```
;               A P E X  V1.0
;          **** RESIDENT CODE ****

;SYSTEM CONFIGURATION DEFINITIONS
        .DEF    SWPSIZ=$40      ;DISK SWAP SIZE (16K)
        .DEF    SWPLOC=$6000    ;MEMORY APEX WILL SWAP
        .DEF    DIRBLK=9        ;DIRECTORY BEGINS HERE
        .DEF    MAXFL=48        ;MAX LEGAL NUMBER OF FILES
        .DEF    COMPAG=$B700    ;COMMUNICATIONS PAGE
        .DEF    INPBUF=$A000    ;SCRATCH BUFFER

;DEFINITIONS OF APPLE ROM ADDRESSES:
        .DEF    MONITR=$FF59    ;ROM MONITOR "RESET" ENTRY

;SOME PAGE ZERO POINTERS FOR ANYBODY TO USE:
        .DEF    PNTR1=0
        .DEF    PNTR2=2
        .DEF    PNTR3=4

;SOME HARDWARE LOCATIONS IN THE APPLE
        .DEF    ASPEAK=$C030    ;THE SPEAKER


;WHERE THIS MODULE GOES:
        .DEF    HIAD=$B000      ;ACTUALLY GOES HERE
        .DEF    LOAD=$2000      ;BUT IT LOADS HERE
        .DEF    BIAS=HIAD-LOAD
```

*** ASM65 *** (V3.0-DS) OCT-79

```
                        .PAGE
                ;       APEX BASIC SYSTEM FUNCTIONS
                ;       **************************

                        .DEF    HERE=$B000
                        .LOC    HERE,HERE-BIAS

                ;"RELOAD" THE APEX SYSTEM FROM THE SYSTEM UNIT
                ;THAT IS, LOAD THE WHOLE THING NO MATTER WHAT

B000 A9 FF8D    RELOAD: MOV#    $FF,EXECUT      ;EXEC MODE OFF
B005 A9 FC              LDA#    $FC             ;INDICATE A RELOAD
B007 8D 50BF    REL1:   STA     SYSENF          ;SAVE THE INDICATOR
B00A D8                 CLD                     ;SECURITY
B00B 78                 SEI                     ;EQUALLY
B00C A2 FF              LDX#    $FF             ;RESET STACK POINTER
B00E 9A                 TXS
B00F AD 52BF            MOV     SYSDEV,UNIT     ;TRANSFER FROM SYSTEM UNIT
B015 A9 3185            DMOV#   SYSFIL,PNTR3    ;POINT TO SYSTEM FILE NAME
B01D 20 76B0            JSR     FSCANR          ;GO FIND THE EXEC CODE.
B020 B0 25              BCS     BARF            ;HEY, NONE HERE!
B022 AD 69BF            DMOV    BLKNO,SYSBLK    ;FOUND, SO FORM THIS BLOCK
B02E 4C B2B0            JMP     BOOT            ;WE GO DO A SYSTEM RELOAD

B031 53 5953    SYSFIL: .ASCII  "SYSTEM  SYS"   ;EXEC FILE NAME
B03C 53 4352    SWPFIL: .ASCII  "SCRATCH SYS"   ;SWAP SPACE FILE NAME

B047 A9 008D    BARF:   MOV#    0,SYSDEV        ;IF TROUBLE, GO BACK TO UNIT 0
B04C A9 53              LDA#    <SERMES         ;SAY WE HAD A BAD SYSTEM UNIT
B04E A2 B0              LDX#    >SERMES
B050 4C FFB1            JMP     HRDERR          ;ITS A HARD ERROR

B053 0D         SERMES: .BYTE   %15
B054 0A                 .BYTE   %12
B055 4E 4F54            .ASCII  "NOT A SYSTEM DISK-"
B067 52 4542            .ASCII  "REBOOT ON 0?"
B073 0D                 .BYTE   %15
B074 0A                 .BYTE   %12
B075 00                 .BYTE   0

                ;LINK TO THE SCAN FUNCTION:
B076 AD 3ABF    FSCANR: DPSH    INBUFD          ;SAVE THE INPUT BUFFER SPEC
B07E A9 008D            DMOV#   INPBUF,INBUFD   ;SET IT UP OUT WAY
B088 20 44B2            JSR     FSCAN1          ;GO FIND THE FILE
B08B 68 8D3B            DPOP    INBUFD          ;AND RESTORE HIS BUFFER SPEC
B093 60                 RTS

                ;REENTER THE APEX EXEC, RELOAD IF NECCESSARY

B094 A9 FD      RENTER: LDA#    $FD             ;INDICATE NON-VALID SWAP AREA
B096 8D 50BF    ENTER:  STA     SYSENF          ;SAVE REENTRY STATUS
B099 D8                 CLD                     ;FOR SECURITY
B09A 78                 SEI
```

```
B09B A2 FF              LDX#    $FF             ;COS IT MUST NOT BE LOW!
B09D 9A                 TXS
B09E AD 28BF            LDA     SYBOMB          ;CHECK RESIDENCY FLAG
B0A1 F0 07              BEQ     TRYSWP          ;=0 IF USER DID NOT BOMB APEX
B0A3 C9 55      ENTR2:  CMP#    $55             ;=$55 IF THE PROGRAM IS APEX
B0A5 D0 0B              BNE     BOOT            ;NEITHER, SO RELOAD APEX
B0A7 4C 03BF            JMP     VSTART          ;IS APEX, SO JUST RESTART IT
B0AA 20 FEB0    TRYSWP: JSR     MOVMEM          ;SWAP BACK SYSTEM PAGE
B0AD AD 28BF            LDA     SYBOMB          ;CHECK THE NEW RESIDENCY FLAG
B0B0 D0 F1              BNE     ENTR2           ;IF WE CAN, REPEAT THE ABOVE
                ;RESET AND BOOT:
B0B2 20 DFBF    BOOT:   JSR     KRESTD          ;RESET DISK DRIVES
B0B5 20 EDB2            JSR     OPCON           ;RESET CONSOLE
                ;CONTINUE WITH BOOT
B0B8 AD 52BF            MOV     SYSDEV,UNIT     ;BOOT IN FROM THE DEFAULT UNIT
B0BE AD 53BF            DMOV    SYSBLK,BLKNO    ;SETUP BLOCK NUMBER

                ;"RUN" THE MEMORY IMAGE WHICH BEGINS AT 'BLKNO'

B0CA 20 DBB0    RUN:    JSR     GETR            ;GET IT
B0CD A9 008D            MOV#    0,RERUNF        ;NOT RUN YET
B0D2 4C 03BF            JMP     VSTART          ;AND ENTER AT IT'S STARTING ADDR

                ;"GET" THE MEMORY IMAGE WHICH BEGINS AT 'BLKNO'

B0D5 20 DBB0    GET:    JSR     GETR            ;GET IT
B0D8 4C 59FF            JMP     MONITR          ;BUT DON'T START IT

                ;THIS ROUTINE READS IN THE MEMORY IMAGE AT 'BLKNO'

B0DB 20 C4B1    GETR:   JSR     SET1            ;SET UP FOR COMMUNICATION AREA
B0DE 20 EFB1            JSR     READUS          ;READ IT
B0E1 20 FEB0            JSR     MOVMEM          ;MOVE ABOUT
B0E4 20 D4B1            JSR     SET3            ;SET UP FOR MAIN MEMORY
B0E7 20 EFB1            JSR     READUS          ;READ IT
B0EA 60                 RTS

                ;THIS ROUTINE WRITES THE CURRENT IMAGE BEGINNING AT 'BLKNO'

B0EB 20 C4B1    SAVE:   JSR     SET1            ;SET UP FOR COMMUNICATION AREA
B0EE 20 FEB0            JSR     MOVMEM          ;SHUFFLE MEMORY
B0F1 20 F5B1            JSR     WRITUS          ;WRITE IT
B0F4 20 FEB0            JSR     MOVMEM          ;SHUFFLE BACK
B0F7 20 D4B1            JSR     SET3            ;SET UP FOR MAIN MEMORY
B0FA 20 F5B1            JSR     WRITUS          ;WRITE IT
B0FD 60                 RTS

                ;THIS ROUTINE PACKS THE PROGRAM SPECIFIC AREAS OF PAGES $0 & $BF
                ;INTO THE "COMMUNICATIONS" PAGE.  ACTUALLY IT SWAPS THEM

B0FE A9 0085    MOVMEM: DMOV#   SYSPAG,PNTR1    ;FIRST DO LOW PART
B106 A9 0085            DMOV#   COMPAG,PNTR2    ;OF PAGE $BF
B10E A0 00              LDY#    0               ;FROM LOCATION 0
B110 A2 50              LDX#    $50             ;FOR $50 BYTES
B112 20 1BB1            JSR     MOVSWP          ;DO THE SHUFFLE
B115 A9 0085            MOV#    0,PNTR1+1       ;THEN HIGH PART
```

```
B119 A2 B0              LDX#    256-$50         ;OF PAGE ZERO
B11B B1 00      MOVSWP: LDA@Y   PNTR1           ;\
B11D 48                 PHA                     ; :
B11E B1 02              LDA@Y   PNTR2           ; : EXCHANGE THE BYTES
B120 91 00              STA@Y   PNTR1           ; :
B122 68                 PLA                     ; :
B123 91 02              STA@Y   PNTR2           ;/
B125 C8                 INY                     ;POINT TO NEXT ONE
B126 CA                 DEX                     ;COUNT THIS ONE
B127 D0 F2              BNE     MOVSWP          ;CONTINUE IF NOT DONE
B129 60                 RTS

                ;READ IN APEX EXEC BUT WRITE OUT CURRENT IMAGE FIRST.

B12A AD 52BF    SAVER:  MOV     SYSDEV,UNIT     ;GET SYSTEM UNIT NUMBER
B130 A9 3C85            DMOV#   SWPFIL,PNTR3    ;POINT TO SCRATCH FILE NAME
B138 20 76B0            JSR     FSCANR          ;GO LOOK FOR IT
B13B 90 03              BCC     NOBARF          ;DID WE FIND IT?
B13D 4C 47B0            JMP     BARF            ;HEY, WE DONT HAVE ONE!
B140 AD 69BF    NOBARF: DMOV    BLKNO,SWPBLK    ;SAVE BLOCK NUMBER FOR LATER
B14C A9 008D            DMOV#   SWPLOC,DSKMEM   ;SETUP SWAP PLACE
B156 A9 408D            MOV#    SWPSIZ,DSKSIZ   ;SETUP LENGTH
B15B 20 EBB0            JSR     SAVE            ;SAVE IT
B15E A9 FE              LDA#    $FE             ;INDICATE VALID SCRATCH AREA
B160 4C 07B0            JMP     REL1            ;THEN BOOT IN THE EXEC

                ;SAVE CURRENT IMAGE THAT IS WRITTEN IN SCRATCH AREA

B163 AD 68BF    SAVESC: LDA     UNIT
B166 48                 PHA                     ;REMEMBER UNIT NUMBER
B167 AD 69BF            DPSH    BLKNO           ;REMEMBER PLACE TO SAVE IT
B16F AD 52BF            MOV     SYSDEV,UNIT     ;SETUP POINTERS TO SCRATCH AREA
B175 AD 55BF            DMOV    SWPBLK,BLKNO
B181 20 DBB0            JSR     GETR            ;GET THE IMAGE INTO MEMORY
B184 68 8D6A            DPOP    BLKNO           ;GET PLACE TO PUT IT BACK AGAIN
B18C 68                 PLA
B18D 8D 68BF            STA     UNIT            ;REMEMBER UNIT
B190 AD 17BF            MOV     PROSIZ,DSKSIZ   ;SETUP THE PROGRAM SIZE
B196 AD 15BF            DMOV    USRMEM,DSKMEM   ;SET ADDRESS OF PROGRAM
B1A2 20 EBB0            JSR     SAVE            ;SAVE IT
B1A5 A9 FF              LDA#    $FF             ;INDICATE WE SAVED AN IMAGE
B1A7 4C 96B0            JMP     ENTER           ;AND ENTER THE EXEC

                ;HERE TO SAVE THE PROGRAM CURRENTLY IN MEMORY

B1AA AD 15BF    SAVEA:  DMOV    USRMEM,DSKMEM   ;SET THE ADDRESS UP
B1B6 AD 17BF            MOV     PROSIZ,DSKSIZ   ;SET THE SIZE UP
B1BC 20 EBB0            JSR     SAVE            ;DO IT
B1BF A9 F0              LDA#    $F0             ;INDICATE WE DID IT
B1C1 4C 96B0            JMP     ENTER           ;AND ENTER THE EXEC

                ;ROUTINE TO SETUP TO TRANSFER COMMUNICATION AREA

B1C4 A9 008D    SET1:   DMOV#   COMPAG,FADDR    ;SETUP ADDR
B1CE A9 018D            MOV#    1,NBLKS         ;ONLY ONE BLOCK
B1D3 60                 RTS                     ;AND RETURN
```

```
                        ;ROUTINE TO SETUP TO TRANSFER MAIN MEMORY SEGMENT

B1D4 EE 69BF    SET3:   DINC    BLKNO           ;POINT TO NEXT BLOCK ON UNIT
B1DC AD 10BF            DMOV    DSKMEM,FADDR    ;SETUP INITIAL ADDR POINTER
B1E8 AD 12BF            MOV     DSKSIZ,NBLKS    ;GET SIZE OF TRANSFER
B1EE 60                 RTS                     ;AND RETURN

                        ;ROUTINE TO READ SEQUENTIAL LOGICAL BLOCKS FROM UNIT.

B1EF 20 E2BF    READUS: JSR     KREAD           ;READ THE BLOCKS
B1F2 B0 07              BCS     WRITER          ;BRANCH IF HARD ERROR
B1F4 60                 RTS                     ;RETURN IF NO ERRORS

                        ;ROUTINE TO WRITE SEQUENTIAL BLOCKS TO UNIT.

B1F5 20 E5BF    WRITUS: JSR     KWRITE          ;WRITE THE BLOCK
B1F8 B0 01              BCS     WRITER          ;BRANCH IF HARD ERROR
B1FA 60                 RTS                     ;RETURN IF NO ERRORS

B1FB A9 21      WRITER: LDA#    <DERMES         ;POINT TO ERROR MESSAGE
B1FD A2 B2              LDX#    >DERMES         ;AND FALL CEASAR

                        ;HERE WITH A,X POINTING TO APPROPRIATE ERROR MESSAGE.
                        ;ISSUE THE ERROR MESSAGE ON CONSOLE UNIT AND EXIT
                        ;DIRECTLY TO THE SYSTEM MONITOR.

B1FF 85 00      HRDERR: STA     PNTR1           ;SAVE ADDR OF MESSAGE
B201 86 01              STX     PNTR1+1         ;INTO A SCRATCH POINTER
B203 20 EDB2            JSR     OPCON           ;RESET CONSOLE OUTPUT SIDE
B206 A0 00              LDY#    0               ;POINT TO FIRST CHAR OF MESSAGE
B208 8C 20B2            STY     LTEMP
B20B AC 20B2    HMLOOP: LDY     LTEMP           ;GET POINTER TO NEXT CHAR
B20E B1 00              LDA@Y   PNTR1           ;GET CHAR
B210 F0 08              BEQ     HMDONE          ;BRANCH IF END OF MESSAGE
B212 20 FAB2            JSR     CONOUT          ;OUTPUT TO CONSOLE
B215 EE 20B2            INC     LTEMP           ;POINT TO NEXT CHAR
B218 D0 F1              BNE     HMLOOP          ;AND OUTPUT IT
B21A 20 04B3    HMDONE: JSR     CONIN           ;WAIT FOR A KEY STRIKE
B21D 4C 00B0            JMP     RELOAD          ;AND TRY A RELOAD

B220 00         LTEMP:  .BYTE   0

B221 0D         DERMES: .BYTE   %15             ;ADVANCE A LINE
B222 0A                 .BYTE   %12
B223 44 4953            .ASCII  "DISK ERROR-"
B22E 52 4554            .ASCII  "RETURN TO RESET"
B23D 0D                 .BYTE   %15
B23E 0A                 .BYTE   %12
B23F 00                 .BYTE   0

                        ;CODE TO LOOK UP A FILE IN THE DIRECTORY. THIS ROUTINE TAKES A
                        ;POINTER TO A FILE NAME BY ADDRESS IN A,Y. THE NAME MUST BE 11
                        ;CHARATERS LONG FILLED WITH BLANKS. NO FUZZY FILE NAMES PLEASE.
                        ;THE FILE NAMES ARE 11 BYTES EACH (8-NAME,3-EXT). THE STATUS
                        ;ARRAY OF ONE BYTE PER FILE BEGINS AT +528. STATUS=0 IF NULL,
```

```
                        ;<128 IF CLOSED, >127 IF TENTATIVE THE FIRST BLOCK ARRAY BEGINS
                        ;AT +576, TWO BYTES PER FILE. THE LAST BLOCK ARRAY BEGINS AT
                        ;+672.

                                .DEF    STATUS=528      ;OFFSET TO STATUS ARRAY
                                .DEF    FIRBLK=576      ;OFFSET TO FIRST BLOCK ARRAY
                                .DEF    LASBLK=672      ;OFFSET TO LAST BLOCK ARRAY
                        ;
B240  85  04    FSCAN:  STA     PNTR3           ;POINTER TO NAME
B242  84  05            STY     PNTR3+1
B244  AD  3ABF  FSCAN1: DMOV    INBUFD,FADDR    ;READ DIRECTORY
B250  A9  098D          DMOV#   DIRBLK,BLKNO    ;INTO INPUT BUFFER
B25A  A9  038D          MOV#    3,NBLKS         ;WE ONLY NEED THREE BLOCKS
B25F  20  E2BF          JSR     KREAD
                        ;SO WE SCAN THROUGH THE FILE NAMES LOOKING FOR THE NAME
                        ;BUT IGNORING FILES WHOSE STATUS IS NOT >0.
B262  AD  3ABF          DMOV    INBUFD,PNTR1    ;SETUP NAME POINTER
B26C  AD  3ABF          LDA     INBUFD          ;\
B26F  18  6910          ADD#    <STATUS         ; \
B272  85  02            STA     PNTR2           ;   SETUP POINTER
B274  AD  3BBF          LDA     INBUFD+1        ;   TO THE STATUS ARRAY
B277  69  02            ADC#    >STATUS         ; /
B279  85  03            STA     PNTR2+1         ;/
B27B  A9  308D          MOV#    MAXFL,FILENO    ;RESET FILE COUNTER
B280  A2  00            LDX#    0               ;COUNT FILES
B282  A0  00    NEXT:   LDY#    0
B284  B1  02            LDA@Y   PNTR2           ;IS STATUS>0
B286  F0  10            BEQ     TRYNXT          ;NOT VALID IF 0
B288  30  0E            BMI     TRYNXT          ;OR IF <0
B28A  B1  00    TSTLP:  LDA@Y   PNTR1           ;DOES THE NAME MATCH?
B28C  D1  04            CMP@Y   PNTR3
B28E  D0  08            BNE     TRYNXT          ;IF NOT TRY NEXT FILE
B290  C0  0A            CPY#    10
B292  F0  20            BEQ     MATCH           ;WHEN WE HAVE A MATCH
B294  C8                INY
B295  4C  8AB2          JMP     TSTLP
B298  E8        TRYNXT: INX                     ;COUNT IT
B299  A5  00            LDA     PNTR1
B29B  18  690B          ADD#    11              ;BUMP THE NAME POINTER
B29E  85  00            STA     PNTR1           ;BY THE SIZE OF ONE NAME
B2A0  90  02            BCC     NOBMP
B2A2  E6  01            INC     PNTR1+1
B2A4  E6  02D0  NOBMP:  DINC    PNTR2           ;BUMP THE STATUS POINTER
B2AA  CE  ECB2          DEC     FILENO          ;SEE IF WE HAVE CHECKED ALL FILE
B2AD  F0  03            BEQ     FAILED          ;IF SO THEN WE HAVE FAILED
B2AF  4C  82B2          JMP     NEXT            ;LOOP BACK FOR NEXT CHECK
B2B2  38        FAILED: SEC                     ;SHOW WE FAILED
B2B3  60                RTS
                        ;IF WE SUCCEED WE COME HERE
B2B4  AD  3ABF  MATCH:  LDA     INBUFD          ;\
B2B7  18  6940          ADD#    <FIRBLK         ; \
B2BA  85  00            STA     PNTR1           ;   MAKE A POINTER TO THE
B2BC  AD  3BBF          LDA     INBUFD+1        ;   FIRST BLOCK ARRAY
B2BF  69  02            ADC#    >FIRBLK         ; /
B2C1  85  01            STA     PNTR1+1         ;/
B2C3  AD  3ABF          LDA     INBUFD          ;\
```

```
B2C6 18 69A0            ADD#    <LASBLK         ; \
B2C9 85 02              STA     PNTR2           ;  AND A POINTER TO THE LAST
B2CB AD 3BBF            LDA     INBUFD+1        ;  BLOCK ARRAY
B2CE 69 02              ADC#    >LASBLK         ; /
B2D0 85 03              STA     PNTR2+1         ;/
B2D2 8A                 TXA                     ;PULL OUT THE BLOCK NUMBERS
B2D3 0A                 ASLA                    ;USING TWICE THE FILE NUMBER
B2D4 A8                 TAY
B2D5 B1 00              LDA@Y   PNTR1           ;FIRST BLOCK
B2D7 8D 69BF            STA     BLKNO           ;SAVE FILE FIRST BLOCK
B2DA B1 02              LDA@Y   PNTR2           ;LAST BLOCK
B2DC 8D 6EBF            STA     ENDBLK          ;SAVE FILE LIMIT BLOCK
B2DF C8                 INY
B2E0 B1 00              LDA@Y   PNTR1           ;DITTO HIGH BYTES
B2E2 8D 6ABF            STA     BLKNO+1         ;SAVED IN THE
B2E5 B1 02              LDA@Y   PNTR2           ;AUXILLIARY LOCATION
B2E7 8D 6FBF            STA     ENDBLK+1
B2EA 18                 CLC
B2EB 60                 RTS

B2EC 00         FILENO: .BYTE   0               ;LOCAL TEMPORARY

                ;FOR LOCAL USE, WE DEFINE THESE CONSOLE I/O FUNCTIONS

B2ED A2 00      OPCON:  LDX#    0               ;CONSOLE IS DEVICE 0
B2EF 8E 5CBF            STX     NOWDEV
B2F2 20 D9BF            JSR     KHAND           ;OPEN FOR INPUT
B2F5 A2 03              LDX#    3
B2F7 4C D9BF            JMP     KHAND           ;OPEN FOR OUTPUT

B2FA A2 00      CONOUT: LDX#    0
B2FC 8E 5CBF            STX     NOWDEV
B2FF A2 09              LDX#    9
B301 4C D9BF            JMP     KHAND           ;OUTPUT TO CONSOLE

B304 A2 00      CONIN:  LDX#    0
B306 8E 5CBF            STX     NOWDEV
B309 A2 06              LDX#    6
B30B 4C D9BF            JMP     KHAND           ;INPUT FROM CONSOLE

                ;NOW SOME JUNK FOR A NULL DEVICE:

B30E 4C 1AB3    NULDEV: JMP     PUNT
B311 4C 1AB3            JMP     PUNT
B314 4C 1AB3            JMP     PUNT
B317 4C 1AB3            JMP     PUNT
B31A A9 1A      PUNT:   LDA#    $1A             ;GIVE EOF INCASE IT WAS INPUT
B31C 18                 CLC
B31D 60                 RTS

                ;ROUTINE TO DISPATCH TO A PARTICULAR FUNCTION AND DEVICE
                ;THE DEVICE IT WILL USE IS IN THE SYSTEM GLOBAL "NOWDEV".
                ;THE FUNCTION CODE =(DEVICE HANDLER ENTRY OFFSET) IS IN THE
                ;X REGISTER. ARGUMENTS, IF ANY ARE TRANSFERED IN THE AC & Y.

B31E 48         HANDY:  PHA                     ;SAVE ARGUMENTS IF ANY
```

```
B31F 98                 TYA
B320 48                 PHA
B321 AD 5CBF            LDA     NOWDEV          ;AND THE DEVICE NUMBER
B324 C9 08              CMP#    8               ;NO MORE THAN 8 DEVICES!
B326 90 04              BLT     HANDOK          ;CONTINUE IF ITS OK
B328 68                 PLA                     ;REPAIR STACK
B329 68                 PLA
B32A 38                 SEC                     ;BAD DEVICE NUMBER
B32B 60                 RTS                     ;EXIT, CANT DO IT
B32C 0A         HANDOK: ASLA                    ;DEVICE NUMBER TIMES 2
B32D A8                 TAY                     ;WILL BE INDEX INTO TABLE
B32E 8A                 TXA                     ;FUNCTION CODE= OFFSET
B32F 18                 CLC
B330 79 C0BF            ADCY    DEVTAB          ;COMPUTE THE ADDRESS
B333 8D 42B3            STA     HANDV+1         ;OF THE ROUTINE
B336 A9 00              LDA#    0               ;THAT DOES THE JOB
B338 79 C1BF            ADCY    DEVTAB+1
B33B 8D 43B3            STA     HANDV+2
B33E 68                 PLA                     ;GET ARGUMENTS BACK
B33F A8                 TAY                     ;INTO Y REGISTER
B340 68                 PLA                     ;AND ACCUMULATOR
B341 4C 41B3    HANDV:  JMP     .               ;AND DISPATCH TO IT
```

*** ASM65 *** (V3.0-DS) OCT-79

```
                        .PAGE
                    ;DISK DRIVER.  USES RWTS WHICH MUST RESIDE AT $B800

                    ;USEFUL DEFINITIONS
                        .DEF    RWTS=$BD00      ;ENTRY POINT OF "RWTS"

                    ;COMMANDS TO RWTS
                        .DEF    NULL=0          ;"NULL" COMMAND
                        .DEF    READ0=1         ;"READ" DATA COMMAND
                        .DEF    WRITE=2         ;"WRITE" DATA COMMAND
                    ;ERROR CODES THAT MAY BE RETURNED BY THIS ROUTINE
                        .DEF    BADINF=1        ;BAD INFORMATION WAS PASSED
                        .DEF    WRTPRT=$10      ;DISKETTE WRITE PROTECTED
                        .DEF    VOLMMT=$20      ;VOLUME MISMATCH ERROR
                        .DEF    DRVERR=$40      ;STRANGE DRIVE ERROR
                        .DEF    RDERR=$80       ;READ ERROR

                    ;ROUTINE TO READ SEQUENTIAL BLOCKS FROM DISK.

B344 20 BEB3    ADEVRB: JSR     SETUP           ;SETUP GOODIES
B347 B0 10              BCS     RDBRET          ;BRANCH IF ERROR DURING SETUP
B349 A9 01      RDBNXT: LDA#    READ0           ;INDICATE THAT WE ARE READING
B34B 20 70B3            JSR     DOIO            ;DO THE I-O
B34E B0 09              BCS     RDBRET          ;BRANCH IF ERROR
B350 20 A0B3            JSR     INCADR          ;ADVANCE POINTERS
B353 CE 3EB4            DEC     ANBLKS          ;MORE BLOCKS TO READ?
B356 D0 F1              BNE     RDBNXT          ;BRANCH IF YES
B358 18         RESTD:  CLC                     ;NO, THEN INDICATE SUCCESS
B359 60         RDBRET: RTS                     ;RETURN TO CALLER

                    ;ROUTINE TO WRITE SEQUENTIAL BLOCKS TO DISK.

B35A 20 BEB3    DEVW0B: JSR     SETUP           ;SETUP GOODIES
B35D B0 10              BCS     WRBRET          ;BRANCH IF ERROR DURING SETUP
B35F A9 02      WRBNXT: LDA#    WRITE           ;INDICATE WE WANT TO WRITE
B361 20 70B3            JSR     DOIO            ;PERFORM THE I-O
B364 B0 09              BCS     WRBRET          ;BRANCH IF ERROR
B366 20 A0B3            JSR     INCADR          ;ADVANCE POINTERS
B369 CE 3EB4            DEC     ANBLKS          ;MORE BLOCKS TO WRITE?
B36C D0 F1              BNE     WRBNXT          ;BRANCH IF YES
B36E 18                 CLC                     ;NO, INDICATE SUCCESS
B36F 60         WRBRET: RTS                     ;RETURN TO CALLER

                    ;ROUTINE TO CALL "RWTS" SUBROUTINE.

B370 8D 5FB4    DOIO:   STA     IBCMD           ;SAVE COMMAND IN IOB
B373 AD 3FB4            DMOV    AFADR,IBADDR    ;COPY CURRENT ADDRESS INTO IOB
B37F 20 BAB3            JSR     TICK0           ;MAKE "TICKING" NOISES
B382 A9 B4              LDA#    >IOB            ;GET ADDR OF IOB
B384 A0 53              LDY#    <IOB
B386 20 00BD            JSR     RWTS            ;CALL "RWTS" TO DO THE WORK
B389 08                 PHP                     ;SAVE PROCESSOR STATUS
B38A A9 008D            MOV#    NULL,IBCMD      ;ZAP COMMAND
```

```
B38F AD 54B4            DMOV    IOBSLT,IBPSLT   ;MOVE CURRENT SLOT AND DRIVE
B39B AD 60B4            LDA     IBSTAT          ;GET STATUS CODE IN CASE ERROR
B39E 28                 PLP                     ;RESTORE PROCESSOR STATUS
B39F 60         DOIORT: RTS                     ;AND RETURN TO CALLER

                ;ROUTINE TO ADVANCE POINTERS IN MEMORY AND ON DISK.

B3A0 EE 40B4    INCADR: INC     AFADR+1         ;ADVANCE MEMORY ADDRESS
B3A3 EE 58B4            INC     IBSECT          ;BUMP TO NEXT SECTOR NUMBER
B3A6 AD 58B4            LDA     IBSECT
B3A9 AE 68BF            LDX     UNIT
B3AC DD 4BB4            CMPX    NUMSEC          ;DID WE OVERFLOW TO NEXT TRACK?
B3AF 90 08              BCC     INCRET          ;BRANCH IF NOT
B3B1 A9 00              LDA#    0               ;YES, THEN SET TO SECTOR ZERO
B3B3 8D 58B4            STA     IBSECT
B3B6 EE 57B4            INC     IBTRK           ;ON NEXT TRACK
B3B9 60         INCRET: RTS                     ;AND RETURN TO CALLER

                ;SUBROUTINE TO TOGGLE THE SPEAKER IN ORDER TO MAKE A TICKING
                ;NOISE BECAUSE THE DISK-II IS TOO QUIET.

B3BA 8D 30C0    TICK0:  STA     ASPEAK          ;TOGGLE SPEAKER PORT
B3BD 60                 RTS                     ;AND RETURN

                ;ROUTINE TO SETUP THINGS PRIOR TO DOING I-O.

B3BE 20 2AB4    SETUP:  JSR     VALDRV          ;IS THIS A PERMITTED UNIT?
B3C1 B0 63              BCS     SETERR          ;NO, ERROR
B3C3 AD 6BBF            DMOV    NBLKS,ANBLKS    ;LOCAL COPY OF NUMBER OF BLKS
B3CF AD 6CBF            DMOV    FADDR,AFADR     ;LOCAL COPY OF BASE ADDRESS
B3DB AE 68BF            LDX     UNIT            ;GET UNIT NUMBER
B3DE BD 43B4            LDAX    DEVSLT          ;CONVERT TO SLOT NUMBER OF CONTR
B3E1 F0 43              BEQ     SETERR          ;BRANCH IF ERROR, BUM INFO
B3E3 48                 PHA
B3E4 29 F0              AND#    $F0
B3E6 8D 54B4            STA     IOBSLT          ;SAVE FOR "RWTS" ROUTINE
B3E9 68                 PLA
B3EA 29 0F              AND#    $0F
B3EC 8D 55B4            STA     IOBDRV          ;SAVE FOR "RWTS" ROUTINE
B3EF A9 008D            MOV#    0,IBTRK         ;SET TRACK NUMBER TO ZERO
B3F4 AD 69BF            MOV     BLKNO,IBSECT    ;GET LOW ORDER BLOCK NUMBER
B3FA AD 6ABF            MOV     BLKNO+1,TEMP    ;STORE HIGH ORDER BLOCK
B400 AD 41B4    NEXTRY: LDA     TEMP            ;GET HIGH ORDER
B403 D0 08              BNE     NOTTRK          ;CAN'T BE DONE IF NON-ZERO
B405 AD 58B4            LDA     IBSECT          ;GET REMAINDER
B408 DD 4BB4            CMPX    NUMSEC          ;LESS THAN A TRACK YET?
B40B 90 18              BCC     GOTIT           ;BRANCH IF YES, THEN WE HAVE IT
B40D AD 58B4    NOTTRK: LDA     IBSECT          ;NO, SO SUBTRACT ONE TRACK SIZE
B410 38                 SEC
B411 FD 4BB4            SBCX    NUMSEC
B414 8D 58B4            STA     IBSECT
B417 AD 41B4            LDA     TEMP
B41A E9 00              SBC#    0
B41C 8D 41B4            STA     TEMP
B41F EE 57B4            INC     IBTRK           ;AND ADVANCE TO NEXT TRACK
B422 4C 00B4            JMP     NEXTRY          ;AND CONTINUE
```

```
B425 60          GOTIT:  RTS                     ;YES, THEN ALL OK

B426 A9 01       SETERR: LDA#    BADINF          ;INDICATE BAD INFO
B428 38                  SEC                     ;INDICATE FAILURE
B429 60                  RTS                     ;AND RETURN

B42A A9 01       VALDRV: LDA#    1               ;AC WILL BE MASK
B42C AE 68BF             LDX     UNIT            ;GET UNIT NUMBER
B42F F0 04               BEQ     VALD1           ;SKIP SHIFT IF UNIT ZERO
B431 0A          VALD2:  ASLA                    ;ELSE SHIFT THE BIT UP
B432 CA                  DEX
B433 D0 FC               BNE     VALD2
B435 2D 51BF     VALD1:  AND     DEVMSK          ;COMPARE TO PERMIT BYTE
B438 D0 02               BNE     VALD3           ;#0 MEANS THE UNIT IS OK
B43A 38                  SEC                     ;ELSE INVALID
B43B 60                  RTS
B43C 18          VALD3:  CLC
B43D 60                  RTS

B43E 00          ANBLKS: .BYTE   0               ;HOLDS NUMBER OF BLOCKS
B43F 00 00       AFADR:  .WORD   0               ;BASE ADDR OF TRANSFER
B441 00 00       TEMP:   .WORD   0               ;GENERAL PURPOSE TEMPORARY

                 ;THESE TABLES ARE INDEXED BY UNIT NUMBER

                 ;ENTRIES TO DEVSLT ARE: SLOT*$10+DRIVE
B443 61          DEVSLT: .BYTE   $61             ;UNIT 0 = SLOT 6 DRIVE 1
B444 62                  .BYTE   $62             ;UNIT 1 = SLOT 6 DRIVE 2
B445 51                  .BYTE   $51             ;UNIT 2 = SLOT 5 DRIVE 1
B446 52                  .BYTE   $52             ;UNIT 3 = SLOT 5 DRIVE 2
B447 71                  .BYTE   $71             ;UNIT 4 = SLOT 7 DRIVE 1
B448 72                  .BYTE   $72             ;UNIT 5 = SLOT 7 DRIVE 2
B449 73                  .BYTE   $73             ;UNIT 6 = SLOT 7 DRIVE 3
B44A 74                  .BYTE   $74             ;UNIT 7 = SLOT 7 DRIVE 4

                 ;NUMBER OF SECTORS PER TRACK
B44B 0D          NUMSEC: .BYTE   13              ;ORIGINAL APPLE FORMAT USES
B44C 0D                  .BYTE   13              ;13 PER TRACK FOR DISK ][.
B44D 0D                  .BYTE   13
B44E 0D                  .BYTE   13
B44F 1A                  .BYTE   26              ;8" USES 26 PER PSUEDO TRACK
B450 1A                  .BYTE   26              ;WHEN ITS A SORRENTO VALLEY
B451 1A                  .BYTE   26              ;CONTROLLER.
B452 1A                  .BYTE   26

                 ;THIS IS THE I-O CONTROL BLOCK FOR "RWTS" SUBROUTINE.

B453 01          IOB:    .BYTE   1               ;TYPE OF IOB (ALWAYS A 1)
B454 00          IOBSLT: .BYTE   0               ;HOLDS SLOT NUMBER OF DISK CONTR
B455 00          IOBDRV: .BYTE   0               ;HOLDS DRIVE NUMBER OF UNIT
B456 00                  .BYTE   0               ;VOLUME NUMBER (0 MATCHES ALL)
B457 00          IBTRK:  .BYTE   0               ;HOLDS TRACK NUMBER
B458 00          IBSECT: .BYTE   0               ;HOLDS SECTOR NUMBER
B459 64 B4               .WORD   DEVT            ;POINTER TO DISK CHARACTER
B45B 00 00       IBADDR: .WORD   0               ;POINTER TO WHERE DATA IS
B45D 00                  .BYTE   0               ;UNUSED
```

```
B45E 00              .BYTE   0           ;UNUSED
B45F 00        IBCMD:.BYTE   0           ;HOLDS COMMAND CODE
B460 00        IBSTAT:.BYTE  0           ;ERROR CODE IF CARRY IS SET
B461 00        IBVOL:.BYTE   0           ;VOLUME NUMBER FOUND
B462 60        IBPSLT:.BYTE  $60         ;SLOT NUMBER OF PREVIOUS ACCESS
B463 01        IBPDRV:.BYTE  1           ;DRIVE OF PREVIOUS ACCESS
B464 00        DEVT: .BYTE   0           ;DISK TYPE (0 FOR DISK-II)
B465 01              .BYTE   1           ;NUMBER OF PHASES (=1)
B466 EF D8           .WORD   $D8EF       ;TIME COUNT ($D8EF FOR DISK-II)
```

```
*** ASM65 ***  (V3.0-DS)  OCT-79

                        .PAGE
                ;               ***SYSTEM PAGE MODULE FOR APEX***

                ;       ***PROGRAM SPECIFIC AREA***

                ;THE FIRST $50 LOCATIONS OF THIS PAGE ARE SAVED WITH THE
                ;USER PROGRAM IN PLACE OF THE FIRST $50 LOCATIONS IN PAGE 0

                        .DEF    HERE=$BF00
                        .LOC    HERE,HERE-BIAS

                        .DEF    SYSPAG=.

                ;PROGRAM START AND EXIT VECTORS:
BF00 4C 00BF    VRSTRT: JMP     .       ;PROGRAM RESTART VECTOR
BF03 4C 03BF    VSTART: JMP     .       ;PROGRAM START VECTOR
BF06 4C D0BF    VEXIT:  JMP     KRENTR  ;PROGRAM NORMAL EXIT ADDRESS
BF09 4C D6BF    VERROR: JMP     KRELOD  ;PROGRAM ERROR EXIT ADDRESS
BF0C 4C D3BF    VABORT: JMP     KSAVER  ;USER ABORT EXIT ADDRESS

                ;INTERNAL USE BY APEX:
BF0F 00                 .BYTE   0       ;SPARE
BF10 00 00      DSKMEM: .WORD   0       ;BASE ADDR OF MEMORY SEGMENT ON DISK
BF12 00         DSKSIZ: .BYTE   0       ;SIZE OF IMAGE ON THE DISK
BF13 00 00              .WORD   0       ;SPARE

                ;PROGRAM MEMORY PARAMETERS:
BF15 00 00      USRMEM: .WORD   0       ;BASE ADDR OF USER PROGRAM
BF17 00         PROSIZ: .BYTE   0       ;USER PROGRAM SIZE IN PAGES

                ;(LEAVE SPACE HERE FOR MORE USER PROGRAM SEGMENTS)

                        .DEF    HERE=SYSPAG+$20
                        .LOC    HERE,HERE-BIAS

BF20 00         RERUNF: .BYTE   0       ;RERUN FLAG
BF21 54 4D50    DEXTO:  .ASCII  "TMP"   ;DEFAULT EXTENSION FOR OUTPUT FILES
BF24 54 4D50    DEXTI:  .ASCII  "TMP"   ;DEFAULT EXTENSION FOR INPUT FILES
BF27 00         DEFAUL: .BYTE   0       ;SINGLE BIT DEFAULT FLAGS
BF28 FF         SYBOMB: .BYTE   $FF     ;$FF IF PROG BOMBS SYSTEM
BF29 B0         USRTOP: .BYTE   $B0     ;LAST PAGE+1 FOR USER PROGRAM
BF2A 00                 .BYTE   0       ;SPARE
BF2B 00 00              .WORD   0       ;SPARE
BF2D 00 00              .WORD   0       ;SPARE

                ;I2L PARAMETERS:
BF2F 00         I2LFLG: .BYTE   0       ;$FF IF PROGRAM USES SYSTEM I2L
BF30 00 00      I2LBAS: .WORD   0       ;START OF PSEUDO CODE FOR SYSTEM I2L
BF32 00 00      I2LHEP: .WORD   0       ;START OF HEAP FOR SYSTEM I2L
BF34 00 00              .WORD   0       ;SPARE

                ;I/O BUFFERS:  (MAY NEED TO BE INTEGRAL PAGES! SEE HANDLER)
BF36 00 63      OTBUFD: .WORD   $6300   ;BASE OF OUTPUT BUFFER TO USE
```

```
BF38 FF 63      OTBUFE: .WORD   $63FF   ;END OF OUTPUT BUFFER
BF3A 00 60      INBUFD: .WORD   $6000   ;BASE OF INPUT BUFFER TO USE
BF3C FF 62      INBUFE: .WORD   $62FF   ;END OF INPUT BUFFER

                ;(LEAVE SPACE HERE FOR MORE INPUT FILE BUFFER SPECS)
```

*** ASM65 *** (V3.0-DS) OCT-79

```
                        .PAGE
                ;       SYSTEM SPECIFIC AREA

                ;THE REST OF THIS PAGE IS NOT SAVED BUT RATHER IS BOOTED
                ;DURING THE ORIGINAL BOOTUP PROCESS.  IT REMAINS UNCHANGED
                ;DURING NORMAL OPERATION.

                        .DEF    HERE=SYSPAG+$50
                        .LOC    HERE,HERE-BIAS

BF50 FC         SYSENF  .BYTE   $FC             ;FLAG SHOWING RE-ENTRY CONDITION
BF51 01         DEVMSK: .BYTE   $01             ;MASK SHOWING VALID UNITS
BF52 00         SYSDEV: .BYTE   0               ;UNIT SYSTEM IS ON
BF53 00 00      SYSBLK: .WORD   0               ;BLOCK SYSTEM FILE IS IN
BF55 00 00      SWPBLK: .WORD   0               ;BLOCK SWAP FILE IS IN
BF57 00 00      SYSDAT: .WORD   0               ;SYSTEM DATE GOES HERE
BF59 00                 .BYTE   0               ;3 LOCS
BF5A FF 02      LINIDX: .WORD   $2FF            ;INPUT LINE POINTER ($FF=NULL)
BF5C 00         NOWDEV: .BYTE   0               ;CURRENT BYTE I/O DEVICE
BF5D FF         EXECUT: .BYTE   $FF             ;ZERO IF EXEC MODE IS ON
BF5E 00         LOWER:  .BYTE   0               ;LOWER CASE SWITCH (0=UPPER)

                        .DEF    HERE=SYSPAG+$68         ;LEAVE SPACE FOR LATER
                        .LOC    HERE,HERE-BIAS

                ;I-O INFORMATION BLOCK FOR UNIT DRIVERS:
BF68 00         UNIT:   .BYTE   0               ;CURRENT UNIT NUMBER
BF69 00 00      BLKNO:  .WORD   0               ;CURRENT BLOCK NUMBER
BF6B 00         NBLKS:  .BYTE   0               ;NUMBER OF BLOCKS TO TRANSFER
BF6C 00 00      FADDR:  .WORD   0               ;ADDRESS POINTER
BF6E 00 00      ENDBLK: .WORD   0               ;AUXILLIARY PARAMETER

                ;OUTPUT FILE INFORMATION:
BF70 00 00      OTLBLK: .WORD   0               ;FIRST BLOCK OF OUTPUT FILE
BF72 00 00      OTHBLK: .WORD   0               ;LAST BLOCK OF OUTPUT FILE
BF74 00         OTFLG:  .BYTE   0               ;STATUS FLAGS
BF75 00         OTNO:   .BYTE   0               ;DIRECTORY NUMBER OF OUTPUT FILE
BF76 00         OTDEV:  .BYTE   0               ;UNIT NUMBER WHICH OUTPUT FILE I
BF77 00                 .BYTE   0               ;SPARE

                ;INPUT FILE INFORMATION:
BF78 00 00      INLBLK: .WORD   0               ;FIRST BLOCK OF INPUT FILE
BF7A 00 00      INHBLK: .WORD   0               ;LAST BLOCK OF INPUT FILE
BF7C 00         INFLG:  .BYTE   0               ;STATUS FLAGS
BF7D 00         INNO:   .BYTE   0               ;DIRECTORY NUMBER OF INPUT FILE
BF7E 00         INDEV:  .BYTE   0               ;UNIT NUMBER WHICH INPUT FILE IS
BF7F 00                 .BYTE   0               ;SPARE

                ;SPACE FOR MORE FILES HERE
```

```
*** ASM65 ***  (V3.0-DS)  OCT-79

                        .PAGE
                        .DEF    HERE=SYSPAG+$C0
                        .LOC    HERE,HERE-BIAS

                ;BYTE I/O DEVICE HANDLER ADDRESS TABLE:
BFC0 C0 B5      DEVTAB: .WORD   $B5C0           ;0=LINE BUFFERED CONSOLE
BFC2 80 B4              .WORD   $B480           ;1=CONSOLE DEVICE
BFC4 0E B3              .WORD   NULDEV          ;2=PRINTER
BFC6 00 AD              .WORD   $AD00           ;3=DISK FILES
BFC8 0E B3              .WORD   NULDEV          ;4=RS232 SERIAL PORT
BFCA 0E B3              .WORD   NULDEV          ;5=UNUSED
BFCC 0E B3              .WORD   NULDEV          ;6=UNUSED
BFCE 0E B3              .WORD   NULDEV          ;7=NULL DEVICE


                ;LINK VECTORS TO RESIDENT CODE:
BFD0 4C 94B0    KRENTR: JMP     RENTER          ;BOOT IN APEX (WARM START)
BFD3 4C 2AB1    KSAVER: JMP     SAVER           ;PRESERVE CURRENT USER IMAGE
BFD6 4C 00B0    KRELOD: JMP     RELOAD          ;RELOAD APEX (COLD START)
BFD9 4C 1EB3    KHAND:  JMP     HANDY           ;LINK TO BYTE I/O DEVICE
BFDC 4C 40B2    KSCAN:  JMP     FSCAN           ;FILE LOOKUP ROUTINE
BFDF 4C 58B3    KRESTD: JMP     RESTD           ;RESET DISK DRIVER
BFE2 4C 44B3    KREAD:  JMP     ADEVRB          ;READ CONTIGUOUS BLOCKS
BFE5 4C 5AB3    KWRITE: JMP     DEVWOB          ;WRITE CONTIGUOUS BLOCKS
                ;LINKS FOR APEX EXEC USE:
BFE8 4C 63B1    KSAVS:  JMP     SAVESC          ;SAVE A SWAPPED MEMORY IMAGE
BFEB 4C CAB0    KRUN:   JMP     RUN             ;RUN A MEMORY IMAGE
BFEE 4C AAB1    KSAVA:  JMP     SAVEA           ;SAVE A MEMORY IMAGE
BFF1 4C D5B0    KGET:   JMP     GET             ;GET A MEMORY IMAGE

                        .END

.       BFF4            ADEVRB  B344            AFADR   B43F
ANBLKS  B43E            ASPEAK  C030            BADINF  0001
BARF    B047            BIAS    9000            BLKNO   BF69
BOOT    B0B2            COMPAG  B700            CONIN   B304
CONOUT  B2FA            DEFAUL  BF27            DERMES  B221
DEVMSK  BF51            DEVSLT  B443            DEVT    B464
DEVTAB  BFC0            DEVWOB  B35A            DEXTI   BF24
DEXTO   BF21            DIRBLK  0009            DOIO    B370
DOIORT  B39F            DRVERR  0040            DSKMEM  BF10
DSKSIZ  BF12            ENDBLK  BF6E            ENTER   B096
ENTR2   B0A3            EXECUT  BF5D            FADDR   BF6C
FAILED  B2B2            FILENO  B2EC            FIRBLK  0240
FSCAN   B240            FSCAN1  B244            FSCANR  B076
GET     B0D5            GETR    B0DB            GOTIT   B425
HANDOK  B32C            HANDV   B341            HANDY   B31E
HERE    BFC0            HIAD    B000            HMDONE  B21A
HMLOOP  B20B            HRDERR  B1FF            I2LBAS  BF30
I2LFLG  BF2F            I2LHEP  BF32            IBADDR  B45B
IBCMD   B45F            IBPDRV  B463            IBPSLT  B462
IBSECT  B458            IBSTAT  B460            IBTRK   B457
IBVOL   B461            INBUFD  BF3A            INBUFE  BF3C
```

INCADR	B3A0	INCRET	B3B9	INDEV	BF7E
INFLG	BF7C	INHBLK	BF7A	INLBLK	BF78
INNO	BF7D	INPBUF	A000	IOB	B453
IOBDRV	B455	IOBSLT	B454	KGET	BFF1
KHAND	BFD9	KREAD	BFE2	KRELOD	BFD6
KRENTR	BFD0	KRESTD	BFDF	KRUN	BFEB
KSAVA	BFEE	KSAVER	BFD3	KSAVS	BFE8
KSCAN	BFDC	KWRITE	BFE5	LASBLK	02A0
LINIDX	BF5A	LOAD	2000	LOWER	BF5E
LTEMP	B220	MATCH	B2B4	MAXFL	0030
MONITR	FF59	MOVMEM	B0FE	MOVSWP	B11B
NBLKS	BF6B	NEXT	B282	NEXTRY	B400
NOBARF	B140	NOBMP	B2A4	NOTTRK	B40D
NOWDEV	BF5C	NULDEV	B30E	NULL	0000
NUMSEC	B44B	OPCON	B2ED	OTBUFD	BF36
OTBUFE	BF38	OTDEV	BF76	OTFLG	BF74
OTHBLK	BF72	OTLBLK	BF70	OTNO	BF75
PNTR1	0000	PNTR2	0002	PNTR3	0004
PROSIZ	BF17	PUNT	B31A	RDBNXT	B349
RDBRET	B359	RDERR	0080	READ0	0001
READUS	B1EF	REL1	B007	RELOAD	B000
RENTER	B094	RERUNF	BF20	RESTD	B358
RUN	B0CA	RWTS	BD00	SAVE	B0EB
SAVEA	B1AA	SAVER	B12A	SAVESC	B163
SERMES	B053	SET1	B1C4	SET3	B1D4
SETERR	B426	SETUP	B3BE	STATUS	0210
SWPBLK	BF55	SWPFIL	B03C	SWPLOC	6000
SWPSIZ	0040	SYBOMB	BF28	SYSBLK	BF53
SYSDAT	BF57	SYSDEV	BF52	SYSENF	BF50
SYSFIL	B031	SYSPAG	BF00	TEMP	B441
TICK0	B3BA	TRYNXT	B298	TRYSWP	B0AA
TSTLP	B28A	UNIT	BF68	USRMEM	BF15
USRTOP	BF29	VABORT	BF0C	VALD1	B435
VALD2	B431	VALD3	B43C	VALDRV	B42A
VERROR	BF09	VEXIT	BF06	VOLMMT	0020
VRSTRT	BF00	VSTART	BF03	WRBNXT	B35F
WRBRET	B36F	WRITE	0002	WRITER	B1FB
WRITUS	B1F5	WRTPRT	0010		

NO ERRORS DETECTED

```
;       APEX  CONSOLE  HANDLER

        .DEF    HIAD=$B480      ;ACTUALLY GOES HERE
        .DEF    LOAD=$2480      ;BUT WE LOAD IT HERE
        .DEF    BIAS=HIAD-LOAD

;WHERE THE DEVICE HANDLER TABLE IS SO WE CAN PATCH INTO IT:
        .DEF    DEVTAB=$BFC0

;SOME SPECIAL CHARATERS:
        .DEF    RETYPE=$15      ;CTRL-U = RETYPE CHARACTER
        .DEF    LINDEL=$18      ;CTRL-X = DELETE LINE
        .DEF    KBABOR=$19      ;CTRL-Y = ABORT PROGRAM
        .DEF    KBEXIT=$3       ;CTRL-C = EXIT PROGRAM
        .DEF    DOBLSH=$F       ;CRTL-O = GET BACKSLASH
        .DEF    DOBRAK=$B       ;CTRL-K = GET LEFT BRACKET
        .DEF    HALTER=$13      ;CTRL-S = STOP OUTPUT
        .DEF    CASSHF=$1D      ;CTRL-SHIFT-M = SHIFT CASE
        .DEF    EOFCOD=$1A      ;CTRL-Z = END-OF-FILE

;ASCII CHARACTERS
        .DEF    BSPCOD=$8       ;BACKSPACE
        .DEF    CRCOD=$D        ;RETURN
        .DEF    LFCOD=$A        ;LINE FEED
        .DEF    ESCCOD=$1B      ;ESCAPE
        .DEF    RUBCHR=$7F      ;RUBOUT
        .DEF    TABCOD=$9       ;TAB
        .DEF    FORMCD=$C       ;FORM FEED
        .DEF    BELCOD=$7       ;BELL

;SOME ROM ROUTINES WE NEED
        .DEF    ESC1=$FC2C      ;ADDRESS OF ESC PROCESSOR
        .DEF    CLREOL=$FC9C    ;CLEAR TO END OF LINE
        .DEF    BELL=$FF3A      ;HONK
        .DEF    APPTVO=$FBFD    ;"VIDOUT"
        .DEF    APHOME=$FC58    ;"HOME"
        .DEF    APVTAB=$FC22    ;"VTAB"
        .DEF    XKBINP=$FD1B    ;"KEYIN"
        .DEF    APBELL=$FBDD    ;SNEAK ENTRY TO "BELL"

;SOME OTHER APPLE LOCATIONS WE NEED TO KNOW ABOUT
        .DEF    IN=$200         ;INPUT LINE BUFFER
        .DEF    BASL=$28        ;LINE BASE POINTER
        .DEF    CH=$24          ;HORIZONTAL INDEX
        .DEF    CV=$25          ;APPLE VERTICAL POSITION REG

;SYSTEM GLOBALS WE NEED TO KNOW ABOUT
        .DEF    VABORT=$BF0C    ;ABORT EXIT VECTOR
        .DEF    VEXIT=$BF06     ;NORMAL EXIT VECTOR
        .DEF    LINIDX=$BF5A    ;INPUT LINE INDEX
        .DEF    EXECUT=$BF5D    ;EXEC MODE FLAG
        .DEF    LOWER=$BF5E     ;LOWER CASE FLAG

;SOME HARDWARE LOCATIONS IN THE APPLE
        .DEF    XKBRES=$C010    ;RESET KEYFLAG
        .DEF    XKBDAT=$C000    ;KEYBOARD DATA PORT
```

```
                        .DEF    XTEXTR=$C051    ;SET TEXT MODE


                        .LOC    HIAD,HIAD-BIAS

                ;ENTRY POINTS:
B480 4C E5B4    CONOPI: JMP     KEYINI          ;0=OPEN FOR INPUT
B483 4C E0B4    CONOPO: JMP     TVINI           ;3=OPEN FOR OUTPUT
B486 4C 95B4    CONIN:  JMP     GETKYO          ;6=INPUT
B489 4C 1CB5    CONOUT: JMP     TVOUT           ;9=OUTPUT
B48C 4C E3B4            JMP     CONCLO          ;12=CLOSE
B48F 4C 79B5            JMP     CONCUR          ;15=DIRECT CURSOR SET
B492 4C EFB4            JMP     CHKUSR          ;18=CHECK FOR INTERRUPTS

                ;SINGLE CHARACTER KEYBOARD INPUT ROUTINE

B495 A4 24      GETKYO: LDY     CH              ;WE HAVE TO FAKE
B497 B1 28              LDA@Y   BASL            ;THE ROM RDKEY ROUTINE
B499 48                 PHA                     ;COS WE DONT WANT TO
B49A 29 3F              AND#    $3F             ;GO THRU $38 AT THIS POINT
B49C 09 40              ORA#    $40
B49E 91 28              STA@Y   BASL            ;SET SCREEN FLASHING
B4A0 68                 PLA
B4A1 20 1BFD    KEYINS: JSR     XKBINP          ;GET A CHAR, UPDATING HASH COUNT
B4A4 29 7F              AND#    $7F             ;KEEP ONLY 7 BIT ASCII
B4A6 C9 03              CMP#    KBEXIT          ;EXIT?
B4A8 F0 64              BEQ     FINEX           ;BRANCH IF YES
B4AA C9 19              CMP#    KBABOR          ;ABORT?
B4AC F0 6B              BEQ     SAVEX           ;BRANCH IF YES
B4AE C9 0F              CMP#    DOBLSH          ;WANTS A BACKSLASH?
B4B0 F0 1B              BEQ     BSLASH          ;YES, TURN IT INTO A "\"
B4B2 C9 0B              CMP#    DOBRAK          ;WANTS A SQUARE BRACKET?
B4B4 F0 1B              BEQ     LBRACK          ;YES, TURN IT INTO A "["
B4B6 C9 1D              CMP#    CASSHF          ;WANTS TO SHIFT CASE?
B4B8 F0 1B              BEQ     SWITCH          ;YES ,GO DO THAT
B4BA C9 08              CMP#    BSPCOD          ;BACKSPACE (THE LEFT ARROW)?
B4BC D0 02              BNE     KEYRET          ;BRANCH IF NOT
B4BE A9 7F              LDA#    RUBCHR          ;YES, CHANGE TO RUBOUT
B4C0 C9 40      KEYRET: CMP#    $40             ;ALPHA?
B4C2 90 07              BLT     KEYSAM          ;NO LEAVE IT
B4C4 AE 5EBF            LDX     LOWER           ;LOWER CASE ON?
B4C7 F0 02              BEQ     KEYSAM          ;NO LEAVE IT
B4C9 09 20              ORA#    $20             ;YES, MAKE LOWER CASE
B4CB 18         KEYSAM: CLC                     ;INDICATE SUCCESS
B4CC 60                 RTS                     ;AND RETURN

B4CD A9 5C      BSLASH: LDA#    '\              ;GET A BACKSLASH SINCE WE CAN'T
B4CF D0 EF              BNE     KEYRET          ;GENERATE IT DIRECTLY.
B4D1 A9 5B      LBRACK: LDA#    '[              ;GET A LEFT BRACKET SINCE
B4D3 D0 EB              BNE     KEYRET          ;WE CAN'T GENERATE IT DIRECTLY.

B4D5 AD 5EBF    SWITCH: LDA     LOWER           ;TOGGLE CASE SWITCH
B4D8 49 FF              EOR#    $FF
B4DA 8D 5EBF            STA     LOWER           ;SAVE IT BACK
B4DD 4C 95B4            JMP     GETKYO          ;AND GO GET ANOTHER KEY
```

```
                        ;ROUTINE TO INITIALISE THE CONSOLE FOR OUTPUT

B4E0 2C 51C0    TVINI:  BIT     XTEXTR          ;SET TEXT MODE, ETC.
B4E3 18         CONCLO: CLC                     ;INDICATE SUCCESS
B4E4 60                 RTS                     ;AND RETURN

B4E5 AD 10C0    KEYINI: LDA     XKBRES          ;DROP KEY STROBE IF ANY
B4E8 A9 008D            MOV#    0,LOWER         ;UPPER CASE.
B4ED 18                 CLC                     ;ALL OK
B4EE 60                 RTS

                        ;ROUTINE TO CHECK TO SEE IF USER TYPED AN ABORT CHAR
                        ;OR ASKED TO FREEZE OUTPUT.  THIS ROUTINE MUST PRESERVE ALL
                        ;THE REGISTERS.  IT MAY DESTROY THE PROCESSOR STATUS.

B4EF 48         CHKUSR: PHA                     ;SAVE THE ACCUMULATOR CONTENTS
B4F0 AD 00C0            LDA     XKBDAT          ;KEY STRUCK WHILE WE WERE GONE?
B4F3 10 0F              BPL     CHKURT          ;BRANCH IF NOT, RETURN
B4F5 C9 93              CMP#    HALTER+$80      ;YES, WAS IT HALT REQUEST?
B4F7 D0 0D              BNE     NOHNG           ;BRANCH IF NOT
B4F9 AD 10C0            LDA     XKBRES          ;YES, EAT THE CHAR
B4FC AD 00C0    KYWAIT: LDA     XKBDAT          ;AND WAIT FOR ANY KEY
B4FF 10 FB              BPL     KYWAIT
B501 AD 10C0            LDA     XKBRES          ;EAT THE KEY
B504 68         CHKURT: PLA                     ;RESTORE ACCUMULATOR CONTENTS
B505 60                 RTS                     ;AND RETURN
B506 C9 83      NOHNG:  CMP#    KBEXIT+$80      ;ABORT?
B508 D0 07              BNE     NOHNG1          ;BRANCH IF NOT
B50A AD 10C0            LDA     XKBRES          ;YES, EAT THE CHAR
B50D 68                 PLA                     ;CLEAN UP STACK
B50E 4C 06BF    FINEX:  JMP     VEXIT           ;NO,EXIT NORMALLY

B511 C9 99      NOHNG1: CMP#    KBABOR+$80      ;ABORT?
B513 D0 EF              BNE     CHKURT          ;BRANCH IF NOT, RETURN TO CALLER
B515 AD 10C0            LDA     XKBRES          ;YES, EAT THE CHAR
B518 68                 PLA                     ;CLEAN UP STACK
B519 4C 0CBF    SAVEX:  JMP     VABORT          ;AND EXIT THROUGH USER ABORT VEC

                        ;TV OUTPUT SUBROUTINE.
                        ;CALL WITH THE ASCII CHAR TO OUTPUT IN THE ACCUMULATOR

B51C 20 EFB4    TVOUT:  JSR     CHKUSR          ;CHECK FOR USER ABORT
B51F 29 7F              AND#    $7F             ;FOR SAFETY
B521 C9 09              CMP#    TABCOD          ;TAB?
B523 F0 34              BEQ     DOTAB           ;YES, FAKE IT
B525 C9 0C              CMP#    FORMCD          ;FORM FEED?
B527 F0 2B              BEQ     DOFORM          ;YES, FAKE IT
B529 C9 0D              CMP#    CRCOD           ;CARRIAGE RETURN?
B52B F0 45              BEQ     DOCR            ;YES, FAKE IT
B52D C9 7F              CMP#    RUBCHR          ;RUBOUT CHARACTER?
B52F F0 35              BEQ     RUBOUT          ;YES, FAKE IT
B531 C9 0A              CMP#    LFCOD           ;LINE FEED?
B533 F0 18              BEQ     TVNOR           ;YES, LET IT PASS
B535 C9 07              CMP#    BELCOD          ;BELL?
B537 F0 14              BEQ     TVNOR           ;YES, LET IT PASS
B539 C9 08              CMP#    BSPCOD          ;BACKSPACE?
```

```
B53B F0 31              BEQ     BACKSP          ;YES, GO HANDLE
B53D C9 61      DIRTV:  CMP#    %141            ;TEST FOR LOWER CASE ALPHA
B53F 90 04              BLT     CHKCTL          ;BRANCH IF NOT
B541 49 60              EOR#    $60             ;MAKE IT NON-BLINKING
B543 10 0A              BPL     TOTV            ;BUT INVERSE VIDEO.
B545 C9 20      CHKCTL: CMP#    %40             ;TEST FOR OTHER CONTROL CHARS
B547 B0 04              BCS     TVNOR           ;BRANCH IF NOT A CONTROL CHAR
B549 09 40              ORA#    $40             ;CONTROL CHAR.  BLINK IT.
B54B D0 02              BNE     TOTV            ;AND PLACE IN INVERSE VIDEO.
B54D 49 80      TVNOR:  EOR#    $80             ;NORMAL CHARACTER
B54F 20 FDFB    TOTV:   JSR     APPTVO          ;CALL APPLE ROM TO OUTPUT
B552 18         NOTV:   CLC                     ;INDICATE SUCCESS
B553 60                 RTS                     ;AND RETURN

B554 20 58FC    DOFORM: JSR     APHOME          ;DO APPLE HOME-ERASE
B557 18                 CLC                     ;INDICATE SUCCESS
B558 60                 RTS                     ;AND RETURN

B559 A9 20      DOTAB:  LDA#    ' '             ;GET A SPACE
B55B 20 3DB5            JSR     DIRTV           ;OUTPUT IT
B55E A5 24              LDA     CH              ;TEST HORIZ POSITION
B560 29 07              AND#    $7              ;REACHED TAB STOP YET?
B562 D0 F5              BNE     DOTAB           ;BRANCH IF NOT, KEEP SPACING
B564 18                 CLC                     ;YES, INDICATE SUCCESS
B565 60                 RTS                     ;AND RETURN

B566 20 6EB5    RUBOUT: JSR     BACKSP          ;REMOVE IT WITH BACKSPACE
B569 A9 20              LDA#    ' '             ;SPACE
B56B 20 4DB5            JSR     TVNOR
B56E A9 08      BACKSP: LDA#    BSPCOD          ;BACKSPACE
B570 D0 DB              BNE     TVNOR           ;OUTPUT AND RETURN

B572 20 9CFC    DOCR:   JSR     CLREOL          ;CLEAR TO END
B575 A9 00              LDA#    0               ;INDICATE HORIZ POSITION OF ZERO
B577 F0 17              BEQ     HOOK            ;MOVE THERE THEN RETURN

                ;TV CURSOR ADDRESSING ROUTINE.
                ;CALLED WITH ROW IN ACCUMULATOR, COLUMN IN Y REGISTER.

B579 C9 18      CONCUR: CMP#    24              ;IS IT WITHIN RANGE?
B57B 90 06              BCC     VEROK           ;BRANCH IF YES
B57D 38                 SEC                     ;NO, THEN WRAP AROUND
B57E E9 18              SBC#    24
B580 4C 79B5            JMP     CONCUR
B583 85 25      VEROK:  STA     CV              ;SET VERTICAL POSITION
B585 98                 TYA                     ;GET THE COLUMN NUMBER
B586 C9 28      HORR:   CMP#    40              ;IS IT ON SCREEN?
B588 90 06              BCC     HOOK            ;YES
B58A 38                 SEC                     ;NO, WRAP AROUND
B58B E9 28              SBC#    40
B58D 4C 86B5            JMP     HORR
B590 85 24      HOOK:   STA     CH              ;SET HORIZONTAL POSITION
B592 20 EFB4            JSR     CHKUSR          ;CHECK FOR USER ABORT
B595 20 22FC            JSR     APVTAB          ;EXECUTE VTAB ROUTINE
B598 18                 CLC                     ;INDICATE SUCCESS
B599 60                 RTS                     ;AND RETURN
```

```
                        .DEF    HERE=$B5C0
                        .LOC    HERE,HERE-BIAS

                ;THIS IS THE LINE BUFFERED CONSOLE HANDLER

B5C0 4C CFB5    LINCON: JMP     OPENLN          ;OPEN FOR INPUT
B5C3 4C 83B4            JMP     CONOPO          ;OPEN FOR OUTPUT
B5C6 4C DEB5            JMP     GETCH           ;GET A BYTE
B5C9 4C 89B4            JMP     CONOUT          ;SEND A BYTE
B5CC 4C DCB5            JMP     NULLOP          ;CLOSE


                ;OPEN FOR INPUT, I.E. FLUSH THE LINE:

B5CF 20 80B4    OPENLN: JSR     CONOPI          ;OPEN NORMAL CONSOLE
B5D2 AD 5DBF            LDA     EXECUT          ;EXEC MODE?
B5D5 F0 05              BEQ     NULLOP          ;NO FLUSH IF EXEC MODE
B5D7 A9 FF8D            MOV#    $FF,LINIDX      ;FLUSH BUFFER
B5DC 18         NULLOP: CLC                     ;SAY OK
B5DD 60                 RTS

                ;GET A CHARACTER:

B5DE AD 5ABF    GETCH:  LDA     LINIDX          ;PICK UP LINE BUFFER INDEX
B5E1 C9 FF              CMP#    $FF             ;ANY CONTENT?
B5E3 F0 2E              BEQ     NEWLIN          ;NO, GO GET SOME
B5E5 AA                 TAX                     ;YES, GET A BYTE
B5E6 BD 0002            LDAX    IN              ;FROM INPUT BUFFER
B5E9 EE 5ABF            INC     LINIDX          ;FOR NEXT CHAR
B5EC C9 1A              CMP#    EOFCOD          ;END OF FILE?
B5EE F0 0A              BEQ     ENDFIL          ;TERMINATE EXEC MODE
B5F0 C9 0D              CMP#    CRCOD           ;WAS A RETURN?
B5F2 F0 11              BEQ     ENDLN           ;YES, EMPTY THE LINE
B5F4 C9 0A              CMP#    LFCOD           ;LINE FEED?
B5F6 F0 E6              BEQ     GETCH           ;YES, IGNORE IT
B5F8 18                 CLC                     ;SUCESS
B5F9 60                 RTS                     ;BACK HOME

                ;WHEN WE ENCOUTER AN END-OF-FILE:
B5FA A9 FF      ENDFIL: LDA#    $FF             ;SET EXEC FLAG TO FALSE=$FF
B5FC 8D 5DBF            STA     EXECUT
B5FF 8D 5ABF            STA     LINIDX          ;SET INPUT LINE EMPTY
B602 4C DEB5            JMP     GETCH           ;AND GO GET FROM KEYBOARD

                ;WHEN WE ENCOUTER A RETURN CHARACTER
B605 AD 5DBF    ENDLN:  LDA     EXECUT          ;CONTINUE IF EXEC MODE
B608 F0 05              BEQ     NOEND
B60A A9 FF8D            MOV#    $FF,LINIDX      ;NOT EXEC SO FLAG IT EMPTY
B60F A9 0D      NOEND:  LDA#    CRCOD           ;GIVE CALLER THE RETURN
B611 18                 CLC
B612 60                 RTS

B613 20 1EB6    NEWLIN: JSR     GETLN           ;GET A LINE
B616 A9 008D            MOV#    0,LINIDX        ;RESET POINTER
```

```
B61B 4C DEB5            JMP     GETCH           ;BACK IN LINE

                ;NORMAL ENTRY POINT TO GET A LINE IS HERE:

B61E A9 008D    GETLN:  MOV#    0,LINIDX        ;RESET POINTER
B623 20 8EB6    NXTCHR: JSR     RDCHAR          ;GO READ A CHAR
B626 C9 15              CMP#    RETYPE          ;IS IT A RETYPE CHAR?
B628 D0 13              BNE     ADDINP          ;NO, GO ON
                ;WAS RETYPE SO WE GOTTA FIX SCREEN TO NORMAL CHAR
B62A A4 24              LDY     CH
B62C B1 28              LDA@Y   BASL            ;YES, PICK UP CHAR FROM SCREEN
B62E C9 80              CMP#    $80             ;IS IT NORMAL?
B630 B0 0B              BGE     ADDINP          ;YES, FINE
B632 C9 40              CMP#    $40             ;WAS IT INVERSE (LOWER CASE)
B634 B0 05              BGE     CTST2           ;NO MUST BE A BLINKER (CTRL)
B636 09 60              ORA#    $60             ;YES, FORCE NORMAL LOWER CASE
B638 4C 3DB6            JMP     ADDINP          ;GO INSERT IT
B63B 29 1F      CTST2:  AND#    $1F             ;MAKE NORMAL CONTROL CHAR
                ;GOT IT, EITHER FOR REAL OR FROM SCREEN
B63D 29 7F      ADDINP: AND#    $7F             ;WE WANT NORMAL ASCII
B63F AE 5ABF            LDX     LINIDX          ;PICK UP THE INDEX
B642 9D 0002            STAX    IN              ;STUFF CHARACTER INTO BUFFER
B645 C9 0D              CMP#    CRCOD           ;WAS IT RETURN?
B647 D0 04              BNE     NOTCR           ;NO, PROCEED
B649 20 83B6            JSR     CROUT           ;YES, JUST ECHO THE RETURN
B64C 60                 RTS                     ;AND EXIT

B64D 20 68B6    NOTCR:  JSR     ECHO            ;ECHO THE CHARACTER
B650 C9 7F              CMP#    RUBCHR          ;WAS IT BACKSPACE?
B652 F0 24              BEQ     BCKSPC          ;YES, DO THAT
B654 C9 18              CMP#    LINDEL          ;WAS IT LINE DELETE?
B656 F0 66              BEQ     CANCEL          ;YES, CANCEL THE LINE
B658 AE 5ABF            LDX     LINIDX          ;NEAR END OF BUFFER?
B65B E0 F8              CPX#    $F8             ;248 TH CHAR?
B65D 90 03              BLT     NOTCR1          ;NO, FINE
B65F 20 3AFF            JSR     BELL            ;YES, WARN HIM
B662 EE 5ABF    NOTCR1: INC     LINIDX          ;READY FOR NEXT
B665 4C 23B6            JMP     NXTCHR

B668 48         ECHO:   PHA
B669 C9 18              CMP#    LINDEL          ;WAS IT LINE CANCEL?
B66B F0 09              BEQ     ECHO1           ;DONT ECHO CANCEL
B66D C9 7F              CMP#    RUBCHR          ;WAS IT "RUBOUT"?
B66F D0 02              BNE     ECHO2           ;NO, NORMAL ECHO
B671 A9 08              LDA#    BSPCOD          ;YES, DO BACKSPACE
B673 20 89B4    ECHO2:  JSR     CONOUT          ;ECHO IT
B676 68         ECHO1:  PLA
B677 60                 RTS

B678 AD 5ABF    BCKSPC: LDA     LINIDX          ;GET THE INDEX
B67B F0 46              BEQ     GETLNZ          ;LINE EMPTY? IF SO NEW LINE
B67D CE 5ABF            DEC     LINIDX          ;DROP BACK ONE PLACE
B680 4C 23B6            JMP     NXTCHR          ;BACK FOR MORE

B683 A9 0D      CROUT:  LDA#    CRCOD           ;GET A RETURN
B685 20 89B4            JSR     CONOUT          ;PUT IT OUT & RETURN TO CALLER
```

```
B688 A9 0A              LDA#    LFCOD           ;ALSO NEEDS A LINE FEED
B68A 20 89B4            JSR     CONOUT          ;SINCE CONSOLE HANDLER
B68D 60                 RTS                     ;DOES BOTH CORRECTLY

B68E 20 86B4    RDCHAR: JSR     CONIN           ;GET ONE
B691 C9 1B              CMP#    ESCCOD          ;ESCAPE?
B693 F0 01              BEQ     ESC             ;YES, GO PROCESS
B695 60                 RTS                     ;NO, RETURN IT

B696 20 86B4    ESC:    JSR     CONIN           ;GET ARGUMENT CHARACTER
B699 20 9FB6            JSR     ESCDO           ;PROCESS IT
B69C 4C 8EB6            JMP     RDCHAR          ;AND GET ANOTHER CHAR

B69F C9 4E      ESCDO:  CMP#    'N              ;CHECK >= N
B6A1 B0 15              BGE     ESCOLD          ;YES SO OLD WAY
B6A3 C9 49              CMP#    'I              ;CHECK < I
B6A5 90 11              BLT     ESCOLD          ;YES SO OLD WAY
B6A7 C9 4C              CMP#    'L              ;IS IT L?
B6A9 F0 0D              BEQ     ESCOLD          ;YES, DO IT THE OLD WAY
B6AB A8                 TAY                     ;USE CHAR AS INDEX
B6AC B9 80B6            LDAY    XLTBL-'I        ;TRANSLATE TO OLD FORM
B6AF 20 B8B6            JSR     ESCOLD          ;AND DO OLD WAY
B6B2 20 86B4            JSR     CONIN           ;PICK UP NEXT TRY
B6B5 4C 9FB6            JMP     ESCDO           ;KEEP THIS MODE

B6B8 09 80      ESCOLD: ORA#    $80             ;HARDWARE WIERD WOZNIAK
B6BA 38                 SEC                     ;SOFTWARE WIERD WOZNIAK
B6BB 4C 2CFC            JMP     ESC1            ;GO THRU TO EITHER ROM

B6BE A9 5C      CANCEL: LDA#    '\              ;SHOW WE DROPPED THE LINE
B6C0 20 89B4            JSR     CONOUT
B6C3 20 83B6    GETLNZ: JSR     CROUT           ;GIVE RETURN, AND REENTER
B6C6 4C 1EB6            JMP     GETLN

B6C9 44         XLTBL:  .BYTE   'D
B6CA 42                 .BYTE   'B
B6CB 41                 .BYTE   'A
B6CC FF                 .BYTE   $FF
B6CD 43                 .BYTE   'C

                ;NOW HOOK THE HANDLER INTO THE DEVICE HANDLER TABLE:
                        .LOC    DEVTAB,DEVTAB-BIAS
BFC0 C0 B5              .WORD   LINCON          ;BUFFERED CONSOLE = DEVICE 1
BFC2 80 B4              .WORD   CONOPI          ;CONSOLE = DEVICE 0
```

*** ASM65 *** (V3.0-DS) OCT-79

```
                        .PAGE
                        .END

.       BFC4            ADDINP  B63D            APBELL  FBDD
APHOME  FC58            APPTVO  FBFD            APVTAB  FC22
BACKSP  B56E            BASL    0028            BCKSPC  B678
BELCOD  0007            BELL    FF3A            BIAS    9000
BSLASH  B4CD            BSPCOD  0008            CANCEL  B6BE
CASSHF  001D            CH      0024            CHKCTL  B545
CHKURT  B504            CHKUSR  B4EF            CLREOL  FC9C
CONCLO  B4E3            CONCUR  B579            CONIN   B486
CONOPI  B480            CONOPO  B483            CONOUT  B489
CRCOD   000D            CROUT   B683            CTST2   B63B
CV      0025            DEVTAB  BFC0            DIRTV   B53D
DOBLSH  000F            DOBRAK  000B            DOCR    B572
DOFORM  B554            DOTAB   B559            ECHO    B668
ECHO1   B676            ECHO2   B673            ENDFIL  B5FA
ENDLN   B605            EOFCOD  001A            ESC     B696
ESC1    FC2C            ESCCOD  001B            ESCDO   B69F
ESCOLD  B6B8            EXECUT  BF5D            FINEX   B50E
FORMCD  000C            GETCH   B5DE            GETKY0  B495
GETLN   B61E            GETLNZ  B6C3            HALTER  0013
HERE    B5C0            HIAD    B480            HOOK    B590
HORR    B586            IN      0200            KBABOR  0019
KBEXIT  0003            KEYINI  B4E5            KEYINS  B4A1
KEYRET  B4C0            KEYSAM  B4CB            KYWAIT  B4FC
LBRACK  B4D1            LFCOD   000A            LINCON  B5C0
LINDEL  0018            LINIDX  BF5A            LOAD    2480
LOWER   BF5E            NEWLIN  B613            NOEND   B60F
NOHNG   B506            NOHNG1  B511            NOTCR   B64D
NOTCR1  B662            NOTV    B552            NULLOP  B5DC
NXTCHR  B623            OPENLN  B5CF            RDCHAR  B68E
RETYPE  0015            RUBCHR  007F            RUBOUT  B566
SAVEX   B519            SWITCH  B4D5            TABCOD  0009
TOTV    B54F            TVINI   B4E0            TVNOR   B54D
TVOUT   B51C            VABORT  BF0C            VEROK   B583
VEXIT   BF06            XKBDAT  C000            XKBINP  FD1B
XKBRES  C010            XLTBL   B6C9            XTEXTR  C051

NO ERRORS DETECTED
```

```
                        ;       APEX BYTE I/O TO DISK FILES HANDLER

                        ;NOTE: THIS HANDLER ASSUMES THAT THE BUFFERS START ON
                        ;A PAGE BOUNDARY AND ARE AN INTEGRAL NUMBER OF PAGES LONG.

                        ;DEFINE THE DEVICE NUMBER WE WANT THIS TO BE:
                                .DEF    DEVNO=3         ;BY CONVENTION IT SHOULD BE 3

                        ;DEFINE THE PLACE IN MEMORY WE WANT THIS HANDLER TO GO:
                                .DEF    HIAD=$AD00      ;REALLY HERE
                                .DEF    LOAD=$1D00      ;BUT LOADS HERE

                        ;EXTERNALS WE CARE ABOUT SEE SYSTEM PAGE DEFINITION:
                                .DEF    KREAD=$BFE2     ;READ DISK BLOCKS
                                .DEF    KWRITE=$BFE5    ;WRITE DISK BLOCKS
                                .DEF    FSCAN=$BFDC     ;LOOKUP A FILE

                        ;LOCATIONS IN PROGRAM AREA THAT SPECIFY THE BUFFERS WE MUST USE
                                .DEF    INBUFE=$BF3C    ;INPUT BUFFER
                                .DEF    INBUFD=$BF3A
                                .DEF    OTBUFE=$BF38    ;OUTPUT BUFFER
                                .DEF    OTBUFD=$BF36

                        ;LOCATIONS IN THE SYSTEM AREA THAT CARRY READ/WRITE PARAMETERS
                                .DEF    UNIT=$BF68      ;UNIT TO DO XFER TO/FROM
                                .DEF    BLKNO=$BF69     ;BLOCK NUMBER TO START XFER
                                .DEF    NBLKS=$BF6B     ;NUMBER OF BLOCKS TO XFER
                                .DEF    FADDR=$BF6C     ;MEMORY ADDRESS TO XFER TO/FROM
                                .DEF    ENDBLK=$BF6E    ;LAST BLOCK OF FOUND FILE

                        ;LOCATIONS IN THE SYSTEM AREA THAT SPECIFY THE INPUT FILE
                                .DEF    INLBLK=$BF78    ;FIRST BLOCK
                                .DEF    INHBLK=$BF7A    ;LAST BLOCK
                                .DEF    INFLG=$BF7C     ;STATUS
                                .DEF    INDEV=$BF7E     ;UNIT IT IS ON

                        ;LOCATIONS IN THE SYSTEM AREA THAT SPECIFY THE OUTPUT FILE
                                .DEF    OTLBLK=$BF70    ;FIRST BLOCK
                                .DEF    OTHBLK=$BF72    ;MAXIMUM BLOCK WE CAN USE
                                .DEF    OTFLG=$BF74     ;STATUS
                                .DEF    OTDEV=$BF76     ;UNIT IT IS ON

                        ;WHERE THE DEVICE HANDLER TABLE IS SO WE CAN PATCH INTO IT:
                                .DEF    DEVTAB=$BFC0

                        ;A PLACE IF PAGE ZERO WE CAN USE FOR A POINTER
                                .DEF    PNTR1=0


                                .LOC    HIAD,LOAD

                        ;ENTRY POINTS:
AD00 4C C9AD            BYTEIO: JMP     OPENIN          ;OPEN FOR INPUT
AD03 4C B9AE                    JMP     OPENOT          ;OPEN FOR OUTPUT
AD06 4C 12AD                    JMP     GETBYT          ;INPUT A BYTE
AD09 4C 1FAE                    JMP     PUTBYT          ;OUTPUT A BYTE
```

```
AD0C 4C 1AAF            JMP     CLOSE           ;CLOSE
AD0F 4C 53AF            JMP     SCAN            ;OPEN AN INPUT FILE BY NAME

                ;BASIC GET A BYTE ROUTINE:
AD12 AD 7CBF    GETBYT: LDA     INFLG           ;DO WE HAVE INPUT OPEN?
AD15 C9 55              CMP#    $55
AD17 F0 02              BEQ     GETB1           ;IF SO, CONTINUE
AD19 38         GETB0:  SEC                     ;ELSE RETURN, FAILED
AD1A 60                 RTS
AD1B AD 83AF    GETB1:  LDA     INBHI           ;FORM INBHI-INPNT
AD1E 38 ED81            SUB     INPNT           ;TO SEE IF POINTER
AD22 AD 84AF            LDA     INBHI+1         ;IS STILL INSIDE THE
AD25 ED 82AF            SBC     INPNT+1         ;INPUT BUFFER
AD28 B0 05              BGE     GETB2           ;OK IF INPNT<=INBHI
AD2A 20 47AD            JSR     RMORE           ;ELSE READ MORE IN
AD2D B0 EA              BCS     GETB0           ;BARF IF TROUBLE
AD2F A0 00      GETB2:  LDY#    0               ;GET THE BYTE
AD31 AD 81AF            DMOV    INPNT,PNTR1     ;MOVE POINTER TO PAGE ZERO
AD3B B1 00              LDA@Y   PNTR1           ;VIA INPNT
AD3D EE 81AF            DINC    INPNT           ;AND BUMP POINTER
AD45 18                 CLC                     ;ALL IS WELL
AD46 60                 RTS

                ;READ ANOTHER BUFFER FULL IN

AD47 AD 7DAF    RMORE:  LDA     LIBLK           ;FORM INSIZ:=LIBLK-FIBLK
AD4A 38 ED7B            SUB     FIBLK           ;TO SEE HOW MUCH OF THE FILE
AD4E 8D 7FAF            STA     INSIZ           ;IS LEFT
AD51 AD 7EAF            LDA     LIBLK+1
AD54 ED 7CAF            SBC     FIBLK+1
AD57 8D 80AF            STA     INSIZ+1
AD5A B0 02              BCS     RMOR1           ;IS FILE USED UP?
AD5C 38         RMOR0:  SEC                     ;BARF EXIT
AD5D 60                 RTS
AD5E AD 80AF    RMOR1:  LDA     INSIZ+1         ;WILL IT FIT IN BUFFER?
AD61 D0 08              BNE     RMOR2           ;NO, >256 BLOCKS
AD63 AD 7FAF            LDA     INSIZ
AD66 CD 87AF            CMP     INBSIZ
AD69 90 0F              BLT     RMOR3           ;YES, IT FITS
                ;IF FILE IS >= BUFFER:
AD6B AD 86AF    RMOR2:  MOV     INBLO+1,FADDR+1 ;SETUP WHOLE BUFFER
AD71 AD 87AF            MOV     INBSIZ,NOBLK    ;FULL SIZE TANSFER
AD77 4C 8DAD            JMP     RMOR4           ;ENTER COMMON CODE
                ;IF FILE IS < BUFFER:
AD7A AD 84AF    RMOR3:  LDA     INBHI+1         ;COMPUTE WHERE TO PUT IT
AD7D 38 ED7F            SUB     INSIZ
AD81 8D 6DBF            STA     FADDR+1
AD84 AD 7FAF            MOV     INSIZ,NOBLK     ;SIZE OF FILE-1
AD8A EE 96AF            INC     NOBLK           ;+1=NO BLKS TO XFER
                ;COMMON CODE
AD8D A9 008D    RMOR4:  MOV#    0,FADDR         ;MUST BE ON  A PAGE
AD92 AD 7BAF            DMOV    FIBLK,BLKNO     ;SETUP BLKNO
AD9E AD 6CBF            DMOV    FADDR,INPNT     ;RESET POINTERS
ADAA AD 96AF            LDA     NOBLK           ;ADJUST NEXT BLOCK
ADAD 18 6D7B            DADM    FIBLK
ADB9 AD 96AF            MOV     NOBLK,NBLKS     ;GO GET IT
```

```
ADBF AD 88AF            MOV     INUNIT,UNIT
ADC5 20 E2BF            JSR     KREAD
ADC8 60                 RTS                     ;RETURN WITH CORRECT CARRY

                ;OPEN INPUT FILE:

ADC9 AD 7CBF    OPENIN: LDA     INFLG           ;DO WE HAVE ONE?
ADCC D0 02              BNE     OPIN1           ;NON ZERO IF OK
ADCE 38                 SEC                     ;ERROR, NO FILE TO OPEN
ADCF 60                 RTS
ADD0 AD 78BF    OPIN1:  DMOV    INLBLK,FIBLK    ;SETUP FIBLK
ADDC AD 7ABF            DMOV    INHBLK,LIBLK    ;AND LIBLK
ADE8 AD 3CBF            DMOV    INBUFE,INBHI    ;LOCAL COPY
ADF4 AD 3ABF            DMOV    INBUFD,INBLO
AE00 AD 84AF            LDA     INBHI+1         ;COMPUTE INBSIZ
AE03 38 ED86            SUB     INBLO+1
AE07 18 6901            INCA
AE0A 8D 87AF            STA     INBSIZ
AE0D AD 7EBF            MOV     INDEV,INUNIT    ;MAKE LOCAL COPY OF UNIT
AE13 A9 FF8D    OPIN2:  MOV#    $FF,INPNT+1     ;FLAG IT EMPTY
AE18 A9 558D            MOV#    $55,INFLG       ;FLAG IT OPEN
AE1D 18                 CLC
AE1E 60                 RTS

                ;OUTPUT A BYTE:

AE1F 8D 95AF    PUTBYT: STA     DATA            ;SAVE THE BYTE
AE22 AD 74BF            LDA     OTFLG           ;IS IT OPEN?
AE25 C9 55              CMP#    $55
AE27 F0 02              BEQ     PUTB1           ;WILL BE $55 IF IT IS
AE29 38         PUTB0:  SEC
AE2A 60                 RTS
AE2B AD 91AF    PUTB1:  LDA     OTBHI           ;FORM OTBHI-OTPNT
AE2E 38 ED8E            SUB     OTPNT           ;TO SEE IF THE POINTER IS
AE32 AD 92AF            LDA     OTBHI+1         ;STILL WITHIN THE
AE35 ED 8FAF            SBC     OTPNT+1         ;OUTPUT BUFFER
AE38 B0 05              BGE     PUTB2           ;OK IF OTPNT<=OTBHI
AE3A 20 5AAE            JSR     WMORE           ;ELSE WRITE BUF OUT
AE3D B0 EA              BCS     PUTB0           ;BARF IF TROUBLE
AE3F A0 00      PUTB2:  LDY#    0               ;STORE THE BYTE
AE41 AD 8EAF            DMOV    OTPNT,PNTR1     ;MOVE POINTER TO PAGE ZERO
AE4B AD 95AF            LDA     DATA
AE4E 91 00              STA@Y   PNTR1           ;VIA OTPNT
AE50 EE 8EAF            DINC    OTPNT           ;AND BUMP POINTER
AE58 18                 CLC                     ;ALL IS WELL
AE59 60                 RTS

                ;WRITE OUT A BUFFER

AE5A AD 8FAF    WMORE:  LDA     OTPNT+1         ;COMPUTE OTSIZ
AE5D 38 ED94            SUB     OTBLO+1         ;TO SEE HOW MANY BLOCKS
AE61 8D 8DAF            STA     OTSIZ           ;WE NEED TO WRITE OUT
AE64 AD 8EAF            LDA     OTPNT
AE67 F0 03              BEQ     WMOR1
AE69 EE 8DAF            INC     OTSIZ           ;ADJUST FOR PARTIAL BLOCK
AE6C AD 89AF    WMOR1:  DMOV    FOBLK,BLKNO     ;SETUP BLKNO
```

```
AE78 AD 94AF            LDA     OTBLO+1
AE7B 8D 6DBF            STA     FADDR+1
AE7E 8D 8FAF            STA     OTPNT+1         ;AND POINTER
AE81 A9 00              LDA#    0
AE83 8D 6CBF            STA     FADDR           ;MUST BE ON  A PAGE
AE86 8D 8EAF            STA     OTPNT
AE89 AD 8DAF            LDA     OTSIZ           ;UPDATE TO NEXT OUT BLOCK
AE8C 18 6D89            DADM    FOBLK
AE98 AD 8BAF            LDA     LOBLK           ;DOES THIS FIT IN THE EMPTY?
AE9B 38 ED89            SUB     FOBLK
AE9F AD 8CAF            LDA     LOBLK+1
AEA2 ED 8AAF            SBC     FOBLK+1
AEA5 B0 02              BGE     WMOR3           ;IF FOBLK <= LOBLK
AEA7 38                 SEC                     ;ATTEMPT TO WRITE TOO FAR
AEA8 60                 RTS                     ;DOES NOT FIT
AEA9 AD 8DAF    WMOR3:  MOV     OTSIZ,NBLKS     ;FITS, SO WRITE IT
AEAF AD 76BF            MOV     OTDEV,UNIT
AEB5 20 E5BF            JSR     KWRITE
AEB8 60                 RTS                     ;RETURN WITH STATUS

                ;OPEN OUTPUT FILE:

AEB9 AD 74BF    OPENOT: LDA     OTFLG           ;DO WE HAVE ONE?
AEBC C9 01              CMP#    $1              ;IF IT WAS NEW
AEBE F0 06              BEQ     OPOT1           ;THEN FINE
AEC0 C9 55              CMP#    $55             ;IF IT WAS NOT CLOSED
AEC2 F0 02              BEQ     OPOT1           ;THEN FINE
AEC4 38                 SEC                     ;ELSE BAD
AEC5 60                 RTS
AEC6 AD 70BF    OPOT1:  DMOV    OTLBLK,FOBLK    ;SETUP FOBLK
AED2 AD 72BF            LDA     OTHBLK          ;AND LOBLK
AED5 18 6901            INCA
AED8 8D 8BAF            STA     LOBLK
AEDB AD 73BF            LDA     OTHBLK+1
AEDE 69 00              ADC#    0
AEE0 8D 8CAF            STA     LOBLK+1
AEE3 AD 38BF            DMOV    OTBUFE,OTBHI    ;LOCAL COPY
AEEF AD 36BF            DMOV    OTBUFD,OTBLO    ;IN CASE USER PROGRAM BOMBS US
AEFB AD 92AF            LDA     OTBHI+1         ;COMPUTE OTBSIZ
AEFE 38 ED94            SUB     OTBLO+1
AF02 18 6901            INCA
AF05 8D 90AF            STA     OTBSIZ
AF08 AD 94AF            LDA     OTBLO+1         ;FLAG BUFFER EMPTY
AF0B 8D 8FAF            STA     OTPNT+1
AF0E A9 008D            MOV#    0,OTPNT
AF13 A9 558D            MOV#    $55,OTFLG       ;FLAG IT OPEN
AF18 18                 CLC
AF19 60                 RTS

                ;CLOSE THE OUT FILE

AF1A AD 74BF    CLOSE:  LDA     OTFLG           ;WAS IT OPEN?
AF1D C9 55              CMP#    $55
AF1F F0 02              BEQ     CLOS1           ;WAS OPEN, SO GO CLOSE IT
AF21 18                 CLC                     ;IT WAS ALREADY CLOSED
AF22 60                 RTS                     ;SO WE JUST IGNORE
```

```
AF23 A9 1A      CLOS1:  LDA#    $1A             ;STASH AN EOF, IN CASE
AF25 20 1FAE            JSR     PUTBYT
AF28 AD 8EAF            LDA     OTPNT           ;DO WE HAVE TO WRITE ANY?
AF2B D0 08              BNE     CLOS3           ;HAS CONTENT
AF2D AD 8FAF            LDA     OTPNT+1
AF30 CD 94AF            CMP     OTBLO+1
AF33 F0 06              BEQ     CLOS2           ;NO CONTENT, ALL DONE
AF35 20 5AAE    CLOS3:  JSR     WMORE           ;ELSE WRITE OUT FINAL BLOCKS
AF38 90 01              BCC     CLOS2           ;CHECK FOR ERRORS
AF3A 60                 RTS
AF3B AD 89AF    CLOS2:  LDA     FOBLK           ;UPDATE LAST BLOCK
AF3E 38 E901            SUB#    1               ;IN SYSTEM PAGE
AF41 8D 72BF            STA     OTHBLK          ;SO APEX WILL KNOW HOW
AF44 AD 8AAF            LDA     FOBLK+1         ;TO UPDATE THE DIRECTORY
AF47 E9 00              SBC#    0
AF49 8D 73BF            STA     OTHBLK+1
AF4C A9 FF              LDA#    $FF             ;MARK IT CLOSED
AF4E 8D 74BF            STA     OTFLG
AF51 18                 CLC
AF52 60                 RTS                     ;RETURN WITH CARRY FLAG CLEAR

                ;THIS ROUTINE USES THE SYSTEM RESIDENT "FSCAN" TO OPEN
                ;AN INPUT FILE BY NAME. THE NAME IS PASSED BY ADDRESS IN
                ;THE AC AND Y REGISTERS. THE FILE WILL BE LOOKED FOR
                ;ON THE UNIT IN THE SYSTEM AREA PARAMETER "UNIT".
                ;NOTE THAT THE FILE OPENED IN THIS WAY LEAVES THE INPUT
                ;FILE BLOCK IN THE SYSTEM AREA UNAFFECTED. THUS IF ANOTHER
                ;CALL IS MADE TO "OPENIN" IT WILL RE-OPEN THE ORIGINAL FILE.
                ;REFER TO THE COMMENTS ON FSCAN FOR FURTHER DETAILS.

AF53 20 DCBF    SCAN:   JSR     FSCAN
AF56 B0 21              BCS     BADSCN          ;FILE NOT FOUND
AF58 AD 69BF            DMOV    BLKNO,FIBLK     ;MOVE STARTING BLOCK
AF64 AD 6EBF            DMOV    ENDBLK,LIBLK    ;AND ENDING BLOCK
AF70 AD 68BF            MOV     UNIT,INUNIT     ;ALSO SAVE NEW UNIT
AF76 4C 13AE            JMP     OPIN2           ;GO JOIN EXISTING CODE
AF79 38         BADSCN: SEC
AF7A 60                 RTS

                ;THE RAM LOCS THAT THESE ROUTINES NEED

AF7B 00 00      FIBLK:  .WORD   0               ;NEXT BLOCK OF INPUT FILE
AF7D 00 00      LIBLK:  .WORD   0               ;LAST BLOCK OF THE INPUT FILE
AF7F 00 00      INSIZ:  .WORD   0               ;REMAINING INPUT FILE SIZE-1
AF81 00 00      INPNT:  .WORD   0               ;INPUT BUFFER POINTER
AF83 00 00      INBHI:  .WORD   0               ;LOCAL COPY START OF BUF
AF85 00 00      INBLO:  .WORD   0               ;LOCAL COPY, END OF BUF
AF87 00         INBSIZ: .BYTE   0               ;INPUT BUFFER SIZE IN PAGES
AF88 00         INUNIT: .BYTE   0               ;LOCAL COPY OF INPUT UNIT

AF89 00 00      FOBLK:  .WORD   0               ;NEXT BLOCK OF OUT FILE
AF8B 00 00      LOBLK:  .WORD   0               ;MAX+1 BLOCK OF OUT FILE
AF8D 00         OTSIZ:  .BYTE   0               ;SIZE FOR THIS WRITE
AF8E 00 00      OTPNT:  .WORD   0               ;OUTPUT BUFFER POINTER
AF90 00         OTBSIZ: .BYTE   0               ;OUTPUT BUFFER SIZE IN PAGES
AF91 00 00      OTBHI:  .WORD   0               ;LOCAL COPY, START OF BUF
```

```
AF93 00 00      OTBLO:  .WORD   0               ;LOCAL COPY, END OF BUF

AF95 00         DATA:   .BYTE   0               ;TEMPORARY TO SAVE DATA BYTE
AF96 00         NOBLK:  .BYTE   0               ;TEMP=NO OF BLOCKS IN XFER

                ;PS: THE LOCAL COPY STUFF IS TO SAVE US FROM LOADERS AND LINKED
                ;FILE READERS WHICH COULD CHANGE THINGS IN THE SYSTEM PAGE
                ;ON THE FLY.

                ;NOW HOOK THE HANDLER INTO THE DEVICE HANDLER TABLE:
                        .LOC    DEVTAB+DEVNO+DEVNO
BFC6 00 AD              .WORD   BYTEIO
```

*** ASM65 *** (V3.0-DS) OCT-79

```
                        .PAGE
                        .END

.       BFC8    BADSCN  AF79    BLKNO   BF69
BYTEIO  AD00    CLOS1   AF23    CLOS2   AF3B
CLOS3   AF35    CLOSE   AF1A    DATA    AF95
DEVNO   0003    DEVTAB  BFC0    ENDBLK  BF6E
FADDR   BF6C    FIBLK   AF7B    FOBLK   AF89
FSCAN   BFDC    GETB0   AD19    GETB1   AD1B
GETB2   AD2F    GETBYT  AD12    HIAD    AD00
INBHI   AF83    INBLO   AF85    INBSIZ  AF87
INBUFD  BF3A    INBUFE  BF3C    INDEV   BF7E
INFLG   BF7C    INHBLK  BF7A    INLBLK  BF78
INPNT   AF81    INSIZ   AF7F    INUNIT  AF88
KREAD   BFE2    KWRITE  BFE5    LIBLK   AF7D
LOAD    1D00    LOBLK   AF8B    NBLKS   BF6B
NOBLK   AF96    OPENIN  ADC9    OPENOT  AEB9
OPIN1   ADD0    OPIN2   AE13    OPOT1   AEC6
OTBHI   AF91    OTBLO   AF93    OTBSIZ  AF90
OTBUFD  BF36    OTBUFE  BF38    OTDEV   BF76
OTFLG   BF74    OTHBLK  BF72    OTLBLK  BF70
OTPNT   AF8E    OTSIZ   AF8D    PNTR1   0000
PUTB0   AE29    PUTB1   AE2B    PUTB2   AE3F
PUTBYT  AE1F    RMOR0   AD5C    RMOR1   AD5E
RMOR2   AD6B    RMOR3   AD7A    RMOR4   AD8D
RMORE   AD47    SCAN    AF53    UNIT    BF68
WMOR1   AE6C    WMOR3   AEA9    WMORE   AE5A
```

NO ERRORS DETECTED