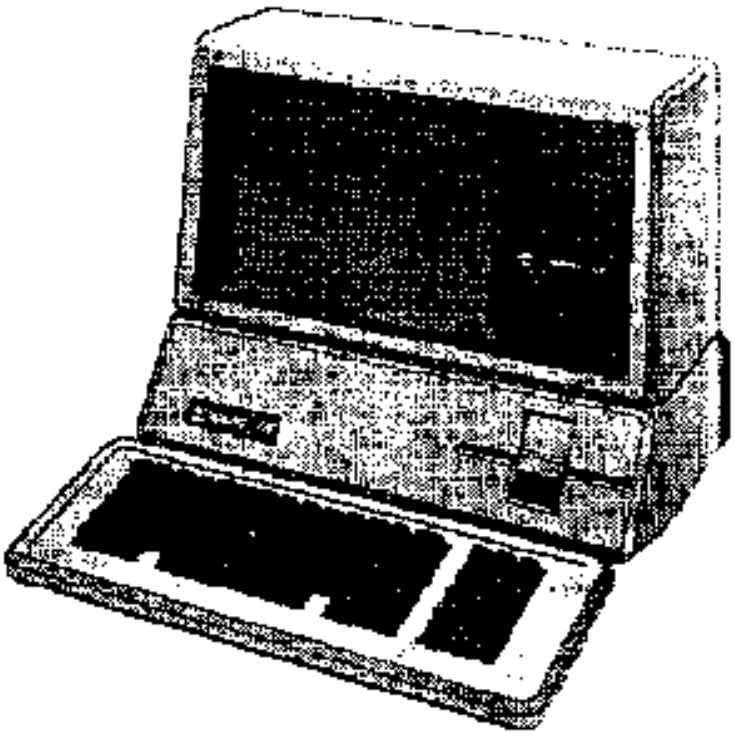
Apple /// Computer Technical Information

Apple ///
ProFile Hard Disk
Driver 1.30
Source Code Listing



Created by David T. Craig
07 January 1998
71533.606@compuserve.com

FORMATTED LISTINGS

```
; ##############################################################################################################
; #   PROJECT  :  Apple /// SOS Profile Driver 1.30 (6502 Assembly Source Code)
; #   FILE NAME:  PROFILE.TEXT
; ##############################################################################################################

000001                  .NOPATCHLIST
000002                  .TITLE       "SOS Profile Driver -- Version 1.30  14-Jan-83"
000003
000004  ;---------------------------------------------------------------------
000005  ;
000006  ;               SOS Profile Driver
000007  ;
000008  ;
000009  ;       Revisions:
000010  ;
000011  ;       1.10R   05-May-82
000012  ;
000013  ;       The GOODEXIT routine was changed to complete the dummy handshake
000014  ;       so any pending Profile spare table updates would get rewritten
000015  ;       on the disk.
000016  ;       The GETBYTES routine was changed so a bad block number would get
000017  ;       flagged as such and not get flagged as a bad byte count.
000018  ;
000019  ;       1.11R   12-May-82
000020  ;
000021  ;       The driver is slowed to 1MHz only when talking to the card and
000022  ;       doing psuedo DMA; this will enable it to make the 5:1 interleave.
000023  ;       The reference to the clear parity address that occurs just after
000024  ;       entering the driver was corrected to be a write instead of a read.
000025  ;
000026  ;       1.11R   14-May-82
000027  ;
000028  ;       The block ID check was removed when reading to speed up the driver
000029  ;       and allow reading the status info which doesn't have a block ID.
000030  ;       The 2nd reset, which prevented the read/write head from being
000031  ;       retracted off the data area on read and write errors, was removed.
000032  ;
000033  ;       1.12R   01-Sep-82
000034  ;
000035  ;       Interrupts were disabled in the LAST_PGE routine before saving
000036  ;       bytes 0 and FE, instead of just before the data transfer, since
000037  ;       those bytes might be changed by an interrupt routine, and the old
000038  ;       instead of the new (correct) values would get restored.
000039  ;
000040  ;       1.30    14-Jan-83
000041  ;
000042  ;       Interrupts are disabled while modifying the environment register
000043  ;       and RSTORENV is changed to leave the screen bit (bit 5) unmodified;
000044  ;       this eleminates spurious screen flashing.  Control code 0 is
000045  ;       processed as a NOP, instead of an error.
000046  ;
000047  DEVTYPE         .EQU         0D1
000048  SUBTYPE         .EQU         01
000049  MANUF           .EQU         0001                    ;Apple Computer Inc.
000050  RELEASE         .EQU         1300
000051  MAXBLOCK        .EQU         2600                    ;4.86 megabytes (+16K spares)
000052                  .PAGE
000053  ;---------------------------------------------------------------------
000054  ;
000055  ;  The macro SWITCH performs an N way branch based on a switch index.
000056  ;
000057  ;               SWITCH  [index], [bounds], adrs_table, [*]
000058  ;
000059  ;---------------------------------------------------------------------
000060
000061                  .MACRO       SWITCH
000062                  .IF          "%1" <> ""              ;If PARM1 is present,
000063                  LDA          %1                      ;  Load A with switch index
000064                  .ENDC
000065                  .IF          "%2" <> ""              ;If PARM2 is present,
000066                  CMP          #%2+1                   ;  Perform bounds checking
000067                  BCS          $010                    ;  on switch index
000068                  .ENDC
000069                  ASL          A
000070                  TAY
000071                  LDA          %3+1,Y                  ;Get switch address from table
000072                  PHA                                  ;  and push onto stack
000073                  LDA          %3,Y
000074                  PHA
000075                  .IF          "%4" <> "*"             ;If PARM4 is omitted,
000076                  RTS                                  ;  Exit to code
000077                  .ENDC                                ;Otherwise, drop through
000078  $010            .ENDM
000079
000080                  .INCLUDE     PROFILE.A.TEXT
000081                  .INCLUDE     PROFILE.B.TEXT
000082
000083                  .END
000084
```

```
; ##############################################################################################################
; #   END OF FILE:  PROFILE.TEXT
; #   LINES      :  84
; #   CHARACTERS :  4045
; #   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 Profile Driver 1.30 (6502 Assembly Source Code)
; #   FILE NAME:  PROFILE.A.TEXT
; ##########################################################################################################

000001                  .WORD       0FFFF
000002                  .WORD       59.
000003                  .ASCII      "Profile Driver -- "
000004                  .ASCII      "Copyright (C) 1983 by Apple Computer Inc."
000005
000006  ;---------------------------------------------------------------------
000007  ;
000008  ;       Device Information Block (DIB)
000009  ;
000010  ;---------------------------------------------------------------------
000011
000012  DIB_LINK1       .WORD       0
000013  DIB_ENTRY1      .WORD       MAIN
000014  DIB_NAME1       .BYTE       8
000015                  .ASCII      ".PROFILE"
000016                  .BLOCK      7,0
000017  DIB_DNUM1       .BYTE       80                      ;active, no page alignment
000018  DIB_SLOT1       .BYTE       0FF
000019  DIB_UNIT1       .BYTE       0
000020  DIB_TYPE1       .BYTE       DEVTYPE
000021  DIB_SUBTYPE1    .BYTE       SUBTYPE
000022                  .BYTE       0
000023  DIB_BLOCK1      .WORD       MAXBLOCK
000024  DIB_MID1        .WORD       MANUF
000025  DIB_RLS1        .WORD       RELEASE
000026
000027  DIB_DCBCNT1     .WORD       0001                    ;Configuration Block
000028  WRTVER          .BYTE       0FF
000029                  .PAGE
000030  ;---------------------------------------------------------------------
000031  ;
000032  ;       SOS Global Equates  (jump table entry points)
000033  ;
000034  ;---------------------------------------------------------------------
000035
000036  ALLOCSIR        .EQU        1913                    ;allocate System Interrupt Resource
000037  DEALCSIR        .EQU        1916                    ;deallocate  "       "        "
000038  SELC800         .EQU        1922                    ;select/deselect i/o expansion space
000039  SYSERR          .EQU        1928                    ;system error routine
000040  DO_DMA          .EQU        18F0                    ;place to reloc code for banking
000041  INDDMA          .EQU        0F0
000042  VECTLO          .EQU        DO_DMA+7
000043  SOS_ZPAGE       .EQU        18
000044
000045  ;---------------------------------------------------------------------
000046  ;
000047  ;       SOS Error Codes
000048  ;
000049  ;---------------------------------------------------------------------
000050
000051  XREQCODE        .EQU        20                      ;Invalid request code
000052  XCTLCODE        .EQU        21                      ;Invalid control/status code
000053  XNORESRC        .EQU        25                      ;resource not available
000054  XBADOP          .EQU        26                      ;Invalid operation
000055  XIOERROR        .EQU        27                      ;I/O error
000056  XNODRIVE        .EQU        28                      ;No drive connected
000057  XNOWRITE        .EQU        2B                      ;Device write protected.(not supported)
000058  XBYTECNT        .EQU        2C                      ;Byte count <> a multiple of 512
000059  XBLKNUM         .EQU        2D                      ;Block number too large
000060  BADOLDDATA      .EQU        27                      ;block has bad data from previous read
000061  SPTBLOVFLW      .EQU        27                      ;spare table overflow
000062
000063  ;---------------------------------------------------------------------
000064  ;
000065  ;       Hardware I/O Addresses
000066  ;
000067  ;---------------------------------------------------------------------
000068
000069  BANK_REG        .EQU        0FFEF
000070  E_REG           .EQU        0FFDF                   ;system environment register
000071  Z_REG           .EQU        0FFD0                   ;Zero page register for psuedo DMA
000072  PSUEDO_DMA      .EQU        0F800                   ;psuedo DMA code in ROM.
000073  WR_PORT         .EQU        0C080                   ;Write to Z8 RAM (byte at a time)
000074  RD_PORT         .EQU        0C081                   ;Read from Z8 RAM (byte at a time)
000075  BUSY            .EQU        0C082                   ;Z8 not ready
000076  CLR_PARITY      .EQU        0C083                   ;Clear parity error.
000077                  .PAGE
000078  ;---------------------------------------------------------------------
000079  ;
000080  ;       Constants
000081  ;
000082  ;---------------------------------------------------------------------
000083
000084  NOTCMD          .EQU        0                       ;Command line Low
000085  SETCMD          .EQU        4                       ;Command line High
000086  SETWRT          .EQU        1                       ;Low
000087  SETRD           .EQU        5                       ;High
000088  INTDSABL        .EQU        2
```

```
000089  INTENABL        .EQU        6
000090  RWLO            .EQU        3
000091  RWHI            .EQU        7
000092  RST             .EQU        0C
000093  CLRRST          .EQU        8
000094
000095  ;---------------------------------------------------------------------
000096  ;
000097  ;       Command codes
000098  ;
000099  ;---------------------------------------------------------------------
000100
000101  WDGTRD          .EQU        0
000102  WDGTWRTVER      .EQU        2
000103  WDGTSTAT        .EQU        3
000104  WDGTWRT         .EQU        1
000105
000106  ;---------------------------------------------------------------------
000107  ;
000108  ;       SIR allocation table
000109  ;
000110  ;---------------------------------------------------------------------
000111
000112  SIRADDR         .WORD       SIRTABLE
000113  SIRTABLE        .BYTE       10,                     0
000114                  .WORD       0
000115  SIRBANK         .BYTE       0
000116  SIRCOUNT        .EQU        *-SIRTABLE
000117                  .PAGE
000118  ;---------------------------------------------------------------------
000119  ;
000120  ;       Local variables
000121  ;
000122  ;---------------------------------------------------------------------
000123
000124  SLOTX           .BYTE       0
000125  ERROR           .BYTE       0                       ;if = 0 then no error
000126  IERROR          .BYTE       0                       ;if = 0 then no initialization error
000127
000128  PREVCMD         .BYTE       0
000129  PREVUNIT        .BYTE       0
000130  LENGTH          .BYTE       0                       ;number of blocks to read.
000131  ORGADR          .WORD       0
000132  ORGBNK          .BYTE       0
000133  SLOTCN          .BYTE       0
000134  ADDRDMA         .BYTE       0
000135  BANKDMA         .BYTE       0
000136  TEMP00          .BYTE       0
000137  TEMPFE          .BYTE       0
000138  MVCNT           .BYTE       0
000139  CNTLO           .BYTE       0
000140  CNTHI           .BYTE       0
000141  ADMODE          .BYTE       0
000142  DATDIR          .BYTE       0
000143  Z8CMD           .BYTE       0
000144  COUNTR          .BYTE       0
000145  MSBLOCK         .BYTE       0                       ;Most significant block # for Profile
000146  BSYLO           .BYTE       0                       ;FLAG set when driver has seen busy lo
000147  LONGWAIT        .BYTE       0
000148  WAITTIME        .BYTE       0
000149  PIPPIN_RESET    .BYTE       0
000150  STATUS1         .BYTE       0
000151  STATUS2         .BYTE       0
000152  STATUS3         .BYTE       0
000153  STATUS4         .BYTE       0
000154  PARITY_ERR      .BYTE       0
000155  RESET_FLAG      .BYTE       0
000156  BAD_RESPONSE    .BYTE       0
000157  CHEKBYTS        .WORD       0
000158                  .WORD       0
000159                  .WORD       0
000160  SVENV           .BYTE       0
000161  DMA_CNT         .BYTE       0
000162                  .PAGE
000163  ;---------------------------------------------------------------------
000164  ;
000165  ;       SOS Call Parameter Locations
000166  ;
000167  ;---------------------------------------------------------------------
000168
000169  SOS_REQCODE     .EQU        0C0
000170  SOS_UNIT        .EQU        0C1
000171
000172  SOS_BUF         .EQU        0C2                     ;D_READ/D_WRITE SOS calls
000173  SOS_BYTES       .EQU        0C4
000174  SOS_BLOCK       .EQU        0C6
000175  SOS_BYTRD       .EQU        0C8
000176
000177  SOS_STCODE      .EQU        0C2                     ;D_STATUS SOS call
000178  SOS_STLIST      .EQU        0C3
000179
000180  SOS_XPAGE       .EQU        1400
000181
```

```
000182  ;-------------------------------------------------------------------
000183  ;
000184  ;       More Zero page equates
000185  ;
000186  ;-------------------------------------------------------------------
000187
000188  ADRLO           .EQU        0CA                     ;Indirect address to user data
000189  ADRHI           .EQU        ADRLO+1
000190  SISADR          .EQU        SOS_XPAGE+ADRHI
000191  INDRCN          .EQU        0CC                     ;Indirect to $CN00 device locations
000192  SISCN           .EQU        SOS_XPAGE+INDRCN+1
000193  TIMOUT          .EQU        0CE
000194  RTRYCNT         .EQU        0CF
000195  RTRYTHRESH      .EQU        0D0
000196  CMD_RTRYCNT     .EQU        0D1
000197  RSPNS           .EQU        0D2
000198  BLK_RTRYCNT     .EQU        0D3
000199  BLOCKLO         .EQU        0D4
000200  BLOCKHI         .EQU        0D5
000201                  .PAGE
000202  ;-------------------------------------------------------------------
000203  ;
000204  ;  Psuedo DMA transfer routine.  Does bank switching first.
000205  ;
000206  ;-------------------------------------------------------------------
000207
000208  RELCODE         LDY         BANK_REG                ;This Code is moved to $18F0
000209                  STX         BANK_REG                ;  so the bank can be switched
000210                  JSR         PSUEDO_DMA              ;  for psuedo DMA transfers to
000211                  STY         BANK_REG                ;  other than driver bank.
000212                  RTS
000213  CDELEN          .EQU        *-RELCODE
000214                  .PAGE
000215  ;-------------------------------------------------------------------
000216  ;
000217  ;  Profile driver -- Main entry point
000218  ;
000219  ;-------------------------------------------------------------------
000220
000221  MAIN            .EQU        *
000222                  LDA         E_REG
000223                  AND         #0DF                    ;Save everything but
000224                  STA         SVENV                   ;  the current screen state
000225                  LDA         SOS_REQCODE
000226                  EOR         #8
000227                  BEQ         INIT_ENT
000228                  LDA         DIB_SLOT1
000229                  JSR         SELC800
000230                  BCC         $20
000231                  JMP         NODRV
000232  $20             LDA         IERROR                  ;did D_INIT encounter error?
000233                  BEQ         ONEMEG
000234                  JSR         SYSERR
000235  INIT_ENT        STA         SLOTCN
000236  ONEMEG          LDA         #0FF
000237                  STA         PIPPIN_RESET
000238                  STA         COUNTR
000239                  LDX         #CDELEN-1               ;Move code to page 18 for DMA
000240  MOVCODE         LDA         RELCODE,X               ;  to other possible banks.
000241                  STA         INDDMA,X
000242                  DEX
000243                  BPL         MOVCODE
000244                  LDA         #0
000245                  STA         PARITY_ERR
000246                  STA         BAD_RESPONSE
000247                  STA         RESET_FLAG
000248                  STA         INDRCN
000249                  STA         SISCN
000250                  LDA         SOS_BLOCK
000251                  STA         BLOCKLO
000252                  LDA         SOS_BLOCK+1
000253                  STA         BLOCKHI
000254                  LDX         SLOTX
000255                  PHP
000256                  SEI
000257                  LDA         E_REG
000258                  ORA         #83                     ;slow to 1mhz, ROM enable
000259                  STA         E_REG
000260                  PLP
000261                  LDA         SLOTCN
000262                  BPL         GO_INIT                 ;Branch if not initialized.
000263                  STA         INDRCN+1
000264                  STA         CLR_PARITY,X            ;Clear any previous parity errors.
000265                  LDY         BUSY,X
000266                  JSR         S2M                     ;set 2MHz mode
000267                  TYA
000268                  AND         #1
000269                  BNE         NODRV
000270                  TYA
000271                  BPL         $40
000272                  LDA         #0FF
000273                  STA         BSYLO
000274                  BNE         GO_INIT
```

```
000275  $40             LDA         BSYLO                   ;has BSY been low?
000276                  BNE         GO_INIT                 ;branch if so
000277                  JSR         RSTORENV
000278                  LDA         #XIOERROR
000279                  JSR         SYSERR                  ;otherwise error exit
000280
000281  GO_INIT         JSR         DISPATCH
000282                  JSR         SETUPWRITE
000283                  LDA         #0
000284                  JSR         SELC800
000285  RSTORENV        PHP
000286                  SEI
000287                  LDA         E_REG
000288                  AND         #20
000289                  ORA         SVENV                   ;restore environment register
000290                  STA         E_REG
000291                  PLP
000292                  RTS                                 ;  and exit to caller
000293
000294  S2M             PHP
000295                  SEI
000296                  LDA         E_REG
000297                  AND         #7F                     ;and out 1MHz bit
000298                  STA         E_REG
000299                  PLP
000300                  RTS
000301                  .PAGE
000302  DISPATCH        SWITCH      SOS_REQCODE,9,SW_TABLE
000303
000304  BADREQ          LDA         #XREQCODE               ;Invalid request code
000305                  JSR         SYSERR
000306
000307  BADOP           LDA         #XBADOP                 ;Invalid operation
000308                  JSR         SYSERR
000309
000310  NODRV           JSR         RSTORENV
000311                  LDA         #XNODRIVE               ;No drive connected (power off)
000312  SYS_ERROR       JSR         SYSERR
000313
000314  ;---------------------------------------------------------------------
000315
000316  SW_TABLE        .EQU        *                       ;Profile driver switch table
000317                  .WORD       DREAD-1                 ;  D_READ system call
000318                  .WORD       DWRITE-1                ;  D_WRITE      "
000319                  .WORD       DSTATUS-1               ;  D_STATUS     "
000320                  .WORD       FDCONTROL-1             ;  D_CONTROL    "
000321                  .WORD       BADREQ-1                ;  GET_DEV_NUM  "
000322                  .WORD       BADREQ-1                ;  D_INFO       "
000323                  .WORD       BADOP-1                 ;  D_OPEN       "
000324                  .WORD       BADOP-1                 ;  D_CLOSE      "
000325                  .WORD       DINIT-1                 ;  D_INIT       "
000326                  .WORD       DREPEAT-1               ;  D_REPEAT     "
000327                  .PAGE
000328  ;---------------------------------------------------------------------
000329  ;
000330  ;  Profile driver -- initialization request.
000331  ;
000332  ;---------------------------------------------------------------------
000333
000334  DINIT           .EQU        *
000335                  LDA         DIB_SLOT1               ;slot index:=dib.slot*16
000336                  TAX
000337                  BMI         DI_ERR1                 ;  error, invalid slot #
000338                  ORA         #0C0
000339                  STA         SLOTCN                  ;Save slot CN address
000340                  STA         INDRCN+1
000341                  ASL         A
000342                  ASL         A
000343                  ASL         A
000344                  ASL         A
000345                  STA         SLOTX
000346                  LDA         #0
000347                  STA         INDRCN
000348                  STA         SISCN
000349                  LDA         DIB_SLOT1               ;compute SIR #
000350                  CLC
000351                  ADC         SIRTABLE                ;  sir:=dib.slot1+16
000352                  STA         SIRTABLE
000353                  LDA         #SIRCOUNT               ;allocate the slot's SIR
000354                  LDX         SIRADDR
000355                  LDY         SIRADDR+1
000356                  JSR         ALLOCSIR
000357                  BCS         DI_ERR2                 ;SIR request failed
000358                  LDA         DIB_SLOT1
000359                  JSR         SELC800
000360                  BCS         DI_ERR1
000361                  JSR         SETUPREAD
000362                  LDY         #INTDSABL
000363                  LDA         (INDRCN),Y
000364                  JSR         SETUPWRITE
000365                  JSR         SETUPREAD
000366                  CLC
000367                  RTS                                 ;normal exit
```

```
000368
000369  DI_ERR1         LDA          #XNODRIVE                ;error, checksum fail/bad slot#
000370                  BNE          DI_ERR3
000371  DI_ERR2         LDA          #XNORESRC                ;error, SIR resource not available
000372  DI_ERR3         STA          IERROR
000373                  JSR          SYSERR
000374                  .PAGE
000375  ;---------------------------------------------------------------------
000376  ;
000377  ;  Profile driver -- status request.
000378  ;
000379  ;  Status request zero returns BUSY (bit 7), and ONLINE (bit 4).
000380  ;  Status request one returns device specific flags.
000381  ;  Lastly, the format status code $FE is also accepted.
000382  ;
000383  ;---------------------------------------------------------------------
000384
000385  DSTATUS         .EQU         *
000386                  LDY          #0
000387                  LDX          SOS_STCODE
000388                  BNE          $10
000389                  LDX          SLOTX
000390                  PHP
000391                  SEI
000392                  LDA          E_REG
000393                  ORA          #80                      ;or in 1MHz bit
000394                  STA          E_REG
000395                  PLP
000396                  LDA          BUSY,X
000397                  EOR          #80
000398                  ASL          A                        ;Save Busy status bit in carry
000399                  PHP                                   ;Save carry
000400                  ASL          A                        ;Now shift On-Line to Bit 5.
000401                  ASL          A
000402                  ASL          A
000403                  ASL          A
000404                  AND          #20                      ;Throw away other (garbage) bits
000405                  PLP                                   ;Get Busy bit again
000406                  ROR          A                        ;And shift it to Bit 7, and On-line to 4
000407                  STA          (SOS_STLIST),Y
000408                  RTS                                   ;(carry is always cleared from shift)
000409
000410  $10             DEX
000411                  BNE          FORMATUS                 ;branch not status code 1.
000412                  LDY          #3
000413  $15             LDA          STATUS1,Y
000414                  STA          (SOS_STLIST),Y           ;put status bytes in callers buf
000415                  DEY
000416                  BPL          $15
000417                  CLC
000418                  RTS
000419                  .PAGE
000420  FORMATUS        .EQU         *
000421                  LDA          SOS_STCODE               ;Test for format request.
000422                  CMP          #0FE
000423                  BNE          FCODE                    ;error, invalid status code
000424                  LDY          #0
000425                  LDA          #0FF
000426                  STA          (SOS_STLIST),Y
000427                  INY
000428                  STA          (SOS_STLIST),Y
000429                  CLC
000430                  RTS
000431
000432  FCODE           LDA          #XCTLCODE                ;invalid status/control code
000433                  JSR          SYSERR
000434                  .PAGE
000435  ;---------------------------------------------------------------------
000436  ;
000437  ;  Profile driver -- control request.
000438  ;
000439  ;  Control code must be $00 or $FE.  The reset request is ignored and
000440  ;  the Profile is preformatted, hence both requests are NOPs.
000441  ;
000442  ;---------------------------------------------------------------------
000443
000444  FDCONTROL       .EQU         *
000445                  LDA          SOS_STCODE               ;fetch control code
000446                  BEQ          $010
000447                  CMP          #0FE
000448                  BNE          FCODE                    ;error, invalid control code
000449  $010            CLC
000450                  RTS
000451
000452
000453  ;---------------------------------------------------------------------
000454  ;
000455  ;  Profile driver -- repeat request.
000456  ;
000457  ;  Retrieves the previous command and calls DREAD or DWRITE.
000458  ;
000459  ;---------------------------------------------------------------------
000460
```

```
000461  DREPEAT         .EQU        *
000462                  LDA         PREVUNIT
000463                  CMP         SOS_UNIT
000464                  BNE         $1                      ;Error exit if unit # changed
000465                  LDA         PREVCMD
000466                  STA         SOS_REQCODE
000467                  BNE         $2
000468                  JMP         DREAD
000469  $2              CMP         #2
000470                  BCS         $1                      ;Error, previous I/O not Read/Write
000471                  JMP         DWRITE
000472  $1              JMP         BADOP
000473                  .PAGE
000474  ;---------------------------------------------------------------------
000475  ;
000476  ;  This routine tests the validity of block number requested.
000477  ;
000478  ;---------------------------------------------------------------------
000479
000480  TSTBLKNUM       LDA         #0FF
000481                  STA         MSBLOCK                 ;assume status block
000482                  CMP         BLOCKHI
000483                  CLC
000484                  BNE         $10
000485                  LDA         BLOCKLO
000486                  CMP         #0FE                    ;RAM or status block?
000487                  BCC         $10                     ;branch if not
000488                  CLC
000489                  RTS                                 ;OK exit
000490  $10             INC         MSBLOCK
000491                  LDA         BLOCKLO                 ;Make sure Block number is
000492                  CMP         DIB_BLOCK1              ;  within widget's range.
000493                  LDA         BLOCKHI
000494                  SBC         DIB_BLOCK1+1            ;If carry clear then within range.
000495  $20             RTS
000496                  .PAGE
000497  ;---------------------------------------------------------------------
000498  ;
000499  ;  Getbytes subroutine.  Moves SOS bytes parameter to length variable.
000500  ;  If bytes are not a multiple of 512, or a block # > maxblock will be
000501  ;  written or read, then XBYTECNT error exit will be done.  If the first
000502  ;  block to be read > MAXBLOCK-1, an XBLKNUM error exit will be done.
000503  ;
000504  ;---------------------------------------------------------------------
000505
000506  GETBYTES        .EQU        *
000507                  JSR         TSTBLKNUM               ;check validity of 1st block to be
000508  ;                                       written or read
000509                  BCC         $01                     ;branch if OK
000510                  PLA
000511                  PLA
000512                  LDA         #XBLKNUM
000513                  JSR         SYSERR                  ;bad block number error
000514  $01             LDA         SOS_BYTES
000515                  BNE         $10
000516                  LDA         SOS_BYTES+1
000517                  LSR         A
000518                  BCS         $10
000519                  STA         LENGTH
000520                  BNE         $05
000521                  PLA
000522                  PLA
000523                  RTS
000524  $05             LDX         BLOCKHI
000525                  LDY         BLOCKLO
000526                  SEC
000527                  SBC         #1
000528                  CLC
000529                  ADC         BLOCKLO
000530                  STA         BLOCKLO
000531                  LDA         BLOCKHI
000532                  ADC         #0
000533                  BCS         $10
000534                  STA         BLOCKHI                 ;compute last block to be wrtn or rd
000535                  JSR         TSTBLKNUM               ;is it valid?
000536                  STX         BLOCKHI
000537                  STY         BLOCKLO
000538                  BCC         CALCHEK                 ;branch if GOOD!
000539
000540  $10             PLA
000541                  PLA
000542                  LDA         #XBYTECNT               ;byte count not multiple of 512
000543                  JSR         SYSERR
000544
000545
000546  CALCHEK         TYA
000547                  STA         CHEKBYTS+2
000548                  EOR         #0FF
000549                  STA         CHEKBYTS+5
000550                  TXA
000551                  STA         CHEKBYTS+1
000552                  EOR         #0FF
000553                  STA         CHEKBYTS+4
```

```
000554                  LDA         #0
000555                  STA         CHEKBYTS
000556                  LDA         #0FF
000557                  STA         CHEKBYTS+3
000558                  RTS
000559                  .PAGE
000560  ;---------------------------------------------------------------------
000561  ;
000562  ;  Profile driver -- read request.
000563  ;
000564  ;  Transfers "bytes/512" blocks from disk to buffer..buffer+bytes-1.
000565  ;
000566  ;---------------------------------------------------------------------
000567
000568  DREAD           .EQU        *
000569                  LDA         #0
000570                  TAY
000571                  STA         (SOS_BYTRD),Y
000572                  INY
000573                  STA         (SOS_BYTRD),Y
000574                  LDA         SOS_UNIT
000575                  STA         PREVUNIT                ;save current sos unit
000576                  LDA         SOS_REQCODE             ;save current command
000577                  STA         PREVCMD
000578                  JSR         GETBYTES                ;move bytes parm to length var.
000579                  JSR         ARBADR                  ;Get beginning address resolved.
000580  RDBLOCK         LDA         #2                      ;Allow 2 retries
000581                  STA         BLK_RTRYCNT
000582  RPAR_RETRY      .EQU        *                       ;Parity error retries enter here
000583                  LDA         #0                      ;Indicate data direction is
000584                  STA         DATDIR                  ;read.
000585                  STA         CMD_RTRYCNT             ;clear # communication tries
000586                  LDA         #0A                     ;set retry count to 10
000587                  STA         RTRYCNT
000588                  LDA         #3
000589                  STA         RTRYTHRESH              ;set reseek/rewrite threshold
000590                  JSR         TSTBLKNUM               ;make sure it's a valid block
000591                  BCC         RCOM_RETRY
000592                  LDA         #XBLKNUM
000593                  JSR         SYSERR                  ;invalid block error exit
000594  RCOM_RETRY      .EQU        *                       ;CMD-BSY bad response retries enter here
000595                  LDA         #0
000596                  STA         LONGWAIT
000597                  LDA         #WDGTRD
000598                  STA         Z8CMD
000599                  LDA         #1
000600                  STA         RSPNS                   ;set up expected Z8 response
000601                  JSR         SNDCMD                  ;do a CMD-BSY handshake
000602                  BCC         $70
000603                  JMP         DR_ERR1
000604  $70             BNE         RCOM_RETRY              ;bad response - try again.
000605
000606  ;               Send command bytes
000607
000608                  JSR         SND_CMDBYTES
000609                  BCS         RDRETRY                 ;try again if parity error
000610                  LDA         #2
000611                  STA         RSPNS                   ;set up for execution
000612                  LDA         #0FF
000613                  STA         LONGWAIT                ;adjust timeout wait
000614                  JSR         SNDCMD                  ;2nd CMD-BSY for read opn
000615                  BCS         DR_ERR1                 ;timeout on cmd-bsy
000616                  BNE         RCOM_RETRY              ;bad response - try again.
000617
000618  ;               read should be complete at this point
000619
000620                  JSR         GETSTAT                 ;get status bytes first
000621                  BCS         RDRETRY                 ;try again if parity error
000622                  BPL         $10
000623                  INC         COUNTR                  ;time to reset PIPPIN?
000624                  BEQ         RCOM_RETRY              ;branch if not
000625                  BNE         DR_ERR1                 ;reset PIPPIN
000626  $10             JSR         DATRANS
000627                  BCS         RDRETRY
000628                  LDA         STATUS1                 ;Now check status for good read
000629                  AND         #1
000630                  BNE         LDAXIO
000631                  LDA         STATUS2
000632                  AND         #58
000633                  BNE         LDAXIO
000634                  LDA         #2
000635                  LDY         #1
000636                  ADC         (SOS_BYTRD),Y
000637                  STA         (SOS_BYTRD),Y
000638                  JSR         TSTMORE                 ;more blocks to read?
000639                  BEQ         GOODEXIT
000640                  JMP         RDBLOCK                 ;jump if more
000641
000642  RDRETRY         JSR         RSTORADR                ;restore 'ORGADR' and try again
000643                  DEC         BLK_RTRYCNT             ;can we retry?
000644                  BMI         DR_ERR1
000645                  JMP         RPAR_RETRY
000646
```

```
000647  DR_ERR1         INC          PIPPIN_RESET
000648                  BNE          $10
000649                  JSR          RESET_PIPPIN
000650                  LDA          #0
000651                  STA          CMD_RTRYCNT
000652                  JMP          RCOM_RETRY
000653  $10             JSR          NOTCMDLN
000654  LDAXIO          LDA          #XIOERROR
000655                  BNE          BADEXIT
000656
000657  TSTMORE         INC          BLOCKLO                  ;Bump the block number
000658                  BNE          $010
000659                  INC          BLOCKHI
000660  $010            LDY          BLOCKLO
000661                  LDX          BLOCKHI
000662                  JSR          CALCHEK
000663                  LDA          ADRHI
000664                  STA          ORGADR+1
000665                  LDA          SISADR
000666                  STA          ORGBNK
000667                  DEC          LENGTH                   ;Are there more blocks to read?
000668                  RTS                                   ;Return Z flag set if done.
000669
000670  GOODEXIT        LDA          #01
000671                  STA          RSPNS                    ;DUMMY HANDSHAKE FOR PROFILE TO UPDATE
000672                  JSR          SNDCMD                   ; ANY PENDING INFO TO DISK.
000673                  LDA          #0FF
000674                  STA          Z8CMD                    ;we're going to send an invalid command
000675                  JSR          SND_CMDBYTES
000676                  JSR          SETCMDLN                 ;raise the CMD line
000677                  JSR          SETCMDLN                 ;give Profile a little time
000678                  JSR          NTCMDLN1                 ;lower the CMD line (so the ready light
000679  ;                                        doesn't go out)
000680                  RTS
000681
000682  BADEXIT         PHA
000683                  JSR          GOODEXIT
000684                  PLA
000685                  JSR          SYSERR
000686
000687  RSTORADR        LDA          ORGADR                   ;Reset addresses
000688                  STA          SOS_BUF
000689                  LDA          ORGADR+1
000690                  STA          SOS_BUF+1
000691                  LDA          ORGBNK
000692                  STA          SOS_XPAGE+SOS_BUF+1
000693                  JMP          ARBADR

; ##############################################################################################################
; #   END OF FILE:  PROFILE.A.TEXT
; #   LINES      :  693
; #   CHARACTERS :  33886
; #   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 Profile Driver 1.30 (6502 Assembly Source Code)
; #   FILE NAME:  PROFILE.B.TEXT
; ###############################################################################################

000001  ;---------------------------------------------------------------------
000002  ;
000003  ;  Profile driver -- write request.
000004  ;
000005  ;  Transfers "bytes/512" blocks from buffer to block..block+(bytes/512).
000006  ;  Error status on return from subroutines is the same as for DREAD.
000007  ;
000008  ;---------------------------------------------------------------------
000009
000010  DWRITE          .EQU        *
000011                  LDA         SOS_REQCODE             ;save current command
000012                  STA         PREVCMD
000013                  LDA         SOS_UNIT
000014                  STA         PREVUNIT
000015                  JSR         GETBYTES                ;move bytes parm to length var.
000016                  JSR         ARBADR                  ;Get beginning address resolved.
000017                  LDA         #0FF                    ;Indicate data direction is Write
000018                  STA         DATDIR
000019
000020  WRBLOCK         LDA         #2
000021                  STA         BLK_RTRYCNT
000022
000023  WPAR_RETRY      LDA         #0                      ;parity error retries enter here
000024                  STA         CMD_RTRYCNT             ;init cmd-bsy retry variable
000025                  STA         RTRYCNT                 ;and read retry and threshold
000026                  STA         RTRYTHRESH              ;variables to 0 for write opn
000027                  JSR         TSTBLKNUM               ;check for valid block#
000028                  BCC         WCOM_RETRY
000029                  JMP         DW_ERR2
000030
000031  ;CMD-BSY retries enter here
000032
000033  WCOM_RETRY      LDA         #WDGTWRT
000034                  LDY         WRTVER
000035                  BEQ         $10
000036                  LDA         #WDGTWRTVER
000037  $10             STA         Z8CMD
000038                  LDA         #1
000039                  STA         RSPNS                   ;set up expected response
000040                  JSR         SNDCMD                  ;1st cmd-bsy handshake
000041                  BCS         DW_ERR1                 ;cmd-bsy timeout error
000042                  BNE         WCOM_RETRY              ;wrong response - try again
000043                  JSR         SND_CMDBYTES            ;send write command string
000044                  BCS         $12                     ;retry if parity error
000045
000046  ;Now set up to send write data to widget
000047
000048                  LDA         Z8CMD
000049                  ADC         #2
000050                  STA         RSPNS                   ;set up expected response
000051                  JSR         SNDCMD                  ;2nd cmd-bsy handshake
000052                  BCS         DW_ERR1                 ;timeout on cmd-bsy
000053                  BNE         WCOM_RETRY              ;retry the handshake if taken
000054                  JSR         DATRANS                 ;now transfer data to widget
000055                  BCC         $15
000056  $12             JMP         WRRETRY                 ;retry on parity error
000057
000058  ;now get status from write - 3rd cmd-bsy handshake
000059
000060  $15             LDA         #6
000061                  STA         RSPNS                   ;set up expected response
000062                  LDA         WRTVER
000063                  BEQ         $20                     ;branch if not write/verify
000064                  LDA         #0FF
000065                  STA         LONGWAIT                ;adjust timeout value
000066  $20             JSR         SNDCMD
000067                  BCS         DW_ERR1                 ;timeout error - cmd-bsy
000068                  BEQ         WCONT                   ;continue if good return status
000069                  JSR         RSTORADR                ;restore 'ORGADR' since already
000070  ;wrote bytes to z8
000071                  JMP         WCOM_RETRY              ;and retry communication
000072
000073  DW_ERR1         INC         PIPPIN_RESET
000074                  BNE         $10
000075                  JSR         RESET_PIPPIN
000076                  LDA         #0
000077                  STA         CMD_RTRYCNT
000078                  JMP         WCOM_RETRY
000079  $10             JSR         NOTCMDLN
000080  LDAXIOERR       LDA         #XIOERROR
000081  CSYSER1         JMP         BADEXIT
000082
000083  DW_ERR2         LDA         #XBLKNUM
000084                  BNE         CSYSER1
000085
000086
000087
000088  WCONT           JSR         GETSTAT
```

```
000089                  BCS         WRRETRY                 ;parity error - try again
000090                  BPL         $10
000091                  INC         COUNTR                  ;time to reset PIPPIN?
000092                  BEQ         WRRETRY                 ;branch if not
000093                  BNE         DW_ERR1
000094  $10             AND         #41
000095                  BNE         LDAXIOERR
000096                  LDA         STATUS2
000097                  AND         #48
000098                  BNE         LDAXIOERR               ;if pippin couldnt read its status
000099                  JSR         TSTMORE                 ;more to write?
000100                  BEQ         DW_EXIT
000101                  JMP         WRBLOCK
000102  DW_EXIT         JMP         GOODEXIT
000103
000104  WRRETRY         JSR         RSTORADR                ;restore 'ORGADR' and try again
000105                  DEC         BLK_RTRYCNT             ;can we retry?
000106                  BMI         DW_ERR1
000107                  JMP         WPAR_RETRY
000108                  .PAGE
000109  ;---------------------------------------------------------------------
000110  ;
000111  ;  Profile Block I/O transfer routine.  This routine will transfer 512
000112  ;  bytes to/from users buffer from/to RAM buffer of Profile's Z8.  It
000113  ;  uses both byte at a time and psuedo DMA as necessary for the fastest
000114  ;  possible transfer rate.  If users buffer is on a page boundary, only
000115  ;  the psuedo DMA is used.
000116  ;
000117  ;  Because of the requirements of the psuedo DMA, a small routine is
000118  ;  relocated to page $18 to swap banks before transfer. This is done
000119  ;  only once per call to the driver.
000120  ;
000121  ;  NOTE: this routine is designed to transfer no less than 512 bytes.
000122  ;---------------------------------------------------------------------
000123
000124  DATRANS         PHP
000125                  SEI
000126                  LDA         #2                      ;Always move 512 bytes at a time
000127                  STA         CNTHI
000128                  LDA         #0
000129                  STA         CNTLO
000130                  BIT         DATDIR                  ;Write or read?
000131                  BMI         $010                    ;Branch if write
000132                  JSR         SETUPREAD
000133                  JMP         $020                    ;Branch always taken.
000134
000135  $010            JSR         SETUPWRITE
000136  $020            LDA         E_REG
000137                  AND         #7F                     ;and out 1MHz bit
000138                  STA         E_REG
000139                  PLP
000140                  LDA         ADRLO                   ;Is transfer on page boundary?
000141                  BEQ         DATADMA                 ;Yes, Do it fast!!
000142                  CMP         #0F5                    ;Should we bother with DMA?
000143                  BCC         $030                    ;Branch if DMA will be faster.
000144                  EOR         #0FF                    ;Set up for move count
000145                  JSR         MOVIT
000146                  JMP         FASTMOV                 ;Done with first (partial) page.
000147  $030            AND         #1                      ;Is it a 2 byte boundary?
000148                  BEQ         $040                    ;Yes, move first partial page.
000149                  LDA         #0                      ;Otherwise move a byte
000150                  STA         MVCNT
000151                  JSR         MOVE                    ; to get things aligned.
000152  $040            LDA         ADRLO
000153  DATADMA         TAX
000154                  CLC                                 ;Round up to next 2 byte boundary
000155                  ADC         #1
000156                  AND         #0FE
000157                  STA         VECTLO                  ;Store as low ROM entry point.
000158                  TXA
000159                  EOR         #0FF
000160                  STA         MVCNT                   ;Save number of bytes (-1) moved.
000161                  STA         DMA_CNT                 ;(Guaranteed to be enough for DMA to
000162                  JSR         GO_DMA                  ; the end of a page)
000163                  JSR         ADJ_ADR                 ;Update address and count
000164                  LDA         CNTHI
000165                  BEQ         LAST_PGE                ;Branch if less than 1 page to move.
000166  FASTMOV         LDA         #0
000167                  BEQ         DATADMA                 ;Branch ALWAYS to move next page.
000168                  .PAGE
000169  TRANSDNE        LDY         #0
000170                  LDX         SLOTX
000171                  PHP
000172                  SEI
000173                  LDA         E_REG
000174                  ORA         #80                     ;or in 1MHz bit
000175                  STA         E_REG
000176                  PLP
000177                  BIT         DATDIR
000178                  BPL         $300
000179  $100            LDA         CHEKBYTS,Y
000180                  STA         WR_PORT,X
000181                  INY
```

```
000182                  CPY         #6
000183                  BNE         $100
000184  $300            JSR         CHKPARITY               ;Test for parity error in transfer.
000185  $999            PHP
000186                  SEI
000187                  JSR         SETUPREAD               ;restore read state
000188                  JSR         S2M                     ;back to 2 MHz
000189                  PLP
000190                  RTS
000191
000192  LAST_PGE        LDA         CNTLO                   ;Anything left to move?
000193                  BEQ         TRANSDNE                ;Branch if not.
000194                  SEC
000195                  SBC         #1                      ;Note: low buffer address is always 0.
000196                  CMP         #20                     ;Is there more than 32 bytes left?
000197                  BCS         $20                     ;If not, move last a byte at a time.
000198                  JSR         MOVIT
000199                  JMP         TRANSDNE
000200
000201  $20             PHA
000202                  LDA         #1                      ;Move first 2 bytes to preserve byte 0.
000203                  STA         MVCNT
000204                  JSR         MOVE                    ;The call to ADJ_ADR is below.
000205                  LDY         #0                      ;Bytes 0 and FE of the current page
000206                  PHP
000207                  SEI                                 ;disallow interupts
000208                  LDA         (ADRLO),Y               ;  must be preserved because of the
000209                  STA         TEMP00                  ;  quirks of the psuedo DMA while doing
000210                  LDY         #0FE                    ;  a partial page transfers.  Since the
000211                  LDA         (ADRLO),Y               ;  branch instuction generates a false
000212                  STA         TEMPFE                  ;  address within the DMA page, byte 0
000213                  PLA                                 ;  is accessed if more than $80 bytes
000214                  TAX                                 ; (save status)
000215                  PLA
000216                  AND         #0FC                    ;  are transfered and byte FE is ac-
000217                  CMP         #84                     ;  cessed if less than $80.
000218                  BNE         $30
000219                  SBC         #04                     ;Transfers of exactly $80 (82) are not
000220  $30             TAY                                 ;  allowed do to DMA code (see Apple 3
000221                  LSR         A                       ;  monitor listing of psuedo DMA code)
000222                  LSR         A
000223                  SBC         #1
000224                  STA         DMA_CNT                 ;Set up for exit of DMA
000225                  TXA
000226                  PHA                                 ;restore status to stack
000227                  TYA
000228                  SEC
000229                  SBC         #3
000230                  STA         MVCNT                   ;Save bytes total bytes transferred.
000231                  TAY
000232                  LDA         #02                     ;Set low entry point for DMA routine.
000233                  STA         VECTLO
000234                  BIT         DATDIR                  ;Read or write?
000235                  BPL         $50                     ;Branch if read.
000236                  LDA         (ADRLO),Y               ;Get last byte and move to both
000237                  LDY         #0                      ;  byte 0 and FE.
000238                  STA         (ADRLO),Y
000239                  LDY         #0FE
000240                  STA         (ADRLO),Y
000241  $50             JSR         GO_DMA                  ;Now transfer the partial page.
000242                  BIT         DATDIR                  ;Was it read or write?
000243                  BMI         $70                     ;Branch if write.
000244                  LDY         #0
000245                  BIT         MVCNT                   ;Now figer out where the last byte
000246                  BMI         $60                     ;  got red.
000247                  LDY         #0FE                    ;Must have been Read into FE.
000248  $60             LDA         (ADRLO),Y
000249                  LDY         MVCNT                   ;Put it where it belongs!
000250                  STA         (ADRLO),Y
000251  $70             LDY         #0                      ;Now restore bytes 0 & FE.
000252                  LDA         TEMP00
000253                  STA         (ADRLO),Y
000254                  LDY         #0FE
000255                  LDA         TEMPFE
000256                  STA         (ADRLO),Y
000257                  PLP                                 ;Interupts OK now.
000258                  JSR         ADJ_ADR                 ;Go fix addresses and count.
000259                  JMP         LAST_PGE
000260
000261  MOVIT           STA         MVCNT                   ;Number of bytes for transfer.
000262                  JSR         MOVE                    ;Do byte at a time.
000263                  .PAGE
000264  ;---------------------------------------------------------------------
000265  ;
000266  ;  This routine adjusts the count and addresses (indirect and bank
000267  ;  direct) after each transfer, either byte at a time or psuedo DMA.
000268  ;
000269  ;  Input is MVCNT.  Only the processor status and accumulator are used.
000270  ;  ADRLO, ADRHI, ADDRDMA, BANKDMA are all assumed to be valid, thus a
000271  ;  call to ARBADR should have taken place before this routine is used.
000272  ;  CNTLO, CNTHI should never be less than MVCNT, as no checking is done.
000273  ;
000274  ;---------------------------------------------------------------------
```

```
000275
000276  ADJ_ADR          CLC
000277                   LDA          CNTLO                    ;Adjust count and Addresses
000278                   SBC          MVCNT                    ;Subtract (MVCNT+1) from count
000279                   STA          CNTLO                    ;  and add (MVCNT+1) to address.
000280                   LDA          CNTHI
000281                   SBC          #0
000282                   STA          CNTHI
000283                   LDA          ADRLO
000284                   ADC          MVCNT                    ;(carry was set)
000285                   STA          ADRLO
000286                   LDA          ADRHI
000287                   ADC          #0
000288                   STA          ADRHI
000289                   BIT          ADMODE                   ;Are Bank Wraps possible?
000290                   BPL          ADJ_DNE                  ;Branch if not.
000291                   CMP          #081                     ;Time to Adjust for Wrap?
000292                   BCC          ADJ_BNK                  ;Branch if not.
000293                   AND          #07F                     ;Otherwise strip hi bit, add 1 to
000294                   INC          SISADR                   ;  bank pair selected indirect
000295  ADJ_BNK          STA          ADRHI
000296                   LDA          SISADR
000297                   STA          BANKDMA
000298                   LDA          ADRHI
000299                   CLC
000300                   ADC          #20                      ;Add $20 for bank address equiv.
000301                   CMP          #0A0                     ;Next bank?
000302                   BCC          ADJ_DNE                  ;Branch if not.
000303                   AND          #07F                     ;Address range must be within 20-9F
000304                   INC          BANKDMA
000305  ADJ_DNE          STA          ADDRDMA                  ;Save absolute address for DMA
000306                   RTS
000307                   .PAGE
000308  ;---------------------------------------------------------------------
000309  ;
000310  ;  The move routines read or write data byte at a time.  Input varibles
000311  ;  are ADRLO, ADRHI and MVCNT, none of which are modified (that is done
000312  ;  by ADJADR).  All registers are used.  Also SLOTX must contain the
000313  ;  slot number times 16 for indexing the device locations for read and
000314  ;  write.  DATDIR indicates the direction (Read=0, Write=FF) of transfer.
000315  ;
000316  ;---------------------------------------------------------------------
000317
000318  MOVE             PHP
000319                   SEI
000320                   LDX          SLOTX                    ;Get index to device locations.
000321                   LDY          #0
000322                   LDA          E_REG
000323                   ORA          #80                      ;or in 1 MHz bit
000324                   STA          E_REG
000325                   BIT          DATDIR                   ;Read or write?
000326                   BPL          MOVIN                    ;Branch if read.
000327
000328  MOVOUT           LDA          (ADRLO),Y                ;Move data to Z8 RAM
000329                   STA          WR_PORT,X
000330                   CPY          MVCNT                    ;Done with Write?
000331                   BEQ          MVDONE                   ;Branch if done.
000332                   INY
000333                   BNE          MOVOUT                   ;Branch always taken.
000334
000335  MOVIN            LDA          RD_PORT,X                ;Get data from Z8 RAM
000336                   STA          (ADRLO),Y
000337                   CPY          MVCNT
000338                   BEQ          MVDONE
000339                   INY
000340                   BNE          MOVIN                    ;Branch always.
000341
000342  MVDONE           JSR          S2M                      ;back to 2 MHz
000343                   PLP
000344                   RTS
000345                   .PAGE
000346  ;---------------------------------------------------------------------
000347  ;
000348  ;  This routine arbitrates the initial user's address in terms of both
000349  ;  indirect and absolute bank address.  It sets the initial states for
000350  ;  ADRLO, ADRHI, ADDRDMA, BANKDMA, and ADMODE.  ADMODE's bit 7 is set
000351  ;  if extented addressing is used, otherwise it is reset (0).  ADDRDMA
000352  ;  is the page address resulting within the 6502 address space,  the
000353  ;  low address is always the same as ADRLO.
000354  ;
000355  ;---------------------------------------------------------------------
000356
000357  ARBADR           LDA          SOS_BUF                  ;Get strait indirect address moved
000358                   STA          ADRLO
000359                   STA          ORGADR
000360                   LDA          SOS_BUF+1
000361                   STA          ADRHI
000362                   STA          ORGADR+1
000363                   STA          ADDRDMA
000364                   LDA          SOS_XPAGE+SOS_BUF+1
000365                   STA          SISADR
000366                   STA          ORGBNK
000367                   STA          ADMODE
```

```
000368                  BPL         NOBANK                  ;Branch if load into current bank.
000369                  AND         #0F
000370                  STA         BANKDMA
000371                  EOR         #0F                     ;Now test for special bank (F)
000372                  BEQ         BANK0
000373                  LDA         SOS_BUF+1
000374                  BPL         $010
000375                  AND         #7F                     ;Subtract $80 from high byte
000376                  INC         BANKDMA                 ;  and increment bank number
000377  $010            CLC
000378                  ADC         #20                     ;Convert to absolute address
000379                  STA         ADDRDMA
000380                  RTS
000381
000382  NOBANK          LDA         BANK_REG                ;Use current bank as bank select
000383                  AND         #0F                     ;  for DMA purposes.
000384  BANK0           STA         BANKDMA
000385                  STA         ADMODE                  ;Clear bit 7 of mode to indicate no
000386                  RTS                                 ;  bank wrapping required.
000387                  .PAGE
000388  ;---------------------------------------------------------------------
000389  ;
000390  ;  In order to perform psuedo DMA, the call to ROM must be outside the
000391  ;  Bank.  This routine sets up the Z_REG and loads the X register with
000392  ;  the bank desired before calling DO_DMA which has been relocated to
000393  ;  page 2.  It also disables interupts for the duration (up to 256 usec)
000394  ;  of the transfer.
000395  ;
000396  ;---------------------------------------------------------------------
000397
000398  GO_DMA          PHP                                 ;Save interupt status
000399                  SEI                                 ;No interupts for now.
000400                  LDA         E_REG                   ;And no NMI's either!!!
000401                  PHA
000402                  AND         #24                     ;Switch out I/O also.
000403                  ORA         #8B                     ;And write protect upper 16K!
000404                  STA         E_REG
000405                  LDA         ADDRDMA                 ;Set Z_REG to DMA page address
000406                  STA         Z_REG
000407                  LDX         BANKDMA
000408                  LDA         DMA_CNT
000409                  SEC
000410                  JSR         DO_DMA
000411                  PLA                                 ;Restore NMI state.
000412                  STA         E_REG
000413                  LDA         #SOS_ZPAGE              ;Restore proper zero page.
000414                  STA         Z_REG
000415                  PLP                                 ;Restore interupts
000416                  RTS
000417                  .PAGE
000418  ;---------------------------------------------------------------------
000419  ;
000420  ;  The following are routines for handling the communications protocol
000421  ;  of sending commands, and receiving result codes.
000422  ;
000423  ;---------------------------------------------------------------------
000424  ;
000425  ;  SNDCMD performs a CMD-BSY handshake with the Z8 and checks for a
000426  ;  correct response.  If the Z8 responds with an incorrect code, a
000427  ;  'no go' code is sent by the Apple and the handshake is retried
000428  ;  up to 2 times.  On return, Carry=1 means a handshake timeout or
000429  ;  three retries attempted.  A non-zero return means an incorrect
000430  ;  response from the Z8 that may be retried.
000431  ;
000432  ;---------------------------------------------------------------------
000433
000434  SNDCMD          PHP
000435                  SEI
000436                  LDA         E_REG
000437                  ORA         #80                     ;or in 1 MHz bit
000438                  STA         E_REG
000439                  PLP
000440                  JSR         WAITBSYLO
000441                  BCS         SENDERR                 ;error exit if BSY isn't low
000442                  JSR         SETCMDLN                ;raise cmd
000443                  JSR         WAITBSYHI               ;wait for bsy to go hi
000444                  BCS         SENDERR                 ;timeout
000445                  LDY         RD_PORT,X               ;read response byte from Z8
000446                  CPY         RSPNS                   ;correct?
000447                  BEQ         CONT                    ;yes if taken
000448                  LDA         #2
000449                  STA         BAD_RESPONSE
000450                  ORA         STATUS3
000451                  STA         STATUS3
000452                  LDA         #0AA                    ;tell Z8 that response not OK
000453                  JSR         BSYACK                  ;drop cmd, wait for bsy to go lo
000454                  BCS         SENDERR                 ;timeout on bsy going lo
000455                  INC         CMD_RTRYCNT             ;bump retry count for bad response
000456                  LDY         CMD_RTRYCNT
000457                  CPY         #2                      ;2 retries yet?
000458                  BCS         SENDERR                 ;yes if taken
000459                  RTS
000460
```

```
000461  CONT            LDA         #055                    ;indicate good response
000462                  JSR         BSYACK
000463                  BCS         SENDERR                 ;bsy timeout
000464                  LDA         #0                      ;indicate good return to caller
000465  SENDERR         RTS
000466
000467  WAITBSYHI       LDY         #0                      ;set .5sec timeout
000468                  STY         TIMOUT
000469                  CLC
000470                  LDX         SLOTX
000471  ALOOP           LDA         BUSY,X
000472                  BPL         BSYHIRET                ;done if taken
000473                  DEY
000474                  BNE         ALOOP
000475                  DEC         TIMOUT
000476                  BNE         ALOOP
000477                  SEC                                 ;timeout
000478  BSYHIRET        RTS
000479
000480  WAITBSYLO       LDY         #1
000481                  LDA         LONGWAIT
000482                  BEQ         $10
000483                  LDY         #10                     ;set up for 8 second wait max.
000484  $10             STY         WAITTIME
000485                  LDY         #0                      ;set .5sec timeout
000486                  STY         TIMOUT
000487                  CLC
000488                  LDX         SLOTX
000489  BLOOP           LDA         BUSY,X
000490                  BMI         BSYLORET                ;done if taken
000491                  DEY
000492                  BNE         BLOOP
000493                  DEC         TIMOUT
000494                  BNE         BLOOP
000495                  DEC         WAITTIME
000496                  BNE         BLOOP
000497                  SEC                                 ;timeout
000498  BSYLORET        RTS
000499
000500  ;---------------------------------------------------------------------
000501  ;
000502  ;  SND_CMDBYTES sends the command string to widget.
000503  ;  Enter with cmd=bsy=lo.  Error return if get parity error - Carry = 1
000504  ;
000505  ;---------------------------------------------------------------------
000506
000507  SND_CMDBYTES    JSR         SETUPWRITE              ;get in proper state
000508                  LDA         Z8CMD                   ;send command string - cmd, blockhi
000509                  STA         WR_PORT,X               ;blocklo, retries, retry threshold
000510                  LDA         MSBLOCK
000511                  STA         WR_PORT,X
000512                  LDA         BLOCKHI
000513                  STA         WR_PORT,X
000514                  LDA         BLOCKLO
000515                  STA         WR_PORT,X
000516                  LDA         RTRYCNT
000517                  STA         WR_PORT,X
000518                  LDA         RTRYTHRESH
000519                  STA         WR_PORT,X
000520                  JSR         SETUPREAD               ;finish writing last byte and
000521                  JSR         CHKPARITY               ;check for parity error
000522                  RTS
000523                  .PAGE
000524  ;---------------------------------------------------------------------
000525  ;
000526  ;GETSTAT retrieves the status bytes from widget.  The one-byte
000527  ;result code is returned in Y.
000528  ;
000529  ;---------------------------------------------------------------------
000530
000531  GETSTAT         LDX         SLOTX                   ;get slot #
000532                  PHP
000533                  SEI
000534                  LDA         E_REG
000535                  ORA         #80                     ;or in 1 MHz bit
000536                  STA         E_REG
000537                  PLP
000538                  LDA         RD_PORT,X
000539                  STA         STATUS1
000540                  LDA         RD_PORT,X
000541                  STA         STATUS2
000542                  LDA         RD_PORT,X
000543                  PHA
000544                  LDA         RD_PORT,X
000545                  STA         STATUS4
000546                  JSR         CHKPARITY
000547                  PLA
000548                  ORA         PARITY_ERR
000549                  ORA         BAD_RESPONSE
000550                  ORA         RESET_FLAG
000551                  STA         STATUS3
000552                  LDA         STATUS1
000553                  RTS
```

```
000554
000555  ;-------------------------------------------------------------------
000556  ;
000557  ;  CHKPARITY checks the parity error line and shifts it into Carry, so
000558  ;  Carry = 1 is a parity error on return to caller.
000559  ;
000560  ;-------------------------------------------------------------------
000561
000562  CHKPARITY       LDX          SLOTX
000563                  LDA          BUSY,X                  ;get parity error - on bit 6
000564                  STA          CLR_PARITY,X            ;clear it for next transfer
000565                  ASL          A
000566                  ASL          A                       ;shift it into carry
000567                  BCC          $10
000568                  LDA          #1
000569                  STA          PARITY_ERR
000570                  ORA          STATUS3
000571                  STA          STATUS3
000572  $10             JMP          S2M                     ;exit via setting 2 MHz mode
000573                  .PAGE
000574  ;-------------------------------------------------------------------
000575  ;
000576  ;  SETUPWRITE sets CRW and DATRW lo on the Apple /// interface board
000577  ;  to prepare for a write operation to widget
000578  ;
000579  ;-------------------------------------------------------------------
000580
000581  SETUPWRITE      LDY          #SETWRT
000582                  PHP
000583                  SEI
000584                  LDA          E_REG
000585                  ORA          #80                     ;or in 1 MHz bit
000586                  STA          E_REG
000587                  PLP
000588                  LDA          (INDRCN),Y              ;set crw lo
000589  SET_WRITEDIR    LDY          #RWLO
000590                  LDA          (INDRCN),Y              ;set datarw lo
000591                  RTS
000592
000593
000594  SETUPREAD       LDY          #SETRD
000595                  PHP
000596                  SEI
000597                  LDA          E_REG
000598                  ORA          #80                     ;or in 1MHz bit
000599                  STA          E_REG
000600                  PLP
000601                  LDA          (INDRCN),Y              ;set crw hi
000602                  LDY          #RWHI
000603                  LDA          (INDRCN),Y              ;set datarw hi
000604                  RTS
000605                  .PAGE
000606  ;-------------------------------------------------------------------
000607  ;
000608  ;  BSYACK completes the cmd-bsy handshake by outputting the response
000609  ;  byte to widget, dropping cmd, and waiting for bsy to go lo.
000610  ;  Enter with the widget response ($55 or $AA) in A.
000611  ;
000612  ;-------------------------------------------------------------------
000613
000614  BSYACK          STA          WR_PORT,X               ;store response byte
000615                  JSR          SET_WRITEDIR            ;enable bus out to widget
000616                  JSR          NTCMDLN1                ;drop cmd
000617                  JSR          WAITBSYLO
000618                  JSR          SETUPREAD               ;restore read state
000619                  JMP          S2M                     ;exit via setting 2 MHz mode
000620
000621
000622  NOTCMDLN        JSR          SETUPREAD
000623  NTCMDLN1        LDY          #NOTCMD
000624                  LDA          (INDRCN),Y
000625                  RTS
000626
000627  SETCMDLN        JSR          SETUPREAD
000628                  LDY          #SETCMD
000629                  LDA          (INDRCN),Y
000630                  RTS
000631
000632  RESET_PIPPIN    LDA          #4
000633                  STA          RESET_FLAG
000634                  ORA          STATUS3
000635                  STA          STATUS3
000636                  LDY          #RST
000637                  PHP
000638                  SEI
000639                  LDA          E_REG
000640                  ORA          #80                     ;or in 1 MHz bit
000641                  STA          E_REG
000642                  PLP
000643                  LDA          (INDRCN),Y
000644                  LDY          #25
000645  $10             DEY
000646                  BNE          $10
```

```
000647                  LDY         #CLRRST
000648                  LDA         (INDRCN),Y              ;clear reset
000649                  JMP         S2M                     ;exit via setting 2 MHz mode

; ###############################################################################################################
; #   END OF FILE:  PROFILE.B.TEXT
; #   LINES      :  649
; #   CHARACTERS :  35175
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; #   Author     :  David T. Craig -- 71533.606@compuserve.com -- Santa Fe, New Mexico USA
; ###############################################################################################################
```

TLA ASSEMBLER LISTING (NOT FORMATTED)

```
PAGE -   0
Current memory available:   25612
0000| 5 .NOMACROLIST
0000| 5 .NOPATCHLIST
0000| 5 .TITLE  "     SOS Profile Driver -- Version 1.30  14-Jan-83"
0000| 5  ;-------------------------------------------------------------------
0000| 5  ;
0000| 5  ;               SOS Profile Driver
0000| 5  ;
0000| 5  ;
0000| 5 ;       Revisions:
0000| 5 ;
0000| 5 ;       1.10R   05-May-82
0000| 5 ;
0000| 5 ;       The GOODEXIT routine was changed to complete the dummy handshake
0000| 5 ;       so any pending Profile spare table updates would get rewritten
0000| 5 ;       on the disk.
0000| 5 ;       The GETBYTES routine was changed so a bad block number would get
0000| 5 ;       flagged as such and not get flagged as a bad byte count.
0000| 5  ;
0000| 5  ;       1.11R   12-May-82
0000| 5 ;
0000| 5 ;       The driver is slowed to 1MHz only when talking to the card and
0000| 5 ;       doing psuedo DMA; this will enable it to make the 5:1 interleave.
0000| 5 ;       The reference to the clear parity address that occurs just after
0000| 5 ;       entering the driver was corrected to be a write instead of a read.
0000| 5  ;
0000| 5  ;       1.11R   14-May-82
0000| 5 ;
0000| 5 ;       The block ID check was removed when reading to speed up the driver
0000| 5 ;       and allow reading the status info which doesn't have a block ID.
0000| 5 ;       The 2nd reset, which prevented the read/write head from being
0000| 5 ;       retracted off the data area on read and write errors, was removed.
0000| 5  ;
0000| 5  ;       1.12R   01-Sep-82
0000| 5 ;
0000| 5 ;       Interrupts were disabled in the LAST_PGE routine before saving
0000| 5 ;       bytes 0 and FE, instead of just before the data transfer, since
0000| 5 ;       those bytes might be changed by an interrupt routine, and the old
0000| 5 ;       instead of the new (correct) values would get restored.
0000| 5  ;
0000| 5  ;       1.30    14-Jan-83
0000| 5 ;
0000| 5 ;       Interrupts are disabled while modifying the environment register
0000| 5 ;       and RSTORENV is changed to leave the screen bit (bit 5) unmodified;
0000| 5 ;       this eleminates spurious screen flashing.  Control code 0 is
0000| 5 ;       processed as a NOP, instead of an error.
0000| 5  ;
0000| 00D1   DEVTYPE         .EQU    0D1
0000| 0001   SUBTYPE         .EQU    01
0000| 0001   MANUF           .EQU    0001            ;Apple Computer Inc.
0000| 1300   RELEASE         .EQU    1300
0000| 2600   MAXBLOCK        .EQU    2600            ;4.86 megabytes (+16K spares)

PAGE -   1            FILE:        SOS Profile Driver -- Version 1.30  14-Jan-83

0000| 5 !               .PAGE
0000| 5  ;-------------------------------------------------------------------
0000| 5  ;
0000| 5  ;  The macro SWITCH performs an N way branch based on a switch index.
0000| 5  ;
0000| 5  ;               SWITCH  [index], [bounds], adrs_table, [*]
0000| 5  ;
0000| 5  ;-------------------------------------------------------------------
0000| 5
0000| 5 .MACRO  SWITCH
0000| 5        .IF     "%1" <> ""      ;If PARM1 is present,
0000| 5        LDA     %1              ;  Load A with switch index
0000| 5        .ENDC
0000| 5        .IF     "%2" <> ""      ;If PARM2 is present,
0000| 5        CMP     #%2+1           ;  Perform bounds checking
0000| 5        BCS     $010            ;  on switch index
0000| 5        .ENDC
0000| 5        ASL     A
0000| 5        TAY
0000| 5        LDA     %3+1,Y          ;Get switch address from table
0000| 5        PHA                     ;  and push onto stack
0000| 5        LDA     %3,Y
0000| 5        PHA
0000| 5        .IF     "%4" <> "*"     ;If PARM4 is omitted,
0000| 5        RTS                     ;  Exit to code
0000| 5        .ENDC                   ;Otherwise, drop through
0000| 5  $010           .ENDM
0000| 5                 .INCLUDE        PROFILE.A.TEXT
2 blocks for procedure code  23390 words left

PAGE -   2  PROFILE   FILE: PROFILE.A.TEXT       SOS Profile Driver -- Version 1.30  14-Jan-83

0000| 5 .PROC   PROFILE
```

```
Current memory available:   24800
0000| FFFF                   .WORD   0FFFF
0002| 3B00  .WORD   59.
0004| 50 72 6F 66 69 6C 65  .ASCII  "Profile Driver -- "
000B| 20 44 72 69 76 65 72
0012| 20 2D 2D 20 )
0016| 43 6F 70 79 72 69 67  .ASCII  "Copyright (C) 1983 by Apple Computer Inc."
001D| 68 74 20 28 43 29 20
0024| 31 39 38 33 20 62 79
002B| 20 41 70 70 6C 65 20
0032| 43 6F 6D 70 75 74 65
0039| 72 20 49 6E 63 2E
003F| 5
003F| 5  ;---------------------------------------------------------------------
003F| 5  ;
003F| 5  ;       Device Information Block (DIB)
003F| 5  ;
003F| 5  ;---------------------------------------------------------------------
003F| 5
003F| 0000   DIB_LINK1       .WORD   0
0041| ****  DIB_ENTRY1       .WORD   MAIN
0043| 08 2  DIB_NAME1        .BYTE   8
0044| 2E 50 52 4F 46 49 4C  '          .ASCII  ".PROFILE"
004B| 45 2
004C| 00 00 00 00 00 00 00  .BLOCK  7, 0
0053| 80 2  DIB_DNUM1        .BYTE   80              ;active, no page alignment
0054| FF 2  DIB_SLOT1        .BYTE   0FF
0055| 00 2  DIB_UNIT1        .BYTE   0
0056| D1 2  DIB_TYPE1        .BYTE   DEVTYPE
0057| 01 2  DIB_SUBTYPE1     .BYTE   SUBTYPE
0058| 00 2 !                .BYTE   0
0059| 0026   DIB_BLOCK1      .WORD   MAXBLOCK
005B| 0100   DIB_MID1        .WORD   MANUF
005D| 0013   DIB_RLS1        .WORD   RELEASE
005F| 5
005F| 0100   DIB_DCBCNT1     .WORD   0001            ;Configuration Block
0061| FF 2  WRTVER           .BYTE   0FF

PAGE -   3  PROFILE   FILE: PROFILE.A.TEXT       SOS Profile Driver -- Version 1.30  14-Jan-83

0062| 5        .PAGE
0062| 5  ;---------------------------------------------------------------------
0062| 5  ;
0062| 5  ;       SOS Global Equates  (jump table entry points)
0062| 5  ;
0062| 5  ;---------------------------------------------------------------------
0062| 5
0062| 1913   ALLOCSIR        .EQU    1913            ;allocate System Interrupt Resource
0062| 1916   DEALCSIR        .EQU    1916            ;deallocate  "       "          "
0062| 1922   SELC800         .EQU    1922            ;select/deselect i/o expansion space
0062| 1928   SYSERR          .EQU    1928            ;system error routine
0062| 18F0   DO_DMA          .EQU    18F0            ;place to reloc code for banking
0062| 00F0   INDDMA          .EQU    0F0
0062| 18F7   VECTLO          .EQU    DO_DMA+7
0062| 0018   SOS_ZPAGE       .EQU    18
0062| 5
0062| 5  ;---------------------------------------------------------------------
0062| 5  ;
0062| 5  ;       SOS Error Codes
0062| 5  ;
0062| 5  ;---------------------------------------------------------------------
0062| 5
0062| 0020   XREQCODE        .EQU    20              ;Invalid request code
0062| 0021   XCTLCODE        .EQU    21              ;Invalid control/status code
0062| 0025   XNORESRC        .EQU    25              ;resource not available
0062| 0026   XBADOP          .EQU    26              ;Invalid operation
0062| 0027   XIOERROR        .EQU    27              ;I/O error
0062| 0028   XNODRIVE        .EQU    28              ;No drive connected
0062| 002B   XNOWRITE        .EQU    2B              ;Device write protected.(not supported)
0062| 002C   XBYTECNT        .EQU    2C              ;Byte count <> a multiple of 512
0062| 002D   XBLKNUM         .EQU    2D              ;Block number too large
0062| 0027   BADOLDDATA      .EQU    27              ;block has bad data from previous read
0062| 0027  SPTBLOVFLW      .EQU    27              ;spare table overflow
0062| 5
0062| 5  ;---------------------------------------------------------------------
0062| 5  ;
0062| 5  ;       Hardware I/O Addresses
0062| 5  ;
0062| 5  ;---------------------------------------------------------------------
0062| 5
0062| FFEF   BANK_REG        .EQU    0FFEF
0062| FFDF   E_REG           .EQU    0FFDF           ;system environment register
0062| FFD0   Z_REG           .EQU    0FFD0           ;Zero page register for psuedo DMA
0062| F800   PSUEDO_DMA      .EQU    0F800           ;psuedo DMA code in ROM.
0062| C080   WR_PORT         .EQU    0C080           ;Write to Z8 RAM (byte at a time)
0062| C081   RD_PORT         .EQU    0C081           ;Read from Z8 RAM (byte at a time)
0062| C082   BUSY            .EQU    0C082           ;Z8 not ready
0062| C083  CLR_PARITY      .EQU    0C083           ;Clear parity error.

PAGE -   4  PROFILE   FILE: PROFILE.A.TEXT       SOS Profile Driver -- Version 1.30  14-Jan-83

0062| 5 .PAGE
0062| 5  ;---------------------------------------------------------------------
```

```
0062| 5  ;
0062| 5  ;       Constants
0062| 5  ;
0062| 5  ;-------------------------------------------------------------------------
0062| 5
0062| 0000  NOTCMD          .EQU    0               ;Command line Low
0062| 0004   SETCMD          .EQU    4               ;Command line High
0062| 0001  SETWRT          .EQU    1               ;Low
0062| 0005  SETRD           .EQU    5               ;High
0062| 0002  INTDSABL        .EQU    2
0062| 0006  INTENABL        .EQU    6
0062| 0003  RWLO            .EQU    3
0062| 0007  RWHI            .EQU    7
0062| 000C  RST             .EQU    0C
0062| 0008  CLRRST          .EQU    8
0062| 5
0062| 5  ;-------------------------------------------------------------------------
0062| 5  ;
0062| 5 ;       Command codes
0062| 5  ;
0062| 5  ;-------------------------------------------------------------------------
0062| 5
0062| 0000   WDGTRD          .EQU    0
0062| 0002  WDGTWRTVER      .EQU    2
0062| 0003  WDGTSTAT        .EQU    3
0062| 0001  WDGTWRT         .EQU    1
0062| 5
0062| 5  ;-------------------------------------------------------------------------
0062| 5  ;
0062| 5  ;       SIR allocation table
0062| 5  ;
0062| 5  ;-------------------------------------------------------------------------
0062| 5
0062| ****   SIRADDR         .WORD   SIRTABLE
0064| 10 00 /  SIRTABLE        .BYTE   10, 0
0066| 0000                    .WORD   0
0068| 00 2  SIRBANK         .BYTE   0
0069| 0005   SIRCOUNT        .EQU    *-SIRTABLE

PAGE -   5  PROFILE   FILE: PROFILE.A.TEXT       SOS Profile Driver -- Version 1.30  14-Jan-83

0069| 5                 .PAGE
0069| 5  ;-------------------------------------------------------------------------
0069| 5  ;
0069| 5  ;       Local variables
0069| 5  ;
0069| 5  ;-------------------------------------------------------------------------
0069| 5
0069| 00 2  SLOTX           .BYTE   0
006A| 00 2  ERROR           .BYTE   0               ;if = 0 then no error
006B| 00 2  IERROR          .BYTE   0               ;if = 0 then no initialization error
006C| 5
006C| 00 2  PREVCMD         .BYTE   0
006D| 00 2  PREVUNIT        .BYTE   0
006E| 00 2  LENGTH          .BYTE   0               ;number of blocks to read.
006F| 0000  ORGADR          .WORD   0
0071| 00 2 ORGBNK          .BYTE   0
0072| 00 2 SLOTCN          .BYTE   0
0073| 00 2  ADDRDMA         .BYTE   0
0074| 00 2  BANKDMA         .BYTE   0
0075| 00 2  TEMP00          .BYTE   0
0076| 00 2  TEMPFE          .BYTE   0
0077| 00 2  MVCNT           .BYTE   0
0078| 00 2  CNTLO           .BYTE   0
0079| 00 2  CNTHI           .BYTE   0
007A| 00 2  ADMODE          .BYTE   0
007B| 00 2  DATDIR          .BYTE   0
007C| 00 2  Z8CMD           .BYTE   0
007D| 00 2  COUNTR          .BYTE   0
007E| 00 2 MSBLOCK         .BYTE   0               ;Most significant block # for Profile
007F| 00 2 BSYLO           .BYTE   0               ;FLAG set when driver has seen busy lo
0080| 00 2 LONGWAIT        .BYTE   0
0081| 00 2 WAITTIME        .BYTE   0
0082| 00 2 PIPPIN_RESET    .BYTE   0
0083| 00 2 STATUS1         .BYTE   0
0084| 00 2 STATUS2         .BYTE   0
0085| 00 2 STATUS3         .BYTE   0
0086| 00 2  STATUS4         .BYTE   0
0087| 00 2 PARITY_ERR      .BYTE   0
0088| 00 2 RESET_FLAG      .BYTE   0
0089| 00 2 BAD_RESPONSE    .BYTE   0
008A| 0000   CHEKBYTS        .WORD   0
008C| 0000  .WORD   0
008E| 0000  .WORD   0
0090| 00 2 SVENV           .BYTE   0
0091| 00 2  DMA_CNT         .BYTE   0

PAGE -   6  PROFILE   FILE: PROFILE.A.TEXT       SOS Profile Driver -- Version 1.30  14-Jan-83

0092| 5 .PAGE
0092| 5  ;-------------------------------------------------------------------------
0092| 5  ;
0092| 5  ;       SOS Call Parameter Locations
```

```
0092| 5  ;
0092| 5  ;---------------------------------------------------------------------
0092| 5
0092| 00C0   SOS_REQCODE     .EQU    0C0
0092| 00C1   SOS_UNIT        .EQU    0C1
0092| 5
0092| 00C2   SOS_BUF         .EQU    0C2             ;D_READ/D_WRITE SOS calls
0092| 00C4   SOS_BYTES       .EQU    0C4
0092| 00C6   SOS_BLOCK       .EQU    0C6
0092| 00C8   SOS_BYTRD       .EQU    0C8
0092| 5
0092| 00C2   SOS_STCODE      .EQU    0C2             ;D_STATUS SOS call
0092| 00C3   SOS_STLIST      .EQU    0C3
0092| 5
0092| 1400   SOS_XPAGE       .EQU    1400
0092| 5
0092| 5  ;---------------------------------------------------------------------
0092| 5  ;
0092| 5  ;       More Zero page equates
0092| 5  ;
0092| 5  ;---------------------------------------------------------------------
0092| 5
0092| 00CA  ADRLO           .EQU    0CA             ;Indirect address to user data
0092| 00CB   ADRHI           .EQU    ADRLO+1
0092| 14CB   SISADR          .EQU    SOS_XPAGE+ADRHI
0092| 00CC  INDRCN          .EQU    0CC             ;Indirect to $CN00 device locations
0092| 14CD  SISCN           .EQU    SOS_XPAGE+INDRCN+1
0092| 00CE   TIMOUT          .EQU    0CE
0092| 00CF  RTRYCNT         .EQU    0CF
0092| 00D0   RTRYTHRESH      .EQU    0D0
0092| 00D1   CMD_RTRYCNT     .EQU    0D1
0092| 00D2   RSPNS           .EQU    0D2
0092| 00D3   BLK_RTRYCNT     .EQU    0D3
0092| 00D4   BLOCKLO         .EQU    0D4
0092| 00D5  BLOCKHI         .EQU    0D5

PAGE -   7  PROFILE   FILE: PROFILE.A.TEXT       SOS Profile Driver -- Version 1.30  14-Jan-83

0092| 5                  .PAGE
0092| 5  ;---------------------------------------------------------------------
0092| 5  ;
0092| 5  ;  Psuedo DMA transfer routine.  Does bank switching first.
0092| 5  ;
0092| 5  ;---------------------------------------------------------------------
0092| 5
0092| AC EFFF -  RELCODE         LDY     BANK_REG        ;This Code is moved to $18F0
0095| 8E EFFF - STX     BANK_REG        ;  so the bank can be switched
0098| 20 00F8 - JSR     PSUEDO_DMA      ;  for psuedo DMA transfers to
009B| 8C EFFF - STY     BANK_REG        ;  other than driver bank.
009E| 60 2 RTS
009F| 000D   CDELEN          .EQU    *-RELCODE

PAGE -   8  PROFILE   FILE: PROFILE.A.TEXT       SOS Profile Driver -- Version 1.30  14-Jan-83

009F| 5 .PAGE
009F| 5  ;---------------------------------------------------------------------
009F| 5  ;
009F| 5  ;  Profile driver -- Main entry point
009F| 5  ;
009F| 5  ;---------------------------------------------------------------------
009F| 5
009F| 009F   MAIN            .EQU    *
009F| AD DFFF - LDA     E_REG
00A2| 29 DF / AND     #0DF            ;Save everything but
00A4| 8D 9000 - STA     SVENV             ;  the current screen state
00A7| A5 C0 /                 LDA     SOS_REQCODE
00A9| 49 08 / EOR     #8
00AB| F0**  BEQ     INIT_ENT
00AD| AD 5400 - LDA     DIB_SLOT1
00B0| 20 2219 - JSR     SELC800
00B3| 90**  BCC     $20
00B5| 4C **** - JMP     NODRV
00B8| AD 6B00 -  $20             LDA     IERROR          ;did D_INIT encounter error?
00BB| F0**  BEQ     ONEMEG
00BD| 20 2819 - JSR     SYSERR
00C0| 8D 7200 -  INIT_ENT        STA     SLOTCN
00C3| A9 FF /  ONEMEG          LDA     #0FF
00C5| 8D 8200 -                  STA     PIPPIN_RESET
00C8| 8D 7D00 - STA     COUNTR
00CB| A2 0C / LDX     #CDELEN-1       ;Move code to page 18 for DMA
00CD| BD 9200 -  MOVCODE         LDA     RELCODE,X       ;  to other possible banks.
00D0| 95 F0 / STA     INDDMA,X
00D2| CA 2 DEX
00D3| 10F8  BPL     MOVCODE
00D5| A9 00 / LDA     #0
00D7| 8D 8700 - STA     PARITY_ERR
00DA| 8D 8900 - STA     BAD_RESPONSE
00DD| 8D 8800 - STA     RESET_FLAG
00E0| 85 CC / STA     INDRCN
00E2| 8D CD14 - STA     SISCN
00E5| A5 C6 / LDA     SOS_BLOCK
00E7| 85 D4 / STA     BLOCKLO
00E9| A5 C7 / LDA     SOS_BLOCK+1
```

```
00EB| 85 D5 / STA     BLOCKHI
00ED| AE 6900 - LDX     SLOTX
00F0| 08 2 PHP
00F1| 78 2 SEI
00F2| AD DFFF - LDA     E_REG
00F5| 09 83 / ORA     #83             ;slow to 1mhz, ROM enable
00F7| 8D DFFF - STA     E_REG
00FA| 28 2 PLP
00FB| AD 7200 - LDA     SLOTCN
00FE| 10**  BPL     GO_INIT         ;Branch if not initialized.
0100| 85 CD / STA     INDRCN+1
0102| 9D 83C0 - STA     CLR_PARITY,X    ;Clear any previous parity errors.
0105| BC 82C0 - LDY     BUSY,X
0108| 20 **** - JSR     S2M             ;set 2MHz mode
010B| 98 2 TYA

PAGE -   9  PROFILE   FILE: PROFILE.A.TEXT       SOS Profile Driver -- Version 1.30  14-Jan-83

010C| 29 01 / AND     #1
010E| D0**  BNE     NODRV
0110| 98 2 TYA
0111| 10**  BPL     $40
0113| A9 FF / LDA     #0FF
0115| 8D 7F00 - STA     BSYLO
0118| D0**  BNE     GO_INIT
011A| AD 7F00 -  $40             LDA     BSYLO           ;has BSY been low?
011D| D0**  BNE     GO_INIT         ;branch if so
011F| 20 **** - JSR     RSTORENV
0122| A9 27 / LDA     #XIOERROR
0124| 20 2819 - JSR     SYSERR          ;otherwise error exit
0127| 5
0127| 20 **** -  GO_INIT         JSR     DISPATCH
012A| 20 **** - JSR     SETUPWRITE
012D| A9 00 / LDA     #0
012F| 20 2219 - JSR     SELC800
0132| 08 2 RSTORENV        PHP
0133| 78 2 SEI
0134| AD DFFF - LDA     E_REG
0137| 29 20 / AND     #20
0139| 0D 9000 - ORA     SVENV           ;restore environment register
013C| 8D DFFF - STA     E_REG
013F| 28 2 PLP
0140| 60 2 RTS                     ;  and exit to caller
0141| 5
0141| 08 2 S2M             PHP
0142| 78 2 SEI
0143| AD DFFF - LDA     E_REG
0146| 29 7F / AND     #7F             ;and out 1MHz bit
0148| 8D DFFF - STA     E_REG
014B| 28 2 PLP
014C| 60 2 RTS

PAGE -  10  PROFILE   FILE: PROFILE.A.TEXT       SOS Profile Driver -- Version 1.30  14-Jan-83

014D| 5 .PAGE
014D| 5  DISPATCH        SWITCH  SOS_REQCODE, 9, SW_TABLE
015E| 5 $
015E| A9 20 /  BADREQ          LDA     #XREQCODE       ;Invalid request code
0160| 20 2819 - JSR     SYSERR
0163| 5
0163| A9 26 /  BADOP           LDA     #XBADOP         ;Invalid operation
0165| 20 2819 -                  JSR     SYSERR
0168| 5
0168| 20 3201 -  NODRV           JSR     RSTORENV
016B| A9 28 / LDA     #XNODRIVE       ;No drive connected (power off)
016D| 20 2819 -  SYS_ERROR       JSR     SYSERR
0170| 5
0170| 5  ;---------------------------------------------------------------------
0170| 5
0170| 0170   SW_TABLE        .EQU    *               ;Profile driver switch table
0170| ****  .WORD   DREAD-1          ;  D_READ system call
0172| ****  .WORD   DWRITE-1         ;  D_WRITE      "
0174| ****  .WORD   DSTATUS-1        ;  D_STATUS     "
0176| ****  .WORD   FDCONTROL-1      ;  D_CONTROL    "
0178| 5D01  .WORD   BADREQ-1         ;  GET_DEV_NUM  "
017A| 5D01  .WORD   BADREQ-1         ;  D_INFO       "
017C| 6201  .WORD   BADOP-1          ;  D_OPEN       "
017E| 6201  .WORD   BADOP-1          ;  D_CLOSE      "
0180| ****  .WORD   DINIT-1          ;  D_INIT       "
0182| ****  .WORD   DREPEAT-1        ;  D_REPEAT     "

PAGE -  11  PROFILE   FILE: PROFILE.A.TEXT       SOS Profile Driver -- Version 1.30  14-Jan-83

0184| 5                 .PAGE
0184| 5  ;---------------------------------------------------------------------
0184| 5  ;
0184| 5  ;  Profile driver -- initialization request.
0184| 5 ;
0184| 5  ;---------------------------------------------------------------------
0184| 5
0184| 0184   DINIT           .EQU    *
0184| AD 5400 - LDA     DIB_SLOT1       ;slot index:=dib.slot*16
0187| AA 2 TAX
```

```
0188| 30**  BMI     DI_ERR1         ;  error, invalid slot #
018A| 09 C0 / ORA     #0C0
018C| 8D 7200 - STA     SLOTCN          ;Save slot CN address
018F| 85 CD / STA     INDRCN+1
0191| 0A 2                 ASL     A
0192| 0A 2 ASL     A
0193| 0A 2 ASL     A
0194| 0A 2 ASL     A
0195| 8D 6900 - STA     SLOTX
0198| A9 00 / LDA     #0
019A| 85 CC / STA     INDRCN
019C| 8D CD14 - STA     SISCN
019F| AD 5400 - LDA     DIB_SLOT1       ;compute SIR #
01A2| 18 2 CLC
01A3| 6D 6400 - ADC     SIRTABLE        ;  sir:=dib.slot1+16
01A6| 8D 6400 - STA     SIRTABLE
01A9| A9 05 / LDA     #SIRCOUNT       ;allocate the slot's SIR
01AB| AE 6200 - LDX     SIRADDR
01AE| AC 6300 - LDY     SIRADDR+1
01B1| 20 1319 - JSR     ALLOCSIR
01B4| B0**  BCS     DI_ERR2         ;SIR request failed
01B6| AD 5400 - LDA     DIB_SLOT1
01B9| 20 2219 - JSR     SELC800
01BC| B0**  BCS     DI_ERR1
01BE| 20 **** -                 JSR     SETUPREAD
01C1| A0 02 / LDY     #INTDSABL
01C3| B1 CC / LDA     (INDRCN),Y
01C5| 20 **** - JSR     SETUPWRITE
01C8| 20 **** - JSR     SETUPREAD
01CB| 18 2 CLC
01CC| 60 2                   RTS                     ;normal exit
01CD| 5
01CD| A9 28 /  DI_ERR1         LDA     #XNODRIVE       ;error, checksum fail/bad slot#
01CF| D0**  BNE     DI_ERR3
01D1| A9 25 /  DI_ERR2         LDA     #XNORESRC       ;error, SIR resource not available
01D3| 8D 6B00 -  DI_ERR3         STA     IERROR
01D6| 20 2819 - JSR     SYSERR

PAGE -  12  PROFILE   FILE: PROFILE.A.TEXT       SOS Profile Driver -- Version 1.30  14-Jan-83

01D9| 5 .PAGE
01D9| 5  ;--------------------------------------------------------------------------
01D9| 5  ;
01D9| 5  ;  Profile driver -- status request.
01D9| 5 ;
01D9| 5 ;  Status request zero returns BUSY (bit 7), and ONLINE (bit 4).
01D9| 5  ;  Status request one returns device specific flags.
01D9| 5 ;  Lastly, the format status code $FE is also accepted.
01D9| 5  ;
01D9| 5  ;--------------------------------------------------------------------------
01D9| 5
01D9| 01D9   DSTATUS         .EQU    *
01D9| A0 00 / LDY     #0
01DB| A6 C2 / LDX     SOS_STCODE
01DD| D0**  BNE     $10
01DF| AE 6900 - LDX     SLOTX
01E2| 08 2 PHP
01E3| 78 2 SEI
01E4| AD DFFF - LDA     E_REG
01E7| 09 80 / ORA     #80             ;or in 1MHz bit
01E9| 8D DFFF - STA     E_REG
01EC| 28 2 PLP
01ED| BD 82C0 - LDA     BUSY,X
01F0| 49 80 /                EOR     #80
01F2| 0A 2 ASL     A               ;Save Busy status bit in carry
01F3| 08 2 PHP                     ;Save carry
01F4| 0A 2 ASL     A               ;Now shift On-Line to Bit 5.
01F5| 0A 2 ASL     A
01F6| 0A 2 ASL     A
01F7| 0A 2 ASL     A
01F8| 29 20 / AND     #20             ;Throw away other (garbage) bits
01FA| 28 2 PLP                     ;Get Busy bit again
01FB| 6A 2 ROR     A               ;And shift it to Bit 7, and On-line to 4
01FC| 91 C3 / STA     (SOS_STLIST),Y
01FE| 60 2 RTS                     ;(carry is always cleared from shift)
01FF| 5 !
01FF| CA 2  $10             DEX
0200| D0**  BNE     FORMATUS                ;branch not status code 1.
0202| A0 03 / LDY     #3
0204| B9 8300 - $15             LDA     STATUS1,Y
0207| 91 C3 / STA     (SOS_STLIST),Y          ;put status bytes in callers buf
0209| 88 2 DEY
020A| 10F8  BPL     $15
020C| 18 2 CLC
020D| 60 2 RTS

PAGE -  13  PROFILE   FILE: PROFILE.A.TEXT       SOS Profile Driver -- Version 1.30  14-Jan-83

020E| 5                  .PAGE
020E| 020E   FORMATUS        .EQU    *
020E| A5 C2 / LDA     SOS_STCODE      ;Test for format request.
0210| C9 FE / CMP     #0FE
0212| D0**  BNE     FCODE           ;error, invalid status code
```

```
0214| A0 00 / LDY     #0
0216| A9 FF / LDA     #0FF
0218| 91 C3 / STA     (SOS_STLIST),Y
021A| C8 2 INY
021B| 91 C3 / STA     (SOS_STLIST),Y
021D| 18 2 CLC
021E| 60 2 RTS
021F| 5
021F| A9 21 /  FCODE           LDA     #XCTLCODE       ;invalid status/control code
0221| 20 2819 - JSR     SYSERR
```



```
0224| 5 .PAGE
0224| 5  ;---------------------------------------------------------------------
0224| 5  ;
0224| 5  ;  Profile driver -- control request.
0224| 5 ;
0224| 5 ;  Control code must be $00 or $FE.  The reset request is ignored and
0224| 5  ;  the Profile is preformatted, hence both requests are NOPs.
0224| 5  ;
0224| 5  ;---------------------------------------------------------------------
0224| 5
0224| 0224  FDCONTROL       .EQU    *
0224| A5 C2 / LDA     SOS_STCODE      ;fetch control code
0226| F0**  BEQ     $010
0228| C9 FE / CMP     #0FE
022A| D0F3  BNE     FCODE           ;error, invalid control code
022C| 18 2 $010           CLC
022D| 60 2 RTS
022E| 5
022E| 5
022E| 5  ;---------------------------------------------------------------------
022E| 5  ;
022E| 5  ;  Profile driver -- repeat request.
022E| 5 ;
022E| 5 ;  Retrieves the previous command and calls DREAD or DWRITE.
022E| 5  ;
022E| 5  ;---------------------------------------------------------------------
022E| 5
022E| 022E   DREPEAT         .EQU    *
022E| AD 6D00 - LDA     PREVUNIT
0231| C5 C1 / CMP     SOS_UNIT
0233| D0**  BNE     $1              ;Error exit if unit # changed
0235| AD 6C00 - LDA     PREVCMD
0238| 85 C0 / STA     SOS_REQCODE
023A| D0**  BNE     $2
023C| 4C **** - JMP     DREAD
023F| C9 02 / $2              CMP     #2
0241| B0**  BCS     $1              ;Error, previous I/O not Read/Write
0243| 4C **** - JMP     DWRITE
0246| 4C 6301 -  $1              JMP     BADOP
```



```
0249| 5 .PAGE
0249| 5  ;---------------------------------------------------------------------
0249| 5 ;
0249| 5 ;  This routine tests the validity of block number requested.
0249| 5 ;
0249| 5  ;---------------------------------------------------------------------
0249| 5
0249| A9 FF / TSTBLKNUM       LDA     #0FF
024B| 8D 7E00 - STA     MSBLOCK         ;assume status block
024E| C5 D5 / CMP     BLOCKHI
0250| 18 2 CLC
0251| D0**  BNE     $10
0253| A5 D4 / LDA     BLOCKLO
0255| C9 FE / CMP     #0FE            ;RAM or status block?
0257| 90**  BCC     $10             ;branch if not
0259| 18 2 CLC
025A| 60 2 RTS                     ;OK exit
025B| EE 7E00 - $10             INC     MSBLOCK
025E| A5 D4 / LDA     BLOCKLO         ;Make sure Block number is
0260| CD 5900 - CMP     DIB_BLOCK1      ;  within widget's range.
0263| A5 D5 /                 LDA     BLOCKHI
0265| ED 5A00 - SBC     DIB_BLOCK1+1    ;If carry clear then within range.
0268| 60 2 $20            RTS
```



```
0269| 5 .PAGE
0269| 5  ;---------------------------------------------------------------------
0269| 5  ;
0269| 5  ;  Getbytes subroutine.  Moves SOS bytes parameter to length variable.
0269| 5 ;  If bytes are not a multiple of 512, or a block # > maxblock will be
0269| 5 ;  written or read, then XBYTECNT error exit will be done.  If the first
0269| 5 ;  block to be read > MAXBLOCK-1, an XBLKNUM error exit will be done.
0269| 5 ;
0269| 5  ;---------------------------------------------------------------------
0269| 5
0269| 0269   GETBYTES       .EQU    *
0269| 20 4902 - JSR     TSTBLKNUM       ;check validity of 1st block to be
```

```
026C| 5 ;                                       written or read
026C| 90**                  BCC     $01             ;branch if OK
026E| 68 2 PLA
026F| 68 2 PLA
0270| A9 2D / LDA     #XBLKNUM
0272| 20 2819 - JSR     SYSERR          ;bad block number error
0275| A5 C4 / $01             LDA     SOS_BYTES
0277| D0**  BNE     $10
0279| A5 C5 / LDA     SOS_BYTES+1
027B| 4A 2 LSR     A
027C| B0**  BCS     $10
027E| 8D 6E00 - STA     LENGTH
0281| D0**  BNE     $05
0283| 68 2 PLA
0284| 68 2 PLA
0285| 60 2 RTS
0286| A6 D5 / $05             LDX     BLOCKHI
0288| A4 D4 / LDY     BLOCKLO
028A| 38 2 SEC
028B| E9 01 / SBC     #1
028D| 18 2 CLC
028E| 65 D4 / ADC     BLOCKLO
0290| 85 D4 / STA     BLOCKLO
0292| A5 D5 / LDA     BLOCKHI
0294| 69 00 / ADC     #0
0296| B0**  BCS     $10
0298| 85 D5 / STA     BLOCKHI         ;compute last block to be wrtn or rd
029A| 20 4902 - JSR     TSTBLKNUM       ;is it valid?
029D| 86 D5 / STX     BLOCKHI
029F| 84 D4 / STY     BLOCKLO
02A1| 90**  BCC     CALCHEK         ;branch if GOOD!
02A3| 5
02A3| 68 2 $10             PLA
02A4| 68 2 PLA
02A5| A9 2C / LDA     #XBYTECNT       ;byte count not multiple of 512
02A7| 20 2819 - JSR     SYSERR
02AA| 5
02AA| 5
02AA| 98 2 CALCHEK         TYA
02AB| 8D 8C00 - STA     CHEKBYTS+2
02AE| 49 FF / EOR     #0FF
02B0| 8D 8F00 - STA     CHEKBYTS+5

PAGE -  17  PROFILE   FILE: PROFILE.A.TEXT       SOS Profile Driver -- Version 1.30  14-Jan-83

02B3| 8A 2 TXA
02B4| 8D 8B00 - STA     CHEKBYTS+1
02B7| 49 FF / EOR     #0FF
02B9| 8D 8E00 - STA     CHEKBYTS+4
02BC| A9 00 / LDA     #0
02BE| 8D 8A00 - STA     CHEKBYTS
02C1| A9 FF / LDA     #0FF
02C3| 8D 8D00 - STA     CHEKBYTS+3
02C6| 60 2 RTS

PAGE -  18  PROFILE   FILE: PROFILE.A.TEXT       SOS Profile Driver -- Version 1.30  14-Jan-83

02C7| 5 .PAGE
02C7| 5  ;---------------------------------------------------------------------
02C7| 5  ;
02C7| 5  ;  Profile driver -- read request.
02C7| 5 ;
02C7| 5 ;  Transfers "bytes/512" blocks from disk to buffer..buffer+bytes-1.
02C7| 5  ;
02C7| 5  ;---------------------------------------------------------------------
02C7| 5
02C7| 02C7   DREAD           .EQU    *
02C7| A9 00 / LDA     #0
02C9| A8 2 TAY
02CA| 91 C8 / STA     (SOS_BYTRD),Y
02CC| C8 2 INY
02CD| 91 C8 / STA     (SOS_BYTRD),Y
02CF| A5 C1 / LDA     SOS_UNIT
02D1| 8D 6D00 - STA     PREVUNIT        ;save current sos unit
02D4| A5 C0 /                LDA     SOS_REQCODE     ;save current command
02D6| 8D 6C00 - STA     PREVCMD
02D9| 20 6902 - JSR     GETBYTES        ;move bytes parm to length var.
02DC| 20 **** - JSR     ARBADR          ;Get beginning address resolved.
02DF| A9 02 /  RDBLOCK          LDA     #2              ;Allow 2 retries
02E1| 85 D3 / STA     BLK_RTRYCNT
02E3| 02E3   RPAR_RETRY      .EQU    *               ;Parity error retries enter here
02E3| A9 00 / LDA     #0              ;Indicate data direction is
02E5| 8D 7B00 - STA     DATDIR          ;read.
02E8| 85 D1 / STA     CMD_RTRYCNT     ;clear # communication tries
02EA| A9 0A / LDA     #0A             ;set retry count to 10
02EC| 85 CF / STA     RTRYCNT
02EE| A9 03 / LDA     #3
02F0| 85 D0 / STA     RTRYTHRESH      ;set reseek/rewrite threshold
02F2| 20 4902 - JSR     TSTBLKNUM       ;make sure it's a valid block
02F5| 90**  BCC     RCOM_RETRY
02F7| A9 2D /               LDA     #XBLKNUM
02F9| 20 2819 - JSR     SYSERR          ;invalid block error exit
02FC| 02FC   RCOM_RETRY      .EQU    *               ;CMD-BSY bad response retries enter here
```

```
02FC| A9 00 / LDA     #0
02FE| 8D 8000 - STA     LONGWAIT
0301| A9 00 / LDA     #WDGTRD
0303| 8D 7C00 - STA     Z8CMD
0306| A9 01 / LDA     #1
0308| 85 D2 / STA     RSPNS           ;set up expected Z8 response
030A| 20 **** - JSR     SNDCMD          ;do a CMD-BSY handshake
030D| 90**  BCC     $70
030F| 4C **** - JMP     DR_ERR1
0312| D0E8  $70              BNE     RCOM_RETRY      ;bad response - try again.
0314| 5
0314| 5  ;               Send command bytes
0314| 5
0314| 20 **** - JSR     SND_CMDBYTES
0317| B0**  BCS     RDRETRY         ;try again if parity error
0319| A9 02 / LDA     #2
031B| 85 D2 / STA     RSPNS           ;set up for execution
031D| A9 FF / LDA     #0FF

PAGE -  19  PROFILE   FILE: PROFILE.A.TEXT       SOS Profile Driver -- Version 1.30  14-Jan-83

031F| 8D 8000 - STA     LONGWAIT        ;adjust timeout wait
0322| 20 **** - JSR     SNDCMD          ;2nd CMD-BSY for read opn
0325| B0**  BCS     DR_ERR1         ;timeout on cmd-bsy
0327| D0D3  BNE     RCOM_RETRY      ;bad response - try again.
0329| 5
0329| 5  ;               read should be complete at this point
0329| 5
0329| 20 **** - JSR     GETSTAT         ;get status bytes first
032C| B0**  BCS     RDRETRY         ;try again if parity error
032E| 10**  BPL     $10
0330| EE 7D00 - INC     COUNTR          ;time to reset PIPPIN?
0333| F0C7  BEQ     RCOM_RETRY      ;branch if not
0335| D0**  BNE     DR_ERR1         ;reset PIPPIN
0337| 20 **** - $10             JSR     DATRANS
033A| B0**  BCS     RDRETRY
033C| AD 8300 - LDA     STATUS1         ;Now check status for good read
033F| 29 01 / AND     #1
0341| D0**  BNE     LDAXIO
0343| AD 8400 - LDA     STATUS2
0346| 29 58 / AND     #58
0348| D0**  BNE     LDAXIO
034A| A9 02 / LDA     #2
034C| A0 01 / LDY     #1
034E| 71 C8 / ADC     (SOS_BYTRD),Y
0350| 91 C8 / STA     (SOS_BYTRD),Y
0352| 20 **** - JSR     TSTMORE         ;more blocks to read?
0355| F0**  BEQ     GOODEXIT
0357| 4C DF02 - JMP     RDBLOCK         ;jump if more
035A| 5
035A| 20 **** -  RDRETRY         JSR     RSTORADR        ;restore 'ORGADR' and try again
035D| C6 D3 /                  DEC     BLK_RTRYCNT     ;can we retry?
035F| 30**  BMI     DR_ERR1
0361| 4C E302 - JMP     RPAR_RETRY
0364| 5
0364| EE 8200 -  DR_ERR1         INC     PIPPIN_RESET
0367| D0**  BNE     $10
0369| 20 **** - JSR     RESET_PIPPIN
036C| A9 00 / LDA     #0
036E| 85 D1 / STA     CMD_RTRYCNT
0370| 4C FC02 - JMP     RCOM_RETRY
0373| 20 **** - $10             JSR     NOTCMDLN
0376| A9 27 / LDAXIO          LDA     #XIOERROR
0378| D0**                  BNE     BADEXIT
037A| 5
037A| E6 D4 / TSTMORE         INC     BLOCKLO         ;Bump the block number
037C| D0**  BNE     $010
037E| E6 D5 / INC     BLOCKHI
0380| A4 D4 / $010            LDY     BLOCKLO
0382| A6 D5 / LDX     BLOCKHI
0384| 20 AA02 - JSR     CALCHEK
0387| A5 CB / LDA     ADRHI
0389| 8D 7000 - STA     ORGADR+1
038C| AD CB14 - LDA     SISADR
038F| 8D 7100 - STA     ORGBNK
0392| CE 6E00 -                 DEC     LENGTH          ;Are there more blocks to read?

PAGE -  20  PROFILE   FILE: PROFILE.A.TEXT       SOS Profile Driver -- Version 1.30  14-Jan-83

0395| 60 2 RTS                    ;Return Z flag set if done.
0396| 5
0396| A9 01 / GOODEXIT        LDA     #01
0398| 85 D2 / STA     RSPNS           ;DUMMY HANDSHAKE FOR PROFILE TO UPDATE
039A| 20 **** - JSR     SNDCMD          ; ANY PENDING INFO TO DISK.
039D| A9 FF / LDA     #0FF
039F| 8D 7C00 - STA     Z8CMD           ;we're going to send an invalid command
03A2| 20 **** - JSR     SND_CMDBYTES
03A5| 20 **** - JSR     SETCMDLN        ;raise the CMD line
03A8| 20 **** - JSR     SETCMDLN        ;give Profile a little time
03AB| 20 **** - JSR     NTCMDLN1        ;lower the CMD line (so the ready light
03AE| 5 ;                                       doesn't go out)
03AE| 60 2                 RTS
03AF| 5
```

```
03AF| 48 2 BADEXIT         PHA
03B0| 20 9603 - JSR     GOODEXIT
03B3| 68 2 PLA
03B4| 20 2819 - JSR     SYSERR
03B7| 5
03B7| AD 6F00 - RSTORADR        LDA     ORGADR          ;Reset addresses
03BA| 85 C2 / STA     SOS_BUF
03BC| AD 7000 - LDA     ORGADR+1
03BF| 85 C3 / STA     SOS_BUF+1
03C1| AD 7100 - LDA     ORGBNK
03C4| 8D C314 - STA     SOS_XPAGE+SOS_BUF+1
03C7| 4C **** - JMP     ARBADR
03CA| 5
03CA| 5 .INCLUDE        PROFILE.B.TEXT

PAGE -  21  PROFILE   FILE: PROFILE.B.TEXT       SOS Profile Driver -- Version 1.30  14-Jan-83

03CA| 5 .PAGE
03CA| 5  ;---------------------------------------------------------------------
03CA| 5  ;
03CA| 5  ;  Profile driver -- write request.
03CA| 5 ;
03CA| 5 ;  Transfers "bytes/512" blocks from buffer to block..block+(bytes/512).
03CA| 5  ;  Error status on return from subroutines is the same as for DREAD.
03CA| 5  ;
03CA| 5  ;---------------------------------------------------------------------
03CA| 5
03CA| 03CA   DWRITE          .EQU    *
03CA| A5 C0 /                 LDA     SOS_REQCODE     ;save current command
03CC| 8D 6C00 - STA     PREVCMD
03CF| A5 C1 / LDA     SOS_UNIT
03D1| 8D 6D00 - STA     PREVUNIT
03D4| 20 6902 - JSR     GETBYTES        ;move bytes parm to length var.
03D7| 20 **** - JSR     ARBADR          ;Get beginning address resolved.
03DA| A9 FF / LDA     #0FF            ;Indicate data direction is Write
03DC| 8D 7B00 - STA     DATDIR
03DF| 5
03DF| A9 02 / WRBLOCK         LDA     #2
03E1| 85 D3 / STA     BLK_RTRYCNT
03E3| 5
03E3| A9 00 /  WPAR_RETRY      LDA     #0              ;parity error retries enter here
03E5| 85 D1 / STA     CMD_RTRYCNT     ;init cmd-bsy retry variable
03E7| 85 CF / STA     RTRYCNT         ;and read retry and threshold
03E9| 85 D0 / STA     RTRYTHRESH      ;variables to 0 for write opn
03EB| 20 4902 - JSR     TSTBLKNUM       ;check for valid block#
03EE| 90**  BCC     WCOM_RETRY
03F0| 4C **** - JMP     DW_ERR2
03F3| 5
03F3| 5  ;CMD-BSY retries enter here
03F3| 5
03F3| A9 01 /  WCOM_RETRY      LDA     #WDGTWRT
03F5| AC 6100 -                  LDY     WRTVER
03F8| F0**  BEQ     $10
03FA| A9 02 / LDA     #WDGTWRTVER
03FC| 8D 7C00 - $10             STA     Z8CMD
03FF| A9 01 / LDA     #1
0401| 85 D2 / STA     RSPNS           ;set up expected response
0403| 20 **** - JSR     SNDCMD          ;1st cmd-bsy handshake
0406| B0**  BCS     DW_ERR1         ;cmd-bsy timeout error
0408| D0E9  BNE     WCOM_RETRY      ;wrong response - try again
040A| 20 **** - JSR     SND_CMDBYTES    ;send write command string
040D| B0**  BCS     $12             ;retry if parity error
040F| 5
040F| 5  ;Now set up to send write data to widget
040F| 5
040F| AD 7C00 - LDA     Z8CMD
0412| 69 02 / ADC     #2
0414| 85 D2 / STA     RSPNS           ;set up expected response
0416| 20 **** - JSR     SNDCMD          ;2nd cmd-bsy handshake
0419| B0**  BCS     DW_ERR1         ;timeout on cmd-bsy
041B| D0D6  BNE     WCOM_RETRY      ;retry the handshake if taken

PAGE -  22  PROFILE   FILE: PROFILE.B.TEXT       SOS Profile Driver -- Version 1.30  14-Jan-83

041D| 20 **** - JSR     DATRANS         ;now transfer data to widget
0420| 90**  BCC     $15
0422| 4C **** - $12             JMP     WRRETRY         ;retry on parity error
0425| 5
0425| 5  ;now get status from write - 3rd cmd-bsy handshake
0425| 5
0425| A9 06 / $15             LDA     #6
0427| 85 D2 / STA     RSPNS           ;set up expected response
0429| AD 6100 - LDA     WRTVER
042C| F0**  BEQ     $20             ;branch if not write/verify
042E| A9 FF / LDA     #0FF
0430| 8D 8000 - STA     LONGWAIT        ;adjust timeout value
0433| 20 **** - $20             JSR     SNDCMD
0436| B0**  BCS     DW_ERR1         ;timeout error - cmd-bsy
0438| F0**  BEQ     WCONT           ;continue if good return status
043A| 20 B703 - JSR     RSTORADR        ;restore 'ORGADR' since already
043D| 5 H;wrote bytes to z8
043D| 4C F303 - JMP     WCOM_RETRY      ;and retry communication
0440| 5
```

```
0440| EE 8200 -  DW_ERR1         INC     PIPPIN_RESET
0443| D0**  BNE     $10
0445| 20 **** - JSR     RESET_PIPPIN
0448| A9 00 / LDA     #0
044A| 85 D1 / STA     CMD_RTRYCNT
044C| 4C F303 - JMP     WCOM_RETRY
044F| 20 **** - $10             JSR     NOTCMDLN
0452| A9 27 / LDAXIOERR       LDA     #XIOERROR
0454| 4C AF03 - CSYSER1         JMP     BADEXIT
0457| 5
0457| A9 2D /  DW_ERR2         LDA     #XBLKNUM
0459| D0F9  BNE     CSYSER1
045B| 5
045B| 5
045B| 5
045B| 20 **** -  WCONT           JSR     GETSTAT
045E| B0**  BCS     WRRETRY         ;parity error - try again
0460| 10**  BPL     $10
0462| EE 7D00 - INC     COUNTR          ;time to reset PIPPIN?
0465| F0**  BEQ     WRRETRY         ;branch if not
0467| D0D7  BNE     DW_ERR1
0469| 29 41 / $10             AND     #41
046B| D0E5  BNE     LDAXIOERR
046D| AD 8400 - LDA     STATUS2
0470| 29 48 / AND     #48
0472| D0DE  BNE     LDAXIOERR       ;if pippin couldnt read its status
0474| 20 7A03 - JSR     TSTMORE         ;more to write?
0477| F0**  BEQ     DW_EXIT
0479| 4C DF03 - JMP     WRBLOCK
047C| 4C 9603 -  DW_EXIT         JMP     GOODEXIT
047F| 5
047F| 20 B703 -  WRRETRY         JSR     RSTORADR        ;restore 'ORGADR' and try again
0482| C6 D3 / DEC     BLK_RTRYCNT     ;can we retry?
0484| 30BA  BMI     DW_ERR1
0486| 4C E303 - JMP     WPAR_RETRY

PAGE -  23  PROFILE   FILE: PROFILE.B.TEXT       SOS Profile Driver -- Version 1.30  14-Jan-83

0489| 5 .PAGE
0489| 5  ;---------------------------------------------------------------------
0489| 5  ;
0489| 5  ;  Profile Block I/O transfer routine.  This routine will transfer 512
0489| 5  ;  bytes to/from users buffer from/to RAM buffer of Profile's Z8.  It
0489| 5  ;  uses both byte at a time and psuedo DMA as necessary for the fastest
0489| 5  ;  possible transfer rate.  If users buffer is on a page boundary, only
0489| 5  ;  the psuedo DMA is used.
0489| 5  ;
0489| 5  ;  Because of the requirements of the psuedo DMA, a small routine is
0489| 5  ;  relocated to page $18 to swap banks before transfer. This is done
0489| 5  ;  only once per call to the driver.
0489| 5  ;
0489| 5 ;  NOTE: this routine is designed to transfer no less than 512 bytes.
0489| 5  ;---------------------------------------------------------------------
0489| 5
0489| 08 2 DATRANS         PHP
048A| 78 2 SEI
048B| A9 02 /                 LDA     #2              ;Always move 512 bytes at a time
048D| 8D 7900 - STA     CNTHI
0490| A9 00 / LDA     #0
0492| 8D 7800 - STA     CNTLO
0495| 2C 7B00 - BIT     DATDIR          ;Write or read?
0498| 30**  BMI     $010            ;Branch if write
049A| 20 **** - JSR     SETUPREAD
049D| 4C **** - JMP     $020            ;Branch always taken.
04A0| 5
04A0| 20 **** - $010            JSR     SETUPWRITE
04A3| AD DFFF - $020            LDA     E_REG
04A6| 29 7F / AND     #7F             ;and out 1MHz bit
04A8| 8D DFFF - STA     E_REG
04AB| 28 2 PLP
04AC| A5 CA / LDA     ADRLO           ;Is transfer on page boundary?
04AE| F0**  BEQ     DATADMA         ;Yes, Do it fast!!
04B0| C9 F5 / CMP     #0F5            ;Should we bother with DMA?
04B2| 90**  BCC     $030            ;Branch if DMA will be faster.
04B4| 49 FF / EOR     #0FF            ;Set up for move count
04B6| 20 **** - JSR     MOVIT
04B9| 4C **** - JMP     FASTMOV         ;Done with first (partial) page.
04BC| 29 01 / $030             AND     #1              ;Is it a 2 byte boundary?
04BE| F0**  BEQ     $040            ;Yes, move first partial page.
04C0| A9 00 / LDA     #0              ;Otherwise move a byte
04C2| 8D 7700 - STA     MVCNT
04C5| 20 **** - JSR     MOVE            ; to get things aligned.
04C8| A5 CA /  $040            LDA     ADRLO
04CA| AA 2  DATADMA         TAX
04CB| 18 2 CLC                     ;Round up to next 2 byte boundary
04CC| 69 01 / ADC     #1
04CE| 29 FE / AND     #0FE
04D0| 8D F718 - STA     VECTLO          ;Store as low ROM entry point.
04D3| 8A 2 TXA
04D4| 49 FF / EOR     #0FF
04D6| 8D 7700 - STA     MVCNT           ;Save number of bytes (-1) moved.
04D9| 8D 9100 - STA     DMA_CNT         ;(Guaranteed to be enough for DMA to
```


```
04DC| 20 **** - JSR     GO_DMA          ; the end of a page)
04DF| 20 **** -                 JSR     ADJ_ADR         ;Update address and count
04E2| AD 7900 - LDA     CNTHI
04E5| F0**  BEQ     LAST_PGE        ;Branch if less than 1 page to move.
04E7| A9 00 /  FASTMOV         LDA     #0
04E9| F0DF  BEQ     DATADMA         ;Branch ALWAYS to move next page.
```



```
04EB| 5 .PAGE
04EB| A0 00 /  TRANSDNE        LDY     #0
04ED| AE 6900 - LDX     SLOTX
04F0| 08 2 PHP
04F1| 78 2 SEI
04F2| AD DFFF - LDA     E_REG
04F5| 09 80 / ORA     #80             ;or in 1MHz bit
04F7| 8D DFFF - STA     E_REG
04FA| 28 2 PLP
04FB| 2C 7B00 - BIT     DATDIR
04FE| 10**  BPL     $300
0500| B9 8A00 - $100             LDA     CHEKBYTS,Y
0503| 9D 80C0 - STA     WR_PORT,X
0506| C8 2 INY
0507| C0 06 / CPY     #6
0509| D0F5  BNE     $100
050B| 20 **** - $300             JSR     CHKPARITY       ;Test for parity error in transfer.
050E| 08 2 $999             PHP
050F| 78 2 SEI
0510| 20 **** - JSR     SETUPREAD       ;restore read state
0513| 20 4101 -                   JSR     S2M             ;back to 2 MHz
0516| 28 2 PLP
0517| 60 2 RTS
0518| 5
0518| AD 7800 -  LAST_PGE        LDA     CNTLO           ;Anything left to move?
051B| F0CE  BEQ     TRANSDNE        ;Branch if not.
051D| 38 2 SEC
051E| E9 01 / SBC     #1              ;Note: low buffer address is always 0.
0520| C9 20 / CMP     #20             ;Is there more than 32 bytes left?
0522| B0**  BCS     $20             ;If not, move last a byte at a time.
0524| 20 **** - JSR     MOVIT
0527| 4C EB04 - JMP     TRANSDNE
052A| 5
052A| 48 2 $20             PHA
052B| A9 01 / LDA     #1              ;Move first 2 bytes to preserve byte 0.
052D| 8D 7700 - STA     MVCNT
0530| 20 **** - JSR     MOVE            ;The call to ADJ_ADR is below.
0533| A0 00 / LDY     #0              ;Bytes 0 and FE of the current page
0535| 08 2 PHP
0536| 78 2 SEI                    ;disallow interupts
0537| B1 CA / LDA     (ADRLO),Y       ;  must be preserved because of the
0539| 8D 7500 - STA     TEMP00          ;  quirks of the psuedo DMA while doing
053C| A0 FE / LDY     #0FE            ;  a partial page transfers.  Since the
053E| B1 CA / LDA     (ADRLO),Y       ;  branch instuction generates a false
0540| 8D 7600 - STA     TEMPFE          ;  address within the DMA page, byte 0
0543| 68 2 PLA                    ;  is accessed if more than $80 bytes
0544| AA 2 TAX                    ; (save status)
0545| 68 2 PLA
0546| 29 FC / AND     #0FC            ;  are transfered and byte FE is ac-
0548| C9 84 / CMP     #84             ;  cessed if less than $80.
054A| D0**  BNE     $30
054C| E9 04 / SBC     #04             ;Transfers of exactly $80 (82) are not
054E| A8 2  $30              TAY                     ;  allowed do to DMA code (see Apple 3
054F| 4A 2 LSR     A               ;  monitor listing of psuedo DMA code)
```



```
0550| 4A 2 LSR     A
0551| E9 01 / SBC     #1
0553| 8D 9100 - STA     DMA_CNT         ;Set up for exit of DMA
0556| 8A 2 TXA
0557| 48 2 PHA                    ;restore status to stack
0558| 98 2 TYA
0559| 38 2 SEC
055A| E9 03 / SBC     #3
055C| 8D 7700 - STA     MVCNT           ;Save bytes total bytes transferred.
055F| A8 2 TAY
0560| A9 02 / LDA     #02             ;Set low entry point for DMA routine.
0562| 8D F718 - STA     VECTLO
0565| 2C 7B00 - BIT     DATDIR          ;Read or write?
0568| 10**  BPL     $50             ;Branch if read.
056A| B1 CA / LDA     (ADRLO),Y       ;Get last byte and move to both
056C| A0 00 / LDY     #0              ;  byte 0 and FE.
056E| 91 CA / STA     (ADRLO),Y
0570| A0 FE / LDY     #0FE
0572| 91 CA / STA     (ADRLO),Y
0574| 20 **** -  $50              JSR     GO_DMA          ;Now transfer the partial page.
0577| 2C 7B00 - BIT     DATDIR          ;Was it read or write?
057A| 30**  BMI     $70             ;Branch if write.
057C| A0 00 / LDY     #0
057E| 2C 7700 - BIT     MVCNT           ;Now figer out where the last byte
0581| 30**  BMI     $60             ;  got red.
```

```
0583| A0 FE / LDY     #0FE            ;Must have been Read into FE.
0585| B1 CA /  $60             LDA     (ADRLO),Y
0587| AC 7700 - LDY     MVCNT           ;Put it where it belongs!
058A| 91 CA / STA     (ADRLO),Y
058C| A0 00 /  $70             LDY     #0              ;Now restore bytes 0 & FE.
058E| AD 7500 - LDA     TEMP00
0591| 91 CA / STA     (ADRLO),Y
0593| A0 FE / LDY     #0FE
0595| AD 7600 - LDA     TEMPFE
0598| 91 CA / STA     (ADRLO),Y
059A| 28 2 PLP                      ;Interupts OK now.
059B| 20 **** - JSR     ADJ_ADR         ;Go fix addresses and count.
059E| 4C 1805 - JMP     LAST_PGE
05A1| 5
05A1| 8D 7700 -  MOVIT           STA     MVCNT           ;Number of bytes for transfer.
05A4| 20 **** - JSR     MOVE            ;Do byte at a time.

PAGE -  27  PROFILE   FILE: PROFILE.B.TEXT       SOS Profile Driver -- Version 1.30  14-Jan-83

05A7| 5 .PAGE
05A7| 5  ;---------------------------------------------------------------------
05A7| 5  ;
05A7| 5  ;  This routine adjusts the count and addresses (indirect and bank
05A7| 5  ;  direct) after each transfer, either byte at a time or psuedo DMA.
05A7| 5  ;
05A7| 5  ;  Input is MVCNT.  Only the processor status and accumulator are used.
05A7| 5  ;  ADRLO, ADRHI, ADDRDMA, BANKDMA are all assumed to be valid, thus a
05A7| 5  ;  call to ARBADR should have taken place before this routine is used.
05A7| 5  ;  CNTLO, CNTHI should never be less than MVCNT, as no checking is done.
05A7| 5  ;
05A7| 5  ;---------------------------------------------------------------------
05A7| 5
05A7| 18 2  ADJ_ADR         CLC
05A8| AD 7800 - LDA     CNTLO           ;Adjust count and Addresses
05AB| ED 7700 - SBC     MVCNT           ;Subtract (MVCNT+1) from count
05AE| 8D 7800 - STA     CNTLO           ;  and add (MVCNT+1) to address.
05B1| AD 7900 - LDA     CNTHI
05B4| E9 00 / SBC     #0
05B6| 8D 7900 - STA     CNTHI
05B9| A5 CA / LDA     ADRLO
05BB| 6D 7700 - ADC     MVCNT           ;(carry was set)
05BE| 85 CA /                   STA     ADRLO
05C0| A5 CB / LDA     ADRHI
05C2| 69 00 / ADC     #0
05C4| 85 CB / STA     ADRHI
05C6| 2C 7A00 - BIT     ADMODE          ;Are Bank Wraps possible?
05C9| 10**  BPL     ADJ_DNE         ;Branch if not.
05CB| C9 81 / CMP     #081            ;Time to Adjust for Wrap?
05CD| 90**  BCC     ADJ_BNK         ;Branch if not.
05CF| 29 7F / AND     #07F            ;Otherwise strip hi bit, add 1 to
05D1| EE CB14 - INC     SISADR          ;  bank pair selected indirect
05D4| 85 CB /  ADJ_BNK         STA     ADRHI
05D6| AD CB14 - LDA     SISADR
05D9| 8D 7400 - STA     BANKDMA
05DC| A5 CB / LDA     ADRHI
05DE| 18 2 CLC
05DF| 69 20 / ADC     #20             ;Add $20 for bank address equiv.
05E1| C9 A0 / CMP     #0A0            ;Next bank?
05E3| 90**  BCC     ADJ_DNE         ;Branch if not.
05E5| 29 7F / AND     #07F            ;Address range must be within 20-9F
05E7| EE 7400 - INC     BANKDMA
05EA| 8D 7300 -  ADJ_DNE          STA     ADDRDMA         ;Save absolute address for DMA
05ED| 60 2 RTS

PAGE -  28  PROFILE   FILE: PROFILE.B.TEXT       SOS Profile Driver -- Version 1.30  14-Jan-83

05EE| 5 .PAGE
05EE| 5  ;---------------------------------------------------------------------
05EE| 5  ;
05EE| 5  ;  The move routines read or write data byte at a time.  Input varibles
05EE| 5  ;  are ADRLO, ADRHI and MVCNT, none of which are modified (that is done
05EE| 5  ;  by ADJADR).  All registers are used.  Also SLOTX must contain the
05EE| 5  ;  slot number times 16 for indexing the device locations for read and
05EE| 5  ;  write.  DATDIR indicates the direction (Read=0, Write=FF) of transfer.
05EE| 5  ;
05EE| 5  ;---------------------------------------------------------------------
05EE| 5
05EE| 08 2  MOVE            PHP
05EF| 78 2 SEI
05F0| AE 6900 -                 LDX     SLOTX           ;Get index to device locations.
05F3| A0 00 / LDY     #0
05F5| AD DFFF -                 LDA     E_REG
05F8| 09 80 / ORA     #80             ;or in 1 MHz bit
05FA| 8D DFFF - STA     E_REG
05FD| 2C 7B00 - BIT     DATDIR          ;Read or write?
0600| 10**  BPL     MOVIN           ;Branch if read.
0602| 5
0602| B1 CA /  MOVOUT          LDA     (ADRLO),Y       ;Move data to Z8 RAM
0604| 9D 80C0 - STA     WR_PORT,X
0607| CC 7700 - CPY     MVCNT           ;Done with Write?
060A| F0**  BEQ     MVDONE          ;Branch if done.
060C| C8 2 INY
060D| D0F3  BNE     MOVOUT          ;Branch always taken.
```

```
060F| 5
060F| BD 81C0 -  MOVIN           LDA     RD_PORT,X       ;Get data from Z8 RAM
0612| 91 CA / STA     (ADRLO),Y
0614| CC 7700 - CPY     MVCNT
0617| F0**  BEQ     MVDONE
0619| C8 2 INY
061A| D0F3  BNE     MOVIN           ;Branch always.
061C| 5
061C| 20 4101 -  MVDONE          JSR     S2M             ;back to 2 MHz
061F| 28 2 PLP
0620| 60 2                 RTS
```



```
0621| 5 .PAGE
0621| 5  ;---------------------------------------------------------------------
0621| 5  ;
0621| 5  ;  This routine arbitrates the initial user's address in terms of both
0621| 5  ;  indirect and absolute bank address.  It sets the initial states for
0621| 5  ;  ADRLO, ADRHI, ADDRDMA, BANKDMA, and ADMODE.  ADMODE's bit 7 is set
0621| 5  ;  if extented addressing is used, otherwise it is reset (0).  ADDRDMA
0621| 5  ;  is the page address resulting within the 6502 address space,  the
0621| 5  ;  low address is always the same as ADRLO.
0621| 5  ;
0621| 5  ;---------------------------------------------------------------------
0621| 5
0621| A5 C2 /  ARBADR          LDA     SOS_BUF         ;Get strait indirect address moved
0623| 85 CA / STA     ADRLO
0625| 8D 6F00 - STA     ORGADR
0628| A5 C3 / LDA     SOS_BUF+1
062A| 85 CB / STA     ADRHI
062C| 8D 7000 - STA     ORGADR+1
062F| 8D 7300 - STA     ADDRDMA
0632| AD C314 - LDA     SOS_XPAGE+SOS_BUF+1
0635| 8D CB14 - STA     SISADR
0638| 8D 7100 - STA     ORGBNK
063B| 8D 7A00 - STA     ADMODE
063E| 10**  BPL     NOBANK          ;Branch if load into current bank.
0640| 29 0F / AND     #0F
0642| 8D 7400 - STA     BANKDMA
0645| 49 0F / EOR     #0F             ;Now test for special bank (F)
0647| F0**  BEQ     BANK0
0649| A5 C3 / LDA     SOS_BUF+1
064B| 10**  BPL     $010
064D| 29 7F / AND     #7F             ;Subtract $80 from high byte
064F| EE 7400 - INC     BANKDMA         ;  and increment bank number
0652| 18 2 $010            CLC
0653| 69 20 / ADC     #20             ;Convert to absolute address
0655| 8D 7300 - STA     ADDRDMA
0658| 60 2 RTS
0659| 5
0659| AD EFFF -  NOBANK          LDA     BANK_REG        ;Use current bank as bank select
065C| 29 0F / AND     #0F             ;  for DMA purposes.
065E| 8D 7400 -  BANK0           STA     BANKDMA
0661| 8D 7A00 - STA     ADMODE          ;Clear bit 7 of mode to indicate no
0664| 60 2 RTS                     ;  bank wrapping required.
```



```
0665| 5 .PAGE
0665| 5  ;---------------------------------------------------------------------
0665| 5  ;
0665| 5  ;  In order to perform psuedo DMA, the call to ROM must be outside the
0665| 5  ;  Bank.  This routine sets up the Z_REG and loads the X register with
0665| 5  ;  the bank desired before calling DO_DMA which has been relocated to
0665| 5  ;  page 2.  It also disables interupts for the duration (up to 256 usec)
0665| 5  ;  of the transfer.
0665| 5  ;
0665| 5  ;---------------------------------------------------------------------
0665| 5
0665| 08 2  GO_DMA          PHP                     ;Save interupt status
0666| 78 2 SEI                    ;No interupts for now.
0667| AD DFFF - LDA     E_REG           ;And no NMI's either!!!
066A| 48 2 PHA
066B| 29 24 / AND     #24             ;Switch out I/O also.
066D| 09 8B / ORA     #8B             ;And write protect upper 16K!
066F| 8D DFFF - STA     E_REG
0672| AD 7300 - LDA     ADDRDMA         ;Set Z_REG to DMA page address
0675| 8D D0FF - STA     Z_REG
0678| AE 7400 - LDX     BANKDMA
067B| AD 9100 - LDA     DMA_CNT
067E| 38 2 SEC
067F| 20 F018 - JSR     DO_DMA
0682| 68 2 PLA                    ;Restore NMI state.
0683| 8D DFFF - STA     E_REG
0686| A9 18 / LDA     #SOS_ZPAGE      ;Restore proper zero page.
0688| 8D D0FF - STA     Z_REG
068B| 28 2 PLP                    ;Restore interupts
068C| 60 2 RTS
```



```
068D| 5 .PAGE
```

```
068D| 5  ;---------------------------------------------------------------------
068D| 5 ;
068D| 5 ;  The following are routines for handling the communications protocol
068D| 5 ;  of sending commands, and receiving result codes.
068D| 5 ;
068D| 5 ;---------------------------------------------------------------------
068D| 5  ;
068D| 5  ;  SNDCMD performs a CMD-BSY handshake with the Z8 and checks for a
068D| 5  ;  correct response.  If the Z8 responds with an incorrect code, a
068D| 5  ;  'no go' code is sent by the Apple and the handshake is retried
068D| 5  ;  up to 2 times.  On return, Carry=1 means a handshake timeout or
068D| 5  ;  three retries attempted.  A non-zero return means an incorrect
068D| 5  ;  response from the Z8 that may be retried.
068D| 5  ;
068D| 5  ;---------------------------------------------------------------------
068D| 5
068D| 08 2  SNDCMD          PHP
068E| 78 2 SEI
068F| AD DFFF - LDA     E_REG
0692| 09 80 / ORA     #80             ;or in 1 MHz bit
0694| 8D DFFF - STA     E_REG
0697| 28 2 PLP
0698| 20 **** - JSR     WAITBSYLO
069B| B0**  BCS     SENDERR         ;error exit if BSY isn't low
069D| 20 ****  -                JSR     SETCMDLN        ;raise cmd
06A0| 20 **** - JSR     WAITBSYHI       ;wait for bsy to go hi
06A3| B0**  BCS     SENDERR         ;timeout
06A5| BC 81C0 - LDY     RD_PORT,X       ;read response byte from Z8
06A8| C4 D2 / CPY     RSPNS           ;correct?
06AA| F0**  BEQ     CONT            ;yes if taken
06AC| A9 02 / LDA     #2
06AE| 8D 8900 - STA     BAD_RESPONSE
06B1| 0D 8500 - ORA     STATUS3
06B4| 8D 8500 - STA     STATUS3
06B7| A9 AA / LDA     #0AA            ;tell Z8 that response not OK
06B9| 20 **** - JSR     BSYACK          ;drop cmd, wait for bsy to go lo
06BC| B0**  BCS     SENDERR         ;timeout on bsy going lo
06BE| E6 D1 / INC     CMD_RTRYCNT     ;bump retry count for bad response
06C0| A4 D1 / LDY     CMD_RTRYCNT
06C2| C0 02 / CPY     #2              ;2 retries yet?
06C4| B0**  BCS     SENDERR         ;yes if taken
06C6| 60 2 RTS
06C7| 5
06C7| A9 55 /  CONT            LDA     #055            ;indicate good response
06C9| 20 **** - JSR     BSYACK
06CC| B0**  BCS     SENDERR         ;bsy timeout
06CE| A9 00 / LDA     #0              ;indicate good return to caller
06D0| 60 2  SENDERR         RTS
06D1| 5
06D1| 5
06D1| 5
06D1| A0 00 /  WAITBSYHI       LDY     #0              ;set .5sec timeout
06D3| 84 CE / STY     TIMOUT

PAGE -  32  PROFILE   FILE: PROFILE.B.TEXT       SOS Profile Driver -- Version 1.30  14-Jan-83

06D5| 18 2 CLC
06D6| AE 6900 - LDX     SLOTX
06D9| BD 82C0 -  ALOOP           LDA     BUSY,X
06DC| 10**  BPL     BSYHIRET        ;done if taken
06DE| 88 2 DEY
06DF| D0F8  BNE     ALOOP
06E1| C6 CE / DEC     TIMOUT
06E3| D0F4  BNE     ALOOP
06E5| 38 2 SEC                     ;timeout
06E6| 60 2 BSYHIRET        RTS
06E7| 5
06E7| A0 01 /  WAITBSYLO       LDY     #1
06E9| AD 8000 - LDA     LONGWAIT
06EC| F0**  BEQ     $10
06EE| A0 10 / LDY     #10             ;set up for 8 second wait max.
06F0| 8C 8100 - $10              STY     WAITTIME
06F3| A0 00 / LDY     #0              ;set .5sec timeout
06F5| 84 CE / STY     TIMOUT
06F7| 18 2 CLC
06F8| AE 6900 - LDX     SLOTX
06FB| BD 82C0 -  BLOOP           LDA     BUSY,X
06FE| 30**  BMI     BSYLORET        ;done if taken
0700| 88 2 DEY
0701| D0F8  BNE     BLOOP
0703| C6 CE / DEC     TIMOUT
0705| D0F4  BNE     BLOOP
0707| CE 8100 - DEC     WAITTIME
070A| D0EF  BNE     BLOOP
070C| 38 2 SEC                     ;timeout
070D| 60 2  BSYLORET        RTS
070E| 5
070E| 5  ;---------------------------------------------------------------------
070E| 5 ;
070E| 5  ;  SND_CMDBYTES sends the command string to widget.
070E| 5 ;  Enter with cmd=bsy=lo.  Error return if get parity error - Carry = 1
070E| 5  ;
070E| 5 ;---------------------------------------------------------------------
```

```
070E| 5
070E| 20 **** -  SND_CMDBYTES    JSR     SETUPWRITE      ;get in proper state
0711| AD 7C00 - LDA     Z8CMD           ;send command string - cmd, blockhi
0714| 9D 80C0 - STA     WR_PORT,X       ;blocklo, retries, retry threshold
0717| AD 7E00 - LDA     MSBLOCK
071A| 9D 80C0 - STA     WR_PORT,X
071D| A5 D5 / LDA     BLOCKHI
071F| 9D 80C0 - STA     WR_PORT,X
0722| A5 D4 / LDA     BLOCKLO
0724| 9D 80C0 - STA     WR_PORT,X
0727| A5 CF / LDA     RTRYCNT
0729| 9D 80C0 - STA     WR_PORT,X
072C| A5 D0 / LDA     RTRYTHRESH
072E| 9D 80C0 - STA     WR_PORT,X
0731| 20 **** - JSR     SETUPREAD       ;finish writing last byte and
0734| 20 **** - JSR     CHKPARITY       ;check for parity error
0737| 60 2 RTS

PAGE -  33  PROFILE   FILE: PROFILE.B.TEXT       SOS Profile Driver -- Version 1.30  14-Jan-83

0738| 5 .PAGE
0738| 5 ;---------------------------------------------------------------------
0738| 5 ;
0738| 5  ;GETSTAT retrieves the status bytes from widget.  The one-byte
0738| 5  ;result code is returned in Y.
0738| 5 ;
0738| 5 ;---------------------------------------------------------------------
0738| 5
0738| AE 6900 -  GETSTAT         LDX     SLOTX           ;get slot #
073B| 08 2 PHP
073C| 78 2 SEI
073D| AD DFFF - LDA     E_REG
0740| 09 80 / ORA     #80             ;or in 1 MHz bit
0742| 8D DFFF - STA     E_REG
0745| 28 2 PLP
0746| BD 81C0 - LDA     RD_PORT,X
0749| 8D 8300 - STA     STATUS1
074C| BD 81C0 - LDA     RD_PORT,X
074F| 8D 8400 - STA     STATUS2
0752| BD 81C0 - LDA     RD_PORT,X
0755| 48 2 PHA
0756| BD 81C0 - LDA     RD_PORT,X
0759| 8D 8600 - STA     STATUS4
075C| 20 **** - JSR     CHKPARITY
075F| 68 2 PLA
0760| 0D 8700 - ORA     PARITY_ERR
0763| 0D 8900 - ORA     BAD_RESPONSE
0766| 0D 8800 - ORA     RESET_FLAG
0769| 8D 8500 - STA     STATUS3
076C| AD 8300 - LDA     STATUS1
076F| 60 2 RTS
0770| 5
0770| 5 ;---------------------------------------------------------------------
0770| 5  ;
0770| 5  ;  CHKPARITY checks the parity error line and shifts it into Carry, so
0770| 5  ;  Carry = 1 is a parity error on return to caller.
0770| 5  ;
0770| 5 ;---------------------------------------------------------------------
0770| 5
0770| AE 6900 -  CHKPARITY       LDX     SLOTX
0773| BD 82C0 -                  LDA     BUSY,X          ;get parity error - on bit 6
0776| 9D 83C0 - STA     CLR_PARITY,X    ;clear it for next transfer
0779| 0A 2 ASL     A
077A| 0A 2 ASL     A               ;shift it into carry
077B| 90**  BCC     $10
077D| A9 01 / LDA     #1
077F| 8D 8700 - STA     PARITY_ERR
0782| 0D 8500 - ORA     STATUS3
0785| 8D 8500 - STA     STATUS3
0788| 4C 4101 - $10             JMP     S2M             ;exit via setting 2 MHz mode

PAGE -  34  PROFILE   FILE: PROFILE.B.TEXT       SOS Profile Driver -- Version 1.30  14-Jan-83

078B| 5 .PAGE
078B| 5 ;---------------------------------------------------------------------
078B| 5  ;
078B| 5  ;  SETUPWRITE sets CRW and DATRW lo on the Apple /// interface board
078B| 5  ;  to prepare for a write operation to widget
078B| 5 ;
078B| 5  ;---------------------------------------------------------------------
078B| 5
078B| A0 01 /  SETUPWRITE      LDY     #SETWRT
078D| 08 2 PHP
078E| 78 2 SEI
078F| AD DFFF -                  LDA     E_REG
0792| 09 80 / ORA     #80             ;or in 1 MHz bit
0794| 8D DFFF - STA     E_REG
0797| 28 2 PLP
0798| B1 CC / LDA     (INDRCN),Y      ;set crw lo
079A| A0 03 /  SET_WRITEDIR    LDY     #RWLO
079C| B1 CC / LDA     (INDRCN),Y      ;set datarw lo
079E| 60 2 RTS
079F| 5
```

```
079F| 5
079F| A0 05 /  SETUPREAD       LDY     #SETRD
07A1| 08 2 PHP
07A2| 78 2 SEI
07A3| AD DFFF - LDA     E_REG
07A6| 09 80 / ORA     #80             ;or in 1MHz bit
07A8| 8D DFFF - STA     E_REG
07AB| 28 2 PLP
07AC| B1 CC / LDA     (INDRCN),Y      ;set crw hi
07AE| A0 07 / LDY     #RWHI
07B0| B1 CC / LDA     (INDRCN),Y      ;set datarw hi
07B2| 60 2 RTS
```



```
07B3| 5 .PAGE
07B3| 5 ;---------------------------------------------------------------------
07B3| 5 ;
07B3| 5  ;  BSYACK completes the cmd-bsy handshake by outputting the response
07B3| 5  ;  byte to widget, dropping cmd, and waiting for bsy to go lo.
07B3| 5  ;  Enter with the widget response ($55 or $AA) in A.
07B3| 5 ;
07B3| 5  ;---------------------------------------------------------------------
07B3| 5
07B3| 9D 80C0 -  BSYACK          STA     WR_PORT,X       ;store response byte
07B6| 20 9A07 - JSR     SET_WRITEDIR    ;enable bus out to widget
07B9| 20 **** - JSR     NTCMDLN1        ;drop cmd
07BC| 20 E706 - JSR     WAITBSYLO
07BF| 20 9F07 - JSR     SETUPREAD       ;restore read state
07C2| 4C 4101 - JMP     S2M             ;exit via setting 2 MHz mode
07C5| 5
07C5| 5
07C5| 20 9F07 - NOTCMDLN        JSR     SETUPREAD
07C8| A0 00 / NTCMDLN1        LDY     #NOTCMD
07CA| B1 CC / LDA     (INDRCN),Y
07CC| 60 2 RTS
07CD| 5
07CD| 20 9F07 - SETCMDLN        JSR     SETUPREAD
07D0| A0 04 /                 LDY     #SETCMD
07D2| B1 CC / LDA     (INDRCN),Y
07D4| 60 2 RTS
07D5| 5
07D5| A9 04 / RESET_PIPPIN    LDA     #4
07D7| 8D 8800 - STA     RESET_FLAG
07DA| 0D 8500 - ORA     STATUS3
07DD| 8D 8500 - STA     STATUS3
07E0| A0 0C /                 LDY     #RST
07E2| 08 2 PHP
07E3| 78 2 SEI
07E4| AD DFFF - LDA     E_REG
07E7| 09 80 / ORA     #80             ;or in 1 MHz bit
07E9| 8D DFFF - STA     E_REG
07EC| 28 2 PLP
07ED| B1 CC / LDA     (INDRCN),Y
07EF| A0 25 /                 LDY     #25
07F1| 88 2 $10            DEY
07F2| D0FD  BNE     $10
07F4| A0 08 / LDY     #CLRRST
07F6| B1 CC / LDA     (INDRCN),Y      ;clear reset
07F8| 4C 4101 - JMP     S2M             ;exit via setting 2 MHz mode
07FB| 5
07FB| 5 .END
```


AB - Absolute LB - Label UD - Undefined MC - Macro
RF - Ref DF - Def PR - Proc FC - Func
PB - Public PV - Private CS - Consts

ADDRDMA	LB 0073	ADJADR	LB 05A7	ADJBNK	LB 05D4	ADJDNE	LB 05EA	ADMODE	LB 007A	ADRHI	AB 00CB	ADRLO	AB 00CA
ALLOCSIR	AB 1913	ALOOP	LB 06D9	ARBADR	LB 0621	BADEXIT	LB 03AF	BADOLDDA	AB 0027	BADOP	LB 0163	BADREQ	LB 015E
BADRESPO	LB 0089	BANK0	LB 065E	BANKDMA	LB 0074	BANKREG	AB FFEF	BLKRTRYC	AB 00D3	BLOCKHI	AB 00D5	BLOCKLO	AB 00D4
BLOOP	LB 06FB	BSYACK	LB 07B3	BSYHIRET	LB 06E6	BSYLO	LB 007F	BSYLORET	LB 070D	BUSY	AB C082	CALCHEK	LB 02AA
CDELEN	AB 000D	CHEKBYTS	LB 008A	CHKPARIT	LB 0770	CLRPARIT	AB C083	CLRRST	AB 0008	CMDRTRYC	AB 00D1	CNTHI	LB 0079
CNTLO	LB 0078	CONT	LB 06C7	COUNTR	LB 007D	CSYSER1	LB 0454	DATADMA	LB 04CA	DATDIR	LB 007B	DATRANS	LB 0489
DEALCSIR	AB 1916	DEVTYPE	AB 00D1	DIBBLOCK	LB 0059	DIBDCBCN	LB 005F	DIBDNUM1	LB 0053	DIBENTRY	LB 0041	DIBLINK1	LB 003F
DIBMID1	LB 005B	DIBNAME1	LB 0043	DIBRLS1	LB 005D	DIBSLOT1	LB 0054	DIBSUBTY	LB 0057	DIBTYPE1	LB 0056	DIBUNIT1	LB 0055
DIERR1	LB 01CD	DIERR2	LB 01D1	DIERR3	LB 01D3	DINIT	LB 0184	DISPATCH	LB 014D	DMACNT	LB 0091	DODMA	AB 18F0
DREAD	LB 02C7	DREPEAT	LB 022E	DRERR1	LB 0364	DSTATUS	LB 01D9	DWERR1	LB 0440	DWERR2	LB 0457	DWEXIT	LB 047C
DWRITE	LB 03CA	EREG	AB FFDF	ERROR	LB 006A	FASTMOV	LB 04E7	FCODE	LB 021F	FDCONTRO	LB 0224	FORMATUS	LB 020E
GETBYTES	LB 0269	GETSTAT	LB 0738	GODMA	LB 0665	GOINIT	LB 0127	GOODEXIT	LB 0396	IERROR	LB 006B	INDDMA	AB 00F0
INDRCN	AB 00CC	INITENT	LB 00C0	INTDSABL	AB 0002	INTENABL	AB 0006	LASTPGE	LB 0518	LDAXIO	LB 0376	LDAXIOER	LB 0452
LENGTH	LB 006E	LONGWAIT	LB 0080	MAIN	LB 009F	MANUF	AB 0001	MAXBLOCK	AB 2600	MOVCODE	LB 00CD	MOVE	LB 05EE
MOVIN	LB 060F	MOVIT	LB 05A1	MOVOUT	LB 0602	MSBLOCK	LB 007E	MVCNT	LB 0077	MVDONE	LB 061C	NOBANK	LB 0659
NODRV	LB 0168	NOTCMD	AB 0000	NOTCMDLN	LB 07C5	NTCMDLN1	LB 07C8	ONEMEG	LB 00C3	ORGADR	LB 006F	ORGBNK	LB 0071
PARITYER	LB 0087	PIPPINRE	LB 0082	PREVCMD	LB 006C	PREVUNIT	LB 006D	PROFILE	PR ----	PSUEDODM	AB F800	RCOMRETR	LB 02FC
RDBLOCK	LB 02DF	RDPORT	AB C081	RDRETRY	LB 035A	RELCODE	LB 0092	RELEASE	AB 1300	RESETFLA	LB 0088	RESETPIP	LB 07D5
RPARRETR	LB 02E3	RSPNS	AB 00D2	RST	AB 000C	RSTORADR	LB 03B7	RSTORENV	LB 0132	RTRYCNT	AB 00CF	RTRYTHRE	AB 00D0
RWHI	AB 0007	RWLO	AB 0003	S2M	LB 0141	SELC800	AB 1922	SENDERR	LB 06D0	SETCMD	AB 0004	SETCMDLN	LB 07CD
SETRD	AB 0005	SETUPREA	LB 079F	SETUPWRI	LB 078B	SETWRITE	LB 079A	SETWRT	AB 0001	SIRADDR	LB 0062	SIRBANK	LB 0068
SIRCOUNT	AB 0005	SIRTABLE	LB 0064	SISADR	AB 14CB	SISCN	AB 14CD	SLOTCN	LB 0072	SLOTX	LB 0069	SNDCMD	LB 068D
SNDCMDBY	LB 070E	SOSBLOCK	AB 00C6	SOSBUF	AB 00C2	SOSBYTES	AB 00C4	SOSBYTRD	AB 00C8	SOSREQCO	AB 00C0	SOSSTCOD	AB 00C2
SOSSTLIS	AB 00C3	SOSUNIT	AB 00C1	SOSXPAGE	AB 1400	SOSZPAGE	AB 0018	SPTBLOVF	AB 0027	STATUS1	LB 0083	STATUS2	LB 0084
STATUS3	LB 0085	STATUS4	LB 0086	SUBTYPE	AB 0001	SVENV	LB 0090	SWITCH	MC ----	SWTABLE	LB 0170	SYSERR	AB 1928
SYSERROR	LB 016D	TEMP00	LB 0075	TEMPFE	LB 0076	TIMOUT	AB 00CE	TRANSDNE	LB 04EB	TSTBLKNU	LB 0249	TSTMORE	LB 037A
VECTLO	AB 18F7	WAITBSYH	LB 06D1	WAITBSYL	LB 06E7	WAITTIME	LB 0081	WCOMRETR	LB 03F3	WCONT	LB 045B	WDGTRD	AB 0000
WDGTSTAT	AB 0003	WDGTWRT	AB 0001	WDGTWRTV	AB 0002	WPARRETR	LB 03E3	WRBLOCK	LB 03DF	WRPORT	AB C080	WRRETRY	LB 047F
WRTVER	LB 0061	XBADOP	AB 0026	XBLKNUM	AB 002D	XBYTECNT	AB 002C	XCTLCODE	AB 0021	XIOERROR	AB 0027	XNODRIVE	AB 0028
XNORESRC	AB 0025	XNOWRITE	AB 002B	XREQCODE	AB 0020	Z8CMD	LB 007C	ZREG	AB FFD0				

```
Current minimum space is 21598 words.

Assembly complete:      1453 lines
     0   Errors flagged on this Assembly

###
```