU CAN CHANGE THE FOURTH
; BYTE OF EITHER DISK OUTPUT ROUTINE TO A NULL (HEX 0). JUST BE SURE
; YOU DON'T DO A "NORMAL" WRITE TO DISK WHILE THE CHANGE IS IN EFFECT
; OR THE CARRIAGE RETURN WILL BE FOLLOWED BY A LINE FEED.
;
23B2 C9 0A    DK1OUT CMP #$0A        IF LINE FEED THEN RETURN
23B4 F0 EA           BEQ DK1IN-1     ($23A0)
23B6 48              PHA             SAVE BYTE TO BE WRITTEN
23B7 A0 17           LDY #$17        SET Y FOR OFFSET
23B9 20 66 24        JSR CKBFEN-2    CHECK FOR END OF BUFFER
23BC D0 03           BNE *+5         ($23C1) CONTINUE IF NOT AT END
23BE 20 CC 23        JSR DK1NXT      WRITE THIS TRACK, READ NEXT
23C1 68              PLA             RESTORE THE OUTPUT BYTE
23C2 8D 7E 31        STA **          PUT IN BUFFER (MODIFIED BY COMINC)
23C5 A2 39           LDX #$39        SET OFFSET FOR COMMON INCREMENT
23C7 8E 2D 23        STX D1BFDR      SET BUFFER DIRTY FLAG
23CA D0 C9           BNE COMINC      BRANCH TO COMMON INCREMENT
;
; DK1NXT : DISK 1 NEXT TRACK READ, USED BY DK1IN AND DK1OUT
;
23CC AD 2D 23 DK1NXT LDA D1BFDR      GET BUFFER 'DIRTY' FLAG
23CF F0 05           BEQ *+7         ($23D6) IF NOT 'DIRTY' CONTINUE
23D1 A2 00           LDX #$00        SET OFFSET
```

```
23D3 20 77 24          JSR WTDKBF      GOSUB TO WRITE DISK BUFFER
23D6 AD 26 23          LDA D1BFLO      RESET READ/WRITE ADDRESS
23D9 8D AC 23          STA D1IADR      AND MEMORY ADDRESS TO START
23DC 8D C3 23          STA D1OADR      OF DISK BUFFER
23DF 85 FE             STA MEMLO
23E1 AD 27 23          LDA D1BFLO+1
23E4 8D AD 23          STA D1IADR+1
23E7 8D C4 23          STA D1OADR+1
23EA 85 FF             STA MEMHI
23EC A2 00             LDX #$00        SET OFFSET
23EE F0 52             BEQ BDRDNX      ALWAYS BRANCH
;
; DK2IN : DISK 2 INPUT ROUTINE (BASIC DEVICE 7)
;
23F0 A2 08    DK2IN    LDX #$08        SET OFFSETS
23F2 A0 51             LDY #$51
23F4 20 68 24          JSR CKBFEN      CHECK FOR END OF BUFFER
23F7 D0 03             BNE *+5         ($23FC) IF NOT END, CONTINUE
23F9 20 20 24          JSR DK2NXT      WRITE THIS BUFFER, READ NEXT
23FC AD 7E 3D          LDA **          LOAD BYTE FROM BUFFER
;                                      MODIFIED BY COMINC
23FF A2 73             LDX #$73        SET OFFSET
2401 D0 92             BNE COMINC      BRANCH TO COMMON INCREMENT
;
; DK2OUT : DISK 2 OUTPUT ROUTINE (BASIC DEVICE 7)
; SEE NOTE @$23B2 ABOUT LINE FEED
;
2403 C9 0A    DK2OUT   CMP #$0A        IF LINE FEED THEN RETURN
2405 F0 6F             BEQ $2476
2407 48                PHA             SAVE BYTE TO BE WRITTEN
2408 A2 08             LDX #$08        SET OFFSETS
240A A0 6A             LDY #$6A
240C 20 68 24          JSR CKBFEN      CHECK FOR END OF BUFFER
240F D0 03             BNE *+5         ($2414) IF NOT END THEN CONTINUE
2411 20 20 24          JSR DK2NXT      WRITE BUFFER, READ NEXT TRACK
2414 68                PLA             GET BYTE TO BE WRITTEN
2415 8D 7E 3D          STA **          PUT IN BUFFER (MODIFIED BY COMINC)
2418 A2 8C             LDX #$8C        SET OFFSET FOR COMINC
241A 8E 35 23          STX D2BFDR      SET BUFFER 'DIRTY' FLAG
241D 4C 95 23          JMP COMINC      DO INCREMENT FOR POINTER
;
; DK2NXT : DISK 2 NEXT TRACK READ, USED BY DK2IN AND DK2OUT
;
2420 AD 35 23 DK2NXT   LDA D2BFDR      GET BUFFER 'DIRTY' FLAG
2423 F0 05             BEQ *+7         ($242A) CONTINUE IF NOT 'DIRTY'
2425 A2 08             LDX #$08        SET OFFSET
2427 20 77 24          JSR WTDKBF      GO WRITE THIS BUFFER
242A AD 2E 23          LDA D2BFLO      RESET READ/WRITE ADDRESSES
242D 8D FD 23          STA D2IADR      AND MEMORY ADDRESS TO
2430 8D 16 24          STA D2OADR      START OF BUFFER
2433 85 FE             STA MEMLO
2435 AD 2F 23          LDA D2BFLO+1
2438 8D FE 23          STA D2IADR+1
```

```
243B 8D 17 24          STA D2OADR+1
243E 85 FF             STA MEMHI
2440 A2 08             LDX #$08          SET OFFSET
;
;
; THE NEXT GROUP OF ROUTINES ARE USED BY BOTH DISK 1 & DISK 2
; X IS SET TO 0 FOR DISK 1 AND TO 8 FOR DISK 2.
;
; BDRDNX : BUFFERED DISK I/O READ NEXT TRACK
;
2442 BD 2C 23 BDRDNX LDA D1CRTK,X      GET CURRENT TRACK NUMBER
2445 18              CLC               GET SET TO ADD 1 TO CURRENT TRACK
2446 F8              SED               SET DECIMAL (TRACK# IN BCD)
2447 69 01           ADC #$01          DO THE ADD
2449 D8              CLD               CLEAR DECIMAL MODE
244A 9D 2C 23        STA D1CRTK,X      SAVE THE TRACK NUMBER
244D 20 53 24        JSR BDMHTK        MOVE HEAD TO TRACK
2450 4C 1D 2B        JMP CALL+12       ($2B1D) READ DISK, UNLOAD HEAD
;                                      AND RETURN
;
; BDMHTK : BUFFERED DISK I/O MOVE HEAD TO TRACK
;
2453 A9 00     BDMHTK LDA #$00         CLEAR BUFFER 'DIRTY' FLAG
2455 9D 2D 23        STA D1BFDR,X
2458 BD 2C 23        LDA D1CRTK,X      COMPARE CURRENT TRACK
245B DD 2B 23        CMP D1LAST,X      WITH LAST TRACK
245E 20 8D 2C        JSR INCTKN+10     ($2C8D) MOVE HEAD TO TRACK, IF
;                                      PAST END OF FILE, ERROR D
2461 C8              INY               SET Y TO 1
2462 D0 2D           BNE PATCH2        ALWAYS BRANCH TO PATCH2
2464 00              BRK               (NOT USED)
2465 00              BRK               (NOT USED)
;
2466 A2 00           LDX #$00          SET OFFSET
;                                      USED BY DK1IN AND DK1OUT
;
; CKBFEN : CHECK FOR END OF BUFFER
;
2468 B9 AC 23 CKBFEN LDA D1IADR,Y      LOW ADDRESS OF BYTE TO BE READ
246B DD 28 23        CMP D1BFHI,X      LOW ADDRESS OF END OF BUFFER
246E D0 06           BNE *+8           ($2476) IF NOT THE SAME THEN RETURN
2470 B9 AD 23        LDA D1IADR+1,Y    HI ADDRESS OF BYTE TO BE READ
2473 DD 29 23        CMP D1BFHI+1,X    HI ADDRESS OF END OF BUFFER
2476 60              RTS               RETURN WITH Z FLAG SET IF END
;
; WTDKBF : WRITE DISK BUFFER
;
2477 BD 29 23 WTDKBF LDA D1BFHI+1,X    HI ADDR OF BUFFER HI ADDR
247A 38              SEC
247B FD 27 23        SBC D1BFLO+1,X    HI ADDR OF BUFFER LOW ADDR
247E 8D 5F 26        STA PGCNT         NUMBER OF PAGES
2481 BD 26 23        LDA D1BFLO,X      SET MEMORY ADDRESS TO LOW
2484 85 FE           STA MEMLO         BUFFER ADDRESS
```

```
2486 BD 27 23          LDA D1BFLO+1,X
2489 85 FF             STA MEMHI
248B 20 53 24          JSR BDMHTK       MOVE HEAD TO TRACK
248E 4C E1 27          JMP DSKWRT       WRITE TO DISK AND RETURN
;
; PATCH2  (FROM $2462)
;
2491 8C 5E 26 PATCH2 STY SECTNM         SET SECTOR NUMBER TO 1
2494 4C 54 27          JMP LDHEAD       LOAD HEAD AND RETURN
;
; MODMIN : MODIFY MEMORY INPUT ADDRESS
;
; THIS ROUTINE IS USED ONLY BY THE INPUT FROM INDIRECT FILE
; FUNCTION (CTRL X). IF YOU WANT TO CHANGE THE LOCATION OF
; THE INDIRECT FILE, YOU MUST CHANGE THE ADDRESS HERE AND IN
; THE ROUTINE @$2551.
;
2497 A9 80    MODMIN LDA #$80           HIGH ADDRESS FOR INDIRECT FILE
2499 8D 8B 23          STA MINADR+1     SAVE IT
249C A9 00             LDA #$00         LOW ADDRESS OF INDIRECT FILE
249E 60                RTS
;
; PTROUT : PARALLEL PRINTER OUTPUT DEVICE (BASIC DEVICE 4)
;
; NOTE:  SOME OF THE NEWER PRINTERS ON THE MARKET ARE EQUIPPED WITH
; GRAPHICS AND NEED THE FULL 8 BITS OF AN OUTPUT BYTE TO USE
; THIS FEATURE. CHANGING THE INSTRUCTION AT $24A7 AND $24A8 TO
; NOP'S ($EA) WILL ALLOW THIS.
;
249F 48        PTROUT PHA               SAVE BYTE TO BE PRINTED
24A0 AD 00 F4          LDA PTRPIA       CHECK PIA STATUS REGISTER
24A3 4A                LSR A
24A4 B0 FA             BCS PTROUT+1     ($24A0) NOT CLEAR, KEEP WAITING
24A6 68                PLA              RESTORE THE OUTPUT BYTE
24A7 29 7F             AND #$7F         KILL THE UPPER BIT
24A9 8D 02 F4          STA PTRPIA+2     OUTPUT THE BYTE
24AC AD 20 F4          LDA PTRPIA+$20   STROBE THE BYTE TO THE PRINTER
24AF 60                RTS
;
; BEFORE USING EITHER OF THE CA10X ROUTINES, THE PORT NUMBER MUST
; BE SET IN X16DEV ($2323)
;
; X16INP : CA10X INPUT ROUTINE (BASIC DEVICE 8)
;
24B0 AE 23 23 X16INP LDX X16DEV        GET ACIA DEVICE#
24B3 BD 00 CF          LDA X16ACI,X     GET ACIA STATUS REGISTER
24B6 4A                LSR A            SHIFT STATUS BIT TO CARRY
24B7 90 F7             BCC X16INP       TRY AGAIN IF NOT READY
24B9 B0 4D             BCS PATCH3       ($2508) GO TO PATCH3 TO INPUT
24BB 00                BRK              (NOT USED)
24BC 00                BRK              (NOT USED)
;
; X16OUT : CA10X OUTPUT ROUTINE (BASIC DEVICE 8)
```

```
;
24BD 48        X16OUT PHA              SAVE THE OUTPUT BYTE
24BE AE 23 23         LDX X16DEV       GET THE CURRENT DEVICE NUMBER
24C1 BD 00 CF         LDA X16ACI,X     GET THE STATUS REGISTER
24C4 4A               LSR A
24C5 4A               LSR A
24C6 90 F6            BCC X16OUT+1     ($24BE) IF NOT READY, TRY AGAIN
24C8 68               PLA              GET THE BYTE TO BE OUTPUT
24C9 9D 01 CF         STA X16ACI+1,X   WRITE IT
24CC 60               RTS
;
; TERMOT : TERMINAL OUTPUT ROUTINE (BASIC DEVICE 1)
;
24CD 48        TERMOT PHA              SAVE THE BYTE TO OUTPUT
24CE AD 00 FC         LDA TERMAC       GET THE ACIA STATUS
24D1 4A               LSR A
24D2 4A               LSR A
24D3 90 F9            BCC TERMOT+1     ($24CE) IF NOT READY, TRY AGAIN
24D5 68               PLA              GET THE BYTE TO PRINT
24D6 8D 01 FC         STA TERMIO       OUTPUT IT
24D9 48               PHA              SAVE IT AGAIN
24DA AD 00 FC         LDA TERMAC       GET THE STATUS AGAIN
24DD 4A               LSR A            CHECK FOR INPUT READY
24DE 90 11            BCC TORTN        NO KEY PRESSED, GO BACK
24E0 20 F6 24         JSR TERMIN       INPUT A CHARACTER
24E3 8D 25 23         STA KPDO         SAVE IT
24E6 C9 13            CMP #$13         CONTROL S?
24E8 D0 07            BNE TORTN        NO, GO BACK
24EA 20 F6 24         JSR TERMIN       YES, INPUT A BYTE
24ED C9 11            CMP #$11         CONTROL Q?
24EF D0 F9            BNE *-5          ($24EA) NO, TRY AGAIN
24F1 68        TORTN  PLA              RESTORE THE OUTPUT BYTE
24F2 8D 63 23         STA A.HOLD       SAVE IT
24F5 60               RTS
;
; TERMIN : SERIAL TERMINAL INPUT ROUTINE (BASIC DEVICE 1)
;
24F6 AD 00 FC TERMIN  LDA TERMAC       GET ACIA STATUS
24F9 EE 24 23         INC RNDSED       BUMP THE RANDOM SEED
24FC 4A               LSR A            CHECK RCV READY
24FD 90 F7            BCC TERMIN       IF NOT TRY AGAIN
24FF AD 01 FC         LDA TERMIO       INPUT THE BYTE
2502 29 7F            AND #$7F         KILL THE UPPER BIT
2504 8D 63 23 TIRTN   STA A.HOLD       SAVE THE CHARACTER
2507 60               RTS
;
; PATCH3 : ADDED TO X16INP ROUTINE  (FROM $24B9)
;
2508 BD 01 CF PATCH3  LDA X16ACI+1,X   GET BYTE FROM ACIA
250B B0 F7            BCS TIRTN        PUT IN A.HOLD AND RETURN
;
; SEROUT : 430 BOARD UART OUTPUT (BASIC DEVICE 3)
;
```

```
250D 48       SEROUT PHA              SAVE THE BYTE TO OUTPUT
250E AD 05 FB        LDA UART+5       GET UART STATUS
2511 10 FB           BPL SEROUT+1     ($250E) NOT READY, TRY AGAIN
2513 68              PLA              RESTORE THE OUTPUT CHARACTER
2514 8D 04 FB        STA UART+4       AND OUTPUT IT
2517 60              RTS
;
; SERINP : 430 BOARD UART INPUT (BASIC DEVICE 3)
;
2518 AD 05 FB SERINP LDA UART+5       GET THE UART STATUS
251B 4A              LSR A
251C 90 FA           BCC SERINP       NOT READY, TRY AGAIN
251E AD 03 FB        LDA UART+3       INPUT A BYTE
2521 8D 07 FB        STA UART+7       ACKNOWLEDGE INPUT
2524 8D 63 23        STA A.HOLD       SAVE THE BYTE
2527 60              RTS
;
; THE FOLLOWING IS A "WHO KNOWS" INSTRUCTION
; THIS IS ANOTHER CASE OF HOW TO USE UP COMPUTER TIME
;
2528 20 3F 25        JSR KIRTN        JUMP SUBROUTINE TO RTS
;
; KBINP : POLLED KEYBOARD INPUT ROUTINE (BASIC DEVICE 2)
;
; THIS ROUTINE USES THE SAME ROUTINE AS THE ROM BASED MACHINES.
; UNFORTUNATELY, THE DISK BASIC USES SOME OF THE SAME MEMORY
; LOCATIONS AS THE ROUTINE AT $FD00. INSTEAD OF DOING THE CORRECT
; THING, WRITING A NEW ROUTINE FOR THE DOS, OSI MADE ANOTHER PATCH.
; EVERY TIME YOU INPUT FROM THE 540 KEYBOARD YOU MUST FIRST SWAP
; OUT 4 BYTES, CALL THE ROUTINE IN ROM @$FD00, AND THEN RESTORE
; THE 4 BYTES. HIGHLY INEFFICIENT!
;
252B 20 44 26 KBINP  JSR SWAP4        SAVE $213-$216
252E EE 24 23        INC RNDSED       BUMP THE RANDOM SEED
2531 20 00 FD        JSR KPOLL        CALL THE ROUTINE IN ROM
2534 F0 F8           BEQ KBINP+3      ($252E) IF NULL THEN TRY AGAIN
;
; THIS IS ANOTHER STRANGE INSTRUCTION. THE PRESENT KEYBOARD ROUTINE
; WAITS UNTIL A KEY IS PRESSED AND THEN RETURNS IT'S ASCII VALUE.
; A NULL IS NEVER RETURNED FROM THE PRESENT KEYBOARD ROUTINE SO
; IT MAKES NO SENSE TO CHECK FOR IT.
;
2536 8D 63 23        STA A.HOLD       SAVE THE INPUT CHARACTER
2539 20 44 26        JSR SWAP4        RESTORE $213-$216
253C AD 63 23        LDA A.HOLD       GET THE INPUT CHARACTER
253F 60         KIRTN RTS
;
; PATCH4 : USED BY 540 VIDEO DRIVER FOR KEY PRESSED DURING OUTPUT
; (FROM $25F2)
;
2540 8D 25 23 PATCH4 STA KPDO         SAVE THE CHARACTER
2543 68              PLA
2544 4C 04 25        JMP TIRTN        SAVE A AND RETURN
```

```
;
; THIS IS AN UNDOCUMENTED RE-ENTRY POINT TO THE OS.  ON VIDEO SYSTEMS
; WHEN YOU EXIT TO THE MONITOR AND THEN RE-ENTER THE OS AT $2A51, YOU
; WILL NORMALLY HAVE PROBLEMS WITH THE POLLED KEYBOARD ROUTINE.  BY
; ENTERING AT $2547, THE 4 BYTES FROM $0213-$0216 ARE RESTORED AND THE
; KEYBOARD ROUTINE WILL WORK CORRECTLY.
;
2547 20 44 26        JSR SWAP4       SWAP THE 4 BYTES
254A 4C 51 2A        JMP OS65D3      JUMP TO THE OS
;
; CKINP : CHECK INPUT FOR INDIRECT FILE COMMANDS AND CONTROL P.
;
; THE CONTROL P IS A NICE FEATURE THAT WE HAVE NEVER SEEN DOCUMENTED
; BY OSI. UNDER VERSION 3.0 IT WAS A CONTROL T, WHILE UNDER
; VERSION 3.2 IT HAS BEEN CHANGED TO A CONTROL P. THIS CONTROL
; CHARACTER, WHICH EVER ONE IT IS, FLIP-FLOPS A FLAG THAT CONTROLS
; PRINTER OUTPUT. THE FIRST TIME THE CONTROL CHARACTER IS ENCOUNTERED
; IT TURNS ON THE PRINTER DEVICE AND THE NEXT TIME IT TURNS IT OFF.
; WARNING! SOME OF THE SOFTWARE PROVIDED ON THE SYSTEM USES THIS
; FUNCTION. WHEN USING WP2 IF YOU USE THIS FEATURE DURING OUTPUT
; THE WORD PROCESSOR TURNS IT OFF WHEN DONE. HOWEVER THE ASSEMBLER
; DOES NOT AFFECT IT AND IT REMAINS ON UNTIL THERE IS ANOTHER
; CONTROL (T/P) INPUT FROM THE KEYBOARD. THE PRINTER DEVICE BIT
; IS AT LOCATION $2592.
;
254D C9 5B    CKINP  CMP #$5B        ([) START INDIRECT FILE?
254F D0 11           BNE CKIFND      NO, CONTINUE
2551 A9 80           LDA #$80        SET UPPER ADDRESS FOR INDIRECT
2553 8D 92 23        STA MOTADR+1    MODIFY MEMORY OUTPUT ROUTINE
2556 A9 00           LDA #$00        SET LOWER ADDRESS FOR INDIRECT
2558 8D 91 23        STA MOTADR      MODIFY MEMORY OUTPUT ROUTINE
255B AD 22 23        LDA OUTDST
255E 09 10           ORA #$10        SET MEMORY OUTPUT
2560 D0 31           BNE CKIRTN      ALWAYS BRANCH TO EXIT
2562 C9 5D    CKIFND CMP #5D         (]) CLOSE INDIRECT FILE?
2564 D0 13           BNE CKCTLX      NO, CONTINUE
2566 20 46 23        JSR PRINT+3     PRINT ']', BYPASS SAVAXY
2569 AD C6 2A        LDA DEFDEV      I/O DEFAULT DEVICE
256C 8D 21 23        STA INDST       RESET INPUT POINTER
256F AD 22 23        LDA OUTDST      GET THE PRESENT OUTPUT DEVICE(S)
2572 29 EF           AND #$EF        TURN OFF MEMORY OUTPUT
2574 8D 22 23        STA OUTDST      SAVE THE OUTPUT DISTRIBUTOR
2577 A9 5D           LDA #$5D        PUT ']' BACK IN A
2579 C9 18    CKCTLX CMP #$18        CONTROL X? (LOAD INDIRECT FILE)
257B D0 0D           BNE CKCTLP      NO, CONTINUE
257D A9 10           LDA #$10
257F 8D 21 23        STA INDST       SET FOR MEMORY INPUT
2582 20 97 24        JSR MODMIN      GOSUB TO SET INPUT HIGH ADDRESS
2585 8D 8A 23        STA MINADR      SET INPUT LOW ADDRESS
2588 B0 0C           BCS CKIRTN+3    ($2596) ALWAYS BRANCH TO EXIT
258A C9 10    CKCTLP CMP #$10        IS IT CONTROL P
258C D0 0A           BNE CKIRTN+5    ($2598) NO, JUMP TO EXIT
258E AD 22 23        LDA OUTDST      GET THE OUTPUT DISTRIBUTOR
```

```
2591 49 08           EOR #$08        FLIP-FLOP THE PRINTER OUTPUT
2593 8D 22 23 CKIRTN STA OUTDST      SAVE THE DISTRIBUTOR
2596 A9 00           LDA #$00        DENOTES CONTROL CHARACTER FOUND
2598 60              RTS
;
; VIDOUT : 540 VIDEO OUTPUT ROUTINE (BASIC DEVICE 2)
;
; AS DELIVERED WITH THE SYSTEM THE 540 VIDEO DRIVER IS NOTHING
; MORE THAN A "GLASS TELETYPE" WITH NON-DESTRUCTIVE BACKSPACE
; AND FORWARD SPACE. CONSIDERING THE SOFTWARE SUPPLIED WITH OTHER
; COMPARABLE SYSTEMS, THIS IS RIDICULOUS. THE ROUTINE WILL NOT EVEN
; ALLOW YOU TO PRINT ANY OF THE OSI GRAPHICS CHARACTERS AND
; FORCES YOU TO "POKE" THEM TO THE SCREEN. ONE CHANGE THAT YOU
; COULD MAKE WOULD BE TO CHANGE THE INSTRUCTIONS FROM $25B9 TO
; $25C0 AND $25A1,$25A2 TO NOP'S. THIS WILL ALLOW YOU TO PRINT SOME
; GRAPHICS CHARACTERS. WARNING! THIS ROUTINE IS BAD ABOUT USING
; SELF MODIFYING CODE.
;
2599 98        VIDOUT TYA            SAVE Y FOR LATER
259A 48               PHA
259B AC 3C 26         LDY LCHAR      GET CHARACTER 'UNDER' CURSOR
259E AD 63 23         LDA A.HOLD     GET OUPUT CHARACTER
25A1 29 7F            AND #$7F       STRIP TO 7 BIT ASCII
25A3 A2 00            LDX **         GET OFFSET IN PRINT LINE
25A5 C9 0D            CMP #$0D       IS IT A 'CR'
25A7 F0 5A            BEQ CR         YES, DO IT
25A9 C9 0A            CMP #$0A       IF IT A 'LF'
25AB F0 62            BEQ LF         YES, DO IT
25AD C9 08            CMP #$08       BACKSPACE? (non dest cut H)
25AF F0 44            BEQ BSPACE     YES, DO IT
25B1 C9 10            CMP #$10       IS IT CNTRL P
25B3 F0 47            BEQ CNTLP      YES, DO IT
25B5 C9 0C            CMP #$0C       IS IT CNTRL L  (forward space non dest)
25B7 F0 43            BEQ CNTLP      YES, DO IT
25B9 C9 20            CMP #$20       IS IT < 'SPACE'
25BB 30 1A            BMI EXIT       YES, INVALID CHARACTER
25BD C9 7B            CMP #$7B       IS IT > '{'
25BF 10 16            BPL EXIT       YES, INVALID CHARACTER
25C1 9D 00 D7         STA PLINE,X    OUTPUT CHARACTER TO SCREEN
25C4 E8               INX            BUMP LINE POINTER
25C5 E0 80            CPX #$80       LAST CHARACTER ON LINE
25C7 F0 42            BEQ SCROLL     YES, DO SCROLL
25C9 BC 00 D7 EXIT    LDY PLINE,X    GET CHAR. 'UNDER' NEW CURSOR
25CC 8C 3C 26         STY LCHAR      SAVE IT
25CF A9 5F            LDA #$5F       GET CURSOR CHARACTER
25D1 9D 00 D7         STA PLINE,X    OUTPUT IT
25D4 8E A4 25         STX VOTOFS     SAVE OFFSET
25D7 68               PLA            RESTORE Y
25D8 A8               TAY
25D9 A9 01            LDA #$01       CHECK FOR 'CNTRL'
25DB 20 3D 26         JSR KEYTST
25DE 50 63            BVS KTRTN      NO, WE ARE DONE
25E0 A9 08            LDA #$08       CHECK FOR 'S'
```

```
25E2 20 3D 26          JSR KEYTST
25E5 10 5C             BPL KTRTN       NO, WE ARE DONE
25E7 AD 63 23          LDA A.HOLD      RESTORE OUTPUT CHARACTER
25EA 48                PHA             AND SAVE
25EB 20 2B 25          JSR KBINP       INPUT FROM POLLED KEYBOARD
25EE C9 13             CMP #$13        CNTRL S?
25F0 F0 F9             BEQ *-5         ($25EB) YES, KEEP LOOPING
25F2 4C 40 25          JMP PATCH4      EXIT THE ROUTINE
25F5 98        BSPACE  TYA             RESTORE CHAR. 'UNDER' CURSOR
25F6 9D 00 D7          STA PLINE,X     PRINT IT
25F9 CA                DEX             BUMP LINE POINTER BACK 1
25FA B0 CD             BCS EXIT        GO BACK
25FC 98        CNTLP   TYA             RESTORE CHAR. 'UNDER' CURSOR
25FD 9D 00 D7          STA PLINE,X
2600 E8                INX             BUMP LINE POINTER
2601 B0 C6             BCS EXIT        EXIT THIS ROUTINE
2603 98        CR      TYA             RESTORE CHAR. 'UNDER' CURSOR
2604 9D 00 D7          STA PLINE,X
2607 A2 40             LDX #$40        RESET LINE POINTER
2609 D0 BE             BNE EXIT        JUMP TO EXIT
260B A2 40     SCROLL  LDX #$40        RESET LINE POINTER
260D D0 04             BNE *+6         ($2613) JUMP A LITTLE
260F 98        LF      TYA             RESTORE CHAR. 'UNDER' CURSOR
2610 9D 00 D7          STA PLINE,X
2613 8E 39 26          STX VLOSAV      SAVE LINE OFFSET
2616 A9 20             LDA #$20        SET TO CLEAR LOWER LINE
2618 A2 80             LDX #$80        SET OFFSET
261A 9D 00 D7          STA PLINE,X     OUTPUT IT
261D E8                INX             BUMP THE OFFSET
261E D0 FA             BNE *-4         ($261A) LOOP UNTIL DONE
2620 A0 CF             LDY #$CF        GET SET TO SCROLL
2622 C8        SETNXT  INY             FIRST TIME THROUGH Y = $D0
2623 8C 2B 26          STY VLP1        ADJUST LINE POINTER
2626 8C 2E 26          STY VLP2        ADJUST LINE POINTER
2629 BD 40 D0  MOVE    LDA **,X        MOVE UP 1 LINE AT A TIME
262C 9D 00 D0          STA **,X
262F E8                INX             BUMP THE LINE POINTER
2630 F0 F0             BEQ SETNXT      IF MOVED LINE, SET FOR NEXT
2632 10 F5             BPL MOVE        KEEP LOOPING
2634 C0 D7             CPY #$D7        HAVE WE DONE THEM ALL
2636 90 F1             BCC MOVE        NO, KEEP LOOPING
2638 A2 00             LDX **          RESTORE LINE OFFSET
263A D0 8D             BNE EXIT        JUMP TO EXIT
263C           LCHAR   = *             CHARACTER 'UNDER' CURSOR
;
; KEYTST : TEST POLLED KEYBOARD FOR KEYDOWN IN ROW IN ACCUM
;
263D 8D 00 DF KEYTST STA KPORT         ENABLE THE ROW
2640 2C 00 DF        BIT KPORT         CHECK FOR KEYDOWN
2643 60       KTRTN  RTS
;
; SWAP4 : PATCH ADDED TO ENABLE USE OF POLLED KEYBOARD ROUTINE
; @$FD00. SWAPS OUT 4 BYTES FROM $213-$216 TO $2657-$265A
```

```
;
2644 A2 03     SWAP4   LDX #$03         SET INDEX FOR 4 BYTES
2646 BD 13 02          LDA SWAP4A,X     SWAP A BYTE
2649 BC 57 26          LDY SWAP4B,X
264C 9D 57 26          STA SWAP4B,X
264F 98                TYA
2650 9D 13 02          STA SWAP4A,X
2653 CA                DEX
2654 10 F0             BPL SWAP4+2      ($2646) IF ANOTHER CONTINUE
2656 60                RTS
;
2657           SWAP4B = *+4             SWAP AREA FOR $0213-$0216
```

```
; DISK DRIVER ROUTINES AND STORAGE
;
265B 20                  .BYTE $20              (UNKNOWN USAGE, IF ANY)
265C 01        DSKDR   = *                   PRESENT DISK DRIVE
265D 67        TKNUM   = *                   CURRENT TRACK NUMBER
265E 01        SECTNM  = *                   PRESENT SECTOR NUMBER
265F 07        PGCNT   = *                   PAGE COUNT
2660 00        LAMB    = *                   LOW ADDRESS OF MEMORY BLOCK
2661 00        HAMB    = *                   HI ADDRESS OF MEMORY BLOCK
2662 00        TRKNM   = *                   HEX TRACK NUMBER
;                                            NOT USED, SEE NOTE @ $26A6
;
; HOME : HOMES HEAD TO TRACK 0 ON CURRENT DISK DRIVE
;
2663 20 8A 26 HOME     JSR STEPOT            STEP HEAD OUT
2666 20 78 26          JSR TENMS             DELAY 10 MS
2669 8C 5D 26          STY TKNUM             SET TRACK# TO 0
266C A9 02    HOLOOP   LDA #$02              CHECK FOR TRACK 0
266E 2C 00 C0          BIT FLOPIN
2671 F0 05             BEQ TENMS             DELAY 10MS AND RETURN IF TR 0
2673 20 83 26          JSR STEPIN            STEP HEAD IN
2676 F0 F4             BEQ HOLOOP            LOOP BACK AND TRY AGAIN
;
; TENMS : 10 MS DELAY. ACTUALLY @ 1MHZ THE DELAY IS CLOSER TO 11 MS
;
2678 A2 0C    TENMS    LDX #$0C
267A A0 31             LDY #$31              LOOP COUNT FOR DELAY
267C 20 00 27          JSR DELAY             DO 1 MS DELAY
267F CA                DEX
2680 D0 F8             BNE TENMS+2           ($267A) NOT DONE, KEEP ON
2682 60                RTS
;
; STEPIN : STEP TOWARDS TRACK 0
; MOVES HEAD ONE TRACK
;
2683 AD 02 C0 STEPIN   LDA FLOPOT            TURN ON STEPIN BIT
2686 09 04             ORA #$04
2688 D0 05             BNE STEP              GO STEP IN
;
; STEPOT : STEP HEAD AWAY FROM TRACK 0
; MOVES HEAD ONE TRACK.
;
268A A9 FB    STEPOT   LDA #$FB              TURN OFF STEP IN BIT
268C 2D 02 C0          AND FLOPOT
268F 8D 02 C0 STEP     STA FLOPOT
2692 20 82 26          JSR STEPIN-1          ($2682) KILLS 12 CLOCK CYCLES
2695 29 F7             AND #$F7              TURN OFF STEP BIT
2697 20 19 27          JSR SETFLO            STA @ $C002 AND RETURN
269A 20 06 27          JSR DELAY+6           ($2706) KILL 14 CYCLES
269D 09 08             ORA #$08              TURN ON STEP BIT
269F 20 19 27          JSR SETFLO            STA @ $C002 AND RETURN
26A2 A6 EF             LDX STEPRT            GET STEP RATE
26A4 D0 D4             BNE TENMS+2           ($267A) DELAY STEP RATE MS
```

```
;
; (ROUTINE @ $26A6)  THIS ROUTINE CONVERTS A HEX TRACK NUMBER
; AT $2662 TO BCD AND STORES IT AT $EE, THEN FALLS INTO THE SET
; TRACK ROUTINE. THE ROUTINE IS NOT USED BY THE OS, SO EITHER
; IT IS USED BY BASIC, OR IT'S LEFT OVER FROM AN OLDER VERSION.
;
26A6 AD 62 26 CNVHTN LDA TRKNM
26A9 38              SEC
26AA A2 FF           LDX #$FF        INIT X TO COUNT 10'S
26AC E8              INX
26AD E9 0A           SBC #10         SUBTRACT 10 FROM TRACK#
26AF B0 FB           BCS *-3         ($26AC) IF >=0 BUMP X AND DO AGAIN
26B1 69 0A           ADC #10         ADD BACK LAST 10 FOR REMAINDER
26B3 85 EE           STA TKNHLD      SAVE REMAINDER
265B 8A              TXA             GET NUMBER OF TENS
26B6 0A              ASL A           SHIFT TO HIGH NIBBLE
26B7 0A              ASL A
26B8 0A              ASL A
26B9 0A              ASL A
26BA 05 EE           ORA TKNHLD      COMBINE WITH REMAINDER
;
; SETTK : CHECK FOR VALID TRACK NUMBER AND MOVE HEAD THERE
; TRACK NUMBER IN ACCUMULATOR
;
26BC 85 EE    SETTK  STA TKNHLD      SAVE TRACK NUMBER
26BE 48              PHA
26BF 2C 9E 26        BIT $269E       CHECK FOR 8 BIT
26C2 F0 04           BEQ ERR8-2      ($26CB) IF NOT, CONTINUE
26C4 29 06           AND #$06        CHECK FOR 4 BIT OR 2 BIT
26C6 D0 05           BNE ERR8        YES, LOW NIBBLE > 9 : ERROR 8
26C8 68              PLA             RESTORE TRACK NUMBER
26C9 C9 77           CMP #$77        TRACK < 77?
26CB 90 04           BCC MOVEHD      YES, CONTINUE
26CD A9 08    ERR8   LDA #$08        ERROR 8, BAD TRACK NUMBER
26CF D0 0D           BNE ERR6+2      ($26DE) JUMP TO ERROR HANDLER
26D1 AD 5C 26 MOVEHD LDA DSKDR       GET DISK DRIVE
26D4 29 01           AND #$01        TOP DRIVE=1, BOTTOM DRIVE=0
26D6 A8              TAY
26D7 20 DA 29        JSR CKRDY       SEE IF DRIVE IS READY
26DA 90 05           BCC CKTK        YES, CONTINUE
26DC A9 06    ERR6   LDA #$06        DRIVE NOT READY : ERROR 6
26DE 4C 4B 2A        JMP ERRENT      JUMP TO OS ERROR ROUTINE
26E1 A5 EE    CKTK   LDA TKNHLD      RETRIEVE TRACK NUMBER
26E3 CD 5D 26        CMP TKNUM       SAME AS PRESENT TRACK NUMBER?
26E6 F0 20           BEQ STCCNT      YES, DON'T MOVE THE HEAD
26E8 B0 07           BCS *+9         ($26F1) BRANCH IF > PRESENT TRACK
26EA 20 83 26        JSR STEPIN      STEP HEAD IN ONE TRACK
26ED A9 99           LDA #$99        SET TO SUBTRACT 1 FROM TKNUM
26EF 90 04           BCC *+6         ($26F5) JUMP
26F1 20 8A 26        JSR STEPOT      MOVE HEAD OUT 1 TRACK
26F4 8A              TXA             X=1 : SET TO ADD 1 TO TKNUM
26F5 F8              SED
26F6 6D 5D 26        ADC TKNUM       ADD OR SUBTRACT 1 FROM TKNUM
```

```
26F9 8D 5D 26        STA TKNUM       AND SAVE
26FC D8              CLD
26FD 4C E1 26        JMP CKTK        GO SEE IF WE ARE DONE
;
; DELAY : DELAY=18*Y+14 CYCLES  (DELAY=896us IF Y=$C1)
;
2700 20 9B 23 DELAY  JSR COMINC+6    ($239B) BNE AND RTS : 14 CYCLES
2703 88              DEY
2704 D0 FA           BNE DELAY       IF NOT DONE DO IT AGAIN
2706 EA              NOP
2707 60              RTS
;
; SET TRACK CODE CONTINUED FROM $26E6
;
2708 C9 43    STCCNT CMP #$43        ARE WE PAST TRACK 42
270A AD 02 C0        LDA FLOPOT
270D 29 BF           AND #$BF        RESET LOW CURRENT BIT
270F A0 00           LDY #$00        WHO KNOWS?
2711 EA              NOP
2712 B0 05           BCS SETFLO      IF PAST TRACK 42, CONTINUE
2714 A9 40           LDA #$40
2716 0D 02 C0        ORA FLOPOT      (PIA2) SET LOW CURRENT BIT
2719 8D 02 C0 SETFLO STA FLOPOT      STORE IT
271C 60              RTS
;
; WAITIH : WAIT FOR INDEX HOLE
;
271D AD 00 C0 WAITIH LDA FLOPIN      GET DISK STATUS
2720 30 FB           BMI WAITIH      IF BIT 7 ON, GO TEST AGAIN
2722 AD 00 C0        LDA FLOPIN      GET DISK STATUS
2725 10 FB           BPL *-3         ($2722) IF BIT 7 OFF, TRY AGAIN
2727 60              RTS
;
; LDHDWI : LOAD HEAD AND WAIT FOR INDEX HOLE
;
2728 20 54 27 LDHDWI JSR LDHEAD      LOAD HEAD
;
; RSACIA : RESET DISK ACIA, WAIT FOR INDEX HOLE
;
272B 20 1D 27 RSACIA JSR WAITIH      WAIT FOR THE INDEX HOLE
272E A9 03           LDA #$03
2730 8D 10 C0        STA ACIA        MASTER RESET FOR ACIA
2733 A9 58           LDA #$58        SET FOR /1, RTS=1, NO INTERUPT
2735 8D 10 C0        STA ACIA
2738 60              RTS
;
; EXAMCN : EXAM COMMAND CONTINUED , FIRST SECTION AT $2B37
;
2739 20 28 27 EXAMCN JSR LDHDWI      LOAD HEAD, WAIT FOR INDEX HOLE
273C A9 00 C0        LDA FLOPIN      GET THE STATUS
273F 10 20           BPL UNLDHD      IF AT INDEX HOLE, UNLOAD HEAD
2741 AD 10 C0        LDA ACIA        GET ACIA STATUS
2744 4A              LSR A
```

```
2745 90 F5          BCC EXAMCN+3    ($273C) NOT READY, WAIT FOR INDEX
2747 AD 11 C0       LDA ACIAIO      READ A BYTE
274A 91 FE          STA (MEMLO),Y   STORE IT IN MEMORY
274C C8             INY
274D D0 ED          BNE EXAMCN+3    ($273C) IF MORE IN THIS PAGE
274F E6 FF          INC MEMHI       BUMP MEMORY ADDRESS
2751 4C 3C 27       JMP EXAMCN+3    ($273C) CONTINUE
;
; LDHEAD : LOAD HEAD TO DISK
;
2754 A9 7F    LDHEAD LDA #$7F
2756 2D 02 C0       AND FLOPOT      SET BIT 7 TO 0
2759 8D 02 C0       STA FLOPOT
275C A2 28          LDX #$28        SET FOR 32 ms DELAY
275E 4C 7A 26       JMP TENMS+2
;
; UNLDHD : UNLOAD HEAD FROM DISK
;
2761 A9 80    UNLDHD LDA #$80
2763 0D 02 C0       ORA FLOPOT      SET BIT 7 TO 1
2766 D0 F1          BNE LDHEAD+5    ($2759) JUMP
;
; INITAL : INITIALIZE ALL TRACKS (EXCEPT ZERO) ON CURRENT DRIVE
;
2768 A9 76    INITAL LDA #$76       SET HIGHEST TRACK NUMBER
276A 85 E5          STA HSTTK
276C 20 63 26       JSR HOME        HOME THE HEAD
276F 20 83 2C       JSR INCTKN      INCREMENT TRACK
2772 20 7D 27       JSR INITTK      INITIALIZE THIS TRACK
2775 AD 5D 26       LDA TKNUM       GET CURRENT TRACK NUMBER
2778 C9 76          CMP #$76        AT 76 YET?
277A D0 F3          BNE INITAL+7    ($276F) NO, KEEP ON
277C 60             RTS
;
; INITTK : INITIALIZE TRACK
;
277D A9 02    INITTK LDA #$02
277F 2C 00 C0       BIT FLOPIN      CHECK FOR TRACK 0
2782 D0 04          BNE *+6         ($2788) NO, CONTINUE
2784 A9 03    ERR3  LDA #$03        DO ERROR #3
2786 D0 09          BNE ERR4+2      JUMP TO ERROR HANDLER
2788 A9 20          LDA #$20
278A 2C 00 C0       BIT FLOPIN      CHECK FOR WRITE PROTECT
278D D0 05          BNE ERR4+5      ($2794) NO, CONTINUE
278F A9 04    ERR4  LDA #$04        DO ERROR #4
2791 4C 4B 2A       JMP ERRENT      JUMP TO ERROR HANDLER
2794 20 28 27       JSR LDHDWI      LOAD HEAD AND WAIT FOR INDEX
2797 A9 FC          LDA #$FC        GET SET TO TURN ON
2799 2D 02 C0       AND FLOPOT      WRITE ENABLE AND ERASE ENABLE
279C 8D 02 C0       STA FLOPOT
279F A2 01          LDX #$01        DO  1 ms DELAY
27A1 20 7A 26       JSR TENMS+2
27A4 A2 43          LDX #$43        TRACK START CODE BYTE1
```

```
27A6 20 C2 27          JSR DKWTX        WRITE IT
27A9 A2 57             LDX #$57         TRACK START CODE BYTE2
27AB 20 C2 27          JSR DKWTX        WRITE IT
27AE AE 5D 26          LDX TKNUM        GET THE TRACK NUMBER
27B1 20 C2 27          JSR DKWTX        WRITE IT
27B4 A2 58             LDX #$58         TRACK TYPE CODE
27B6 20 C2 27          JSR DKWTX        WRITE IT
27B9 AD 00 C0          LDA FLOPIN       WAIT FOR INDEX, ERASE IS ON
27BC 30 FB             BMI *-3          ($27B9) NOT YET, TRY AGAIN
27BE A9 83             LDA #$83         TURN OFF WRITE ENABLE, ERASE
27C0 D0 A1             BNE UNLDHD+2     ($2763) ENABLE, UNLOAD HEAD & RET
;
; DKWTX : WRITE X TO DISK
;
27C2 AD 10 C0 DKWTX    LDA ACIA         GET ACIA STATUS
27C5 4A                LSR A
27C6 4A                LSR A
27C7 90 F9             BCC DKWTX        NOT READY, TRY AGAIN
27C9 8E 11 C0          STX ACIAIO       WRITE X TO DISK
27CC 60                RTS
;
; DSKBYT : GET BYTE FROM DISK
;
27CD AD 10 C0 DSKBYT   LDA ACIA         GET ACIA STATUS
27D0 4A                LSR A
27D1 90 FA             BCC DSKBYT       NOT READY, TRY AGAIN
27D3 AD 11 C0          LDA ACIAIO       READ THE BYTE
27D6 60                RTS
;
; THE FOLLOWING IS NOT USED BY THE OS. MAY BE USED BY BASIC
;
27D7 AD 60 26          LDA LAMB         GET LOW ADDRESS OF MEMORY BLOCK
27DA 85 FE             STA MEMLO        SAVE AT MEMORY ADDRESS
27DC AD 61 26          LDA HAMB         GET HIGH ADDRESS
27DF 85 FF             STA MEMHI        AND SAVE
;
; DSKWRT : WRITE SECTOR TO DISK ROUTINE
;
; TO USE THIS ROUTINE THE HEAD MUST ALREADY BE POSITIONED TO THE
; PROPER TRACK, THE NUMBER OF PAGES TO WRITE IN PGCNT ($265F), AND
; THE SECTOR NUMBER TO WRITE IN SECTNM ($265E). STARTING
; ADDRESS OF DATA MUST BE IN MEMLO,MEMHI ($FE,$FF).
;
27E1 AD 5F 26 DSKWRT   LDA PGCNT        GET NUMBER OF PAGES
27E4 F0 02             BEQ ERRB         IF 0 DO ERROR B
27E6 10 04             BPL *+6          ($27EC) IF BIT 7 IS ON DO ERROR B
27E8 A9 0B     ERRB    LDA #$0B         ERROR B ROUTINE
27EA D0 A5             BNE ERR4+2       ($2791) JUMP
27EC C9 0E             CMP #$0E         IF>D, THEN ERROR B
27EE 10 F8             BPL ERRB
27F0 A9 02             LDA #$02         TEST FOR TRACK 0
27F2 2C 00 C0          BIT FLOPIN
27F5 F0 DF             BEQ DSKBYT+9     ($27D6) IF TRACK 0 THEN RETURN
```

```
27F7 4A              LSR A
27F8 85 FA           STA SCTLEN      PUT 1 IN SECTOR LENGTH
27FA A9 20           LDA #$20        TEST FOR WRITE PROTECT
27FC 2C 00 C0        BIT FLOPIN
27FF D0 04           BNE *+6         ($2805) NOT WRITE PROTECT, CONTINUE
2801 A9 04           LDA #$04        WRITE PROTECT IS ON, ERROR 4
2803 D0 E5           BNE DSKWRT+8    ($27EA) JUMP
2805 A9 01           LDA #$01
2807 85 F6           STA WRTRTY      SET RETRY COUNT
2809 A9 03    REWRT  LDA #$03
2808 85 F8           STA RDRTYN      READ VERIFICATION RETRY COUNT
280D 20 C4 28        JSR SETSCT      POSITION TO START OF SECTOR
2810 20 9F 28        JSR DLYFA       DO 800us DELAY (SCTLEN = 1)
2813 A9 FE           LDA #$FE        SET WRITE ENABLE
2815 2D 02 C0        AND FLOPOT
2818 8D 02 C0        STA FLOPOT
281B A2 02           LDX #$02        DELAY 200us
281D 20 A2 28        JSR HUNDUS
2820 A9 FD           LDA #$FD        TURN ON ERASE ENABLE
2822 2D 02 C0        AND FLOPOT
2825 8D 02 C0        STA FLOPOT
2828 20 9F 28        JSR DLYFA       ANOTHER 800us DELAY
282B A2 76           LDX #$76
282D 20 C2 27        JSR DKWTX       WRITE SECTOR START CODE
2830 AE 5E 26        LDX SECTNM      GET SECTOR NUMBER
2833 20 C2 27        JSR DKWTX       WRITE IT
2836 AE 5F 26        LDX PGCNT       GET THE PAGE COUNT
2839 86 FD           STX TS2         SAVE IT
283B 20 C2 27        JSR DKWTX       WRITE PAGE COUNT
283E A0 00           LDY #$00        SET INDEX
2840 B1 FE    WRTPG  LDA MEMLO,Y     WRITE PAGE OF MEMORY TO DISK
2842 AA              TAX
2843 20 C2 27        JSR DKWTX       WRITE TO DISK
2846 C8              INY
2847 D0 F7           BNE WRTPG       ($2840) NOT DONE, LOOP BACK
2849 E6 FF           INC MEMHI       BUMP HIGH MEMORY ADDRESS
284B C6 FD           DEC TS2         DROP PAGE COUNT
284D D0 F1           BNE WRTPG       IF ANOTHER PAGE THEN CONTINUE
284F A2 47           LDX #$47        WRITE 'G' TO DISK
;                                    (SECTOR START CODE)
2851 20 C2 27        JSR DKWTX
2854 A2 53           LDX #$53        WRITE 'S' TO DISK
;                                    (SECTOR START CODE)
2856 20 C2 27        JSR DKWTX
2859 AD 5F 26        LDA PGCNT       GET PAGE COUNT
285C 0A              ASL A           MULTIPLY BY 2
285D 85 FD           STA TS2         SAVE IT
285F 0A              ASL A           MULTIPLY BY 2 AGAIN
2860 65 FD           ADC TS2         ADD TOGETHER, = 6*PAGE COUNT
2862 AA              TAX
2863 20 A2 28        JSR HUNDUS      100us DELAY*PAGE COUNT
2866 AD 02 C0        LDA FLOPOT
2869 09 01           ORA #$01        TURN OFF WRITE ENABLE
```

```
286B 8D 02 C0          STA FLOPOT
286E A2 05             LDX #$05
2870 20 A2 28          JSR HUNDUS      500us DELAY
2873 A9 02             LDA #$02
2875 20 16 27          JSR SETFLO-3    ($2716) TURN OFF ERASE ENABLE
2878 18        RTYCMP  CLC
2879 8A                TXA             ADD X TO HIGH MEMORY ADDRESS
287A 65 FF             ADC MEMHI       X=0 FIRST TIME WE COMPARE
287C 38                SEC             X=# OF SECTORS NOT COMPARED
;                                      IF THIS IS A RETRY
287D ED 5F 26          SBC PGCNT       RESET HIGH MEMORY ADDRESS
2880 85 FF             STA MEMHI
2882 20 07 29          JSR RDCDSK      COMPARE DATA WRITTEN TO DISK
;
; WARNING! IF WRITE STARTED FROM PAGE 0, ABOVE ROUTINE WILL
; READ FROM DISK INSTEAD OF COMPARE.
;
2885 B0 28             BCS DKBT9-1     ($28AF) NO FAULT SO RETURN
2887 C6 F8             DEC RDRTYN      DROP COMPARE RETRY COUNT
2889 D0 ED             BNE RTYCMP      IF NOT 0 THEN TRY AGAIN
288B C6 F6             DEC WRTRTY      DROP WRITE RETRY COUNT
288D 30 0C             BMI ERR2        IF DONE THEN ERROR #2
288F 8A                TXA
2890 65 FF             ADC MEMHI       RESET MEMORY ADDRESS
2892 38                SEC
2893 ED 5F 26          SBC PGCNT
2896 85 FF             STA MEMHI
2898 4C 09 28          JMP REWRT       WRITE TO DISK AGAIN
289B A9 02     ERR2    LDA #$02        ERROR #2
289D D0 22             BNE ERR9+2      ($28C1) ALWAYS JUMP
;
; DLYFA : 800us DELAY TIMES VALUE IN SCTLEN ($FA)
;
289F 20 56 29 DLYFA    JSR DLYFA1      GO COMPUTE VALUE FOR X
;
; HUNDUS : APPROXIMATELY 100us DELAY PER X
;
28A2 AD 7B 26 HUNDUS   LDA NMHZ        GET DELAY COUNT
28A5 24 00             BIT PAGE0       KILL 3 CYCLES
28A7 38                SEC
28A8 E9 05             SBC #$05        SUBTRACT 5 FROM DELAY COUNT
28AA B0 F9             BCS HUNDUS+3    ($28A5) IF >=0 THEN DO AGAIN
28AC CA                DEX
28AD D0 F3             BNE HUNDUS      DO X TIMES
28AF 60                RTS
;
; DKBT9 : GET BYTE FROM DISK, ERROR #9 IF INDEX HOLE
;
28B0 AD 00 C0 DKBT9    LDA FLOPIN      GET DISK STATUS
28B3 10 0A             BPL ERR9        IF INDEX HOLE THEN ERROR #9
;
; WARNING: $28B4 IS MODIFIED BY THE D9 COMMAND @ $2823
;
```

```
28B5 AD 10 C0          LDA ACIA        CHECK ACIA STATUS
28B8 4A                LSR A
28B9 90 F5             BCC DKBT9       NOT READY, KEEP LOOKING
28BB AD 11 C0          LDA ACIAIO      GET BYTE FROM DISK
28BE 60                RTS
28BF A9 09     ERR9    LDA #$09        ERROR #9, CAN'T FIND SECTOR
28C1 4C 4B 2A          JMP ERRENT
;
; SETSCT : SETUP FOR SECTOR IN SECTNM
;
28C4 A9 05     SETSCT  LDA #$05
28C6 85 F5             STA SCTRTY      SET RETRY COUNT
28C8 20 2B 27          JSR RSACIA      WAIT FOR INDEX HOLE
28CB 20 B0 28          JSR DKBT9       GET FIRST BYTE FROM DISK
28CE C9 43             CMP #'C 43      CHECK FOR TRACK START CODE
28D0 D0 F9             BNE *-5         ($28CB) IF NOT 'C' TRY AGAIN
28D2 20 B0 28          JSR DKBT9       GET SECOND BYTE FROM DISK
28D5 C9 57             CMP #'W 57      CHECK FOR TRACK START CODE
28D7 D0 F5             BNE *-9         ($28CE) IF NOT 'W' THEN CHECK FOR 'C'
28D9 20 B0 28          JSR DKBT9       GET ANOTHER BYTE FROM DISK
28DC 45 EE             EOR TKNHLD      IS THIS THE RIGHT TRACK?
28DE F0 0B             BEQ SSOK        YES, CONTINUE
28E0 A9 05     ERR5    LDA #$05        SET FOR ERROR #5, SEEK ERROR
28E2 C6 F5             DEC SCTRTY      BUT FIRST CHECK RETRY COUNT
28E4 10 61             BPL SEEKRT      FLIP SEEK RATE AND TRY AGAIN
28E6 CD A9 0A  ERRA    CMP $0AA9
;
; THE INSTRUCTION AT $28E6 IS AN OLD ASSEMBLER PROGRAMMING TRICK
; THAT SHOULD NORMALLY BE AVOIDED BECAUSE OF THE DANGERS INVOLVED.
; THIS IS A PRIME EXAMPLE OF MISUSE. THE TRICK IS TO TAKE A TWO
; BYTE INSTRUCTION; IN THIS CASE, LDA #$0A (A9 0A); AND ADD A BYTE
; TO THE FRONT WHICH CREATES A "HARMLESS" THREE BYTE INSTRUCTION.
; THEN YOU CAN FALL THROUGH FROM PRECEEDING CODE WITH NO EFFECT, OR
; BRANCH TO THE SECOND BYTE OF THIS INSTRUCTION FOR A DIFFERENT
; EFFECT, AS IS DONE AT $28FC. THIS ALLOWS REPORTING AN ERROR #5
; ON FALL THROUGH, OR ERROR A WHEN ENTERING AT $28E7. THE RATIONALE
; FOR USING THIS TRICK IS TO SAVE ONE LOUSY BYTE OF CODE. THE DANGER
; IS THAT QUITE OFTEN THE "HARMLESS" THREE BYTE INSTRUCTION CAN CAUSE
; CONSIDERABLE HARM. SUCH IS THE CASE HERE. SINCE OSI CHOSE TO USE A
; CMP INSTRUCTION, IF THE VALUE AT $0AA9 IS EQUAL TO 5, THE TEST AT
; $28E9 WILL FAIL AND THE PROGRAM WILL FALL INTO THE CODE USED WHEN
; THERE IS NO SEEK ERROR. ALSO $F9 WILL BE INITIALIZED TO 5 INSTEAD
; OF 0. ANOTHER ERROR, SUCH AS ERROR A, WILL PROBABLY OCCUR, BUT TO
; US PROBABLY IS NOT NEARLY GOOD ENOUGH.
;
28E9 D0 D6             BNE ERR9+2      ($28C1) GO REPORT ERROR 5 (OR A)
28EB 85 F9     SSOK    STA SCTBYP      SET SECTORS BYPASSED TO 0
28ED 20 B0 28          JSR DKBT9       GET FIRST BYTE FROM SECTOR
28F0 AD 5E 26          LDA SECTNM      GET SECTOR NUMBER
28F3 E9 01             SBC #$01        SUBTRACT 1
28F5 F0 0F             BEQ RDCDSK-1    ($2906) RETURN IF WANT SECTOR 1
28F7 48                PHA             SAVE SECTORS TO SKIP
28F8 20 98 29          JSR BPSECT      SKIP A SECTOR
```

```
28FB 68              PLA             GET BACK SECTORS TO SKIP
28FC 90 E9           BCC $28E7       IF CARRY CLEAR, ERROR A
28FE C5 F9           CMP SCTBYP      HAVE WE SKIPPED ENOUGH?
2900 D0 F5           BNE *-9         ($28F7) NO, CONTINUE
2902 C5 FB           CMP SCTNUM      SECTOR NUMBER JUST SKIPPED
;                                    SET @$29A4
2904 D0 E1           BNE $28E7       IF NOT RIGHT ONE, ERROR A
2906 60              RTS
;
; RDCDSK : READ (OR COMPARE) FROM DISK, THIS TRACK INTO MEMORY @($FE)
;
2907 48        RDCDSK PHA            SAVE READ/COMPARE FLAG
;                                    (0=READ)
2908 20 C4 28        JSR SETSCT      POSITION HEAD
290B 20 B0 28        JSR DKBT9       GET BYTE FROM DISK
290E C9 76           CMP #$76        IS IT A SECTOR START CODE?
2910 D0 F9           BNE *-5         ($290B) NO, TRY AGAIN
2912 20 B0 28        JSR DKBT9       GET ANOTHER BYTE
2915 CD 5E 26        CMP SECTNM      IS THIS THE RIGHT SECTOR?
2918 F0 03           BEQ *+5         ($291D) YES, CONTINUE
291A 68              PLA             IF NOT, RETURN WITH CARRY
;                                    CLEAR AS FAULT FLAG
291B 18        SETFF CLC
291C 60              RTS
;
291D 20 B0 28        JSR DKBT9       GET SECTOR LENGTH FROM DISK
2920 AA              TAX             PUT PAGE COUNT IN X
2921 8D 5F 26        STA PGCNT       STORE SECTOR LENGTH
2924 A0 00           LDY #$00        SET Y FOR INDEXING
2926 68              PLA             RESTORE READ/COMPARE FLAG
2927 69 FE           ADC #$FE        FORCE CARRY FLAG IF COMPARE
2929 A9 01     RDCONT LDA #$01
292B 2C 10 C0        BIT ACIA        CHECK ACIA READY AND PARITY
292E F0 FB           BEQ *-3         ($292B) IF NOT, TRY AGAIN
2930 AD 11 C0        LDA ACIAIO      GET BYTE FROM ACIA
2933 70 E6           BVS SETFF       PARITY ERROR, RETURN
2935 90 04           BCC *+6         ($293B) CARRY CLEAR, THIS IS A READ
2937 D1 FE           CMP (MEMLO),Y   COMPARE TO BYTE IN MEMORY
2939 D0 E0           BNE SETFF       IF NOT THE SAME, ERROR
293B 91 FE           STA (MEMLO),Y   STORE BYTE IN MEMORY
293D C8              INY             BUMP THE INDEX
293E D0 E9           BNE RDCONT      IF MORE THIS PAGE, CONTINUE
2940 E6 FF           INC MEMHI       SET ADDRESS FOR NEXT PAGE
2942 CA              DEX             DROP PAGE COUNT
2943 D0 E4           BNE RDCONT      IF ANOTHER PAGE, CONTINUE
2945 38              SEC             SET CARRY AS NO FAULT FLAG
2946 60              RTS
;
; SEEKRT : SEEK RETRY ROUTINE FOR ADAPTIVE STEP RATE
;
2947 A5 EF     SEEKRT LDA STEPRT     GET CURRENT STEP RATE
2949 49 0E           EOR #$0E        CHANGE 8 TO 6 OR 6 TO 8
294B 85 EF           STA STEPRT      STORE NEW STEP RATE
```

```
294D 20 63 26          JSR HOME          MOVE HEAD TO TRACK 0
2950 20 D1 26          JSR MOVEHD        GO MOVE HEAD TO PROPER TRACK
2953 4C C8 28          JMP SETSCT+4      AND TRY FOR SECTOR AGAIN
;
; DLYFA1 : COMPUTE 8 TIMES VALUE IN SCTLEN. USED BY DLYFA @ $289F
;
2956 A5 FA    DLYFA1   LDA SCTLEN        GET SECTOR LENGTH
2958 0A                ASL A             MULTIPLY BY EIGHT
2959 0A                ASL A
295A 0A                ASL A
295B AA                TAX               PUT IN X (FOR HUNDUS)
295C 60                RTS
;
; SET MEMORY ADDRESS POINTER TO DISK BUFFER ADDRESS
; NOT USED BY OS.
;
295D AD 60 26          LDA LAMB
2960 85 FE             STA MEMLO
2962 AD 61 26          LDA HAMB
2965 85 FF             STA MEMHI
;
; READDK : READ DISK, THIS TRACK INTO MEMORY @($FE)
;
2967 A9 03    READDK   LDA #$03          SET RETRY COUNT WHEN HEAD MOVED
2969 85 F7             STA RDRTYM
296B A9 07             LDA #$07          SET RETRY COUNT W/O MOVING HEAD
296D 85 F8             STA RDRTYN
296F A9 00    RTYRD    LDA #$00          DENOTES READ
2971 20 07 29          JSR RDCDSK        READ SECTOR INTO MEMORY
2974 90 04             BCC DKRDRY+3      ($297A) IF FAULT OCCURED, RETRY
2976 60                RTS
;
; DKRDRY : DISK READ RETRY
;
2977 C6 FF    DKRDRY   DEC MEMHI         RESET MEMORY ADDRESS
2979 E8                INX
297A EC 5F 26          CPX PGCNT
297D D0 F8             BNE DKRDRY        NOT DONE, CONTINUE
297F C6 F8             DEC RDRTYN        DROP RETRY COUNT
2981 D0 EC             BNE RTYRD         NOT 0, TRY AGAIN W/O MOVING HEAD
2983 20 83 26          JSR STEPIN        STEP HEAD IN
2986 20 78 26          JSR TENMS         10ms DELAY
2989 20 8A 26          JSR STEPOT        STEP HEAD OUT
298C 20 78 26          JSR TENMS
298F C6 F7             DEC RDRTYM        DROP RETRY COUNT
2991 10 D8             BPL READDK+4      ($296B) IF >=0 THEN TRY AGAIN
2993 A9 01    ERR1     LDA #$01          ALL RETRIES FAILED, ERROR #1
2995 4C 4B 2A          JMP ERRENT
;
; BPSECT : BYPASS SECTOR
;
2998 20 B6 29 BPSECT   JSR DKBTCI        GET BYTE FROM DISK
299B C9 76             CMP #$76          SECTOR START CODE?
```

```
299D D0 F9            BNE BPSECT       NO, TRY AGAIN
299F A2 02            LDX #$02         SET TO READ 2 BYTES
29A1 20 B6 29         JSR DKBTCI       GET BYTE FROM DISK
29A4 95 F9            STA SCTNUM-2,X   STORE SECTOR NUMBER IN $FB
;                                      STORE SECTOR LENGTH IN $FA (PAGES)
29A6 CA               DEX
29A7 D0 F8            BNE *-6          ($29A1) BACK FOR SECOND BYTE
29A9 E6 F9            INC SCTBYP       BUMP SECTORS BYPASSED
29AB A8               TAY              SECTOR LENGTH IN PAGES
29AC 20 B6 29         JSR DKBTCI       GET ANOTHER BYTE FROM DISK
29AF CA               DEX
29B0 D0 FA            BNE *-4          ($29AC) IF NOT END OF PAGE, CONTINU
29B2 88               DEY
29B3 D0 F7            BNE *-7          ($29AC) IF MORE PAGES TO GO, CONTIN
29B5 60               RTS
;
; DKBTCI : GET BYTE FROM DISK, IF INDEX HOLE SEEN POP STACK AND RETURN
;
29B6 AD 10 C0 DKBTCI  LDA ACIA         GET ACIA STATUS
29B9 4A               LSR A
29BA B0 07            BCS SETDRV-3     ($29C3) ACIA READY, GO AHEAD
29BC AD 00 C0         LDA FLOPIN       TEST FOR INDEX HOLE
29BF 30 F5            BMI DKBTCI       NO, TRY AGAIN
29C1 68               PLA              PULL LAST KNOWN RETURN ADDRESS
29C2 68               PLA              OFF OF STACK AND RETURN
29C3 4C BB 28         JMP DKBT9+11     ($28BB) LOAD BYTE AND RETURN
;
; SETDRV : SET FOR DRIVE IN ACCUMULATOR
;
29C6 8D 5C 26 SETDRV  STA TKNUM        SET TRACK NUMBER
29C9 0A               ASL A            MULTI BY 2 : A=2,B=4,C=6,D=8
29CA AA               TAX
29CB 29 02            AND #$02         ISOLATE DRIVE : A=1,B=0,C=1,D=0
29CD A8               TAY
29CE BD E9 29         LDA DKINIT-2,X   INITIALIZE PIA FROM INIT TABLE
29D1 8D 00 C0         STA FLOPIN
29D4 BD EA 29         LDA DKINIT-1,X
29D7 8D 02 C0         STA FLOPOT
;
; CKRDY : CHECK FOR DRIVE READY, RETURNS WITH CARRY CLEAR IF READY
;
29DA AD 00 C0 CKRDY   LDA FLOPIN       PUT READY BIT IN CARRY FLAG
29DD 4A               LSR A
29DE 08               PHP              SAVE CARRY STATUS
29DF C0 00            CPY #$00         IF TOP DRIVE THEN RETURN
29E1 D0 06            BNE DKINIT-2     ($29E9)
29E3 28               PLP              RESTORE STATUS
2934 4A               LSR A            PUT BIT 4 IN CARRY
2935 4A               LSR A
29E6 4A               LSR A
29E7 4A               LSR A
29E8 60               RTS
29E9 28               PLP              RESTORE STATUS
```

```
29EA 60              RTS
;
; DISK INITIALIZATION TABLE
;
29EB 40       DKINIT .BYTE $40        DRIVE A
29EC FF              .BYTE $FF
29ED 00              .BYTE $00        DRIVE B
29EE FF              .BYTE $FF
29EF 40              .BYTE $40        DRIVE C
29F0 DF              .BYTE $DF
29F1 00              .BYTE $00        DRIVE D
29F2 DF              .BYTE $DF
;
; DIRCNT : DIR COMMAND CONTINUED  (FROM $2B2C)
;
29F3 AA       DIRCNT TAX              PUT TRACK NUMBER IN X
29F4 F0 F1           BEQ DKINIT-4    ($29E7) IF 0 THEN RETURN
29F6 48              PHA             SAVE TRACK NUMBER
29F7 20 BC 26        JSR SETTK       MOVE HEAD TO TRACK
29FA 20 73 2D        JSR STROUT      PRINT THE FOLLOWING MESSAGE
29FD 0D 0A 54        .BYTE $0D,$0A,"TRACK ",0
2A00 52 41 43
2A03 4B 20 00
2A06 68              PLA             RESTORE THE TRACK NUMBER
2A07 20 92 2D        JSR PRT2HX      PRINT TRACK NUMBER
2A0A BA              TSX             SAVE STACK ADDRESS
2A0B 86 FC           STX STKADR
2A0D 20 54 27        JSR LDHEAD      LOAD HEAD TO DISK
2A10 E8              INX
2A11 8E 5E 26        STX SECTNM      PUT 1 IN SECTOR NUMBER
2A14 20 C4 28        JSR SETSCT      POSITION FOR SECTOR 1
2A17 A9 00           LDA #$00        CLEAR SECTORS BYPASSED COUNT
2A19 85 F9           STA SCTBYP
2A1B 20 98 29        JSR BPSECT      BYPASS THIS SECTOR
2A1E A5 FB           LDA SCTNUM
2A20 48              PHA             SAVE SECTOR NUMBER
2A21 A5 FA           LDA SCTLEN
2A23 48              PHA             SAVE SECTOR LENGTH
2A24 B0 F5           BCS *-9         ($2A1B) IF WE DIDN'T HIT THE INDEX
;                                    HOLE, TRY AGAIN
2A26 A6 FC           LDX STKADR      GET ORIGINAL STACK ADDRESS
2A28 90 0D           BCC *+15        ($2A37) AND JUMP
2A2A 20 6A 2D        JSR CRLF        PRINT CR/LF
2A2D A9 20           LDA #$20        PRINT SPACE AND SECTOR NUMBER
2A2F 20 41 2A        JSR DCPRNT
2A32 A9 2D           LDA #$2D        PRINT - AND SECTOR LENGTH
2A34 20 41 2A        JSR DCPRNT
2A37 C6 F9           DEC SCTBYP      DROP SECTORS BYPASSED COUNT
2A39 10 EF           BPL *-15        ($2A2A) IF MORE TO DO, CONTINUE
2A3B A6 FC           LDX STKADR      RESET STACK ADDRESS
2A3D 9A              TXS
2A3E 4C 61 27        JMP UNLDHD      UNLOAD HEAD AND RETURN
2A41 20 43 23 DCPRNT JSR PRINT       PRINT ACCUMULATOR
```

```
2A44 BD 00 01        LDA STACK,X     GET NEXT BYTE OFF STACK
2A47 CA              DEX             GET SET FOR THE NEXT ONE
2A48 4C 92 2D        JSR PRT2HX      PRINT AS 2 HEX CHAR. AND RETURN
```

```
; ** KERNEL **
;
; ERRENT : OS ERROR ENTRY. ERROR # IN ACCUMULATOR
;
2A4B 20 C4 2A ERRENT JSR OSERR (2AC4)
2A4E 4C 51 2A        JMP **
;
; WHILE IT MAKES LITTLE SENSE TO DO A DIRECT JUMP TO THE NEXT
; MEMORY LOCATION, THIS MAKES IT POSSIBLE TO ALTER THE EXIT
; FROM THE OS ERROR ROUTINE SO THAT IT WILL RETURN TO ANOTHER
; PLACE OTHER THAN THE OS. THIS CAN BE DONE WITH THE SETERR
; ROUTINE @ $2A7D. IF YOU ARE USING THE OS FROM YOUR OWN PROGRAM,
; YOU MAY ALSO WISH TO MODIFY THE OSERR ROUTINE TO NOT PRINT THE
; ERROR MESSAGE, IN WHICH CASE YOU WOULD NEED TO SET A FLAG TO INFORM
; YOUR PROGRAM THAT AN ERROR HAD OCCURED.
;
; OS65D3 : ENTRY POINT FOR OS65D MAIN LOOP
;
2A51 A2 28    OS65D3 LDX #$28
2A53 9A              TXS              SET STACK
;
; THE TOP OF STACK IS SET TO $28 SINCE THE NON-MASKABLE INTERRUPT
; VECTOR IS SET TO $0130.  WE WON'T EVEN COMMENT ON HOW ASININE IT
; IS TO PUT THE INTERRUPT VECTORS IN THE STACK AREA.
;
2A54 A9 51           LDA #$51         SET OS ERROR RETURN TO OS
2A56 A0 2A           LDY #$2A
2A58 20 7D 2A        JSR SETERR
2A5B 20 6A 2D        JSR CRLF
2A5E AD 5C 26        LDA DSKDR        GET PRESENT DISK DRIVE
2A61 18              CLC
2A62 69 40           ADC #$40         ACCUM NOW HAS LETTER OF
;                                     PRESENT DISK DRIVE
2A64 20 43 23        JSR PRINT        PRINT IT
2A67 A9 2A           LDA #'*
2A69 20 43 23        JSR PRINT        PRINT '*'
2A6C 20 9B 2C        JSR OSINP        DO INPUT TO OS BUFFER
2A6F A9 2E           LDA #$2E         OS INPUT BUFFER HI ADDRESS
2A71 85 E2           STA OSIBAD+1
2A73 A9 1E           LDA #$1E         OS INPUT BUFFER LOW ADDRESS
2A75 85 E1           STA OSIBAD
2A77 20 84 2A        JSR EXCOM        GO EXECUTE COMMAND
2A7A 4C 51 2A        JMP OS65D3       LOOP BACK FOR ANOTHER COMMAND
;
; SETERR : SET OS ERROR RETURN, LOW ADDRESS IN A
;          HIGH ADDRESS IN Y
;
2A7D 8D 4F 2A SETERR STA ERRENT+4     ($2A4F)
2A80 8C 50 2A        STY ERRENT+5     ($2A50)
2A83 60              RTS
;
; EXCOM : EXECUTE OS COMMAND SUBROUTINE
;
```

```
; TO EXECUTE OS COMMANDS FROM OTHER PROGRAMS EITHER PLACE THE COMMAND
; IN THE OS BUFFER (@$2E1E) AND DO A JSR TO EXCOM, OR PUT THE COMMAND
; IN MEMORY, SET THE BUFFER POINTER ($E1,E2) TO YOUR BUFFER. THEN DO
; A JSR TO EXCOM. YOU WOULD PROBABLY ALSO WANT TO SET THE OS ERROR
; RETURN TO YOUR OWN PROGRAM.
;
2A84 A2 00    EXCOM  LDX #$00        X=OFFSET INTO DISPATCH TABLE
2A86 8E E5 2C        STX BUFOFS      CLEAR BUFFER OFFSET
;                                    USED BY BUFBYT
2A89 A0 00           LDY #$00        Y=OFFSET INTO BUFFER
2A8B BD 30 2E        LDA DSPTBL,X    FIRST CHARACTER IN DISPATCH
;                                    TABLE ENTRY
2A8E F0 30           BEQ ERR7        IF 0 THEN DO ERROR #7
2A90 D1 E1           CMP (OSIBAD),Y  COMPARE TO BUFFER
2A92 D0 26           BNE NXTENT      IF NOT GO CHECK NEXT ENTRY
2A94 C8              INY             BUMP BUFFER INDEX
2A95 BD 31 2E        LDA DSPTBL+1,X  SECOND CHAR. IN TABLE ENTRY
2A98 D1 E1           CMP (OSIBAD),Y  COMPARE TO BUFFER
2A9A D0 1E           BNE NXTENT      IF NOT CHECK NEXT ENTRY
2A9C BD 33 2E        LDA DSPTBL+3,X  GET HIGH ADDRESS FROM TABLE
2A9F 48              PHA
2AA0 BD 32 2E        LDA DSPTBL+2,X  GET LOW ADDRESS
2AA3 48              PHA
2AA4 20 E4 2C        JSR BUFBYT      GET BYTE FROM BUFFER
2AA7 C9 0D           CMP #$0D        CHECK FOR 'CR'
2AA9 F0 0E           BEQ NXTENT-1    ($2AB9) IF IT IS, EXECUTE COMMAND
2AAB C9 20           CMP #$20        CHECK FOR A 'SPACE'
2AAD D0 F5           BNE *-9         ($2AA4) IF NOT, TRY AGAIN
2AAF 20 E4 2C        JSR BUFBYT      GET BYTE FROM BUFFER
2AB2 C9 20           CMP #$20        CHECK FOR A 'SPACE'
2AB4 F0 F9           BEQ *-5         ($2AAF) IF SO, LOOK AGAIN
2AB6 CE E5 2C        DEC BUFOFS      POINT TO FIRST NONSPACE
2AB9 60              RTS             JUMP TO ADDRESS FROM TABLE
2ABA E8       NXTENT INX             INCREMENT TO NEXT TABLE ENTRY
2ABB E8              INX             EACH ENTRY IS 4 BYTES
2ABC E8              INX
2ABD E8              INX
2ABE D0 C9           BNE EXCOM+5     ($2A89) GO BACK IF MORE TABLE
2AC0 A9 07    ERR7   LDA #$07        SYNTAX ERROR #7
2AC2 D0 87           BNE ERRENT      JUMP TO OS ERROR ENTRY (2A4B)
;
; OSERR : OS ERROR ROUTINE, ERROR # IN ACCUMULATOR
;
; ALWAYS CALLED FROM $2A4B. NOTE THAT THE I/O DISTRIBUTORS ARE
; RESET TO THE DEFAULT DEVICE ON ANY ERROR.
;
2AC4 48       OSERR  PHA
2AC5 A9 01           LDA #$01        GET DEFAULT I/O DISTRIBUTOR
2AC7 8D 21 23        STA INDST       AND RESET
2ACA 8D 22 23        STA OUTDST
2ACD 20 73 2D        JSR STROUT      PRINT THE FOLLOWING
2AD0 20 45 52        .BYTE ' ERR # ',0
2AD3 52 20
```

```
2AD5 23 00
2AD7 68              PLA
2AD8 20 9B 2D        JSR PRTHEX      PRINT THE ERROR NUMBER
2ADB 4C 61 27        JMP UNLDHD      UNLOAD HEAD AND RETURN
;
; ASM : ASSEMBLER COMMAND
;
2ADE A9 05    ASM    LDA #$05        FIRST TRACK NUMBER
2AE0 20 EE 2A        JSR LDCMN       COMMON CODE
2AE3 4C 00 13        JMP STASM       JUMP TO START OF ASSEMBLER
;
; BASIC : BASIC COMMAND
;
2AE6 A9 02    BASIC  LDA #$02        FIRST TRACK NUMBER
2AE8 20 EE 2A        JSR LDCMN       COMMON CODE
2AE8 4C E4 20        JMP STBAS       JUMP TO START OF BASIC
;
; LDCMN : LOAD LANGUAGE COMMON ROUTINE
; LOADS 3 TRACKS STARTING WITH TRACK IN ACCUM INTO MEMORY @ $0200 & UP
;
2AEE 20 BC 26 LDCMN  JSR SETTK       POSITION HEAD TO FIRST TRACK
2AF1 A2 02           LDX #$02
2AF3 86 E0           STX TS1         # OF TRACKS-1 TO READ
2AF5 86 FF           STX MEMHI       MEMORY ADDRESS HIGH=2
2AF7 CA              DEX
2AF8 8E 5E 26        STX SECTNM      SET SECTOR TO 1
2AFB CA              DEX
2AFC 86 FE           STX MEMLO       MEMORY ADDRESS LOW=0
2AFE CA              DEX
2AFF 86 E5           STX HSTTK       HIGHEST TRACK = $FF
2B01 20 54 27        JSR LDHEAD      LOAD HEAD TO DISK
2B04 20 67 29        JSR READDK      READ TRACK INTO MEMORY
2B07 C6 E0           DEC TS1
2B09 30 15           BMI D9-3        ($2B20) IF NO MORE TRACKS, DONE
2B0B 20 83 2C        JSR INCTKN      BUMP TRACK NUMBER AND SET HEAD
2B0E 4C 04 2B        JMP *-10        ($2B04) CONTINUE
;
; CALL : CALL COMMAND, READ SECTOR INTO MEMORY
;
2B11 20 23 2D CALL   JSR GETADR      MEMORY ADDRESS @$FE,FF
2B14 20 58 2D        JSR CKEQL       LOOK FOR = SIGN
2B17 20 60 2C        JSR GETTK       GET TRACK # AND SECTOR
2B1A 20 54 27        JSR LDHEAD      LOAD HEAD TO DISK
2B1D 20 67 29        JSR READDK      READ DISK INTO MEMORY
2B20 4C 61 27        JMP UNLDHD      UNLOAD HEAD AND RETURN
;
; D9 : DISABLE ERROR 9
;
2B23 A9 00    D9     LDA #$00
2B25 8D B4 28        STA DKBT9+4     ($28B4) CHANGE DKBT9 ROUTINE
2B28 60              RTS
;
; DIR : DIRECTORY COMMAND, PRINT SECTOR MAP OF TRACK
```

```
;
2B29 20 2E 2D DIR    JSR BLDHEX      GET TRACK NUMBER FROM BUFFER
2B2C 4C F3 29        JMP DIRCNT      GOTO ACTUAL CODE
;
; EM : CALL AND ENABLE EXTENDED MONITOR
;
2B2F A9 05     EM    LDA #$05        GET FIRST TRACK NUMBER
2B31 20 EE 2A        JSR LDCMN       COMMON CODE
2B34 4C 00 17        JMP STEM        GOTO START OF EXTENDED MONITOR
;
; EXAM : EXAM TRACK INCLUDING FORMATTING INFORMATION
;
; THIS IS A REALLY NICE COMMAND, EXCEPT THEY DON'T GIVE YOU ANY
; EASY WAY TO PUT THE DATA BACK ONTO THE DISK.
;
2B37 20 23 2D EXAM   JSR GETADR      MEMORY ADDRESS @$FE,FF
2B3A 20 58 2D        JSR CKEQL       LOOK FOR EQUAL SIGN
2B3D 20 2E 2D        JSR BLDHEX      GET TRACK NUMBER
2B40 20 BC 26        JSR SETTK       MOVE HEAD TO TRACK
2B43 4C 39 27        JMP EXAMCN      JUMP TO REST OF CODE
;
; GO : GO COMMAND
;
2B46 20 2E 2D GO     JSR BLDHEX      GET HIGH ORDER ADDRESS
2B49 8D 54 2B        STA GOADR+2     ($2B54) SAVE IT
2B4C 20 2E 2D        JSR BLDHEX      GET LOW ORDER ADDRESS
2B4F 8D 53 2B        STA GOADR+1     ($2B53) SAVE IT
2B52 4C 00 00 GOADR  JMP **          GO TO ADDRESS ENTERED
;
; INIT : INITIALIZATION COMMAND
;
2B55 20 E4 2C INIT   JSR BUFBYT      GET BYTE FROM BUFFER
2B58 C9 0D           CMP #$0D
2B5A F0 0C           BEQ FULINT      IF 'CR' THEN DO ENTIRE DISK
;                                    OTHERWISE, DO ONE TRACK
2B5C CE E5 2C        DEC BUFOFS      RESET BUFFER POINTER
2B5F 20 2E 2D        JSR BLDHEX      GET TRACK NUMBER
2B62 20 BC 26        JSR SETTK       MOVE HEAD TO TRACK
2B65 4C 7D 27        JMP INITTK      INITIALIZE TRACK AND RETURN
2B68 20 73 2D FULINT JSR STROUT      PRINT THE MESSAGE
2B6B 41 52 45        .BYTE 'ARE YOU SURE?',0
2B6F 20 59 4F
2B71 55 20 53 55
2B75 52 45 3F
2B78 00
2B79 20 40 23        JSR INECHO      INPUT AND ECHO 1 CHARACTER
2B7C C9 59           CMP #'Y
2B7E D0 26           BNE LOAD-1      ($2BA6) IF NOT 'Y' THEN RETURN
2B80 4C 68 27        JMP INITAL      DO REST OF CODE
;
; IO : I/O COMMAND (SEE NOTE AT $2339)
;
2B83 20 E4 2C IO     JSR BUFBYT      GET BYTE FROM BUFFER
```

```
2B86 C9 2C           CMP #',
2B88 F0 16           BEQ ONLYO       IF ',' DO OUTPUT ONLY
2B8A CE E5 2C        DEC BUFOFS      RESET BUFFER POINTER
2B8D 20 2E 2D        JSR BLDHEX      GET INPUT FLAG
2B90 8D 21 23        STA INDST       SAVE IT
2B93 20 E4 2C        JSR BUFBYT      GET BYTE FROM BUFFER
2B96 C9 0D           CMP #$0D
2B98 F0 0C           BEQ LOAD-1      ($2BA6) IF 'CR' THEN RETURN
2B9A CE E5 2C        DEC BUFOFS      RESET BUFFER POINTER
2B9D 20 5B 2D        JSR CKEQL+3     CHECK FOR COMMA
2BA0 20 2E 2D ONLYO  JSR BLDHEX      GET OUTPUT FLAG
2BA3 8D 22 23        STA OUTDST      STORE IT
2BA6 60              RTS
;
; LOAD : LOAD COMMAND
;
2BA7 20 A6 2D LOAD   JSR FNDFL       FIND FILE NAME IN DIRECTORY
2BAA 20 70 2C        JSR SETPGM      SET MEMORY ADDRESS & LOAD HEAD
2BAD 86 E0           STX TS1         X=0  USED AS # OF TRACKS READ
2BAF F0 03           BEQ *+5         ($2BB4) SKIP NEXT INSTR 1ST TIME
2BB1 20 83 2C        JSR INCTKN      BUMP TRACK NUMBER
2BB4 20 67 29        JSR READDK      READ TRACK INTO MEMORY
2BB7 E6 E0           INC TS1         BUMP TRACKS READ
2BB9 CE 7D 31        DEC $317D       DROP NUMBER OF TRACKS TO READ
2BBC D0 F3           BNE *-11        ($2BB1) IF MORE TRACKS, CONTINUE
2BBE A5 E0           LDA TS1
2BC0 8D 7D 31        STA $317D       RESET NUMBER OF TRACKS IN FILE
2BC3 4C 61 27        JMP UNLDHD      UNLOAD HEAD AND RETURN
;
; MEM : MEMORY COMMAND
;
2BC6 A2 00     MEM   LDX #$00        SET OFFSET FOR INPUT ADDRESS
2BC8 20 D0 2B        JSR *+8         ($2BD0) GET FROM BUFFER AND SAVE IT
2BCB 20 5B 2D        JSR CKCOMA      CHECK FOR COMMA
2BCE A2 07           LDX #$07        SET OFFSET FOR OUTPUT ADDRESS
2BD0 20 2E 2D        JSR BLDHEX      GET HIGH ORDER ADDRESS
2BD3 9D 8B 23        STA MINADR+1,X  SAVE IT
2BD6 20 2E 2D        JSR BLDHEX      GET LOW ORDER ADDRESS
2BD9 9D 8A 23        STA MINADR,X    SAVE IT
2BDC 60              RTS
;
; PUT : PUT COMMAND
;
; THERE IS A SERIOUS FLAW IN THE PUT COMMAND. IT ALWAYS WRITES WHOLE
; TRACKS STARTING @ $3179. IF YOU HAVE A VERY LARGE FILE IN MEMORY,
; SUCH AS A WORD PROCESSOR FILE, AND IT GOES BEYOND $B578 THEN THE
; LANGUAGES (BASIC, ASSEMBLER, WORD PROCESSOR) WILL COMPUTE 13
; TRACKS TO BE PUT TO DISK. UNFORTUNATELY, ATTEMPTING TO PUT OUT
; 13 TRACKS WILL CAUSE THE SYSTEM TO WRITE THE DISK CONTROLLER
; MEMORY TO DISK!!! THE READ AFTER WRITE CHECK WILL FAIL AND YOU
; WILL GET AN ERROR 2. IF YOU DON'T SEE THE ERROR WHEN IT OCCURS,
; AND ATTEMPT TO LOAD THE FILE LATER, VERY CURIOUS ERRORS HAPPEN.
; THE SIMPLEST FIX FOR THIS PROBLEM IS TO LIMIT THE AMOUNT OF MEMORY
```

```
; THE COMPUTER THINKS YOU HAVE BY CHANGING HIMEM @ $2300 TO $B4.
;
2BDD 20 A6 2D PUT    JSR FNDFL       FIND FILE NAME IN DIRECTORY
2BE0 20 70 2C        JSR SETPGM      SET MEMORY ADDRESS, LOAD HEAD
2BE3 AD 7D 31        LDA $317D       GET NUMBER OF TRACKS
2BE6 85 E0           STA TS1         SAVE IT
2BE8 A9 0B           LDA #$0B        NUMBER OF PAGES
;
; YET ANOTHER EXAMPLE OF AN OSI BLUNDER. EACH TRACK ON THE DISK
; IS CAPABLE OF HOLDING 13 SECTORS BUT THE PROGRAMMERS AT OSI
; ONLY USE 11 IN THE PUT COMMAND. THERE IS NO LOGICAL REASON
; TO DO THIS, MAYBE THEY THOUGHT THAT THIS WOULD HELP THEM TO
; SELL MORE DISKS. YOU MAY CHANGE THIS, AS WE HAVE, TO USE 12
; OR 13 SECTORS PER TRACK BY CHANGING THE PREVIOUS LDA #$0B TO
; LDA #$0C OR $0D. IF YOU DO DECIDE TO UTILIZE THE WASTED
; SECTORS WE WOULD ADVISE YOU TO GO TO A 12 SECTOR PER TRACK
; FORMAT AS THIS IS THE MOST THAT BASIC WILL RECOGNIZE.
; THIS WILL NOT HELP YOU WHEN SAVING BASIC OR ASSEMBLER PROGRAMS
; OR WORD PROCESSOR FILES AS ALL OF THESE LANGUAGES CALCULATE
; THE NUMBER OF TRACKS TO BE WRITTEN TO DISK BASED ON 11
; SECTORS PER TRACK. HOWEVER, IF YOU ARE DOING DISK I/O FROM
; YOUR OWN MACHINE LANGUAGE PROGRAMS, SUCH AS THE TEXT EDITOR
; USED TO PREPARE THIS DOCUMENT, YOU CAN USE 12 SECTORS PER TRACK
; WITHOUT ANY PROBLEM.
;
2BEA 8D 5F 26        STA PGCNT       SAVE IT
2BED 20 E1 27        JSR DSKWRT      WRITE TO DISK
2BF0 C6 E0           DEC TS1         DROP TRACK COUNT
2BF2 F0 06           BEQ *+8         ($2BFA) IF NO MORE THEN DONE
2BF4 20 83 2C        JSR INCTKN      BUMP TRACK NUMBER AND STEP HEAD
2BF7 4C ED 2B        JMP *-10        ($2BED) LOOP BACK & WRITE THIS TRACK
2BFA 4C 61 27        JMP UNLDHD      UNLOAD HEAD AND RETURN
;
; RET : RESTART COMMAND
;
; (*) NOTE, NOT ALL OF THESE WILL BE SET AT THE SAME TIME.
; EACH LANGUAGE SETS IT'S OWN RESTART ADDRESS AND SETS THE
; OTHERS TO REPORT AN ERROR. OF COURSE, THE ASSEMBLER/EXTENDED MONITOR
; SETS BOTH RETURN ADDRESSES.
;
2BFD 20 E4 2C RET    JSR BUFBYT      GET BYTE FROM BUFFER
2C00 C9 41           CMP #'A
2C02 D0 03           BNE *+5         ($2C07) NOT 'A' THEN CONTINUE
2C04 4C 03 13        JMP RTASM       REENTER ASSEMBLER (*)
2C07 C9 42           CMP #'B
2C09 D0 03           BNE *+5         ($2C0E) NOT 'B' THEN CONTINUE
2C0B 4C C4 20        JMP RTBAS       REENTER BASIC (*)
2C0E C9 45           CMP #'E
2C10 D0 03           BNE *+5         ($2C15) NOT 'E' THEN CONTINUE
2C12 4C 00 17        JMP STEM        ENTER EXTENDED MONITOR (*)
2C15 C9 4D           CMP #'M
2C17 D0 06           BNE *+8         ($2C1F) NOT 'M' THEN ERROR #7
2C19 20 44 26        JSR SWAP4       SWAP 4 BYTES FOR VIDEO ROUTINE
```

```
2C1C 6C FC FE          JMP ($FEFC)      JUMP TO RESET VECTOR
2C1F 4C C0 2A          JMP ERR7         DO ERROR #7
;
; XQT : LOAD FILE AND GO @$317E
;
; ONE USEFUL CHANGE TO THIS ROUTINE IS TO MAKE THE JUMP AT $2C25
; INTO AN INDIRECT JUMP TO $3179 (6C 79 31). SINCE THE PROGRAM MUST
; BE IN LOAD FORMAT ANYWAY, THIS WOULD ALLOW YOU TO HAVE A DISK
; BUFFER OR TWO AT THE FRONT OF THE WORKSPACE AND USE THE BASIC
; DISK I/O ROUTINES IN A STRAIGHTFORWARD FASHION.
;
2C22 20 A7 2B XQT      JSR LOAD         DO LOAD
2C25 4C 7E 31          JMP $317E        JUMP TO START OF PROGRAM
;
; SAVE : SAVE COMMAND, WRITE SECTOR TO DISK
;
2C28 20 60 2C SAVE     JSR GETTK        GET TRACK# AND POSITION HEAD
2C2B 20 58 2D          JSR CKEQL        CHECK FOR =
2C2E 20 23 2D          JSR GETADR       GET MEMORY ADDRESS AND PUT @$FE,FF
2C31 20 5E 2D          JSR CKEQL+6      CHECK FOR '/'
2C34 20 3D 2D          JSR GETHEX       GET NUMBER OF PAGES FROM BUFFER
2C37 8D 5F 26          STA PGCNT        SAVE IT
2C3A 20 54 27          JSR LDHEAD       LOAD HEAD
2C3D 20 E1 27          JSR DSKWRT       WRITE TO DISK
2C40 4C 61 27          JMP UNLDHD       UNLOAD HEAD AND RETURN
;
; SELECT : SELECT DISK DRIVE
; SETS PARAMETERS FOR DRIVE AND HOMES HEAD
;
2C43 20 E4 2C SELECT   JSR BUFBYT       GET BYTE FROM BUFFER
2C46 C9 41             CMP #'A          CHECK FOR A-D
2C48 30 0E             BMI ERR6-3       ($2C48) LESS THAN 'A', ERROR #7
2C4A C9 45             CMP #'E
2C4C 10 0A             BPL ERR6-3       ($2C58) >= 'E', ERROR #7
2C4E 29 0F             AND #$0F         KILL UPPER 4 BITS : A=1,D=4
2C50 20 C6 29          JSR SETDRV       SET FOR DRIVE
2C53 B0 06             BCS ERR6         ERROR #6 IF DRIVE NOT READY
2C55 4C 63 26          JMP HOME         HOME HEAD AND RETURN
2C58 4C C0 2A          JMP ERR7         DO ERROR #7
2C5B A9 06     ERR6    LDA #$06         DO ERROR #6
2C5D 4C 4B 2A          JMP ERRENT
```

```
; COMMON ROUTINES USED BY KERNEL
;
; GETTK : GET TRACK NUMBER & SECTOR FROM BUFFER & POSITION HEAD
;
2C60 20 2E 2D GETTK  JSR BLDHEX      GET TRACK NUMBER
2C63 20 BC 26        JSR SETTK       CHECK TRACK AND MOVE HEAD THERE
2C66 20 5B 2D        JSR CKCOMA      CHECK FOR COMMA
2C69 20 3D 2D        JSR GETHEX      GET SECTOR NUMBER
2C6C 8D 5E 26        STA SECTNM      SAVE IT
2C6F 60              RTS
;
; SETPGM : SET UP FOR PROGRAM
;
2C70 20 BC 26 SETPGM JSR SETTK       SET HEAD TO TRACK
2C73 A9 31           LDA #$31        SET MEMORY ADDRESS TO $3179
2C75 85 FF           STA MEMHI
2C77 A9 79           LDA #$79
2C79 85 FE           STA MEMLO
2C7B A9 01           LDA #$01
2C7D 8D 5E 26        STA SECTNM      SET SECTOR NUMBER TO 1
2C80 4C 54 27        JMP LDHEAD      LOAD HEAD AND RETURN
;
; INCTKN : INCREMENT TRACK NUMBER
;
2C83 AD 5D 26 INCTKN LDA TKNUM       GET TRACK NUMBER
2C86 18              CLC
2C87 F8              SED
2C88 69 01           ADC #$01        ADD 1 IN DECIMAL
2C8A D8              CLD
2C8B C5 E5           CMP HSTTK       IS THIS HIGHEST TRACK NUMBER?
2C8D F0 02           BEQ *+4         ($2C91) YES, LET'S CONTINUE
2C8F B0 03           BCS *+5         $(2C94) HIGHER, DO ERROR D
2C91 4C BC 26        JMP SETTK       SET HEAD AT TRACK AND RETURN
2C94 A9 0D    ERRD   LDA #$0D        ERROR D
2C96 D0 C5           BNE ERR6+2      ($2C5D) JUMP TO ERROR
;
2C98 20 6A 2D NXTOSN JSR CRLF        SET FOR NEXT OS INPUT
;
; OSINP : OS INPUT ROUTINE
;
; NOTE: THIS ROUTINE DOES NOT TRAP ILLEGAL CONTROL CHARACTERS.
; IF YOU PRESS 'BACKSPACE' ($08), THE PREVIOUS CHARACTER WILL
; BE ERASED, BUT BOTH THE 'BACKSPACE' AND THE CHARACTER WILL
; STILL BE IN THE BUFFER AND YOU WILL GET AN ERROR #7 EVEN
; THOUGH THE INPUT COMMAND LOOKS CORRECT.
;
2C9B A9 11    OSINP  LDA #$11        SET BUFFER SIZE
2C9D 8D ED 2C        STA MAXBUF
2CA0 A2 00           LDX #$00        X=CHARACTER COUNT
2CA2 20 40 23 NXTOSI JSR INECHO      GET A CHARACTER
2CA5 C9 5F           CMP #$5F        IS IT THE 'UNDERLINE'
2CA7 D0 13           BNE OSIOK       CONTINUE IF NOT
2CA9 CA              DEX             MOVE BACK ONE CHARACTER
```

```
2CAA 30 EC            BMI NXTOSN       TRY AGAIN
;                                      BACKSPACED AT FIRST CHARACTER
2CAC 9D 1E 2E         STA OSBUF,X      PUT IT IN BUFFER
2CAF 20 73 2D         JSR STROUT       DO BACKSPACE
;
; THIS PRINT FIRST DOES 2 BACKSPACES TO POSITION THE CURSOR
; AT THE CHARACTER TO BE DELETED. THE FIRST IS NECESSARY TO GET
; PAST THE UNDERLINE OR LEFT ARROW AND THE SECOND TO GET TO THE
; CHARACTER THAT WAS INPUT. THE ROUTINE THEN PRINTS 2 SPACES, 1
; TO ELIMINATE THE CHARACTER THAT WAS ENTERED AND ANOTHER TO
; ELIMINATE THE UNDERLINE OR LEFT ARROW. THE CURSOR IS THEN
; BACKSPACED TWICE TO REPOSITION IT SO YOU ARE READY TO
; ENTER THE CORRECT CHARACTER.
;
2CB2 08 08 20         .BYTE 8,8,'  ',8,8,0
2CB5 20 08 08
2CB8 00
2CB9 4C A2 2C         JMP NXTOSI       CONTINUE
2CBC C9 15    OSIOK   CMP #$15         CHECK FOR CONTROL U
2CBE F0 D8            BEQ NXTOSN       IF SO IGNORE INPUT UP TO NOW
2CC0 9D 1E 2E         STA OSBUF,X      PUT IN BUFFER
2CC3 C9 0D            CMP #$0D         CHECK FOR 'CR'
2CC5 F0 09            BEQ *+11         ($2CD0) IF SO THEN WE ARE DONE
2CC7 E8               INX              BUMP INDEX
2CC8 E0 11            CPX #$11         CHECK FOR MAXIMUM LENGTH
2CCA D0 D6            BNE NXTOSI       NOT DONE SO CONTINUE
2CCC A9 0D            LDA #$0D         BUFFER FULL, STOP INPUT AND PROCESS
2CCE D0 F0            BNE OSIOK+4      ($2CC0) JUMP
2CD0 4C 6A 2D         JMP CRLF
2CD3 00               BRK              (UNUSED)
2CD4 00               BRK              (UNUSED)
2CD5 00               BRK              (UNUSED)
;
; INPUT : INPUT ROUTINE. CHECKS FOR CONTROL CHARACTERS.
;
; (SEE NOTE AT $2339)
; WHEN WRITING YOUR OWN INPUT ROUTINES TO BE USED WITH THE OS
; YOU SHOULD STORE THE INPUT CHARACTER IN A.HOLD BEFORE RETURNING
; FROM YOUR ROUTINE SINCE THE INPUT ROUTINE RESTORES A,X,Y
; WHEN IT RETURNS. IF YOU DO NOT DO THIS YOUR INPUT WILL BE THE
; CHARACTER IN A WHEN THE ROUTINE WAS CALLED.
;
2CD6 20 67 23 INPUT   JSR SAVAXY
2CD9 20 39 23         JSR DOINP        GO DO INPUT
2CDC 20 4D 25         JSR CKINP        CHECK FOR CONTROL CHARACTERS
2CDF F0 F8            BEQ INPUT+3      ($2CDF) IF SO CONTINUE INPUT
2CE1 4C 5E 23         JMP RSTAXY       RESTORE REGISTERS AND GO BACK
;
; BUFBYT : GET BYTE FROM BUFFER
;
2CE4 A0 07    BUFBYT  LDY #BUFOFS      GET OFFSET INTO BUFFER
;                                      MORE SELF MODIFYING CODE
2CE6 B1 E1            LDA (OSIBAD),Y   LOAD BYTE
```

```
2CE8 C9 0D            CMP #$0D        CHECK FOR 'CR'
2CEA F0 07            BEQ *+9         ($2CF3) IF SO WE ARE DONE
2CEC C0 11            CPY #$11        CHECK FOR END OF BUFFER
2CEE F0 04            BEQ *+6         ($2CF4) IF SO THEN RETURN
2CF0 EE E5 2C         INC BUFOFS      BUMP THE OFFSET
2CF3 60               RTS
2CF4 A9 0D            LDA #$0D        LOAD A 'CR'
2CF6 60               RTS             RETURN, BUFFER IS FULL
;
; SWAP : SWAP PAGE 0 AND 1 WITH $2F79 AND UP (USED BY BASIC)
; THIS ROUTINE IS NOT CALLED ANYWHERE BY THE OS
;
2CF7 68        SWAP   PLA             CHANGE RETURN ADDRESS
2CF8 18               CLC             INTO JUMP @$2D20
2CF9 69 01            ADC #$01
2CFB 8D 21 2D         STA GETADR-2    ($2D21)
2CFE 68               PLA
2CFF 69 00            ADC #$00
2D01 8D 22 2D         STA GETADR-1    ($2D22)
2D02 A2 00            LDX #$00        SET THE OFFSET
2D06 BD 00 01 SWAPLP  LDA STACK,X     GET BYTE FROM PAGE 1
2D09 BC 79 30         LDY SWAP1,X     GET BYTE FROM SWAP AREA
2D0C 9D 79 30         STA SWAP1,X     SAVE THE BYTE FROM PAGE 1
2D0F 98               TYA
2D10 9D 00 01         STA STACK,X     SAVE THE BYTE FROM SWAP AREA
2D13 B5 00            LDA PAGE0,X     GET BYTE FROM PAGE 0
2D15 BC 79 2F         LDY SWAP0,X     GET BYTE FROM SWAP AREA
2D18 9D 79 2F         STA SWAP0,X     SAVE BYTE FROM PAGE 0
2D1B 94 00            STY PAGE0,X     SAVE BYTE FROM SWAP AREA
2D1D E8               INX             BUMP THE OFFSET
2D1E D0 E6            BNE SWAPLP      NOT DONE, KEEP ON
2D20 4C C7 14         JMP **          ADDRESS FOR JUMP IS CHANGED ABOVE
;
; GETADR : GET MEMORY ADDRESS FROM BUFFER
;
2D23 20 2E 2D GETADR  JSR BLDHEX
2D26 85 FF            STA MEMHI       HIGH ORDER BYTE
2D28 20 2E 2D         JSR BLDHEX
2D2B 85 FE            STA MEMLO       LOW ORDER BYTE
2D2D 60               RTS
;
; BLDHEX : BUILD HEX BYTE FROM BUFFER
; RESULT IS IN ACCUM
;
2D2E 20 3D 2D BLDHEX  JSR GETHEX      GET BYTE FROM BUFFER
2D31 0A               ASL A
2D32 0A               ASL A
2D33 0A               ASL A
2D34 0A               ASL A
2D35 85 E0            STA TS1         SAVE UPPER FOUR BITS
2D37 20 3D 2D         JSR GETHEX      GET SECOND BYTE
2D3A 05 E0            ORA TS1         COMBINE WITH FIRST BYTE
2D3C 60               RTS
```

```
;
; GETHEX : GET 1 HEX DIGIT FROM BUFFER
;
2D3D 20 E4 2C GETHEX JSR BUFBYT      GET BYTE FROM BUFFER
2D40 38              SEC
2D41 E9 30           SBC #$30
2D43 C9 0A           CMP #$0A
2D45 90 08           BCC *+10        ($2D4F) IF <10 THEN RETURN
2D47 E9 11           SBC #$11
2D49 C9 06           CMP #$06
2D4B B0 08           BCS *+10        ($2D55) IF > F THEN ERROR
2D4D 69 0A           ADC #$0A
2D4F 60              RTS
;
; THIS CODE USED BY BASIC AND POSSIBLY THE OTHER LANGUAGES
;
2D50 A5 00           LDA PAGE0       DO WE NEED TO SWAP PAGES 0/1
2D52 F0 A3           BEQ SWAP        YES, GO DO IT
2D54 60              RTS
;
2D55 4C C0 2A        JMP ERR7        GOT HERE FROM $2D4B
;
; CKEQL ; CHECK FOR '=' OR ',' OR '/'
;
; THREE ENTRY POINTS -> CKEQL=$2D58 : CKCOMA=$2D5B : CKSLSH=$2D5E
; ANOTHER EXAMPLE OF TURNING A TWO BYTE INSTRUCTION INTO A THREE
; BYTE 'HARMLESS' INSTRUCTION. (SEE NOTE @ $28E6)
; FORTUNATELY, THIS TIME THEY ARE HARMLESS.
;
2D58 A9 3D    CKEQL  LDA #'=
2D5A 2C A9 2C        BIT $2CA9
2D5D 2C A9 2F        BIT $2FA9
2D60 85 E0           STA TS1         SAVE CHARACTER TO TEST
2D62 20 E4 2C        JSR BUFBYT      GET BYTE FROM BUFFER
2D65 C5 E0           CMP TS1
2D67 D0 EC           BNE CKEQL-3     ($2D55) IF NOT THEN ERROR #7
2D69 60              RTS
;
; CRLF : PRINT CR,LF TO ALL ACTIVE DEVICES
;
2D6A A9 0D    CRLF   LDA #$0D        DO 'CR'
2D6C 20 43 23        JSR PRINT
2D6F A9 0A           LDA #$0A        DO 'LF'
2D71 D0 30           BNE FNDFL-3     ($2DA3) JUMP TO PRINT
;
; STROUT : PRINT STRING FOLLOWING JSR THAT GOT US HERE
;
; THE STRING CAN BE ANYTHING, BUT MUST BE TERMINATED BY A NULL.
; STRING LENGTH IS LIMITED TO 255 CHARACTERS.
; THIS IS A VERY USEFUL ROUTINE, BUT BE WARNED THAT IT CAN REALLY
; PLAY HAVOC WITH YOUR PROGRAM IF YOU FORGET TO PUT THE NULL
; DELIMITER ON YOUR STRING.
;
```

```
2D73 68        STROUT PLA              PULL RETURN ADDRESS OFF STACK
2D74 85 E3            STA STROAD       STORE LOW ADDRESS
2D76 68               PLA
2D77 85 E4            STA STROAD+1     STORE HIGH ADDRESS
2D79 A0 01            LDY #$01         SET TO INDEX THROUGH STRING
2D7B B1 E3            LDA (STROAD),Y   GET BYTE FROM STRING
2D7D F0 06            BEQ *+8          ($2D85) IF NULL THEN WE ARE DONE
2D7F 20 43 23         JSR PRINT        PRINT IT IF NOT
2D82 C8               INY              GET SET FOR NEXT CHARACTER
2D83 D0 F6            BNE *-8          ($2D7B) JUMP AND CONTINUE
2D85 98               TYA
2D86 38               SEC              GET SET TO FIND RETURN ADDRESS
2D87 65 E3            ADC STROAD       ADD LENGTH OF STRING TO ADDRESS
2D89 85 E3            STA STROAD       SAVE IT
2D88 90 02            BCC *+4          ($2D8F) NO CARRY SO UPPER BYTE IS O
2D8D E6 E4            INC STROAD+1     BUMP THE UPPER BYTE
2D8F 6C E3 00         JMP (STROAD)     JUMP PAST PRINTED STRING
;
; PRT2HX : PRINT 2 HEX CHARACTERS OF ACCUMULATOR
;
2D92 48        PRT2HX PHA              SAVE THE CHARACTER
2D93 4A               LSR A            PUT UPPER NIBBLE IN LOWER NIBBLE
2D94 4A               LSR A
2D95 4A               LSR A
2D96 4A               LSR A
2D97 20 9B 2D         JSR PRTHEX       PRINT THE UPPER 4 BITS
2D9A 68               PLA              RESTORE THE CHARACTER
;
; PRTHEX : PRINT HEX OF LOW NIBBLE IN ACCUMULATOR
; GOOD HEX TO ASCII CONVERSION
;
2D9B 29 0F     PRTHEX AND #$0F         MASK UPPER 4 BITS
2D9D C9 0A            CMP #$0A         SET CARRY IF >9 AND CLEAR
;                                      CARRY IF <10
2D9F F8               SED
2DA0 69 30            ADC #$30         IF CARRY SET THEN A=$41
;                                      IF CARRY CLEAR 9=$39
2DA2 D8               CLD
2DA3 4C 43 23         JMP PRINT
;
; FNDFL : FIND FILE NAME IN DIRECTORY
;
; ONE OF THE MOST USEFUL ROUTINES IN THE OS IF YOU ARE WRITTING
; YOUR OWN MACHINE LANGUAGE PROGRAMS. PUT THE NAME OF THE FILE YOU ARE
; LOOKING FOR IN A BUFFER, EITHER THE OS BUFFER @ $2E1E OR YOUR OWN.
; THE FILE NAME SHOULD BE DELIMITED BY A CR IF IT IS SHORTER THAN
; SIX CHARACTERS. IF YOU ARE USING YOUR OWN BUFFER, IT'S ADDRESS
; SHOULD BE PUT IN $E1,$E2. THE BUFFER OFFSET @ $2CE5 MUST BE
; SET, EITHER TO ZERO IF THE FILE NAME IS AT THE BEGINNING OF THE
; BUFFER, OR TO WHATEVER OFFSET IN THE BUFFER THE FIRST CHARACTER
; OF THE FILE NAME IS AT. THEN CALL THIS ROUTINE.
; RETURNS WITH STARTING TRACK IN A, LAST TRACK @ $E5.
;
```

```
2DA6 A9 76     FNDFL  LDA #$76
2DA8 85 E5            STA HSTTK        SET HIGHEST TRACK NUMBER
2DAA 20 E4 2C         JSR BUFBYT       GET BYTE FROM BUFFER
2DAD CE E5 2C         DEC BUFOFS       SET POINTER BACK
2DB0 C9 41            CMP #$41
2DB2 10 0A            BPL *+12         ($2DBE) IF ALPHA THEN LOOK FOR NAME
2DB4 A2 00            LDX #$00         HERE IF TRACK NUMBER ENTERED
2DB6 A9 76            LDA #$76
2DB8 9D 7A 2E         STA SCRBUF+1,X
2DBB 4C 2E 2D         JMP BLDHEX       GET TRACK# IN A AND RETURN
;
; LOAD BY NAME
;
2DBE AD E5 2C         LDA BUFOFS       GET POINTER
2DC1 85 E0            STA TS1          SAVE IT
2DC3 A9 08            LDA #$08         TRACK NUMBER FOR DIRECTORY
2DC5 20 BC 26         JSR SETTK        MOVE TO TRACK 8
2DC8 8C 5E 26         STY SECTNM       SECTNM=0
2DCB 4C 00 2E         JMP NEWDS        JUMP AROUND A LITTLE
2DCE A2 00            LDX #$00
2DD0 20 E4 2C NXTCHR  JSR BUFBYT       GET BYTE FROM BUFFER
2DD3 C9 0D            CMP #$0D
2DD5 D0 02            BNE *+4          ($2DD9) IF NOT 'CR' THEN CONTINUE
2DD7 A9 20            LDA #$20         IF 'CR' USE 'SPACE' FOR COMPARE
2DD9 DD 79 2E         CMP SCRBUF,X     COMPARE TO DIRECTORY ENTRY
2DDC D0 11            BNE NXTDE        IF NOT = TRY NEXT ENTRY
2DDE E8               INX              BUMP OFFSET
2DDF 8A               TXA
2DE0 29 07            AND #$07
2DE2 C9 06            CMP #$06
2DE4 D0 EA            BNE NXTCHR       IF WE HAVEN'T LOOKED AT ALL
;                                      6 CHARACTERS THEN CONTINUE
2DE6 BD 7A 2E         LDA SCRBUF+1,X   MATCH! GET LAST TRACK#
2DE9 85 E5            STA HSTTK        SAVE IT
2DEB BD 79 2E         LDA SCRBUF,X     GET STARTING TRACK#
2DEE 60               RTS
2DEF 8A        NXTDE  TXA              NEXT DIR ENTRY SETUP
2DF0 29 F8            AND #$F8         KILL LOWER 3 BITS IN OFFSET
2DF2 18               CLC
2DF3 69 08            ADC #$08         SET TO NEXT DIR ENTRY
2DF5 AA               TAX
2DF6 F0 08            BEQ NEWDS        IF END OF SECTOR GET NEXT
2DF8 A5 E0     NXTDS  LDA TS1          RESTORE BUFFER OFFSET
2DFA 8D E5 2C         STA BUFOFS
2DFD 4C D0 2D         JMP NXTCHR       JUMP BACK AND TRY AGAIN
2E00 EE 5E 26  NEWDS  INC SECTNM       BUMP SECTOR NUMBER
2E03 AD 5E 26         LDA SECTNM
2E06 C9 03            CMP #$03
2E08 30 05            BMI *+7          ($2E0F) IF < 3 THEN CONTINUE
2E0A A9 0C     ERRC   LDA #$0C         ERROR C, FILE NOT FOUND
2E0C 4C 4B 2A         JMP ERRENT
2E0F A9 79            LDA #$79         SET MEMORY ADDRESS TO $2E79
2E11 85 FE            STA MEMLO        (SCRATCH BUFFER, 256 BYTES)
```

```
2E13 A9 2E          LDA #$2E
2E15 85 FF          STA MEMHI
2E17 20 1A 2B       JSR CALL+9      ($2B1A) LOAD HEAD, READ DISK,
;                                   UNLOAD HEAD
2E1A A2 00          LDX #$00        BUFFER OFFSET
2E1C F0 DA          BEQ NXTDS       SEARCH THIS DIRECTORY SECTOR
```

```
; TABLES AND STORAGE FOR OS65D
;
; OS65D INPUT BUFFER @$2E1E TO $2E2F
;
2E1E            OSBUF   = *+17
;
; OS65D DISPATCH TABLE
;
; ADDRESS IN TABLE = ACTUAL ADDRESS OF ROUTINE - 1
; ADDRESS IN TABLE IS PUSHED ON STACK AND THEN CALLED
; BY DOING AN RTS.
;
2E30 41 53      DSPTBL .BYTE 'AS'
2E32 DD 2A             .WORD ASM-1
2E34 42 41             .BYTE 'BA'
2E36 E5 2A             .WORD BASIC-1
2E38 43 41             .BYTE 'CA'
2E3A 10 2B             .WORD CALL-1
2E3C 44 39             .BYTE 'D9'
2E3E 22 2B             .WORD D9-1
2E40 44 49             .BYTE 'DI'
2E42 28 2B             .WORD DIR-1
2E44 45 4D             .BYTE 'EM'
2E46 2E 2B             .WORD EM-1
2E48 45 58             .BYTE 'EX'
2E4A 36 2B             .WORD EXAM-1
2E4C 47 4F             .BYTE 'GO'
2E4E 45 2B             .WORD GO-1
2E50 48 4F             .BYTE 'HO'
2E52 62 26             .WORD HOME-1
2E54 49 4E             .BYTE 'IN'
2E56 54 2B             .WORD INIT-1
2E58 49 4F             .BYTE 'IO'
2E5A 82 2B             .WORD IO-1
2E5C 4C 4F             .BYTE 'LO'
2E5E A6 2B             .WORD LOAD-1
2E60 4D 45             .BYTE 'ME'
2E62 C5 2B             .WORD MEM-1
2E64 50 55             .BYTE 'PU'
2E66 DC 2B             .WORD PUT-1
2E68 52 45             .BYTE 'RE'
2E6A FC 2B             .WORD RET-1
2E6C 58 51             .BYTE 'XQ'
2E6E 21 2C             .WORD XQT-1
2E70 53 41             .BYTE 'SA'
2E72 27 2C             .WORD SAVE-1
2E74 53 45             .BYTE 'SE'
2E76 42 2C             .WORD SELECT-1
2E78 00                .BYTE 0
;
; THE REST OF THE OS MEMORY AREA IS WORKING STORAGE LOCATIONS
;
2E79            SCRBUF = *+256          SCRATCH BUFFER FOR DIRECTORY
```

```
;
; THIS AREA IS ALSO USED BY THE BASIC GET/PUT LOGIC.
; YOU CAN USE THIS PAGE FOR TRANSIENT CODE BY CALLING IT HERE.
; JUST BE SURE THAT YOU DON'T DO A DIRECTORY SEARCH OR USE BASIC'S
; RANDOM DISK I/O. A GOOD PLACE TO PUT SUCH CODE IS ON TRACK 8
; IN SECTORS 6 & UP SINCE THIS AREA IS NOT USED FOR ANY OTHER
; PURPOSE.
;
2F79            SWAPØ   = *+256         PAGE Ø HOLD AREA (USED BY BASIC)
3Ø79            SWAP1   = *+256         STACK    "   "       "   "   "
;
; AND THAT (FINALLY!) BRINGS US UP TO THE START OF THE BASIC WORKSPACE.
```

```
D1OADR  *23C3  23DC  23E7

D2BFDR  *2335  241A  2420

D2BFHI  *2330

D2BFLO  *232E  242A  2435

D2CRTK  *2334

D2FRST  *2332

D2IADR  *23FD  242D  2438

D2LAST  *2333

D2OADR  *2416  2430  243B

D9      *2B23  2B09  2E3E

DCPRNT  *2A41  2A2F  2A34

DEFDEV  *2AC6  228B  2569

DELAY   *2700  267C  269A  2704

DIR     *2B29  2E42

DIRCNT  *29F3  2B2C

DK1IN   *23A1  230B  23B4

DK1NXT  *23CC  23A8  23BE

DK1OUT  *23B2  231B

DK2IN   *23F0  230D

DK2NXT  *2420  23F9  2411

DK2OUT  *2403  231D

DKBT9   *28B0  2885  28B9  28CB  28D2  28D9  28ED  290B  2912  291D  29C3
         2B25

DKBTCI  *29B6  2998  29A1  29AC  29BF

DKINIT  *29EB  29CE  29D4  29E1  29F4

DKRDRY  *2977  2974  297D

DKWTX   *27C2  27A6  27AB  27B1  27B6  27C7  282D  2833  283B  2843  2851
         2856
```

```
DLYFA   *289F  2810  2828
DLYFA1  *2956  289F
DOINP   *2339  2CD9
DOIO    *234B  233E  235C  2374
DONXIO  *235C  2351
DSKBYT  *27CD  27D1  27F5
DSKDR   *265C  26D1  2A5E
DSKWRT  *27E1  248E  2803  2BED  2C3D
DSPTBL  *2E30  2A8B  2A95  2A9C  2AA0
EM      *2B2F  2E46
ERR1    *2993
ERR2    *289B  288D
ERR3    *2784
ERR4    *278F  2786  278D  27EA
ERR5    *28E0
ERR6    *26DC *2C5B  26CF  2C48  2C4C  2C53  2C96
ERR7    *2AC0  2A8E  2C1F  2C58  2D55
ERR8    *26CD  26C2  26C6
ERR9    *28BF  289D  28B3  28E9
ERRA    *28E6
ERRB    *27E8  27E4  27EE
ERRC    *2E0A
ERRD    *2C94
ERRENT  *2A4B  26DE  2791  28C1  2995  2A7D  2A80  2AC2  2C5D  2E0C
EXAM    *2B37  2E4A
EXAMCN  *2739  2745  274D  2751  2B43
EXCOM   *2A84  2A77  2ABE
```

```
EXIT    *25C9  25BB  25BF  25FA  2601  2609  263A

FLOPIN  *C000  2235  223A  223D  266E  271D  2722  273C  277F  278A  27B9
         27F2  27FC  28B0  29BC  29D1  29DA

FLOPOT  *C002  2683  268C  268F  270A  2716  2719  2756  2759  2763  2799
         279C  2815  2818  2822  2825  2866  286B  29D7

FNDFL   *2DA6  2BA7  2BDD  2D71

FULINT  *2B68  2B5A

GETADR  *2D23  2B11  2B37  2C2E  2CFB  2D01

GETHEX  *2D3D  2C34  2C69  2D2E  2D37

GETTK   *2C60  2B17  2C28

GO      *2B46  2E4E

GOADR   *2B52  2B49  2B4F

GOBAS   *22B3  2294

HAMB    *2661  27DC  2962

HIMEM   *2300  2280

HMFND   *2280  227B

HOLOOP  *266C  2676

HOME    *2663  276C  294D  2C55  2E52

HSTTK   *00E5  276A  2AFF  2C8B  2DA8  2DE9

HUNDUS  *28A2  281D  2863  2870  28AA  28AD

IN1     *2336  2340

INCTKN  *2C83  245E  276F  2B0B  2BB1  2BF4

INDST   *2321  233B  256C  257F  2AC7  2B90

INECHO  *2340  2B79  2CA2

INIT    *2B55  2E56

INITAL  *2768  277A  2B80

INITTK  *277D  2772  2B65

INPUT   *2CD6  2336  2CDF
```

```
IO       *2B83  2E5A
IODISP   *2376  2356
IOOFS    *2378  2371
IOTABL   *2301  237A  237E
KBINP    *252B  2303  2534  25EB
KEYTST   *263D  25D8  25E2
KIRTN    *253F  2528
KPDO     *2325  24E3  2540
KPOLL    *FD00  2531
KPORT    *DF00  2221  263D  2640
KTRTN    *2643  25DE  25E5
LAMB     *2660  27D7  295D
LCHAR    *263C  259B  25CC
LDCMN    *2AEE  2AE0  2AE8  2B31
LDHDWI   *2728  2739  2794
LDHEAD   *2754  220C  2494  2728  2766  2A0D  2B01  2B1A  2C3A  2C80
LF       *260F  25AB
LOAD     *2BA7  2B7E  2B98  2C22  2E5E
MAXBUF   *2CED  2C9D
MEM      *2BC6  2E62
MEMCHK   *22EC  2278  229D
MEMHI    *00FF  220A  2262  226F  22EC  23EA  243E  2489  274F  27DF  2849
          287A  2880  2890  2896  2940  2965  2977  2AF5  2C75  2D26  2E15
MEMIN    *2389  2309
MEMLO    *00FE  220F  2266  226A  22EE  22F4  22F8  23DF  2433  2484  274A
          27DA  2840  2937  293B  2960  2AFC  2C79  2D2B  2E11
MEMOT    *2390  2319
MEMTST   *2276  227E
```

```
MINADR  *238A  2398  239D  2499  2585  2BD3  2BD9
MODMIN  *2497  2582
MOTADR  *2391  2553  2558
MOVE    *2629  2632  2636
MOVEHD  *26D1  26CB  2950
NEWDS   *2E00  2DCB  2DF6
NMHZ    *267B  28A2
NULLIN  *2386  2307
NXTCHR  *2DD0  2DE4  2DFD
NXTDE   *2DEF  2DDC
NXTDS   *2DF8  2E1C
NXTENT  *2ABA  2A92  2A9A  2AA9
NXTOSI  *2CA2  2CB9  2CCA
NXTOSN  *2C98  2CAA  2CBE
ONLYO   *2BA0  2B88
OS65D3  *2A51  254A  2A7A
OSBUF   *2E1E  2CAC  2CC0
OSERR   *2AC4  2A4B
OSIBAD  *00E1  2A71  2A75  2A90  2A98  2CE6
OSINP   *2C9B  2A6C
OSIOK   *2CBC  2CA7  2CCE
OUTDST  *2322  2346  255B  256F  2574  258E  2593  2ACA  2BA3
PAGE0   *0000  2274  28A5  2D13  2D1B  2D50
PATCH0  *22B6  2202
PATCH1  *2371  234D
PATCH2  *2491  2462
PATCH3  *2508  24B9
```

```
PATCH4  *2540  25F2
PGCNT   *265F  247E  27E1  2836  2859  287D  2893  2921  297A  2BEA  2C37
PLINE   *D700  25C1  25C9  25D1  25F6  25FD  2604  2610  261A
PRINT   *2343  2566  2A41  2A64  2A69  2D6C  2D7F  2DA3
PRT2HX  *2D92  2A07  2A48
PRTHEX  *2D9B  2AD8  2D97
PTROUT  *249F  2317  24A4
PTRPIA  *F400  2217  221A  221D  2224  222F  2232  24A0  24A9  24AC
PUT     *2BDD  2E66
RDCDSK  *2907  2882  28F5  2971
RDCONT  *2929  293E  2943
RDRTYM  *00F7  2969  298F
RDRTYN  *00F8  280B  2887  296D  297F
READDK  *2967  2211  2991  2B04  2B1D  2BB4
RET     *2BFD  2E6A
REWRT   *2809  2898
RNDSED  *2324  24F9  252E
RSACIA  *272B  28C8
RSTAXY  *235E  2CE1
RTASM   *1303  2C04
RTBAS   *20C4  2C0B
RTYCMP  *2878  2889
RTYRD   *296F  2981
SAVAXY  *2367  2343  2CD6
SAVE    *2C28  2E72
SCRBUF  *2E79  2DB8  2DD9  2DE6  2DEB
SCROLL  *260B  25C7
```

```
SCTBYP  *00F9  28EB  28FE  29A9  2A19  2A37

SCTLEN  *00FA  27F8  2956  2A21

SCTNUM  *00FB  2902  29A4  2A1E

SCTRTY  *00F5  28C6  28E2

SECTNM  *265E  22B6  2491  2830  28F0  2915  2A11  2AF8  2C6C  2C7D  2DC8
         2E00  2E03

SEEKRT  *2947  28E4

SELECT  *2C43  2E76

SERINP  *2518  2305  251C

SEROUT  *250D  2315  2511

SETDRV  *29C6  2242  29BA  2C50

SETERR  *2A7D  2A58

SETFF   *291B  2933  2939

SETFLO  *2719  2697  269F  2712  2875

SETNXT  *2622  2630

SETPGM  *2C70  2BAA  2BE0

SETSCT  *28C4  280D  2908  2953  2A14

SETTK   *26BC  2205  29F7  2AEE  2B40  2B62  2C63  2C70  2C91  2DC5

SSOK    *28EB  28DE

STACK   *0100  2A44  2D06  2D10

STASM   *1300  2AE3

STBAS   *20E4  2AE8

STCCNT  *2708  26E6

STEM    *1700  2B34  2C12

STEP    *268F  2688

STEPIN  *2683  2673  2692  26EA  2983

STEPOT  *268A  2663  26F1  2989
```

```
STEPRT  *00EF  22BB  26A2  2947  294B
STKADR  *00FC  2A0B  2A26  2A3B
STROAD  *00E3  2D74  2D77  2D7B  2D87  2D89  2D8D  2D8F
STROUT  *2D73  29FA  2ACD  2B68  2CAF
SWAP    *2CF7  2D52
SWAP0   *2F79  2D15  2D18
SWAP1   *3079  2D09  2D0C
SWAP4   *2644  252B  2539  2547  2654  2C19
SWAP4A  *0213  2646  2650
SWAP4B  *2657  2649  264C
SWAPLP  *2D06  2D1E
TENMS   *2678  2666  2671  2680  26A4  275E  27A1  2986  298C
TERMAC  *FC00  2247  224C  24CE  24DA  24F6
TERMIN  *24F6  2301  24E0  24EA  24FD
TERMIO  *FC01  24D6  24FF
TERMOT  *24CD  2311  24D3
TIRTN   *2504  250B  2544
TKNHLD  *00EE  26B3  26BA  26BC  26E1  28DC
TKNUM   *265D  2669  26E3  26F6  26F9  2775  27AE  29C6  2C83
TORTN   *24F1  24DE  24E8
TRKNM   *2662  26A6
TS1     *00E0  2AF3  2B07  2BAD  2BB7  2BBE  2BE6  2BF0  2D35  2D3A  2D60
         2D65  2DC1  2DF8
TS2     *00FD  22F6  22FC  2839  284B  285D  2860
UART    *FB00  2227  222A  250E  2514  2518  251E  2521
UNLDHD  *2761  2214  273F  27C0  2A3E  2ADB  2B20  2BC3  2BFA  2C40
VIDOUT  *2599  22AD  2313
```

```
VIDSIZ  *DE00  2291
VLOSAV  *2639  2613
VLP1    *262B  2623
VLP2    *262E  2626
VOTOFS  *25A4  25D4
WAITIH  *271D  2720  272B
WRTPG   *2840  2847  284D
WRTRTY  *00F6  2807  288B
WTDKBF  *2477  23D3  2427
X.HOLD  *235F  236D
X16ACI  *CF00  2251  2255  24B3  24C1  24C9  2508
X16DEV  *2323  24B0  24BE
X16INP  *24B0  230F  24B7
X16OUT  *24BD  231F  24C6
XQT     *2C22  2E6E
Y.HOLD  *2361  236A
```

ASCII Key Board

Does NOT leave value in "A"

```
2528 2@4425 JSR $2544    delay
252B EE2423 INC $2324    rand seed
252E AD@1DF LDA $DF@1
2531 3@F5   BMI $2528
2533 8D6323 STA $2363
2536 EA     NOP          ok pha
2537 EA     NOP
2538 EA     NOP
2539 EA     NOP
253A EA     NOP
253B 2@4425 JSR $2544    delay
253E AD@1DF LDA $DF@1    wait key-up
2541 1@F8   BPL $253B    ok pla
2543 6@     RTS
2544 A276   LDX *$76     delay
2546 CA     DEX
2547 2@4C25 JSR $254C
254A D@FA   BNE $2546
254C 6@     RTS
254D C95B   CMP *$5B     "[" start indirect file
254F D@11   BNE $2562
2551 A98@   LDA *$8@
2553 8D9223 STA $2392

2556 A9@@   LDA *$@@
2558 8D9123 STA $2391
255B AD2223 LDA $2322
255E @91@   ORA *$1@
256@ D@31   BNE $2593    "J" close file
2562 C95D   CMP *$5D
2564 D@13   BNE $2579
2566 2@4623 JSR $2346
2569 ADC62A LDA $2AC6
256C 8D2123 STA $2321
256F AD2223 LDA $2322
2572 29EF   AND *$EF
2574 8D2223 STA $2322
2577 A95D   LDA *$5D     control X ? Load file
2579 C918   CMP *$18
257B D@@D   BNE $258A
257D A91@   LDA *$1@
257F 8D2123 STA $2321
2582 2@9724 JSR $2497
2585 8D8A23 STA $238A
2588 B@@C   BCS $2596
2
```

```
2599 98       TYA
259A 48       PHA
259B AC5B26   LDY $265B      under cursor
259E AD6323   LDA $2363      A. Hold
25A1 297F     AND *$7F       strip bit 7  — EA EA
25A3 A247     LDX *$52
25A5 C90D     CMP *$0D       cr
25A7 F078     BEQ $2621
25A9 C908     CMP *$08       back up
25AB F066     BEQ $2613
25AD C910     CMP *$10       contrl P - printer on/off
25AF F069     BEQ $261A
25B1 C90C     CMP *$0C       control L
25B3 F065     BEQ $261A
25B5 C90A     CMP *$0A       LF
25B7 F074     BEQ $262D
25B9 C920     CMP *$20       < space   EA
25BB 301C     BMI $25D9      exit      EA
25BD C97B     CMP *$7B       > {       EA
25BF 1018     BPL $25D9      exit      EA
25C1 9D00D7   STA $D700,X    to screen
25C4 E8       INX
25C5 E080     CPX *$80

25C7 F060     BEQ $2629      scroll
25C9 D000     BNE $25CB
25CB BC00D7   LDY $D700,X
25CE 8C5B26   STY $265B      under cursor
25D1 A95F     LDA *$5F       cursor
25D3 9D00D7   STA $D700,X
25D6 8EA425   STX $25A4
25D9 68       PLA
25DA A8       TAY
25DB AD6323   LDA $2363      A. Hold
25DE 48       PHA
25DF AD01DF   LDA $DF01
25E2 3011     BMI $25F5
25E4 8D2523   STA $2325      — save keypressed for cc check $0819
25E7 C913     CMP *$13       control S
25E9 D00A     BNE $25F5
25EB 204425   JSR $2544      delay
25EE AD01DF   LDA $DF01
25F1 C911     CMP *$11
25F3 D0F9     BNE $25EE
25F5 4CF124   JMP $24F1      Restore A + RTS
25F8 60       RTS
25F9 8A       TXA
```