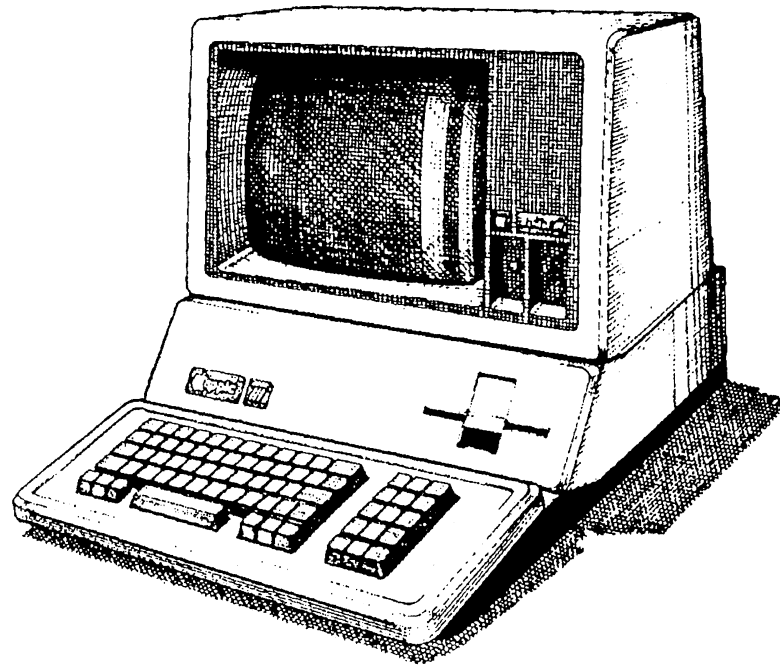




Apple /// Computer Technical Information

**Apple ///  
CONSOLE DRIVER  
Version 1.31  
Source Code Listing**



```

; #####
; #   PROJECT   :   Apple /// SOS Console Driver 1.31 (6502 Assembly Source Code)
; #   FILE NAME:   CONSOLE.TEXT
; #####

```

```

000001          .TITLE          "SOS Console Driver"
000002          .NOPATCHLIST
000003          .NOMACROLIST
000004
000005 ; -----
000006 ;
000007 ;           SOS Console Driver
000008 ;
000009 ;           Copyright (C) 1983 by Apple Computer Inc.
000010 ;           All Rights Reserved
000011 ;
000012 ;           Previous Copyright (C) 1980, 1981
000013 ;
000014 ;
000015 ;   Revisions:
000016 ;
000017 ;   1.00   14-Nov-80       Initial Release
000018 ;
000019 ;   1.12   23-Sep-81
000020 ;       Bug fixes:
000021 ;       Download 1-8 characters.
000022 ;       Download entire character set.
000023 ;       Include saved screen state in console state table.
000024 ;       Adjust all pointers for proper extended addressing.
000025 ;       Fix SYNC to monitor positive edge of vertical blanking.
000026 ;       Delete extraneous data returned by status calls 12, 13, & 14.
000027 ;       Fix erase option of character and line delete.
000028 ;   Extensions:
000029 ;       Add video toggle on control-5.
000030 ;       Add dump & restore contents of viewport.
000031 ;       Change keyboard transform table to include alpha-lock data.
000032 ;       Retain cursor on SYNC.
000033 ;
000034 ;   1.30   11-Jan-83
000035 ;       Bug fixes:
000036 ;       Wait for pending download on close.
000037 ;       Fix branch in 40 column horizontal shift right.
000038 ;       Fix cursor in dump & restore contents of viewport.
000039 ;       Disable interrupts while setting events and screen mode.

```

```

000040 ;           Extensions:
000041 ;           Turn on video iff buffer is empty.
000042 ;           Set bit 7 on control characters read from screen
000043 ;           (applies to char copy and screen read status).
000044 ;           Don't dump viewport when displaying control characters.
000045 ;           Add status request 9, read screen with normal/inverse flag.
000046 ;           1.31 17-Mar-83
000047 ;           Fix VERIFY to eliminate noise when setting screen switches.
000048 ;
000049 ;-----
000050
000051 DEVTYPE      .EQU      61
000052 SUBTYPE      .EQU      01
000053 APPLE       .EQU      0001
000054 RELEASE    .EQU      1310
000055           .PAGE
000056 ;-----
000057 ;
000058 ; The macro SWITCH performs an N way branch based on a switch index. The
000059 ; maximum value of the switch index is 127 with bounds checking provided
000060 ; as an option. The macro uses the A and Y registers and alters the C,
000061 ; Z, and N flags of the status register, but the X register is unchanged.
000062 ;
000063 ;           SWITCH [index], [bounds], adrs_table, [*]
000064 ;
000065 ;           index This is the variable that is to be used as the switch index.
000066 ;           If omitted, the value in the accumulator is used.
000067 ;
000068 ;           bounds This is the maximum allowable value for index. If index
000069 ;           exceeds this value, the carry bit will be set and execution
000070 ;           will continue following the macro. If bounds is omitted,
000071 ;           no bounds checking will be performed.
000072 ;
000073 ;           adrs_table This is a table of addresses (low byte first) used by the
000074 ;           switch. The first entry corresponds to index zero.
000075 ;
000076 ;           * If an asterisk is supplied as the fourth parameter, the
000077 ;           macro will push the switch address but will not exit to
000078 ;           it; execution will continue following the macro. The
000079 ;           program may then load registers or set the status before
000080 ;           exiting to the switch address.
000081 ;
000082 ;-----
000083 ;
000084           .MACRO      SWITCH

```

```

000085      .IF          "%1" <> ""          ;If PARM1 is present,
000086      LDA          %1                    ; Load A with switch index
000087      .ENDC
000088      .IF          "%2" <> ""          ;If PARM2 is present,
000089      CMP          #2+1                  ; Perform bounds checking
000090      BCS          $3579                  ; on switch index
000091      .ENDC
000092      ASL          A
000093      TAY
000094      LDA          %3+1,Y                  ;Get switch address from table
000095      PHA
000096      LDA          %3,Y
000097      PHA
000098      .IF          "%4" <> "*"          ;If PARM4 is omitted,
000099      RTS
000100      .ENDC
000101      .IF          "%2" <> ""          ;Otherwise, drop through
000102  $3579
000103      .ENDC
000104      .ENDM
000105
000106      .INCLUDE      :CONS.DAT1.TEXT
000107      .INCLUDE      :CONS.DAT2.TEXT
000108      .INCLUDE      :CONS.DAT3.TEXT
000109      .INCLUDE      :CONS.MAIN.TEXT
000110      .INCLUDE      :CONS.READ.TEXT
000111      .INCLUDE      :CONS.WRIT.TEXT
000112      .INCLUDE      :CONS.FCTN.TEXT
000113      .INCLUDE      :CONS.STAT.TEXT
000114      .INCLUDE      :CONS.CNTL.TEXT
000115      .INCLUDE      :CONS.DNLD.TEXT
000116      .INCLUDE      :CONS.MISC.TEXT
000117      .INCLUDE      :CONS.UTL1.TEXT
000118      .INCLUDE      :CONS.UTL2.TEXT
000119
000120      .END
000121
; #####
; #   END OF FILE:  CONSOLE.TEXT
; #   LINES       :  121
; #   CHARACTERS  :  6057
; #   Formatter   :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; #   Author      :  David T. Craig -- 71533.606@compuserve.com -- Santa Fe, New Mexico USA
; #####

```

```

; #####
; #   PROJECT : Apple /// SOS Console Driver 1.31 (6502 Assembly Source Code)
; #   FILE NAME: CONS.DAT1.TEXT
; #####

000001      .PROC      CONSOLE
000002      .WORD      0FFFF
000003      .WORD      59.
000004      .ASCII     "Console Driver -- "
000005      .ASCII     "Copyright (C) 1983 by Apple Computer Inc."
000006 ;-----
000007 ;
000008 ; Device Handler Identification Block
000009 ;
000010 ;-----
000011 ;
000012 IDBLK      .WORD      0000      ;Link to next device handler
000013      .WORD      CNSLDH      ;Entry point address
000014      .BYTE      8          ;Length of device name
000015      .ASCII     ".CONSOLE    "
000016      .BYTE      80,00,00      ;Device, Slot & Unit numbers
000017      .BYTE      DEVTYPE
000018      .BYTE      SUBTYPE
000019      .BYTE      00
000020      .WORD      0000
000021      .WORD      APPLE
000022      .WORD      RELEASE
000023      .WORD      00          ;No configuration block
000024      .PAGE
000025 ;-----
000026 ;
000027 ; Global Data:
000028 ;
000029 ; SUSPFLSH: Suspend and Flush Output Flags
000030 ;     7 => Suspend Output
000031 ;     6 => Flush Output
000032 ;
000033 ; SCRNMODE: Current Screen Mode
000034 ;     7 => Off / On
000035 ;     6 => Text / Graphics
000036 ;     2 => Page 1 / Page 2
000037 ;     1 => 40 Col / 80 Col
000038 ;     0 => B & W / Color
000039 ;

```

```
000040 ;
000041 ; State Flags:
000042 ;
000043 ; HMODE: Hardware Mode
000044 ; 7 => 40 Col / 80 Col
000045 ; 1 => 40 Col / 80 Col
000046 ; 0 => B & W / Color
000047 ;
000048 ; SMODE: Software Mode
000049 ; 5 => Normal / Inverse
000050 ; 4 => Disable / Enable Cursor
000051 ; 3 => Disable / Enable Scroll
000052 ; 2 => Disable / Enable Auto Carriage Return
000053 ; 1 => Disable / Enable Auto Line Feed
000054 ; 0 => Disable / Enable Auto Advance
000055 ;
000056 ;
000057 ; Permanant Zero Page Data:
000058 ;
000059 ; BASE1, BASE2: Screen Memory Pointers
000060 ; The base pointers point to the beginning of the current line. In
000061 ; 40 column mode, BASE1 points to the ASCII data while BASE2 points
000062 ; to the color information. In 80 column mode, BASE1 points to col-
000063 ; umn 0 of the viewport while BASE2 points to column 1.
000064 ;
000065 ;
000066 ; Temporary Zero Page Data:
000067 ;
000068 ; WORK1, WORK2:
000069 ; These pointers are used in conjunction with BASE1 and BASE2 for
000070 ; scrolling, shifting, etc.
000071 ;
000072 ; COUNT:
000073 ; Number of bytes read or written.
000074 ;
000075 ; ONEBYTE:
000076 ; Boolean flag for single byte read requests.
000077 ;
000078 ; BLANK:
000079 ; Holds an ASCII space in the current video mode (normal or inverse)
000080 ; for use in clearing the viewport.
000081 ;
000082 ; TEMPX:
000083 ; Temporary storage for X.
000084 ;
```

```
000085 ;   FLAGS:
000086 ;       Miscellaneous flags for use by SCROLL, SHIFT, SCRNDUMP, etc.
000087 ;
000088 ;   TEMP1, TEMP2, TEMP3, TEMP4:
000089 ;       General temporary storage for use by SCROLL, SHIFT, SCRNDUMP, etc.

; #####
; #   END OF FILE:  CONS.DAT1.TEXT
; #   LINES       :   89
; #   CHARACTERS  :  3512
; #   Formatter   :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; #   Author      :  David T. Craig -- 71533.606@compuserve.com -- Santa Fe, New Mexico USA
; #####
```

```

; #####
; #   PROJECT : Apple /// SOS Console Driver 1.31 (6502 Assembly Source Code)
; #   FILE NAME: CONS.DAT2.TEXT
; #####

```

```

000001          .PAGE
000002 ;
000003 ;   SOS Global Data & Subroutines
000004 ;
000005 SUSPFLSH      .EQU      1902          ;Suspend & Flush flags
000006 SCRNMODE     .EQU      1906          ;Current Screen Mode
000007 ALLOCSIR      .EQU      1913
000008 DEALCSIR      .EQU      1916
000009 QUEEVENT      .EQU      191F
000010 SYSERR       .EQU      1928
000011 ;
000012 ;   SOS Error Codes
000013 ;
000014 XREQCODE       .EQU      20           ;Invalid request code
000015 XCTLCODE       .EQU      21           ;Invalid controlstatus code
000016 XCTLPARM      .EQU      22           ;Invalid controlstatus parm
000017 XNOTOPEN       .EQU      23           ;Device not open
000018 XNOTAVIL      .EQU      24           ;Device not available
000019 XNORESRC       .EQU      25           ;Unable to obtain resource
000020 ;
000021 ;   Hardware I/O Addresses
000022 ;
000023 KAPORT          .EQU      0C000
000024 KBPORT        .EQU      0C008
000025 KYBDSTRB       .EQU      0C010
000026 KYBDCLR        .EQU      01          ;Clear keyboard interrupt flag
000027 KYBDSBL        .EQU      01          ;Disable keyboard interrupts
000028 KYBDENBL       .EQU      81          ;Enable keyboard interrupts
000029 BELL           .EQU      0C040
000030 VMODE0          .EQU      0C050          ;Video mode switches
000031 VMODE1          .EQU      0C052
000032 VMODE2          .EQU      0C054
000033 VMODE3          .EQU      0C056
000034 SCRLLSBL     .EQU      0C0D8          ;Disable graphics scroll
000035 DNLLDSBL       .EQU      0C0DA          ;Disable character download
000036 DNLDENBL      .EQU      0C0DB          ;Enable character download
000037 VBLCLR         .EQU      18          ;Clear both VBL interrupt flags
000038 VBLDSBL        .EQU      18          ;Disable both VBL interrupts
000039 VBLENBL        .EQU      90          ;Enable VBL interrupt on CB2

```



```

000040 E_REG          .EQU      0FFDF          ;Environment register
000041 E_IORB        .EQU      0FFE0          ;6522 input/output register B
000042 E_PCR        .EQU      0FFEC          ;6522 peripheral control register
000043 E_IFR        .EQU      0FFED          ;6522 interrupt flag register
000044 E_IER        .EQU      0FFEE          ;6522 interrupt mask register
000045 B_REG        .EQU      0FFEF          ;Bank register
000046 ;
000047 ; ASCII Equates and Special Keys
000048 ;
000049 ASC_NUL        .EQU      00              ;Null
000050 ASC_SOH        .EQU      01              ;Start of Header
000051 ASC_STX        .EQU      02              ;Start of Text
000052 ASC_ETX        .EQU      03              ;End of Text
000053 ASC_ENQ        .EQU      05              ;Enquiry
000054 ASC_ACK        .EQU      06              ;Acknowledgement
000055 ASC_BS        .EQU      08              ;Backspace
000056 ASC_HT        .EQU      09              ;Horizontal Tab
000057 ASC_LF        .EQU      0A              ;Line Feed
000058 ASC_VT        .EQU      0B              ;Vertical Tab
000059 ASC_FF        .EQU      0C              ;Form Feed
000060 ASC_CR        .EQU      0D              ;Carriage Return
000061 ASC_NAK        .EQU      15              ;Negative Acknowledge
000062 ASC_CAN        .EQU      18              ;Cancel
000063 ASC_ESC        .EQU      1B              ;Escape
000064 ASC_FS        .EQU      1C              ;File Separator
000065 ASC_GS        .EQU      1D              ;Group Separator
000066 ASC_US        .EQU      1F              ;Unit Separator
000067 ASC_SP        .EQU      20              ;Space
000068 LARROW        .EQU      ASC_BS          ;Left Arrow
000069 RARROW        .EQU      ASC_NAK        ;Right Arrow
000070 UARROW        .EQU      ASC_VT          ;Up Arrow
000071 DARROW        .EQU      ASC_LF          ;Down Arrow
000072 ;
000073 ; Miscellaneous Equates
000074 ;
000075 TRUE          .EQU      80
000076 FALSE        .EQU      00
000077 BITON0        .EQU      01
000078 BITON2        .EQU      04
000079 BITON3        .EQU      08
000080 BITON4        .EQU      10
000081 BITON5        .EQU      20
000082 BITON6        .EQU      40
000083 BITON7        .EQU      80
000084 BITOFF0       .EQU      0FE

```

```

000085 BITOFF4      .EQU      0EF
000086 BITOFF5      .EQU      0DF
000087 BITOFF7      .EQU      07F
000088 BUFMAX        .EQU      80          ;Maximum buffer size
000089 TEXTCSA      .EQU      0C00       ;Text character set address
000090              .PAGE
000091 ;-----
000092 ;
000093 ;   SOS Device Handler Interface
000094 ;
000095 ;-----
000096 ;
000097 SOSINT           .EQU      0C0
000098 REQCODE          .EQU      SOSINT+0    ;SOS request code
000099 BUFPTR           .EQU      SOSINT+2    ;Buffer pointer
000100 REQCNT           .EQU      SOSINT+4    ;Requested count
000101 RTNCNT           .EQU      SOSINT+8    ;Returned count
000102 SCCODE          .EQU      SOSINT+2    ;Status / Control code
000103 SCLIST           .EQU      SOSINT+3    ;Status / Control list
000104 ;
000105 ;
000106 ;-----
000107 ;
000108 ;   Zero Page Data (preserved) and Zero Page Save Area
000109 ;
000110 ;-----
000111 ;
000112 ZPDATA           .EQU      SOSINT+10.
000113 BASEPTRS         .EQU      ZPDATA+0    ;Screen memory base pointers
000114 BASE1            .EQU      BASEPTRS+0  ; even col. / text bytes
000115 BASE2            .EQU      BASEPTRS+2  ; odd col. / color bytes
000116 ZPLENGTH        .EQU      4
000117 ;
000118 ZPSAVE           .BLOCK    ZPLENGTH
000119 ;
000120 ;
000121 ;-----
000122 ;
000123 ;   Zero Page Data (temporary)
000124 ;
000125 ;-----
000126 ;
000127 WORKPTRS         .EQU      ZPDATA+ZPLENGTH
000128 WORK1            .EQU      WORKPTRS+0
000129 WORK2            .EQU      WORKPTRS+2

```

```
000130 COUNT      .EQU      WORKPTRS+4      ;Current I/O count
000131 ONEBYTE    .EQU      COUNT+2          ;One byte console read flag
000132 BLANK      .EQU      ONEBYTE+1
000133 TEMPX      .EQU      BLANK+1
000134 FLAGS      .EQU      TEMPX+1
000135 TEMP1      .EQU      FLAGS+1
000136 TEMP2      .EQU      TEMP1+1
000137 TEMP3      .EQU      TEMP2+1
000138 TEMP4      .EQU      TEMP3+1
```

```
; #####
; #   END OF FILE:  CONS.DAT2.TEXT
; #   LINES       :  138
; #   CHARACTERS  :  7595
; #   Formatter   :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; #   Author      :  David T. Craig -- 71533.606@compuserve.com -- Santa Fe, New Mexico USA
; #####
```

```

; #####
; #   PROJECT   :   Apple /// SOS Console Driver 1.31 (6502 Assembly Source Code)
; #   FILE NAME :   CONS.DAT3.TEXT
; #####

000001          .PAGE
000002 ;-----
000003 ;
000004 ;   Console State Table
000005 ;
000006 ;-----
000007 ;
000008 CONSTTBL    .EQU          *                ;Console state table
000009 ;
000010 ANYKEYEVNT   .BLOCK        5                ;Any Key Event parameters
000011 ATTNEVNT     .BLOCK        5                ;Attention Event parameters
000012 ATTNCHAR     .BYTE         0                ;Attention character
000013 ;
000014 DFLTBL      .EQU          *                ;This block initialized from default values
000015 ;
000016 KYBDMODE     .BYTE         0                ;Console/Keyboard mode flag
000017 NEWLINE      .BYTE         0                ;New Line flag
000018 NEWLNCHR     .BYTE         0                ;New Line character
000019 NOWAIT       .BYTE         0                ;No Wait flag
000020 ECHO         .BYTE         0                ;Screen Echo flag
000021 CHCPYFLG     .BYTE         0                ;Character Copy flag
000022 CHCPYCHR     .EQU          ASC_NAK          ;Character Copy character
000023 CHDELFLG     .BYTE         0                ;Character Delete flag
000024 CHDELCHR     .EQU          ASC_BS          ;Character Delete character
000025 LNDELFLG     .BYTE         0                ;Line Delete flag
000026 LNDELCHR     .EQU          ASC_CAN        ;Line Delete character
000027 ESCAPE      .BYTE         0                ;Escape Mode flag
000028 ;
000029 SCRSTTBL    .EQU          *                ;Screen state table
000030 ;
000031 HMODE        .BYTE         0                ;Hardware mode
000032 SMODE        .BYTE         0                ;Software mode
000033 TPX          .BYTE         0                ;Text position
000034 TPY          .BYTE         0
000035 VPL          .BYTE         0                ;Viewport
000036 VPR          .BYTE         79.
000037 VPT          .BYTE         0
000038 VPB          .BYTE         23.
000039 TCF          .BYTE         0F             ;Text color

```

```

000040 TCB          .BYTE      00
000041 ;
000042 SCRSTLEN    .EQU      *-SCRSTTBL
000043 ;
000044 DFLTLEN      .EQU      *-DFLTTBL
000045 ;
000046 SCRSTSAV    .BLOCK     SCRSTLEN          ;Saved screen state table
000047 ;
000048 CONSTLEN     .EQU      *-CONSTTBL
000049 .PAGE
000050 ;-----
000051 ;
000052 ;  Default Values for State Table
000053 ;
000054 ;-----
000055 ;
000056 DFLTVAL      .BYTE      FALSE          ;Console / Keyboard flag
000057 .BYTE      FALSE          ;Newline flag
000058 .BYTE      ASC_CR        ;Newline character
000059 .BYTE      FALSE        ;Nowait flag
000060 .BYTE      TRUE         ;Screen echo flag
000061 .BYTE      TRUE         ;Character copy flag
000062 .BYTE      TRUE         ;Character delete flags
000063 .BYTE      TRUE         ;Line delete flags
000064 .BYTE      TRUE         ;Escape mode flags
000065 .BYTE      02           ;Hardware mode
000066 .BYTE      0D           ;Software mode
000067 .BYTE      0.           ;Cursor position
000068 .BYTE      0.
000069 .BYTE      0.           ;Viewport
000070 .BYTE      79.
000071 .BYTE      0.
000072 .BYTE      23.
000073 .BYTE      0F           ;Text colors
000074 .BYTE      00
000075 .PAGE
000076 ;-----
000077 ;
000078 ;  Private Variable Storage
000079 ;
000080 ;-----
000081 ;
000082 KYBDBUFS     .EQU      01500          ;Type ahead buffers
000083 KABUF       .EQU      KYBDBUFS
000084 KBBUF       .EQU      KYBDBUFS+BUFMAX

```

```

000085 XFORMTBL      .EQU          01700          ;Keyboard transform table
000086 ;
000087 KADATA        .BYTE          0              ;Temp Storage
000088 KBDATA        .BYTE          0              ; for Interrupt Processing
000089 ;
000090 KEYCNT         .BYTE          0              ;Buffered keystroke count
000091 BUFSIZ        .BYTE          0              ;Current buffer size
000092 BUFHEAD       .BYTE          0              ;Index of first character
000093 BUFTAIL       .BYTE          0              ;Index of last character
000094 ;
000095 OPENFLG       .BYTE          0              ;Device open flag
000096 READING       .BYTE          0              ;Read in progress flag
000097 DSPLYCTL      .BYTE          0              ;Display control characters
000098 ;
000099 SMFLAGS       .EQU          *
000100 SMINV         .BYTE          0              ;Inverse video
000101 SMCURSOR     .BYTE          0              ;Cursor enabled
000102 SMSROLL      .BYTE          0              ;Scroll flag
000103 SMAUTOOCR    .BYTE          0              ;Auto CR
000104 SMAUTOLF     .BYTE          0              ;Auto LF
000105 SMAUTOADV    .BYTE          0              ;Auto advance
000106 ;
000107 VPHMAX       .BYTE          79.            ;viewport maximum horizontal index
000108 VPVMAX       .BYTE          23.            ;viewport maximum vertical index
000109 TCOLOR       .BYTE          0F0           ;text fg/bg color byte
000110 ;
000111 CTLINDX      .BYTE          0              ;function buffer index
000112 CTLBUFF      .BLOCK          8              ;control function buffer
000113 CTLQUOTA     .BYTE          0              ;parameter quota
000114 ;
000115 DNLDFLG     .BYTE          00             ;Bit 7=Active, Bit 6=Request
000116 DNLDCEL     .BYTE          00             ;Current download cell number
000117 DNLDCHR     .BYTE          00             ;Current download ASCII code
000118 DNLDIMG     .WORD          0000           ;Pointer to character image
000119 .PAGE
000120 ;-----
000121 ;
000122 ; Addresses used as subroutine parameters and SIR request tables
000123 ;
000124 ;-----
000125 ;
000126 ANYKYPARM    .WORD          ANYKYEVRT
000127 ATTNPARM    .WORD          ATTNEVRT
000128 ;
000129 KYBDSADR    .WORD          KYBDSTBL

```

```

000130 KYBDSTBL      .BYTE      2,0                ;Keyboard interrupt
000131              .WORD      KYBDMIH
000132 KYBDBANK     .BYTE      0
000133 KYBDSSIZ     .EQU      *-KYBDSTBL
000134 ;
000135 DNLDADR       .WORD      DNLDSTBL
000136 DNLDSTBL      .BYTE      5,0,0,0,0          ;VBL positive
000137              .BYTE      6,0                ;VBL negative
000138              .WORD      DNLDINT
000139 DNLDDBANK     .BYTE      0
000140              .BYTE      10,0,0,0,0        ;Character download / Graphics scroll
000141 DNLDSSIZ      .EQU      *-DNLDSTBL
000142 ;
000143 SYNCADR        .WORD      SYNCSTBL
000144 SYNCSTBL       .BYTE      5,0,0,0,0        ;VBL positive
000145 SYNCSSIZ      .EQU      *-SYNCSTBL
000146 ;
000147 ;
000148 ;-----
000149 ;
000150 ;   Base Calculator Address Tables
000151 ;
000152 ;-----
000153 ;
000154 BASL           .BYTE      000,080,000,080
000155              .BYTE      000,080,000,080
000156              .BYTE      028,0A8,028,0A8
000157              .BYTE      028,0A8,028,0A8
000158              .BYTE      050,0D0,050,0D0
000159              .BYTE      050,0D0,050,0D0
000160 BASH          .BYTE      004,004,005,005
000161              .BYTE      006,006,007,007
000162              .BYTE      004,004,005,005
000163              .BYTE      006,006,007,007
000164              .BYTE      004,004,005,005
000165              .BYTE      006,006,007,007
000166              .PAGE
000167 ;-----
000168 ;
000169 ;   Escape Command and Escape Operator Tables
000170 ;
000171 ;-----
000172 ;
000173 ESCCMD         .BYTE      "B"                ;Viewport bottom right
000174              .BYTE      "T"                ;Viewport top left

```

```

000175      .BYTE      "V"          ;Clear Viewport
000176      .BYTE      "S"          ;Clear Screen
000177      .BYTE      "P"          ;Clear to End of Page
000178      .BYTE      "L"          ;Clear to End of Line
000179      .BYTE      "H"          ;Home Cursor
000180      .BYTE      ASC_BS        ;Move left
000181      .BYTE      ASC_NAK      ;Move right
000182      .BYTE      ASC_VT      ;Move up
000183      .BYTE      ASC_LF      ;Move down
000184  ECMDCNT      .EQU      *-ESCCMD
000185      ;
000186  ESCOP      .BYTE      ASC_ETX
000187      .BYTE      ASC_STX
000188      .BYTE      ASC_SOH
000189      .BYTE      ASC_FS
000190      .BYTE      ASC_GS
000191      .BYTE      ASC_US
000192      .BYTE      ASC_FF
000193      .BYTE      ASC_BS
000194      .BYTE      ASC_HT
000195      .BYTE      ASC_VT
000196      .BYTE      ASC_LF
000197

```

```

; #####
; #   END OF FILE:  CONS.DAT3.TEXT
; #   LINES       :  197
; #   CHARACTERS  :  10599
; #   Formatter   :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; #   Author      :  David T. Craig -- 71533.606@compuserve.com -- Santa Fe, New Mexico USA
; #####

```



```

; #####
; # PROJECT : Apple /// SOS Console Driver 1.31 (6502 Assembly Source Code)
; # FILE NAME: CONS.MAIN.TEXT
; #####

```

```

000001          .PAGE
000002 ;-----
000003 ;
000004 ; Console Device Handler
000005 ;
000006 ; This is the device handler's entry point. It sets the extended
000007 ; addressing bytes to zero and moves in the permanent zero page
000008 ; data, then switches to the appropriate request handler. If the
000009 ; request handler modifies the permanent zero page data, it must
000010 ; call ZPOUT before it exits to SOS.
000011 ;
000012 ;-----
000013 ;
000014 CNSLDH      .EQU      *
000015          LDX      #0FF-ZPDATA
000016          LDY      #00
000017          TYA
000018 $010      STA      1400+ZPDATA,X          ;Set extend bytes to zero
000019          CPX      #ZPLENGTH
000020          BCS      $020
000021          LDA      ZPSAVE,X
000022          STA      ZPDATA,X          ;Set up zero page data
000023          TYA
000024 $020      DEX
000025          BPL      $010
000026 ;
000027          SWITCH   REQCODE,8,CREQSW
000028 ;
000029 ;
000030 CBADREQ     LDA      #XREQCODE          ;Invalid request code
000031          JSR      SYSERR
000032 ;
000033 CNOTOPEN   LDA      #XNOTOPEN          ;Console is not open
000034          JSR      SYSERR
000035 ;
000036 CREQSW     .WORD    CNSLREAD-1
000037          .WORD    CNSLWRIT-1
000038          .WORD    CNSLSTAT-1
000039          .WORD    CNSLCNTL-1

```

```

000040          .WORD      CBADREQ-1
000041          .WORD      CBADREQ-1
000042          .WORD      CNSLOPEN-1
000043          .WORD      CNSLCLOS-1
000044          .WORD      CNSLINIT-1
000045          .PAGE
000046 ;-----
000047 ;
000048 ; Keyboard Interrupt Handler
000049 ;
000050 ;-----
000051 ;
000052 KYBDMIH      .EQU      *
000053 ;
000054 ; Read keyboard data and clear interrupt
000055 ;
000056          LDX          #KYBDCLR
000057          LDA          KAPORT          ;Read data port
000058          BMI          $010
000059          STX          E_IFR          ;No data ready -- clear
000060          RTS          ; interrupt and exit
000061 $010          AND          #BITOFF7
000062          STA          KADATA
000063          LDA          KBPORT          ;Read status port
000064          EOR          #3C
000065          STA          KBDATA
000066          STX          E_IFR          ;Clear interrupt
000067          STX          KYBDSTRB       ; and keyboard strobe
000068          BMI          KIHSPCL
000069          LDA          KADATA
000070          CMP          #ASC_CR
000071          BNE          KIHXFORM
000072          LDA          KBDATA
000073          AND          #BITON2          ;Transform CR iff
000074          BNE          KIHXFORM       ; CTRL is held down
000075          JMP          KIH1KY
000076 ;
000077 ; Special key
000078 ; Check for console control commands
000079 ; Do not transform character code
000080 ;
000081 KIHSPCL      AND          #36          ;Isolate A1, A2, CTRL, & SHIFT
000082          CMP          #BITON2
000083          BEQ          $050
000084 $010          JMP          KIH1KY          ;Not a console control command

```

```

000085 $050      LDA      KADATA
000086          CMP      #"5"          ;Toggle video?
000087          BCC      $010
000088          BNE      $060
000089          LDA      SCRNMODE
000090          EOR      #BITON7
000091          STA      SCRNMODE
000092          RTS
000093 $060      CMP      #"6"          ;Flush input buffer?
000094          BNE      $070
000095          LDA      #00
000096          STA      KEYCNT
000097          STA      BUFHEAD
000098          STA      BUFTAIL
000099          RTS
000100 $070      CMP      #"7"          ;Suspend screen output?
000101          BNE      $080
000102          LDA      SUSPFLSH
000103          EOR      #BITON7
000104          STA      SUSPFLSH
000105          RTS
000106 $080      CMP      #"8"          ;Display control characters?
000107          BNE      $090
000108          LDA      DSPLYCTL
000109          EOR      #BITON7
000110          STA      DSPLYCTL
000111          RTS
000112 $090      CMP      #"9"          ;Flush screen output?
000113          BNE      KIH1KY
000114          LDA      SUSPFLSH
000115          AND      #BITOFF7
000116          EOR      #BITON6
000117          STA      SUSPFLSH
000118          RTS
000119          ;
000120          ; Standard key
000121          ; Transform character code
000122          ; Check for alpha lock
000123          ;
000124 KIHIFORM   LDA      KADATA
000125 $010        CMP      #7B          ;Convert ASCII code to
000126          BCC      $030          ; transform table index
000127          CMP      #7E
000128          BCC      $020
000129          EOR      #0C0

```

```

000130          BNE          $040
000131  $020          AND          #5F
000132  $030          ORA          #0C0
000133  $040          TAX
000134          LDA          KBDATA          ;Get control & shift keys
000135          LSR          A
000136          AND          #03
000137          ORA          XFORMTBL,X          ;OR in key number
000138          TAX
000139          LDA          XFORMTBL,X          ;Need to test alpha lock?
000140          BPL          $050
000141          LDA          KBDATA          ;Check alpha lock key
000142          AND          #BITON3
000143          BEQ          $050
000144          TXA
000145          ORA          #BITON0          ;Force shift key on
000146          TAX
000147  $050          LDA          XFORMTBL,X          ;Get key code
000148          AND          #BITOFF7
000149          STA          KADATA
000150  ;
000151  ; Set bit 7 according to Apple 1 key
000152  ;
000153  KIHAlKY          LDA          KBDATA
000154          AND          #BITON4
000155          BEQ          KIHCKEV
000156          LDA          KADATA
000157          ORA          #BITON7
000158          STA          KADATA
000159  ;
000160  ; Check for Any Key and Attention events
000161  ;
000162  KIHCKEV          BIT          READING          ;If reading,
000163          BMI          $010          ; ignore Any Key event
000164          LDA          ANYKYEVNT          ;Check Any Key Event
000165          BEQ          $010
000166          LDX          ANYKYPARM
000167          LDY          ANYKYPARM+1
000168          JSR          QUEEVENT          ;Queue the event
000169          LDA          #FALSE
000170          STA          ANYKYEVNT          ;Disable Any Key event
000171          BEQ          $020
000172  $010          LDA          ATTNEVNT          ;Check Attention Event
000173          BEQ          KIHBFCH
000174          LDA          KADATA

```

```

000175          CMP          ATTNCHAR
000176          BNE          KIHBFCH
000177          LDX          ATTNPARM
000178          LDY          ATTNPARM+1
000179          JSR          QUEEVENT          ;Queue the event
000180          LDA          #FALSE
000181          STA          ATTNEVNT          ;Disable Attention event
000182 $020      STA          READING          ;Terminate any read in progress
000183          STA          KEYCNT          ;Flush the input buffer
000184          STA          BUFHEAD
000185          STA          BUFTAIL
000186          STA          SUSPFLSH          ;Clear suspend & flush flags
000187 ;
000188 ; Buffer the character
000189 ;
000190 KIHBFCH     LDX          BUFSIZ          ;Buffering enabled?
000191          BEQ          $030
000192          DEX
000193          CPX          KEYCNT          ;Any room in buffer?
000194          BCS          $010
000195          BIT          BELL          ;Buffer overflow
000196          BCC          $030
000197 $010      INC          KEYCNT          ;Bump the key count
000198          LDX          BUFTAIL
000199          LDA          KADATA
000200          STA          KABUF,X          ;Buffer the keystroke
000201          LDA          KBDATA
000202          STA          KBBUF,X
000203          INX
000204          CPX          BUFSIZ          ;Bump buffer tail pointer
000205          BCC          $020
000206          LDX          #0
000207 $020      STX          BUFTAIL
000208 $030      RTS
000209          .PAGE
000210 ;-----
000211 ;
000212 ; Subroutine GETKEY
000213 ;
000214 ; This subroutine gets the next keystroke from the type ahead buffer.
000215 ; On entry, the interrupt system must be enabled but the keyboard
000216 ; interrupt must be masked. On exit, if carry is clear, A contains
000217 ; the keyboard A port data and X contains the keyboard B port data;
000218 ; Y is undefined. If carry is set, no data is returned; either the
000219 ; buffer was empty and the NOWAIT flag is true, or the read was

```

```

000220 ; terminated by the interrupt handler.
000221 ;
000222 ;-----
000223 ;
000224 GETKEY      .EQU      *
000225          LDA      KEYCNT          ;Anything in the buffer?
000226          BNE      $030          ; Yes
000227          PHP
000228          SEI
000229          LDA      SCRNMODE
000230          ORA      #BITON7
000231          STA      SCRNMODE          ;Turn on video
000232          LDA      E_REG
000233          ORA      #BITON5
000234          STA      E_REG
000235          PLP
000236          BIT      NOWAIT          ;Check the NOWAIT flag
000237          BPL      $010
000238          ASL      READING          ;Clear the READING flag,
000239          RTS          ; set carry, and exit
000240 ;
000241 $010        LDX      BUFSIZ          ;Preserve buffer size in X
000242          LDA      #1          ;Set buffer size to 1
000243          STA      BUFSIZ
000244          LDA      #KYBDENBL          ;Unmask the keyboard
000245          STA      E_IER
000246          BIT      ESCAPE          ;In ESCAPE mode?
000247          BVC      $020          ; No
000248          CLC
000249          LDA      TPX          ;Preserve current cursor and
000250          LDY      HMODE          ; replace it with plus sign
000251          BPL      $015
000252          LSR      A
000253          BCC      $015
000254          TAY
000255          LDA      (BASE2),Y
000256          PHA
000257          AND      #BITON7
000258          ORA      #2B
000259          STA      (BASE2),Y
000260          BCS      $020
000261 $015        TAY
000262          LDA      (BASE1),Y
000263          PHA
000264          AND      #BITON7

```

```

000265          ORA          #2B
000266          STA          (BASE1),Y
000267  $020     LDA          KEYCNT          ;Wait for a keystroke
000268          BEQ          $020
000269          BVC          $026          ;Not in ESCAPE mode
000270          PLA          ;Restore original cursor
000271          BCC          $024
000272          STA          (BASE2),Y
000273          BCS          $026
000274  $024     STA          (BASE1),Y
000275  $026     LDA          #KYBDDSDL          ;Mask the keyboard
000276          STA          E_IER
000277          STX          BUFSIZ          ;Restore the buffer size
000278          SEC
000279          BIT          READING          ;Check the reading flag
000280          BPL          $060          ; Exit with carry set
000281  ;
000282  $030     LDY          BUFHEAD          ;Get buffer index of keystroke
000283          DEC          KEYCNT
000284          BNE          $040          ;If KEYCNT = 0
000285          LDA          #0
000286          STA          BUFHEAD          ; then BUFHEAD := BUFTAIL := 0
000287          STA          BUFTAIL
000288          BEQ          $050
000289  $040     INC          BUFHEAD          ; else BUFHEAD := BUFHEAD + 1
000290          LDA          BUFHEAD
000291          CMP          BUFSIZ          ;If BUFHEAD >= BUFSIZ
000292          BCC          $050
000293          LDA          #0          ; then BUFHEAD := 0
000294          STA          BUFHEAD
000295  $050     LDA          KABUF,Y          ;Load the A and B port data
000296          LDX          KBBUF,Y
000297          CLC
000298  $060     RTS
000299          .PAGE
000300  ;-----
000301  ;
000302  ; Subroutine SCRNECHO
000303  ;
000304  ; This subroutine writes a single character to the screen. On entry,
000305  ; the character must be in A. On exit, all registers are undefined.
000306  ;
000307  ;-----
000308  SCRNECHO      .EQU          *
000309          BIT          ECHO          ;Screen Echo enabled?

```

```

000310          BPL          $010
000311          PHA
000312          JSR          CURSOR          ;Remove cursor
000313          PLA
000314          JSR          PRINT          ;Print the character
000315          JSR          CURSOR          ;Restore cursor
000316 $010      RTS
000317 ;
000318 ;
000319 ;-----
000320 ;
000321 ; Subroutine BACKSP
000322 ;
000323 ; This subroutine performs the screen backspace when the console
000324 ; deletes an input character. On entry, the input buffer pointer
000325 ; must point to the character to be deleted and the overflow flag
000326 ; must be set to indicate that the character should be erased, or
000327 ; clear to indicate that it should be left on the screen. On exit,
000328 ; all registers are undefined.
000329 ;
000330 ;-----
000331 BACKSP      .EQU          *
000332          LDA          ECHO
000333          BPL          $020          ;Screen Echo not enabled
000334          LDY          #0
000335          LDA          (BUFPTR),Y    ;Printable character?
000336          CMP          #ASC_SP
000337          BCC          $020
000338          PHP          ;Save overflow flag
000339          JSR          CURSOR          ;Remove cursor
000340          LDA          #ASC_BS
000341          JSR          PRINT          ;Backspace
000342          PLP
000343          BVC          $010          ;Don't erase
000344          LDA          #ASC_SP
000345          JSR          PRINT          ;Erase the character
000346          LDA          #ASC_BS
000347          JSR          PRINT
000348 $010      JSR          CURSOR          ;Restore cursor
000349 $020      RTS

; #####
; # END OF FILE: CONS.MAIN.TEXT
; # LINES : 349
; # CHARACTERS : 17347

```



```
; #   Formatter   :   Assembly Language Reformatter 1.0.2 (07 January 1998)
; #   Author      :   David T. Craig -- 71533.606@compuserve.com -- Santa Fe, New Mexico USA
; #####
```

```

; #####
; #   PROJECT   :   Apple /// SOS Console Driver 1.31 (6502 Assembly Source Code)
; #   FILE NAME:   CONS.READ.TEXT
; #####

000001          .PAGE
000002 ;-----
000003 ;
000004 ;   Console Read Request
000005 ;
000006 ;   Parameters:
000007 ;     BUFPTR:  Pointer to caller's data buffer
000008 ;     REQCNT:  Requested read count
000009 ;     RTNCNT:  Pointer to actual read count
000010 ;
000011 ;   Zero Page Temporary Storage
000012 ;     COUNT:  Number of bytes read
000013 ;     ONEBYTE: TRUE if REQCNT = 1
000014 ;
000015 ;   If the ECHO or ESCAPE functions are enabled, this segment will call
000016 ;   PRINT to display a character or perform a screen control function.
000017 ;
000018 ;-----
000019 ;
000020 CNSLREAD      .EQU      *
000021 ;
000022 ;   Initialize read variables
000023 ;
000024             BIT      OPENFLG
000025             BMI      $010
000026             JMP      CNOTOPEN
000027 $010        BIT      KYBDMODE           ;Keyboard mode?
000028             BMI      $030
000029             LDA      SMCursor          ;Save cursor status
000030             PHA
000031             BMI      $020
000032             LDA      #ASC_ENQ          ;Turn on cursor
000033             JSR      SCRNECHO
000034 $020        LDA      #FALSE
000035             STA      ONEBYTE           ;Clear one byte read flag
000036             LDA      REQCNT+1
000037             BNE      $040
000038             LDA      REQCNT
000039             CMP      #1

```

```

000040          BNE          $040
000041          ROR          ONEBYTE          ;Set one byte read flag
000042          BMI          $040
000043          ;
000044          $030          LDA          REQCNT          ;Make requested count even
000045          AND          #BITOFF0
000046          STA          REQCNT
000047          ;
000048          $040          LDA          ESCAPE
000049          AND          #BITON7
000050          STA          ESCAPE          ;Clear escape pending
000051          LDA          #0
000052          STA          COUNT
000053          STA          COUNT+1          ;Zero bytes read count
000054          PHP
000055          SEI
000056          STA          SUSPFLSH          ;Clear suspend & flush flags
000057          LDA          #KYBDDSBL
000058          STA          E_IER          ;Mask the keyboard
000059          LDA          #TRUE
000060          STA          READING          ;Set the READING flag
000061          PLP
000062          ;
000063          ; Main read loop
000064          ;
000065          CNSLLOOP      LDA          COUNT          ;If COUNT >= REQCNT
000066          CMP          REQCNT          ; then goto CNSLEXIT
000067          LDA          COUNT+1
000068          SBC          REQCNT+1
000069          BCC          $020
000070          $010          JMP          CNSLEXIT
000071          ;
000072          $020          JSR          GETKEY          ;Get next keystroke
000073          BCS          $010
000074          BIT          KYBDMODE          ;Console or Keyboard mode?
000075          BPL          TSTESCAPE
000076          ;
000077          ; Keyboard mode read
000078          ;
000079          KYBDRDY      PHA          ;Save ASCII byte
000080          LDY          #0
000081          STA          (BUFFPTR),Y          ;Store data byte in buffer
000082          INY
000083          TXA
000084          STA          (BUFFPTR),Y          ;Store status byte in buffer

```

```

000085          LDA          #02
000086          JMP          BUMPCNT          ;Go update COUNT and BUFFPTR
000087      ;
000088      ; Console mode read
000089      ;
000090      TSTESCAPE      BIT          ECHO          ;Test for Escape Mode
000091          BPL          TSTCHDEL
000092          BIT          ESCAPE
000093          BPL          TSTCHDEL
000094          BVC          $040          ;Escape not pending
000095          LDY          #ECMDCNT-1
000096          CMP          #"a"
000097          BCC          $010
000098          CMP          #"{"
000099          BCS          $010
000100          AND          #BITOFF5          ;Upshift lower case alpha
000101      $010      CMP          ESCCMD,Y          ;Search for escape command
000102          BEQ          $020
000103          DEY
000104          BPL          $010
000105          ASL          ESCAPE          ;Not found -- clear pending flag
000106          BCS          $030
000107      $020      LDA          ESCOP,Y          ;Get screen control character
000108          JSR          SCRNECHO
000109      $030      JMP          CNSLLOOP
000110      ;
000111      $040      CMP          #ASC_ESC          ;Is this an ESC?
000112          BNE          TSTCHDEL
000113          ROR          ESCAPE          ;Set escape pending
000114          BMI          $030
000115      ;
000116      TSTCHDEL      BIT          ONEBYTE          ;Test for character delete
000117          BMI          TSTLNDEL
000118          BIT          CHDELFLG
000119          BPL          TSTLNDEL
000120          CMP          #CHDELCHR
000121          BNE          TSTLNDEL
000122          LDA          COUNT          ;Anything to delete?
000123          ORA          COUNT+1
000124          BEQ          $030
000125          LDA          COUNT
000126          BNE          $010
000127          DEC          COUNT+1          ;Decrement current read count
000128      $010      DEC          COUNT
000129          LDA          BUFFPTR

```

```

000130          BNE          $020
000131          DEC          BUFFPTR+1          ;Decrement buffer pointer
000132  $020      DEC          BUFFPTR
000133          JSR          BACKSP          ;Backspace
000134  $030      JMP          CNSLLOOP
000135  ;
000136  TSTLNDEL  BIT          ONEBYTE          ;Test for line delete
000137          BMI          TSTCHCPY
000138          BIT          LNDELFLG
000139          BPL          TSTCHCPY
000140          CMP          #LNDELCHR
000141          BNE          TSTCHCPY
000142          LDA          ECHO
000143          BPL          $050
000144          BVC          $040
000145  ;
000146  $010      LDA          COUNT          ;Anything to delete?
000147          ORA          COUNT+1
000148          BEQ          $060
000149          LDA          COUNT
000150          BNE          $020
000151          DEC          COUNT+1          ;Decrement current read count
000152  $020      DEC          COUNT
000153          LDA          BUFFPTR
000154          BNE          $030
000155          DEC          BUFFPTR+1        ;Decrement buffer pointer
000156  $030      DEC          BUFFPTR
000157          BIT          LNDELFLG
000158          JSR          BACKSP          ;Backspace
000159          JMP          $010
000160  ;
000161  $040      LDA          #"\ "
000162          JSR          SCRNECHO        ;Write "\ CR LF"
000163          LDA          #ASC_CR
000164          JSR          SCRNECHO
000165          LDA          #ASC_LF
000166          JSR          SCRNECHO
000167  $050      SEC
000168          LDA          BUFFPTR          ;Reset buffer pointer
000169          SBC          COUNT
000170          STA          BUFFPTR
000171          LDA          BUFFPTR+1
000172          SBC          COUNT+1
000173          STA          BUFFPTR+1
000174          LDA          #0          ;Reset current read count

```

```

000175          STA          COUNT
000176          STA          COUNT+1
000177  $060          JMP          CNSLLOOP
000178          ;
000179  TSTCHCPY      BIT          ECHO          ;Test for character copy
000180          BPL          CNSLRDY
000181          BIT          CHCPYFLG
000182          BPL          CNSLRDY
000183          CMP          #CHCPYCHR
000184          BNE          CNSLRDY
000185          JSR          SCRNPICK          ;Copy character from screen
000186          ASL          A
000187          CMP          #40
000188          ROR          A
000189          EOR          #BITON7
000190          ;
000191  CNSLRDY      PHA          ;Save character for new line test
000192          LDY          #0
000193          STA          (BUFFPTR),Y          ;Store character in buffer
000194          ;
000195          BIT          ECHO          ;Echo enabled?
000196          BPL          $020
000197          BVS          $010
000198          CMP          #20          ;Check for control character
000199          BCC          $020
000200  $010          JSR          SCRNECHO
000201  $020          LDA          #01
000202          ;
000203  BUMPCNT      PHA
000204          CLC
000205          ADC          COUNT          ;Update current read count
000206          STA          COUNT
000207          BCC          $010
000208          INC          COUNT+1
000209  $010          PLA
000210          CLC
000211          ADC          BUFFPTR          ;Update buffer pointer
000212          STA          BUFFPTR
000213          BCC          TSTNEWLN
000214          INC          BUFFPTR+1
000215          LDA          BUFFPTR+1
000216          CMP          #0FF
000217          BCC          TSTNEWLN
000218          SBC          #080          ;Wrap buffer at FF page
000219          STA          BUFFPTR+1

```

```

000220             INC             1400+BUFFPTR+1
000221 ;
000222 TSTNEWLN      PLA             ;Test for New Line
000223             BIT             NEWLINE
000224             BPL             $010
000225             CMP             NEWLNCHR
000226             BEQ             CNSLEXIT
000227 $010        JMP             CNSLLOOP
000228 ;
000229 CNSLEXIT     ASL             READING          ;Clear the READING flag
000230             LDA             #KYBDENBL
000231             STA             E_IER             ;Unmask the keyboard
000232             LDY             #0
000233             LDA             COUNT           ;Return the actual byte count
000234             STA             (RTNCNT),Y
000235             INY
000236             LDA             COUNT+1
000237             STA             (RTNCNT),Y
000238             BIT             KYBDMODE
000239             BMI             $020
000240             PLA
000241             BMI             $010
000242             LDA             #ASC_ACK         ;Turn off cursor
000243             JSR             SCRNECHO
000244 $010        JSR             ZPOUT
000245 $020        RTS

```

```

; #####
; #   END OF FILE:  CONS.READ.TEXT
; #   LINES       :   245
; #   CHARACTERS  :  11718
; #   Formatter   :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; #   Author      :  David T. Craig -- 71533.606@compuserve.com -- Santa Fe, New Mexico USA
; #####

```

```

; #####
; #   PROJECT : Apple /// SOS Console Driver 1.31 (6502 Assembly Source Code)
; #   FILE NAME: CONS.WRIT.TEXT
; #####

```

```

000001             .PAGE
000002 ;-----
000003 ;
000004 ; Console Write Request
000005 ;
000006 ; Parameters:
000007 ;     BUFPTR: Pointer to caller's data buffer
000008 ;     REQCNT: Number of bytes to write
000009 ;
000010 ; Zero Page Temporary Storage
000011 ;     COUNT: Number of bytes written
000012 ;
000013 ; Additional zero page data may be used to perform screen
000014 ; control functions.
000015 ;
000016 ;-----
000017 ;
000018 CNSLWRIT      .EQU      *
000019             BIT        OPENFLG
000020             BMI        $010
000021             JMP        CNOTOPEN
000022 $010         JSR        CURSOR           ;Remove Cursor
000023             LDA        #0
000024             STA        COUNT           ;Zero COUNT
000025             STA        COUNT+1
000026 ;
000027 $020         LDA        COUNT           ;Check for end of buffer
000028             CMP        REQCNT
000029             LDA        COUNT+1
000030             SBC        REQCNT+1
000031             BCS        $060           ;Go Exit
000032 $030         BIT        SUSPFLSH       ;Check suspend and flush flags
000033             BMI        $030           ; Suspend
000034             BVS        $050           ; Flush
000035             LDY        #0
000036             LDA        (BUFPTR),Y     ;Get next byte
000037             JSR        PRINT          ;Print the byte
000038             INC        BUFPTR
000039             BNE        $040           ;Bump pointer

```



```

000040          INC          BUFFPTR+1
000041          BNE          $040
000042          LDA          #80
000043          STA          BUFFPTR+1          ;Process buffer wrap around
000044          INC          1400+BUFFPTR+1
000045  $040      INC          COUNT
000046          BNE          $020          ;Bump bytes read count
000047          INC          COUNT+1
000048          JMP          $020
000049          ;
000050  $050      LDA          #00
000051          STA          CTLINDX          ;Clear any pending cntl function
000052          ;
000053  $060      JSR          CURSOR          ;Restore cursor
000054          JMP          ZPOUT          ;Save Zero Page data and exit
000055          .PAGE
000056          ;-----
000057          ;
000058          ; Subroutine PRINT
000059          ;
000060          ; This routine processes a single byte of output. Characters are
000061          ; printed by calling DISPLAY. Screen control functions are processed
000062          ; by accumulating any required parameters in CTLBUFF then switching
000063          ; to the appropriate screen control routine.
000064          ;
000065          ; Parameters:
000066          ;   A: The byte to process
000067          ;
000068          ; Exit:
000069          ;   A, X, Y: Undefined
000070          ;
000071          ;-----
000072          ;
000073  PRINT      .EQU          *
000074          LDY          CTLINDX          ;Get control function index
000075          BNE          $010
000076          ORA          DSPLYCTL
000077          CMP          #ASC_SP          ;Display or control?
000078          BCS          DISPLAY
000079          TAX
000080          LDA          QUOTATBL,X          ;Get function quota
000081          BEQ          $020
000082          STA          CTLQUOTA
000083          TXA
000084  $010      STA          CTLBUFF,Y          ;Save function character

```

```

000085             INY
000086             STY             CTLINDX             ;Update buffer index
000087             CPY             CTLQUOTA           ;See if quota filled
000088             BCC             $020
000089             LDY             #0
000090             STY             CTLINDX             ;Zero buffer index
000091             SWITCH          CTLBUFF,,CTLSWTBL,*
000092 $020          RTS
000093             .PAGE
000094 ;-----
000095 ;
000096 ; Subroutine DISPLAY
000097 ;
000098 ; This routine displays a single character.  If auto advance is
000099 ; enabled, it calls CF_HT to advance the cursor.
000100 ;
000101 ; Parameters:
000102 ;   A: The character to be displayed
000103 ;
000104 ; Exit:
000105 ;   A, X, Y: Undefined
000106 ;
000107 ;-----
000108 ;
000109 DISPLAY         .EQU           *
000110             ORA             #80             ;set hi-bit
000111             EOR             SMINV          ;set normal or inverse
000112             PHA             ;(for safe keeping)
000113             BIT             HMODE          ;80 column text?
000114             BPL             $010
000115             LDA             TPX
000116             LSR             A             ;80 col: X=TPX/2
000117             TAY             ;carry bit clear?
000118             BCC             $020          ;yes: use page 1
000119             PLA
000120             STA             (BASE2),Y      ;80 col page two
000121             BCS             $030
000122 $010          LDY             TPX         ;40 col: X=TPX
000123             LDA             TCOLOR
000124             STA             (BASE2),Y      ;set color byte
000125 $020          PLA
000126             STA             (BASE1),Y      ;80 col page one
000127 $030          BIT             SMAUTOADV     ;if auto advance,
000128             BPL             $040
000129             JMP             CF_HT         ;advance cursor

```

```

000130 $040          RTS
000131          .PAGE
000132 ;-----
000133 ;
000134 ;   Control Function Quota and Switch Tables
000135 ;
000136 ;-----
000137 QUOTATBL      .EQU          *                ;The Control Function Quota Table
000138          .BLOCK          1,0                ; contains the total number of
000139          .BLOCK          15.,1              ; bytes required by the function,
000140          .BLOCK          1,2                ; including the function character
000141          .BLOCK          2,1                ; itself. A zero indicates that
000142          .BLOCK          2,2                ; the function is unimplemented.
000143          .BLOCK          1,2
000144          .BLOCK          1,1
000145          .BLOCK          3,2
000146          .BLOCK          1,3
000147          .BLOCK          1,0
000148          .BLOCK          4,1
000149 CTLSWTBL      .EQU          *
000150          .WORD          CF_NUL-1            ;00 no-op
000151          .WORD          CF_SOH-1            ;01 Save Environment & Release Viewport
000152          .WORD          CF_STX-1            ;02 Set Viewport Upper Left
000153          .WORD          CF_ETX-1            ;03 Set Viewport Lower Right
000154          .WORD          CF_EOT-1            ;04 Restore Environment
000155          .WORD          CF_ENQ-1            ;05 Cursor On
000156          .WORD          CF_ACK-1            ;06 Cursor Off
000157          .WORD          CF_BEL-1            ;07 Audible signal
000158          .WORD          CF_BS-1             ;08 Backspace
000159          .WORD          CF_HT-1             ;09 Forward Space
000160          .WORD          CF_LF-1             ;0A Line Feed
000161          .WORD          CF_VT-1             ;0B Reverse Line Feed
000162          .WORD          CF_FF-1             ;0C Home Cursor
000163          .WORD          CF_CR-1             ;0D Carriage Return
000164          .WORD          CF_SO-1             ;0E Screen Off
000165          .WORD          CF_SI-1             ;0F Screen On
000166          .WORD          CF_DLE-1            ;10 Set Text Mode
000167          .WORD          CF_DC1-1            ;11 Normal Video
000168          .WORD          CF_DC2-1            ;12 Inverse Video
000169          .WORD          CF_DC3-1            ;13 Foreground Color
000170          .WORD          CF_DC4-1            ;14 Background Color
000171          .WORD          CF_NAK-1            ;15 Set Text Options
000172          .WORD          CF_SYN-1            ;16 Sync on VBL
000173          .WORD          CF_ETB-1            ;17 Horizontal Shift
000174          .WORD          CF_CAN-1            ;18 Go to X

```

```
000175          .WORD      CF_EM-1          ;19  Go to Y
000176          .WORD      CF_SUB-1         ;1A  Go to X,Y
000177          .WORD      CF_ESC-1         ;1B  No-op
000178          .WORD      CF_FS-1          ;1C  Clear Screen
000179          .WORD      CF_GS-1         ;1D  Clear to End of Screen
000180          .WORD      CF_RS-1          ;1E  Clear Line
000181          .WORD      CF_US-1          ;1F  Clear to End of Line
000182
```

```
; #####
; #   END OF FILE:  CONS.WRIT.TEXT
; #   LINES       :   182
; #   CHARACTERS  :  10016
; #   Formatter   :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; #   Author      :  David T. Craig -- 71533.606@compuserve.com -- Santa Fe, New Mexico USA
; #####
```

```

; #####
; #   PROJECT : Apple /// SOS Console Driver 1.31 (6502 Assembly Source Code)
; #   FILE NAME: CONS.FCTN.TEXT
; #####

000001          .PAGE
000002 ;-----
000003 ;
000004 ; Screen Control Functions
000005 ;
000006 ; These routines perform all screen control functions.
000007 ;
000008 ;   Parameters:
000009 ;     All parameters are accumulated in CTLBUFF
000010 ;
000011 ;   Exit:
000012 ;     A, X, Y: Undefined
000013 ;
000014 ;-----
000015 ;
000016 CF_NUL      .EQU      *
000017 CF_ESC      .EQU      *
000018 RTS                          ;NO-OP
000019 ;
000020 CF_SOH      .EQU      *                          ;Save & Release Viewport
000021 LDY          #SCRSTLEN
000022 $010        LDA      SCRSTTBL-1,Y
000023             STA      SCRSTSAV-1,Y
000024             DEY
000025             BNE      $010
000026             CLC
000027             LDA      TPX
000028             ADC      VPL
000029             STA      TPX                          ;retain X posn
000030             LDA      TPY
000031             ADC      VPT
000032             STA      TPY                          ;retain y posn
000033             LDA      #0
000034             STA      VPL                          ;zero left margin
000035             STA      VPT                          ;zero top margin
000036             LDA      #0FF
000037             STA      VPR                          ;Let VERIFY set the right edge
000038             STA      VPB                          ; and bottom margin
000039             JMP      VERIFY

```

```

000040 ;
000041 CF_STX      .EQU      *              ;SET VIEWPORT UPPER LEFT
000042          CLC
000043          LDA          VPL
000044          ADC          TPX              ;at cursor posn
000045          STA          VPL              ;set left margin
000046          LDA          VPT
000047          ADC          TPY
000048          STA          VPT              ;set top margin
000049          LDA          #0
000050          STA          TPX              ;reset cursor X
000051          STA          TPY              ; and cursor Y
000052          JMP          VERIFY          ;and verify
000053 ;
000054 CF_ETX      .EQU      *              ;SET VIEWPORT LOWER RIGHT
000055          CLC
000056          LDA          TPX
000057          ADC          VPL
000058          STA          VPR              ;set left margin
000059          LDA          TPY
000060          ADC          VPT              ;& bottom margin
000061          STA          VPB
000062          JMP          VERIFY          ;and verify
000063 ;
000064 CF_EOT      .EQU      *              ;RESTORE VIEWPORT
000065          LDY          #SCRSTLEN
000066          $010      LDA          SCRSTSAV-1,Y
000067          STA          SCRSTTBL-1,Y
000068          DEY
000069          BNE          $010
000070          JMP          VERIFY
000071 ;
000072 CF_ENQ      .EQU      *              ;ENABLE CURSOR
000073          LDA          SMODE
000074          ORA          #BITON4
000075          STA          SMODE
000076          LDA          #TRUE
000077          STA          SMCURSOR
000078          RTS
000079 ;
000080 CF_ACK      .EQU      *              ;DISABLE CURSOR
000081          LDA          SMODE
000082          AND          #BITOFF4
000083          STA          SMODE
000084          LDA          #FALSE

```

```

000085          STA          SMCURSOR
000086          RTS
000087          ;
000088  CF_BEL          .EQU          *          ;Sound Bell
000089          BIT          BELL
000090          RTS
000091          ;
000092  CF_BS          .EQU          *          ;BACKSPACE
000093          DEC          TPX
000094          BPL          $020
000095          BIT          SMAUTOCR          ;BS at left:
000096          BPL          $010
000097          LDA          VPHMAX          ;Wrap to right
000098          STA          TPX          ; edge of viewport
000099          JMP          CF_VT
000100  $010          INC          TPX
000101  $020          RTS
000102          ;
000103  CF_HT          .EQU          *          ;ADVANCE
000104          LDA          TPX
000105          CMP          VPHMAX
000106          BCS          $010          ;at edge?
000107          INC          TPX          ;no: advance
000108          RTS
000109  $010          BIT          SMAUTOCR          ;auto CR on?
000110          BPL          CF_EXIT
000111          LDA          #0          ;yes: wrap to
000112          STA          TPX          ;left margin
000113          JMP          CF_LF          ;& line feed
000114          ;
000115  CF_LF          .EQU          *          ;LINE FEED
000116          LDA          TPY
000117          CMP          VPVMAX
000118          BCS          $010          ;at edge?
000119          INC          TPY          ;no: move down
000120          JMP          TBASCAL          ;calc base address
000121  $010          BIT          SMSCROLL          ;auto scroll?
000122          BPL          CF_EXIT
000123          LDA          #00
000124          JMP          SCROLL          ;yes: go to it
000125          ;
000126  CF_VT          .EQU          *          ;REVERSE LINE FEED
000127          LDA          TPY
000128          BEQ          $010          ;at top?
000129          DEC          TPY          ;no: do it

```

```

000130          JMP          TBASCAL          ;calc base address
000131  $010          BIT          SMSCROLL          ;auto scroll?
000132          BPL          CF_EXIT
000133          LDA          #80
000134          JMP          SCROLL
000135  ;
000136  CF_FF          .EQU          *          ;FORM FEED
000137          LDA          #0
000138          STA          TPX          ;reset TPX
000139          STA          TPY          ; and TPY
000140          JMP          TBASCAL          ;calc base address
000141  ;
000142  CF_CR          .EQU          *          ;CARRIAGE RETURN
000143          LDA          #0
000144          STA          TPX          ;reset TPX
000145          BIT          SMAUTOLF          ;auto LF set?
000146          BPL          CF_EXIT
000147          JMP          CF_LF          ;yes: go to it
000148  ;
000149  CF_SO          .EQU          *          ;SCREEN OFF
000150          LDA          #FALSE
000151          STA          SCRNMODE
000152          JMP          VERIFY
000153  ;
000154  CF_SI          .EQU          *          ;SCREEN ON
000155          LDA          #TRUE
000156          STA          SCRNMODE
000157          JMP          VERIFY
000158  ;
000159  CF_DLE          .EQU          *          ;SET HARDWARE MODE
000160          LDA          CTLBUFF+1
000161          STA          HMODE
000162          JMP          VERIFY
000163  ;
000164  CF_DC1          .EQU          *          ;NORMAL VIDEO
000165          LDA          SMODE
000166          AND          #BITOFF5          ;reset INVERSE bit
000167          STA          SMODE
000168          LDA          #FALSE
000169          STA          SMINV
000170  CF_EXIT          RTS
000171  ;
000172  CF_DC2          .EQU          *          ;INVERSE VIDEO
000173          LDA          SMODE
000174          ORA          #BITON5          ;set INVERSE bit

```



```

000175          STA          SMODE
000176          LDA          #TRUE
000177          STA          SMINV
000178          RTS
000179          ;
000180  CF_DC3          .EQU          *                ;FOREGROUND COLOR
000181          LDA          CTLBUFF+1
000182          STA          TCF
000183          JMP          VERIFY                ;set TCOLOR
000184          ;
000185  CF_DC4          .EQU          *                ;BACKGROUND COLOR
000186          LDA          CTLBUFF+1
000187          STA          TCB
000188          JMP          VERIFY                ;set TCOLOR
000189          ;
000190  CF_NAK         .EQU          *                ;SET SOFTWARE MODE
000191          LDA          CTLBUFF+1
000192          AND          #0F
000193          STA          CTLBUFF+1
000194          LDA          SMODE                ;Save bits 7-4
000195          AND          #0F0
000196          ORA          CTLBUFF+1
000197          STA          SMODE
000198          JMP          VERIFY
000199          ;
000200  CF_SYN         .EQU          *                ;SYNCHRONIZE WITH VBL
000201          LDA          #SYNCSIZ
000202          LDX          SYNCSADR
000203          LDY          SYNCSADR+1
000204          JSR          ALLOCSIR                ;Allocate CB2 for VBL
000205          BCS          CF_EXIT
000206          JSR          CURSOR                ;Restore cursor while waiting
000207          PHP
000208          SEI
000209          LDA          E_PCR
000210          AND          #1F
000211          ORA          #60                ;Set up CB2 to monitor
000212          STA          E_PCR                ; VBL positive edge
000213          LDA          #08
000214          STA          E_IER
000215          STA          E_IFR
000216          PLP
000217  $010          BIT          E_IFR                ;Wait for VBL edge
000218          BEQ          $010
000219          JSR          CURSOR                ;Remove cursor

```

```

000220          LDA          #SYNCSSIZ
000221          LDX          SYNCSEADR
000222          LDY          SYNCSEADR+1
000223          JMP          DEALCSIR          ;Release CB2 resource
000224          ;
000225          CF_ETB          .EQU          *          ;HORIZONTAL SCROLL
000226          LDA          CTLBUFF+1
000227          $010          JMP          SHIFT
000228          ;
000229          CF_CAN          .EQU          *          ;Go To X
000230          LDA          CTLBUFF+1
000231          CMP          VPHMAX          ;out of range?
000232          BCC          $010
000233          LDA          VPHMAX          ;Set to right margin
000234          $010          STA          TPX
000235          RTS
000236          ;
000237          CF_EM          .EQU          *          ;Go To Y
000238          LDA          CTLBUFF+1
000239          CMP          VPVMAX          ;out of range?
000240          BCC          $010
000241          LDA          VPVMAX          ;Set to top
000242          $010          STA          TPY
000243          JMP          TBASCAL          ;get base address
000244          ;
000245          CF_SUB          .EQU          *          ;Go To X, Y
000246          JSR          CF_CAN
000247          LDA          CTLBUFF+2
000248          STA          CTLBUFF+1
000249          JMP          CF_EM
000250          ;
000251          CF_FS          .EQU          *          ;CLEAR SCREEN
000252          JSR          CF_FF
000253          JMP          CLREOS
000254          ;
000255          CF_GS          .EQU          *          ;CLEAR TO EOS
000256          JMP          CLREOS
000257          ;
000258          CF_RS          .EQU          *          ;CLEAR LINE
000259          LDA          #0
000260          STA          TPX
000261          JMP          CLREOL
000262          ;
000263          CF_US          .EQU          *          ;CLEAR TO EOL
000264          JMP          CLREOL

```

000265

```
; #####  
; #   END OF FILE:  CONS.FCTN.TEXT  
; #   LINES       :   265  
; #   CHARACTERS  :  12878  
; #   Formatter   :  Assembly Language Reformatter 1.0.2 (07 January 1998)  
; #   Author      :  David T. Craig -- 71533.606@compuserve.com -- Santa Fe, New Mexico USA  
; #####
```

```

; #####
; # PROJECT : Apple /// SOS Console Driver 1.31 (6502 Assembly Source Code)
; # FILE NAME: CONS.STAT.TEXT
; #####

```

```

000001          .PAGE
000002 ;-----
000003 ;
000004 ; Console Status Request
000005 ;
000006 ; Parameters:
000007 ;     SCCODE: Status / Control code
000008 ;     SCLIST: Pointer to caller's status / control list
000009 ;
000010 ;     Before switching to the appropriate request handling code,
000011 ;     Y is set to zero.
000012 ;
000013 ;-----
000014 ;
000015 CNSLSTAT      .EQU          *
000016             BIT          OPENFLG          ;Is the Console open?
000017             BMI          $010
000018             JMP          CNOTOPEN
000019 $010         SWITCH      SCCODE,18.,CSTATSW,*
000020             BCS          CBADCTL
000021             LDY          #0
000022             RTS
000023 ;
000024 CBADCTL       LDA          #XCTLCODE          ;Invalid control code
000025             JSR          SYSERR
000026 ;
000027 CSTATSW       .WORD        CSTAT00-1
000028             .WORD        CSTAT01-1
000029             .WORD        CSTAT02-1
000030             .WORD        CSTAT03-1
000031             .WORD        CSTAT04-1
000032             .WORD        CSTAT05-1
000033             .WORD        CSTAT06-1
000034             .WORD        CBADCTL-1
000035             .WORD        CSTAT08-1
000036             .WORD        CSTAT09-1
000037             .WORD        CSTAT10-1
000038             .WORD        CSTAT11-1
000039             .WORD        CSTAT12-1

```

```

000040          .WORD          CSTAT13-1
000041          .WORD          CSTAT14-1
000042          .WORD          CSTAT15-1
000043          .WORD          CSTAT16-1
000044          .WORD          CSTAT17-1
000045          .WORD          CSTAT18-1
000046 ;
000047 CSTAT00      RTS                      ;0 -- NOP
000048 ;
000049 CSTAT01      .EQU          *                      ;1 -- Console Status Table
000050          LDA          (SCLIST),Y
000051          CMP          #CONSTLEN
000052          BCS          $010
000053          LDA          #XCTLPARM
000054          JSR          SYSERR
000055          LDA          $010          #CONSTLEN
000056          STA          (SCLIST),Y
000057          TAY
000058          LDA          $020          CONSTTBL-1,Y
000059          STA          (SCLIST),Y
000060          DEY
000061          BNE          $020
000062          RTS
000063 ;
000064 CSTAT02      .EQU          *                      ;2 -- New Line
000065          LDA          NEWLINE
000066          STA          (SCLIST),Y
000067          INY
000068          LDA          NEWLNCHR
000069          STA          (SCLIST),Y
000070          RTS
000071 ;
000072 CSTAT03      .EQU          *                      ;3 -- Console / Keyboard mode
000073          LDA          KYBDMODE
000074          STA          (SCLIST),Y
000075          RTS
000076 ;
000077 CSTAT04      .EQU          *                      ;4 -- Buffer Size
000078          LDA          BUFSIZ
000079          STA          (SCLIST),Y
000080          RTS
000081 ;
000082 CSTAT05      .EQU          *                      ;5 -- Current Key Count
000083          LDA          KEYCNT
000084          STA          (SCLIST),Y

```

```

000085          RTS
000086      ;
000087  CSTAT06      LDY          #5                      ;6 -- Attention Event
000088      $010      LDA          ATTNEVNT,Y
000089              STA          (SCLIST),Y
000090              DEY
000091              BPL          $010
000092          RTS
000093      ;
000094  CSTAT08      LDY          #4                      ;8 -- Any Key Event
000095      $010      LDA          ANYKYEVRT,Y
000096              STA          (SCLIST),Y
000097              DEY
000098              BPL          $010
000099          RTS
000100      ;
000101  CSTAT09      .EQU          *                      ;09 -- Read Screen with norm/inv
000102              JSR          SCRNPICK
000103              EOR          #BITON7
000104              EOR          SMCURSOR
000105              LDY          #0
000106              STA          (SCLIST),Y
000107          RTS
000108      ;
000109  CSTAT10      .EQU          *                      ;10 -- No Wait Input
000110              LDA          NOWAIT
000111              STA          (SCLIST),Y
000112          RTS
000113      ;
000114  CSTAT11      .EQU          *                      ;11 -- Screen Echo
000115              LDA          ECHO
000116              STA          (SCLIST),Y
000117          RTS
000118      ;
000119  CSTAT12      .EQU          *                      ;12 -- Character Copy
000120              LDA          CHCPYFLG
000121              STA          (SCLIST),Y
000122          RTS
000123      ;
000124  CSTAT13      .EQU          *                      ;13 -- Character Delete
000125              LDA          CHDELFLG
000126              STA          (SCLIST),Y
000127          RTS
000128      ;
000129  CSTAT14      .EQU          *                      ;14 -- Line Delete

```

```

000130          LDA          LNDELFLG
000131          STA          (SCLIST),Y
000132          RTS
000133          ;
000134  CSTAT15          .EQU          *                      ;15 -- Escape Functions
000135          LDA          ESCAPE
000136          STA          (SCLIST),Y
000137          RTS
000138          ;
000139  CSTAT16          .EQU          *                      ;16 -- Cursor Position
000140          LDA          TPX
000141          STA          (SCLIST),Y
000142          INY
000143          LDA          TPY
000144          STA          (SCLIST),Y
000145          RTS
000146          ;
000147  CSTAT17          .EQU          *                      ;17 -- Pick Character
000148          JSR          SCRNPICK
000149          ASL          A
000150          CMP          #40
000151          ROR          A
000152          EOR          #BITON7
000153          LDY          #0
000154          STA          (SCLIST),Y
000155          RTS
000156          ;
000157  CSTAT18          .EQU          *                      ;18 -- Screen Dump
000158          LDA          HMODE
000159          STA          (SCLIST),Y
000160          INY
000161          LDA          VPHMAX
000162          STA          (SCLIST),Y
000163          INY
000164          LDA          VPVMAX
000165          STA          (SCLIST),Y
000166          LDA          #00
000167          BIT          DSPLYCTL
000168          BMI          $010
000169          JMP          SCRNDUMP
000170  $010          STA          (SCLIST),Y          ;If control characters are being
000171          DEY          ; displayed, dump a null viewport
000172          STA          (SCLIST),Y
000173          RTS
000174

```

```
; #####  
; #   END OF FILE:  CONS.STAT.TEXT  
; #   LINES       :   174  
; #   CHARACTERS  :  7340  
; #   Formatter   :  Assembly Language Reformatter 1.0.2 (07 January 1998)  
; #   Author      :  David T. Craig -- 71533.606@compuserve.com -- Santa Fe, New Mexico USA  
; #####
```



```

; #####
; # PROJECT : Apple /// SOS Console Driver 1.31 (6502 Assembly Source Code)
; # FILE NAME: CONS.CNTL.TEXT
; #####

```

```

000001          .PAGE
000002 ;-----
000003 ;
000004 ; Console Control Request
000005 ;
000006 ; Parameters:
000007 ;     SCCODE: Status / Control code
000008 ;     SCLIST: Pointer to caller's status / control list
000009 ;
000010 ;     Before switching to the appropriate request handler, Y is
000011 ;     set to zero and A is loaded with the first byte of the list.
000012 ;
000013 ;-----
000014 ;
000015 CNSLCNTL      .EQU          *
000016             BIT          OPENFLG          ;Console open?
000017             BPL          $010
000018             SWITCH       SCCODE,18.,CCNTLSW,*
000019             BCS          $020
000020             LDY          #00
000021             LDA          (SCLIST),Y
000022             RTS
000023 ;
000024 $010          JMP          CNOTOPEN
000025 ;
000026 $020          JMP          CBADCTL
000027 ;
000028 CCNTLSW      .WORD         CCNTL00-1
000029             .WORD         CCNTL01-1
000030             .WORD         CCNTL02-1
000031             .WORD         CCNTL03-1
000032             .WORD         CCNTL04-1
000033             .WORD         CCNTL05-1
000034             .WORD         CCNTL06-1
000035             .WORD         CBADCTL-1
000036             .WORD         CCNTL08-1
000037             .WORD         CBADCTL-1
000038             .WORD         CCNTL10-1
000039             .WORD         CCNTL11-1

```

```

000040          .WORD      CCNTL12-1
000041          .WORD      CCNTL13-1
000042          .WORD      CCNTL14-1
000043          .WORD      CCNTL15-1
000044          .WORD      LOADSET-1
000045          .WORD      LOAD8-1
000046          .WORD      CCNTL18-1
000047 ;
000048 CCNTL00      LDA          E_IER              ;0 -- Reset
000049          PHA              ;Save current interrupt state
000050          LDA          #KYBDDSBL          ; and mask off interrupts
000051          STA          E_IER
000052          LDA          #BUFMAX
000053          STA          BUFSIZ              ;Set buffer size to maximum
000054          LDA          #00
000055          STA          KEYCNT              ;Flush buffer
000056          STA          BUFHEAD
000057          STA          BUFTAIL
000058          STA          READING              ;No read in progress
000059          STA          ANYKYEVT          ;Disable any key event
000060          STA          ATTNEVT          ;Disable attention event
000061          STA          CTLINDX          ;Abort control function in progress
000062          STA          DSPLYCTL          ;Clear display control char. flag
000063          STA          SUSPFLSH          ;Clear suspend & flush output flags
000064          JSR          CURSOR              ;Remove cursor
000065          LDX          #DFLTLEN
000066          LDA          $010             DFLTVAL-1,X          ;Copy configuration block
000067          STA          DFLTTLBL-1,X
000068          DEX
000069          BNE          $010
000070          JSR          CF_SOH              ;Save screen state & verify
000071          JSR          CURSOR              ;Restore the cursor
000072          JSR          ZPOUT              ;Save screen zero page
000073          PLA
000074          AND          #KYBDENBL          ;Restore previous interrupt state
000075          ORA          #BITON7
000076          STA          E_IER
000077          RTS
000078 ;
000079 CCNTL01      .EQU          *              ;1 -- Console Status Table
000080          CMP          #CONSTLEN
000081          BEQ          $010
000082          LDA          #XCTLPARM
000083          JSR          SYSERR
000084          JSR          CURSOR

```

```

000085          LDY          #CONSTLEN
000086  $020          LDA          (SCLIST),Y
000087          DEY
000088          STA          CONSTTBL,Y
000089          BNE          $020
000090          JSR          VERIFY
000091          JSR          CURSOR
000092          JSR          ZPOUT
000093          RTS
000094  ;
000095  CCNTL02          .EQU          *                               ;2 -- New Line
000096          AND          #BITON7
000097          STA          NEWLINE
000098          INY
000099          LDA          (SCLIST),Y
000100          STA          NEWLNCHR
000101          RTS
000102  ;
000103  CCNTL03          .EQU          *                               ;3 -- Console / Keyboard mode
000104          AND          #BITON7
000105          STA          KYBDMODE
000106          RTS
000107  ;
000108  CCNTL04          .EQU          *                               ;4 -- Buffer Size
000109          CMP          #BUFMAX+1
000110          BCC          $010
000111          LDA          #XCTLPARM
000112          JSR          SYSERR
000113  $010          LDX          #KYBDDSBL
000114          STX          E_IER
000115          STY          KEYCNT
000116          STY          BUFHEAD
000117          STY          BUFTAIL
000118          STA          BUFSIZ
000119          LDX          #KYBDENBL
000120          STX          E_IER
000121          RTS
000122  ;
000123  CCNTL05          LDA          E_IER                               ;5 -- Flush Buffer
000124          PHA
000125          LDA          #KYBDDSBL
000126          STA          E_IER
000127          STY          KEYCNT
000128          STY          BUFHEAD
000129          STY          BUFTAIL

```

```

000130          PLA
000131          AND          #KYBDENBL
000132          ORA          #BITON7
000133          STA          E_IER
000134          RTS
000135 ;
000136 CCNTL06          PHP          ;6 -- Attention Event
000137          SEI
000138          LDY          #5
000139 $010          LDA          (SCLIST),Y
000140          STA          ATTNEVNT,Y
000141          DEY
000142          BPL          $010
000143          PLP
000144          RTS
000145 ;
000146 CCNTL08          PHP          ;8 -- Any Key Event
000147          SEI
000148          LDY          #4
000149 $010          LDA          (SCLIST),Y
000150          STA          ANYKYEVNT,Y
000151          DEY
000152          BPL          $010
000153          PLP
000154          RTS
000155 ;
000156 CCNTL10          .EQU          *          ;10 -- No Wait Input
000157          AND          #BITON7
000158          STA          NOWAIT
000159          RTS
000160 ;
000161 CCNTL11          .EQU          *          ;11 -- Screen Echo
000162          AND          #BITON7+BITON6
000163          STA          ECHO
000164          RTS
000165 ;
000166 CCNTL12          .EQU          *          ;12 -- Character Copy
000167          AND          #BITON7
000168          STA          CHCPYFLG
000169          RTS
000170 ;
000171 CCNTL13          .EQU          *          ;13 -- Character Delete
000172          AND          #BITON7+BITON6
000173          STA          CHDELFLG
000174          RTS

```

```

000175 ;
000176 CCNTL14      .EQU      *                ;14 -- Line Delete
000177           AND      #BITON7+BITON6
000178           STA      LNDELFLG
000179           RTS
000180 ;
000181 CCNTL15      .EQU      *                ;15 -- Escape Functions
000182           AND      #BITON7
000183           STA      ESCAPE
000184           RTS
000185 ;
000186 CCNTL18      .EQU      *                ;18 -- Restore contents of viewport
000187           BIT      DSPLYCTL
000188           BMI      $020
000189           INY
000190           EOR      HMODE
000191           BMI      $010
000192           LDA      (SCLIST),Y
000193           CMP      VPHMAX
000194           BNE      $010
000195           INY
000196           LDA      (SCLIST),Y
000197           CMP      VPVMAX
000198           BNE      $030
000199           LDA      #80
000200           JMP      SCRNDUMP
000201 ;
000202 $010          LDA      (SCLIST),Y
000203           INY
000204           ORA      (SCLIST),Y
000205           BNE      $030
000206 $020          RTS
000207 ;
000208 $030          LDA      #XCTLPARM
000209           JSR      SYSERR
000210
; #####
; #   END OF FILE:  CONS.CNTL.TEXT
; #   LINES       :   210
; #   CHARACTERS  :  9290
; #   Formatter   :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; #   Author      :  David T. Craig -- 71533.606@compuserve.com -- Santa Fe, New Mexico USA
; #####

```

```
; #####
; # PROJECT : Apple /// SOS Console Driver 1.31 (6502 Assembly Source Code)
; # FILE NAME: CONS.DNLD.TEXT
; #####
```

```
000001          .PAGE
000002 ;-----
000003 ;
000004 ; Subroutine LOADCHR
000005 ;
000006 ; This subroutine is called to load an ASCII code and a character
000007 ; image into one of the character download cells in the text pages.
000008 ;
000009 ; LOADCHR requires four bytes of zero page storage for pointers. In
000010 ; order to make it callable from either a device handler or an
000011 ; interrupt processor, all zero page references are indexed by X.
000012 ; On entry, the X register must contain the zero page offset to the
000013 ; character image pointer. The two bytes following the image
000014 ; pointer are used to address the download locations in the text
000015 ; page.
000016 ;
000017 ; Input Parameters:
000018 ;     DNLDCEL -- character download cell number: [0,7]
000019 ;     DNLDCHR -- ASCII character code: [0,7F]
000020 ;     X reg   -- zero page offset to pointers
000021 ;             (0,X) image pointer set by caller
000022 ;             (2,X) download cell pointer set by LOADCHR
000023 ;
000024 ; On exit, DNLDCEL, DNLDCHR, and X will be unchanged. The image
000025 ; pointer will have been incremented by eight. A and Y are destroyed.
000026 ;
000027 ;-----
000028 ;
000029 DIMGPTR      .EQU      00          ;Zero page pointer to image
000030 DCELPTR      .EQU      02          ;Zero page pointer to cell
000031 ;
000032 LOADCHR      .EQU      *
000033             LDY        #00          ;Use Y for row counter
000034 $010        LDA        DNLDCEL     ;Set up cell pointer
000035             AND        #03          ; for ASCII code
000036             ORA        DCPTRL,Y
000037             STA        DCELPTR,X
000038             LDA        DNLDCEL
000039             LSR        A
```

```

000040          LSR          A
000041          CPY          #04
000042          ROL          A
000043          ORA          #08
000044          STA          DCELPTR+1,X
000045          LDA          DNLDCHR          ;Store ASCII code into
000046          STA          (DCELPTR,X)      ; download cell
000047          LDA          DCELPTR+1,X     ;Fix cell pointer
000048          EOR          #0C             ; for character image
000049          STA          DCELPTR+1,X
000050          LDA          (DIMGPTR,X)      ;Store character image
000051          STA          (DCELPTR,X)      ; into download cell
000052          INC          DIMGPTR,X       ;Increment the image pointer
000053          BNE          $020
000054          INC          DIMGPTR+1,X
000055          INY          $020           ;Increment the row number
000056          CPY          #08
000057          BCC          $010           ;Not done yet
000058          RTS
000059          ;
000060          DCPTRL      .EQU          *           ;Table of download cell addresses
000061          .BYTE          078,07C,0F8,0FC
000062          .BYTE          078,07C,0F8,0FC
000063          .PAGE
000064          ;-----
000065          ;
000066          ; Subroutine DNLDINT
000067          ;
000068          ; This subroutine processes the VBL interrupt that signals the
000069          ; completion of a character download cycle. If the REQUEST bit of
000070          ; DNLDFLG is set, another block of eight characters will be
000071          ; downloaded; otherwise, the CB1 and CB2 resources will be
000072          ; released and the ACTIVE bit will be cleared. DNLDINT assumes
000073          ; that the X register points to a four byte area on the zero page
000074          ; that can be used for LOADCHR.
000075          ;
000076          ;-----
000077          ;
000078          DNLDINT      .EQU          *
000079          BIT          DNLDDSBL          ;Disable download
000080          LDA          #VBLDSBL
000081          STA          E_IER            ;Mask VBL interrupts
000082          BIT          DNLDFLG         ;Test REQUEST bit
000083          BVC          $030
000084          CLI          ;Enable interrupts

```

```

000085         LDA         #07
000086         STA         DNLDCCEL           ;Start with cell 7
000087         LDA         DNLDDIMG
000088         STA         DIMGPTR,X         ;Set up IMAGE pointer
000089         LDA         DNLDDIMG+1
000090         STA         DIMGPTR+1,X
000091   $010      JSR         LOADCHR         ;Load one character image
000092         INC         DNLDCCHR         ;Bump character code
000093         BPL         $020
000094         ASL         DNLDDFLG         ;Clear REQUEST bit
000095   $020      DEC         DNLDCCEL         ;Bump cell number
000096         BPL         $010         ;More to do
000097         LDA         DIMGPTR,X
000098         STA         DNLDDIMG         ;Save IMAGE pointer
000099         LDA         DIMGPTR+1,X
000100         STA         DNLDDIMG+1
000101         JMP         DNLDD_GO         ;Enable downloading
000102   ;
000103   $030      ASL         DNLDDFLG         ;Clear ACTIVE bit
000104         LDA         #DNLDDSSIZ
000105         LDX         DNLDDSADR
000106         LDY         DNLDDSADR+1
000107         JSR         DEALCSIR         ;Deallocate SIRs
000108         RTS
000109         .PAGE
000110   ;-----
000111   ;
000112   ; Subroutine GETSIRS
000113   ;
000114   ; This subroutine allocates SIRs 5 & 6 and initializes them to
000115   ; monitor VBL for character downloading.  If the SIRs can not be
000116   ; allocated, it sets an error code and returns directly to the
000117   ; dispatcher.
000118   ;
000119   ;-----
000120   ;
000121   GETSIRS     .EQU         *
000122         BIT         DNLDDFLG         ;Wait for any previous
000123         BMI         GETSIRS         ; request to finish
000124         LDA         #DNLDDSSIZ
000125         LDX         DNLDDSADR
000126         LDY         DNLDDSADR+1
000127         JSR         ALLOCSIR
000128         BCS         $010
000129         PHP

```



```

000130          SEI
000131          LDA          E_PCR          ;Set CB1 to monitor VBL
000132          AND          #0F          ; negative edge and
000133          ORA          #60          ; CB2 to monitor
000134          STA          E_PCR          ; positive edge
000135          LDA          #VBLDSBL
000136          STA          E_IER
000137          PLP
000138          RTS
000139 ;
000140 $010          PLA          ; pull caller's
000141          PLA          ; address, and
000142          LDA          #XNORESRC      ; return to dispatcher
000143          JSR          SYSERR        ; with an error
000144          .PAGE
000145 ;-----
000146 ;
000147 ; Subroutine LOADSET
000148 ;
000149 ; This subroutine is called to initiate downloading of the entire
000150 ; text screen character set. LOADSET calls GETSIRS to set up the
000151 ; 6522 to monitor VBL and interrupt on the negative edge. It then
000152 ; copys the character set to the screen's local data area, sets the
000153 ; request bit, and enables the VBL interrupt. The VBL interrupt
000154 ; processor, DNLDINT, will complete the actual downloading.
000155 ;
000156 ; Parameters:
000157 ; SCLIST: Pointer to caller's 1024 byte character set
000158 ;
000159 ; Zero Page Temporary Storage:
000160 ; WORK1: Pointer to system's character set
000161 ;
000162 ;-----
000163 ;
000164 LOADSET      .EQU          *
000165          JSR          GETSIRS
000166          LDA          #TEXTCSA%100
000167          STA          WORK1
000168          STA          DNLDIMG
000169          LDA          #TEXTCSA/100
000170          STA          WORK1+1
000171          STA          DNLDIMG+1
000172          LDA          #ASC_NUL
000173          STA          DNLDCHR
000174          LDX          #4          ;Set X to move 4 pages

```

```

000175         LDY             #0                ;Set Y to move full page
000176         LDA             SCLIST+1
000177         CMP             #0FB
000178         BCC             $010
000179         SBC             #080                ;Adjust address to avoid
000180         STA             SCLIST+1           ; bank wrap around
000181         INC             1400+SCLIST+1
000182 $010     LDA             (SCLIST),Y      ;Copy character set to
000183         STA             (WORK1),Y        ; text char set buffer
000184         INY
000185         BNE             $010
000186         INC             SCLIST+1
000187         INC             WORK1+1
000188         DEX
000189         BNE             $010
000190         LDA             #0C0                ;Set download active
000191         STA             DNLDIFLG         ; and request flags
000192         LDA             #VBLENBL
000193         STA             E_IER            ;Enable interrupts in VBL neg
000194         RTS
000195         .PAGE
000196 ;-----
000197 ;
000198 ; Subroutine LOAD8
000199 ;
000200 ; This subroutine is called to download up to eight text character
000201 ; images. LOAD8 calls GETSIRS to set up the 6522 to monitor VBL
000202 ; and interrupt on the negative edge. It then loads the character
000203 ; images into the screen's download cells and enables downloading
000204 ; and the VBL interrupt. The download operation is completed by
000205 ; the interrupt processor DNLDINT.
000206 ;
000207 ; Parameters:
000208 ;     SCLIST: Pointer to caller's character sets
000209 ;
000210 ; Zero Page Temporary Data:
000211 ;     COUNT: Number of characters to download
000212 ;     WORK1: Pointer to character image for LOADCHR
000213 ;     WORK2: Work area for LOADCHR
000214 ;
000215 ;-----
000216 ;
000217 LOAD8      .EQU             *
000218         CMP             #01                ;Check download count
000219         BCS             $010

```

```

000220          RTS
000221      ;
000222      $010          CMP          #09
000223          BCC          $020
000224          LDA          #XCTLPARM          ;Too many
000225          JSR          SYSERR
000226      ;
000227      $020          STA          COUNT
000228          JSR          GETSIRS
000229      ;
000230          INC          SCLIST          ;Bump list address
000231          BNE          $030          ; to first character
000232          INC          SCLIST+1
000233      ;
000234      $030          LDA          #08
000235          STA          DNLDCEL
000236      ;
000237      $040          LDY          #00
000238          LDA          (SCLIST),Y          ;Get character code
000239          STA          DNLDCHR
000240          INC          SCLIST          ;Bump list address
000241          BNE          $050          ; to character image
000242          INC          SCLIST+1
000243      ;
000244      $050          LDA          #03
000245          STA          WORK1+1
000246          LDA          DNLDCHR
000247          ASL          A
000248          ASL          A
000249          ROL          WORK1+1          ;Set up address of character
000250          ASL          A          ; image in C00 to FFF space
000251          ROL          WORK1+1
000252          STA          WORK1
000253      ;
000254          LDY          #07
000255      $060          LDA          (SCLIST),Y          ;Copy character image
000256          STA          (WORK1),Y          ; to C00 image space
000257          DEY
000258          BPL          $060
000259      ;
000260          DEC          DNLDCEL
000261          LDX          #WORK1
000262          JSR          LOADCHR          ;Download this character
000263      ;
000264          LDA          DNLDCEL

```

```

000265          CMP          COUNT
000266          BCS          $050          ;Do same character again
000267          LDA          #08
000268          ADC          SCLIST        ;Bump list address
000269          STA          SCLIST        ; to next character
000270          BCC          $070
000271          INC          SCLIST+1
000272 $070      DEC          COUNT
000273          BNE          $040
000274 ;
000275          LDA          #080          ;Set download active
000276          STA          DNLDFLG
000277 DNLD_GO   BIT          DNLDENBL
000278          LDA          #VBLCLR
000279          STA          E_IFR          ;Clear both VBL flags
000280 $080      BIT          E_IORB        ;Check composite blanking
000281          BVC          $090
000282          BIT          E_IFR          ;Check VBL flags
000283          BEQ          $080
000284 $090      STA          E_IFR        ;Clear VBL flags
000285          LDA          #VBLENBL        ;Enable VBL interrupt
000286          STA          E_IER
000287          RTS
000288

; #####
; #   END OF FILE:  CONS.DNLD.TEXT
; #   LINES       :   288
; #   CHARACTERS  :  14806
; #   Formatter   :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; #   Author      :  David T. Craig -- 71533.606@compuserve.com -- Santa Fe, New Mexico USA
; #####

```

```

; #####
; #   PROJECT : Apple /// SOS Console Driver 1.31 (6502 Assembly Source Code)
; #   FILE NAME: CONS.MISC.TEXT
; #####

000001          .PAGE
000002 ;-----
000003 ;
000004 ; Console Open Request
000005 ;
000006 ;-----
000007 ;
000008 CNSLOPEN      .EQU      *
000009             BIT        OPENFLG          ;Console open?
000010             BPL        $010            ; No
000011             LDA        #XNOTAVIL
000012             JSR        SYSERR
000013 ;
000014 $010          LDA        #KYBDSSIZ          ;Allocate the keyboard interrupt
000015             LDX        KYBDSADR
000016             LDY        KYBDSADR+1
000017             JSR        ALLOCSIR
000018             BCS        $020
000019             LDA        #TRUE
000020             STA        OPENFLG          ;Set console open
000021             JSR        CCNTL00          ;Reset console parameters
000022             PHP
000023             SEI
000024             LDA        E_PCR
000025             AND        #0F1
000026             ORA        #002            ;Set up keyboard interrupt
000027             STA        E_PCR
000028             PLP
000029             LDA        #KYBDCLR
000030             STA        E_IFR          ;Clear keyboard flag
000031             BIT        KYBDSTRB        ;Clear the keyboard strobe
000032             LDA        #KYBDENBL
000033             STA        E_IER          ;Unmask keyboard interrupts
000034             CLC
000035             RTS
000036 ;
000037 $020          LDA        #XNORESRC        ;Couldn't get keyboard resource
000038             JSR        SYSERR
000039             .PAGE

```

```

000040 ;-----
000041 ;
000042 ; Console Close Request
000043 ;
000044 ;-----
000045 ;
000046 CNSLCLOS      .EQU      *
000047              ASL        OPENFLG      ;Console open?
000048              BCS        $010         ; Yes
000049              JMP        CNOTOPEN
000050 ;
000051 $010          BIT        DNLDFLG      ;Wait for pending download
000052              BMI        $010
000053              LDA        #KYBDDSBLL
000054              STA        E_IER          ;Mask keyboard interrupts
000055              BIT        KYBDSTRB       ;Clear the keyboard strobe
000056              LDA        #KYBDSSIZ
000057              LDX        KYBDSADR
000058              LDY        KYBDSADR+1
000059              JSR        DEALCSIR      ;Deallocate the keyboard interrupt
000060              RTS
000061              .PAGE
000062 ;-----
000063 ;
000064 ; Console Initialization Request
000065 ;
000066 ;-----
000067 ;
000068 CNSLINIT      .EQU      *
000069              LDA        #FALSE
000070              STA        OPENFLG
000071              LDA        B_REG          ;Set bank register for
000072              STA        KYBDBANK      ; keyboard and download
000073              STA        DNLDDBANK     ; interrupt handlers
000074              LDA        #TEXTCSA%100 ;Set up character download call
000075              STA        SCLIST
000076              LDA        #TEXTCSA/100
000077              STA        SCLIST+1
000078              LDA        #00
000079              STA        1400+SCLIST+1
000080              JSR        LOADSET
000081              CLC
000082              RTS
000083

```

```
; #####  
; #   END OF FILE:  CONS.MISC.TEXT  
; #   LINES       :   83  
; #   CHARACTERS  :  4017  
; #   Formatter   :  Assembly Language Reformatter 1.0.2 (07 January 1998)  
; #   Author      :  David T. Craig -- 71533.606@compuserve.com -- Santa Fe, New Mexico USA  
; #####
```

```

; #####
; # PROJECT : Apple /// SOS Console Driver 1.31 (6502 Assembly Source Code)
; # FILE NAME: CONS.UTL1.TEXT
; #####

```

```

000001          .PAGE
000002 ;-----
000003 ;
000004 ; Subroutine VERIFY
000005 ;
000006 ; This subroutine checks the screen's hardware mode, software mode,
000007 ; and viewport parameters for self consistency. It also sets the
000008 ; screen switches and the following internal variables:
000009 ; HMODE, SMINV, SMCURSOR, SMSCROLL, SMAUTOCR, SMAUTOLF,
000010 ; SMAUTOADV, VPHMAX, VPVMAX, and TCOLOR
000011 ;
000012 ; Parameters: none
000013 ;
000014 ; Exit:
000015 ; A, X, Y: Undefined
000016 ;
000017 ;-----
000018 ;
000019 VERIFY          .EQU          *
000020                LDA           HMODE          ;Validate HMODE
000021                AND           #03           ; and set 80 column
000022                ASL           A             ; flag in bit 7
000023                CMP           #04
000024                BCC           $010
000025                LDA           #04
000026 $010          ROR           A
000027                STA           HMODE
000028                LDY           SMODE          ;Preserve SMODE
000029                LDA           #00
000030                LDX           #5
000031 $020          STA           SMFLAGS,X
000032                LSR           SMODE          ;Set SMODE flags
000033                ROR           SMFLAGS,X
000034                DEX
000035                BPL           $020
000036                STY           SMODE
000037                LDA           #79.
000038                BIT           HMODE          ;Screen width := If 80 column,
000039                BMI           $100          ; then 79.

```



```

000040          LDA          #39.          ; else 39.
000041  $100      CMP          VPR          ;VPR <= Screen width
000042          BCS          $110
000043          STA          VPR
000044  $110      LDA          VPR
000045          CMP          VPL          ;VPL <= VPR
000046          BCS          $120
000047          STA          VPL
000048  $120      SEC
000049          LDA          VPR          ;VPHMAX :=
000050          SBC          VPL          ; VPR - VPL
000051          STA          VPHMAX
000052          CMP          TPX          ;TPX <= VPHMAX
000053          BCS          $200
000054          STA          TPX
000055  $200      LDA          #23.
000056          CMP          VPB          ;VPB <= Screen height
000057          BCS          $210
000058          STA          VPB
000059  $210      LDA          VPB
000060          CMP          VPT          ;VPT <= VPB
000061          BCS          $220
000062          STA          VPT
000063  $220      SEC
000064          LDA          VPB          ;VPVMAX :=
000065          SBC          VPT          ; VPB - VPT
000066          STA          VPVMAX
000067          CMP          TPY          ;TPY <= VPVMAX
000068          BCS          $300
000069          STA          TPY
000070  $300      LDA          TCB
000071          AND          #0F          ;TCB=TCB MOD 16
000072          STA          TCB
000073          LDA          TCF
000074          AND          #0F          ;TCF=TCF MOD 16
000075          STA          TCF
000076          ASL          A
000077          ASL          A
000078          ASL          A          ;SET TCOLOR :=
000079          ASL          A
000080          ORA          TCB          ;TCF * 16 + TCB
000081          STA          TCOLOR
000082          PHP
000083          SEI
000084          LDA          SCRNMODE      ;Check screen mode

```

```

000085         ASL          A
000086         BMI          $500                ; Graphics
000087         LDA          E_REG
000088         ORA           #BITON5
000089         BCS          $400
000090         AND           #BITOFF5
000091  $400      STA          E_REG
000092         LDA          HMODE
000093         AND           #03
000094         BCC          $410
000095         ORA           #BITON7
000096  $410      STA          SCRNMODE        ;Set screen mode
000097         LSR          A
000098         AND           #01
000099         TAY
000100         LDA          #00
000101         ROL          A
000102         TAX
000103         LDA          VMODE0,X            ;B&W / Color
000104         LDA          VMODE1,Y            ;40 / 80 Column
000105         BIT          VMODE2            ;Page 1 always
000106         BIT          VMODE3            ;Text of course
000107  $500      PLP
000108         JSR          TBASCAL            ;New base addr.
000109         RTS
000110         .PAGE
000111 ;-----
000112 ;
000113 ; Subroutine CURSOR
000114 ;
000115 ; This subroutine displays or removes the cursor by inverting the
000116 ; character at the current cursor position.
000117 ;
000118 ; Parameters: none
000119 ;
000120 ; Exit:
000121 ; A, X, Y: Undefined
000122 ;
000123 ;-----
000124 ;
000125 CURSOR      .EQU          *
000126         BIT          SMCURSOR            ;is cursor enabled?
000127         BPL          $020                ;if not, exit
000128         LDA          TPX
000129         BIT          HMODE

```

```

000130          BPL          $010          ;40 col: X=TPX
000131          LSR          A          ;80 col: X=TPX/2
000132          BCC          $010
000133          TAY
000134          LDA          (BASE2),Y          ;get character
000135          EOR          #80          ;and invert it
000136          STA          (BASE2),Y          ;put it back
000137          RTS
000138 $010          TAY
000139          LDA          (BASE1),Y          ;get character
000140          EOR          #80          ;and invert it
000141          STA          (BASE1),Y          ;put it back
000142 $020          RTS
000143          .PAGE
000144 ;-----
000145 ;
000146 ; Single Character Screen Read (Console character copy)
000147 ;
000148 ; This subroutine returns the character at the current cursor position.
000149 ;
000150 ; Parameters: none
000151 ;
000152 ; Exit:
000153 ; A: character
000154 ; X, Y: Undefined
000155 ;
000156 ;
000157 ;-----
000158 ;
000159 SCRNPICK      .EQU      *
000160          LDA          TPX
000161          TAY
000162          BIT          HMODE
000163          BPL          $010          ;40 Col -- Y := TPX
000164          LSR          A          ;80 Col -- Y := TPX/2
000165          TAY
000166          BCC          $010
000167          LDA          (BASE2),Y          ;Read odd text page
000168          BCS          $020
000169 $010          LDA          (BASE1),Y          ;Read even text page
000170 $020          RTS
000171          .PAGE
000172 ;-----
000173 ;
000174 ; Subroutine TBASCAL -- Text Base Address Calculator

```

```

000175 ;
000176 ; This subroutine sets the base address registers, BASE1 and BASE2,
000177 ; to point to the current line in screen memory. BASE1 always points
000178 ; to column 0 of the current viewport while BASE2 points to column 1.
000179 ;
000180 ; Entry TBASCAL:
000181 ;   Parameters: none
000182 ;
000183 ; Entry TBASCAL1:
000184 ;   Parameters:
000185 ;     X: Absolute screen line number
000186 ;
000187 ; Exit (either entry point):
000188 ;   A: Undefined
000189 ;   X: Absolute screen line number
000190 ;   Y: Unchanged
000191 ;
000192 ;-----
000193 ;
000194 TBASCAL      .EQU      *
000195              CLC
000196              LDA      TPY              ;vertical position
000197              ADC      VPT              ; + viewport top
000198              TAX
000199 TBASCAL1    .EQU      *
000200              CLC
000201              LDA      VPL              ;viewport left:
000202              BIT      HMODE
000203              BPL      $010            ;if 80 column mode,
000204              LSR      A              ;then divide by two
000205 $010        PHP
000206              ADC      BASL,X          ;base address (LO)
000207              STA      BASE1           ;same for both pages
000208              STA      BASE2
000209              LDA      BASH,X         ;base address (HI)
000210              PLP
000211              BCC      $020
000212              DEC      BASE1           ;Odd window adjustment
000213              EOR      #0C
000214 $020      STA      BASE1+1         ;even page address
000215              EOR      #0C
000216              STA      BASE2+1       ;odd page address
000217              RTS
000218              .PAGE
000219 ;-----

```

```

000220 ;
000221 ; Subroutine CLREOL -- Clear to End of Line
000222 ;
000223 ; This subroutine clears the current line from the current cursor
000224 ; position to the end of the line. The starting position may be
000225 ; passed in Y using the CLREOL1 entry point. The text base address
000226 ; pointers, BASE1 and BASE2, must point to the current line.
000227 ;
000228 ; Entry CLREOL:
000229 ;   Parameters: none
000230 ;
000231 ; Entry CLREOL1:
000232 ;   Parameters:
000233 ;     Y: Starting horizontal position
000234 ;
000235 ; Zero Page Temporary Storage:
000236 ;   BLANK, TEMPX
000237 ;
000238 ; Exit (either entry point):
000239 ;   A, Y: Undefined
000240 ;   X: Preserved
000241 ;
000242 ;-----
000243 ;
000244 CLREOL      .EQU      *
000245           LDY          TPX                ;horizontal position
000246 CLREOL1    .EQU      *
000247           LDA          #80+ASC_SP         ;Set up a blank
000248           EOR          SMINV
000249           STA          BLANK
000250           BIT          HMODE
000251           BPL          $200
000252           TYA
000253           BNE          $150
000254 ;
000255 ; 80 column clear full line
000256 ;
000257           LDA          VPHMAX             ;Start at right edge
000258           LSR          A
000259           TAY
000260           LDA          BLANK             ;Load the blank
000261           BCC          $110
000262 $100       STA          (BASE2),Y        ;Clear odd column
000263 $110       STA          (BASE1),Y        ; then even column
000264           DEY

```

```

000265          BPL          $100          ;Repeat to BOL
000266          RTS
000267 ;
000268 ; 80 column clear to end of line
000269 ;
000270 $150      STX          TEMPX        ;Save X
000271          CLC
000272          SBC          VPHMAX        ;Calculate negative number
000273          TAX          ; of bytes to blank
000274          TYA
000275          LSR          A
000276          TAY
000277          LDA          BLANK        ;Load the blank
000278          BCS          $170
000279 $160      STA          (BASE1),Y
000280          INX
000281          BPL          $180
000282 $170      STA          (BASE2),Y
000283          INY
000284          INX
000285          BMI          $160
000286 $180      LDX          TEMPX        ;Restore X
000287          RTS
000288 ;
000289 ; 40 column clear to end of line
000290 ;
000291 $200      LDA          BLANK
000292          STA          (BASE1),Y
000293          LDA          TCOLOR
000294          STA          (BASE2),Y
000295          CPY          VPHMAX
000296          INY
000297          BCC          $200
000298          RTS
000299

; #####
; #   END OF FILE:  CONS.UTL1.TEXT
; #   LINES       :   299
; #   CHARACTERS  :  13475
; #   Formatter   :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; #   Author      :  David T. Craig -- 71533.606@compuserve.com -- Santa Fe, New Mexico USA
; #####

```

```
; #####
; # PROJECT : Apple /// SOS Console Driver 1.31 (6502 Assembly Source Code)
; # FILE NAME: CONS.UTL2.TEXT
; #####
```

```
000001          .PAGE
000002 ;-----
000003 ;
000004 ; Subroutine CLREOS -- Clear to End of Screen
000005 ;
000006 ; This subroutine clears the screen from the current cursor position
000007 ; to the end of the viewport. The CLREOS1 entry allows the line number
000008 ; to be passed in X and the starting column number in Y.
000009 ;
000010 ; Entry CLREOS:
000011 ;   Parameters: none
000012 ;
000013 ; Entry CLREOS1:
000014 ;   Parameters:
000015 ;     X: Starting absolute line number
000016 ;     Y: Starting column number
000017 ;
000018 ; Exit:
000019 ;   A, X, Y: Undefined
000020 ;
000021 ;-----
000022 ;
000023 CLREOS          .EQU          *
000024                CLC
000025                LDA          TPY
000026                ADC          VPT
000027                TAX
000028                LDY          TPX
000029 CLREOS1         .EQU          *
000030 $010           JSR          TBASCAL1
000031                JSR          CLREOL1
000032                LDY          #0
000033                CPX          VPB
000034                INX
000035                BCC          $010
000036                JMP          TBASCAL
000037                .PAGE
000038 ;-----
000039 ;
```

```

000040 ; Scroll Text Viewport
000041 ;
000042 ; This subroutine scrolls the text within the viewport up or down by
000043 ; one line. On entry, A must contain an UP/DOWN flag ( $00 => UP,
000044 ; $80 => DOWN ).
000045 ;
000046 ; Parameters:
000047 ;     A:  Up / Down flag
000048 ;
000049 ; Zero Page Temporary Storage:
000050 ;     WORK1, WOR2:  Screen pointers
000051 ;     FLAGS:  Bit 7 -- even / odd flag for scroll loop
000052 ;             Bit 6 -- up / down flag
000053 ;     TEMP1:  Starting Y index for scroll loop
000054 ;
000055 ; Subroutines called:
000056 ;     TBASCAL1, CLREOL1
000057 ;
000058 ; Exit:
000059 ;     A, X, Y:  Undefined
000060 ;
000061 ;-----
000062 ;
000063 SCROLL      .EQU      *
000064           STA      FLAGS           ;Save UP/DOWN flag
000065           SEC
000066           LDA      VPHMAX
000067           BIT      HMODE
000068           BPL      $010
000069           LSR      A
000070 $010      STA      TEMP1          ;Get starting loop index
000071           ROR      FLAGS          ;Save even/odd flag
000072           LDX      VPT
000073           BIT      FLAGS
000074           BVC      $020
000075           LDX      VPB
000076 $020      JSR      TBASCAL1      ;Get starting base addresses
000077           ;
000078 $030      BIT      FLAGS
000079           BVC      $040
000080           CPX      VPT           ;Scroll down
000081           BEQ      $080          ; All done
000082           DEX
000083           BPL      $050          ; Go up one line
000084 $040      CPX      VPB           ;Scroll up

```



```

000085          BEQ          $080          ; All done
000086          INX          ; Go down one line
000087 ;
000088 $050      LDA          BASE1
000089          LDY          BASE1+1        ;Copy source address
000090          STA          WORK1         ; to destination address
000091          STY          WORK1+1
000092          LDA          BASE2
000093          LDY          BASE2+1
000094          STA          WORK2
000095          STY          WORK2+1
000096          JSR          TBASCAL1      ;Get next source address
000097          LDY          TEMP1
000098          BIT          FLAGS
000099          BPL          $070
000100 $060      LDA          (BASE2),Y    ;Scroll this line
000101          STA          (WORK2),Y    ; move odd column
000102 $070      LDA          (BASE1),Y    ; move even column
000103          STA          (WORK1),Y
000104          DEY
000105          BPL          $060
000106          BMI          $030
000107 ;
000108 $080      LDY          #0
000109          JSR          CLREOL1      ;Clear last line
000110          JMP          TBASCAL
000111          .PAGE
000112 ;-----
000113 ;
000114 ;   Horizontal Shift
000115 ;
000116 ;   This subroutine shifts the text within the viewport left or right.
000117 ;   On entry, A must contain an eight bit signed shift offset, negative
000118 ;   for left shifts and positive for right shifts.
000119 ;
000120 ;   Parameters:
000121 ;       A:   Signed shift offset
000122 ;
000123 ;   Zero Page Temporary Storage:
000124 ;       BLANK, TEMPX
000125 ;       WORK1, WORK2:  Screen pointers
000126 ;       FLAGS:  Bit 7 -- right / left flag
000127 ;               Bit 6 -- odd / even flag for shift right
000128 ;       TEMP1:  Positive shift offset
000129 ;       TEMP2:  Absolute shift column

```

```

000130 ;      TEMP3:  shift right -- starting shift index
000131 ;      shift left -- shift count
000132 ;      TEMP4:  shift right -- starting clear index
000133 ;      shift left -- column for clear
000134 ;
000135 ; Subroutines Called:
000136 ;      CLREOS1, CLREOL1
000137 ;
000138 ; Exit:
000139 ;      A, X, Y:  Undefined
000140 ;
000141 ;-----
000142 ;
000143 SHIFT      .EQU      *
000144          TAY
000145          BEQ          $020          ;Nothing to do
000146          AND          #BITON7
000147          STA          FLAGS          ;Set right / left flag
000148          TYA
000149          CMP          #80
000150          BCC          $010
000151          EOR          #0FF          ;Make shift count positive
000152 $010     ADC          #00
000153          STA          TEMP1          ;Absolute shift offset
000154          ADC          VPL          ;Absolute column number
000155          STA          TEMP2          ; for base address
000156          LDX          VPT
000157          SEC
000158          LDA          VPHMAX
000159          SBC          TEMP1
000160          BCS          $030
000161          LDY          #00          ;Shift entire viewport
000162          JSR          CLREOS1
000163 $020     RTS
000164 $030     BIT          FLAGS
000165          BMI          $060
000166 ;
000167          SEC          ;Set up for shift right
000168          BIT          HMODE
000169          BPL          $040
000170          LSR          A
000171 $040     STA          TEMP3          ;Set starting index for shifting
000172          LDA          #BITON6
000173          BCS          $050
000174          LDA          #00

```

```

000175 $050      ORA      FLAGS      ;Set odd / even flag
000176          STA      FLAGS
000177          LDY      TEMP1
000178          DEY
000179          STY      TEMP4      ;Set index for clearing
000180          LDA      #80+ASC_SP
000181          EOR      SMINV
000182          STA      BLANK      ;Set up space character
000183          JMP      SHIFT1
000184 ;
000185 $060      TAY          ;Set up for shift left
000186          BIT      HMODE
000187          BMI      $070
000188          SEC
000189          ROL      A
000190 $070      STA      TEMP3      ;Set count for shifting
000191          INY
000192          STY      TEMP4      ;Set index for clearing
000193 ;
000194 SHIFT1     JSR      TBASCAL1    ;Get base address
000195          CLC
000196          LDA      TEMP2
000197          BIT      HMODE
000198          BPL      $010
000199          LSR      A
000200 $010      PHP
000201          ADC      BASL,X
000202          STA      WORK1      ;Get shifted base address
000203          STA      WORK2
000204          LDA      BASH,X
000205          PLP
000206          BCC      $020
000207          DEC      WORK1
000208          EOR      #0C
000209 $020      STA      WORK1+1
000210          EOR      #0C
000211          STA      WORK2+1
000212          BIT      FLAGS
000213          BMI      SHFTLF
000214 ;
000215 SHFTRT     LDY      TEMP3      ;Shift this line right
000216          BVC      $020
000217 $010      LDA      (BASE2),Y
000218          STA      (WORK2),Y
000219 $020      LDA      (BASE1),Y

```

```

000220          STA          (WORK1),Y
000221          DEY
000222          BPL          $010
000223          LDA          TEMP4          ;Clear beginning of line
000224          BIT          HMODE
000225          BPL          $050
000226          LSR          A
000227          TAY
000228          LDA          BLANK
000229          BCC          $040
000230  $030      STA          (BASE2),Y
000231  $040      STA          (BASE1),Y
000232          DEY
000233          BPL          $030
000234          BMI          SHIFT2
000235  $050      TAY
000236  $060      LDA          BLANK
000237          STA          (BASE1),Y
000238          LDA          TCOLOR
000239          STA          (BASE2),Y
000240          DEY
000241          BPL          $060
000242  ;
000243  SHIFT2    CPX          VPB
000244          INX          ;Go to next line
000245          BCC          SHIFT1
000246          JMP          TBASCAL
000247  ;
000248  SHFTLF    LDY          #00          ;Shift this line left
000249          STX          TEMPX
000250          LDX          TEMP3          ;Get shift count
000251  $010      LDA          (WORK1),Y
000252          STA          (BASE1),Y
000253          DEX
000254          BMI          $020
000255          LDA          (WORK2),Y
000256          STA          (BASE2),Y
000257          INY
000258          DEX
000259          BPL          $010
000260  $020      LDX          TEMPX
000261          LDY          TEMP4
000262          JSR          CLREOL1
000263          JMP          SHIFT2
000264          .PAGE

```

```

000265 ;-----
000266 ;
000267 ; Dump and Restore Contents of Viewport
000268 ;
000269 ; This subroutine will dump or restore the contents of the viewport to
000270 ; or from the caller's buffer. On entry, A must contain a dump/restore
000271 ; flag. ($00 => Dump $80 => Restore)
000272 ;
000273 ; Parameters:
000274 ;     A:  Dump / Restore flag
000275 ;
000276 ; Zero Page Temporary Storage:
000277 ;     WORK1, WORK2:  Extended pointers to caller's buffer
000278 ;     FLAGS:  Bit 7 -- odd / even move count flag
000279 ;             Bit 6 -- dump / restore flag
000280 ;     TEMP1:  Starting move index
000281 ;     TEMP2:  Move count
000282 ;
000283 ; Exit:
000284 ;     A, X, Y:  Undefined
000285 ;
000286 ;-----
000287 ;
000288 SCRNDUMP      .EQU      *
000289             STA      FLAGS
000290             JSR      CURSOR           ;Turn cursor off
000291             LDA      VPHMAX
000292             STA      TEMP2
000293             INC      TEMP2
000294             BIT      HMODE
000295             BMI      $010
000296             ASL      TEMP2
000297             ASL      A
000298 $010         LSR      A
000299             STA      TEMP1
000300             ROR      FLAGS
000301             CLC
000302             LDA      SCLIST
000303             ADC      #03             ;Set work pointers to
000304             STA      WORK1         ; to caller's buffer
000305             LDA      SCLIST+1
000306             ADC      #00
000307             CMP      #0F0
000308             LDX      1401+SCLIST
000309             BCC      $020

```

```

000310          SBC          #80          ;Adjust extended address
000311          INX
000312  $020     STA          WORK1+1
000313          STX          1401+WORK1
000314          LDA          TEMP2
000315          LSR          A
000316          ADC          WORK1
000317          STA          WORK2
000318          LDA          WORK1+1
000319          ADC          #00
000320          STA          WORK2+1
000321          STX          1401+WORK2
000322  ;
000323  ; Copy the contents of the window
000324  ;
000325          LDX          VPT
000326  $100     JSR          TBASCAL1
000327          LDY          TEMP1
000328          BIT          FLAGS
000329          BVS          $120
000330  ;
000331          BPL          $115
000332  $110     LDA          (BASE2),Y
000333          STA          (WORK2),Y
000334  $115     LDA          (BASE1),Y
000335          STA          (WORK1),Y
000336          DEY
000337          BPL          $110
000338          BMI          $140
000339  ;
000340  $120     BPL          $135
000341  $130     LDA          (WORK2),Y
000342          STA          (BASE2),Y
000343  $135     LDA          (WORK1),Y
000344          STA          (BASE1),Y
000345          DEY
000346          BPL          $130
000347  ;
000348  $140     CLC
000349          LDA          WORK1
000350          ADC          TEMP2
000351          STA          WORK1
000352          BCC          $150
000353          INC          WORK1+1
000354  $150     CLC

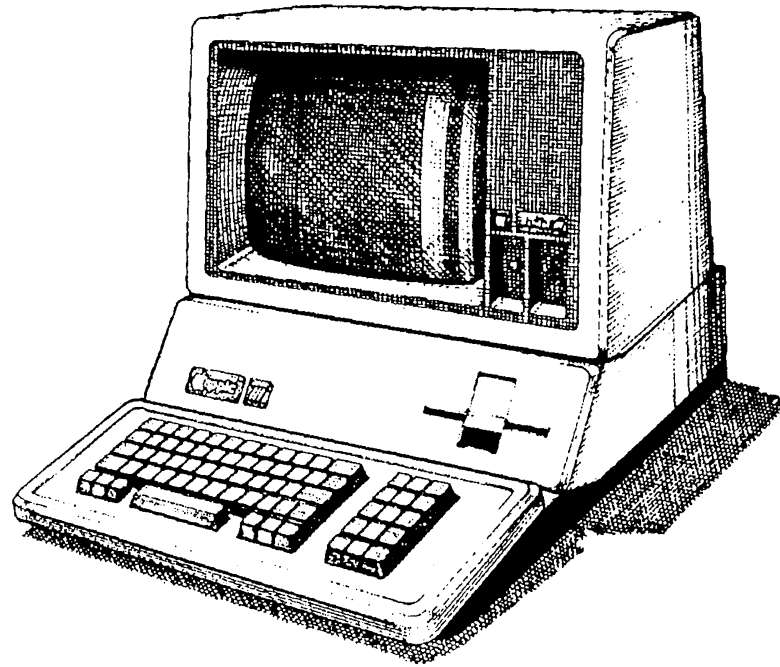
```

```

000355          LDA          WORK2
000356          ADC          TEMP2
000357          STA          WORK2
000358          BCC          $160
000359          INC          WORK2+1
000360 $160      CPX          VPB
000361          INX
000362          BCC          $100
000363 ;
000364          JSR          TBASCAL
000365          JSR          CURSOR          ;Restore cursor
000366          RTS
000367          .PAGE
000368 ;-----
000369 ;
000370 ; ZPOUT
000371 ;
000372 ; This subroutine saves the driver's zero page data.
000373 ;
000374 ;-----
000375 ;
000376 ZPOUT      LDX          #ZPLENGTH-1          ;Zero Page save area length
000377 $010      LDA          ZPDATA,X
000378          STA          ZPSAVE,X
000379          DEX
000380          BPL          $010
000381          RTS
000382

; #####
; #   END OF FILE:  CONS.UTL2.TEXT
; #   LINES       :   382
; #   CHARACTERS  :  16487
; #   Formatter   :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; #   Author      :  David T. Craig -- 71533.606@compuserve.com -- Santa Fe, New Mexico USA
; #####

```



**The End**