Atari 800 OS Source Listing#

```
LIST X
; THIS IS THE MODIFIED SEPTEMBER ATARI 400/800 COMPUTER OPERATING
; SYSTEM LISTING. MODIFIED TO ASSEMBLE ON THE MICROTEC CROSS
; ASSEMBLER.
; THIS VERSION IS THE ONE WHICH WAS BURNED INTO ROM.
; THERE IS A RESIDUAL PIECE OF CODE WHICH IS FOR LNBUG. THIS
; IS AT LOCATION $9000 WHICH IS NOT IN ROM.
; THIS IS THE REVISION B EPROM VERSION
          .PAGE
;
        COLLEEN OPERATING SYSTEM EQUATE FILE
        NTSC/PAL ASSEMBLY FLAG
PALFLG = 0
                       ;0 = NTSC 1 = PAL
        MODULE ORIGIN TABLE
                $E000
                           ; CHARACTER SET
CHRORG =
VECTBL =
                 $E400
                             ; VECTOR TABLE
                           ;RAM VECTOR INITIAL VALUE TABLE ;CENTRAL I/O HANDLER ;INTERRUPT HANDLER
VCTABL =
                 $E480
CIOORG =
                $E4A6
INTORG =
                $E6D5
SIOORG =
                $E944
                             ;SERIAL I/O DRIVER
                             ;DISK HANDLER
DSKORG =
                $EDEA
               $EE78 ; PRINTER HANDLER
$EF41 ; CASSETTE HANDLER
$F0E3 ; MONITOR/POWER UP MODULE
$F3E4 ; KEYBOARD/DISPLAY HANDLER
PRNORG =
CASORG =
MONORG =
KBDORG =
       VECTOR TABLE
; HANDLER ENTRY POINTS ARE CALLED OUT IN THE FOLLOWING VECTOR
;TABLE. THESE ARE THE ADDRESSES MINUS ONE.
; EXAMPLE FOR EDITOR
             OPEN
CLOSE
   E400
;
        2
         4
                GET
                PUT
         6
        8
                STATUS
               SPECIAL
JUMP TO POWER ON INITIALIZATION ROUTINE
NOT USED
         Α
        С
        F
EDITRV =
                $E400
                             ;EDITOR
                 $E410
SCRENV =
                             ;TELEVISION SCREEN
KEYBDV =
                 $E420
                            ;KEYBOARD
```

```
PRINTV =
                     $E430
                                   ; PRINTER
CASETV =
                     $E440
                                   ; CASSETTE
         JUMP VECTOR TABLE
;THE FOLLOWING IS A TABLE OF JUMP INSTRUCTIONS
;TO VARIOUS ENTRY POINTS IN THE OPERATING SYSTEM.
                 $E450 ;DISK INITIALIZATION
$E453 ;DISK INTERFACE
$E456 ;CENTRAL INPUT OUTPUT ROUTINE
$E459 ;SERIAL INPUT OUTPUT ROUTINE
$E45C ;SET SYSTEM TIMERS ROUTINE
$E45F ;SYSTEM VERTICAL BLANK CALCULATIONS
$E462 ;EXIT VERTICAL BLANK CALCULATIONS
$E465 ;SERIAL, INPUT OUTPUT INITIALIZATION
$E468 ;SEND ENABLE ROUTINE
$E468 ;INTERRUPT HANDLER INITIALIZATION
$E46E ;CENTRAL INPUT OUTPUT INITIALIZATION
$E471 ;BLACKBOARD MODE
$E474 ;WARM START ENTRY POINT
$E477 ;COLD START ENTRY POINT
$E477 ;CASSETTE READ BLOCK ENTRY POINT VECTOR
$E470 ;CASSETTE OPEN FOR INPUT VECTOR
DISKIW =
DSKINV =
CIOV =
SIOV
SETVBV =
SYSVBV =
XITVBV =
SIOINV =
SENDEV =
INTINV =
CIOINV =
BLKBDV =
WARMSV =
COLDSV =
RBLOKV =
CSOPIV =
; VCTABL = $E480
; OPERATING SYSTEM EQUATES
; COMMAND CODES FOR IOCB
OPEN = 3
                                   ;OPEN FOR INPUT/OUTPUT
GETREC =
                5
7
                                   GET RECORD (TEXT)
GETCHR =
                                   ;GET CHARACTER(S)
                  9
$B
$C
$D
PUTREC =
                                   ; PUT RECORD (TEXT)
                                 ; PUT CHARACTER(S)
PUTCHR =
                                 ;CLOSE DEVICE
CLOSE =
                                  ;STATUS REQUEST
STATIS =
SPECIL = $E ;BEGINNING OF SPECIAL ENTRY COMMANDS
; SPECIAL ENTRY COMMANDS
DRAWLN = $11
                                       ; DRAW LINE
                  $12
FILLIN =
                                       ;DRAW LINE WITH RIGHT FILL
                   $20
RENAME =
                                      ; RENAME DISK FILE
                  $21
$22
DELETE =
                                      ; DELETE DISK FILE
                                      ;FORMAT
FORMAT =
LOCKFL =
                   $23
                                      ;LOCK FILE TO READ ONLY
                  $24
UNLOCK =
                                      ;UNLOCK LOCKED FILE
                                     ; POINT SECTOR
POINT =
                   $25
NOTE =
                   $26
                                      ; NOTE SECTOR
IOCFRE = $FF
                                      ; IOCB "FREE"
; AUX1 EQUATES
; () INDICATES WHICH DEVICES USE BIT
APPEND =
              $1 ;OPEN FOR WRITE APPEND (D), OR SCREEN READ (
               $2
$4
DIRECT =
                                   ;OPEN FOR DIRECTORY ACCESS (D)
                                  ;OPEN FOR INPUT (ALL DEVICES)
OPNIN =
                  $8 ;OPEN FOR OUTPUT (ALL DEVICES)
OPNIN+OPNOT ;OPEN FOR INPUT AND OUTPUT (ALL DEVICES)
OPNOT =
OPNINO =
MXDMOD =
                     $10
                                      ;OPEN FOR MIXED MODE (E,S)
```

```
;OPEN WITHOUT CLEARING SCREEN (E,S)
; DEVICE NAMES
SCREDT = 'E
KBD = 'K
                           ; SCREEN EDITOR (R/W)
                      ;SCREEN EDITOR
;KEYBOARD (R ONLY)
           'S
D1SPLY =
                         ;SCREEN DISPLAY (R/W)
PRINTR =
               ' P
                           ; PRINTER (W ONLY)
               ' C
CASSET =
                           ; CASSETTE
MODEM =
               ' M
                           ; MODEM
DISK =
               'D
                           ;DISK (R/W)
; SYSTEM EOL (CARRIAGE RETURN)
CR =
              $9В
;
;
       OPERATING SYSTEM STATUS CODES
SUCCES =
                $01
                              ;SUCCESSFUL OPERATION
                $80
                              ; BREAK KEY ABORT
BRKABT =
PRVOPN =
                $81
                              ; IOCB ALREADY OPEN
NONDEV =
                $82
                              ; NON-EXISTANT DEVICE
WRONLY =
              $83
                             ; IOCB OPENED FOR WRITE ONLY
                             ; INVALID COMMAND
NVALID =
              $84
NOTOPN =
               $85
                             ;DEVICE OR FILE NOT OPEN
BADIOC =
               $86
                             ; INVALID IOCB NUMBER
RDONLY =
               $87
                             ; IOCB OPENED FOR READ ONLY
EOFERR =
               $88
                              ; END OF FILE
TRNRCD =
              $89
                             ;TRUNCATED RECORD
              $8A
TIMOUT =
                              ; PERIPHERAL DEVICE TIME OUT
DNACK =
              $8B
                              ; DEVICE DOES NOT ACKNOWLEDGE COMMAND
FRMERR =
              $8C
                              ; SERIAL BUS FRAMING ERROR
CRSROR =
              $8D
                              ; CURSOR OVERRANCE
              $8E
                              ;SERIAL BUS DATA OVERRUN
OVRRUN =
                              ; SERIAL BUS CHECKSUM ERROR
CHKERR =
               $8F
DERROR =
               $90
                             ; PERIPHERAL DEVICE ERROR (OPERATION NOT COMP
BADMOD =
               $91
                              ; BAD SCREEN MODE NUMBER
FNCNOT =
               $92
                              ; FUNCTION NOT IMPLEMENTED IN HANDLER
                              ; INSUFICIENT MEMORY FOR SCREEN MODE
SCRMEM =
               $93
       PAGE ZERO RAM ASSIGNMENTS
        *=$0000
LINZBS: .RES 2
                         ;LINBUG RAM (WILL BE REPLACED BY MONITOR RAM
; THESE LOCATIONS ARE NOT CLEARED
CASINI: .RES 2 ;CASSETTE INIT LOCATION
RAMLO: .RES 2
TRAMSZ: .RES 1
                           ; RAM POINTER FOR MEMORY TEST
                           TEMPORARY REGISTER FOR RAM SIZE
TSTDAT: .RES 1
                            ; RAM TEST DATA REGISTER
; CLEARED ON COLOSTART ONLY
WARMST: .RES 1
                           ;WARM START FLAG
```

INSCLR = \$20

```
BOOT?: .RES
              1
                            ;SUCCESSFUL BOOT FLAG
DOSVEC: .RES
               2
                            ;DISK SOFTWARE START VECTOR
DOSINI: .RES
               2
                            ;DISK SOFTWARE INIT ADDRESS
APPMHI: .RES 2
                            ; APPLICATIONS MEMORY HI LIMIT
; CLEARED ON COLD OR WARM START
INTZBS =*
                              ; INTERRUPT HANDLER
              1
POKMSK: .RES
                            ;SYSTEM MASK FOR POKEY IRQ ENABLE
BRKKEY: .RES
               1
                            ;BREAK KEY FLAG
RTCLOK: .RES 3
                            ; REAL TIME CLOCK (IN 16 MSEC UNITS)
;
BUFADR: .RES
                            ; INDIRECT BUFFER ADDRESS REGISTER
ICCOMT: .RES 1
                            ; COMMAND FOR VECTOR
DSKFMS: .RES
               2
                            ; DISK FILE MANAGER POINTER
DSKUTL: .RES
               2
                            ; DISK UTILITIES POINTER
              1
PTIMOT: .RES
                            ; PRINTER TIME OUT REGISTER
PBPNT: .RES
               1
                            ; PRINT BUFFER POINTER
PBUFSZ: .RES
               1
                            ; PRINT BUFFER SIZE
PTEMP: .RES
                1
                            ;TEMPORARY REGISTER
ZIOCB =*
                              ; ZERO PAGE I/O CONTROL BLOCK
              16
IOCBSZ =
                          ; NUMBER OF BYTES PER IOCB
               8*IOCBSZ ; LENGTH OF THE IOCB AREA
MAXIOC =
IOCBAS =*
ICHIDZ: .RES
                            ; HANDLER INDEX NUMBER (FF = IOCB FREE)
                1
ICDNOZ: .RES
                            ; DEVICE NUMBER (DRIVE NUMBER)
               1
ICCOMZ: .RES
                            ; COMMAND CODE
               1
ICSTAZ: .RES
                            ;STATUS OF LAST IOCB ACTION
ICBALZ: .RES
               1
                            ;BUFFER ADDRESS LOW BYTE
               1
ICBAHZ: .RES
ICPTLZ: .RES
              1
                            ; PUT BYTE ROUTINE ADDRESS - 1
ICPTHZ: .RES
               1
               1
ICBLLZ: .RES
                            ;BUFFER LENGTH LOW BYTE
ICBLHZ: .RES
               1
ICAX1Z: .RES
               1
                            ; AUXILIARY INFORMATION FIRST BYTE
ICAX2Z: .RES
               1
ICSPRZ: .RES
                           ;TWO SPARE BYTES (CIO LOCAL USE)
ICIDNO =
               ICSPRZ+2
                          ; IOCB NUMBER X 16
CIOCHR =
                ICSPRZ+3
                          ; CHARACTER BYTE FOR CURRENT OPERATION
              1
STATUS: .RES
                            ; INTERNAL STATUS STORAGE
               1
CHKSUM: .RES
                            ; CHECKSUM (SINGLE BYTE SUM WITH CARRY)
BUFRLO: .RES
               1
                            ; POINTER TO DATA BUFFER (LO BYTE)
BUFRHI: .RES
               1
                            ; POINTER TO DATA BUFFER (HI BYTE)
BFENLO: .RES
               1
                            ; NEXT BYTE PAST END OF THE DATA BUFFER (LO B
BFENHI: .RES
               1
                            ; NEXT BYTE PAST END OF THE DATA BUFFER (HI B
               1
CRETRY: .RES
                            ; NUMBER OF COMMAND FRAME RETRIES
DRETRY: .RES
               1
                            ; NUMBER OF DEVICE RETRIES
BUFRFL: .RES
               1
                           ;DATA BUFFER FULL FLAG
RECVDN: .RES
               1
                            ; RECEIVE DONE FLAG
XMTDON: .RES
               1
                           TRANSMISSION DONE FLAG
CHKSNT: .RES
               1
                           ; CHECKSUM SENT FLAG
NOCKSM: .RES
                1
                            ; NO CHECKSUM FOLLOWS DATA FLAG
;
BPTR: .RES
```

```
FTYPE: .RES
               1
     .RES
FEOF:
FREQ: .RES
               1
SOUNDR: .RES
               1
                            ; NOISY I/O FLAG. (ZERO IS QUIET)
CRITIC: .RES 1
                            ; DEFINES CRITICAL SECTION (CRITICAL IF NON-Z
FMSZPG: .RES
              7
                            ;DISK FILE MANAGER SYSTEM ZERO PAGE
CKEY: .RES
             1
                            ;FLAG SET WHEN GAME START PRESSED
CASSBT: .RES
               1
                            ; CASSETTE BOOT FLAG
DSTAT: .RES
               1
                           ;DISPLAY STATUS
ATRACT: .RES
              1
                           ;ATRACT FLAG
DRKMSK: .RES
               1
                           ; DARK ATRACT MASK
COLRSH: .RES 1
                            ;ATRACT COLOR SHIFTER (EOR'ED WITH PLAYFIELD
LEDGE =
              2
                           ;LMARGN'S VALUE AT COLD START
               39
REDGE =
                          ; RMARGN'S VALUE AT COLD START
TMPCHR: .RES
               1
HOLD1: .RES
               1
LMARGN: .RES
               1
                            ;LEFT MARGIN (SET TO 1 AT POWER ON)
RMARGN: .RES
              1
                           ; RIGHT MARGIN (SET TO 38 AT POWER ON)
ROWCRS: .RES
               1
                           ; CURSOR COUNTERS
COLCRS: .RES
              2
DINDEX: .RES
               1
SAVMSC: .RES
               2
OLDROW: .RES
               1
OLDCOL: .RES
               1
OLDCHR: .RES
                            ; DATA UNDER CURSOR
OLDADR: .RES
              2
NEWROW: .RES
               1
                           ; POINT DRAW GOES TO
NEWCOL: .RES
              2
              1
LOGCOL: .RES
                           ; POINTS AT COLUMN IN LOGICAL LINE
ADRESS: .RES
               2
MLTTMP: .RES
               2
OPNTMP =
              MLTTMP ; FIRST BYTE IS USED IN OPEN AS TEMP
SAVADR: .RES
              2
RAMTOP: .RES
              1
                            ; RAM SIZE DEFINED BY POWER ON LOGIC
              1
BUFCNT: .RES
                           ;BUFFER COUNT
BUFSTR: .RES
              2
                           ;EDITOR GETCH POINTER
BITMSK: .RES
               1
                            ;BIT MASK
SHFAMT: .RES
               1
ROWAC: .RES
               2
COLAC: .RES
               2
ENDPT: .RES
DELTAR: .RES
               1
DELTAC: .RES
               2
ROWINC: .RES
               1
COLINC: .RES
               1
SWPFLG: .RES
              1
                            ;NON-0 IF TXT AND REGULAR RAM IS SWAPPED
HOLDCH: .RES
               1
                            ;CH IS MOVED HERE IN KGETCH BEFORE CNTL & SH
               1
INSDAT: .RES
COUNTR: .RES
```

30 - FF ARE RESERVED FOR USER APPLICATIONS

```
NOTE : SEE FLOATING POINT SUBROUTINE AREA FOR ZERO PAGE CELLS
       PAGE 1 - STACK
       PAGE TWO RAM ASSIGNMENTS
        *=$0200
INTABS =*
                              ;INTERRUPT RAM
VDSLST: .RES
               2
                            ;DISPLAY LIST NMI VECTOR
VPRCED: .RES
               2
                            ; PROCEED LINE IRO VECTOR
              2
                            ;INTERRUPT LINE IRO VECTOR
VINTER: .RES
VBREAK: .RES
              2
                            ;SOFTWARE BREAK (00) INSTRUCTION IRQ VECTOR
VKEYBD: .RES
              2
                            ; POKEY KEYBOARD IRQ VECTOR
VSERIN: .RES
              2
                           ; POKEY SERIAL INPUT READY IRQ
VSEROR: .RES
              2
                            ; POKEY SERIAL OUTPUT READY IRQ
              2
2
VSEROC: .RES
                            ; POKEY SERIAL OUTPUT COMPLETE IRQ
VTIMR1: .RES
                            ; POKEY TIMER 1 IRQ
              2
VTIMR2: .RES
                            ; POKEY TIMER 2 IRQ
              2
VTIMR4: .RES
                            ; POKEY TIMER 4 IRQ
VIMIRU: .RES
              2
                            ; IMMEDIATE IRO VECTOR
CDTMV1: .RES
              2
                            ; COUNT DOWN TIMER 1
CDTMV2: .RES
              2
                            ; COUNT DOWN TIMER 2
CDTMV3: .RES
              2
                            ; COUNT DOWN TIMER 3
              2
2
CDTMV4: .RES
                            ; COUNT DOWN TIMER 4
                            ; COUNT DOWN TIMER S
CDTMV5: .RES
              2
VVBLKI: .RES
                            ; IMMEDIATE VERTICAL BLANK NMI VECTOR
VVBLKD: .RES
              2
                            ; DEFERRED VERTICAL BLANK NMI VECTOR
              2
                            ; COUNT DOWN TIMER 1 JSR ADDRESS
CDTMA1: .RES
CDTMA2: .RES
              2
                            ; COUNT DOWN TIMER 2 JSR ADDRESS
CDTMF3: .RES
              1
                            ; COUNT DOWN TIMER 3 FLAG
SRTIMR: .RES
              1
                            ;SOFTWARE REPEAT TIMER
               1
CDTMF4: .RES
                            ; COUNT DOWN TIMER 4 FLAG
INTEMP: .RES
                            ; IAN'S TEMP (RENAMED FROM T1 BY POPULAR DEMA
              1
CDTMF5: .RES
              1
                            ; COUNT DOWN TIMER FLAG 5
              1
SDMCTL: .RES
                            ; SAVE DMACTL REGISTER
SDLSTL: .RES
              1
                            ; SAVE DISPLAY LIST LOW BYTE
SDLSTH: .RES
               1
                           ;SAVE DISPLAY LIST HI BYTE
               1
SSKCTL: .RES
                            ;SKCTL REGISTER RAM
        .RES
                1
LPENH: .RES
                1
                            ;LIGHT PEN HORIZONTAL VALUE
LPENV: .RES
                1
                            ;LIGHT PEN VERTICAL VALUE
BRKKY: .RES
               2
                            ; BREAK KEY VECTOR
         .RES
                2
                            ;SPARE
CDEVIC: .RES
                1
                            ; COMMAND FRAME BUFFER - DEVICE
CCOMND: .RES
               1
                            ; COMMAND
CAUX1: .RES
               1
                           COMMAND AUX BYTE 1
CAUX2: .RES
                1
                            ; COMMANDAUX BYTE 2
```

```
NOTE: MAY NOT BE THE LAST WORD ON A PAGE
TEMP: .RES
                           ;TEMPORARY RAM CELL
; NOTE: MAY NOT BE THE LAST WORD ON A PAGE
ERRFLG: .RES 1
                           ; ERROR FLAG - ANY DEVICE ERROR EXCEPT TIME
DFLAGS: .RES 1
                           ;DISK FLAGS FROM SECTOR ONE
DBSECT: .RES
                           ; NUMBER OF DISK BOOT SECTORS
               1
              2
BOOTAD: .RES
                           ; ADDRESS WHERE DISK BOOT LOADERWILL BE PUT
COLDST: .RES 1
                           ; COLDSTART FLAG (1=IN MIDDLE OF COLDSTART)
         .RES 1
                           ;SPARE
DSKTIM: .RES
             1
                           ;DISK TIME OUT REGISTER
LINBUF: .RES 40
                         ; CHAR LINE BUFFER
GPRIOR: .RES 1
                           ;GLOBAL PRIORITY CELL
PADDL0: .RES
              1
                           ; POTENTIOMETER 0 RAM CELL
PADDL1: .RES
               1
PADDL2: .RES
               1
PAODL3: .RES
PADDL4: .RES
PADDL5: .RES
              1
PADDL6: .RES
               1
PADDL7: .RES
STICK0: .RES
              1
                            ; JOYSTICK 0 RAM CELL
              1
STICK1: .RES
STICK2: .RES
              1
STICK3: .RES
PTRIGO: .RES
              1
                           ; PADDLE TRIGGER 0
PTRIG1: .RES
PTRIG2: .RES
PTRIG3: .RES
PTRIG4: .RES
PTRIG5: .RES
               1
PTRIG6: .RES
              1
PTRIG7: .RES
              1
STRIGO: .RES
                            ;JOYSTICK TRIGGER 0
STRIG1: .RES
              1
STRIG2: .RES
              1
STRIG3: .RES
CSTAT: .RES
WMODE: .RES
BLIM: .RES
IMASK: .RES
               1
JVECK: .RES
              2
         .RES 2
                           ;SPARE
             1
                           ;TEXT ROWCRS
TXTROW: .RES
TXTCOL: .RES
              2
                            ;TEXT COLCRS
              1
                           ;TEXT INDEX
TINDEX: .RES
TXTMSC: .RES
              2
                          ; FOOLS CONVRT INTO NEW MSC
TXTOLD: .RES
                            ;OLDROW & OLDCOL FOR TEXT (AND THEN SOME)
```

```
TMPX1: .RES
HOLD3: .RES
SUBTMP: .RES
HOLD2: .RES
              1
DMASK: .RES
TMPLBT: .RES
ESCFLG: .RES
              1
                          ; ESCAPE FLAG
TABMAP: .RES
             15
LOGMAP: .RES
                          ;LOGICAL LINE START BIT MAP
INVFLG: .RES
             1
                          ; INVERSE VIDEO FLAG (TOGGLED BY ATARI KEY)
             1
FILFLG: .RES
                          ; RIGHT FILL FLAG FOR DRAW
TMPROW: .RES
              2
TMPCOL: .RES
SCRFLG: .RES
              1
                          ;SET IF SCROLL OCCURS
HOLD4: .RES
              1
                          ;TEMP CELL USED IN DRAW ONLY
HOLD5: .RES
              1
                          ;DITTO
              1
SHFLOK: .RES
BOTSCR: .RES
              1
                          ;BOTTOM OF SCREEN : 24 NORM 4 SPLIT
PCOLR0: .RES 1
                          ;P0 COLOR
PCOLR1: .RES
              1
                          ;P1 COLOR
PCOLR2: .RES
              1
                         ;P2 COLOR
PCOLP3: .RES
              1
                         ;P3 COLOR
COLOR0: .RES
              1
                          ; COLOR 0
COLOR1: .RES
              1
CQLOR2: .RES
              1
COLOR3: .RES
COLOR4: .RES
        .RES 23
                        ;SPARE
GLBABS =*
                            GLOBAL VARIABLES
        .RES 4
                          ;SPARE
RAMSIZ: .RES 1
                          ; RAM SIZE (HI BYTE ONLY)
MEMTOP: .RES
              2
                          ;TOP OF AVAILABLE USER MEMORY
MEMLO: .RES
              2
                          ;BOTTOM OF AVAILABLE USER MEMORY
       .RES 1
                          ;SPARE
DVSTAT: .RES 4
                          ;STATUS BUFFER
              1
CBAUDL: .RES
                          ; CASSETTE BAUD RATE LOW BYTE
CBAUDH: .RES
                          ; CURSOR INHIBIT (00 = CURSOR ON)
CRSINH: .RES 1
KEYDEL: .RES
                          ;KEY DELAY
CH1: .RES
              1
             1
CHACT: .RES
                          ; CHACTL REGISTER RAM
CHBAS: .RES
             1
                          ; CHBAS REGISTER RAM
        .RES 5
                          ;SPARE BYTES
CHAR: .RES
                        ;ATASCII CHARACTER
ATACHR: .RES
              1
                      ;GLOBAL VARIABLE FOR KEYBOARD
CH: .RES 1
```

```
FILDAT: .RES
              1
                          ; RIGHT FILL DATA (DRAW)
DSPFLG: .RES
             1
                           ;DISPLAY FLAG : DISPLAY CNTLS IF NON-ZERO
SSFLAG: .RES 1
                           ;START/STOP FLAG FOR PAGING (CNTL 1). CLEARE
       PAGE THREE RAM ASSIGNMENTS
DCB =*
                  ; DEVICE CONTROL BLOCK
DDEVIC: .RES
                          ; PERIPHERAL UNIT 1 BUS ID. NUMBER
              1
              1
DUNIT: .RES
                           ;UNIT NUMBER
DCOMND: .RES
              1
                           ;BUS COMMAND
DSTATS: .RES
              1
                           ; COMMAND TYPE/STATUS RETURN
DBUFLO: .RES
              1
                           ;DATA BUFFER POINTER LOW BYTE
DBUFHI: .RES
              1
DTIMLO: .RES
                           ; DEVICE TIME OUT IN 1 SECOND UNITS
              1
DUNUSE: .RES
              1
                           ;UNUSED BYTE
DBYTLO: .RES
              1
                           ; NUMBER OF BYTES TO BE TRANSFERRED LOW BYTE
DBYTHI: .RES
              1
DAUX1: .RES
              1
                           ; COMMAND AUXILIARY BYTE 1
DAUX2: .RES
              1
TIMER1: .RES 2
                           ; INITIAL TIMER VALUE
ADDCOR: .RES
              1
                           ; ADDITION CORRECTION
CASFLG: .RES
              1
                          ; CASSETTE MODE WHEN SET
TIMER2: .RES
              2
                           ; FINAL TIMER VALUE. THESE TWO TIMER VALUES
; ARE USED TO COMPUTE INTERVAL FOR BAUD RATE
TEMP1: .RES 2
                          ;TEMPORARY STORAGE REGISTER
TEMP2: .RES
              1
                           ;TEMPORARY STORAGE REGISTER
TEMP3: .RES
              1
                          ;TEMPORARY STORAGE REGISTER
SAVIO: .RES
              1
                          ; SAVE SERIAL IN DATA PORT
             1
TIMFLG: .RES
                           ;TIME OUT FLAG FOR BAUD RATE CORRECTION
STACKP: .RES
                          ;SIO STACK POINTER SAVE CELL
              1
TSTAT: .RES 1
                          ;TEMPORARY STATUS HOLDER
                          ; HANDLER ADDRESS TABLE
HATABS: .RES
               38
MAXDEV = *-HATABS-5 ; MAXIMUM HANDLER ADDRESS INDEX
    NOTE: THE ENTIRE IOCB DEFINITIONS HAVE BEEN MODIFIED
IOCB:
      .ORG
                           ;I/O CONTROL BLOCKS
ICHID: .RES
              1
                           ; HANDLER INDEX NUMBER (FF = IOCB FREE)
ICDNO: .RES
                           ; DEVICE NUMBER (DRIVE NUMBER)
              1
ICCOM: .RES
              1
                           ; COMMAND CODE
                           ;STATUS OF LAST IOCB ACTION
ICSTA: .RES
              1
ICBAL: .RES
              1
                           ;BUFFER ADDRESS LOW BYTE
ICBAH: .RES
              1
ICPTL: .RES
              1
                           ; PUT BYTE ROUTINE ADDRESS - 1
ICPTH: .RES
              1
ICBLL: .RES
               1
                           ;BUFFER LENGTH LOW BYTE
ICBLH: .RES
              1
ICAX1: .RES
              1
                          ; AUXILIARY INFORMATION FIRST BYTE
ICAX2: .RES
```

```
ICSPR: .RES 4
                    ; FOUR SPARE BYTES
       .RES MAXIOC-IOCBSZ
PRNBUF: .RES 40 ;PRINTER BUFFER
        RES 21 ;SPARE BYTES
      PAGE FOUR RAM ASSIGNMENTS
CASBUF: .RES 131 ;CASSETTE BUFFER
; USER AREA STARTS HERE AND GOES TO END OF PAGE FIVE
USAREA: .RES 128
                          ;SPARE
      PAGE FIVE RAM ASSIGNMENTS
      PAGE FIVE IS RESERVED AS A USER WORK SPACE
      NOTE: SEE FLOATING POINT SUBROUTINE AREA FOR PAGE FIVE CELLS
       PAGE SIX RAM ASSIGNMENTS
; PAGE SIX IS RESERVED AS A USER'S USER WORK SPACE
      FLOATING POINT SUBROUTINES
                        ;FLOATING PT PRECISION (# OF BYTES)
FPREC =
             6
; IF CARRY USED THEN CARRY CLEAR => NO ERROR, CARR
   = $D800 ;ASCII->FLOATING POINT(FP)
AFP
                                  INBUFF+CIX -> FR0, CIX, CARRY
FASC = $D8E6
IFP = $D9AA
                        ;FP -> ASCII FR0 -> LBUFF (INBUFF)
                   ;INTEGER -> FP
                                   0-\$FFFF (LSB,MSB) IN FR0,FR0+1->FR0
          $D9D2 ;FP -> INTEGER FRO -> FRO,FRO+1, CARRY
FPI =
      = $DA60
                        ;FR0 <- FR0 - FR1 ,CARRY
FSUB
FADD
      =
              $DA66
                        ;FR0 <- FR0 + FR1 ,CARRY
              $DADB
                        ;FR0 <- FR0 * FR1 ,CARRY
FMUL
      =
FDIV
                        ;FR0 <- FR0 / FR1 ,CARRY
             $DB28
      =
                      ;FLOATING LOAD REGO FRO <- (X,Y)
FLDOR =
             $DD89
                       ; "
; "
             $DD80
                                FLDOP =
             $DD98
                                 " REG1 FR1 \leftarrow (X,Y)
FLD1R =
                        ; " " FR1 <- (FLPTR)
FLD1P =
             $DD9C
              $DDA7 ;FLOATING STORE REGO (X,Y) <- FRO
FSTOR =
```

```
; " " (FLPTR) <- FR0
FSTOP =
               $DDAB
FMOVE =
               $DDB6
                        ;FR1 <- FR0
                         ;FR0 \leftarrow P(Z) = SUM(I=N TO 0) (A(I)*Z**I) CAR
PLYEVL =
               $DD40
                               INPUT: (X,Y) = A(N), A(N-1)...A(0) \rightarrow PLYARG
                                        ACC = # OF COEFFICIENTS = DEGREE+1
;
                                        FR0 = Z
          $DDC0
                     ;FR0 <- E**FR0 = EXP10(FR0 * LOG10(E)) CARRY
EXP =
EXP10 =
           $DDCC
                      ;FR0 <- 10**FR0 CARRY
LOG =
           $DECD
                     ; FR0 \leftarrow LN(FR0) = LOG10(FR0)/LOG10(E) CARRY
             $DED1
LOG10 =
                         ;FR0 <- LOG10 (FR0) CARRY
; THE FOLLOWING ARE IN BASIC CARTRIDGE:
SIN = $BDB1 ;FR0 <- SIN(FR0) DEGFLG=0 =>RADS, 6=>DEG. CA
COS =
          $BD73
                     ;FR0 <- COS(FR0) CARRY
ATAN
           $BE43
                        ;FR0 <- ATAN(FR0) CARRY
SOR =
           $BEB1 ;FR0 <- SOUAREROOT(FR0) CARRY
; FLOATING POINT ROUTINES ZERO PAGE (NEEDED ONLY IF V.P. ROUTINES ARE CA
        *=$D4
FR0:
       .RES
             FPREC
                        ;FP REGO
       .RES FPREC
FRE:
                        ;FP REG1
       .RES FPREC
FR1:
FR2:
       .RES FPREC
       .RES
FRX:
              1
                          ;FP SPARE
EEXP:
      .RES
             1
                         ; VALUE OF E
NSIGN: .RES
             1
                         ;SIGN OF #
ESIGN: .RES
             1
                         ;SIGN OF EXPONENT
             1
FCHRFLG: .RES
                         ;1ST CHAR FLAG
DIORT: .RES
              1
                         ;# OF DIGITS RIGHT OF DECIMAL
       .RES 1
                         CURRENT INPUT INDEX
CIX:
INBUFF: .RES
             2
                         ; POINTS TO USER'S LINE INPUT BUFFER
ZTEMPI: .RES
              2
ZIEMP4: .RES
             2
ZTEMP3: .RES 2
DEGFLG
RADFLG: .RES 1
                         ;0=RADIANS, 6=DEGREES
RADON =
             0
                         ; INDICATES RADIANS
             6
                          ; INDICATES DEGREES
DEGON
              2
                          ; POINTS TO USER'S FLOATING PT NUMBER
FLPTR: .RES
FPTR2: .RES
              2
; FLOATING PT ROUTINES' NON-ZERO PAGE RAM
; (NEEDED ONLY IF F.P. ROUTINES CALLED)
        *=$57E
      .RES
LBPR1:
              1
                          ;LBUFF PREFIX 1
              1
LBPR2: .RES
                          ;LBUFF PREFIX 2
LBUFF: .RES
              128
                            ;LINE BUFFER
              LBUFF+$60 ; POLYNOMIAL ARGUMENTS
PLYARG =
FPSCR =
             PLYARG+FPREC
FPSCR1 =
             FPSCR+FPREC
             FPSCR
     =
FSCR
FSCR1 =
             FPSCR1
LBFEND =
             * – 1
                        ;END OF LBUFF
```

```
COLLEEN MNEMONICS
POKEY
                 $D200
                             ; VBLANK ACTION:
                                                           DESCRIPTION:
        =
POT0
                              ;POT0-->PADDL0
                                                        0-227 IN RAM CELL
                 POKEY+0
                                                        0-227 IN RAM CELL
POT1
                 POKEY+1
                               ;POT1-->PADDL1
                                                        0-227 IN RAM CELL
POT2
                 POKEY+2
                               ;POT2-->PADDL2
                                                        0-227 IN RAM CELL
                               ;POT3-->PADDL3
POT3
         =
                 POKEY+3
                                                        0-227 IN RAM CELL
POT4
                 POKEY+4
                               ;POT4-->PADDL4
POT5
                 POKEY+5
                               ;POT5-->PADDL5
                                                        0-227 IN RAM CELL
                              ;POT6-->PADDL6
                                                        0-227 IN RAM CELL
POT6
        =
                 POKEY+6
POT7
                 POKEY+7
                              ;POT7-->PADDL7
                                                        0-227 IN RAM CELL
ALLPOT =
                 POKEY+8
                              ; ? ? ? ?
KBCODE =
                 POKEY+9
RANDOM =
                 POKEY+10
POTGO
                 POKEY+11
                             ;STROBED
SERIN
                 POKEY+13
IRQST
                 POKEY+14
        =
SKSTAT =
                 POKEY+15
AUDF1
                 POKEY+0
AUDC1
       =
                 POKEY+1
AUDF2
        =
                 POKEY+2
AUDC2
                 POKEY+3
AUDF3
       =
                 POKEY+4
AUDC3
                 POKEY+5
       =
AUDF4
       =
                 POKEY+6
AUDC4
                 POKEY+7
AUDCTL =
                                                      AUDCTL<--[SIO]
                 POKEY+8
                              ; NONE
STIMER =
                 POKEY+9
                 POKEY+10
                             ; NONE
                                                       SKRES<--[SIO]
SKRES =
SEROUT =
                             ; NONE
                                                      SEROUT<--[SIO]
                 POKEY+13
                             ; POKMSK-->IRQEN (AFFECTED BY OPEN S: OR E:)
IRQEN
                 POKEY+14
       =
SKCTL
                                                  SSKCTL<--[SIO]
                 POKEY+15
                             ;SSKCTL-->SKCTL
                 $D000
                             ; VBLANK ACTION:
                                                      DESCRIPTION:
CTIA
       =
HPOSP0 =
                 CTIA+0
HPOSP1 =
                 CTIA+1
HPOSP2 =
                 CTIA+2
HPOSP3 =
                 CTIA+3
HPOSM0
                 CTIA+4
HPOSM1 =
                 CTIA+5
HPOSM2 =
                 CTIA+6
HPOSM3 =
                 CTIA+7
SIZEPO =
                 CTIA+8
SIZEP1 =
                 CTIA+9
SIZEP2 =
                 CTIA+10
SIZEP3 =
                 CTIA+11
SIZEM
                 CTIA+12
GRAFP0 =
                 CTIA+13
GRAFP1 =
                 CTIA+14
GRAFP2 =
                 CTIA+15
GRAFP3 =
                 CTIA+16
GRAFM
                 CTIA+17
COLPMO =
                 CTIA+18
                               ; PCOLR0-->COLPM0
                                                      WITH ATTRACT MODE
COLPM1 =
                 CTIA+19
                               ; PCOLR1-->COLPM1
                                                      WITH ATTRACT MODE
                 CTIA+20
                               ;PCOLR2-->COLPM2
                                                      WITH ATTRACT MODE
COLPM2 =
COLPM3 =
                 CTIA+21
                               ; PCOLR3-->COLPM3
                                                      WITH ATTRACT MODE
                               ;COLOR0-->COLPF0
                                                      WITH ATTRACT MODE
COLPF0 =
                 CTIA+22
                                                      WITH ATTRACT MODE
COLPF1 =
                 CTIA+23
                               ;COLOR1-->COLPF1
COLPF2 =
                 CTIA+24
                               ;COLOR2-->COLPF2
                                                      WITH ATTRACT MODE
```

```
;COLOR3-->COLPF3
COLPF3 =
               CTIA+25
                                             WITH ATTRACT MODE
COLBK =
               CTIA+26
                          ;COLOR4-->COLBK
                                               WITH ATTRACT MODE
                          ; (ON OPEN S: OR E:) GPRIOR-->PRIOR
PRIOR =
               CTIA+27
VDELAY =
               CTIA+28
GRACTL =
               CTIA+29
HITCLR =
              CTIA+30
CONSOL =
                                              TURN OFF SPEAKER
              CTIA+31
                         ;$08-->CONSOL
MOPF
              CTIA+0
M1PF
              CTIA+1
      =
               CTIA+2
M2PF
M3PF
              CTIA+3
              CTIA+4
P0PF
P1PF
       =
              CTIA+5
              CTIA+8
P2PF
       =
P3PF
              CTIA+7
MOPL
               CTIA+8
M1PL
              CTIA+9
       =
M2PL
              CTIA+10
       =
M3PL
              CTIA+11
P0PL
              CTIA+12
       =
p1pt.
      =
              CTIA+13
P2PL
               CTIA+14
P3PL
      =
              CTIA+15
TRIGO =
              CTIA+16
                         ;TRIG0-->STRIG0
TRIG1 =
              CTIA+17
                          ;TRIG1-->STRIG1
TRIG2 =
               CTIA+18
                         ;TRIG2-->STRIG2
               CTIA+19
TRIG3 =
                         ;TRIG3-->STRIG3
;
                        ; VBLANK ACTION DESCRIPTION
ANTIC =
              $D400
                          ;DMACTL<--SDMCTL
DMACTL =
              ANTIC+0
                                            ON OPEN S: OR E:
CHACTL =
             ANTIC+1
                         ;CHACTL<--CHACT
                                               ON OPEN S: OR E:
                         ;DLISTL<--SDLSTL
                                             ON OPEN S: OR E:
DLISTL =
             ANTIC+2
DLISTH =
              ANTIC+3
                          ;DLISTH<--SDLSTH
                                            ON OPEN S: OR E:
NSCROL =
             ANTIC+4
VSCROL =
             ANTIC+5
PMBASE =
              ANTIC+7
CHBASE =
             ANTIC+9
                         ;CHBASE<--CHBAS
                                               ON OPEN S: OR E:
WSYNC =
             ANTTC+10
VCOUNT =
             ANTIC+11
PENH =
             ANTIC+12
PENV
      =
             ANTIC+13
NMIEN =
             ANTIC+14
                        ;NMIEN<--40 POWER ON AND [SETVBV]
NMIRES =
             ANTIC+15
                        ;STROBED
NMIST =
             ANTIC+15
          $D300 ; VBLANK ACTION DESCRIPTION
PIA =
PORTA =
             PIA+0
                       ; PORTA-->STICK0,1 X-Y CONTROLLERS
PORTB =
              PIA+1
                         ;PORTB-->STICK2,3
                                             X-Y CONTROLLERS
PACTL =
              PIA+2
                         ; NONE
                                               PACTL<--3C [INIT]
PBCTL =
             PIA+3
                         ; NONE
                                               PBCTL<--3C [INIT]
;
; .PAGE
         .PAGE
        LIST S
         .TITLE 'CENTRAL INPUT/OUTPUT (CIO) 2-7-79'
                UPDATED BY AL MILLER 3-9-79
ASCZER =
              ' 0
                         ;ASCII ZERO
COLON
               $3A
                           ;ASCII COLON
```

```
EOL =
           $9B
                         ;END OF RECORD
         .PAGE
; CIO JUMP VECTOR FOR USERS
         *=CIOV
                       ;GO TO CIO
         JMP CIO
; CIO INIT JUMP VECTOR FOR POWER UP
         *=CIOINV
         JMP CIOINT ;GO TO INIT
;
; ERROR ROUTINE ADDRESS EQUATE
; ERRTNH =ERRTN/256
                          "MOVED TO LINE 788"
; ERRTNL =-ERRTNH*256+ERRTN "MOVED TO LINE 789"
;
         *=CIOORG
; CIO INITIALIZATION (CALLED BY MONITOR AT POWER UP)
CIOINT: LDX #0
CIOI1: LDA #IOCFRE ;SET ALL IOCB'S TO FREE
         STA ICHID, X ; BY SETTING HANDLER ID'S=$FF
         LDA
               #ERRTNL
               ICPTL,X
         STA
                         ; POINT PUT TO ERROR ROUTINE
         LDA
               #ERRTNH
         STA ICPTH, X
         TXA
         CLC
         ADC
               #IOCBSZ
                          ;BUMP INDEX BY SIZE
         TAX
         CMP
               #MAXIOC
                           ; DONE?
         CMP #MAXIOC ;DO
BCC CIOI1 ;NO
                               ;YES, RETURN
         RTS
; ERROR ROUTINE FOR ILLEGAL PUT
ERRTN =*-1
ERRTNH = ERRTN/256
ERRTNL = (-ERRTNH)*256+ERRTN
         LDY #NOTOPN ; IOCB NOT OPEN
         RTS
         .PAGE
; CIO LOCAL RAM (USES SPARE BYTES IN ZERO PAGE IOCB)
ENTVEC = ICSPRZ
; CIO MAIN ROUTINE
; CIO INTERFACES BETWEEN USER AND INPUT/OUTPUT DE
CIO: STA CIOCHR ;SAVE POSSIBLE OUTPUT CHARACTER STX ICIDNO ;SAVE IOCB NUMBER * N
; CHECK FOR LEGAL IOCB
         TXA
               #$F
CIERR1
#MAXIOC
                            ; IS IOCB MULTIPLE OF 16? ;NO, ERROR
         AND
         BNE
                          ; IS INDEX TOO LARGE?
         CPX
         BCC
               IOC1
```

```
CIERR1: LDY #BADIOC ; ERROR CODE
        JMP CIRTN1 ;RETURN
; MOVE USER IOCB TO ZERO PAGE
IOC1: LDY #0
IOC1A: LDA IOCB,X
                          ;USER IOCB
         STA IOCBAS,Y ;TO ZERO PAGE
         INX
         INY
                       ;12 BYTES
         CPY
               #12
         BCC
                IOC1A
; COMPUTE CIO INTERNAL VECTOR FOR COMMAND
         I-DX
               #NVALID ;ASSUME INVALID CODE
               ICCOMZ ; COMMAND CODE TO INDEX #OPEN ; IS COMMAND LEGAL?
         LDA
         CMP
                CIERR4
         BCC
                             ; NO
         TAY
; MOVE COMMAND TO ZERO BASE FOR INDEX
         CPY
               #SPECIL ; IS COMMAND SPECIAL?
       BCC IOC2 ;NO
LDY #SPECIL ;YES, SET SPECIAL OFFSET INDEX
STY ICCOMT ;SAVE COMMAND FOR VECTOR
IOC2:
         LDA COMTAB-3,Y ;GET VECTOR OFFSET FROM TABLE
         BEQ CIOPEN ;GO IF OPEN COMMAND CMP #2 ;IS IT CLOSE?
BEQ CICLOS ;YES
         CMP #8
BCS CISTSP
                               ; IS IT STATUS OR SPECIAL?
                            ;YES
         CMP
                #4
                               ; IS IT READ?
         BEQ CIREAD JMP CIWRIT
                           ;YES
;ELSE, MUST BE WRITE
         .PAGE
; OPEN COMMAND
; FIND DEVICE HANDLER IN HANDLER ADDRESS TABLE
CIOPEN: LDA ICHIDZ ;GET HANDLER ID
         CMP #IOCFRE ;IS THIS IOCB CLOSED?
         BEQ
                IOC6
                            ;YES
; ERROR -- IOCB ALREADY OPEN
CIERR3: LDY #PRVOPN ; ERROR CODE
CIERR4: JMP CIRTN1
                          ; RETURN
; GO FIND DEVICE
IOC6: JSR DEVSRC ; CALL DEVICE SEARCH
BCS CIERR4 ; GO IF DEVICE NOT FOUND
; DEVICE FOUND, INITIALIZE IOCB FOR OPEN
; COMPUTE HANDLER ENTRY POINT
IOC7: JSR COMENT
        BCS CIERR4
                        GO IF ERROR IN COMPUTE
; GO TO HANDLER FOR INITIALIZATION
         JSR GOHAND ; USE INDIRECT JUMP
```

; INVALID IOCB NUMBER -- RETURN ERROR

```
; STORE PUT BYTE ADDRESS-1 INTO IOCB
               #PUTCHR ;SIMULATE PUT CHARACTER
         LDA
         STA ICCOMT

JSR COMENT ; COMPUTE ENTRY I

LDA ICSPRZ ; MOVE COMPUTED V

STA ICPTLZ ; TO PUT BYTE ADI

LDA ICSPRZ+1

STA ICPTHZ

JMP CIRTN2 ; RETURN TO USER
                             ; COMPUTE ENTRY POINT
                              ; MOVE COMPUTED VALUE
                              ;TO PUT BYTE ADDRESS
          .PAGE
; CLOSE COMMAND
CICLOS: LDY #SUCCES ;ASSUME GOOD CLOSE
         STY ICSTAZ
                              ; COMPUTE HANDLER ENTRY POINT
         JSR
                COMENT
         BCS
                CICLO2
                              GO IF ERROR IN COMPUTE
         JSR GOHAND
                               ;GO TO HANDLER TO CLOSE DEVICE
CICLO2: LDA #IOCFRE ;GET IOCB "FREE" VALUE
         STA ICHIDZ
LDA #ERRTNH
STA ICPTHZ
                          ;SET HANDLER ID
                              ;SET PUT BYTE TO POINT TO ERROR
         LDA #ERRTNL
STA ICPTLZ
JMP CIRTN2 ;RETURN
; STATUS AND SPECIAL REQUESTS
; DO IMPLIED OPEN IF NECESSARY AND GO TO DEVICE
CISTSP: LDA ICHIDZ ; IS THERE A HANDLER ID?
         CMP #IOCFRE
         BNE CIST1 ;YES
; IOCB IS FREE, DO IMPLIED OPEN
         JSR DEVSRC ;FIND DEVICE IN TABLE
         BCS
                CIERR4
                              ;GO IF ERROR IN COMPUTE
; COMPUTE AND GO TO ENTRY POINT IN HANDLER
CIST1: JSR COMENT ; COMPUTER HANDLER ENTRY VECTOR
                               ;GO TO HANDLER
         JSR GOHAND
; RESTORE HANDLER INDEX (DO IMPLIED CLOSE)
         LDX ICIDNO
                          ; IOCB INDEX
                ICHID,X ;GET ORIGINAL HANDLER ID
         LDA
         STA
                ICHIDZ
                            RESTORE ZERO PAGE
         JMP
               CIRTN2
                              ;RETURN
          .PAGE
; READ -- DO GET COMMANDS
CIREAD: LDA ICCOMZ ;GET COMMAND BYTE
         AND ICAX1Z
                          ; IS THIS READ LEGAL?
                RCI1A
                           ;YES
         BNE
; ILLEGAL READ -- IOCB OPENED FOR WRITE ONLY
        LDY #WRONLY ; ERROR CODE
RCI1B: JMP CIRTN1 ; RETURN
; COMPUTE AND CHECK ENTRY POINT
```

```
RCI1A: JSR COMENT ; COMPUTE ENTRY POINT
         BCS RCI1B
                            ;GO IF ERROR IN COMPUTE
; GET RECORD OR CHARACTERS
          LDA ICBLLZ
          ORA
                 ICBLLZ+1 ; IS BUFFER LENGTH ZERO?
                 RCI3
          BNE
                             ; NO
          JSR GOHAND
STA CIOCHR
JMP CIRTN2
; LOOP TO FILL BUFFER OR END RECORD
RCI3: JSR GOHAND ;GO TO HANDLER TO GET BYTE
          STA CIOCHR ;SAVE BYTE

BMI RCI4 ;END TRANSFER IF ERROR

LDY #0

STA (ICBALZ),Y ;PUT BYTE IN USER BUFFER
                INCBFP ; INCREMENT BUFFER POINTER
          JSR
          LDA ICCOMZ
AND #2
                            GET COMMAND CODE; IS IT GET RECORD?
          BNE RCI1 ;NO
; CHECK FOR EOL ON TEXT RECORDS
          LDA CIOCHR ;GET BYTE
          CMP #EOL ;IS IT AN EOL?
BNE RCI1 ;NO

JSR DECBFL ;YES, DECREMENT BUFFER LENGTH
JMP RCI4 ;END TRANSFER
; CHECK BUFFER FULL
        JSR DECBFL ;DECREMENT BUFFER LENGTH
BNE RCI3 ;CONTINUE IF NON ZERO
RCI1: JSR DECBFL
          .PAGE
; BUFFER FULL. RECORD NOT ENDED
; DISCARD BYTES UNTIL END OF RECORD
RCI2: LDA ICCOMZ ;GET COMMAND BYTE
         AND #2
                             ; IS IT GET CHARACTER?
          BNE RCI4 ; YES, END TRANSFER
         STA CIOCHR ;SAVE GUARTELES

BMT DGG:
; LOOP TO WAIT FOR EOL
RCI6: JSR GOHAND
         BMI RCI4 ;GO IF ERROR
; TEXT RECORD. WAIT FOR EOL
          LDA CIOCHR ;GET GOT BYTE
CMP #EOL ;IS IT EOL?
BNE RCI6
          BNE RCI6
                             ; NO , CONTINUE
; END OF RECORD. BUFFER FULL -- SEND TRUNCATED RECORD MESSAGE
RCI1I: LDA #TRNRCD ; ERROR CODE
        STA ICSTAZ
                            ;STORE IN 10GB
; TRANSFER DONE
RCI4: JSR SUBBFL ;SET FINAL BUFFER LENGTH JMP CIRTN2 ;RETURN
         .PAGE
```

```
; WRITE -- DO PUT COMMANDS
CIWRIT: LDA ICCOMZ ;GET COMMAND BYTE
          AND ICAX1Z ;IS THIS WRITE LEGAL?
BNE WCIIA ;YES
         AND ICAX1Z
; ILLEGAL WRITE -- DEVICE OPENED FOR READ ONLY
         LDY #RDONLY ; ERROR CODE
WCI1B: JMP CIRTN1
                            ; RETURN
; COMPUTE AND CHECK ENTRY POINT
WCIIA: JSR COMENT ; COMPUTE HANDLER ENTRY POINT
        BCS WCI1B
                           GO IF ERROR IN COMPUTE
; PUT RECORD OR CHARACTERS
          LDA ICBLLZ
                ICBLLZ+1 ;IS BUFFER LENGTH ZERO? WCI3 ;NO
          ORA
          BNE
                              GET CHARACTER; SET SUFFER LENOTHI
          LDA
                 CIOCHR
          INC ICBLLZ ;SET SUFFER LENOTHI
BNE WCI4 ;THEN JUST TRANSFER ONE BYTE
; LOOP TO TRANSFER BYTES FROM BUFFER TO HANDLER
WCI3: LDY #0
          LDA
                (ICBALZ),Y ;GET BYTE FROM BUFFER
          STA CIOCHR
                                ;SAVE
WCI4: JSR GOHAND ;GO PUT BYTE

BMI WCI5 ;END IF ERROR

JSR INCBFP ;INCREMENT I
                              ; INCREMENT BUFFER POINTER
; CHECK FOR TEXT RECORD
          DR TEAT RECE
                               GET COMMAND BYTE; IS IT PUT RECORD?
          AND #2;
BNE WCI1;NO
; TEXT RECORD -- CHECK FOR EOL TRANSFER
          LDA CIOCHR ;GET LAST CHARACTER
          CMP #EOL ;IS IT AN EOL?
BNE WCI1 ;NO

JSR DECBFL ;DECREMENT BUFFER LENGTH
JMP WCI5 ;END TRANSFER
; CHECK FOR BUFFER EMPTY
WCI1: JSR DECBFL ;DECREMENT BUFFER LENGTH
BNE WCI3 ;CONTINUE IF NON ZERO
         .PAGE
; BUFFER EMPTY, RECORD NOT FILLED
; CHECK TYPE OF TRANSFER
WCI2: LDA ICCOMZ ;GET COMMAND CODE
                 #2 ; IS IT PUT CHARACTER?
WCI5 ; YES, END TRANSFER
         AND #2
          BNE
; PUT RECORD (TEXT), BUFFER , EMPTY, SEND EOL
          LDA #EOL
          JSR GOHAND ;GO TO HANDLER
; END PUT TRANSFER
WCI5: JSR SUBBFL ;SET ACTUAL PUT BUFFER LENGTH

JMP CIRTN2 ;RETURN
```

```
; CIO RETURNS
; RETURNS WITH Y=STATUS
CIRTN1: STY ICSTAZ ;SAVE STATUS
; RETURNS WITH STATUS STORED IN ICSTAZ
; MOVE IOCB IN ZERO PAGE BACK TO USER AREA
CIRTN2: LDY ICIDNO
                    GET IOCB INDEX
               ICBAL,Y
         LDA
                            ; RESTORE USER BUFFER POINTER
         STA
               ICBALZ
         LDA
               ICBAH,Y
         STA
               ICBAHZ
         LDX
                             ;LOOP COUNT AND INDEX
               #0
CIRT3: LDA IOCBAS,X ;ZERO PAGE
         STA IOCB,Y
                            ;TO USER AREA
         INX
         INY
         CPX #12
BCC CIRT3
                            ;12 BYTES
; RESTORE A,X, & Y
        LDA CIOCHR
                        GET LAST CHARACTER
         LDX
               ICIDNO
                            ; IOCB INDEX
         LDY
               ICSTAZ
                            GET STATUS AND SET FLAGS
                              ; RETURN TO USER
         RTS
         .PAGE
; CIO SUBROUTINES
; COMENT -- CHECK AND COMPUTE HANDLER ENTRY POINT
COMENT: LDY ICHIDZ ;GET HANDLER INDEX
             #MAXDEV+1 ; IS IT A LEGAL INDEX?
         CPY
         BCC
               COM1
                          ;YES
; ILLEGAL HANDLER INDEX MEANS DEVICE NOT OPEN FOR OPERATION
        LDY #NOTOPN ; ERROR CODE
         BCS COM2
                           ; RETURN
; USE HANDLER ADDRESS TABLE AND COMMAND TABLE TO GET VECTOR
COM1:
       LDA HATABS+1,Y ;GET LOW BYTE OF ADDRESS
         STA ICSPRZ ; AND SAVE IN POINTER
               HATABS+2,Y ;GET HI BYTE OF ADDRESS
         LDA
              ICSPRZ+1
ICCOMT
         STA
         LDY
                            GET COMMAND CODE
               COMTAB-3,Y ;GET COMMAND OFFSET
         LDA
         TAY
         LDA (ICSPRZ),Y ;GET LOW BYTE OF VECTOR FROM
         TAX
                              ; HANDLER ITSELF AND SAVE
         INY
         LDA
               (ICSPRZ),Y ;GET HI BYTE OF VECTOR
               ICSPRZ+1
         STA
         STX
               ICSPRZ
                            ;SET LO BYTE
                              ;SHOW NO ERROR
        CLC
COM2: RTS
; DECBFL -- DECREMENT BUFFER LENGTH DOUBLE BYTE
```

. PAGE

```
DECBFL: DEC ICBLLZ ; DECREMENT LOW BYTE
         LDA ICBLLZ
                            ;CHECK IT
              #$FF ;DID IT GO BELOW?
DECBF1 ;NO
         CMP
         BNE
                ICBLLZ+1 ; DECREMENT HI BYTE
         DEC
DECBF1: ORA ICBLLZ+1 ;SET Z IF BOTH ARE ZERO
        RTS
; INCBFP -- INCREMENT WORKING BUFFER POINTER
INCBFP: INC ICBALZ ;BUMP LOW BYTE
        BNE INCBF1
                          ;GO IF NOT ZERO
         INC ICBALZ+1 ; ELSE, BUMP HI BYTE
INCBF1: RTS
; SUBBFL -- SET BUFFER LENGTH = BUFFER LENGTH - WORKING BYTE COUNT
SEC
         LDA ICBLL,X ;GET LOW BYTE OF INITIAL LENGTH
SBC ICBLLZ ;SUBTRACT FINAL LOW BYTE
STA ICBLLZ ;AND SAVE BACK
LDA ICBLH,X ;GET HI BYTE
SBC ICBLLZ+1
STA ICBLHZ
         RTS
;
; GOHAND -- GO INDIRECT TO A DEVICE HANDLER
; Y= STATUS ON RETURN, N FLAG=1 IF ERROR ON RETURN
GOHAND: LDY #FNCNOT ; PREPARE NO FUNCTION STATUS FOR HANDLER RTS
         JSR CIJUMP ;USE THE INDIRECT JUMP STY ICSTAZ ;SAVE STATUS
         CPY
               #0
                              ; AND SET N FLAG
         RTS
; INDIRECT JUMP TO HANDLER BY PAUL'S METHOD
CIJUMP: TAX
                            ;SAVE A
         LDA ICSPRZ+1 ;GET JUMP ADDRESS HI BYTE
                                ; PUT ON STACK
         PHA
               ICSPRZ
                              ;GET JUMP ADDRESS LO BYTE
         LDA
         PHA
                                ; PUT ON STACK
         TXA
                                ; RESTORE A
         RTS
                               GO TO HANDLER INDIRECTLY
         .PAGE
; DEVSRC -- DEVICE SEARCH, FIND DEVICE IN HANDLER ADDRESS TABLE
; LOOP TO FIND DEVICE
DEVSRC: LDY #0
                (ICBALZ), Y ; GET DEVICE NAME FROM USER
         LDA
         BEO
                CIERR2
         LDY #MAXDEV ; INITIAL COMPARE INDEX
      CMP HATABS,Y ; IS THIS THE DEVICE?
         BEQ DEVS2
                         ;YES
         DEY
         DEY
                                ; ELSE, POINT TO NEXT DEVICE NAME
```

; Z FLAG = 0 ON RETURN IF LENGTH = 0 AFTER DECREMENT

```
BPL
                DEVS1 ; CONTINUE FOR ALL DEVICES
; NO DEVICE FOUND, DECLARE NON-EXISTENT DEVICE ERROR
CIERR2: LDY #NONDEV ; ERROR CODE
         SEC
                               ;SHOW ERROR
                DEVS4 ; AND RETURN
         BCS
; FOUND DEVICE, SET ICHID, ICDNO, AND INIT DEVICE
DEVS2: TYA
         STA ICHIDZ ;SAVE HANDLER INDEX
         SEC
         LDY
                #1
               (ICBALZ),Y ;GET DEVICE NUMBER (DRIVE NUMBER)
#ASCZER ;SUBTRACT ASCII ZERO
#$A ;IS NUMBER IN RANGE?
         LDA
         SBC
         CMP
               #$A ;I
DEVS3 ;YES
         BCC
                               ; NO. DEFAULT TO ONE
         LDA
                #1
DEVS3: STA ICDNOZ ;SAVE DEVICE NUMBER
        CLC
                                ;SHOW NO ERROR
; RETURN
DEVS4: RTS
         .PAGE
; CIO ROM TABLES
; COMMAND TABLE
; MAPS EACH COMMAND TO OFFSET FOR APPROPRIATE VECTOR IN HANDLER
COMTAB: BYTE 0,4,4,4,6,6,6,6,6,2,8,10
LENGTH = *-CIOINT
CRNTP1 =*
         *=$14
CIOSPR: BYTE INTORG-CRNTP1 ; GCIOL IS TOO LONG
         .TITLE 'INTERRUPT HANDLER'
;LIVES ON DK1:INTHV.SRC
SRTIM2 = 6
                            ;SECOND REPEAT INTERVAL
; THIS IS TO MAKE DOS 2 WORK WHICH USED AN ABSOLUTE ADDRESS
         *=$E912
         JMP SETVBL
         *=SETVBV
         JMP SETVBL
         JMP
                SYSVBL
         JMP XITVBL
         *=INTINV
         JMP IHINIT
         *=VCTABL+INTABS-VDSLST
          .WORD SYRTI
                           ;VDSLST
         .WORD SYIRQB
                         ; VPRCED
          .WORD SYIROB
                          ; VINTER
          .WORD SYIRQB
                          ; VBREAK
```

DEY

```
.RES 8
          .WORD SYIRQB
                          ;VTMIR1
          .WORD SYIRQB
                          ;VTIMR2
          .WORD SYIRQB ;VTMIR4
.WORD SYIRQ ;VIMIRQ
                          ;VIMIRQ
          .WORD 0,0,0,0,0 ;CDTMV1-4
          .WORD SYSVBL
                          ;VVBLKI
                          ;VVSLKD
          .WORD XITVBL
         *=$900C
                           ;SET UP RAM VECTORS FOR LINBUG VERSION
         LDA
                #PIROH
         STA
                $FFF9
               #PIRQL
         LDA
         STA
                $FFF8
         LDA #PNMIH
STA $FFFB
LDA #PNMIL
STA $FFFA
         RTS
          .PAGE
; IRQ HANDLER
; JUMP THRU IMMEDIATE IRQ VECTOR, WHICH ORDINARILY POINTS TO
; SYSTEM IRO; DETERMINE & CLEAR CAUSE, JUMP THRU SOFTWARE VECTOR.
         *=INTORG
IHINIT: LDA #$40 ;VBL ON BUF DLIST OFF***FOR NOW***
         STA NMIEN ;ENABLE DISPLAY LIST, VERTICAL BLANK
                            ;LOOK AT DATA DIRECTION REGISTERS IN PIA
         LDA
                #$38
         STA
                PACTL
         STA PBCTL LDA #0
                               ; MAKE ALL INPUTS
         STA PORTA
STA PORTB
LDA #$3C
                           ;BACK TO PORTS
         STA
                PACTL
                PBCTL
         STA
         RTS
PIRQ: JMP (VIMIRU)
CMPTAB: .BYTE $80
                          ;BREAK KEY
         .BYTE $40
                          ;KEY STROKE
          .BYTE $04
                          ;TIMER 4
          .BYTE $02
                          ;TIMER 2
         .BYTE $01
                          ;TIMER 1
          .BYTE $08
                          ;SERIAL OUT COMPLETE
          .BYTE $10
                          ;SERIAL OUT READY
          .BYTE $20
                          ;SERIAL IN READY
; THIS IS A TABLE OF OFFSETS INTO PAGE 2. THEY POINT TO
ADRTAB: .BYTE BRKKY-INTABS
          .BYTE VKEYBD-INTABS
          .BYTE VTIMR4-INTABS
          .BYTE VTIMR2-INTABS
          .BYTE VTIMR1-INTABS
```

```
.BYTE VSERIN-INTABS
SYIRO: PHA
                           ; SAVE ACCUMULATOR
                        ; CHECK FOR SERIAL IN
         LDA
               IROST
         AND
               #$20
         BNE
               SYIRQ2
         LDA
               #$DF
                          ; MASK ALL OTHERS
              IRQEN
         STA
               POKMSK
         LDA
         STA
               IRQEN
         JMP
               (VSERIN)
                          ; PUT X INTO ACC
SYIRQ2: TXA
        PHA
                              ;SAVE K ONTO STACK
         LDX #$6
                             ;START WITH SIX OFFSET
LOOPM: LDA CMPTAB, X ; LOAD MASK
         CPX #5
                             CHECK TO SEE IF COMPLETE IS SET
         BNE
                LOOPM2
         AND
               POKMSK
                          ; IS THIS INTERUPT ENABLED?
         BEO
               T.T.
LOOPM2: BIT IRQST ; IS IT THE INTERUPT?
         BEQ JMPP
LL: DEX
                           ; NO DEC X AND TRY NEXT MASK
        BPL LOOPM
JMP SYIRQ8
                         ; IF NOT NEC 0010 LOOPH
                            ; DONE BUT NO INTERUPT
       EOR #$FF ; COMPLEMENT MASK
JMPP:
             IRQEN
         STA
                         ; ENABLE ALL OTHERS
         LDA
               POKMSK
                            ; GET POKE MASK
               IROEN ; ENABLE THOSE IN POKE MASK
         STA
         LDA
               ADRTAB,X
         TAX
        LDA INTABS,X ; GET ADDRESS LOW PART
STA JVECK ; PUT IN VECTOR
LDA INTABS+1,X ; GET ADDRESS HIGH PART
         STA
               JVECK+1 ; PUT IN VECTOR HIGH PART
         PLA
                               ; PULL X REGISTER FROM STACK
         TAX
                               ; PUT IT INTO X
         JMP (JVECK)
                          ; JUMP TO THE PROPER ROUTINE
BRKKY2: LDA #0
                          ; BREAK KEY ROUTINE
         STA
              BRKKEY
                            ; SET BREAK KEY FLAG
         STA
                SSFLAG
                            ; START/STOP FLAG
         STA
               CRSINH
                            ; CURSOR INHIBIT
         STA ATRACT
                            ; TURN OFF ATRACT MODE
         PLA
         RTI
                              ;EXIT FROM INT
SYIRQ8: PLA
         TAX
         BIT
               PACTL ; PROCEED ***I GUESS***
         BPL
               SYIRQ9
         LDA
               PORTA
                         CLEAR INT STATUS BIT
              (VPRCED)
         JMP
SYIRO9: BIT PBCTL ;INTERRUPT ***I GUESS***
         BPL SYIROA
               PORTB ; CLEAR INT STATUS
         LDA
         JMP (VINTER)
SYIRQA: PLA
         STA
              JVECK
```

.BYTE VSEROC-INTABS
.BYTE VSEROR-INTABS

PLA

```
PHA
         AND
               #$10
                        ;B BIT OF P REGISTER
         BEQ
               SYRTI2
         LDA JVECK
         PHA
         JMP (VBREAK)
SYRTI2: LDA JVECK
        PHA
SYIRQB: PLA
SYRTI: RTI
                      ;UNIDENTIFIED INTERRUPT, JUST RETURN
         .PAGE
; NMI HANDLER
; DETERMINE CAUSE AND JUMP THRU VECTOR
     BIT NMIST
PNMI:
        BPL PNMI1 ;SEE IF DISPLAY LIST
         JMP
               (VDSLST)
PNMI1: PHA
         LDA NMIST AND #$20
                          ;SEE IF RESET
         BEO
               *+5
                            ;DO THRU WARM START JUMP
         JMP
               WARMSV
         TXA
                               ;SAVE REGISTERS
         PHA
         TYA
         PHA
         STA
               NMIRES
                             ; RESET INTERRUPT STATUS
               (VVBLKI) ;JUMP THRU VECTOR
         JMP
         .PAGE
; SYSTEM VBLANK ROUTINE
; INC FRAME COUNTER. PROCESS COUNTDOWN TIMERS. EXIT IF I WAS SET. CLEAR
; SET DLISTL, DLISTH, DMACTL FROM RAM CELLS. DO SOFTWARE REPEAT.
SYSVBL: INC RTCLOK+2 ; INC FRAME COUNTER
         BNE
             SYSVB1
               ATRACT ; INCREMENT ATRACT (CAUSES ATRACT WHEN MINUS)
         INC
         INC
               RTCLOK+1
         BNE
               SYSVB1
               RTCLOK
         INC
SYSVB1: LDA #$FE ;{ATRACT] SET DARK MASK TO NORMAL LDX #0 ;SET COLRSH TO NORMAL
               ATRACT
         LDY
                             ;TEST ATRACT FOR NEGATIVE
               VBATRA
ATRACT
         BPL
                             ; WHILE POSITIVE DONT GO INTO ATRACT
         STA
                             ; IN ATRACTI SO STAY BY STA $FE
               RTCLOK+1 ; COLOR SHIFT FOLLOWS RICLOK+1
         LDX
             #$F6 ;SET DARK MASK TO DARK
         LDA
VBATRA: STA DRKMSK
         STX COLRSH
                              ; POINT TO TIMER1
         LDX
               #0
               DCTIMR ;GO DECREMENT TIMER1
SYSVB2 ;BRANCH IF STILL COU
         JSR
         BNE
                            BRANCH IF STILL COUNTING
             JTIMR1
         JSR
                             ;GO JUMP TO ROUTINE
SYSVB2: LDA CRITIC
         BNE XXIT ;GO IF CRITICAL SET
         TSX
                               ;SEE IF I WAS SET
```

```
$104,X
#$04
                            GET STACKED P
                          ;I BIT
         AND
                #$04
         BEQ
               SYSVB3
                            ;BRANCH IF OK
XXIT: JMP XITVBL ; I WAS SET, EXIT
SYSVB3: LDA PENV
         STA LPENV
               PENH
         LDA
         STA
               LPENH
         LDA
               SDLSTH
              DLISTH
SDLSTL
         STA
         LDA
         STA
               DLISTL
         LDA
               SDMCTL
              DMACTL
GPRIOR
PRIOR
#$08
         STA
                            ;GLOBAL PRIOR
         LDA
         STA
                          ;TURN OFF KEYBOARD SPEAKER
         LDX
         STX
               CONSOL
SCOLLP: CLI
                           ; DISABLE INTERUPTS
        LDA PCOLRO, X ; LOAD COLOR REGISTERS FROMRAM
EOR COLRSH ; DO COLOR SHIFT
              DRKMSK
         AND
                             ; AND DARK ATRACT
         STA
               COLPM0,X
         DEX
         BPL
               SCOLLP
         LDA
               CHBAS
              CHBASE
CHACT
         STA
         LDA
         STA
               CHACTL
         LDX
               #2
                             ; POINT TO TIMER 2
         JSR
               DCTIMR
         BNE
               SYSVB4
                            ; IF DIDNT GO ZERO
         JSR JTIMR2
                            ;GO JUMP TO TIMER2 ROUTINE
SYSVB4: LDX #2
                          ; RESTORE X
SYSVBB: INX
         INX
         LDA
               CDTMV1,X
         ORA
               CDTMV1+1,X
         BEO
               SYSVBA
                        ; DECREMENT AND SET FLAG IF NONZERO
              DCTIMR
CDTMF3-4,X
         JSR
         STA
SYSVBA: CPX #8
                          ;SEE IF DONE ALL 3
         BNE
               SYSVBB
                            ;LOOP
; CHECK DEBOUNCE COUNTER
             SKSTAT
         LDA
         AND
               #$04
                          KEY DOWN BIT
               SYVB6A
                            ; IF KEY DOWN
         BEQ
; KEY UP SO COUNT IT
                            ;KEY DELAY COUNTER
         LDA KEYDEL
         BEO
               SYVB6A
                             ; IF COUNTED DOWN ALREADY
                             ; COUNT IT
         DEC
               KEYDEL
; CHECK SOFTWARE REPEAT TIMER
SYVB6A: LDA SRTIMR
         BEQ SYSVB7
                            ;DOESN'T COUNT
         LDA
               SKSTAT
         AND
               #$04
                           ; CHECK KEY DOWN BIT
               SYSVB6
                            ;BRANCH IF NO LONGER DOWN
         BNE
         DEC
               SRTIMR
                             ; COUNT FRAME OF KEY DOWN
         BNE
                SYSVB7
                             ;BRANCH IF NOT RUN OUT
```

LDA

```
; TIMER RAN OUT - RESET AND SIMULATE KEYBOARD IRQ
              #SRTIM2 ;TIMER VALUE
         LDA
                            ;SET TIMER
         STA
               SRTIMR
         LDA
               KBCODE
                            GET THE KEY
                             ; PUT INTO CH
         STA
               CH
; READ GAME CONTROLLERS
SYSVB7: LDY #1
        LDX
               #3
STLOOP: LDA PORTA, Y
         LSR A
               Α
         LSR
         LSR
               Α
         LSR
        STA
               STICKO,X ;STORE JOYSTICK
         DEX
             PORTA,Y
         LDA
         AND
               #$F
         STA
               STICKO,X ;STORE JOYSTICK
         DEX
        DEY
         BPL STLOOP
;
        LDX #3
STRL: LDA TRIGO,X
                      ; MOVE JOYSTICK TRIGGERS
         STA STRIGO,X
        LDA POT0,X
STA PADDL0,X
                            ; MOVE POT VALUES
              POT4,X
        LDA
         STA
               PADDL4,X
         DEX
        BPL STRL STA POTGO
                        ;START POTS FOR NEXT TIME
        LDX #6
        LDY
               #3
PTRLP: LDA STICKO,Y ;TRANSFER BITS FROM JOYSTICKS
                          ;TO PADDLE TRIGGERS
        LSR A
         LSR
               A
         LSR
               Α
              PTRIG1,X
         STA
         LDA
               #0
               A
         ROL
        STA
               PTRIG0,X
         DEX
         DEX
        DEY
         \mathtt{BPL}
               PTRLP
        JMP
               (VVBLKD) ;GO TO DEFERRED VBLANK ROUTINE
SV7H
       =
               SYSVB7/256
               (-256)*SV7H+SYSVB7
SV7L
       =
SYSVB6: LDA #0
                           ;ZERO TIMER
         STA
               SRTIMR
         BEQ
                            ; UNCOND
               SYSVB7
JTIMR1: JMP (CDTMA1)
JTIMR2: JMP (CDTMA2)
; SUBROUTINE TO DECREMENT A COUNTDOWN TIMER
; ENTRY X=OFFSET FROM TIMER 1
```

```
; EXIT A, P=ZERO IF WENT ZERO, FF OTHERWISE
DCTIMR: LDY CDTMV1,X ;LO BYTE
         BNE DCTIM1 ; NONZERO, GO DEC IT
               CDTMV1+1,X ;SEE IF BOTH ZERO
         LDY
               DCTXF ;YES, EXIT NONZERO
         BEO
               CDTMV1+1,X ;DEC HI BYTE
         DEC
DCTIM1: DEC CDTMV1,X ;DEC LO BYTE
         BNE DCTXF
         LDY
               CDTMV1+1,X
               DCTXF
         BNE
         LDA
               #0
                              ; WENT ZERO. RETURN ZERO
         RTS
DCTXF: LDA #$FF ;RETURN NONZERO
        RTS
         .PAGE
; SUBROUTINE TO SET VERTICAL BLANK VECTORS AND TIMERS
; ENTRY X=HI,Y=LO BYTE TO SET
        A= 1-5 TIMERS 1-5
           6 IMM VBLANK
           7 DEF VBLANK
SETVBL: ASL A
                       ; MUL BY 2
         STA INTEMP
         TXA
         LDX
               #5
         STA WSYNC ; WASTE 20 CPU CYCLES
SETLOP: DEX
                          ;TO ALOWD VBLANK TO HAPPEN
              SETLOP
INTEMP
CDTMV1-1,X
         BNE
                             ; IF THIS IS LINE "7C"
         LDX
         STA
         TYA
         STA CDTMV1-2,X
         RTS
; EXIT FROM VERTICAL BLANK
XITVBL: PLA
                         ;UNSTACK Y
         TAY
                               ;UNSTACK X
         PLA
         TAX
         PLA
                               ;UNSTACK A
                               ; AND GO BACK FROM WHENCE.
        RTI
PIRQH = PIRQ/256
PIRQL =
              (-256)*PIRQH+PIRQ
PNMIH =
              PNMI/256
PNMIL
               (-256)*PNMIH+PNMI
; SPARE BYTE OR MODULE TOO LONG FLAG
CRNTP2 =*
         *=$14
INTSPR: .BYTE SIOORG-CRNTP2 ; GINTHV IS TOO LONG
         .TITLE 'SIO ( SERIAL BUS INPUT/OUTPUT CONTROLLER )'
       COLLEEN OPERATING SYSTEM
        SIO ( SERIAL BUS INPUT/OUTPUT CONTROLLER )
       WITH SOFTWARE BAUD RATE CORRECTION ON CASSETTE
```

```
AL MILLER 3-APR-19
; THIS MODULE HAS ONE ENTRY POINT. IT IS CALLED BY THE DEVICE
; HANDLERS. IT INTERPRETS A PREVIOUSLY ESTABLISHED DEVICE CONTROL
; BLOCK (STORED IN GLOBAL RAM) TO ISSUE COMMANDS
; TO THE SERIAL BUS TO CONTROL TRANSMITTING AND RECEIVING DATA.
        .PAGE
; EQUATES
; DCD DEVICE BUS ID NUMBERS
          $30
FLOPPY =
;PRINTR =
              $40
;CASSET =
              $80
                             ;!!!!! ****
CASET =
                             ;!!!!! ****
              $60
;BUS COMMANDS
READ
               'R
WRITE =
               ' W
;STATIS = 'S
;FORMAT = '!
; COMMAND AUX BYTES
                     ; PRINT 18 CHARACTERS SIDEWAYS
SIDWAY =
              'S
NORMAL =
               'N
                          ; PRINT 40 CHARACTERS NORMALLY
              'D
                         ; PRINT 20 CHARACTERS DOUBLE WIDE
DOUBLE =
              ' P
PLOT =
                          ; PLOT MODE
; BUS RESPONSES
          'A
ACK =
                     ; DEVICE ACKNOWLEDGES INFORMATION
            'N
NACK
                       ; DEVICE DID NOT UNDERSTAND
COMPLT =
               ' C
                          ; DEVICE SUCCESSFULLY COMPLETED OPERATION
                          ; DEVICE INCURRED AN ERROR IN AN ATTEMPTED OP
               ' E
ERROR =
;
; MISCELLANEOUS EQUATES
             $28
B192LO =
                             ;19200 BAUD RATE POKEY COUNTER VALUES (LO BY
              $00
B192HI =
                             ; (HI BYTE)
B600LO =
              $CC
                             ;600 BAUD (LO BYTE)
B600HI =
              $05
                             ;(HI BYTE)
              $05
HITONE =
                             ;FSK HI FREQ POKEY COUNTER VALUE (5326 HZ)
LOTONE =
              $07
                             ;FSK LO FREQ POKEY COUNTER VALUE (3995 HZ)
       .IF PALFLG
               150
                             ; WRITE INTER RECORD GAP (IN 1/60 SEC)
WIRGLO =
RIRGLO =
              100
                             ; READ INTER RECORD GAP (IN 1/60 SEC)
WSIRG =
               13
                             ; SHORT WRITE INTER RECORD GAP
```

```
RSIRG =
                            ;SHORT READ INTER RECORD GAP
        .ENDIF
        .IF PALFLG-1
WIRGLO =
              180
                            ; WRITE INTER RECORD GAP (IN 1/60 SEC)
                            ; READ INTER RECORD GAP (IN 1/60 SEC)
RIRGLO =
              120
WSIRG =
               15
                         ;SHORT WRITE INTER RECORD GAP
                         ;SHORT READ INTER RECORD GAP
RSIRG =
               10
       .ENDIF
WIRGHI =
              0
RIRGHI =
           0
;
NCOMLO =
              $34
                             ; PIA COMMAND TO LOWER NOT COMMAND LINE
NCOMHI =
               $3C
                             ; PIA COMMAND TO RAISE NOT COMMAND LINE
                             ; PIA COMMAND TO TURN ON CASSETTE MOTOR
MOTRGO =
               $34
                             ; PIA COMMAND TO TURN OFF MOTOR
MOTRST =
               $3C
TEMPHI = TEMP/256 ; ADDRESS OF TEMP CELL (HI BYTE)
TEMPLO =
              (-256)*TEMPHI+TEMP ; (LO BYTE)
              CDEVIC/256 ; ADDRESS OF COMMAND BUFFER (HI BYTE)
CBUFHI =
CBUFLO =
              (-256)*CBUFHI+CDEVIC ;(LO BYTE)
            13
CRETRI =
                          ; NUMBER OF COMMAND FRAME RETRIES
DRETRI =
              1
                          ; NUMBER OF DEVICE RETRIES
              2
CTIMLO =
                          ; COMMAND FRAME ACK TIME OUT (LO BYTE)
CTIMHI =
              0
                           ; COMMAND FRAME ACK TIME OUT (HI BYTE)
;JTADRH = JTIMER/256 ;HI BYTE OF JUMP TIMER ROUTINE ADDR
;JTADRL =
             (-256)*JTADRH+JTIMER ;"MOVED TO LINE 1428"
        .PAGE
        STO
         *=SIOV
         JMP
               SIO
                            ;SIO ENTRY POINT
         *=SIOINV
         JMP
                          ;SIO INITIALIZATION ENTRY POINT
              SIOINT
         *=SENDEV
         JMP SENDEN
                            ; SEND ENABLE ENTRY POINT
         *=VCTABL-INTABS+VSERIN
                        ;VSERIN
         .WORD ISRSIR
         .WORD ISRODN
                         ; VSEROR
         .WORD ISRTD ; VSEROC
         *=SIOORG
; SIO INITIALIZATION SUBROUTINE
SIOINT: LDA #MOTRST
         STA PACTL
                         ;TURN OFF MOTOR
         LDA
               #NCOMHI
```

```
STA
               PBCTL ; RAISE NOT COMMAND LINE
         LDA
                #3
         STA
                SSKCTL
                             GET POKEY OUT OF INITIALIZE MODE
               SOUNDR
                             ; INIT POKE ADDRESS FOR QUIET I/O
         STA
         STA
               SKCTL
;
         RTS
                               ; RETURN
SIO:
       TSX
         STX
               STACKP
                            ; SAVE STACK POINTER
         LDA
                #1
         STA
               CRITIC
             DDEVIC
         LDA
         CMP
               #CASET
                             ; BRANCH IF NOT CASSETTE
         BNE
                NOTCST
         JMP
                CASENT
                             ;OTHERWISE JUMP TO CASSETTE ENTER
; ALL DEVICES EXCEPT CASSETTE ARE INTELLIGENT
NOTCST: LDA #0
               CASFLG
                             ; INIT CASSETTE FLAG TO NO CASSETTE
         STA
         LDA
                #DRETRI ;SET NUMBER OF DEVICE RETRIES
         STA
                DRETRY
                       ;SET NUMBER OF COMMAND FRAMERETRIES
COMMND: LDA #CRETRI
               CRETRY
         STA
; SEND A COMMAND FRAME
COMFRM: LDA #B192LO ;SET BAUD RATE TO 19200
         STA
               AUDF3
         LDA
                #B192HI
         STA
                AUDF4
         CLC
                               ;SET UP COMMAND BUFFER
             DDEVIC
         LDA
         ADC
               DUNIT
         ADC
                #$FF
                           ;SUBTRACT 1
                CDEVIC
         STA
                             ;SET BUS ID NUMBER
             DCOMND
         LDA
         STA
                             ;SET BUS COMMAND
               CCOMND
         LDA
               DAUX1
                          ;STORE COMMAND FRAME AUX BYTES 1 AND 2
         STA
                CAUX1
         LDA
               DAUX2
         STA
                CAUX2
                      ; DONE SETTING UP COMMAND BUFFER
;
         CLC
                               ;SET BUFFER POINTER TO COMMAND FRAME BUFFER
         LDA
               #CBUFLO
```

```
STA
               BUFRLO
                           ; AND BUFFER END ADDRESS
        ADC
               #4
        STA
               BFENLO
        LDA
              #CBUFHI
             BUFRHI
BFENHI
        STA
                           ; DONE SETTING UP BUFFER POINTER
        STA
        LDA
              #NCOMLO
PBCTL
                         ;LOWER NOT COMMAND LINE
        STA
        JSR SENDIN ; SEND THE COMMAND FRAME TO A SMART DEVICE
        LDA
              ERRFLG
                            ;BRANCH IF AN ERROR RECEIVED
        BNE
               BADCOM
        TYA
        BNE ACKREC ; BRANCH IF ACK RECEIVED
;
BADCOM: DEC CRETRY
                        ; A NACK OR TIME OUT OCCURED
        BPL COMFRM
                          ; SO BRANCH IF ANY RETRIES LEFT
        JMP DERR1 ;OTHERWISE, JUMP TO RETURN SECTION
;
;
ACKREC: LDA DSTATS
                        ; ACK WAS RECEIVED
               WATCOM
                            ; BRANCH TO WAIT FOR COMPLETE
        BPI.
; IF THERE IS NO DATA TO BE SENT
; SEND A DATA FRAME TO PERIPHERAL
              #CRETRI ;SET NUMBER OF RETRIES
        LDA
        STA
              CRETRY
                           ;LOAD BUFFER POINTER WITH DCB INFORMATION
        JSR LDPNTR
        JSR SENDIN ;GO SEND THE DATA FRAME TO A SMART DEVICE
              BADCOM
                            ;BRANCH IF BAD
        BEQ
; WAIT FOR COMPLETE SIGNAL FROM PERIPHERAL
WATCOM: JSR STTMOT ;SET DDEVICE TIME OUT VALUES IN Y,X
;
        LDA
               #$00
              ERRFLG ; CLEAR ERROR FLAG
        STA
        JSR
              WAITER
                            ;SET UP TIMER AND WAIT
                         ;BRANCH IF TIME OUT
        BEQ
               DERR
; DEVICE DID NOT TIME OUT
        BIT
               DSTATS
        BVS
               MODATA
                            ; BRANCH IF MORE DATA FOLLOWS
```

```
ERRFLG
         LDA
        BNE DERR1 ;BRANCH IF AN ERROR OCCURRED BEQ RETURN ;OTHERWISE RETURN
; RECEIVE A DATA FRAME FROM PERIPHERAL
MODATA: JSR LDPNTR ;LOAD BUFFER POINTER WITH DCB INFORMATION
        JSR
               RECEIV
                            ;GO RECEIVE A DATA FRAME
DERR: LDA ERRFLG
        BEQ NOTERR ; BRANCH IF NO ERROR PRECEEDED DATA
                        GET TEMP STATUS
        LDA
              TSTAT
               STATUS
        STA
                            ;STORE IN REAL STATUS
NOTERR: LDA STATUS
        CMP #SUCCES
                           ; BRANCH IF COMPLETELY SUCCESSFUL
        BEQ
               RETURN
DERR1: DEC DRETRY
        BMI RETURN ;BRANCH IF OUT OF DEVICE RETRIES
        JMP COMMND
                           ;OTHERWISE ONE MORE TIME
RETURN: JSR SENDDS ; DISABLE POKEY INTERRUPTS
        LDA #0
         STA
               CRITIC
                            ; RETURN STATUS IN Y
         LDY
              STATUS
        STY DSTATS
RTS RETURN
                            ; AND THE DCB STATUS WORD
; WAIT SUBROUTINE
; WAITS FOR COMPLETE OR ACK
; RETURNS Y=$FF IF SUCCESSFUL, Y=$00 IF NOT
;
WAIT:
      LDA #$00
         STA ERRFLG ; CLEAR ERROR FLAG
         CLC
                              ;LOAD BUFFER POINTER WITH ADDRESS
                         OF TEMPORARY RAM CELL
               #TEMPLO
         LDA
         STA
               BUFRLO
         ADC
               #1
              BFENLO
         STA
                          ;ALSO SET BUFFER END +1 ADDRESS
              #TEMPHI
         LDA
         STA
              BUFRHI
         STA
               BFENHI
                            ; DONE LOADING POINTER
```

```
LDA
               #$FF
               NOCKSM
        STA
                           ;SET NO CHECKSUM FOLLOWS DATA FLAG
        JSR
            RECEIV
                          GO RECEIVE A BYTE
        LDY
              #$FF
                         ; ASSUME SUCCESS
        LDA
               STATUS
        CMP
               #SUCCES
            NWOK
        BNE
                      BRANCH IF IT DID NOT WORK OK
;
WOK:
      LDA
               TEMP
                         ; MAKE SURE THE BYTE SUCCESSFULLY RECEIVED
        CMP
                         ; WAS ACTUALLY AN ACK OR COMPLETE
               #ACK
        BEQ
              GOOD
        CMP
              #COMPLT
        BEO
               GOOD
        CMP
               #ERROR
        BNE
               NOTDER
                         BRANCH IF DEVICE DID NOT SEND BACK
; A DEVICE ERROR CODE
              #DERROR
        LDA
              STATUS
                           ;SET DEVICE ERROR STATUS
        STA
        BNE
              NWOK
NOTDER: LDA #DNACK ;OTHERWISE SET HACK STATUS
       STA STATUS
NWOK:
      LDA STATUS
        CMP #TIMOUT
        BEQ
              BAD
                           ;BRANCH IF TIME OUT
        LDA
              #$FF
        STA
               ERRFLG
                           ;SET SOME ERROR FLAG
                         ; RETURN WITH OUT SETTING Y = 0
        BNE
               GOOD
BAD:
    LDY #0
GOOD: LDA STATUS
        STA TSTAT
        RTS
                             ; RETURN
; SEND SUBROUTINE
; SENDS A BUFFER OF BYTES OUT OVER THE SERIAL BUS
SEND:
      LDA #SUCCES ; ASSUME SUCCESS
        STA STATUS
        JSR SENDEN
                           ; ENABLE SENDING
        LDY
               #0
```

```
STY
               CHKSUM
                            ;CLEAR CHECK SUM
         STY
                CHKSNT
                             ; CHECKSUM SENT FLAG
                             ;TRANSMISSION DONE FLAG
         STY
                XMTDON
              (BUFRLO), Y ; PUT FIRST BYTE FROM BUFFER
         LDA
         STA
                            ; INTO THE SERIAL OUTPUT REGISTER
                SEROUT
;
         STA CHKSUM ; PUT IT IN CHECKSUM
NOTDON: LDA BRKKEY
         BNE
               NTBRKO
               BROKE
                         ;JUMP IF BREAK KEY PRESSED
         JMP
NTBRKO: LDA XMTDON ;LOOP UNTIL TRANSMISSION IS DONE
         BEQ
               NOTDON
;
         JSR
               SENDDS
                            ; DISABLE SENDING
         RTS RETURN
; OUTPUT DATA NEEDED INTERRUPT SERVICE ROUTINE
ISRODN: TYA
                               ; SAVE Y REG ON STACK
         PHA
         INC BUFRLO ; INCREMENT DUFFER POINTER
         BNE
               NOWRPO
         INC
               BUFRHI
NOWRPO: LDA BUFRLO ; CHECK IF PAST END OF BUFFER
         CMP BFENLO
               BUFRHI
         LDA
                            ;HIGH PART
         SBC
               BFENHI
         BCC
               NOTEND
                            ;BRANCH IF NOT PAST END OF BUFFER
         LDA CHKSNT
BNE RELONE
                             ; BRANCH IF CHECKSUM ALREADY SENT
;
              CHKSUM
         LDA
         STA
               SEROUT
                             ; SEND CHECK SUM
         LDA
               #$FF
         STA
               CHKSNT
                             ;SET CHECKSUM SENT FLAG
         BNE CHKDON
RELONE: LDA POKMSK
                        ; ENABLE TRANSMIT DONE INTERRUPT
         ORA #$08
         STA
               POKMSK
         STA
               IRQEN
CHKDON: PLA
         TAY
                               ; RESTORE Y REG
         PLA
                               ; RETURN FROM INTERRUPT
```

```
NOTEND: LDY #0
               (BUFRLO),Y ; PUT NEXT BYTE FROM BUFFER
         LDA
         STA
               SEROUT
                            ; INTO THE SERIAL OUTPUT REGISTER
         CLC
                               ; ADD IT TO CHECKSUM
         ADC
               CHKSUM
         ADC
                #0
             CHKSUM
         STA
         JMP
               CHKDON
                            ;GO RETURN
; TRANSMIT DONE INTERRUPT SERVICE ROUTINE
ISRTD: LDA CHKSNT
        BEQ FOOEY ; BRANCH IF CHECKSUM NOT YET SENT
         STA
               XMTDON
                             ;OTHERWISE SET TRANSMISSION DONE FLAG
         LDA POKMSK
                           ; DISABLE TRANSMIT DONE INTERRUPT
         AND
               #$F7
         STA
               POKMSK
         STA
               IRQEN
FOOEY: PLA
                          ; RETURN FROM INTERRUPT
         RTI
; RECEIVE SUBROUTINE
RECEIV: LDA #0
         LDY
               CASFLG
         BNE
               NOCLR
                          ;BRANCH IF CASSETTE
         STA
               CHKSUM
                            ;CLEAR CHKSUM
NOCLR: STA BUFRFL
STA RECVDN
                        ;BUFFER FULL FLAG
         STA RECVDN
                             ; RECEIVE DONE FLAG
;
         LDA
               #SUCCES
                            ;SET GOOD STATUS FOR DEFAULT CASE.
         STA
                STATUS
         JSR
               RECVEN
                            ;DO RECEIVE ENABLE
         LDA
               #NCOMHI
                          ; COMMAND FRAME HI COMMAND
         STA
                PBCTL
                          ;STORE IN PIA
```

RTI

```
CHKTIM: LDA BRKKEY
        BNE
             NTBRK1
              BROKE ; JUMP IF BREAK KEY PRESSED
        JMP
NTBRK1: LDA TIMFLG
                     ;NO,
        BEQ TOUT
                        ; IF TIMEOUT, GO SET ERROR STATUS
        LDA
              RECVDN
        BEQ
                           ;DONE ?
              CHKTIM
GOBACK: RTS
TOUT: LDA #TIMOUT ; YES,
        STA STATUS ;SET TIMEOUT STATUS
RRETRN: RTS
                         ; RETURN
; SERIAL INPUT READY INTERRUPT SERVICE ROUTINE
ISRSIR: TYA
                             ; SAVE Y REG ON STACK
        PHA
        LDA SKSTAT
               SKRES
                        ; RESET STATUS REGISTER
        STA
 ****** THIS MAY NOT BE THE PLACE TO DO IT ******
        BMI NTFRAM
                           ; BRANCH IF NO FRAMING ERROR
        LDY #FRMERR
                           ;SET FRAME ERRORR STATUS
        STY
              STATUS
NTFRAM: AND #$20
        BNE NTOVRN
                           ;BRANCH IF NO OVERRUN ERROR
        LDY #OVRRUN
        STY
              STATUS
                           ;SET OVERRUN ERROR STATUS
NTOVRN: LDA BUFRFL
        BEQ NOTYET
                           ; BRANCH IF BUFFER WAS NOTYET FILLED
        LDA SERIN ; THIS INPUT BYTE 15 THE CHECKSUM
        CMP
              CHKSUM
              SRETRN
        BEQ
                           ; BRANCH IF CHECKSUMS MATCH
        LDY #CHKERR
STY STATUS
                           ;SET CHECKSUM ERROR STATUS
SRETRN: LDA #$FF ;SET RECEIVE DONE FLAG
        STA RECVDN
```

```
SUSUAL: PLA
         TAY
                               ; RESTORE Y REG
         PLA
                               ; RETURN FROM INTERRUPT
         RTI
;
NOTYET: LDA SERIN
         LDY
                #0
         STA
               (BUFRLO), Y ; STORE INPUT REGISTER INTO BUFFER
         CLC
                               ; ADD IT TO CHECKSUM
         ADC
               CHKSUM
         ADC
                #0
             CHKSUM
         STA
         INC
               BUFRLO
                             ; INCREMENT BUFFER POINTER
         BNE
                NTWRP1
         INC
                BUFRHI
NTWRP1: LDA BUFRLO
         CMP BFENLO
         LDA
               BUFRHI
         SBC
               BFENHI
         DCC
               SUSUAL
                            ; BRANCH IF NEW BUFFER ADDRESS IS IN BUFFER L
             NOCKSM
         LDA
         BEO
               GOON
                          ; BRANCH IF A CHECKSUM WILL FOLLOW DATA
         LDA
               #0
         LDA
STA
               NOCKSM
                            ;CLEAR NO CHECKSUM FLAG
         BEQ SRETRN ;GO RETURN AND SET RECEIVE DONE FLAG
GOON:
       LDA #$FF
         STA BUFRFL ;SET BUFFER FULL FLAG
         BNE SUSUAL
                       ;GO RETURN
; LOAD BUFFER POINTER SUBROUTINE
; LOAD BUFFER POINTER WITH DCB BUFFER INFORMATION
LDPNTR: CLC
         LDA
               DBUFLO
         STA
               BUFRLO
         ADC
               DBYTLO
         STA
                             ;ALSO SET SUFFER END + 1 ADDRESS
               BFENLO
         LDA
               DBUFHI
```

```
STA
             BUFRHI
        ADC
               DBYTHI
        STA
               BFENHI
        RTS
                             ; RETURN
; CASSETTE HANDLING CODE
CASENT: LDA DSTATS
        BPL CASRED ; BRANCH IF INPUT FROM CASSETTE
;
; WRITE A RECORD
        LDA
              #B600LO ;SET BAUD RATE TO 600
        STA
               AUDF3
        LDA
              #B600HI
        STA
              AUDF4
;
        JSR SENDEN
                          ;TURN ON POKEY MARK TONE
                        ;LOAD SHORT WRITE INTER RECORD GAP TIME
        LDY #WSIRG
        LDA
              DAUX2
                           ;BRANCH IF SHORT GAP IS DESIRED
        BMI
              SRTIR0
        LDY
              #WIRGLO ;SET WRITE IRQ TIME
SRTIRO: LDX #WIRGHI
             SETVBX
        JSR
              #MOTRGO
        LDA
        STA
              PACTL
                       ;TURN ON MOTOR
TIMIT: LDA TIMFLG ;LOOP UNTIL DONE
        BNE TIMIT
                           ;LOAD BUFFER POINTER WITH DCB INFORMATION
        JSR
              LDPNTR
        JSR SEND
                      ;SEND A BUFFER
        JMP CRETRN
                          ;GO, RETURN
; RECEIVE A RECORD
CASRED: LDA #$FF
        STA CASFLG
                           ;SET SET CASSETTE FLAG
                           ;LOAD SHORT READ INTER RECORD GAP TIME
        LDY
              #RSIRG
        LDA
               DAUX2
        BMI
                           BRANCH IF SHORT GAP IS DESIRED
              SRTIR1
        LDY
              #RIRGLO ;SET TIME OUT FOR READ IRQ
```

```
SRTIR1: LDX #RIRGHI
        JSR SETVBX
        LDA
STA
              #MOTRGO
              PACTL ; TURN ON MOTOR
TIMIT1: LDA TIMFLG
                      ;LOOP UNTIL DONE
        BNE
             TIMIT1
        JSR LDPNTR ;LOAD BUFFER POINTER WITH DOS INFORMATION
        JSR STTMOT
                         ;SET DEVICE TIME OUT IN Y,X
        JSR
              SETVBX
        JSR BEGIN ; SET INITIAL BAUD RATE
        JSR RECEIV ;GO RECEIVE A BLOCK
CRETRN: LDA DAUX2
                       BRANCH IF DOING SHORT INTER RECORD GAPS
       BMI
              SRTIR2
; DON'T TURN OFF CASSETTE MOTOR
              #MOTRST
        LDA
        STA
              PACTL ;TURN OFF MOTOR
SRTIR2: JMP RETURN ;GO RETURN
JTIMER: LDA #$00
            JTIMER/256 ;HI BYTE OF JUMP TIMER ROUTINE ADDR
JTADRH =
JTADRL =
             (-256)*JTADRH+JTIMER
             TIMFLG ;SET TIME OUT FLAG
        STA
        RTS
; SEND ENABLE SUBROUTINE
SENDEN: LDA #$07 ; MASK OFF PREVIOUS SERIAL BUS CONTROL BITS
        AND SSKCTL
        ORA
              #$20
                        ;SET TRANSMIT MODE
        LDY
             DDEVIC
        CPY
              #CASET
            NOTCAS ; BRANCH IF NOT CASSETTE
        BNE
ï
        ORA
             #$08
                       ;SET THE FSK OUTPUT BIT
        LDY
              #LOTONE
                       ;SET FSK TONE FREQUENCIES
        STY
              AUDF2
        LDY
              #HITONE
        STY
              AUDF1
NOTCAS: STA SSKCTL ;STORE NEW VALUE TO SYSTEM MASK
```

```
STA
               SKCTL
                          ;STORE TO ACTUAL REGISTER
                #$C7
                          ; MASK OFF PREVIOUS SERIAL BUS INTERRUPT BITS
         LDA
         AND
                POKMSK
         ORA
                #$10
                          ; ENABLE OUTPUT DATA NEEDED INTERRUPT
         JMP
               CONTIN
                            GO CONTINUE IN RECEIVE ENABLE SUBROUTINE
; RECEIVE ENABLE SUBROUTINE
RECVEN: LDA #$07 ;MASK OFF PREVIOUS SERIAL BUS CONTROL BITS
             SSKCTL
         AND
         ORA
               #$10
                          ;SET RECEIVE MODE ASYNCH.
         STA
               SSKCTL
                            ;STORE NEW VALUE TO SYSTEM MASK
               SKCTL
         STA
                         ;STORE TO ACTUAL REGISTER
         STA SKRES ; RESET SERIAL PORT/KEYBOARD STATUS REGISTER
         LDA
              #$C7
                          ; MASK OFF PREVIOUS SERIAL BUS INTERRUPTBITS
         AND
                POKMSK
         ORA
                #$20
                          ; ENABLE RECEIVE INTERRUPT
CONTIN: STA POKMSK
                        ;STORE NEW VALUE TO SYSTEM MASK
               IRQEN
         STA
                         ;STORE TO ACTUAL. REGISTER
;
                #$28
                          ;CLOCK CH.3 WITH 1.79 MHZ
         LDA
                #$28
AUDCTL
         STA
                            ;CLOCK CH.4 WITH CH. 3
              #6
                             ;SET PURE TONES, NO VOLUME
         LDX
         LDA
               #$A8
         LDY
                SOUNDR
                            ;TEST QUIET I/O FLAG
         BNE
                NOISE1
                             ; NE IS NORMAL (NOISY)
                #$A0
         LDA
NOISE1: STA AUDC1,X
         DEX
         DEX
         BPL
               NOISE1
         LDA
               #$A0
         STA
               AUDC3 ; TURN OFF SOUND ON CHANNEL 3
         LDY
               DDEVIC
         CPY
               #CASET
               CAS31
                          BRANCH IF CASSETTE IS DESIRED
         BEQ
         STA
               AUDC1
                         ;OTHERWISE TURN OFF CHANNELS 1 AND 2
         STA
               AUDC2
CAS31: RTS
                         ; RETURN
```

```
; DISABLE SEND AND DISABLE RECEIVE SUBROUTINES
SENDDS: NOP
RECVDS: LDA #$C7 ;MASK OFF SERIAL BUS INTERRUPTS
        AND POKMSK
        STA
              POKMSK
                           ;STORE NEW VALUE TO SYSTEM MASK
        STA IRQEN ;STORE TO ACTUAL REGISTER
        LDX
               #6
        LDA
               #0
ZERIT: STA AUDC11X
        DEX
        DEX
        BPL ZERIT ;TURN OFF AUDIO VOLUME
        RTS
                             ; RETURN
; SET DDEVICE TIME OUT VALUES IN Y,X SUBROUTINE
                    GET DEVICE TIME OUT IN 1 SECOND INCR
STTMOT: LDA DTIMLO
                          ; PUT 6 HI BITS IN X, LO 2 BITS IN Y
        ROR
            A
        ROR
               A
                             ;TEMP SAVE
        TAY
            #$3F
        AND
                          ; MASK OFF 2 HI BITS
                             ;THIS IS HZ BYTE OF TIME OUT
        TAX
                             ; RESTORE
        TYA
        ROR
               A
                         ; MASK OFF ALL BUT 2 HI BITS
               #$C0
        AND
                             ;THIS IS LO BYTE OF TIME OUT
        TAY
        RTS
```

```
INTTBL: .WORD ISRSIR ;SERIAL INPUT READY .WORD ISRODN ;OUTPUT DATA NEEDED
         .WORD ISRTD
                           ;TRANSMISSION DONE
               ISRSIR/256 ;SERIAL INPUT READY ISR ADDRESS
SIRHI =
               (-256)*SIRHI+ISRSIR
SIRLO =
               ISRODN/256 ;OUTPUT DATA NEEDED ISR ADDRESS
ODNHI =
ODNLO =
               (-256)*ODNHI+ISRODN
TDHI =
               ISRTD/256
                          ;TRANSMISSION DONE ISR ADDRESS
               (-256)*TDHI+ISRTD
IDLO
      =
; SEND A DATA FRAME TO AN INTELLIGENT PERIPHERAL SUBROUTINE
SENDIN: LDX #$01
DELAYO: LDY #$FF
DELAY1: DEY
         BNE DELAY1
         DEX
         BNE
               DELAY0
;
         JSR SEND ;GO SEND THE DATA FRAME
         LDY #CTIMLO ;SET ACK TIME OUT
         LDX
                #CTIMHI
WAITER: JSR SETVBX
         JSR WAIT ; WAIT FOR ACK
                                 ; IF Y=0, A TIME OUT OR NACK OCCURED
         TYA
         RTS
                                 ; RETURN
; COMPUTE VALUE FOR POKEY FREQ REGS FOR THE BAUD RATE AS
; MEASURED BY AN INTERVAL OF THE 'VCOUNT' TIMER.
COMPUT: STA TIMER2
         STY TIMER2+1 ; SAVE FINAL TIMER VALUE
         JSR
                ADJUST
                             ; ADJUST VCOUNT VALUE
         STA TIMER2
LDA TIMER1
JSR ADJUST
STA TIMER1
                              ; SAVE ADJUSTED VALUE
                            ;ADJUST
                              ; SAVE ADJUSTED TIMER1 VALUE
                TIMER2
         LDA
         SEC
```

```
SBC
               TIMER1
         STA
                TEMP1
                         ;FIND VCOUNT DIFFERENCE
         LDA
               TIMER2+1
         SEC
             TIMER1+1
         SBC
         TAY
                               ;FIND VBLANK COUNT DIFFERENCE
              PALFLG
         .IF
               #-$9C
         LDA
HITIMR: CLC
         ADC #$9C
         .ENDIF
         .IF
              PALFLG-1
         LDA
               #-$83
HITIMR: CLC
         ADC #$83 ;ACCUMULATE MULTIPLICATION
         .ENDIF
         DEY
         BPL
               HITIMR
                             ; DONE?
         CLC
         ADC
               TEMP1
                         ;TOTAL VCOUNT DIFFERENCE
FINDX: TAY
                           ;SAVE ACCUM
            А
         LSR
         LSR
               A
         LSR
               Α
         ASL
               Α
         SEC
         SBC #22
                             ; ADJUST TABLE INDEX
                               ; DIVIDE INTERVAL BY 4 TO CET TABLE INDEX
         TAX
         TYA
                               ; RESTORE ACCUM
               #7
         AND
         TAY
                               ; PULL OFF 3 LO BITS OF INTERVAL
         LDA #-11
DOINTP: CLC
         ADC #11
                             ;ACCUMULATE INTERPOLATION CONSTANT
         DEY
         BPL
                             ; INTERPOLATION CONSTANT COMPUTATION DONE?
               DOINTP
ENINTP: LDY #0
         STY
             ADDCOR ; CLEAR ADDITION CORRECTION FLAG
         SEC
         SBC
               #7
                             ; ADJUST INTERPOLATION CONSTANT
               PLUS
         BPL
         DEC
               ADDCOR
       CLC
PLUS:
         ADC POKTAB, X ; ADD CONSTANT TO LO BYTE TABLE VALUE
         TAY
                               ;LO BYTE POKEY FREQ VALUE
         LDA
               ADDCOR
                POKTAB+1,X ; ADD CARRY TO HI BYTE TABLEVALUE
         ADC
; HI BYTE POKEY FREO VALUE
         RTS
        ROUTINE TO ADJUST VCOUNT VALUE
ADJUST: CMP #$7C
                          ;LAROER THAN '7C' ?
         BMI ADJ1
         SEC
                              ;YES
         SBC
               #$7C
```

```
ADJ1:
       CLC
        .IF
               PALFLG
             #$20
         ADC
         .ENDIF
         .IF PALFLG-1
         ADC
               #$7
         .ENDIF
         RTS
        INITIAL BAUD RATE MEASUREMENT -- USED TO SET THE
                 BAUD RATE AT THE START OF A RECORD.
                  IT IS ASSUMED THAT THE FIRST TWO BYTES OF EVERY
; RECORD ARE 'AA' HEX.
BEGIN: LDA BRKKEY
        BNE
               NTBRK2
               BROKE ; JUMP IF BREAK KEY PRESSED
         JMP
NTBRK2: SEI
         LDA
               TIMFLG
               OKTIM1
         BNE
                            ;BRANCH IF NOT TIMEDOUT
         BEO
               TOUT1
                         ;BRANCH IF TIME OUT
OKTIM1: LDA SKSTAT
         AND #$10
                          ; READ SERIAL PORT
         BNE
               BEGIN
                         ;START BIT?
         STA
               SAVIO
                         ; SAVE SER. DATA IN
               VCOUNT
        LDX
                            ; READ VERTICAL LINECOUNTER
               RTCLOK+2 ; READ LO BYTE OF VBLANK CLOCK
         LDY
              TIMER1
TIMER1+1 ;SAVE INITIAL TIMER VALUE
         STX
         STY
                             ;SET MODE FLAG
         LDX
               #1
         STX
               TEMP3
        LDY
                            ;SET BIT COUNTER FOR 10 BITS
               #10
COUNT: LDA BRKKEY
        BEQ BROKE
                         ;BRANCH IF BREAK KEY PRESSED
        LDA
               TIMFLG
        BNE
               OKTIMR
                            ;BRANCH IF NOT TIMED OUT
TOUT1: CLI
        JMP TOUT ;BRANCH IF TIME OUT
OKTIMR: LDA SKSTAT
         AND
               #$10
                          ; READ SERIAL PORT
                         ; DATA IN CHANGED YET?
         CMP
               SAVIO
         BEQ
               COUNT
         STA
               SAVIO
                         ;YES,SAVE SER. DATA IN
         DEY
                             ; DECR. BIT COUNTER
         BNE COUNT
                          ; DONE?
```

RTS

```
DEC
               TEMP3 ;YES,
               GOREAD
                          ; DONE WITH BOTH MODES?
         BMI
         LDA
              VCOUNT
              RTCLOK+2 ; READ TIMER LO & HI BYTES
COMPUT ; NO, COMPUTE SAUD RATE
         LDY
         JSR
         STY
               CBAUDL
              CBAUDH
                            ;SET BAUD RATE INTO RAM CELLS
         STA
         LDY
               #9
                            ;SET BIT COUNTER FOR 9 BITS
         BNE COUNT
GOREAD: LDA CBAUDL
         STA
               AUDF3
         LDA
               CBAUDH
         STA
              AUDF4
                        ;SET POKEY FREQ REGS FOR BAUD RATE
               #0
         LDA
              SKSTAT
         STA
         LDA
              SSKCTL
              SKSTAT
         STA
                           ; INIT. POKEY SERIAL PORT
         LDA
               #$55
         STA (BUFRLO), Y ; STORE '$55' AS FIRST RCV. BUFFER
         INY
              (BUFRLO),Y
         STA
         LDA
               #$AA
         STA
               CHKSUM
                       ;STORE CHECKSUM FOR 2 BYTES OF '$AA'
         CLC
        LDA BUFRLO
         ADC
               #2
         STA
              BUFRLO
         LDA
               BUFRHI
        ADC
               #0
         STA
              BUFRHI ; INCR. BUFFER POINTER BY 1
         CLI
         RTS
;
BROKE: JSR SENDDS ; BREAK KEY WAS PRESSED, SO PREPARE
        LDA #MOTRST
                         ;TO RETURN
         STA
               PACTL
                         ;TURN OFF MOTOR
         STA
               PBCTL
                         ; RAISE NOT COMMAND LINE
;
         LDA
              #BRKABT
         STA STATUS
                       ;STORE BREAK ABORT STATUS CODE
;
              STACKP
         LDX
         TXS
                              ; RESTORE STACK POINTER
         DEC
               BRKKEY
                            ;SET BREAK KEY FLAG TO NONZERO
         CLI
                              ;ALLOW IRQ'S
         JMP
               RETURN
                            ;GO RETURN
SETVBX: LDA #JTADRL
                     ;STORE TIME OUT ROUTINE ADDRESS
```

STA CDTMA1

```
LDA
               #JTADRH
         STA
                CDTMA1+1
         LDA #1
                              ;SET FOR TIMER 1
                               ;THE SETVBL ROUTINE NEEDS THIS TO CUT SHORT
         SEI
               SETVBV
                             ; ANY VBLANKS THAT OCCUR
         JSR
                              ;SET FOR TIMER 1
         LDA
                #1
               TIMFLG ;SET FLAG TO NOT TIMED OUT
         STA
         CLI
         RTS
 'VCOUNT' INTERVAL TIMER MEASUREMENT -- TO -- POKEY FREQ REG VALUE
                 CONVERSION TABLE
; THE VALUES STORED IN THE TABLE ARE 'AUDF+7'.
;
        THE FOLLOWING FORMULAS WERE USED TO DETERMINE THE TABLE VALUES:
                  F OUT F IN/(2*(AUDF+M)) , WHERE F IN=1.78979 MHZ. & M=7
         FROM THIS WAS DERIVED THE FORMULA USED TO COMPUTE THE
         TABLE VALUES BASED ON A MEASUREMENT OF THE PERIOD BY
         AN INTERVAL OF THE 'VCOUNT' TIMER.
                   AUDF+7=(11.365167)*T OUT, WHERE T OUT=# OF COUNTS
                   (127 USEC. RESOLUTION) OF 'VCOUNT' FOR 1
                   CHARACTER TIME (10 BIT TIMES).
                                 BAUD RATE VCOUNT INTERVAL
                  AUDF+7
;
        .WORD $27C
                                  ;1407
                                                               56
                                                               64
        .WORD $2D7
                                 ;1231
        .WORD $332
                                                               72
                                 ;1094
                                                                80
        .WORD $38D
                                  ;985
;
POKTAB: .WORD $3E8
                                 ;895
                                                                 88
         .WORD $443
                                  ;820
                                                                 96
                                  ;757
         .WORD $49E
                                                                 104
         .WORD $4F9
                                                                 112
                                 ;703
         .WORD $554
                                                                 120
                                  ;656
         .WORD $5AF
                                  ;615
                                                                 128
         .WORD $60A
                                 ;579
                                                                 136
         .WORD $665
                                  ;547
                                                                 144
         .WORD $6C0
                                                                 152
                                 ;518
         .WORD $71A
                                 ;492
                                                                 160
         .WORD $775
                                  ;469
                                                                 168
         .WORD $7D0
                                                                 176
                                 ;447
        .WORD $828
                                                                 184
                                 ;428
        .WORD $886
                                  ;410
                                                                 192
```

```
200
        .WORD $8E1
                                    ;394
         .WORD $93C
                                    ;379
                                                                    208
        .WORD $997
                                    ;365
                                                                    216
         .WORD $9F2
                                    ;352
                                                                    224
        .WORD $A4D
                                    ;339
                                                                    232
        .WORD $AA8
                                                                    240
                                   ;328
        .WORD $803
                                                                    248
                                   ;318
CRNTP3 =*
         *=$14
SIOSPR: .BYTE DSKORG-CRNTP3 ; GSIOL IS TOO LONG
          .TITLE 'DISK ***** DISKP.SRC ***** 3/9/79 ***** 4:00:00 P.M.'
              DVSTAT/256 (-256)*STATVH+DVSTAT ;STATUS POINTER
STATVH =
... =
STATVL =
.
        CONSTANT EQUATES
DISKID =
                $31
                               ; SERIAL BUS DISK I.D.
PUTSEC =
                $50
                               ; DISK PUT SECTOR DCB COMMAND
               $52
                               ; DISK GET SECTOR DCB COMMAND
; READ =
; WRITE =
               $57
                               ; DISK PUT SECTOR WITH READ CHECK DCB COMMAND
                               ;DISK STATUS DCB COMMAND
               $53
STATC =
                           ;DISK FORMAT DCB COMMAND !!!!! ***;
;SIO COMMAND FOR "NO DATA" OPERATION
FOMAT =
                              ;DISK FORMAT DCB COMMAND !!!!! *****
               $21
               0
NODAT =
GETDAT =
               $40
                              ;SIO COMMAND FOR "DATA FROM DEVICE"
PUTDAT =
               $80
                               ;SIO COMMAND FOR "DATA TO DEVICE"
        VECTORS
          *=$E450
                DINIT ;DISK INIT. VECTOR
DSKIF ;DISK INTERFACE ENTRY POINT
          JMP
          JMP
        CONSTANTS
          *=DSKORG
```

```
DISK INTERFACE ROUTINE STARTS HERE
       DISK INTERFACE INITIALIZATION ROUTINE
DINIT: LDA #160
         STA DSKTIM
                            ;SET INITIAL DISK TIMEOUT TO 160 SEC
         RTS
       DISK INTERFACE ENTRY POINT
DSKIF: LDA #DISKID
         STA
              DDEVIC
                            ;SET SERIAL BUS ID IN DCB
               DSKTIM
         LDA
         LDX
               DCOMND
         CPX
               #FOMAT
                            ; IS COMMAND A FORMAT COMMAND?
         BEO
               PUTDTO
               #7
                             ; NO, SET TIMEOUT TO 7 SECS.
         LDA
PUTDTO: STA DTIMLO
                         ; PUT DISK TIMEOUT IN DCB
               #GETDAT
                          ;SET "GET DATA" COMMAND FOR SIO
         LDX
         LDY
               #$80
                          ;SET BYTE COUNT TO 128
                            ; READ COMMAND IN DCB
         LDA
               DCOMND
                            ; IS COMMAND A "PUT SECTOR" COMMAND?
         CMP
               #WRITE
         BNE
               CKSTC
              #PUTDAT
                         ;YES, SET "PUT DATA" COMMAND FOR 610
         LDX
CKSTC: CMP #STATC
                         ; IS COMMAND A STATUS COMMAND?
         BNE
               PUTCNT
               #STATVL
         LDA
         STA
               DBUFLO
         LDA
               #STATVH
                            ;SET BUFFER ADDR TO GLOBAL STATUS BUFFER
         STA
               DBUFHI
         LDY
               #4
                             ;YES, SET BYTE COUNT TO 4
PUTCNT: STX DSTATS
                        ; PUT STATUS COMMAND FOR SIO IN DCB
         STY
                DBYTLO
         LDA
                #0
               DBYTHI
                            ; PUT BYTE COUNT IN DCB
         STA
               SIOV
                           ; CALL SERIAL I/O.
         JSR
               GOODST
         \mathsf{BPL}
                             ; NO ERROR
                              ;NO, GO BACK
         RTS
                        ; READ THE COMMAND
GOODST: LDA DCOMND
                            ; WAS IT A STATUS COMMAND?
         CMP
               #STATC
               PUTBC
         BNE
         JSR
               PUTADR
                         ; PUT BUFFER ADDR IN TEMP REQ.
               #2
         LDY
                (BUFADR), Y ; READ DISK TIMEOUT VALUE BYTE OF STATUS
         LDA
         STA
                            ; PUT IT IN DISK TIMEOUT REQ.
               DSKTIM
```

```
PUTBC: LDA DCOMND
        CMP
              #FOMAT
                         ; WAS COMMAND A FORMAT COMMAND?
       SR PUTADR ;YES PUT BUFFER, ADDR INTO TEMP REC
      JSR PUTADR
FMTD:
TWICE: INY
                            ; INCR BUFFER POINTER BY 2
       INY
RDBAD: LDA (BUFADR),Y ;READ LO BYTE BAD SECTOR DATA
       CMP #$FF
        BNE TWICE
                       ; IS IT "FF" ?
        INY
                            ;YES,
             (BUFADR),Y ; READ HI BYTE BAD SECTOR DATA
        LDA
        INY
        CMP
             #$FF
             RDBAD ; IS IT "FF" ?
        BNE
        DEY
                            ;YES
        DEY
                          ; PUT BAD SECTOR BYTE COUNT INTO DCB
        STY
             DBYTLO
        LDA
              #0
        STA
              DBYTHI
ENDDIF: LDY DSTATS
        RTS
      SUBROUTINES
       PUT BUFFER ADDR FROM DCB INTO TEMP REQ
PUTADR: LDA DBUFLO
        STA BUFADR
        LDA
              DBUFHI
              BUFADR+1 ; PUT BUFFER ADDR IN TEMP REQ
        STA
        RTS
SPARE BYTE OR MODULE TOO LONG FLAG
CRNTP4 =
DSKSPR: .BYTE PRNORG-CRNTP4 ; GDISKP TOO LONG
;
        .PAGE
        .TITLE 'PRINTER ***** PRINTP.SRC ***** 3/9/79 ***** 4:00:00 P
```

```
$2
                          ; IOCB OPEN FOR OUTPUT COMMAND
OPNOUT =
NBUFSZ =
               40
                          ; PRINT NORMAL BUFFER SIZE
               20
                          ; PRINT DOUBLE BUFFFER SIZE
DBUFSZ =
               29
                          ; PRINT SIDEWAYS BUFFER SIZE
SBUFSZ =
                             ; PRINTER DEVICE NUMBER
PDEVN =
               $40
                             ;DCB STATUS COMMAND CODE
; STATC =
               $53
WRITEC =
              $57
                             ;DCB WRITE COMMAND
SPACE
     =
               $20
                             ; ASCII SPACE CHAR.
Ν
               $4E
                             ;ASCII "N" CHAR.
D
               $44
                             ;ASCII "D" CHAR.
S
               $53
                             ;ASCII "S" CHAR.
       =
;
       PRINTER HANDLER ENTRY POINTS
         *=$E430
         .WORD PHOPEN-1 ; PRINTER HANDLER OPEN
         .WORD PHCLOS-1
                          ; PH CLOSE
         .WORD BADST-1 ;PH READ
         .WORD PHWRIT-1
                         ; PH WRITE
         .WORD PHSTAT-1
                          ; PH STATUS
         .WORD BADST-1 ; PH SPECIAL
         JMP PHINIT
                            ; PH INIT.
         .BYTE 0
                           ;ROM FILLER
         *=PRNORG
        PRINTER HANDLER INITIALIZATION ROUTINE
PHINIT: LDA #30
                          ;SET UP INITIAL PRINTER TIMEOUT OF 30 SEC.
         STA PTIMOT
         RTS
       PRINTER HANDLER CONSTANTS
PHSTLO: .WORD DVSTAT
                        ;STATUS BUFFER POINTER
PHCHLO: .WORD PRNBUF
                        ; CHAR. BUFFER POINTER
        ********
       PRINTER HANDLER ROUTINES
        *******
```

DEVICE NUMBER OR CODE EQUATES

```
PRINTER HANDLER STATUS ROUTINE
PHSTAT: LDA #4
                             ;SET BUFFER SIZE TO 4 BYTES
         STA
                PBUFSZ
         LDX
                PHSTLO
               PHSTLO+1 ;SET POINTER TO STATUS BUFFER
         LDY
         LDA
               #STATC
                             ;SETCOMMAND TO "STATUS"
               DCOMND
         STA
                             ; SET STATUS COMMAND
         STA
               DAUX1
         JSR SETDCB ;GO SETUP DCH

JSR SIOV ;SEND STATUS COMMAND

BMI BADST ;GO IF ERROR

JSR PHPUT ;YES, PUT STATUS INTO GLOBAL BUFFER.
BADST: RTS
       PRINTER HANDLER OPEN ROUTINE
PHOPEN: JSR PHSTAT
                         ;DO STATUS COMMAND TO SIO
         LDA #0
               PBPNT ;CLEAR PRINT BUFFER POINTER
         STA
         RTS
;
       PRINTER HANDLER WRITE ROUTINE
PHWRIT: STA PTEMP ; SAVE ACCUM
         JSR PRMODE ;GO DETERMINE PRINTMODE
         LDX
               PBPNT
         LDA
               PTEMP ;GET CHAR. SENT BY CID
         STA PRNBUF, X ; PUT CHAR. IN PRINT BUFFER
         INX
                                ; INCR. BUFFER POINTER
               PBUFSZ
                              ;BUFFER POINTERBUFFER SIZE?
         CPX
         BEQ
               BUFFUL
         STX
               PBPNT
                          ;SAVE SUFFER POINTER
         CMP
               #CR
                             ; IS CHAR. = EOL ?
               BLFILL
                             ; IF YES, GO DO BLANK FILL.
         BEQ
               #SUCCES ; PUT GOOD STATUS IN Y REQ FOR CIO.
         LDY
        RTS
BLFILL: LDA #SPACE
                         ; PUT BLANK IN ACCUM.
FILLBF: STA PRNBUF, X ; STORE IT IN PRINT BUFFER.
         INX
         CPX
               PBUFSZ
         BNE
                FILLBF ;BUFFER BLANK FILLED?
BUFFUL: LDA #0
               PBPNT
                          ;CLEAR PRINT BUFFER POINTER
         STA
         LDX
               PHCHLO
               PHCHLO+1 ;SET POINTER TO PRINT BUFFER
         LDY
              SETDCB
SIOV
         JSR
                             ;GO SETUP OCR
         JSR
                           ;SEND PRINT COMMAND
         RTS
                               ;YES.
```

```
PRINTER HANDLER CLOSE ROUTINE
PHCLOS: JSR PRMODE
                        GO DETERMINE PRINT MODE
         LDX
               PBPNT
         BNE
               BLFILL
         LDY
               #SUCCES
         RTS
;
       SUBROUTINES
       SET UP DCB TO CALL SIO
SETDCB: STX DBUFLO
         STY DBUFHI ;SET BUFFER POINTER
         LDA
               #PDEVN
         STA
                            ;SET PRINTER BUS I.D. FOR DCB
               DDEVIC
         LDA
               #1
         STA
               DUNIT
                         ;SET UNIT NUMBER TO 1
              #$80
DCOMND
                          ;DEVICE WILL EXPECT DATA
         LDA
         LDX
         CPX
               #STATC
                            ;STATUS COMMAND?
         BNE
               PSIOC
               #$40
                          ; EXPECT DATA FROM DEVICE
         LDA
                        ;SET SIO MODE COMMAND
PSIOC: STA DSTATS
         LDA
               PBUFSZ
         STA
                           ;SET LO BYTE COUNT
               DBYTLO
         LDA
               #0
         STA
              DBYTHI
                            ;SET HI BYTE COUNT
         LDA
               PTIMOT
                            ;SET DEVICE TIMEOUT COUNT
         STA
              DTIMLO
         RTS
; GET DEVICE TIMEOUT FROM STATUS & SAVE IT
PHPUT: LDA DVSTAT+2
         STA PTIMOT
                           ;SAVE DEVICE TIMEOUT
        RTS
;
; DETERMINE PRINT MODE & SETUP PRINT BUFFER SIZE, DCB PRINT
```

```
PRMODE: LDY #WRITEC ; PUT WRITE COMMAND IN Y REG
        LDA ICAX2Z ; READ PRINT MODE
CMODE: CMP #N
        BNE CDUBL ; PRINT NORMAL ?
         LDX
                #NBUFSZ
                            ;YES, SET NORMAL CHAR. BUFFER SIZE
                SETBSZ
         BNE
CDUBL: CMP #D
         BNE CSIDE ;PRINT DOUBLE?
LDX #DBUFSZ ;YES, SET DOU
                           ;YES, SET DOUBLE CHAR. BUFFER SIZE
         BNE SETBSZ
CSIDE: CMP #S
                            ; PRINT SIDEWAYS ?
         BNE GOERR ; IF NOT, GO TO ERROR ROUTINE
LDX #SBUFSZ ; YES, SET SIDEWAYS BUFFER SIZE
SETBSZ: STX PBUFSZ ;STORE PRINT BUFFER SIZE
STY DCOMND ;STORE DCB COMMAND
STA DAUX1 ;STORE DCB AUX1 PRINT MODE
         RTS
                            ;SET DEFAULT PRINT MODE TO NORMAL
GOERR: LDA #N
        BNE CMODE
SPARE BYTE OR MODULE TOO LONG FLAG
CRNTP5 = *
PRNSPR: .BYTE CASORG-CRNTP5 ; GPRINTP TOO LONG
         .PAGE
         .TITLE 'CASSET HANDLER 3/12 (DK1:CASCV)'
CBUFH = CASBUF/256
CBUFL = (-256)*CBUF
                (-256)*CBUFH+CASBUF
SRSTA =
               $40
                              ;SIO READ STATUS
              $80
$34
$3C
SWSTA =
                               ;SIO WRITE STATUS
;MOTRGO =
;MOTRST =
                      ;DATA RECORD TYPE BYTE ;LAST DATA RECORD
DTA = $FC
DT1 =
           $FA
                         ;END OF TAPE
;HEADER
EOT = $FE
HDR = $FB
TONE1 = 2
TONE2 = 1
                           ; CHANGE TO RECORD MODE TONE ; PRESS PLAY TONE
          *=CASETV
          .WORD OPENC-1, CLOSEC-1, GBYTE-1, PBYTE-1, STATU-1, SPECIAL-1
          JMP INIT
                            ;ROM FILLER BYTE
          .BYTE 0
```

; COMMAND, &. DCB AUX1 FOR PRINT MODE

```
; USED IN MONITP FOR CASSETTE BOOT
         *=RBLOKV
         JMP RBLOK
         *=CSOPIV
         JMP OPINP
         *=CASORG
; INIT ROUTINE
INIT: LDA #$CC
         STA CBAUDL
               #$05
         LDA
               CBAUDH
         STA
                            ;SET CASSET BAUD RATE TO 600
                              ;THATS ALL FOLKS
SPECIAL:
        RTS
         .PAGE
; OPEN FUNCTION - WITH NO TIMING ADJUST
               AX2Z ;GET AX2
FTYPE ;SAVE IT FOR FUTURE REFERENCE
OPENC: LDA ICAX2Z
        STA
               ICAX1Z
         LDA
         AND
               #$0C
                          ; IN AND OUT BITS
         CMP
               #$04
         BEO
               OPINP
         CMP
               #$08
                          ;SEE IF OPEN FOR OUTPUT
        BEO
               OPOUT
                               ; IF ALREADY OPEN, RETURN LEAVING STATUS=$84
        RTS
OPINP: LDA #0
              WMODE
        STA
                         ;SET READ MODE
        STA
               FEOF
                          ;NO EOF YET
               #TONE 2
SFH:
      LDA
                            ;TONE FOR PRESS PLAY
              BEEP
OPNRTN
                          ;GO BEEP
         JSR
         BMI
                            ; IF ERROR DURING BEEP
         LDA
               #MOTRGO
               PACTL
                         ;TURN MOTOR ON
         STA
         .IF
               PALFLG
               #$E0
         LDY
         LDX
               #1
         .ENDIF
              PALFLG-1
         .IF
                       ;5-31-79 9 SEC READ LEADER
         LDY
               #$40
         LDX
               #2
         .ENDIF
         LDA
               #3
         STA
               CDTMF3
                            ;SET UP YBLANK TIMER
         JSR
                SETVBV
WAITTM: LDA CDTMF3
                            ; WAIT FOR MOTOR TO COME UP TO SPEED
         BNE
               MTTIAW
         LDA
               #$80
                           ; NEXT BYTE=NO BYTES IN BUFFER
         STA
               BPTR
         STA
               BLIM
         JMP
               OPOK
                          ;OPEN OK
```

```
; OPEN FOR OUTPUT
     LDY #BRKABT ;BREAK KEY ABORT STATUS
DEC BRKKEY ;RESET BREAK KEY
PBRK:
OPNRTN: LDA #0
                          ;CLEAR WRITE MODE FLAG
         STA WMODE
         RTS
                               ; AND EXIT.
OPOUT: LDA #$80
         STA WMODE ;SET WRITE MODE
         LDA
               #TONE1
                            ;TELL USER TO TURN ON RECORD MODE
         JSR
               BEEP
             OPNRTN
#$CC
AUDF3
#$05
                            ; IF ERROR DURING BEEP
         BMI
                         ;SET BAUD RATE
         LDA
                         ; WHICH SEEMS TO BE NESSECARY
         STA
                          ; FOR SOME OBSCURE REASON
         LDA
         STA
               AUDF4
         LDA
               #$60
              #$60
DDEVIC
SENDEV ;TELL POKEY TO WRITE N
#MOTRGO ;WRITE 5 SEC BLANK TAPE
PACTL
         STA
                             ;TELL POKEY TO WRITE MARKS
         JSR
         LDA
         STA
         LDA
               #3
               PALFLG
         .IF
         LDX
               #$3
         LDY #$C0
         .ENDIF
         .IF PALFLG-1
                             ;5/30/79 20 SEC LEADER
         LDX
               #4
         LDY
               #$80
         .ENDIF
         JSR
              SETVBV
         LDA
               #$FF
         STA
               CDTMF3
       LDA BRKKEY
WDLR:
                          ; IF BREAK DURING WRITE LEADER
         BEQ PBRK
         LDA
               CDTMF3
         BNE
               WDLR
         LDA
               #0
                             ; INIT BUFFER POINTER
        STA
               BPTR
OPOK:
     LDY #SUCCES
        RTS
         .PAGE
; GET BYTE
GBYTE: LDA FEOF ; IF AT EOF ALREADY
        BMI ISEOF
                        ; RETURN EOF STATUS
         LDX
               BPTR
                           ;BUFFER POINTER
               BLIM
         CPX
                          ; IF END OF BUFFER
               RBLOK
         BEO
                          ; READ ANOTHER BLOCK
               CASBUF+3,X ;GET NEXT BYTE
         LDA
         INC
               BPTR
                          ;DUMP POINTER
               #SUCCES ;OK STATUS
        LDY
GBX:
       RTS
                        ; READ OPCODE
RBLOK: LDA #'R
         JSR SIOSB ;SIO ON SYS BUF
         TYA
```

```
GBX
         BMI
                              ; IF SIO ERRORS, RETURN
         LDA
                #0
         STA BPTR
LDX #$80
LDA CASBUF+2
CMP #BOT
                            ; RESET POINTER
                          ;DEFAULT # BYTES
         CMP
                #EOT
         BEQ ATEOF
CMP #DT1
BNE NLR
                          ; IF HEADER, GO READ AGAIN
                           ; IF LAST DATA REC
               CASBUF+130 ;LAST DATA RECORD, GET # BYTES
         LDX
        STX BLIM

JMP GBYTE ;GET NEXT BYTE
       STX
NLR:
ATEOF: DEC FEOF ;SET FEOF ISEOF: LDY #EOFERR ;ENDFILE STATUS
        RTS
         .PAGE
; PUT BYTE TO BUFFER
PBYTE: LDX BPTR ;BUFFER POINTER
         STA CASBUF+3,X ;STORE CHAR AWAY
                BPTR ;BUMP POINTER

#SUCCES ;OK STATUS

#127 ;IF BUFFER FULL
         INC
         LDY
         CPX
         BEO
                 *+3
         RTS
; WRITE OUT THE BUFFER
         LDA #DTA ;RECORD TYPE = DATA
         JSR
                WSIOSB
                            ;DO WRITE ON SYSTEM BUFFER
         LDA
                #0
         STA BPTR ;RESET BUFFER POINTER
         RTS
                                ;EXIT.
         .PAGE
; STATUS - RETURN STATUS INFO THRU DVSTAT
STATU: LDY #SUCCES
        RTS
         .PAGE
; CLOSE
CLOSEC: LDA WMODE ;SEE IF WRITING
        BMI CLWRT ;300 CLOSE FOR WRITE
; CLOSE FOR READ - FLAG CLOSED
        LDY #SUCCES ;SUCCESSFULL
       LDA #MOTRST ;STOP THE MOTOR IN CASE WAS SHORT IRQ MODE
FCAX:
         STA PACTL
         RTS
CLWRT: LDX BPTR ;BUFFER POINTER BEQ WTLR ;IF NO DATA
                        ; IF NO DATA BYTES IN BUFFER, NO DT1 REC
         STX
                CASBUF+130 ; WRITE TO LAST RECORD
         LDA
                #DT1 ;REC TYPE
         JSR WSIOSB ;WRITE OUT USER BUFFER
BMI FCAX ;GO IF ERROR
       LDX #127 ;ZERO BUFFER
WTLR:
        LDA #0
ZTBUF: STA CASBUF+3,X
         DEX
```

```
ZTBUF
         BPL
         LDA
               #EOT
                          ;WRITE EOT RECORD
         JSR
               WSIOSB
             FCAX ;FLAG CLOSED AND EXIT
         JMP
         .PAGE
; SUBROUTINES
; BEEP - GENERATE TONE ON KEYBOARD SPEAKER
; ON ENTRY A= FREQ
BEEP:
      STA FREQ
BEEP1: LDA RTCLOK+2 ; CURRENT CLOCK
        CLC
        .IF
               PALFLG
         ADC
               #25
        .ENDIF
         .IF
              PALFLG-1
             #30
                            ; 1 SEC TONE
        ADC
        .ENDIF
        TAX
              #$FF
CONSOL
WFL:
       LDA
        STA
                         ;TURN ONSPEAKER
         LDA
               #0
         LDY
               #$F0
         DEY
              *-1
CONSOL ;TURN OFF SPEAKER
         BNE
         STA
         LDY
               #$F0
         DEY
              *-1
RTCLOK+2 ;SEE IF 1 SEC IS UP YET
         BNE
         CPX
         BNE
               WFL
              FREQ
WFAK
                         COUNT BEEPS; IF ALL DONE GO WAIT FOR KEY
        DEC
         BEO
         TXA
         CLC
         .IF
               PALFLG
         ADC
               #8
         .ENDIF
            PALFLG-1
         .IF
         ADC
               #10
         .ENDIF
         TAX
        CPX
              RTCLOK+2
        BNE
               *-2
               BEEP1
        BEQ
                         ;UNCOND DO BEEP AGIN
      JSR WFAK1 ; USE SIMULATED "JMP (KGETCH)"
WEAK:
        TYA
         RTS
WFAK1: LDA KEYBDV+5
         PHA
               KEYBDV+4 ;SIMULATE "JMP (KGETCH)"
         LDA
         PHA
         RTS
; SIOBS - CALL SIO ON SYSTEM BUFFER
SIOSB: STA DCOMND ; SAVE COMMAND
```

```
LDA
                #0
         STA
                DBYTHI
                              ;SET BUFFER LENGTH
         LDA
                #131
         STA DBYTLO
LDA #CBUFH
STA DBUFHI
                              ;SET BUFFER ADDRESS
         LDA
                #CBUFL
         STA DBUFLO
      LDA #$60 ;CASSET PSEUDO DEVICE
CSIO:
         STA DDEVIC
         LDA
                #0
         STA
                DUNIT
         LDA #35
STA DTIMLO
LDA DCOMND
LDY #SRSTA
CMP #'R
                              ;DEVICE TIMEOUT (5/30/79)
                             ;GET COMMAND SACK
                              ;SIO READ STATUS COMMAND
         BEO
                *+4
               #SWSTA
DSTATS
FTYPE
DAUX2
SIOV
         LDY
                              ;SIC WRITE STATUS COMMAND
         STY
                              ;SET STATUS FOR SIO
         LDA
                          ; INDICATE IF SHORT IRQ MODE
         STA
         JSR
                            ;GO CALL SIO
         RTS
; WSIOSB - WRITE SIC SYSTEM SUFFER
WSIOSB: STA CASBUF+2 ;STORE TYPE BYTE
         LDA #$55
         STA
                CASBUF+0
         STA
               CASBUF+1
         LDA
                #'W
                              ;WRITE
         JSR SIOSB ; CALL SIO ON SYSTEM BUFFER
         RTS
                               ; RETURN
                AND
CRNTP6 =*
         *=$14
CASSPR: .BYTE MONORG-CRNTP6 ; GCASCV IS TOO LONG
         .TITLE 'MONITOR **** MONITP.SRC **** 3/9/79 **** 4:00:00 P
       CONSTANT EQUATES
                $9
                              ; "PUT TEXT RECORD" CIO COMMANDCODE
PUTTXT =
GETCAR =
                $7
                              ; "GET CHARACTER" CIO COMMAND CODE
PUTCAR =
                              ; "PUT CHARACTER" CIO COMMAND CODE
                $В
INIMLL =
               $00
                                ; INITIAL HEM LO LOW BYTE
INIMLH =
               $07
                                ; INITIAL HEM LO HIGH BYTE
; GOOD =
                $1
                              GOOD STATUS CODE
; WRITE =
                                ;WRITE COMMAND
               $57
; READ -
; STATC =
; $0
$7D
               $52
                                ; READ COMMAND
           $53
$0
$7D
                                 ;STATUS COMMAND
                         ;SCREEN EDITOR 10CR INDEX
                           CLEAR SCREEN CODE; KEYBOARD CODE FOR 'COI; CASSETTE END OF FILE CODE
CTRLC = $88
                $92
                                 ;KEYBOARD CODE FOR 'CONTROL C'
            $88
LIRO
      =
            $0
                              ;LONG IRQ TYPE CODE
```

```
BUFFL =
              (-256)*BUFFH+CASBUF+3;BUFFER POINTER
; THE FOLLOWING EQUATES ARE IN THE CARTRIDGE ADDRESS SPACE.
; "B" CARTRIDGE ADDR'S ARE 8000-9FFF (36K CONFIG. ONLY)
; "A" CART. ADDR'S ARE A000-BFFF (36K CONFIG. ONLY)
; "A" CART. ADDR'S ARE B000-BFFF (48K CONFIG. ONLY)
        *=$BFFA
CARTCS: .RES 2
                             ; CARTRIDGE COLD START ADDRESS.
CART: .RES
              1
                             ; CARTRIDGE AVAILABLE FLAG BYTE.
              1
                             ; CARTRIDGE FLAG BYTE. BIT 0=FLAG1,
CARTFG: .RES
CARTAD: .RES
              2
                             ;2-BYTE CARTRIDGE START VECTOR
       CARTRIDGE FLAG ACTION DEFINITIONS
                         ACTION IF SET
       BIT
;
       7
                        SPECIAL -- DON'T POWER-UP, JUST RUN CARTRIDGE
       6-3
                         NONE
       2
                         RUN CARTRIDGE
       1
                        NONE
       0
                         BOOT DOS
        ****
       NOTE
        ****
;
       1. IF BIT2 IS 0, GOTO BLACKBOARD MODE.
;
       2.IF BITO SET THE DISK WILL BE BOOTED BEFORE ANY
          OTHER ACTION.
       POWER-UP VECTOR
*=$FFFC
; PVECT .WORD PWRUP
                               POWER-UP VECTOR
```

BUFFH =

(CASBUF+3)/256

```
ENTRY POINT VECTOR
         *=BLKBDV
;
         JMP SIGNON ;BLACK BOARD VECTOR
         *=WARMSV
         JMP RESET ; WARM START VECTOR
         *=COLDSV
         JMP PWRUP ; COLD START VECTOR (9000 FOR RAM VECTOR WRIT
         *=$9000
         JSR
              $900C
                          ;(TO HANDLE RAM VECTOR WRITING)
         JMP
               PWRUP
         JSR $900C
JMP RESET
         *=MONORG
       HANDLER TABLE ENTRIES
TBLENT: .BYTE 'P'
         .WORD PRINTV
         .BYTE 'C'
         .WORD CASETV
         .BYTE 'E'
         .WORD EDITRV
         .BYTE 'S'
         .WORD SCRENV
         .BYTE 'K'
         .WORD KEYBDV
;TBLLEN = IDENT-TBLENT-1 HANDLER TABLE LENGTH. "MOVED TO LINE 8
       ***** PRINT MESSAGES *****
IDENT: .BYTE CLS, 'ATARI COMPUTER - MEMO PAD', CR
IDENTH = IDENT/256
IDENTL =
              (-256)*IDENTH+IDENT ;SYSTEM I.D. MSG POINTER
TBLLEN =
              IDENT-TBLENT-1 ; HANDLER TABLE LENGTH
```

```
DERR5: .BYTE 'BOOT ERROR', CR
```

```
DERRH = DERR5/256
DERRL =
            (-256)*DERRH+DERR5; DISK ERROR MSG POINTER
      DEVICE/FILENAME SPECIFICATIONS
OPNEDT: .BYTE 'E:',CR ;"OPEN SCREEN EDITOR" DEVICE SPEC.
            OPNEDT/256
OPNH
      =
            (-256)*OPNH+OPNEDT ;SCREEN EDITOR OPEN POINTER
OPNL
RESET BUTTON ROUTINE STARTS HERE
RESET: SEI
                       ; DISABLE IRQ INTERRUPTS
       LDA COLDST
BNE PWRUP
                         ; WERE WE IN MIDDLE OF COLDSTART?
                      ;YES, GO TRY IT AGAIN
       LDA
             #$FF
                        ;SET WARM START FLAG
       BNE
             PWRUP1
POWER UP ROUTINES START HERE
PWRUP: SEI
                       ; DISABLE IRQ INTERRUPTS
       LDA #0
                         ;CLEAR WARMSTART FLAG
PWRUP1: STA WARMST
       CLD
                          ;CLEAR DECIMAL FLAG.
       LDX
             #$FF
       TXS
                          ;SET STACK POINTER
                     ;CARTRIDGE SPECIAL CASE?
       JSR SPECL
JSR HARDI
                      ; DO HARDWARE INITIALIZATION
       LDA
             WARMST
                        ; IS IT WARMSTART?
                         ;YES, ONLY ZERO OS RAM
       BNE
             ZOSRAM
ZERORM: LDA #0
       LDY
             #WARMST
       STA
             RAMLO
       STA
             RAMLO+1
                      ; INITIALIZE RAM POINTER
CLRRAM: STA (RAMLO), Y ; CLEAR MEMORY LOC.
       INY
            #0
CLRRAM
                         ;AT END OF PAGE?
       CPY
       BNE
                      ; YES. INCR PAGE POINTER
       INC
             RAMLO+1
       LDX
             RAMLO+1
       CPX
             TRAMSZ
                        ;AT END OF MEM?
```

```
BNE CLRRAM
                            ;NO.
; INITIALIZE DOSVEC TO POINT TO SIGNON (BLACKBOARD)
         LDA BLKBDV+1
                         ;USE BLACKBOARD VECTOR
               DOSVEC ; USE BLAC
BLKBDV+2 ; FOR DOSVEC
         STA
         LDA
               DOSVEC+1
         STA
         LDA
               #$FF
         STA COLDST ;SET TO SHOW IN ME
BNE ESTSCM ;GO AROUND ZOSRAM
                             ;SET TO SHOW IN MIDDLE OF COLDSTART
; CLEAR OS RAM (FOR WARMSTART)
ZOSRAM: LDX #0
        TXA
ZOSRM2: STA $200,X ;CLEAR PAGES 2 AND 3
         STA $300,X
         DEX
         BNE
               ZOSRM2
         LDX #INTZBS
ZOSRM3: STA 0,X ;CLEAR ZERO PAGE LOCATIONS INTZBS-7F
         INX
         BPL ZOSRM3
; ESTABLISH SCREEN MARGINS
ESTSCM: LDA #LEDGE
        STA LMARGN
         LDA #REDGE
STA RMARGN
        LDA
; MOVE VECTOR TABLE FROM ROM TO RAM
OPSYS: LDX #$25
MOVVEC: LDA VCTABL, X ; ROM TABLE
         STA INTABS, X ; TO RAM
        BPL MOVVEC
JSR OSP*
               OSRAM
                         ;DO O.S. RAM SETUP
         CLI
                               ; ENABLE IRQ INTERRUPTS
       LINK HANDLERS
        LDX
               #TBLLEN
NXTENT: LDA TBLENT, X ; READ HANDLER TABLE ENTRY
         STA HATABS,X ; PUT IN TABLE
         DEX
         BPL NXTENT
                            ; DONE WITH ALL ENTRIES?
; INTERROGATE CARTRIDGE ADDR. SPACE TO SEE WHICH CARTRIDGES THERE ARE
         LDX
               #0
                            ;CLEAR "B" CART. FLAG
               TSTDAT
         STX
         STX
               TRAMSZ
                            ;CLEAR "A" CART. FLAG
         LDX
               RAMSIZ
         CPX
               #$90
                          ; RAM IN "B" CART. SLOT?
```

```
BCS
               ENDBCK
         LDA
               CART-$2000 ;NO.
         BNE
               ENDBCK ; CART. PLUGGED INTO "B" SLOT'?
         INC TSTDAT ;YES, SET "B" CART, FLAG
JSR CBINI ;INITIALIZE CARTRIDGE "B"
ENDBCK: LDX RAMSIZ
                          ; RAM IN "A" CART. SLOT?
         CPX #$B0
              ENDACK
CART
ENDACK
TRAMSZ
         BCS
         LDX
                           ;NO,
                           ; CART. PLUGGED INTO "A" SLOT?
         BNE
         INC
                             ;YES, SET "A" CART. FLAG
               CAINI ; INITIALIZE CARTRIDGE "A"
         JSR
; OPEN SCREEN EDITOR
ENDACK: LDA #3
         LDX
               #SEX
               ICCOM, X ; OPEN I/O COMMAND
         STA
         LDA
               #OPNL
         STA ICBAL,X
LDA #OPNH
               ICBAH,X
#$C
                           ;SET BUFFER POINTER TO OPEN SCREEN EDITOR
         STA
         LDA
         STA ICAX1,X ;SET UP OPEN FOR INPUT/OUTPUT JSR CIOV ;GO TO CIO
;
         BPL SCRNOK
                             ;BR IF NO ERROR
               PWRUP
                          ; RETRY PWRUP IF ERROR (SHOULD NEVER HAPPEN!)
         JMP
        .NX
BNE SCRNOK
                           ;SCREEN OK, SO WAIT FOR YBLANK TO
SCRNOK: INX
                             BRING UP THE DISPLAY
         INY
         BPL SCRNOK
;
; DO CASSETTE BOOT
         JSR CSBOOT ; CHECK, BOOT, AND INIT
; CHECK TO SEE IF EITHER CARTRIDGE WANTS DISK BOOT
         LDA TRAMSZ ; CHECK BOTH CARTRIDGES
               TSTDAT
NOCART
         ORA
                          ; NEITHER C:
; "A" CART?
; NO
                             ; NEITHER CARTRIDGE LIVES
         BEQ
               TRAMSZ
         LDA
         BEQ
              NOA1
CARTFG
        LDA
                            GET CARTRIDGE MODE FLAG
                         ;"B" CART?
NOA1: LDX TSTDAT
        BEQ NOB1
                           ; NO
        ORA CARTFG-$2000 ; ADD OTHER FLAG
                          ; DOES EITHER CART WANT BOOT?
NOB1: AND #1
        BEQ NOBOOT
                             ; NO
; DO DISK BOOT
NOCART: JSR BOOT ; CHECK. BOOT. AND INIT
; GO TO ONE OF THE CARTRIDGES IF THEY SO DESIRE
NOBOOT: LDA #0
                             ; RESET TO SHOW DONE WITH COLDSTART
               COLDST
         STA
         LDA TRAMSZ
                             ; "A" CART?
```

```
NOA2 ;NO
CARTFG ;GET CARTRIDGE MODE FLA
#4 ;DOES IT WANT TO RUN?
         BEO
                          GET CARTRIDGE MODE FLAG
         LDA
         AND
               NOA2
                          ;NO
         BEQ
            (CARTCS) ;RUN "A" CARTRIDGE
         JMP
NOA2:
     LDA TSTDAT ; "B" CART?
              NOCAR2
         BEQ
                            ; NO
              CARTFG-$2000 ;GET "B" MODE FLAG
         LDA
         AND
               #4
                             ; DOES IT WANT TO RUN?
         BEQ
               NOCART
                            ; NO
               (CARTCS-$2000) ; RUN "B" CARTRIDGE
         JMP
; NO CARTRIDGES, OR NEITHER WANTS TO RUNS
; SO GO TO DOSVEC (DOS, CASSETTE, OR BLACKBOARD)
NOCAR2: JMP (DOSVEC)
; PRINT SIGN-ON MESSAGE
SIGNON: LDX #IDENTL
         LDY #IDENTH
               PUTLIN ;GO PUT SIGN-ON MSG ON SCREEN
         JSR
;
       BLACKBOARD ROUTINE
BLACKB: JSR BLKB2 ; "JSR EGETCH"
       JMP BLACKB ; FOREVER
BLKB2: LDA EDITRV+5 ; HIGH BYTE
         PHA
         LDA
               EDITRV+4 ;LOW BYTE
         PHA
         RTS
                              ;SIMULATES "JMP (EDITRV)"
; CARTRIDGE INITIALIZATION INDIRECT JUMPS
CAINI: JMP (CARTAD)
CBINI: JMP (CARTAD-$2000)
        .PAGE
                  SUBROUTINES
```

```
; CHECK FOR HOW MUCH RAM & SPECIAL CARTRIDGE CASE.
; IF SPECIAL CARTRIDGE CASE, DON'T GO BACK -- GO TO CART.
                       ; CHECK FOR RAM OR CART
SPECL: LDA CART
         BNE ENSPE2
                         GO IF NOTHING OR MAYBE RAM
              CART
CART
ENSPEC
         INC
                          ; NOW DO RAM CHECK
                          ; IS IT ROM?
         LDA
                            ;NO
         BNE
         LDA
               CARTFG
                            ;YES,
               ENSPEC
                            ;BIT SET?
         \mathtt{BPL}
         JMP (CARTAD) ; YES, GO RUN CARTRIDGE
; CHECK FOR AMOUNT OF RAM
ENSPEC: DEC CART
                  ; RESTORE RAM IF NEEDED
ENSPE2: LDY #0
         STY RAMLO+1
         LDA
                #$10
         STA
               TRAMSZ
                            ;SET RAM POINTER TO 4K.
HOWMCH: LDA (RAMLO+1), Y ; READ RAM LOCATION
         EOR
              #$FF
                        ; INVERT IT.
         STA
               (RAMLO+1),Y; WRITE INVERTED DATA.
         CMP
               (RAMLO+1),Y ;READ RAM AGAIN
              ENDRAM
         BNE
         EOR
               #$FF
                          ; CONVERT IT BACK
                (RAMLO+1), Y ; RESTORE ORIGINAL RAMDATA
         STA
         LDA
               TRAMSZ
         CLC
         ADC #$10
         STA
               TRAMSZ
                           ; INCR. RAM POINTER BY 4K.
               HOWMCH
                            ;GO FIND HOW MUCH RAM.
         BNE
ENDRAM: RTS
       HARDWARE INITIALIZATION
HARDI: LDA #0
        TAX
CLRCHP: STA $D000,X
         STA $D400,X
         STA
               $D200,X
         STA
               $D300,X
         INX
         BNE CLRCHP
         RTS
       O.S. RAM SETUP
                    ;TURN OFF BREAK KEY FLAG
OSRAM: DEC BRKKEY
         LDA #.LOW.BRKKY2
         STA
               BRKKY
```

```
STA
                BRKKY+1
               TRAMSZ
                            ; READ RAM SIZE IN TEMP. REG.
         LDA
               RAMSIZ
         STA
                            ; SAVE IT IN RAM SIZE.
              MEMTOP+1 ;INIT. MEMTOP ADDR HI BYTE
         STA
         LDA
               #0
         STA
               MEMTOP
                            ; INIT. MEMTOP ADDR LO BYTE
         LDA
               #INIMLL
         STA
               MEMLO
              MEMLO
#INIMLH
MEMLO+1 ;INITIALIZE MEMLO ADDR VECTOR
         LDA
         STA
         JSR
               EDITRV+$C ; EDITOR INIT.
               SCRENV+$C ; SCREEN INIT.
         JSR
               KEYBDV+$C ; KEYBOARD INIT.
         JSR
               PRINTV+SC ; PRINTER HANDLER INIT
         JSR
              CASETV+$C ;CASSETTE HANDLER INIT
CIOINV ;CIO INIT.
         JSR
         JSR
         JSR
               SIOINV
                            ;SIO INIT.
              INTINV
CONSOL
         JSR
                            ; INTERRUPT HANDLER INIT.
         LDA
         AND
               #$1
              NOKEY
                         ;GAME START KEY DEPRESSED?
         BNE
         INC
                          ; YES. SET KEY FLAG.
NOKEY: RTS
; DO BOOT OF DISK
;
BOOT: LDA WARMST
                        ;W.
        BEQ NOWARM
                            ; WARMSTART?
         LDA
               BOOT?
         AND
               #1
              NOINIT
DINI
                           ; VALID BOOT?
                         ; VALID BOOL.; YES, RE-INIT. DOS SOFTWARE
         BEQ
         JSR
NOINIT: RTS
NOWARM: LDA #1
              DUNIT ; ASSIGN DISK DRIVE NO.
         STA
               #STATC
         LDA
               DCOMND
         STA
                            ;SET UPSTATUS COMMAND
         JSR
               DSKINV
                            GO DO DISK STATUS
                            ; IS STATUS FROM 510 GOOD?
         BPL
               DOBOOT
         RTS
                              ; NO, GO BACK WITH BAD BOOT STATUS
DOBOOT: LDA #0
         STA DAUX2
         LDA
               #1
         STA
               DAUX1
                         ;SET SECTOR # TO 1.
         LDA
               #BUFFL
         STA
               DBUFLO
         LDA
               #BUFFH
               DBUFHI
         STA
                            ;SET UP BUFFER ADDR
SECT1: JSR GETSEC
                        GET SECTOR
         BPL ALLSEC
                          ;STATUS O.K.?
BADDSK: JSR DSKRDE
                        ; NO, GO PRINT DISK READ ERROR
         LDA
              CASSBT
               DOBOOT
         BEQ
                             ; CASSETTE BOOT?
         RTS
                              ;YES, QUIT
ALLSEC: LDX #3
RDBYTE: LDA CASBUF+3,X ; READ A BUFFER BYTE
```

LDA

#.HIGH.BRKKY2

```
DEX
                              ; DONE WITH 4 BYTE TRANSFER
         \mathtt{BPL}
                RDBYTE
               BOOTAD
RAMLO
BOOTAD+1
         LDA
                              ;YES.
         STA
         LDA
         STA
                RAMLO+1
                            ; PUT BOOT ADDR INTO Z. PAGE RAM
                CASBUF+7
         LDA
         STA
                DOSINI
                              ; ESTABLISH DOS INIT ADDRESS
         LDA
                CASBUF+8
         STA DOSINI+1
MVBUFF: LDY #$7F ;YES, SET BYTE COUNT
MVNXB: LDA CASBUF+3,Y
         STA (RAMLO), Y ; MOVE A BYTE FROM SECTOR BUFFER TO BOOT ADDR
         DEY
         BPL MVNXB ; DONE ?
         CLC
                                ;YES,
         LDA
               RAMLO
         ADC
                #$80
         STA
                RAMLO
               RAMLO+1
         LDA
               #0
RAMLO+1 ;INCR BOOT LOADER BUFFER POINTER
         ADC
         STA
                            DECR # OF SECTORS.
                DBSECT
         DEC
         BEQ
                ENBOOT
                              ; MORE SECTORS ?
INC DAUX1 ;YES INCR SECTOR.

SECTX: JSR GETSEC ;GO GET SECTOR.

BPL MVBUFF ;STATUS O.K
                          ; MORE SECTORS ?
; YES INCR SECTOR #
                               ;STATUS O.K. ?
         JSR
                DSKRDE
                               ;NO, GO PRINT DISK READ ERROR
         LDA
                CASSBT
         BNE BADDSK ; IF CASSETTE, QUIT.
BEQ SECTX ; IF DISK, TRY SECTOR AGAIN.
ENBOOT: LDA CASSBT
         BEQ XBOOT ;A CASSETTE BOOT ?

JSR GETSEC ;YES, GET EOF RI
                            ; YES, GET EOF RECORD, BUT DON'T USE IT.
XBOOT: JSR BLOAD ;GO EXECUVE BOOT LOADER
         BCS BADDSK ; IF BAD BOOT, DO IT OVER AGAIN
                DINI
         JSR
                            ;GO INIT. SOFTWARE
         INC BOOT? ; SHOW BOOT SUCCESS
         RTS
BLOAD: CLC
               BOOTAD
         LDA
         ADC
                #6
         STA RAMLO
LDA BOOTAD+1
         ADC
                #0
         STA
                RAMLO+1 ; PUT START ADDR OF BOOTLOADER INTO RAM
         JMP (RAMLO)
DINI: JMP (DOSINI)
; DISPLAY DISK READ ERROR MSG
DSKRDE: LDX #DERRL
         LDY #DERRH
;
```

STA

DFLAGS,X ;STORE IT

```
; PUT LINE ON SCREEN AT PRESENT CURSOR POSITION
   X-REG -- LO BYTE, BEGIN ADDR OF LINE
   Y-REG -- HI BYTE, BEGIN ADDR OF LINE
;
PUTLIN: TXA
         LDX
                #SEX
         STA
                ICBAL,X
         TYA
         STA ICBAH,X ;SET UP ADDR OF BEGIN OF LINE
         LDA
                #PUTTXT
         STA
               ICCOM,X
#$FF
                            ; "PUT TEXT RECORD" COMMAND
         LDA
         STA ICBLL,X ;SET BUFFER LENGTH
JSR CIOV ;PUT LINE ON SCREEN
         RTS
; GET SECTOR FROM DISK 0
GETSEC: LDA CASSBT
         BEQ DISKM ; CASSETTE BOOT?
         JMP RBLOKV
                           ; YES, GO TO READ BLOCK ROUTINE
DISKM: LDA #READ
         STA DCOMND ;SET READ SECTOR COMMAND
         LDA
                #1
         STA DUNIT
JSR DSKINV
                          ;SET DRIVE NO. TO DRIVE 0
                             GET SECTOR
         RTS
; DO CHECK FOR CASSETTE BOOT & IF SO DO BOOT
CSBOOT: LDA WARMST ;WARMSTART?
BEQ CSBOT2 ;NO
LDA BOOT? ;GET BOOT FLAG
         AND
                #2
                              ; WAS CASSETTE BOOT SUCCESFULL?
         BEQ NOCSB2
JSR CINI
                              ;NO
                          ;YES, INIT CASSETTE SOFTWARE
NOCSB2: RTS
;
CSBOT2: LDA CKEY
         BEQ NOCSBT
                              ; "C" KEY FLAG SET ?
         LDA
                #$80
                            ;YES,
               FTYPE
CASSBT
CSOPIV
         STA
                           ;SET LONG IRO TYPE
         INC
                             ;SET CASSETTE BOOT FLAG
         JSR
                              ;OPEN CASSETTE FOR INPUT
         JSR
                SECT1
                           ;DO BOOT & INIT.
         LDA
                #0
               CASSBT
CKEY
         STA
                              ; RESET CASSETTE BOOT FLAG
                           CLEAR KEY FLAG
         STA
               BOOT? ;SHIFT BOOT FLAG (NOW=2 IF SUCCESS)
         ASL
         LDA
               DOSINI
         STA
                CASINI
                              ; MOVE INIT ADDRESS FOR CASSETTE
         LDA
                DOSINI+1
```

```
STA CASINI+1
NOCSBT: RTS
CINI: JMP (CASINI) ; INIT CASSETTE
; SPARE BYTE OR MODULE TOO LONG FLAG
CRNTP7 =*
        *=$14
MONSPR: .BYTE KBDORG-CRNTP7 ; GMONITP TOO LONG
        .PAGE
        .TITLE 'DISPLAY HANDLER -- 10-30-78 -- DISPLC'
; HANDLER DEPENDENT EQUATES
CLRCOD =
             $7D
                           ;CLEAR SCREEN ATASCI CODE
CNTL1 =
             $9F
                           ; POKEY KEY CODE FOR ^1
FRMADR = SAVADR
TOADR =
             MLTTMP
        .PAGE
;
        *=EDITRV
; SCREEN EDITOR HANDLER ENTRY POINT
EDITOR: .WORD EOPEN-1
        .WORD RETUR1-1 ; (CLOSE)
        .WORD EGETCH-1
        .WORD EOUTCH-1
        .WORD RETUR1-1
                        ; (STATUS)
        .WORD NOFUNC-1
                       ; (SPECIAL)
        JMP PWRONA
        .BYTE 0
                         ;ROM FILLER BYTE
        *=SCRENV
; DISPLAY HANDLER ENTRY POINT
DISPLA: .WORD DOPEN-1
        .WORD RETUR1-1
                       ; (CLOSE)
        .WORD GETCH-1
        .WORD OUTCH-1
        .WORD RETUR1-1
                        ; (STATUS)
                       ; (SPECIAL)
        .WORD DRAW-1
        JMP PWRONA
        .BYTE 0
                         ; ROM FILLER BYTE
; KEYBOARD HANDLER ENTRY POINT
```

```
KBDHND: .WORD RETUR1-1
          .WORD RETUR1-1
                            ; (CLOSE)
          .WORD KGETCH-1
          .WORD NOFUNC-1 ; (OUTCH)
          .WORD RETUR1-1
                           ; (STATUS)
          .WORD NOFUNC-1
                            ; (SPECIAL)
          JMP PWRONA
          .BYTE 0
                            ;ROM FILLER BYTE
; INTERRUPT VECTOR TABLE ENTRY
          *=VCTABL-INTABS+VKEYBD
          .WORD PIRQ5 ;KEYBOARD IRQ INTERRUPT VECTOR
          *=KBDORG
PWRONA: LDA #$FF
          STA
                CH
               MEMTOP+1
          LDA
          AND
                            ; INSURE 4K PAGE BOUNDARY
                #$F0
         STA RAMTOP
LDA #$40
STA SHFLOK
                            ; DEFAULT TO UPPER CASE ALPHA AT PWRON
         RTS
                                 ; POWER ON COMPLETED
          .PAGE
; BEGIN DISPLAY HANDLER OPEN PROCESSING
DOPEN: LDA ICAX2Z
                      GET AUX 2 BYTE
         AND #$F
         BNE OPNCOM ; IF MODE ZERO, CLEAR ICAX1Z
DA ICAX1Z ; CLEAR "CLR INHIBIT" AND "MXD MODE" BITS
EOPEN: LDA ICAX1Z
          AND #$F
          STA
                ICAX1Z
         LDA
                 #0
OPNCOM: STA DINDEX
          LDA #$E0
                         ; INITIALIZE GLOBAL VBLANK RAM
          STA CHBAS
                #2
          LDA
         STA CHACT
STA SDMCTL
LDA #SUCCES
STA DSTAT
LDA #$C0
ORA POKMSK
                               ;TURN OFF DMA NEXT VBLANK
                           ;CLEAR STATUS
                            ;DO IRQEN
                POKMSK
IRQEN
          STA
          STA
          LDA
                #0
         STA TINDEX ;TEXT INDEX MUST ALWAYS BE 0
STA ADRESS
STA SWPFLG
                 CRSINH
                               ;TURN CURSOR ON ATOPEN
          STA
          LDY
                 #14
                               ;CLEAR TAB STOPS
                               ; INIT TAB STOPS TO EVERY 8 CHARACTERS
          LDA
                 #1
CLRTBS: STA TABMAP, Y
          DEY
          BPL
                CLRTBS
          LDX
                 #4
                                ;LOAD COLOR REGISTERS
```

```
DOPEN8: LDA COLRTB,X
         STA
               COLOR0,X
         DEX
              DOPEN8
         BPL
                RAMTOP
                          ;DO TXTMSC=$2C40 (IF MEMTOP=3000)
         LDY
         DEY
         STY
               TXTMSC+1
         LDA
                #$60
         STA
                TXTMSC
         LDX
               DINDEX
               ANCONV,X ; CONVERT IT TO ANTIC CODE DOPENA ; IF ZERO, IT IS ILLEGA
         LDA
         BNE
                              ; IF ZERO, IT IS ILLEGAL
OPNERR: LDA #BADMOD ;SET ERROR STATUS
         STA DSTAT
DOPENA: STA HOLD1
         LDA RAMTOP
                          ;SET UP AN INDIRECT POINTER
         STA
                ADRESS+1
         LDY
               ALOCAT, X ; ALLOCATE N BLOCKS OF 40 BYTES
DOPEN1: LDA #40
                DBSUB
         JSR
         DEY
             DOPEN1
         BNE
         LDA
               GPRIOR
                             ; ICLEAR GTIA MODES
         AND
                #$3F
         STA
                OPNTMP+1
         TAY
         CPX #8
BCC NOT8
                              ;TEST IF 320X1
         TXA
                               GET 2 LOW BITS
               A
         ROR
         ROR
               A
         ROR
                Α
             #$C0
OPNTMP+1
                       ;NOW 2 TOP BITS
         AND
         ORA
         TAY
               #16
DBSUB
                              ;SUBTRACT 16 MORE FOR PAGE BOUNDARY
         LDA
         JSR
         CPX
               #11
                             ;TEST MODE 11
             NOT8
                           ; IF MODE = 11
         BNE
         LDA
               #6
                              ; PUT GTIA LUM VALUE INTO BACKGROUND REGISTER
         STA
                COLOR4
                         ;STORE NEW PRIORITY
NOT8:
       STY GPRIOR
         LDA ADRESS
                         ; SAVE MEMORY SCAN COUNTER ADDRESS
         STA
                SAVMSC
         LDA
                ADRESS+1
              COUNT ; WAIT FOR NEXT VBLANK BEFORE MESSING #$7A ; WITH THE DISTRICT
         STA
VBWAIT: LDA VCOUNT
         CMP
               VBWAIT

DBDEC ;START PUTTING DISPLAY LIST RIGHT UNDER 1

PAGETB,X ;TEST IF DISPLAY LIST WILL BE IN TROUBLE
         BNE
         JSR
                          ;START PUTTING DISPLAY LIST RIGHT UNDER RAM
         LDA
                         OF CROSSING A 256 BYTE PAGE BOUNDARY
         BEO
               NOMOD
         LDA
                #$FF
                           ; IF SO, DROP DOWN A PAGE
         STA
                ADRESS
         DEC
               ADRESS+1
                         ; SAVE END OF DISPLAY LIST FOR LATER
NOMOD: LDA ADRESS
         STA SAVADR
         LDA
                ADRESS+1
         STA
                SAVADR+1
```

```
JSR
               DBDDEC
                            ; (DOUBLE BYTE DOUBLE DECREMENT)
         LDA
               #$41
                          ; (ANTIC) WAIT FOR VBLANK AND JMP TO TOP
         JSR
               STORE
              OPNTMP
         STX
                            ; INITIALIZE BOTSCR
         LDA
               #24
         STA
               BOTSCR
                            ; DISALLOW MIXED MODE IF MODE.GE.9
         LDA
               DINDEX
         CMP
               #9
         BCS
              NOTMXD
               ICAX1Z
                            ;TEST MIXED MODE
         LDA
               #$10
         AND
         BEQ
               NOTMXD
         LDA
               #4
         STA
               BOTSCR
               #2
                             ; ADD 4 LINES OF TEXT AT BOTTOM OF SCREEN
         LDX
DOPEN2: LDA #2
              STORE
         JSR
         DEX
         BPL
               DOPEN2
                            ; RELOAD MSC FOR TEXT
         LDY
               RAMTOP
         DEY
         TYA
              STORE
         JSR
               #$60
         LDA
              STORE
         JSR
               #$42
         LDA
         JSR STORE
         CLC
         LDA
               #MXDMDE-NUMDLE ; POINT X AT MIXED MODE TABLE
         ADC
                OPNTMP
         STA
                OPNTMP
NOTMXD: LDY OPNTMP
         LDX
                NUMDLE, Y ; GET NUMBER OF DISPLAY LIST ENTRIES
DOPEN3: LDA HOLD1 ;STORE N DLE'S
               STORE
         JSR
         DEX
         BNE
               DOPEN3
                         ; DO THE MESSY 320X1 PROBLEM
         LDA
               DINDEX
         CMP
               #8
              DOPEN5
         BCC
         LDX
               #93
                            ;GET REMAINING NUMBER OF DLE'S
               RAMTOP
         LDA
                            ; RELOAD MEMORY SCAN COUNTER
         SEC
             #$10
STORE
         SBC
         JSR
         LDA
               #0
         JSR
               STORE
                          ; (ANTIC) RELOAD MSC CODE
         LDA
               #$4F
         JSR
               STORE
DOPEN4: LDA HOLD1 ; DO REMAINING DLE'S
         JSR STORE
         DEX
         BNE
               DOPEN4
DOPEN5: LDA SAVMSC+1 ; POLISH OFF DISPLAY LIST
         JSR
               STORE
         LDA
               SAVMSC
         JSR
               STORE
         LDA
               HOLD1
         ORA
               #$40
```

```
LDA
               #$70
                         ;24 BLANK LINES
         JSR
               STORE
         LDA
               #$70
              STORE
         JSR
                           ; SAVE DISPLAY LIST ADDRESS
         LDA
               ADRESS
         STA
               SDLSTL
         LDA
               ADRESS+1
         STA
               SDLSTL+1
              #$70
STORE
                          ; ADD LAST BLANK LINE ENTRY
         LDA
                        ; ADD LAST BLANK LINE ENT: ; POSITION ADRESS=SDLSTL-1
         JSR
         LDA
               ADRESS
                            ;STORE NEW MEMTOP
               MEMTOP
         STA
         LDA
               ADRESS+1
         STA
              MEMTOP+1
              SAVADR
ADRESS
         LDA
         STA
         LDA
               SAVADR+1
         STA
              ADRESS+1
SDLSTL+1
         LDA
              STORE
SDLSTL
STORE
         JSR
         LDA
         JSR
               DSTAT
                        ; IF ERROR OCURRED ON ALLOCATION, OPEN THE ED
         LDA
               DOPEN9
         BPL
         PHA
                              ;SAVE STATUS
        JSR EOPEN ;OPEN THE EDITOR
         PLA
                              ; RESTORE STATUS
         TAY
                              ; AND RETURN IT TO CIO
         RTS
DOPEN9: LDA ICAX1Z
                        ;TEST CLEAR INHIBIT BIT
        AND #$20
               DOPEN7
         BNE
                           ;CLEAR SCREEN
         JSR
               CLRSCR
                            ; AND HOME TEXT CURSOR (AC IS ZERO)
         STA
               TXTROW
        LDA
               LMARGN
         STA TXTCOL
DOPEN7: LDA #$22 ; EVERYTHING ELSE IS SET UP
        ORA SDMCTL ;SO TURN ON DMACTL
         STA
               SDMCTL
        JMP
               RETUR2
;
GETCH: JSR RANGE ;GETCH DOES INCRSR. GETPLT DOESN'T
         JSR GETPLT
         JSR
               INATAC
                            ; CONVERT INTERNAL CODE TO ATASCII
         JSR
               INCRSB
         JMP
               RETUR1
GETPLT: JSR CONVRT
                        ; CONVERT ROW/COLUMN TO ADRESS
               (ADRESS),Y
         LDA
                DMASK
         AND
SHIFTD: LSR SHFAMT
                        ; SHIFT DATA DOWN TO LOW BITS
         BCS SHIFT1
         LSR
               Α
         BPL SHIFTD ; (UNCONDITIONAL)
SHIFT1: STA CHAR
                            ; RESTORE FLAGS ALSO
        CMP #0
         RTS
```

JSR

STORE

```
OUTCH: STA ATACHR
        JSR
              RANGE
       JSR
               OFFCRS
;
                        ;TEST FOR CLEAR SCREEN
OUTCHA: LDA ATACHR
        CMP
              #CLRCOD
        BNE
               OUTCHE
        JSR
               CLRSCR
            RETUR2
        JMP
                       ;TEST FOR CARRIAGE RETURN
OUTCHE: LDA ATACHR
        CMP #CR
        BNE
               OUTCHB
                           ;DO CR
        JSR
              DOCRWS
        JMP RETUR2
OUTCHB: JSR OUTPLT
        JSR
              INCRSR
        JMP
              RETUR2
;
OUTPLT: LDA SSFLAG ; *****LOOP HERE IF START/STOP FLAG ISNON-0
        BNE OUTPLT
        LDX
               #2
CRLOOP: LDA ROWCRS,X ; SAVE CURSOR LOCATION FOR DRAW LINE TO DRAW
        STA OLDROW, X
        DEX
        BPL CRLOOP
LDA ATACHR
                         ; CONVERT ATASCII (ATACHR) TO INTERNAL (CHAR)
        TAY
                             ;SAVE ATACHR
        ROL
              А
        ROL
              Α
        ROL
              A
        ROL
              Α
        AND #3
                              ;X HAS INDEX INTO ATAINT
        TAX
        TYA
                              ; RESTORE ATACHR
              #$9F ;STRIP OFF COLUMN ADDRESS
        AND
              ATAINT,X ;OR IN NEW COLUMN ADDRESS
        ORA
OUTCH2: STA CHAR
        JSR CONVRT
        LDA
               CHAR
SHIFTU: LSR SHFAMT
                       ;SHIFT UP TO PROPER POSITION
        BCS SHIFT2
               Α
        ASL
        JMP
               SHIFTU
SHIFT2: AND DMASK
              TMPCHR
        STA
                           ; SAVE SHIFTED DATA
        LDA
              DMASK
                         ; INVERT MASK
        EOR
              #$FF
        AND (ADRESS),Y ;MASK OFF OLD DATA
ORA TMPCHR ;OR IN NEW DATA
STA (ADRESS),Y
        RTS
;
RETUR2: JSR GETPLT ; DO CURSOR ON THE WAY OUT
        STA OLDCHR
        LDX
              DINDEX
                           GRAPHICS HAVE INVISIBLE CURSOR
        BNE
               RETUR1
```

```
LDX
               CRSINH
                              ;TEST CURSOR INHIBIT
                RETUR1
         BNE
         EOR
                #$80
                            ;TOGGLE MSB
         JSR OUTCH2 ;DISPLAY IT
RETUR1: LDY DSTAT ; RETURN TO CIO WITH STATUS IN Y
                DSTAT ;SET STATUS= SUCCESSFUL COMPLETION
ATACHR ;PUT ATACHR IN AC BOX -
         LDA #SUCCES
         STA
                           ; PUT ATACHR IN AC FOR RETURN TO CIO
         LDA
                           ; (NON-EXISTENT FUNCTION RETURN POINT)
NOFUNC: RTS
; END OF DISPLAY HANDLER
         .PAGE
EGETCH: JSR SWAP
         JSR ERANGE
         LDA
                BUFCNT
                            ;ANYTHING IN THE BUFFER?
         BNE
                EGETC3
                              ;YES
                ROWCRS
                              ; NO, SO SAVE BUFFER START ADDRESS
         LDA
         STA
                BUFSTR
         LDA
                COLCRS
         STA BUFSTR+1
EGETC1: JSR KGETCH ;LET'S FILL OUR BUFFER STY DSTAT ;SAVE KEYBOARD STATUS
         LDA AILL
CMP #CR
BEQ EGETC2
JSR DOSS
JSR SWAP
LOGCOL
                             ;TEST FOR CR
                       ;NO, GO PRINT IT
;JSR DOSS DID SWAP SO SWAP BACK
;BEEP IF NEARING LOGICAL COL 120
         CMP
                #113
         BNE
                EGETC6
         JSR
                BELL
EGETC6: JMP EGETC1
EGETC2: JSR OFFCRS ;GET BUFFER COUNT
         JSR DOBUFC
         LDA
                BUFSTR
                              ; RETURN A CHARACTER
         STA
                ROWCRS
         LDA
                BUFSTR+1
         STA
                COLCRS
EGETC3: LDA BUFCNT
         BEQ EGETC5
                          ; AND RETURN TILL BUFCNT=0
EGETC7: DEC BUFCNT
         BEO EGETC5
                DSTAT
                          ; IF ERR, LOOP ON EGETC7 UNTIL BUFR IS E1IPTIE
         LDA
         BMI
                EGETC7
         JSR
                GETCH
         STA
                ATACHR
         JMP
                SWAP
                            ; AND RETURN WITHOUT TURNING CURSOR BACK ON
                        ;DO REAL CARRIAGE RETURN
EGETC5: JSR DOCRWS
LDA #CR
                              ; AND RETURN EOL
         STA
                ATACHR
         JSR
                RETUR2
                              ;TURN ON CURSOR THEN SWAP
         STY DSTAT ;SAVE KEYBOARD STATUS
```

```
JSRIND: JMP (ADRESS) ; J5R TO THIS CAUSES JSR INDIRECT
EOUTCH: STA ATACHR ;SAVE ATASCII VALUE
        JSR SWAP
         JSR
               ERANGE
       JSR OFFCRS
                        ;TURN OFF CURSOR
DOSS:
               TSTCTL
         JSR
                          ;TEST FOR CONTROL CHARACTERS (Z=1 IF CTL)
         BEQ EOUTC5
EOUTC6: ASL ESCFLG
                        ; ESCFLG ONLY WORKS ONCE
        JSR OUTCHE
ERETN: JMP SWAP
                      ; AND RETURN THROUGH RETUR1
EOUTC5: LDA DSPFLG
                        ;DO DSPFLG AND ESCFLC
         ORA ESCFLG
               EOUTC6
         BNE
                            ; IF NON-0 DISPLAY RATHER THAN EXECUTE IT
              ESCFLG
         ASL
         INX
                               ; PROCESS CONTROL CHARACTERS
         LDA
              CNTRLS,X ;GET DISPLACEMENT INTO ROUTINE
         STA
               ADRESS
              CNTRLS+1,X ;GET HIGH BYTE
ADRESS+1
JSRIND ;DO COMPUTED
         LDA
         STA
                           ;DO COMPUTED JSR
;DO CURSOR
         USR RETUR2

JMP SWAP
         JSR
               SWAP ;ALL DONE SO RETURN THROUGH RETUR1
 END SCREEN EDITOR.
; BEGIN KEYBOARD HANDLER
;
KGETC2: LDA #$FF
         STA CH
KGETCH: LDA ICAX1Z
                      ;TEST LSB OF AUX1 FOR SPECIAL EDITOR READ MO
         LSR
               Α
         BCS
               GETOUT
         LDA
               #BRKABT
         LDX BRKKEY
BEQ K7
                            ;TEST BREAK
                             ; IF BREAK, PUT BRKABT IN DSTAT AND CRIN ATA
         LDA
               CH
               #$FF
         CMP
               KGETCH
         BEQ
         STA
               HOLDCH
                            ; SAVE CH FOR SHIFT LOCK PROC
         LDX
              #$FF
                          ; "CLEAR" CH
         STX
               CH
         JSR
               CLICK
                          ; DO KEYBOARD AUDIO FEEDBACK (A IS OK)
KGETC3: TAX
                           ; DO ASCCON
              #$C0
         CPX
                          ;TEST FOR CTL & SHIFT TOGETHER
         BCC
               ASCC01
                             ;BAD CODE
         LDX
                #3
ASCCO1: LDA ATASCI,X
                            ; DONE
         STA
              ATACHR
                         ;DO NULLS
         CMP
                #$80
```

; AND RETURN THROUGH RETUR1

SWAP

JMP

```
CMP
                #$81
                           ;CHECK ATARI KEY
         BNE
               KGETC1
               INVFLG
         LDA
               #$80
         EOR
         STA INVFLG
JMP KGETC2
                             ; DONT RETURN A VALUE
KGETC1: CMP #$82 ;CAPS/LOWER.
         BNE K1
         LDA
                #0
                              ;CLEAR SHFLOK
         STA
                SHFLOK
         BEQ KGETC2
K1: CMP #$83 ;SHIFT CAPS/LOWER
         BNE K2
         LDA
               #$40
                SHFLOK ;SHIFT BIT
         STA
         BNE KGETC2
K2: CMP #$84 ;CNTL CAPS/LOWER
         BNE K3
               #$80 ;CNTL BIT
         LDA
                SHFLOK
         STA
         BNE KGETC2
K3: CMP #$85 ;DO EOF
         BNE K6
         LDA #EOFERR
K7: STA DSTAT
STA BRKKEY ; RESTORE BREAK

GETOUT: LDA #CR ; PUT CR IN ATACHR

BNE K8 ; (UNCONDITIONAL)

K6: LDA HOLDCH ; PROCESS SHIFT LOCKS
                         REGULAR SHIFT AND CONTROL TAKE PRECEDENCE
         CMP #$40
BCS K5
         BCS K5 , OVER LS.
LDA ATACHR ; TEST FOR ALPHA
CMP #$61 ; LOWER CASE A
; NOT ALPHA IF :
                              ;NOT ALPHA IF LT
               #$7B
K5
                          ;LOWER CASE Z+1
         CMP
         BCS
                              ;NOT ALPHA IF GE
         LDA SHFLOK
BEQ K5
                           ;DO SHIFT/CONTROL LOCK
                              ; IF NO LOCK. DONT RE-DO IT
         ORA HOLDCH
JMP KGETC3
                             ; DO RETRY
     JSR TSTCTL
                         ; DONT INVERT M58 OF CONTROL CHARACTERS
K5:
         BEQ K4
         LDA
                ATACHR
         EOR INVFLG
K8: STA ATACHR
K4: JMP
           RETUR1 ; ALL DONE
         .PAGE
; CONTROL CHARACTER PROCESSORS
ESCAPE: LDA #$80 ;SET ESCAPE FLAG
         STA ESCFLG
         RTS
CRSRUP: DEC ROWCRS
         BPL COMRET
```

BEO

KGETC2

```
LDX BOTSCR
                          ;WRAPAROUND
        DEX
UPDNCM: STX ROWCRS
COMRET: JMP STRBEG
                    ; CULVERT ROW AND COL TO LOGCOL AND RETURN
CRSRDN: INC ROWCRS
        LDA
             ROWCRS
        CMP
              BOTSCR
        BCC
              COMRET
        LDX
              #0
        BEQ UPDNCM
                     ; (UNCONDITIONAL)
CRSRLF: DEC COLCRS
        LDA COLCRS
                           ;(IF LMARGN=0, THIS ELIMINATES PROBLEM CASE)
        BMI
              CRSRL1
        CMP
              LMARGN
        BCS COMRE1
CRSRL1: LDA RMARGN
LFRTCM: STA COLCRS
COMRE1: JMP DOLCOL ; COLVERT ROW AND COL TO LOGCOL AND RETURN
CRSRRT: INC COLCRS
        LDA COLCRS
              RMARGN
        CMP
        BCC
              COMRE1
        BEQ
              COMRE1
                        ;(CAUSE OLE)
        LDA
              LMARGN
        JMP
             LFRTCM
                          ;UNCONDITIONAL TO COMMON STORE
CLRSCR: JSR PUTMSC
        LDY #0
                             ; PUT 0 IN THEAC
        TYA
CLRSC2: STA (ADRESS), Y ; (AC IS ZERO)
        TNY
        BNE
              CLRSC2
        INC
              ADRESS+1
             ADRESS+1
        LDX
              RAMTOP
        CPX
        BCC
              CLRSC2
        LDA
              #$FF
                         CLEAN UP LOGICAL LINE BITMAP
CLRSC3: STA LOGMAP, Y ; (Y IS ZERO AFTER CLRSC2 LOOP)
        INY
        CPY
              #4
        BCC
              CLRSC3
      JSR COLCR ; PLACE COLCRS AT LEFT EDGE
HOME:
        STA LOGCOL
        STA
              BUFSTR+1
        LDA
              #0
        STA ROWCRS
STA COLCRS+1
              BUFSTR
        STA
        RTS
BS: LDA LOGCOL ; BACKSPACE
        CMP
             LMARGN
        BEQ
              BS1
                           ;LEFT EDGE?
BSA:
      LDA
              COLCRS
        CMP
              LMARGN
        BNE
              BS3
                           ; NO
             DELTIM
CRSRLF
```

;YES, SEE IF LINE SHOULD BE DELETED

JSR

JSR

BS3:

LIDA COLCRS
CMP

```
BNE
               BS2
         LDA
               ROWCRS
         BEQ
               BS2
              CRSRUP
#$20
ATACHR
         JSR
                         ; MAKE BACKSPACE DESTRUCTIVE
BS2:
       LDA
        STA
        JSR
               OUTPLT
      JMP DOLCOL
JSR CRSRRT
LDA COLCRS
CMP LMARGN
                            ; AND RETURN
BS1:
TAB:
                            ;BEGIN SEARCH
                            ;TEST FOR NEW LINE
              TABL
DOCR ;DU C...
LOGGET ;CHECK IF END C
TAB1 ;NO, CONTINUE
;(UNCONDITIONAL)
FOR TAB STO
                          ;NO
;DO CARRIAGE RETURN
         JSR
                            ; CHECK IF END OF LOGICAL LINE
         JSR
         BCC
             TAB2
         BCS
     LDA LOGCOL
                        ; CHECK FOR TAB STOP
TAB1:
        JSR BITGET
                             ; NO, SO KEEP LOOKING
        BCC
                TAB
TAB2: JMP DOLCOL
                     ; CULVERT ROW AND COL TO LOGCOL AND RETURN
SETTAB: LDA LOGCOL
               BITSET ;SET BIT IN MAP AND RETURN
        JMP
CLRTAB: LDA LOGCOL
                            ;CLEAR " " " " "
        JMP BITCLR
INSCHR: JSR PHACRS
         JSR GETPLT
                            GET CHARACTER UNDER CURSOR
               INSDAT
         STA
         LDA
               #0
         STA SCRFLG
INSCH4: JSR OUTCH2
                         ;STORE DATA
         LDA LOGCOL
                            ; SAVE LOGCOL: IF AFTER INCRSA LOGCOL IS
         PHA
                              ; < THAN IT IS NOW, END LOOP
         JSR INCRSA ;SPECIAL INCRSR ENTRY POINT
         PLA
         CMP
               LOGCOL
         BCS INSCH3
                            ;QUIT
INSCH1: LDA INSDAT
                        ;KEEP GOING
        PHA
         JSR GETPLT
         STA
               INSDAT
         PLA
         JMP INSCH4
INSCH3: JSR PLACRS
INSCH6: DEC SCRFLG
              INSCH5 ; IF SCROLL OCCURRED
         BMI
         DEC
               ROWCRS
                            ; MOVE CURSOR UP
               INSCH6
                             ; (UNCOND) CONTINUE UNTIL SCRFLG IS MINUS
         BNE
INSCH5: JMP DOLCOL ;CULVERT ROW AND COL TO LOGCOL AND RETURN
DELCHR: JSR PHACRS
DELCH1: JSR CONVRT
                        ;GET DATA TO THE RIGHT OF THE CURSOR
         LDA ADRESS
              SAVADR
         STA
                            ; SAVE ADRESS TO KNOW WHERE TO PUT DATA
         LDA
               ADRESS+1
              SAVADR+1
LOGCOL
         STA
         LDA
         PHA
         JSR INCRSB
                            ; PUT CURSOR OVER NEXT CHARACTER
```

```
LOGCOL
DELCH2
           CMP
                                   ;TEST NEW LOGCOL AGAINST OLD LOGCOL
                                   ; IF OLD.GE.NEW THEN QUIT
           BCS
                  ROWCRS
BOTSCR
DELCH2
           LDA
                                   ; IS ROW OFF SCREEN?
           CMP
                                  ;YES, SO QUIT
           BCS
                                   GET DATA UNDER CURSOR
           JSR
                  GETPLT
                   #0
           LDY
           STA (SAVADR),Y ;PUT IT IN PREVIOUS POSITION BEQ DELCH1 ;AND LOOP (UNCONDITIONAL)
DELCH2: LDY #0
           TYA
           STA (SAVADR),Y ;CLEAR THE LAST POSITION
JSR DELTIA ;TRY TO DELETE A LINE
JSR PLACRS
JMP DOLCOL ;AND RETURN
                               ;AND RETURN
;INSLIN PUTS "1" INTO BIT MAP
INSLIN: SEC
INSLIA: JSR EXTEND ; ENTRY POINT FOR C=0
           LDA LMARGN
                                   ; DO CARRIAGE RETURN (NO LF)
           STA
                   COLCRS
                 COLCRS
CONVRT ;GET ADDRESS
ADRESS ;SET UP TO=40
FRMADR
           JSR
           LDA
                                   ;SET UP TO=40+FROM (FROM = CURSOR)
           STA
           CLC
                  #40
TOADR
ADRESS+1
FRMADR+1
           ADC
           STA
           LDA
           STA
           ADC
                  #0
                  TOADR+1
ROWCRS
           STA
           LDX
                                  ;SET UP LOOP COUNTER
           CPX
                   #23
           BEQ INSLI2
INSLI1: JSR MOVLIN
           INX
           CPX
                   #23
           BNE INSLI1
INSLI2: JSR CLRLIN ;CLEAR CURRENT LINE

JMP DOLCOL ;COLVERT ROW AND COL TO LOGCOL

DELLIN: JSR DOLCOL ;GET BEGINNING OF LOG LINE (HOLD1)

DELLIA: LDY HOLD1 ;SQUEEZE BIT MAP
                                   ; COLVERT ROW AND COL TO LOGCOL AND RETURN
                               ; PUT CURSOR THERE
           STY ROWCRS
DELLIB: LDY ROWCRS
DELLI1: TYA
           SEC
           JSR LO2GET ;GET NEXT BIT
           PHP
           TYA
           CLC
           ADC
                #120
           PLP
           JSR
                  BITPUT
                                   ;WRITE IT OVER PRESENT BIT
           INY
           CPY
                   #24
                   DELLI1
                                  ;LOOP
           BNE
                   LOGMAP+2 ;SET LSB
           LDA
           ORA
                   #1
           STA
                   LOGMAP+2
DELLI2: LDA LMARGN
                              ; DELETE LINE OF DATA USING PART OF SCROLL
```

PLA

```
COLCRS
        STA
                            ;CR NO LF
         JSR
               CONVRT
        JSR
               SCROL1
        JSR
              LOGGET
                            ;TEST NEXT LINE FOR CONTINUATION
; IS IT A NEW LOG LINE?
        BCC DELLIB
                           ; NO SO DELETE ANOTHER
        JMP
               DOLCOL
                            ;YES SO DOLCOL AND RETURN
      LDY #$20
BELL:
BELL1: JSR CLICK
        DEY
        BPL BELL1
        RTS
         .PAGE
; ROUTINES
; DOUBLE BYTE DECREMENT OF INDIRECT POINTER
; INCLUDING DB SUBTRACT AND DB DOUBLE DECREMENT
DBDDEC: LDA #2
       BNE DBSUB ; (UNCONDITIONAL)
; STORE DATA INDIRECT AND DECREMENT POINTER
; (PLACED HERE TO SAVE JMP DBDEC AFTER STORE)
                  ; RETURN ON ERROR
STORE: LDY DSTAT
        BMI STROK
        LDY
               #0
STOREI: STA (ADRESS),Y
      JMP
             DBDEC ; DECREMENT AND RETURN
DBDEC: LDA #1
DBSUB: STA SUBTMP
        LDA DSTAT
                         ; RETURN ON ERROR
              STROK
        BMI
        LDA
               ADRESS
        SEC
        SBC SUBTMP STA ADRESS
        BCS
              DBSUB1
        DEC
              ADRESS+1
DBSUB1: LDA APPMHI+1 ; MARE SURE NOTHING EVER OVERWRITES APPMHI
        CMP
              ADRESS+1
        BCC
              STROK ;OK
        BNE
              STRERR
                          ; ERROR
               APPMHI
        LDA
        CMP
               ADRESS
        BCC STROK
STRERR: LDA #SCRMEM ;SHOW MEM TOO SMALL FOR SCREEN ERROR
        STA DSTAT
STROK: RTS
; CONVERT ROW/COLUMN CURSOR INTO REAL ADDRESS (FROM SAVMSC ON UP)
CONVRT: LDA ROWCRS ; SAVE CURSOR
        PHA
```

```
LDA
               COLCRS
         PHA
         LDA
               COLCRS+1
         PHA
              PUTMSC
         JSR
                            ; PUT 10*ROWCRS INTO MLTTMP
         LDA
               ROWCRS
         STA
               MLTTMP
         LDA
                #0
         STA
               MLTTMP+1
               MLTTMP
                          ;QUICK X8
         LDA
         ASL
               Α
         ROL
               MLTTMP+1
         STA
               HOLD1
                         ;(SAVE 2X VALUE)
         LDY
               MLTTMP+1
         STY
               HOLD2
         ASL
         ROL
              MLTTMP+1
         ASL
               Α
         ROL
               MLTTMP+1
         CLC
                              ;ADD IN 2X
              HOLD1
         ADC
         STA
               MLTTMP
         LDA
               MLTTMP+1
         ADC
               HOLD2
               MLTTMP+1
         STA
         LDX
               DINDEX
                            ; NOW SHIFT MLTTMP LEFT DHLINE TIMES TO FINIS
         LDY DHLINE,X ;MULTIPLY
CONVR1: DEY
                          ;LOOP N TIMES
              CONVR2
         BMI
         ASL
               MLTTMP
         ROL
               MLTTMP+1
         JMP
               CONVR1
CONVR2: LDY DIV2TB,X ; NOW DIVIDE HCRSR TO ACCOUNT FOR PARTIAL BYT
         LDA
               COLCRS
         LDX
               #7
                             ; * TRICKY *
CONVR3: DEY
               CONVR4
         BMI
         DEX
             COLCRS+1
         LSR
         ROR
               Α
               TMPLBT
                            ; SAVE LOW BITS FOR MASK
         ROR
         JMP
               CONVR3
CONVR4: INY
                           ;SO Y IS ZERO UPON RETURN FROM THIS ROUTINE
         CLC
                            ; ADD SHIFTED COLCRS TO MLTThP
         ADC
              MLTTMP
         STA
               MLTTMP
         BCC
                CONVR5
         INC
               MLTTMP+1
CONVR5: SEC
                           ; * TRICKY *
CONVR6: ROR TMPLBT ;SLIDE A "1" UP AGAINST LOW BITS (CONTINUE T
         CLC
         DEX
                               ; AND FINISH SHIFT SO LOW BITS ARE
         BPL
               CONVR6
                             ; RIGHT JUSTIFIED.
         LDX
               TMPLBT
                             ;TMPLBT IS NOW THE INDEX INTO DMASKTB
                             ; PREPARE FOR RETURN
         LDA
               MLTTMP
         CLC
         ADC
              ADRESS
         STA
               ADRESS
         STA
                OLDADR
                             ; REMEMBER THIS ADDRESS FOR CURSOR
```

```
LDA
               MLTTMP+1
         ADC
               ADRESS+1
         STA
               ADRESS+1
              OLDADR+1
DMASKT,X
         STA
         LDA
         STA
               DMASK
         STA
               SHFAMT
         PLA
              COLCRS+1
         STA
         PLA
         STA COLCRS
         PLA
               ROWCRS
         STA
         RTS
; INCREMENT CURSOR AND DETECT BOTH END OF LINE AND END OF SCREEN
INCRSB: LDA #0
                         ; NON-EXTEND ENTRY POINT
        BEQ INCREC
INCRSR: LDA #$9B ;SPECIAL CASE ELIMINATOR
INCREC: STA INSDAT
INCRSA: INC LOGCOL ; (INSCHR ENTRY POINT)
        INC
              COLCRS
         BNE
               INCRS2
                            ;DO HIGH BYTE
        INC
               COLCRS+1
                       ;TEST END OF LINE
INCRS2: LDA COLCRS
         LDX DINDEX
         CMP
               COLUMN,X ; TEST TABLED VALUE FOR ALL SCREEN MODES
               INC2A ; DO CR IF EQUAL
         BEQ
        CPX
              #0
                             ; MODE 0?
              INCRS3
RMARGN
INCRS3
         BNE
                            ; IF NOT. JUST RETURN
        CMP
                            ;TEST AGAINST RMARGN
                            ; EGUAL IS OK
        BEQ
        BCS
               INC2A
                         ; IF GREATER THAN, DO CR
INCRS3: RTS
INC2A: CPX #8
                         ; CHECK MODE
        BCC DOCR1 ;NOT 320X1 $0 DO IT
LDA COLCRS+1 ;TEST MED
        BEQ
               INCRS3
                            ;ONLY AT 64 SO DON'T DO IT
DOCR1: LDA DINDEX
                        ;DON'T MESS WITH LOGMAP IF NO MODE ZERO
         BNE DOCR
              LOGCOL ;TEST LINE OVERRUN
         LDA
        CMP
               #81
              DOCR1B ; IF LESS THAN 81 IT IS DEFINITELY NOT LINE 3
         BCC
         LDA
               INSDAT
                          ;ONLY DO LOG LINE OVERFLOW IF INSDAT <>0
         BEQ
               DOCR
         JSR
               DOCRWS
                            ;LOG LINE OVERFLOW IS SPECIAL CASE
        JMP INCRS1
                            ; RETURN
DOCR1B: JSR DOCR ;GET IT OVER WITH
         LDA ROWCRS
         CLC
                              ;TEST LOGICAL LINE BIT MAP
         ADC
              #120
         JSR
               BITGET
                            ;DON'T EXTEND IF OVERRUN IS INTO MIDDLE OF L
         BCC
               DOCR1A
              INSDAT
DOCR1A
                            ;DON'T EXTEND IF INSDAT IS ZERO
         LDA
                            ; (INSCHR SPECIAL CASE)
         BEQ
         CLC
                              ; INSERT "0" INTO BIT MAP
         JSR INSLIA
```

```
DOCR1A: JMP DOLCOL ;CONVERT ROW AND COL TO LOGCOL AND RETURN NOSCRL: LDA #0 ;DOCR WITHOUT SCROLL
        LDA #0 ;DOCR WITHOUT SCROLL BEQ NOSCR1 ;(UNCONDITIONAL)
DOCRWS: LDA #$9B ; DOCR WITH SCROLLING (NORMAL MODE)
NOSCR1: STA INSDAT
DOCR: JSR COLCR ; PLACE COLCRS AT LEFT EDGE
         LDA
                #0
         STA
                 COLCRS+1
         INC
                ROWCRS
DOCR2: LDX DINDEX
                             ;SET UP SCROLL LOOP COUNTER
         LDY #24
         BIT
                SWPFLG
         BPL
                DOCR2A
                              ;BRANCH IF NORMAL
         LDY
                #4
         TYA
         BNE DOCR2B
                         ; (UNCONDITIONAL)
DOCR2A: LDA NOROWS,X ;GET NO OF ROWS
DOCR2B: CMP ROWCRS
         BNE
                 INCRS1
         STY
                HOLD3
         TXA
                               ;DON'T SCROLL IF MODE <> 0
         BNE INCRS1
LDA INSDAT
                              ;OR IF INSDAT = 0
         BEQ INCRS1

LDA INSDAT IF INSDAT <> $9B THEN ROLL II

CMP #$9B ;TO EXTEND BOTTOM LOGICAL LINE
         BEQ
                              IF INSDAT <> $9B THEN ROLL IN A 0
        LDA
         SEC
         BEQ DOCR4B
         CLC
DOCR4B: JSR SCROLL ;LOOP SACK TO HERE IF >1 SCROLLS
         INC SCRFLG
         DEC
                BUFSTR ; ROWS MOVE UP SO BUFSTR SHOULD TOO
               HOLD3
LOGMAP
         DEC
         LDA
                                 FOR PARTIAL LINES ROLL IN A "1"
         SEC
         BPL
                DOCR4B
                              ; AGAIN IF PARTIAL LOGICAL LINE
                HOLD3
                          ; PLACE CURSOR AT NEW LINE NEAR THE BOTTOM
         LDA
         STA
                ROWCRS
INCRS1: JMP DOLCOL ; COLVERT ROW AND COL TO LOGCDL AND RETURN
; SUBEND: SUBTRACT ENDPT FROM ROWAC OR COLAC. (X=0 OR 2)
SUBEND: SEC
         LDA ROWAC,X
         SBC
                ENDPT
         STA ROWAC, X
LDA ROWAC+1, X
SBC ENDPT+1
STA ROWAC+1, X
         RTS
; RANGE: DO CURSOR RANGE TEST. IF ERROR, POP STACK TWICE AND JMP RETURN
         (ERANGE IS EDITOR ENTRY POINT AND TEST IF EDITOR IS OPEN.
          IF IT ISNT IT OPENS THE EDITOR AND CONTINUES)
ERANGE: LDA BOTSCR ; IF BOTSCR=4
         CMP #4
```

```
BEO
                RANGE ; THEN IT IS IN MIXED NODE AND OK
                DINDEX
         LDA
                             ; IF MODE = 0
         BEQ RANGE ;THEN IT IS INEDITOR MODE AND OK JSR EOPEN ;IF NOT, OPEN EDITOR
                          ;**** RANGE CHECK RMARGN **** SET UP AC
RANGE: LDA #39
         CMP RMARGN
                              ; **** RANGE CHECK RMARGN **** COMPARE
         BCS
                RANGE 3
                               ; **** RANGE CHECK RMARGN **** BRANCH GE
                               ; **** RANGE CHECK RMARGN **** BAD SO STORE
         STA
                RMARGN
RANGE3: LDX DINDEX
               NOROWS,X ; CHECK ROWS
         I_1DA
                ROWCRS
RNGERR
         CMP
         BCC
                              ; (ERROR IF TABLE.GE.ROWCRS)
         BEQ RNGERR
CPX #8 ;CHECK FOR 320X1
BNE RANGE1 ;SPECIAL CASE IT
LDA COLCRS+1
BEQ RNGOK ;IF HIGH BYTE IS 0, COL IS OK
         CMP
                #1
                RNGERR
         BNE
                              ; IF >1, BAD
         BEQ RANGE2
                              ; IF 1, GO CHECK LOWBYTE
RANGE1: LDA COLCRS+1 ; FOR OTHERS, NON-ZERO HIGH BYTE IS BAD
         BNE RNGERR
RANGE2: LDA COLUMN, X ; CHECK LOWBYTE
         CMP
               COLCRS
         BCC
                RNGERR
         BEQ RNGERR
RNGOK: LDA #SUCCES ;SET STATUS OK
         STA DSTAT
         LDA
                #BRKABT ; PREPARE BREAK ABORT STATUS
         STA BRKKEY
BEQ RNGFP^
RTS
                BRKKEY
                              ; CHECK BREAK KEY FLAG
                              ; 'CLEAR' BREAK
                RNGER2
                              ; IF BREAK, QUIT IMMEDIATELY AND RETURN TO CI
RNGERR: JSR HOME ;ON RANGE ERROR, BRING CURSOR BACK
        LDA #CRSROR ;SHOW CURSOR OVERRANGE ERROR
RNGER2: STA DSTAT
RNGER1: PLA
                            ; RESTORE STACK (THIS ROUTINE IS ALWAYS 1 LEV
                                ; AWAY FROM RETURN TO CIO)
         PTA
               SWPFLG
                              ; IF SWAPPED. SWAP BACK
         LDA
                RETUR3
         BPL
                SWAPA ; AND DONT DO RETUR1

FUR1 ; RETURN TO CIO
         JSR
RETUR3: JMP RETUR1
; OFFCRS: RESTORE OLD DATA UNDER CURSOR SO IT CAN BE MOVED
OFFCRS: LDY #0
         LDA OLDCHR
         STA
                (OLDADR),Y
         RTS
;
; BITMAP ROUTINES:
; BITCON: PUT MASK IN BITMSK AND INDEX IN X
; BITPUT: PUT CARRY INTO BITMAP
; BITROL: ROL CARRY INTO BOTTOM OF BITMAP (SCROLL)
```

```
; BITCLR: CLEAR PROPER BIT
; BITGET: RETURN CARRY SET IF BIT IS THERE
; LOGGET: DO BITGET FOR LOGMAP INSTEAD OF TABMAP
BITCON: PHA
               #7
         AND
         TAX
                               GET MASK
         LDA MASKTB,X
STA BITMSK
         PLA
                              ; PROCESS INDEX
               Α
         LSR
         LSR
LSR
               Α
               Α
         TAX
         RTS
;
BITROL: ROL LOGMAP+2
         ROL LOGMAP+1
         ROL LOGMAP
         RTS
BITPUT: BCC BITCLR
                    ; AND RETURN
; OTHERWISE FALL THROUGH TO BITSET AND RETURN
BITSET: JSR BITCON
         LDA TABMAP,X
         ORA
               BITMSK
         STA
               TABMAP,X
         RTS
BITCLR: JSR BITCON
         LDA BITMSK
         EOR
               #$FF
         AND
               TABMAP,X
         STA
               TABMAP,X
         RTS
LOGGET: LDA ROWCRS
LO1GET: CLC
LO2GET: ADC #120
BITGET: JSR BITCON
         CLC
         LDA
              TABMAP,X
         AND
               BITMSK
               BITGE1
         BEQ
         SEC
BITGE1: RTS
; INATAC: INTERNAL(CHAR) TO ATASCII(ATACHR) CONVERSION
INATAC: LDA CHAR
         LDY DINDEX
                           ; IF GRAPHICS MODES
         CPY
               #3
         BCS INATA1
                             ;THEN DON'T CHANGE CHAR
```

; BITSET: SET PROPER BIT

```
ROL
               Α
         ROL
                Α
         ROL
               Α
         ROL
               Α
               #3
         AND
         TAX
         LDA
               CHAR
         AND
               #$9F
             INTATA,X
         ORA
INATA1: STA ATACHR
        RTS
; MOVLLN: MOVE 40 BYTES AT FRMADR TO TOADR SAVING OLD TOADR
       DATA IN THE LINBUF. THEN MAKE NEXT FRMADR
       BE AT LINBUF FOR NEXT TRANSFER & TOADR=TOADR+40
MOVLIN: LDA #LINBUF/256 ;SET UP ADRESS=LINBUF$=247
         STA ADRESS+1
         LDA
               #LINBUF.AND.$FF
         STA
                ADRESS
         LDY
               #39
MOVLI1: LDA (TOADR), Y ; SAVE TO DATA
         STA
               TMPCHR
         LDA
               (FRMADR),Y ;STORE DATA
         STA
               (TOADR),Y
              TMPCHR
         LDA
         STA
               (ADRESS),Y
         DEY
              MOVLI1
         \mathsf{BPL}
         LDA
               ADRESS+1 ;SET UP FRMADR=LAST LINE
         STA
               FRMADR+1
         LDA
               ADRESS
         STA
               FRMADR
                              ; ADD 40 TO TOADR
         CLC
              TOADR
         LDA
         ADC
               #40
              TOADR
         STA
         BCC
               MOVLI2
         INC
               TOADR+1
MOVLI2: RTS
; EXTEND: EXTEND BIT MAP FROM ROWCRS (EXTEND LOGICAL LINE
EXTEND: PHP
                          ; SAVE CARRY
        LDY
               #23
EXTEN1: TYA
             LO1GET
         JSR
         PHP
         TYA
         CLC
         ADC
               #121
         PLP
         JSR
               BITPUT
EXTEN3: DEY
         BMI EXTEN4
```

```
CPY ROWCRS
         BCS
                 EXTEN1
EXTEN4: LDA ROWCRS
         CLC
                #120
         ADC
         PLP
         JMP BITPUT ;STORE NEW LINE'S BIT AND RETURN
;
; CLRLIN: CLEAR LINE CURSOR IS ON
CLRLIN: LDA LMARGN
         STA COLCRS
         JSR CONVRT
         LDY
                #39
                #0
         LDA
CLRLI1: STA (ADRESS), Y
         DEY
         BPL CLRLI1
         RTS
;
; SCROLL: SCROLL SCREEN
SCROLL: JSR BITROL ;ROLL IN CARRY LDA SAVMSC ;SET UP WOR
                          ;SET UP WORKING REGISTERS
         STA
                ADRESS
         LDA
                SAVMSC+1
         STA ADRESS+1
SCROL1: LDY #40 ;LOOP
         LDA (ADRESS),Y
         LDX
                RAMTOP
                              ;TEST FOR LAST LINE
         DEX
         CPX ADRESS+1
BNE SCROL2
LDX #$D7
CPX ADRESS
         BCS
                SCROL2
         LDA
                #0
                               ;YES SO STORE ZERO DATA FOR THIS ENTIRE LINE
SCROL2: LDY #0
               (ADRESS),Y
ADRESS
         STA
         INC
         BNE
                SCROL1
         INC ADRESS+1
LDA ADRESS+1
CMP RAMTOP
BNE SCROL1
JMP DOLCOL
                           ; AND RETURN
; DOLCOL: DO LOGICAL COLUMN FROM BITMAP AND COLCRS
DOLCOL: LDA #0
                           ;START WITH ZERO
         STA LOGCOL
         LDA
                ROWCRS
          STA
                HOLD1
```

```
DOLCO1: LDA HOLD1 ;ADD IN ROW COMPONENT
         JSR LO1GET
         BCS
               DOLCO2
                           ; FOUND BEGINNING OF LINE
         LDA
               LOGCOL
                          ; ADD 40 AND LOOK BAC ONE
         CLC
              #40
         ADC
               LOGCOL
         STA
         DEC
               HOLD1
                         ;UP ONE LINE
         JMP
               DOLCO1
DOLCO2: CLC
                          ;ADD IN COLCRS
        LDA LOGCOL ADC COLCRS
         STA
               LOGCOL
         RTS
;
; DOBUFC: COMPUTE BUFFER COUNT AS THE NUMBER OF BYTES FROM
          BUFSTR TO END OF LOGICAL LINE WITH TRAILING SPACES REMOVED
DOBUFC: JSR PHACRS
         LDA LOGCOL
         PHA
        LDA
              BUFSTR
                            ;START
              ROWCRS
BUFSTR+1
COLCRS
         STA
        LDA
         STA
        LDA #1
STA BUFCNT
DOBUF1: LDX #23
                        ;NORMAL
         LDA SWPFLG
                          ; IF SWAPPED, ROW 3 IS THE LAST LINE ON SCREE
         BPL
               DOB1
         LDX
               #3
       CPX ROWCRS
                        ;TEST IF CRSR IS AT LAST SCREEN POSITION
DOB1:
        BNE DOBU1A
         LDA
               COLCRS
         CMP
               RMARGN
         BNE
               DOBU1A
         INC BUFCNT JMP DOBUF2
                          ; YES, SO FAKE INCRSP TO AVOID SCROLLING
DOBU1A: JSR INCRSB
         INC
              BUFCNT
         LDA
               LOGCOL
         CMP
               LMARGN
                           ; NOT YET EOL
         BNE
              DOBUF1
         DEC
               ROWCRS
                            ;BACK UP ONE INCRSR
         JSR
               CRSRLF
                        ;TEST CURRENT COLUMN FOR NON-ZERO DATA
DOBUF2: JSR GETPLT
         BNE DOBUF4
                            ;OUIT IF NON-ZERO
         DEC
               BUFCNT
                            ; DECREMENT COUNTER
                            ;BEGINNING OF LOGICAL LINE YET?
         LDA
               LOGCOL
         CMP
               LMARGN
                            ; YES, SO QUIT
         BEQ
               DOBUF4
                            ;BACK UP CURSOR
         JSR
               CRSRLF
               COLCRS
                            ; IF LOGCOL=RMARGN, GO UP 1 ROW
         LDA
         CMP
               RMARGN
         BNE
               DOBUF3
        DEC
               ROWCRS
DOBUF3: LDA BUFCNT
```

```
BNE
               DOBUF2
                            ;LOOP UNLESS BUFCNT JUST WENT TO ZERO
DOBUF4: PLA
         STA
               LOGCOL
         JSR PLACRS
         RTS
; STRBEG: MOVE BUFSTR TO BEGINNING OF LOGICAL LINE.
STRBEG: JSR DOLCOL
                        ;USE DOLCOL TO POINT HOLD1 AT BOL
              HOLD1
         LDA
              BUFSTR
         STA
         LDA
               LMARGN
         STA BUFSTR+1
         RTS
; DELTIM: TIME TO DELETE A LINE IF IT IS EMPTY AND AN EXTENSION
DELTIA: LDA LOGCOL ; IF LOGCOL<>LMARGN
        CMP LMARGN
                          THEN DONT MOVE UP ONE
         BNE
               DELTIG
                            ;LINE BEFORE TESTING DELTIM
         DEC
               ROWCRS
DELTIG: JSR DOLCOL
DELTIM: LDA LOGCOL
                      ;TEST FOR EXTENSION
         CMP LMARGN
         BEQ DELTI3 ;NO

JSR CONVRT

LDA RMARGN ;SET UP COUNT
         SEC
         SBC
               LMARGN
         TAY
DELTI1: LDA (ADRESS),Y
         BNE
             DELTI3
                       ;FOUND A NON-0 SD OUIT AND RETURN
        DEY
               DELTI1
        \mathtt{BPL}
                        ; DELETE A LINE AND RETURN
DELTI2: JMP DELLIB
DELTI3: RTS
; TSTCTL: SEARCH CNTRLS TABLE TO SEE IF ATACHR IS A CNTL CHAR
TSTCTL: LDX #45
                        ; PREPARE TO SEARCH TABLE
TSTCT1: LDA CNTRLS,X
         CMP
              ATACHR
         BEO
               TSTCT2
         DEX
         DEX
         DEX
         BPL TSTCT1
TSTCT2: RTS
```

```
; PUSH ROWCRS, COLCRS AND COLCRS+1
PHACRS: LDX #2
PHACR1: LDA ROWCRS,X
         STA TMPROW, X
         DEX
         BPL
               PHACR1
         RTS
; PULL COLCRS+1, COLCRS AND ROWCRS
PLACRS: LDX #2
PLACR1: LDA TMPROW, X
         STA ROWCRS, X
         DEX
         BPL PLACR1
         RTS
; SWAP: IF MIXED MODE, SWAP TEXT CURSORS WITH REGULAR CURSORS
SWAP: JSR SWAPA ; THIS ENTRY POINT DOESRETUR1
        JMP RETURI
SWAPA: LDA BOTSCR
         CMP
               #24
         BEO
                SWAP3
         LDX
                #11
SWAP1: LDA ROWCRS,X
        PHA
         LDA TXTROW, X
STA ROWCRS, X
         PLA
         STA
               TXTROW,X
         DEX
         BPL SWAP1
LDA SWPFLG
EOR #$FF
STA SWPFLG
SWAP3: RTS
; CLICK: MAKE CLICK THROUGH KEYBOARD SPEAKER
CLICK: LDX #$7F
CLICK1: STX CONSOL
         STX WSYNC
         DEX
         BPL CLICK1
         RTS
; COLCR: PUTS EITHER 0 OR LMARQN INTO COLCRS BASED ON MODE AND SWPFLG
COLCR: LDA #0
         LDX SWPFLG
         BNE COLCR1
```

```
BNE
                    COLCR2
COLCR1: LDA LMARGN
COLCR2: STA COLCRS
           RTS
;
; PUTMSC: PUT SAVMSC INTO ADRESS
PUTMSC: LDA SAVMSC ;SETUP ADDRESS
           STA ADRESS
           LDA
                   SAVMSC+1
                   ADRESS+1
           STA
           RTS
;
           .PAGE
; DRAW -- DRAW A LINE FROM OLDROW, OLDCOL TO NEWROW, NEWCOL
; (THE AL MILLER METHOD FROM BASKETBALL)
DRAW: LDX #0
                ICCOMZ ;TEST COMMAND: $11=DRAW $12=FILL
           LDA
           CMP
                   #$11
           BEQ DRAWA
CMP #$12 ;TEST FILL
BEQ DRAWB ;YES
LDY #NVALID ;NO, SO RETURN INVALID COMMAND
           RTS
DRAWB: INX
DRAWA: STX FILFLG
           LDA ROWCRS
                                   ; PUT CURSOR INTO NEWROW, NEWCOL
           STA NEWROW
LDA COLCRS
STA NEWCOL
LDA COLCRS+1
           STA
                  NEWCOL+1
           LDA
                   #1
           STA ROWINC STA COLINC
                                   ;SET UP INITIAL DIRECTIONS
           SEC
                 NEWROW ; DETERMINE DELTA ROW
OLDROW
DELTAR
DRAW1 ; DO DIRECTION AND ABSOLUTE VALUE
#$FF ; BORROW WAS ATTEMPTED
ROWINC ; SET DIRECTION DOWN
           LDA
           SBC
           STA
           BCS
           LDA
           STA
           LDA
                  DELTAR
#$FF
           EOR
                                 ;DELTAR = |DELTAR|
           CLC
           ADC #1
STA DELTAR
DRAW1: SEC
                  NEWCOL ; NOW DELTA COLUMN OLDCOL
DELTAC
NEWCOL+1 ; TWO-BYTE QUANTITY
OLDCOL+1
           LDA
                                   ; NOW DELTA COLUMN
           SBC
           STA
           LDA
           SBC
           STA
                   DELTAC+1
           BCS
                   DRAW2
                              ; DIRECTION AND ABSOLUTE VALUE
```

LDX DINDEX

```
LDA
               #$FF
                         ; BORROW WAS ATTEMPTED
               COLINC
         STA
                           ;SET DIRECTION = LEFT
        LDA
               DELTAC
        EOR
               #$FF
                          ;DELTAC = |DELTAC|
              DELTAC
        STA
        LDA
              DELTAC+1
        EOR
               #$FF
               DELTAC+1
        STA
        INC
              DELTAC
                           ; ADD ONE FOR TWOS COMPLEMENT
        BNE
               DRAW2
             DELTAC+1
        INC
DRAW2: LDX #2
                         ; ZERO RAM FOR DRAW LOOP
              #0
        LDY
        STY
               COLAC+1
DRAW3A: TYA
            ROWAC,X
        STA
              OLDROW,X
        LDA
        STA
               ROWCRS,X
        DEX
        BPL
              DRAW3A
        LDA
               DELTAC
                            ; FIND LARGER ONE (ROW OR COL)
        STA
                            (PREPARE COUNTR AND ENDPT)
;
               COUNTR
       STA
               ENDPT
                             ;MAKE X 0
        TNX
        TAY
        LDA
              DELTAC+1
        STA
               COUNTR+1
        STA
              ENDPT+1
        BNE
              DRAW3
                         ; AUTOMATICALLY LARGER IF MSD>0
               DELTAC
        LDA
        CMP
              DELTAR
                            ;LOW COL >LOW ROW?
        BCS
              DRAW3
                        ;YES
              DELTAR
        LDA
        LDX
               #2
        TAY
DRAW3: TYA
                          ; PUT IN INITIAL CONDITIONS
              COUNTR
        STA
        STA
               ENDPT
        PHA
                              ;SAVE AC
              ENDPT+1 ; PUT LSB OF HIGH BYTE
        LDA
        LSR
                          ; INTO CARRY
               A
        PLA
                              ; RESTORE AC
                          ; ROR THE 9 BIT ACUMULATOR
        ROR
                A
        STA
               ROWAC,X
DRAW4A: LDA COUNTR
                  ;TEST ZERO
        ORA COUNTR+1
                            ; IF COUNTER IS ZERO, LEAVE DRAW
        BNE
               DRAWI1
        JMP
               DRAW10
DRAWI1: CLC
                          ; ADD ROW TO ROWAC (PLOT LOOP)
            ROWAC
        LDA
        ADC
              DELTAR
        STA
               ROWAC
        BCC
               DRAW5
        INC
               ROWAC+1
DRAW5: LDA ROWAC+1 ; COMPARE ROW TO ENDPOINT
               ENDPT+1
        CMP
                         ; IF HIGH BYTE OF ROW IS .LT. HIGH
        BCC
                         ;BYTE OF ENDPT, BLT TO COLUMN
               DRAW6
        BNE
               DRAW5A
        LDA
               ROWAC
```

```
CMP
               ENDPT
                         ;LOW BYTE
         BCC
                DRAW6
                          ;ALSO TILT
DRAW5A: CLC
                          GE SO MOVE POINT
              ROWCRS
         LDA
         ADC
               ROWINC
         STA
               ROWCRS
                             ; AND SUBTRACT ENDPT FROM ROWAC
         LDX
               #0
         JSR
               SUBEND
DRAW6: CLC
                          ; DO SAME FOR COLUMN (DOUBLE BYTE ADD)
        LDA COLAC
ADC DELTAC
                          ; ADD
         STA
               COLAC
               COLAC+1
         LDA
         ADC
               DELTAC+1
         STA
               COLAC+1
               ENDPT+1 ; COMPARE HIGH BYTE
         CMP
              DRAW8
         BCC
         BNE
               DRAW6A
         LDA
                COLAC
                          ; COMPARE LOW BYTE
         CMP
               ENDPT
         BCC
               DRAW8
                        ;+ OR - ?
DRAW6A: BIT COLINC
         \mathsf{BPL}
              DRAW6B
                            ; DO DOUBLE BYTE DECREMENT
         DEC
               COLCRS
         LDA
               COLCRS
         CMP
               #$FF
         BNE
               DRAW7
              DRAM
COLCRS+1
         LDA
         BEO
                        ;DON'T DEC IF ZERO
                COLCRS+1
         DEC
         \mathtt{BPL}
               DRAW7
                        ; (UNCONDITIONAL)
DRAW6B: INC COLCRS
                         ; DO DOUBLE BYTE INCREMENT
         BNE
                DRAW7
         INC
               COLCRS+1
                          ; AND SUBTRACT ENDPT FROM COLAC
DRAW7: LDX #2
         JSR
                SUBEND
DRAW8: JSR RANGE
                            ; PLOT POINT
         JSR OUTPLT
         LDA
               FILFLG
                             ;TEST RIGHT FILL
         BEQ
               DRAW9
         JSR
               PHACRS
         LDA
                ATACHR
         STA
               HOLD4
                    ;SAVE ROW IN CASE OF CR
DRAW8A: LDA ROWCRS
         PHA
         JSR
                INCRSA
                             ; POSITION CURSOR ONE PAST DOT
         PLA
                               ; RESTORE ROWCRS
                ROWCRS
         STA
DRAW8C: JSR RANGE
         JSR
                             GET DATA
               GETPLT
                             ;STOP IF NON-ZERO DATA IS ENCOUNTERED
         BNE
               DRAW8B
         LDA
               FILDAT
                             ;FILL DATA
         STA
                ATACHR
         JSR
                OUTPLT
                            ;DRAW IT
         JMP
               DRAW8A
                             ;LOOP
DRAW8B: LDA HOLD4
         STA
               ATACHR
         JSR
               PLACRS
DRAW9: SEC
                           ; DO DOUBLE BYTE SUBTRACT
```

```
LDA
                COUNTR
         SBC
                 #1
         STA
                 COUNTR
                COUNTR+1
         LDA
         SBC
                 #0
         STA
                COUNTR+1
         BMI
                 DRAW10
         JMP
                DRAW4A
DRAW10: JMP RETUR1
          .PAGE
;
; TABLES
; MEMORY ALLOCATION
ALOCAT: .BYTE 24,16,10,10,16,28,52,100,196,196,196,196
;
; NUMBER OF DISPLAY LIST ENTRIES
NUMDLE: .BYTE 23,23,11,23,47,47,95,95,97,97,97
MXDMDE: .BYTE 19,19,9,19,39,39,79,79,65,65,65,65; (EXT OF NUMDLE)
;
; ANTIC CODE FROM INTERNAL MODE CONVERSION TABLE
   INTERNAL
                       ANTIC CODE
                                                DESCRIPTION
;
                                                40X2X8 CHARACTERS
;
         0
                            2
;
        1
                            6
                                                20X5X8
                            7
         2
                                                20X5X16
         3
                            8
                                                40X4X8 GRAPHICS
                                                        11 11
                           9
        4
                                                80X2X4
                                                80X4X4
                                                         11 11
        5
                           Α
                                                160X2X2 ""
;
        4
                           В
        7
                                                160X4X2 ""
                           D
                                                320X2X1 ""
        8
                           F
        9
                           SAME AS 8 BUT GTIA 'LUM' MODE
        10
                         SAME AS 8 BUT GTIA 'COL/LUM REGISTER' MODE
                         SAME AS 8 BUT GTIA 'COLOR' MODE
        11
ANCONV: .BYTE 2,6,7,8,9,$A,$B,$D,$F,$F,$F,$F;ZEROS FOR RANGE TEST IN
;
; PAGE TABLE TELLS WHICH DISPLAY LISTS ARE IN DANGER OF
; CROSSING A 256 BYTE PAGE BOUNDARY
PAGETB: .BYTE 0,0,0,0,0,0,1,1,1,1,1
```

```
; THIS IS THE NUMBER OF LEFT SHIFTS NEEDED TO MULTIPLY
; COLCRS BY 10,20, OR 40. (ROWCRS*10)/(2**DHLINE)
DHLINE: .BYTE 2,1,1,0,0,1,1,2,2,2,2,2
; COLUMN: NUMBER OF COLUMNS
COLUMN: .BYTE 40,20,20,40,80,80,160,160,64,80,80,80; MODE 8 IS SPECIAL
; NOROWS: NUMBER OF ROWS
NOROWS: .BYTE 24,24,12,24,48,48,96,96,192,192,192,192
; DIV2TB: HOW MANY RIGHT SHIFTS FOR HCRSR FOR PARTIAL BYTE MODES
DIV2TB: .BYTE 0,0,0,2,3,2,3,2,3,1,1,1
; DMASKT: DISPLAY MASK TABLE
DMASKT: .BYTE $00,$FF,$F0,$0F
          .BYTE $C0,$30,$0C,$03
; MASKTB: BIT MASK. (ALSO PART OF DMASKTB DO NOT SEPARATE)
MASKTB: .BYTE $80,$40,$20,$10,$08,$04,$02,$01
COLRTB: .BYTE $28,$CA,$94,$46,$00
; CNTRLS: CONTROL CODES AND THEIR DISPLACEMENTS INTO THE
         CONTROL CHARACTER PROCESSORS
CNTRLS: .BYTE
              $1B
          .WORD ESCAPE
          .BYTE $1C
```

```
.BYTE $1D
          .WORD CRSRDN
          .BYTE $1E
          .WORD CRSRLF
          .BYTE $1F
          .WORD CRSRRT
          .BYTE $7D
          .WORD CLRSCR
          .BYTE $7E
          .WORD BS
          .BYTE $7F
          .WORD TAB
          .BYTE $9B
          .WORD DOCRWS
          .BYTE $9C
          .WORD DELLIN
          .BYTE $9D
          .WORD INSLIN
          .BYTE $9E
          .WORD CLRTAB
          .BYTE $9F
          .WORD SETTAB
          .BYTE $FD
          .WORD BELL
          .BYTE $FE
          .WORD DELCHR
          .BYTE $FF
          .WORD INSCHR
; ATAINT: ATASCI TO INTERNAL TABLE
ATAINT: .BYTE $40,$00,$20,$60
; INTATA: INTERNAL TO ATASCI TABLE
INTATA: .BYTE $20,$40,$00,$60
; ATASCI: ATASCII CONVERSION TABLE
ATASCI: .BYTE
               $6C,$6A,$3B,$80,$80,$6B,$2B,$2A ;LOWER CASE
          .BYTE $6F,$80,$70,$75,$9B,$69,$2D,$3D
          .BYTE $76,$80,$63,$80,$80,$62,$78,$7A
          .BYTE $34,$80,$33,$36,$1B,$35,$32,$31
          .BYTE $2C,$20,$2E,$6E,$80,$6D,$2F,$81
          .BYTE $72,$80,$65,$79,$7F,$74,$77,$71
```

.WORD CRSRUP

```
.BYTE $66,$68,$64,$80,$82,$67,$73,$61
          .BYTE $4C,$4A,$3A,$80,$80,$4B,$5C,$5E ;UPPER CASE
          .BYTE $4F,$80,$50,$55,$9B,$49,$5F,$7C
          .BYTE $56,$80,$43,$80,$80,$42,$58,$5A
          .BYTE $24,$80,$23,$26,$1B,$25,$22,$21
          .BYTE $5B,$20,$5D,$4E,$80,$4D,$3F,$81
          .BYTE $52,$80,$45,$59,$9F,$54,$57,$51
          .BYTE $28,$80,$29,$27,$9C,$40,$7D,$9D
          .BYTE $46,$48,$44,$80,$83,$47,$53,$41
          .BYTE $0C,$0A,$7B,$80,$80,$0B,$1E,$1F ; CONTROL
          .BYTE $0F,$80,$10,$15,$9B,$09,$1C,$1D
          .BYTE $16,$80,$03,$80,$80,$02,$18,$1A
          .BYTE $80,$80,$85,$80,$1B,$80,$FD,$80
          .BYTE $00,$20,$60,$0E,$80,$0D,$80,$81
          .BYTE $12,$80,$05,$19,$9E,$14,$17,$11
          .BYTE $80,$80,$80,$80,$FE,$80,$7D,$FF
          .BYTE $06,$08,$04,$80,$84,$07,$13,$01
PIRQ5: LDA
            KBCODE
                                ;TEST AGAINST LAST KEY PRESSED
          CMP
                  CH1
          BNE
                  PIRQ3
                             ; IF NOT, GO PROCESS KEY
          LDA
                  KEYDEL
                                ; IF KEY DELAY BYTE > 0
                  PIRQ4
                             ; IGNORE KEY AS BOUNCE
          BNE
PIRQ3: LDA KBCODE
                            ; RESTORE AC
```

.BYTE \$39,\$80,\$30,\$37,\$7E,\$38,\$3C,\$3E

```
#CNTL1
                             ;TEST CONTROL 1 (SSFLAG)
               PIRQ1
SSFLAG
         BNE
         LDA
         EOR
               #$FF
              SSFLAG
PIRQ4
         STA
                         ; (UNCONDITIONAL) MAKE ^1 INVISIBLE
         BCS
PIRQ1: STA CH
               CH1
         STA
         LDA
                #3
         STA
                KEYDEL
                           ; INITIALIZE KEY DELAY FOR DEBOUNCE
                #0
                              ;CLEAR COLOR SHIFT BYTE
         LDA
         STA
                ATRACT
PIRQ4: LDA #$30
         STA SRTIMR
PIRO2: PLA
         RTI
ï
         .BYTE $FF,$FF,$FF,$FF,$FF
CRNTPC =*
         *=$14
KBDSPR: .BYTE $FFF8-CRNTPC ; GDISPLC IS TOO LONG
         .END
LIST X
; THIS IS THE MODIFIED SEPTEMBER ATARI 400/800 COMPUTER OPERATING
; SYSTEM LISTING. MODIFIED TO ASSEMBLE ON THE MICROTEC CROSS
; ASSEMBLER.
; THIS VERSION IS THE ONE WHICH WAS BURNED INTO ROM.
; THERE IS A RESIDUAL PIECE OF CODE WHICH IS FOR LNBUG. THIS
; IS AT LOCATION $9000 WHICH IS NOT IN ROM.
; THIS IS THE REVISION B EPROM VERSION
         .PAGE
;
       COLLEEN OPERATING SYSTEM EQUATE FILE
       NTSC/PAL ASSEMBLY FLAG
PALFLG = 0
                     ;0 = NTSC 1 = PAL
       MODULE ORIGIN TABLE
CHRORG =
                $E000
                           ; CHARACTER SET
VECTBL =
                           ; VECTOR TABLE
                $E400
VCTABL =
                           ; RAM VECTOR INITIAL VALUE TABLE
                $E480
CIOORG =
                $E4A6
                           ;CENTRAL I/O HANDLER
INTORG =
                          ; INTERRUPT HANDLER
               $E6D5
SIOORG =
              $E944
                           ;SERIAL I/O DRIVER
                           ; DISK HANDLER
DSKORG =
                $EDEA
PRNORG =
                           ; PRINTER HANDLER
               $EE78
CASORG =
              $EF41
                          ; CASSETTE HANDLER
                         ;CASSELLE HANDLE
;MONITOR/POWER UP MODULE
;KEYBOARD/DISPLAY HANDLER
MONORG =
               $F0E3
KBDORG =
              $F3E4
```

CMP

```
VECTOR TABLE
; HANDLER ENTRY POINTS ARE CALLED OUT IN THE FOLLOWING VECTOR
;TABLE. THESE ARE THE ADDRESSES MINUS ONE.
; EXAMPLE FOR EDITOR
    E400
                OPEN
          2
                     CLOSE
            4
                     GET
                     PUT
            6
                    STATUS
SPECIAL
JUMP TO POWER ON INITIALIZATION ROUTINE
NOT USED
           8
           A
          С
          F
                                    ;EDITOR
EDITRV =
                       $E400
SCRENV =
                                      ;TELEVISION SCREEN
                       $E410
KEYBDV =
                     $E420
                                      ;KEYBOARD
PRINTV =
                     $E430
                                       ; PRINTER
CASETV =
                       $E440
                                       ; CASSETTE
          JUMP VECTOR TABLE
;THE FOLLOWING IS A TABLE OF JUMP INSTRUCTIONS
;TO VARIOUS ENTRY POINTS IN THE OPERATING SYSTEM.
                    $E450 ;DISK INITIALIZATION
$E453 ;DISK INTERFACE
$E456 ;CENTRAL INPUT OUTPUT ROUTINE
$E459 ;SERIAL INPUT OUTPUT ROUTINE
$E45C ;SET SYSTEM TIMERS ROUTINE
$E45F ;SYSTEM VERTICAL BLANK CALCULATIONS
$E462 ;EXIT VERTICAL BLANK CALCULATIONS
$E465 ;SERIAL, INPUT OUTPUT INITIALIZATION
$E468 ;SEND ENABLE ROUTINE
$E468 ;INTERRUPT HANDLER INITIALIZATION
$E46E ;CENTRAL INPUT OUTPUT INITIALIZATION
$E471 ;BLACKBOARD MODE
$E474 ;WARM START ENTRY POINT
$E477 ;COLD START ENTRY POINT
$E477 ;CASSETTE READ BLOCK ENTRY POINT VECTOR
$E470 ;CASSETTE OPEN FOR INPUT VECTOR
DISKIW =
DSKINV =
CIOV =
SIOV
          =
SETVBV =
SYSVBV =
XITVBV =
SIOINV =
SENDEV =
INTINV =
CIOINV =
BLKBDV =
WARMSV =
COLDSV =
RBLOKV =
CSOPIV =
; VCTABL = $E480
; OPERATING SYSTEM EQUATES
; COMMAND CODES FOR IOCB
OPEN = 3
                                        ;OPEN FOR INPUT/OUTPUT
GETREC =
                                        GET RECORD (TEXT)
                     7
                                        GET CHARACTER(S)
GETCHR =
                    9
$B
                                        ; PUT RECORD (TEXT)
PUTREC =
PUTCHR =
                                       ; PUT CHARACTER(S)
                       $C
                                        ; CLOSE DEVICE
CLOSE =
```

```
STATIS =
               $D
                         ;STATUS REQUEST
SPECIL =
               $E
                          ;BEGINNING OF SPECIAL ENTRY COMMANDS
; SPECIAL ENTRY COMMANDS
DRAWLN =
                             ;DRAW LINE
               $11
              $12
                            ; DRAW LINE WITH RIGHT FILL
FILLIN =
RENAME =
              $20
                            ; RENAME DISK FILE
DELETE =
              $21
                            ;DELETE DISK FILE
FORMAT =
              $22
                            ; FORMAT
LOCKFL =
              $23
                            ;LOCK FILE TO READ ONLY
                            ;UNLOCK LOCKED FILE
UNLOCK =
              $24
              $25
                            ; POINT SECTOR
POINT =
      =
NOTE
              $26
                            ; NOTE SECTOR
IOCFRE =
                            ; IOCB "FREE"
               $FF
; AUX1 EQUATES
; () INDICATES WHICH DEVICES USE BIT
APPEND = $1 ;OPEN FOR WRITE APPEND (D), OR SCREEN READ (
                         ;OPEN FOR DIRECTORY ACCESS (D)
DIRECT =
              $2
              $4
                         ;OPEN FOR INPUT (ALL DEVICES)
OPNIN =
            $8 ;OPEN FOR OUTPUT (ALL DEVICES)
OPNIN+OPNOT ;OPEN FOR INPUT AND OUTPUT (ALL DEVICES)
OPNOT =
OPNINO =
              $10 ;OPEN FOR MIXED MODE (E,S)
MXDMOD =
              $20
INSCLR =
                            ;OPEN WITHOUT CLEARING SCREEN (E,S)
; DEVICE NAMES
           ' E
SCREDT =
                          ;SCREEN EDITOR (R/W)
           ' K
KBD =
                     ;KEYBOARD (R ONLY)
           'S
D1SPLY =
                      ;SCREEN DISPLAY (R/W)
PRINTR =
              ' P
                          ; PRINTER (W ONLY)
CASSET =
              ' C
                          ; CASSETTE
MODEM =
              ' M
                          ; MODEM
DISK =
              'D
                          ;DISK (R/W)
; SYSTEM EOL (CARRIAGE RETURN)
CR =
           $9В
;
       OPERATING SYSTEM STATUS CODES
               $01
                             ;SUCCESSFUL OPERATION
SUCCES =
BRKABT =
               $80
                            ; BREAK KEY ABORT
PRVOPN =
               $81
                             ; IOCB ALREADY OPEN
NONDEV =
               $82
                             ;NON-EXISTANT DEVICE
WRONLY =
               $83
                            ; IOCB OPENED FOR WRITE ONLY
NVALID =
               $84
                             ; INVALID COMMAND
NOTOPN =
               $85
                            ;DEVICE OR FILE NOT OPEN
BADIOC =
               $86
                            ; INVALID IOCB NUMBER
RDONLY =
               $87
                             ; IOCB OPENED FOR READ ONLY
EOFERR =
               $88
                            ; END OF FILE
TRNRCD =
              $89
                            ;TRUNCATED RECORD
TIMOUT =
               $8A
                             ; PERIPHERAL DEVICE TIME OUT
DNACK =
               $8B
                            ; DEVICE DOES NOT ACKNOWLEDGE COMMAND
FRMERR =
               $8C
                            ;SERIAL BUS FRAMING ERROR
CRSROR =
               $8D
                            ; CURSOR OVERRANCE
                            ;SERIAL BUS DATA OVERRUN
              $8E
OVRRUN =
CHKERR =
               $8F
                            ; SERIAL BUS CHECKSUM ERROR
```

```
DERROR =
                $90
                              ; PERIPHERAL DEVICE ERROR (OPERATION NOT COMP
BADMOD =
                $91
                              ; BAD SCREEN MODE NUMBER
FNCNOT =
                $92
                              ; FUNCTION NOT IMPLEMENTED IN HANDLER
SCRMEM =
               $93
                              ; INSUFICIENT MEMORY FOR SCREEN MODE
       PAGE ZERO RAM ASSIGNMENTS
         *=$0000
LINZBS: .RES 2
                           ;LINBUG RAM (WILL BE REPLACED BY MONITOR RAM
; THESE LOCATIONS ARE NOT CLEARED
CASINI: .RES 2
                            ; CASSETTE INIT LOCATION
RAMLO: .RES
               2
                           ; RAM POINTER FOR MEMORY TEST
TRAMSZ: .RES 1
TSTDAT: .RES 1
                            ;TEMPORARY REGISTER FOR RAM SIZE
                           ; RAM TEST DATA REGISTER
; CLEARED ON COLOSTART ONLY
WARMST: .RES 1
                           ;WARM START FLAG
BOOT?: .RES
               1
                           ;SUCCESSFUL BOOT FLAG
DOSVEC: .RES 2
DOSINI: .RES 2
                            ; DISK SOFTWARE START VECTOR
              2
                           ; DISK SOFTWARE INIT ADDRESS
APPMHI: .RES 2
                            ; APPLICATIONS MEMORY HI LIMIT
; CLEARED ON COLD OR WARM START
INTZBS =*
                              ; INTERRUPT HANDLER
              1
POKMSK: .RES
                           ;SYSTEM MASK FOR POKEY IRQ ENABLE
BRKKEY: .RES
                           BREAK KEY FLAG
RTCLOK: .RES 3
                            ; REAL TIME CLOCK (IN 16 MSEC UNITS)
BUFADR: .RES
              2
                            ; INDIRECT BUFFER ADDRESS REGISTER
ICCOMT: .RES 1
                            ; COMMAND FOR VECTOR
             2
DSKFMS: .RES
                            ; DISK FILE MANAGER POINTER
DSKUTL: .RES
              2
                            ;DISK UTILITIES POINTER
             1
1
PTIMOT: .RES
                            ; PRINTER TIME OUT REGISTER
PBPNT: .RES
                            ; PRINT BUFFER POINTER
PBUFSZ: .RES
               1
                            ; PRINT BUFFER SIZE
PTEMP: .RES 1
                            ;TEMPORARY REGISTER
ZIOCB =*
                              ; ZERO PAGE I/O CONTROL BLOCK
              16
IOCBSZ =
                          ; NUMBER OF BYTES PER IOCB
              8*IOCBSZ ;LENGTH OF THE IOCB AREA
MAXIOC =
IOCBAS =*
                            ; HANDLER INDEX NUMBER (FF = IOCB FREE)
ICHIDZ: .RES
              1
ICDNOZ: .RES
               1
                            ; DEVICE NUMBER (DRIVE NUMBER)
ICCOMZ: .RES
               1
                            ; COMMAND CODE
ICSTAZ: .RES
              1
                           ;STATUS OF LAST IOCB ACTION
ICBALZ: .RES
               1
                           BUFFER ADDRESS LOW BYTE
ICBAHZ: .RES
               1
               1
ICPTLZ: .RES
                           ; PUT BYTE ROUTINE ADDRESS - 1
ICPTHZ: .RES
               1
ICBLLZ: .RES 1
                            ;BUFFER LENGTH LOW BYTE
```

```
ICBLHZ: .RES
              1
ICAX1Z: .RES
                1
                           ; AUXILIARY INFORMATION FIRST BYTE
ICAX2Z: .RES
              1
                           ;TWO SPARE BYTES (CIO LOCAL USE)
ICSPRZ: .RES
              4
              ICSPRZ+2 ; IOCB NUMBER X 16
ICSPRZ+3 ; CHARACTER BYTE FO
ICIDNO =
                          ; CHARACTER BYTE FOR CURRENT OPERATION
CIOCHR =
                            ; INTERNAL STATUS STORAGE
STATUS: .RES
              1
CHKSUM: .RES
                           ; CHECKSUM (SINGLE BYTE SUM WITH CARRY)
BUFRLO: .RES
               1
                           ; POINTER TO DATA BUFFER (LO BYTE)
BUFRHI: .RES
              1
                           ; POINTER TO DATA BUFFER (HI BYTE)
BFENLO: .RES
                           ; NEXT BYTE PAST END OF THE DATA BUFFER (LO B
BFENHI: .RES
               1
                           ; NEXT BYTE PAST END OF THE DATA BUFFER (HI B
CRETRY: .RES
                           ; NUMBER OF COMMAND FRAME RETRIES
              1
DRETRY: .RES
               1
                           ;NUMBER OF DEVICE RETRIES
BUFRFL: .RES
              1
                            ;DATA BUFFER FULL FLAG
RECVDN: .RES
              1
                           ; RECEIVE DONE FLAG
XMTDON: .RES
               1
                           ;TRANSMISSION DONE FLAG
CHKSNT: .RES
              1
                           ; CHECKSUM SENT FLAG
NOCKSM: .RES 1
                           ; NO CHECKSUM FOLLOWS DATA FLAG
BPTR: .RES 1
FTYPE: .RES
               1
FEOF: .RES
               1
FREO: .RES
SOUNDR: .RES
               1
                            ; NOISY I/O FLAG. (ZERO IS QUIET)
              1
CRITIC: .RES
                            ; DEFINES CRITICAL SECTION (CRITICAL IF NON-Z
             7
FMSZPG: .RES
                            ;DISK FILE MANAGER SYSTEM ZERO PAGE
;
CKEY: .RES 1
                            ;FLAG SET WHEN GAME START PRESSED
               1
                            ; CASSETTE BOOT FLAG
CASSBT: .RES
DSTAT: .RES
               1
                           ;DISPLAY STATUS
                           ;ATRACT FLAG
ATRACT: .RES
             1
DRKMSK: .RES
              1
                           ;DARK ATRACT MASK
COLRSH: .RES 1
                            ;ATRACT COLOR SHIFTER (EOR'ED WITH PLAYFIELD
LEDGE =
              2
                           ;LMARGN'S VALUE AT COLD START
REDGE =
               39
                          ; RMARGN'S VALUE AT COLD START
TMPCHR: .RES
               1
HOLD1: .RES
               1
LMARGN: .RES
              1
                            ;LEFT MARGIN (SET TO 1 AT POWER ON)
RMARGN: .RES
              1
                           ; RIGHT MARGIN (SET TO 38 AT POWER ON)
ROWCRS: .RES
               1
                            ; CURSOR COUNTERS
COLCRS: .RES
               2
DINDEX: .RES
SAVMSC: .RES
OLDROW: .RES
               1
OLDCOL: .RES
               2
OLDCHR: .RES
               1
                            ; DATA UNDER CURSOR
              2
OLDADR: .RES
NEWROW: .RES
               1
                           ; POINT DRAW GOES TO
NEWCOL: .RES
              2
LOGCOL: .RES
              1
                            ; POINTS AT COLUMN IN LOGICAL LINE
ADRESS: .RES
              2
MLTTMP: .RES
```

```
OPNTMP =
              MLTTMP
                        FIRST BYTE IS USED IN OPEN AS TEMP
SAVADR: .RES
RAMTOP: .RES
              1
                           ; RAM SIZE DEFINED BY POWER ON LOGIC
BUFCNT: .RES
              1
                           ;BUFFER COUNT
BUFSTR: .RES
              2
                           ;EDITOR GETCH POINTER
              1
BITMSK: .RES
                          ;BIT MASK
SHFAMT: .RES
              1
ROWAC: .RES
              2
COLAC: .RES
ENDPT: .RES
DELTAR: .RES
               1
DELTAC: .RES
ROWINC: .RES
              1
COLINC: .RES
              1
SWPFLG: .RES
                           ;NON-0 IF TXT AND REGULAR RAM IS SWAPPED
              1
HOLDCH: .RES
              1
                           ;CH IS MOVED HERE IN KGETCH BEFORE CNTL & SH
              1
INSDAT: .RES
COUNTR: .RES
              2
       30 - FF ARE RESERVED FOR USER APPLICATIONS
       NOTE: SEE FLOATING POINT SUBROUTINE AREA FOR ZERO PAGE CELLS
       PAGE 1 - STACK
       PAGE TWO RAM ASSIGNMENTS
        *=$0200
INTABS =*
                            ; INTERRUPT RAM
                          ;DISPLAY LIST NMI VECTOR
VDSLST: .RES
              2
VPRCED: .RES
              2
                          ; PROCEED LINE IRQ VECTOR
VINTER: .RES
             2
                          ; INTERRUPT LINE IRQ VECTOR
VBREAK: .RES
              2
                          ;SOFTWARE BREAK (00) INSTRUCTION IRQ VECTOR
              2
VKEYBD: .RES
                          ; POKEY KEYBOARD IRQ VECTOR
VSERIN: .RES
              2
                          ; POKEY SERIAL INPUT READY IRQ
VSEROR: .RES
              2
                          ; POKEY SERIAL OUTPUT READY IRQ
VSEROC: .RES
              2
                          ; POKEY SERIAL OUTPUT COMPLETE IRQ
VTIMR1: .RES
              2
                           ; POKEY TIMER 1 IRO
VTIMR2: .RES
              2
                           ; POKEY TIMER 2 IRQ
              2
VTIMR4: .RES
                           ; POKEY TIMER 4 IRQ
VIMIRU: .RES
              2
                          ; IMMEDIATE IRQ VECTOR
              2
CDTMV1: .RES
                           ; COUNT DOWN TIMER 1
CDTMV2: .RES
              2
                          COUNT DOWN TIMER 2
              2
CDTMV3: .RES
                          ; COUNT DOWN TIMER 3
CDTMV4: .RES
              2
                           COUNT DOWN TIMER 4
              2
CDTMV5: .RES
                          COUNT DOWN TIMER S
VVBLKI: .RES
              2
                          ; IMMEDIATE VERTICAL BLANK NMI VECTOR
VVBLKD: .RES 2
                           ; DEFERRED VERTICAL BLANK NMI VECTOR
```

```
CDTMA1: .RES
              2
                           ; COUNT DOWN TIMER 1 JSR ADDRESS
CDTMA2: .RES
                           ; COUNT DOWN TIMER 2 JSR ADDRESS
                           COUNT DOWN TIMER 3 FLAG
CDTMF3: .RES
              1
SRTIMR: .RES
              1
                           ;SOFTWARE REPEAT TIMER
CDTMF4: .RES
              1
                           COUNT DOWN TIMER 4 FLAG
INTEMP: .RES
              1
                           ; IAN'S TEMP (RENAMED FROM T1 BY POPULAR DEMA
              1
CDTMF5: .RES
                           ; COUNT DOWN TIMER FLAG 5
SDMCTL: .RES
              1
                          ;SAVE DMACTL REGISTER
SDLSTL: .RES
              1
                          ; SAVE DISPLAY LIST LOW BYTE
SDLSTH: .RES
              1
                          ;SAVE DISPLAY LIST HI BYTE
SSKCTL: .RES
              1
                           ;SKCTL REGISTER RAM
        .RES 1
LPENH: .RES 1
                           ;LIGHT PEN HORIZONTAL VALUE
LPENV: .RES
              1
                           ;LIGHT PEN VERTICAL VALUE
BRKKY: .RES 2
                           ; BREAK KEY VECTOR
        .RES 2
                           ;SPARE
CDEVIC: .RES 1
                           ; COMMAND FRAME BUFFER - DEVICE
CCOMND: .RES
              1
                          ; COMMAND
CAUX1: .RES
               1
                           COMMAND AUX BYTE 1
CAUX2: .RES
              1
                           ; COMMANDAUX BYTE 2
  NOTE: MAY NOT BE THE LAST WORD ON A PAGE
TEMP: .RES 1
                           ;TEMPORARY RAM CELL
; NOTE: MAY NOT BE THE LAST WORD ON A PAGE
ERRFLG: .RES 1
                           ; ERROR FLAG - ANY DEVICE ERROR EXCEPT TIME
;
DFLAGS: .RES 1
                           ;DISK FLAGS FROM SECTOR ONE
              1
DBSECT: .RES
                           ; NUMBER OF DISK BOOT SECTORS
             2
BOOTAD: .RES
                           ; ADDRESS WHERE DISK BOOT LOADERWILL BE PUT
COLDST: .RES 1
                           ; COLDSTART FLAG (1=IN MIDDLE OF COLDSTART)
        .RES 1
                           ;SPARE
DSKTIM: .RES 1
                           ;DISK TIME OUT REGISTER
LINBUF: .RES 40
                        ; CHAR LINE BUFFER
GPRIOR: .RES 1
                          ;GLOBAL PRIORITY CELL
PADDL0: .RES
             1
                           ; POTENTIOMETER 0 RAM CELL
PADDL1: .RES
PADDL2: .RES
PAODL3: .RES
PADDL4: .RES
              1
PADDL5: .RES
PADDL6: .RES
              1
PADDL7: .RES
              1
              1
STICK0: .RES
                           ;JOYSTICK 0 RAM CELL
              1
STICK1: .RES
STICK2: .RES
              1
STICK3: .RES
              1
PTRIGO: .RES
                           ; PADDLE TRIGGER 0
              1
PTRIG1: .RES
              1
PTRIG2: .RES
PTRIG3: .RES
PTRIG4: .RES
              1
PTRIG5: .RES
```

```
PTRIG6: .RES
PTRIG7: .RES
STRIGO: .RES
              1
                           ;JOYSTICK TRIGGER 0
STRIG1: .RES
              1
STRIG2: .RES
              1
STRIG3: .RES
CSTAT: .RES
               1
WMODE: .RES
BLIM: .RES
              1
IMASK: .RES
               1
JVECK: .RES
        .RES 2
                          ;SPARE
TXTROW: .RES
             1
                           ;TEXT ROWCRS
TXTCOL: .RES
              2
                          ;TEXT COLCRS
TINDEX: .RES
              1
                          ;TEXT INDEX
TXTMSC: .RES
              2
                           ; FOOLS CONVRT INTO NEW MSC
TXTOLD: .RES
             6
                          ;OLDROW & OLDCOL FOR TEXT (AND THEN SOME)
              1
TMPX1: .RES
HOLD3: .RES
              1
SUBTMP: .RES
              1
HOLD2: .RES
              1
DMASK: .RES
              1
TMPLBT: .RES
ESCFLG: .RES
              1
                           ; ESCAPE FLAG
TABMAP: .RES
              15
LOGMAP: .RES
                           ;LOGICAL LINE START BIT MAP
INVFLG: .RES
              1
                           ; INVERSE VIDEO FLAG (TOGGLED BY ATARI KEY)
              1
FILFLG: .RES
                           ; RIGHT FILL FLAG FOR DRAW
TMPROW: .RES
              1
TMPCOL: .RES
              2
                           ;SET IF SCROLL OCCURS
SCRFLG: .RES
              1
HOLD4: .RES
              1
                           ;TEMP CELL USED IN DRAW ONLY
HOLD5: .RES
              1
                           ;DITTO
SHFLOK: .RES
              1
BOTSCR: .RES
              1
                          ;BOTTOM OF SCREEN : 24 NORM 4 SPLIT
;
PCOLR0: .RES 1
                          ;P0 COLOR
              1
PCOLR1: .RES
                          ;P1 COLOR
PCOLR2: .RES
              1
                          ;P2 COLOR
PCOLP3: .RES
              1
                          ;P3 COLOR
COLOR0: .RES
              1
                          ; COLOR 0
COLOR1: .RES
              1
CQLOR2: .RES
              1
               1
COLOR3: .RES
COLOR4: .RES
              1
        .RES 23 ;SPARE
GLBABS =*
                             ;GLOBAL VARIABLES
```

```
.RES 4
                            ;SPARE
RAMSIZ: .RES 1
                            ; RAM SIZE (HI BYTE ONLY)
              2
                            ;TOP OF AVAILABLE USER MEMORY
MEMTOP: .RES
MEMLO: .RES
              2
                            ;BOTTOM OF AVAILABLE USER MEMORY
        .RES 1
                            ;SPARE
DVSTAT: .RES 4
                            ;STATUS BUFFER
CBAUDL: .RES
                            ; CASSETTE BAUD RATE LOW BYTE
CBAUDH: .RES 1
;
CRSINH: .RES
                            ; CURSOR INHIBIT (00 = CURSOR ON)
KEYDEL: .RES
               1
                            ;KEY DELAY
CH1: .RES
               1
CHACT: .RES 1
                            ; CHACTL REGISTER RAM
CHBAS: .RES
                            ; CHBAS REGISTER RAM
        .RES 5
                            ;SPARE BYTES
CHAR: .RES
ATACHR: .RES
                            ;ATASCII CHARACTER
CH: .RES 1
                       ;GLOBAL VARIABLE FOR KEYBOARD
FILDAT: RES 1
DSPFLG: RES 1
SSFLAG: RES 1
                           ; RIGHT FILL DATA (DRAW)
                            ;DISPLAY FLAG : DISPLAY CNTLS IF NON-ZERO
                            ;START/STOP FLAG FOR PAGING (CNTL 1). CLEARE
      PAGE THREE RAM ASSIGNMENTS
DCB =*
                  ;DEVICE CONTROL BLOCK
DDEVIC: .RES
              1
                            ; PERIPHERAL UNIT 1 BUS ID. NUMBER
DUNIT: .RES
              1
                            ;UNIT NUMBER
DCOMND: .RES
               1
                            ;BUS COMMAND
DSTATS: .RES
              1
                            ; COMMAND TYPE/STATUS RETURN
               1
DBUFLO: .RES
                            ;DATA BUFFER POINTER LOW BYTE
              1
DBUFHI: .RES
DTIMLO: .RES
              1
                            ; DEVICE TIME OUT IN 1 SECOND UNITS
              1
DUNUSE: .RES
                            ;UNUSED BYTE
DBYTLO: .RES
              1
                            ; NUMBER OF BYTES TO BE TRANSFERRED LOW BYTE
DBYTHI: .RES
               1
               1
DAUX1: .RES
                            ; COMMAND AUXILIARY BYTE 1
DAUX2: .RES
              1
TIMER1: .RES 2
                            ; INITIAL TIMER VALUE
ADDCOR: .RES
               1
                            ; ADDITION CORRECTION
CASFLG: .RES
               1
                            ; CASSETTE MODE WHEN SET
TIMER2: .RES
              2
                            ; FINAL TIMER VALUE. THESE TWO TIMER VALUES
; ARE USED TO COMPUTE INTERVAL FOR BAUD RATE
                            ;TEMPORARY STORAGE REGISTER
TEMP1: .RES 2
TEMP2: .RES
              1
                            ;TEMPORARY STORAGE REGISTER
              1
TEMP3: .RES
                           ;TEMPORARY STORAGE REGISTER
SAVIO: .RES
              1
                           ;SAVE SERIAL IN DATA PORT
TIMFLG: .RES 1
                            ;TIME OUT FLAG FOR BAUD RATE CORRECTION
```

```
1
TSTAT: .RES
              1
                           ;TEMPORARY STATUS HOLDER
HATABS: .RES 38
                          ; HANDLER ADDRESS TABLE
MAXDEV =
              *-HATABS-5 ; MAXIMUM HANDLER ADDRESS INDEX
    NOTE: THE ENTIRE IOCB DEFINITIONS HAVE BEEN MODIFIED
IOCB: .ORG
                           ;I/O CONTROL BLOCKS
ICHID: .RES
                           ; HANDLER INDEX NUMBER (FF = IOCB FREE)
ICDNO: .RES
               1
                           ; DEVICE NUMBER (DRIVE NUMBER)
ICCOM: .RES
                           ; COMMAND CODE
              1
              1
                           ;STATUS OF LAST IOCB ACTION
ICSTA: .RES
ICBAL: .RES
              1
                           ;BUFFER ADDRESS LOW BYTE
ICBAH: .RES
              1
ICPTL: .RES
              1
                           ; PUT BYTE ROUTINE ADDRESS - 1
ICPTH: .RES
              1
                           ;BUFFER LENGTH LOW BYTE
ICBLL: .RES
              1
ICBLH: .RES
              1
ICAX1: .RES
               1
                           ; AUXILIARY INFORMATION FIRST BYTE
ICAX2: .RES
              1
ICSPR: .RES
                           ; FOUR SPARE BYTES
        .RES MAXIOC-IOCBSZ
PRNBUF: .RES 40
                        ; PRINTER BUFFER
        .RES 21
                        ;SPARE BYTES
       PAGE FOUR RAM ASSIGNMENTS
CASBUF: .RES 131
                            ;CASSETTE BUFFER
; USER AREA STARTS HERE AND GOES TO END OF PAGE FIVE
USAREA: .RES 128
                             ;SPARE
       PAGE FIVE RAM ASSIGNMENTS
       PAGE FIVE IS RESERVED AS A USER WORK SPACE
        NOTE: SEE FLOATING POINT SUBROUTINE AREA FOR PAGE FIVE CELLS
        PAGE SIX RAM ASSIGNMENTS
; PAGE SIX IS RESERVED AS A USER'S USER WORK SPACE
```

;SIO STACK POINTER SAVE CELL

STACKP: .RES

```
FLOATING POINT SUBROUTINES
FPREC =
                           ;FLOATING PT PRECISION (# OF BYTES)
               6
; IF CARRY USED THEN CARRY CLEAR \Rightarrow NO ERROR, CARR
    = $D800 ;ASCII->FLOATING POINT(FP)
                                     INBUFF+CIX -> FR0, CIX, CARRY
               $D8E6
FASC
                          ;FP -> ASCII FRO -> LBUFF (INBUFF)
IFP =
        $D9AA
                     ;INTEGER -> FP
                                     0-\$FFFF (LSB,MSB) IN FR0,FR0+1->FR0
FPI =
          $D9D2
                     ;FP -> INTEGER FR0 -> FR0,FR0+1, CARRY
               $DA60
                         ;FR0 <- FR0 - FR1 ,CARRY
FSUB
               $DA66
                          ;FR0 <- FR0 + FR1 ,CARRY
FADD
FMUL
       =
               $DADB
                         ;FR0 <- FR0 * FR1 ,CARRY
                        ;FR0 <- FR0 / FR1 ,CARRY
;FLOATING LOAD REG0 FR0 <- (X,Y)
; " " FR0 <- (FLFTR)
FDIV
       =
              $DB28
FLDOR =
              $DD89
              $DD80
FLDOP =
                         ; "
                                   " REG1 FR1 \leftarrow (X,Y)
FLD1R =
              $DD98
                         ; "
             $DD9C
$DDA7
                                  " " FR1 <- (FLPTR)
FLD1P =
FSTOR =
                         ;FLOATING STORE REGO (X,Y) <- FRO
             $DDAB
                         ; " " (FLPTR) <- FR0
FSTOP =
                         ;FR1 <- FR0
FMOVE =
              $DDB6
PLYEVL =
             $DD40
                        ;FR0 \leftarrow P(Z) = SUM(I=N TO 0) (A(I)*Z**I) CAR
                                INPUT: (X,Y) = A(N), A(N-1)...A(0) \rightarrow PLYARG
                                         ACC
                                              = # OF COEFFICIENTS = DEGREE+1
                                         FR0
                                             = Z
          $DDC0 ;FR0 <- E**FR0 = EXP10(FR0 * LOG10(E)) CARRY
EXP =
           $DDCC
                      ;FR0 <- 10**FR0 CARRY
EXP10 =
           $DECD
                     ;FR0 \leftarrow LN(FR0) = LOG10(FR0)/LOG10(E) CARRY
LOG10 = $DED1 ;FR0 <- LOG10 (FR0) CARRY
; THE FOLLOWING ARE IN BASIC CARTRIDGE:
SIN = $BDB1 ;FR0 <- SIN(FR0) DEGFLG=0 =>RADS, 6=>DEG. CA
                     ;FR0 <- COS(FR0) CARRY
           $BD73
COS
ATAN
           $BE43
                         ;FR0 <- ATAN(FR0) CARRY
SQR = $BEB1 ;FR0 <- SQUAREROOT(FR0) CARRY
; FLOATING POINT ROUTINES ZERO PAGE (NEEDED ONLY IF V.P. ROUTINES ARE CA
       *=$D4
                         ;FP REG0
              FPREC
FR0:
       .RES
FRE:
       .RES FPREC
FR1:
       .RES FPREC
                         ;FP REG1
       .RES FPREC
FR2:
FRX:
       .RES 1
                          ;FP SPARE
EEXP:
      .RES
                          ; VALUE OF E
NSIGN: .RES
              1
                          ;SIGN OF #
ESIGN: .RES
              1
                          ;SIGN OF EXPONENT
FCHRFLG: .RES
              1
                          ;1ST CHAR FLAG
DIORT: .RES
              1
                          ;# OF DIGITS RIGHT OF DECIMAL
       .RES 1
                          ;CURRENT INPUT INDEX
CIX:
INBUFF: .RES
              2
                           ; POINTS TO USER'S LINE INPUT BUFFER
ZTEMPI: .RES
              2
ZIEMP4: .RES
ZTEMP3: .RES
DEGFLG
RADFLG: .RES 1
                          ;0=RADIANS, 6=DEGREES
RADON =
              0
                          ; INDICATES RADIANS
               6
                           ; INDICATES DEGREES
DEGON =
```

```
FLPTR: .RES
FPTR2: .RES
; FLOATING PT ROUTINES' NON-ZERO PAGE RAM
; (NEEDED ONLY IF F.P. ROUTINES CALLED)
         *=$57E
                            ;LBUFF PREFIX 1
LBPR1:
      .RES
               1
                            ;LBUFF PREFIX 2
LBPR2: .RES
               1
LBUFF: .RES
               128
                             ;LINE BUFFER
PLYARG =
               LBUFF+$60 ; POLYNOMIAL ARGUMENTS
FPSCR =
               PLYARG+FPREC
FPSCR1 =
               FPSCR+FPREC
FSCR =
              FPSCR
FSCR1
       =
               FPSCR1
               *-1
                             ;END OF LBUFF
LBFEND =
       COLLEEN MNEMONICS
POKEY
                $D200
                          ; VBLANK ACTION:
                                                     DESCRIPTION:
               POKEY+0
                           ;POT0-->PADDL0
                                                   0-227 IN RAM CELL
POT0
                                                   0-227 IN RAM CELL
POT1
               POKEY+1
                           ;POT1-->PADDL1
        =
POT2
               POKEY+2
                           ;POT2-->PADDL2
                                                  0-227 IN RAM CELL
                                                   0-227 IN RAM CELL
POT3
               POKEY+3
                            ;POT3-->PADDL3
       =
POT4
               POKEY+4
                           ;POT4-->PADDL4
                                                   0-227 IN RAM CELL
       =
POT5
                           ;POT5-->PADDL5
                                                  0-227 IN RAM CELL
               POKEY+5
       =
                            ;POT6-->PADDL6
                                                  0-227 IN RAM CELL
POT6
               POKEY+6
                                                  0-227 IN RAM CELL
POT7
       =
               POKEY+7
                           ;POT7-->PADDL7
ALLPOT =
               POKEY+8
                           ;???
KBCODE =
               POKEY+9
RANDOM =
               POKEY+10
POTGO =
                         ;STROBED
               POKEY+11
SERIN
               POKEY+13
IROST
       =
               POKEY+14
SKSTAT =
               POKEY+15
AUDF1
               POKEY+0
       =
AUDC1
       =
               POKEY+1
AUDF2 =
               POKEY+2
AUDC2
       =
               POKEY+3
AUDF3 =
               POKEY+4
AUDC3
       =
               POKEY+5
AUDF4
     =
               POKEY+6
AUDC4 =
               POKEY+7
AUDCTL =
                                                AUDCTL<--[SIO]
               POKEY+8
                            ; NONE
STIMER =
               POKEY+9
SKRES =
               POKEY+10
                           ; NONE
                                                  SKRES<--[SIO]
SEROUT =
               POKEY+13
                           ; NONE
                                                 SEROUT<--[SIO]
                          ; POKMSK-->IRQEN (AFFECTED BY OPEN S: OR E:)
IRQEN =
               POKEY+14
SKCTL =
               POKEY+15 ;SSKCTL-->SKCTL
                                            SSKCTL<--[SIO]
;
               $D000
CTIA
       =
                          ; VBLANK ACTION:
                                                 DESCRIPTION:
HPOSP0 =
               CTIA+0
HPOSP1 =
               CTIA+1
```

; POINTS TO USER'S FLOATING PT NUMBER

2

```
HPOSP2 =
                 CTIA+2
HPOSP3
                 CTIA+3
HPOSMO =
                 CTIA+4
                 CTIA+5
HPOSM1
       =
HPOSM2
                 CTIA+6
HPOSM3 =
                 CTIA+7
SIZEPO =
                 CTIA+8
SIZEP1
                 CTIA+9
       =
SIZEP2 =
                 CTIA+10
                 CTIA+11
SIZEP3 =
                 CTIA+12
SIZEM
GRAFP0 =
                 CTIA+13
GRAFP1 =
                 CTIA+14
                 CTIA+15
GRAFP2 =
GRAFP3 =
                 CTIA+16
GRAFM
                 CTIA+17
                               ;PCOLR0-->COLPM0
COLPM0
                 CTIA+18
                                                       WITH ATTRACT MODE
       =
COLPM1 =
                 CTIA+19
                               ; PCOLR1-->COLPM1
                                                       WITH ATTRACT MODE
COLPM2
                 CTIA+20
                               ; PCOLR2-->COLPM2
                                                       WITH ATTRACT MODE
COLPM3
                 CTIA+21
                               ; PCOLR3-->COLPM3
                                                       WITH ATTRACT MODE
       =
COLPF0 =
                 CTIA+22
                               ;COLOR0-->COLPF0
                                                       WITH ATTRACT MODE
                 CTIA+23
                                                       WITH ATTRACT MODE
COLPF1
                               ;COLOR1-->COLPF1
COLPF2 =
                 CTIA+24
                               ;COLOR2-->COLPF2
                                                       WITH ATTRACT MODE
                 CTIA+25
                               ;COLOR3-->COLPF3
                                                       WITH ATTRACT MODE
COLPF3 =
COLBK
        =
                 CTIA+26
                               ;COLOR4-->COLBK
                                                        WITH ATTRACT MODE
PRIOR
                 CTIA+27
                               ;(ON OPEN S: OR E:) GPRIOR-->PRIOR
VDELAY =
                 CTIA+28
GRACTL =
                 CTIA+29
HITCLR =
                 CTIA+30
                                                       TURN OFF SPEAKER
CONSOL =
                               ;$08-->CONSOL
                 CTIA+31
M0PF
                 CTIA+0
                 CTIA+1
M1PF
M2PF
                 CTIA+2
                 CTIA+3
M3PF
         =
POPF
                 CTIA+4
P1PF
                 CTIA+5
P2PF
                 CTIA+8
         =
P3PF
                 CTIA+7
         =
MOPL
                 CTIA+8
M1PL
         =
                 CTIA+9
M2PL
                 CTIA+10
         =
M3PL
                 CTIA+11
POPL
         =
                 CTIA+12
P1PL
                 CTIA+13
         =
P2PL
                 CTIA+14
P3PL
                 CTIA+15
                               ;TRIG0-->STRIG0
TRIG0
        =
                 CTIA+16
TRTG1
                 CTIA+17
                               ;TRIG1-->STRIG1
        =
                               ;TRIG2-->STRIG2
TRIG2
        =
                 CTIA+18
TRIG3
                 CTIA+19
                               ;TRIG3-->STRIG3
        =
;
ANTIC
                 $D400
                              ; VBLANK ACTION
                                                   DESCRIPTION
        =
DMACTL =
                 ANTIC+0
                               ;DMACTL<--SDMCTL
                                                       ON OPEN S: OR E:
                 ANTIC+1
                                                        ON OPEN S: OR E:
CHACTL
                               ;CHACTL<--CHACT
       =
DLISTL =
                               ;DLISTL<--SDLSTL
                                                       ON OPEN S: OR E:
                 ANTTC+2
DLISTH
                 ANTIC+3
                               ;DLISTH<--SDLSTH
                                                       ON OPEN S: OR E:
NSCROL
       =
                 ANTIC+4
VSCROL =
                 ANTIC+5
                 ANTIC+7
PMBASE
```

```
CHBASE =
             ANTIC+9 ; CHBASE<--CHBAS
                                               ON OPEN S: OR E:
             ANTIC+10
WSYNC =
             ANTIC+11
ANTIC+12
ANTIC+13
VCOUNT =
PENH =
      =
PENV
             ANTIC+14 ; NMIEN<--40 POWER ON AND [SETVBV]
NMIEN =
             ANTIC+15
NMIRES =
                         ;STROBED
           ANTIC+15
NMIST =
NMIST = ANTIC+15

PIA = $D300 ; VBLANK ACTION DESCRIPTION
          PIA+0 ;PORTA-->STICK0,1 X-Y CONTROLLERS
PIA+1 ;PORTB-->STICK2,3 X-Y CONTROLLERS
PORTA =
                                            X-Y CONTROLLERS
PORTB =
PACTL =
             PIA+2
                        ; NONE
                                               PACTL<--3C [INIT]
             PIA+3
PBCTL =
                         ; NONE
                                               PBCTL<--3C [INIT]
; .PAGE
        .PAGE
        LIST S
         .TITLE 'CENTRAL INPUT/OUTPUT (CIO) 2-7-79'
               UPDATED BY AL MILLER 3-9-79
                        ;ASCII ZERO
              ' 0
ASCZER =
COLON =
              $3A
                         ;ASCII COLON
EOL = $9B
                        ;END OF RECORD
        .PAGE
; CIO JUMP VECTOR FOR USERS
        *=CIOV
        JMP CIO ;GO TO CIO
; CIO INIT JUMP VECTOR FOR POWER UP
        *=CIOINV
        JMP CIOINT ;GO TO INIT
;
; ERROR ROUTINE ADDRESS EQUATE
; ERRTNH =ERRTN/256 "MOVED TO LINE 788"
; ERRTNL =-ERRTNH*256+ERRTN "MOVED TO LINE 789"
;
        *=CIOORG
; CIO INITIALIZATION (CALLED BY MONITOR AT POWER UP)
CIOINT: LDX #0
CIOI1: LDA #IOCFRE ;SET ALL IOCB'S TO FREE
        STA ICHID,X ;BY SETTING HANDLER ID'S=$FF
        LDA
               #ERRTNL
        STA
              ICPTL,X ; POINT PUT TO ERROR ROUTINE
              #ERRTNH
        LDA
        STA ICPTH, X
        TXA
        CLC
              #IOCBSZ ;BUMP INDEX BY SIZE
        ADC
        TAX
              #MAXIOC
        CMP
                         ; DONE?
        BCC
              CIOI1
                         ; NO
        RTS
                             ;YES, RETURN
; ERROR ROUTINE FOR ILLEGAL PUT
```

```
ERRTNH = ERRTN/256
ERRTNL = (-ERRTNH) * 256 + ERRTN
        LDY #NOTOPN ; IOCB NOT OPEN
         RTS
         .PAGE
; CIO LOCAL RAM (USES SPARE BYTES IN ZERO PAGE IOCB)
ENTVEC = ICSPRZ
; CIO MAIN ROUTINE
; CIO INTERFACES BETWEEN USER AND INPUT/OUTPUT DE
CIO: STA CIOCHR ;SAVE POSSIBLE OUTPUT CHARACTER
        STX ICIDNO ;SAVE IOCB NUMBER * N
; CHECK FOR LEGAL IOCB
         TXA
         AND
               #$F ; IS IOCB MULTIPLE OF 16?
CIERR1 ;NO, ERROR
#MAXIOC ;IS INDEX TOO LARGE?
         BNE
         CPX
         BCC IOC1
                            ;NO
; INVALID IOCB NUMBER -- RETURN ERROR
CIERR1: LDY #BADIOC ; ERROR CODE
        JMP CIRTN1 ; RETURN
; MOVE USER IOCB TO ZERO PAGE
IOC1: LDY #0
IOC1A: LDA IOCB,X ;USER IOCB
         STA IOCBAS, Y ; TO ZERO PAGE
         INX
         INY
               #12
         CPY
                           ;12 BYTES
         BCC
               IOC1A
; COMPUTE CIO INTERNAL VECTOR FOR COMMAND
         LDY #NVALID ;ASSUME INVALID CODE
               ICCOMZ ; CUMINIAND LEGAL?
         LDA
                           ; COMMAND CODE TO INDEX
         CMP
         BCC
         TAY
; MOVE COMMAND TO ZERO BASE FOR INDEX
         CPY #SPECIL ; IS COMMAND SPECIAL?
BCC IOC2 ;NO
LDY #SPECIL ;YES, SET SPECIAL OFFSET INDEX
IOC2: STY ICCOMT ;SAVE COMMAND FOR VECTOR
        LDA COMTAB-3,Y ;GET VECTOR OFFSET FROM TABLE
         BEQ
               CIOPEN ;GO IF OPEN COMMAND
               #2
         CMP
                              ; IS IT CLOSE?
               CICLOS
         BEO
                             ;YES
         CMP #8 ;IS
BCS CISTSP ;YES
                              ; IS IT STATUS OR SPECIAL?
         CMP
               #4
                              ; IS IT READ?
         BEQ CIREAD
JMP CIWRIT
                           ;YES
;ELSE, MUST BE WRITE
         .PAGE
```

ERRTN =*-1

```
; OPEN COMMAND
; FIND DEVICE HANDLER IN HANDLER ADDRESS TABLE
CIOPEN: LDA ICHIDZ ;GET HANDLER ID
         CMP #IOCFRE ; IS THIS IOCB CLOSED?
                IOC6
         BEQ
                            ;YES
; ERROR -- IOCB ALREADY OPEN
CIERR3: LDY #PRVOPN ; ERROR CODE
CIERR4: JMP CIRTN1
                        ; RETURN
; GO FIND DEVICE
                        ;CALL DEVICE SEARCH
IOC6: JSR DEVSRC
        BCS CIERR4
                            ;GO IF DEVICE NOT FOUND
; DEVICE FOUND, INITIALIZE IOCB FOR OPEN
; COMPUTE HANDLER ENTRY POINT
IOC7: JSR COMENT
        BCS CIERR4 ;GO IF ERROR IN COMPUTE
; GO TO HANDLER FOR INITIALIZATION
         JSR GOHAND ; USE INDIRECT JUMP
; STORE PUT BYTE ADDRESS-1 INTO IOCB
         LDA #PUTCHR ;SIMULATE PUT CHARACTER
         STA ICCOMT

JSR COMENT ; COMPUTE ENTRY POINT

LDA ICSPRZ ; MOVE COMPUTED VALUE

STA ICPTLZ ; TO PUT BYTE ADDRESS

LDA ICSPRZ+1

STA ICPTHZ

JMP CIRTN2 ; RETURN TO USER
         .PAGE
; CLOSE COMMAND
CICLOS: LDY #SUCCES ; ASSUME GOOD CLOSE
         STY ICSTAZ
                              ; COMPUTE HANDLER ENTRY POINT
         JSR
                COMENT
         BCS
                CICLO2
                              ;GO IF ERROR IN COMPUTE
         JSR GOHAND
                              ;GO TO HANDLER TO CLOSE DEVICE
CICLO2: LDA #IOCFRE ;GET IOCB "FREE" VALUE
         STA ICHIDZ
LDA #ERRTNH
                              ;SET HANDLER ID
         STA
                ICPTHZ
                              ;SET PUT BYTE TO POINT TO ERROR
         LDA
                #ERRTNL
         STA
         STA ICPTLZ
JMP CIRTN2 ;RETURN
; STATUS AND SPECIAL REQUESTS
; DO IMPLIED OPEN IF NECESSARY AND GO TO DEVICE
CISTSP: LDA ICHIDZ
                      ; IS THERE A HANDLER ID?
         CMP #IOCFRE
         BNE
                CIST1 ;YES
; IOCB IS FREE, DO IMPLIED OPEN
         JSR DEVSRC
                         ;FIND DEVICE IN TABLE
```

```
; COMPUTE AND GO TO ENTRY POINT IN HANDLER
CIST1: JSR COMENT ; COMPUTER HANDLER ENTRY VECTOR JSR GOHAND ; GO TO HANDLER
; RESTORE HANDLER INDEX (DO IMPLIED CLOSE)
           LDA ICHID,X ;GET ORIGINAL HANDLER ID
STA ICHIDZ ;RESTORE ZERO PAGE
JMP CIRTN2 ;RETURN
           .PAGE
; READ -- DO GET COMMANDS
CIREAD: LDA ICCOMZ ;GET COMMAND BYTE
AND ICAX1Z ;IS THIS READ LEGAL?
BNE RCI1A ;YES
; ILLEGAL READ -- IOCB OPENED FOR WRITE ONLY
          LDY #WRONLY ; ERROR CODE
RCI1B: JMP CIRTN1 ;RETURN
; COMPUTE AND CHECK ENTRY POINT
RCI1A: JSR COMENT ; COMPUTE ENTRY POINT
         BCS RCI1B ;GO IF ERROR IN COMPUTE
; GET RECORD OR CHARACTERS
           LDA ICBLLZ
           ORA
                   ICBLLZ+1 ; IS BUFFER LENGTH ZERO?
           BNE RCI3
JSR GOHAND
STA CIOCHR
JMP CIRTN2
                                ;NO
; LOOP TO FILL BUFFER OR END RECORD
RCI3: JSR GOHAND ;GO TO HANDLER TO GET BYTE
           STA CIOCHR ;SAVE BYTE

BMI RCI4 ;END TRANSFER IF ERROR

LDY #0

STA (ICBALZ),Y ;PUT BYTE IN USER BUFFER
           JSR INCBFP ;INCREMENT BUFFER POINTER
LDA ICCOMZ ;GET COMMAND CODE
AND #2 ;IS IT GET RECORD?
                                ; IS IT GET RECORD?
           BNE RCI1 ;NO
; CHECK FOR EOL ON TEXT RECORDS
           LDA CIOCHR ;GET BYTE
CMP #EOL ;IS IT AN EOL?
BNE RCI1 ;NO
                 .. 20L
RCI1
DECBFL
RCI4
                                ;YES, DECREMENT BUFFER LENGTH;END TRANSFER
           JSR
           JMP
; CHECK BUFFER FULL
          SR DECBFL ; DECREMENT BUFFER LENGTH
BNE RCI3 ; CONTINUE IF NON ZERO
RCI1: JSR DECBFL
           .PAGE
; BUFFER FULL. RECORD NOT ENDED
; DISCARD BYTES UNTIL END OF RECORD
```

BCS CIERR4 ;GO IF ERROR IN COMPUTE

```
RCI2: LDA ICCOMZ ;GET COMMAND BYTE
         AND #2
                             ; IS IT GET CHARACTER?
                RCI4
         BNE
                            ;YES, END TRANSFER
; LOOP TO WAIT FOR EOL
RCI6: JSR GOHAND ;GET BYTE FROM HANDLER
                          ;SAVL
;GO IF ERROR
          STA CIOCHR
                             ;SAVE CHARACTER
                RCI4
          BMI
; TEXT RECORD. WAIT FOR EOL
         LDA CIOCHR ;GET GOT BYTE
CMP #EOL ;IS IT EOL?
BNE RCI6 ;NO, CONTINUE
; END OF RECORD. BUFFER FULL -- SEND TRUNCATED RECORD MESSAGE
RCI1I: LDA #TRNRCD ; ERROR CODE
        STA ICSTAZ ;STORE IN 10GB
; TRANSFER DONE
RCI4: JSR SUBBFL ;SET FINAL BUFFER LENGTH

JMP CIRTN2 ;RETURN
         .PAGE
; WRITE -- DO PUT COMMANDS
CIWRIT: LDA ICCOMZ ;GET COMMAND BYTE
        AND ICAX1Z ; IS THIS WRITE LEGAL?
BNE WCIIA ; YES
; ILLEGAL WRITE -- DEVICE OPENED FOR READ ONLY
        LDY #RDONLY ; ERROR CODE
WCI1B: JMP CIRTN1
                          ; RETURN
; COMPUTE AND CHECK ENTRY POINT
WCIIA: JSR COMENT ; COMPUTE HANDLER ENTRY POINT
        BCS WCI1B
                          GO IF ERROR IN COMPUTE
; PUT RECORD OR CHARACTERS
         LDA ICBLLZ
               ICBLLZ+1 ;IS BUFFER LENGTH ZERO? WCI3 ;NO
          ORA
         BNE
         LDA CIOCHR
INC ICBLLZ
BNE WCI4
                             GET CHARACTER; SET SUFFER LENOTHI
                            ;THEN JUST TRANSFER ONE BYTE
; LOOP TO TRANSFER BYTES FROM BUFFER TO HANDLER
WCI3: LDY #0
         LDA (ICBALZ),Y ;GET BYTE FROM BUFFER STA CIOCHR ;SAVE
       JSR GOHAND ;GO PUT BYTE

BMI WCI5 ;END IF ERROR

JSR INCBFP ;INCREMENT I
                              ; INCREMENT BUFFER POINTER
; CHECK FOR TEXT RECORD
         LDA ICCOMZ
                            GET COMMAND BYTE; IS IT PUT RECORD?
          AND #2 ;
BNE WCI1 ;NO
; TEXT RECORD -- CHECK FOR EOL TRANSFER
          LDA CIOCHR
                               GET LAST CHARACTER
```

```
#EOL
                         ;IS IT AN EOL?
;NO
         CMP
         BNE
         JSR DECBFL
JMP WCI5
                          ; DECREMENT BUFFER LENGTH
                        ;END TRANSFER
; CHECK FOR BUFFER EMPTY
                        ; DECREMENT BUFFER LENGTH
WCI1: JSR DECBFL
        BNE WCI3
                        ; CONTINUE IF NON ZERO
        .PAGE
; BUFFER EMPTY, RECORD NOT FILLED
; CHECK TYPE OF TRANSFER
WCI2: LDA ICCOMZ
                        GET COMMAND CODE
        AND #2
                          ; IS IT PUT CHARACTER?
        BNE WCI5 ;YES, END TRANSFER
; PUT RECORD (TEXT), BUFFER , EMPTY, SEND EOL
        LDA #EOL
         JSR
              GOHAND
                           ;GO TO HANDLER
; END PUT TRANSFER
WCI5: JSR SUBBFL ;SET ACTUAL PUT BUFFER LENGTH JMP CIRTN2 ;RETURN
        .PAGE
; CIO RETURNS
; RETURNS WITH Y=STATUS
CIRTN1: STY ICSTAZ ;SAVE STATUS
; RETURNS WITH STATUS STORED IN ICSTAZ
; MOVE IOCB IN ZERO PAGE BACK TO USER AREA
CIRTN2: LDY ICIDNO ;GET IOCB INDEX
        LDA ICBAL,Y
        STA
              ICBALZ
                            ; RESTORE USER BUFFER POINTER
         LDA
              ICBAH,Y
        STA ICBAH
LDX #0
              ICBAHZ
                            ;LOOP COUNT AND INDEX
CIRT3: LDA IOCBAS, X ; ZERO PAGE
        STA IOCB, Y ; TO USER AREA
         INX
         INY
        CPX #12
BCC CIRT3
                            ;12 BYTES
; RESTORE A,X, & Y
         LDA CIOCHR
                           GET LAST CHARACTER
         LDX
               ICIDNO
                            ; IOCB INDEX
              ICSTAZ ;GET STATUS AND SET FLAGS
        LDY
        RTS
                             ; RETURN TO USER
         .PAGE
; CIO SUBROUTINES
; COMENT -- CHECK AND COMPUTE HANDLER ENTRY POINT
COMENT: LDY ICHIDZ ;GET HANDLER INDEX
        CPY #MAXDEV+1 ; IS IT A LEGAL INDEX?
         BCC
              COM1 ;YES
```

```
LDY #NOTOPN ; ERROR CODE
         BCS
               COM2
                          ; RETURN
; USE HANDLER ADDRESS TABLE AND COMMAND TABLE TO GET VECTOR
COM1: LDA HATABS+1,Y ;GET LOW BYTE OF ADDRESS
                             ; AND SAVE IN POINTER
         STA
               ICSPRZ
         LDA
               HATABS+2,Y ;GET HI BYTE OF ADDRESS
         STA
               ICSPRZ+1
         LDY ICCOMT ;GET COMMAND CODE
LDA COMTAB-3,Y ;GET COMMAND OFFSET
         TAY
               (ICSPRZ),Y ;GET LOW BYTE OF VECTOR FROM
         LDA
         TAX
                               ; HANDLER ITSELF AND SAVE
         INY
              (ICSPRZ),Y ;GET HI BYTE OF VECTOR
         LDA
         STA
               ICSPRZ+1
               ICSPRZ
         STX
                            ;SET LO BYTE
        CLC
                               ; SHOW NO ERROR
COM2: RTS
; DECBFL -- DECREMENT BUFFER LENGTH DOUBLE BYTE
; Z FLAG = 0 ON RETURN IF LENGTH = 0 AFTER DECREMENT
DECBFL: DEC ICBLLZ ; DECREMENT LOW BYTE
        LDA ICBLLZ ; CHECK IT
CMP #$FF ; DID IT GO F
BNE DECBF1 ; NO
                          ;DID IT GO BELOW?
               ICBLLZ+1 ; DECREMENT HI BYTE
        DEC
DECBF1: ORA ICBLLZ+1 ;SET Z IF BOTH ARE ZERO
        RTS
; INCBFP -- INCREMENT WORKING BUFFER POINTER
INCBFP: INC ICBALZ ; BUMP LOW BYTE
         BNE INCBF1
                          ;GO IF NOT ZERO
         INC
               ICBALZ+1 ; ELSE, BUMP HI BYTE
INCBF1: RTS
; SUBBFL -- SET BUFFER LENGTH = BUFFER LENGTH - WORKING BYTE COUNT
SUBBFL: LDX ICIDNO
                     GET IOCB INDEX
         SEC
              LDA
         SBC
         STA
               ICBLH,X
ICBLLZ+1
         LDA
                          GET HI BYTE
         SBC
         STA
               ICBLHZ
         RTS
; GOHAND -- GO INDIRECT TO A DEVICE HANDLER
; Y= STATUS ON RETURN, N FLAG=1 IF ERROR ON RETURN
GOHAND: LDY #FNCNOT ; PREPARE NO FUNCTION STATUS FOR HANDLER RTS
         JSR CIJUMP ;USE THE INDIRECT JUMP STY ICSTAZ ;SAVE STATUS
                            ;SAVE STATUS
         CPY
               #0
                             ;AND SET N FLAG
         RTS
```

; ILLEGAL HANDLER INDEX MEANS DEVICE NOT OPEN FOR OPERATION

```
; INDIRECT JUMP TO HANDLER BY PAUL'S METHOD
CIJUMP: TAX
                           ;SAVE A
         LDA ICSPRZ+1 ;GET JUMP ADDRESS HI BYTE
                                ; PUT ON STACK
         PHA
         LDA ICSPRZ
                              GET JUMP ADDRESS LO BYTE
                                ; PUT ON STACK
         PHA
         TXA
                                ; RESTORE A
         LDX ICIDNO
                             GET IOCB INDEX
         RTS
                                ;GO TO HANDLER INDIRECTLY
         .PAGE
; DEVSRC -- DEVICE SEARCH, FIND DEVICE IN HANDLER ADDRESS TABLE
; LOOP TO FIND DEVICE
DEVSRC: LDY #0
               (ICBALZ),Y ;GET DEVICE NAME FROM USER
         LDA
         BEO
                CIERR2
                #MAXDEV
         LDY
                           ; INITIAL COMPARE INDEX
DEVS1: CMP HATABS, Y ; IS THIS THE DEVICE?
         BEO DEVS2 ; YES
         DEY
         DEY
                                 ; ELSE, POINT TO NEXT DEVICE NAME
         DEY
         BPL DEVS1 ; CONTINUE FOR ALL DEVICES
; NO DEVICE FOUND, DECLARE NON-EXISTENT DEVICE ERROR
CIERR2: LDY #NONDEV ; ERROR CODE
         SEC
                               ;SHOW ERROR
         BCS DEVS4
                           ;AND RETURN
; FOUND DEVICE, SET ICHID, ICDNO, AND INIT DEVICE
DEVS2: TYA
         STA ICHIDZ ;SAVE HANDLER INDEX
         SEC
         LDY #1

LDA (ICBALZ),Y ;GET DEVICE NUMBER (DRIVE NUMBER)

SBC #ASCZER ;SUBTRACT ASCII ZERO

:TS NUMBER IN RANGE?
               DEVS3 ;YES
         BCC
        LDA
                              ; NO. DEFAULT TO ONE
DEVS3: STA ICDNOZ ;SAVE DEVICE NUMBER
        CLC
                                ;SHOW NO ERROR
; RETURN
DEVS4: RTS
         .PAGE
; CIO ROM TABLES
; COMMAND TABLE
; MAPS EACH COMMAND TO OFFSET FOR APPROPRIATE VECTOR IN HANDLER
COMTAB: BYTE 0,4,4,4,6,6,6,6,6,2,8,10
LENGTH = *-CIOINT
CRNTP1 =*
        *=$14
```

```
.TITLE 'INTERRUPT HANDLER'
;LIVES ON DK1:INTHV.SRC
                           ;SECOND REPEAT INTERVAL
; THIS IS TO MAKE DOS 2 WORK WHICH USED AN ABSOLUTE ADDRESS
         *=$E912
         JMP SETVBL
         *=SETVBV
         JMP SETVBL
         JMP
                SYSVBL
         JMP
               XITVBL
         *=INTINV
         JMP IHINIT
         *=VCTABL+INTABS-VDSLST
         .WORD SYRTI
                          ; VDSLST
                       ;VPRCED;VINTER
         .WORD SYIRQB
         .WORD SYIRQB
         .WORD SYIRQB
                         ; VBREAK
         .RES
              8
         .WORD SYIRQB
                         ;VTMIR1
         .WORD SYIRQB
                         ;VTIMR2
         .WORD SYIRQB ;VTMIR4
.WORD SYIRO
                          ;VIMIRO
         .WORD 0,0,0,0,0 ;CDTMV1-4
         .WORD SYSVBL ;VVBLKI
.WORD XITVBL ;VVSLKD
         *=$900C
         STA $FFF8
LDA #PNMIH
STA $FFFB
LDA #PNMIL
STA $FFFA
         RTS
         .PAGE
; IRQ HANDLER
; JUMP THRU IMMEDIATE IRQ VECTOR, WHICH ORDINARILY POINTS TO
; SYSTEM IRQ; DETERMINE & CLEAR CAUSE, JUMP THRU SOFTWARE VECTOR.
         *=INTORG
IHINIT: LDA #$40 ;VBL ON BUF DLIST OFF***FOR NOW***
         STA NMIEN ; ENABLE DISPLAY LIST, VERTICAL BLANK
               #$38
         LDA
                           ;LOOK AT DATA DIRECTION REGISTERS IN PIA
         STA
               PACTL
         STA
               PBCTL
```

CIOSPR: BYTE INTORG-CRNTP1 ; GCIOL IS TOO LONG

```
LDA
               #0
                             ; MAKE ALL INPUTS
         STA
               PORTA
         STA
               PORTB
                         ;BACK TO PORTS
         LDA
               #$3C
              #$3C
PACTL
         STA
         STA
               PBCTL
        RTS
     JMP (VIMIRU)
PIRO:
CMPTAB: .BYTE $80
                        ; BREAK KEY
         .BYTE $40
                        ;KEY STROKE
         .BYTE $04
                        ;TIMER 4
         .BYTE $02
                        ;TIMER 2
         .BYTE $01
                         ;TIMER 1
         .BYTE $08
                        ;SERIAL OUT COMPLETE
         .BYTE $10
                        ;SERIAL OUT READY
                      ;SERIAL IN READY
         .BYTE $20
; THIS IS A TABLE OF OFFSETS INTO PAGE 2. THEY POINT TO
ADRTAB: .BYTE BRKKY-INTABS
         .BYTE VKEYBD-INTABS
         .BYTE VTIMR4-INTABS
         .BYTE VTIMR2-INTABS
         .BYTE VTIMR1-INTABS
         .BYTE VSEROC-INTABS
         .BYTE VSEROR-INTABS
         .BYTE VSERIN-INTABS
SYIRQ: PHA
                           ; SAVE ACCUMULATOR
              IROST
        LDA
                         ; CHECK FOR SERIAL IN
         AND
               #$20
         BNE
              SYIRQ2
#$DF
         LDA
                          ; MASK ALL OTHERS
              IRQEN
POKMSK
         STA
         LDA
         STA
              IRQEN
(VSERIN)
         JMP
                           ; PUT X INTO ACC
SYIRQ2: TXA
        PHA
                              ; SAVE K ONTO STACK
         LDX #$6
                             ;START WITH SIX OFFSET
LOOPM: LDA CMPTAB, X ; LOAD MASK
         CPX
              #5
                             ; CHECK TO SEE IF COMPLETE IS SET
         BNE
               LOOPM2
         AND
               POKMSK
                          ; IS THIS INTERUPT ENABLED?
         BEO LL
LOOPM2: BIT IRQST ; IS IT THE INTERUPT?
        BEQ JMPP
LL: DEX
                           ;NO DEC X AND TRY NEXT MASK
        BPL LOOPM
JMP SYIRQ8
                         ; IF NOT NEC 0010 LOOPH
                            ; DONE BUT NO INTERUPT
JMPP: EOR #$FF
                       ; COMPLEMENT MASK
         STA IRQEN ; ENABLE ALL OTHERS
         LDA
               POKMSK
                            ; GET POKE MASK
              IRQEN
         STA
                         ; ENABLE THOSE IN POKE MASK
         LDA
               ADRTAB,X
         TAX
         LDA
              INTABS,X ; GET ADDRESS LOW PART
         STA
               JVECK ; PUT IN VECTOR
         LDA
               INTABS+1,X ; GET ADDRESS HIGH PART
                JVECK+1 ; PUT IN VECTOR HIGH PART
         STA
```

```
; PULL X REGISTER FROM STACK
         TAX
                               ; PUT IT INTO X
         JMP
               (JVECK)
                          ; JUMP TO THE PROPER ROUTINE
BRKKY2: LDA #0
                          ; BREAK KEY ROUTINE
         STA BRKKEY
                            ; SET BREAK KEY FLAG
         STA
               SSFLAG
                            ; START/STOP FLAG
         STA
               CRSINH
                            ; CURSOR INHIBIT
         STA
                            ; TURN OFF ATRACT MODE
               ATRACT
         PLA
         RTI
                              ;EXIT FROM INT
SYIRQ8: PLA
         TAX
         BIT
               PACTL
                         ;PROCEED ***I GUESS***
         BPL
               SYIRQ9
               PORTA
                         CLEAR INT STATUS BIT
         LDA
         JMP (VPRCED)
SYIRQ9: BIT PBCTL ;INTERRUPT ***I GUESS***
         BPL
              SYIROA
         LDA
               PORTB
                         ;CLEAR INT STATUS
         JMP
               (VINTER)
SYIRQA: PLA
         STA JVECK
         PLA
         PHA
              #$10
         AND
                      ;B BIT OF P REGISTER
         BEO
               SYRTI2
         LDA
               JVECK
         PHA
        JMP
              (VBREAK)
SYRTI2: LDA JVECK
        PHA
SYIROB: PLA
SYRTI: RTI
                        ;UNIDENTIFIED INTERRUPT, JUST RETURN
        .PAGE
; NMI HANDLER
; DETERMINE CAUSE AND JUMP THRU VECTOR
PNMI: BIT NMIST
        BPL PNMI1
                        ;SEE IF DISPLAY LIST
         JMP
               (VDSLST)
PNMI1: PHA
        LDA NMIST AND #$20
                         ;SEE IF RESET
         BEQ
               *+5
         JMP
               WARMSV
                            ;DO THRU WARM START JUMP
         TXA
                              ; SAVE REGISTERS
         PHA
         TYA
         PHA
               NMIRES
         STA
                            ; RESET INTERRUPT STATUS
                (VVBLKI) ;JUMP THRU VECTOR
         JMP
         .PAGE
; SYSTEM VBLANK ROUTINE
; INC FRAME COUNTER. PROCESS COUNTDOWN TIMERS. EXIT IF I WAS SET. CLEAR
; SET DLISTL, DLISTH, DMACTL FROM RAM CELLS. DO SOFTWARE REPEAT.
```

PLA

```
SYSVBL: INC RTCLOK+2 ; INC FRAME COUNTER
         BNE SYSVB1
                          ; INCREMENT ATRACT (CAUSES ATRACT WHEN MINUS)
         INC
                ATRACT
         INC
                RTCLOK+1
         BNE
                SYSVB1
         INC
                RTCLOK
SYSVB1: LDA #$FE ;{ATRACT] SET DARK MASK TO NORMAL
         LDX #0
                               ;SET COLRSH TO NORMAL
               ATRACT
         IDY
                              ;TEST ATRACT FOR NEGATIVE
                VBATRA
                              ; WHILE POSITIVE DONT GO INTO ATRACT
         BPL
         STA
                ATRACT
                              ; IN ATRACTI SO STAY BY STA $FE
                RTCLOK+1 ; COLOR SHIFT FOLLOWS RICLOK+1
         LDX
         LDA
                #$F6 ;SET DARK MASK TO DARK
VBATRA: STA DRKMSK
         STX COLRSH
         LDX
                               ; POINT TO TIMER1
                #0
         JSR
                DCTIMR
                              GO DECREMENT TIMER1
         BNE SYSVB2 ;BRANCH IF STILL COUNTING JSR JTIMR1 ;GO JUMP TO ROUTINE
SYSVB2: LDA CRITIC
         BNE XXIT ;GO IF CRITICAL SET
         TSX
                                ;SEE IF I WAS SET
                $104,X
#$04
                              GET STACKED P
         LDA
         AND
                            ;I BIT
         BEQ
                SYSVB3
                            ;BRANCH IF OK
     JMP XITVBL ;I WAS SET, EXIT
XXIT:
SYSVB3: LDA PENV
         STA LPENV
                PENH
         LDA
         STA LPENH
LDA SDLSTH
STA DLISTH
LDA SDLSTL
STA DLISTL
               SDMCTL
DMACTL
GPRIOR
PRIOR
#$08
         LDA
         STA
                           ;GLOBAL PRIOR
         LDA
         STA
                            ;TURN OFF KEYBOARD SPEAKER
         LDX
         STX
                CONSOL
SCOLLP: CLI
                             ; DISABLE INTERUPTS
         LDA PCOLRO, X ; LOAD COLOR REGISTERS FROMRAM
EOR COLRSH ; DO COLOR SHIFT
AND DRKMSK ; AND DARK ATRACT
STA COLPMO, X
         DEX
         BPL
                SCOLLP
         LDA
                CHBAS
               CHBASE
CHACT
         STA
         LDA
         STA
                CHACTL
                               ; POINT TO TIMER 2
         LDX
                #2
               DCTIMR
         JSR
                          ; IF DIDNT GO ZERO
         BNE
                SYSVB4
         JSR JTIMR2
                               ;GO JUMP TO TIMER2 ROUTINE
SYSVB4: LDX #2
                           ; RESTORE X
SYSVBB: INX
```

INX

```
LDA CDTMV1,X
        ORA
               CDTMV1+1,X
        BEQ
               SYSVBA
                        ;DECREMENT AND SET FLAG IF NONZERO
        JSR
              DCTIMR
        STA CDTMF3-4,X
SYSVBA: CPX #8
                ;SEE IF DONE ALL 3
             SYSVBB
        BNE
                           ;LOOP
; CHECK DEBOUNCE COUNTER
        LDA SKSTAT
            #$04 ;KEY DOWN BIT
SYVB6A ;IF KEY DOWN
        AND
        BEO
; KEY UP SO COUNT IT
             KEYDEL
                           KEY DELAY COUNTER
        LDA
                           ; IF COUNTED DOWN ALREADY
        BEQ
               SYVB6A
              KEYDEL
        DEC
                           ; COUNT IT
; CHECK SOFTWARE REPEAT TIMER
SYVB6A: LDA SRTIMR
        BEQ SYSVB7
                           ;DOESN'T COUNT
        LDA SKSTAT
AND #$04
                         ; CHECK KEY DOWN BIT
        BNE SYSVB6
DEC SRTIMR
BNE SYSVB7
                           ;BRANCH IF NO LONGER DOWN
                           ; COUNT FRAME OF KEY DOWN
                           ;BRANCH IF NOT RUN OUT
; TIMER RAN OUT - RESET AND SIMULATE KEYBOARD IRQ
        LDA #SRTIM2 ;TIMER VALUE
              SRTIMR
                         ;SET TIMER
        STA
        LDA
              KBCODE
                           GET THE KEY
             CH
        STA
                            ; PUT INTO CH
; READ GAME CONTROLLERS
SYSVB7: LDY #1
        LDX #3
STLOOP: LDA PORTA, Y
        LSR A
        LSR
               Α
        LSR
               Α
        LSR A STICKO,X ;STORE JOYSTICK
        DEX
        LDA PORTA,Y
        AND
              #$F
        STA
              STICKO,X ;STORE JOYSTICK
        DEX
        DEY
        BPL STLOOP
;
        LDX #3
STRL: LDA TRIGO,X ; MOVE JOYSTICK TRIGGERS
        STA STRIGO,X
        LDA POT0,X
STA PADDL0,X
LDA POT4,X
                        ; MOVE POT VALUES
        STA
              PADDL4,X
        DEX
             STRL
POTGO ;START POTS FOR NEXT TIME
        BPL
        STA
        LDX #6
        LDY
               #3
PTRLP: LDA STICKO, Y ; TRANSFER BITS FROM JOYSTICKS
```

```
LSR
              A
                         ;TO PADDLE TRIGGERS
         LSR
                Α
         LSR
               Α
         STA PTRIG1,X
               #0
         LDA
         ROL
               Α
        STA
              PTRIG0,X
         DEX
        DEX
        DEY
        \mathsf{BPL}
             PTRLP
        JMP
               (VVBLKD)
                         ;GO TO DEFERRED VBLANK ROUTINE
      =
             SYSVB7/256
SV7H
             (-256)*SV7H+SYSVB7
       =
SV7L
SYSVB6: LDA #0
        STA
                           ;ZERO TIMER
              SRTIMR
        BEQ
               SYSVB7
                            ; UNCOND
JTIMR1: JMP (CDTMA1)
JTIMR2: JMP (CDTMA2)
; SUBROUTINE TO DECREMENT A COUNTDOWN TIMER
; ENTRY X=OFFSET FROM TIMER 1
; EXIT A, P=ZERO IF WENT ZERO, FF OTHERWISE
DCTIMR: LDY CDTMV1,X ;LO BYTE
        BNE
               DCTIM1 ; NONZERO, GO DEC IT
               CDTMV1+1,X ;SEE IF BOTH ZERO
        LDY
              DCTXF ;YES, EXIT NONZERO
              CDTMV1+1,X ;DEC HI BYTE
         DEC
DCTIM1: DEC CDTMV1,X ;DEC LO BYTE
        BNE
              DCTXF
        LDY
              CDTMV1+1,X
              DCTXF
        BNE
                            ; WENT ZERO. RETURN ZERO
        LDA
               #0
        RTS
DCTXF: LDA #$FF ;RETURN NONZERO
        RTS
         .PAGE
; SUBROUTINE TO SET VERTICAL BLANK VECTORS AND TIMERS
; ENTRY X=HI,Y=LO BYTE TO SET
       A= 1-5 TIMERS 1-5
           6 IMM VBLANK
           7 DEF VBLANK
;
SETVBL: ASL A
                      ; MUL BY 2
         STA
               INTEMP
         TXA
         LDX
               #5
        אנעם #5
STA WSYNC
                        ; WASTE 20 CPU CYCLES
SETLOP: DEX
                         ;TO ALOWD VBLANK TO HAPPEN
              SETLOP
        BNE
                           ; IF THIS IS LINE "7C"
         LDX
               INTEMP
         STA
              CDTMV1-1,X
         TYA
         STA CDTMV1-2,X
        RTS
```

```
XITVBL: PLA
                           ;UNSTACK Y
         TAY
                                ;UNSTACK X
         PLA
         TAX
         PLA
                                ;UNSTACK A
         RTI
                                ; AND GO BACK FROM WHENCE.
            PIRQ/256
PIRQH =
PIRQL =
               (-256)*PIRQH+PIRQ
              PNMI/256
PNMIH =
PNMIL =
               (-256)*PNMIH+PNMI
; SPARE BYTE OR MODULE TOO LONG FLAG
CRNTP2 =*
        *=$14
INTSPR: .BYTE SIOORG-CRNTP2; GINTHV IS TOO LONG
         .TITLE 'SIO ( SERIAL BUS INPUT/OUTPUT CONTROLLER )'
        COLLEEN OPERATING SYSTEM
        SIO ( SERIAL BUS INPUT/OUTPUT CONTROLLER )
        WITH SOFTWARE BAUD RATE CORRECTION ON CASSETTE
                   AL MILLER 3-APR-19
; THIS MODULE HAS ONE ENTRY POINT. IT IS CALLED BY THE DEVICE
; HANDLERS. IT INTERPRETS A PREVIOUSLY ESTABLISHED DEVICE CONTROL
; BLOCK (STORED IN GLOBAL RAM) TO ISSUE COMMANDS
; TO THE SERIAL BUS TO CONTROL TRANSMITTING AND RECEIVING DATA.
         .PAGE
; EQUATES
; DCD DEVICE BUS ID NUMBERS
FLOPPY = $30
;PRINTR =
              $40
                              ;!!!!! *****
;CASSET =
               $80
                              ;!!!!! ****
              $60
CASET =
;BUS COMMANDS
READ =
                'R
WRITE =
                ' W
;STATIS = 'S
;FORMAT = '!
; COMMAND AUX BYTES
SIDWAY =
                           ; PRINT 18 CHARACTERS SIDEWAYS
                           ; PRINT 40 CHARACTERS NORMALLY
                'N
NORMAL =
DOUBLE =
               'D
                           ; PRINT 20 CHARACTERS DOUBLE WIDE
                ' P
PLOT =
                           ; PLOT MODE
```

; EXIT FROM VERTICAL BLANK

```
; BUS RESPONSES
ACK = 'A ; DEVICE ACKNOWLEDGES INFORMATION
                      ;DEVICE DID NOT UNDERSTAND
NACK =
            'N
COMPLT =
               'C
                          ; DEVICE SUCCESSFULLY COMPLETED OPERATION
ERROR =
               ' E
                           ; DEVICE INCURRED AN ERROR IN AN ATTEMPTED OP
; MISCELLANEOUS EOUATES
B192LO =
              $28
                             ;19200 BAUD RATE POKEY COUNTER VALUES (LO BY
              $00
B192HI =
                             ; (HI BYTE)
              $CC
                             ;600 BAUD (LO BYTE)
B600LO =
B600HI =
              $05
                             ;(HI BYTE)
HITONE =
              $05
                            ;FSK HI FREQ POKEY COUNTER VALUE (5326 HZ)
LOTONE =
              $07
                             ;FSK LO FREQ POKEY COUNTER VALUE (3995 HZ)
       .IF Paul
               PALFLG
WIRGLO =
                             ; WRITE INTER RECORD GAP (IN 1/60 SEC)
              100
RIRGLO =
                            ; READ INTER RECORD GAP (IN 1/60 SEC)
               13
                            ;SHORT WRITE INTER RECORD GAP;SHORT READ INTER RECORD GAP
WSIRG =
RSIRG =
                 8
        .ENDIF
        .IF PALFLG-1
                        ;WRITE INTER RECORD GAP (IN 1/60 SEC);READ INTER RECORD GAP (IN 1/60 SEC);SHORT WRITE INTER RECORD GAP
WIRGLO =
              180
              120
RIRGLO =
WSIRG =
              15
RSIRG =
               10
                          ; SHORT READ INTER RECORD GAP
       .ENDIF
WIRGHI =
              0
RIRGHI =
               Ω
NCOMLO = $34
                             ; PIA COMMAND TO LOWER NOT COMMAND LINE
NCOMHI =
               $3C
                             ; PIA COMMAND TO RAISE NOT COMMAND LINE
                             ; PIA COMMAND TO TURN ON CASSETTE MOTOR
MOTRGO =
              $34
MOTRST = $3C
                             ; PIA COMMAND TO TURN OFF MOTOR
TEMPHI = TEMP/256 ;ADDRESS OF TEMP CELL (HI BYTE)
             (-256)*TEMPHI+TEMP ;(LO BYTE)
CDEVIC/256 ;ADDRESS OF COMMAND BUFFER (HI BYTE)
TEMPLO =
CBUFHI =
CBUFLO = (-256)*CBUFHI+CDEVIC;(LO BYTE)
                         ; NUMBER OF COMMAND FRAME RETRIES
CRETRI = 13
DRETRI =
              1
                          ; NUMBER OF DEVICE RETRIES
CTIMLO =
               2
                           ; COMMAND FRAME ACK TIME OUT (LO BYTE)
CTIMHI =
              0
                           ; COMMAND FRAME ACK TIME OUT (HI BYTE)
;JTADRH = JTIMER/256 ;HI BYTE OF JUMP TIMER ROUTINE ADDR
;JTADRL =
              (-256)*JTADRH+JTIMER ;"MOVED TO LINE 1428"
        .PAGE
        SIO
         *=SIOV
         JMP
               SIO
                             ;SIO ENTRY POINT
```

```
*=SIOINV
         JMP SIOINT ;SIO INITIALIZATION ENTRY POINT
         *=SENDEV
         JMP SENDEN ;SEND ENABLE ENTRY POINT
         *=VCTABL-INTABS+VSERIN
         .WORD ISRSIR ;VSERIN .WORD ISRODN ;VSEROR
         .WORD ISRTD ; VSEROC
         *=SIOORG
; SIO INITIALIZATION SUBROUTINE
SIOINT: LDA #MOTRST
        STA PACTL ; TURN OFF MOTOR
         LDA #NCOMHI
         STA
               PBCTL ; RAISE NOT COMMAND LINE
;
        LDA #3
STA SSKCTL ;GET POKEY OUT OF INITIALIZE MODE
              SOUNDR
SKCTL
         STA
                           ; INIT POKE ADDRESS FOR OUIET I/O
         STA
         RTS
                            ; RETURN
      TSX
SIO:
         STX
              STACKP
                       ; SAVE STACK POINTER
        LDA
               #1
              CRITIC
        STA
        LDA DDEVIC
         CMP
               #CASET
                            ; BRANCH IF NOT CASSETTE
         BNE
               NOTCST
                            ;OTHERWISE JUMP TO CASSETTE ENTER
         JMP
               CASENT
; ALL DEVICES EXCEPT CASSETTE ARE INTELLIGENT
NOTCST: LDA #0
              CASFLG ; INIT CASSETTE FLAG TO NO CASSETTE
        STA
               #DRETRI ;SET NUMBER OF DEVICE RETRIES
         LDA
               DRETRY
COMMND: LDA #CRETRI ;SET NUMBER OF COMMAND FRAMERETRIES
        STA CRETRY
```

```
; SEND A COMMAND FRAME
COMFRM: LDA #B192LO ;SET BAUD RATE TO 19200
        STA
              AUDF3
        LDA
              #B192HI
        STA
              AUDF4
        CLC
                             ;SET UP COMMAND BUFFER
              DDEVIC
        LDA
              DUNIT
        ADC
              #$FF
                         ;SUBTRACT 1
        ADC
        STA
              CDEVIC
                           ;SET BUS ID NUMBER
        LDA DCOMND
STA CCOMND
                           ;SET BUS COMMAND
        LDA DAUX1 ;STORE COMMAND FRAME AUX BYTES 1 AND 2
        STA
              CAUX1
        LDA
               DAUX2
              CAUX2 ; DONE SETTING UP COMMAND BUFFER
        STA
        CLC
                             ;SET BUFFER POINTER TO COMMAND FRAME BUFFER
              #CBUFLO
        LDA
        STA
                           ; AND BUFFER END ADDRESS
               BUFRLO
        ADC
               #4
        STA
              BFENLO
              #CBUFHI
        LDA
              BUFRHI
        STA
        STA
              BFENHI
                           ; DONE SETTING UP BUFFER POINTER
        LDA #NCOMLO
STA PBCTL
                         ;LOWER NOT COMMAND LINE
        JSR SENDIN
                         ; SEND THE COMMAND FRAME TO A SMART DEVICE
              ERRFLG
        LDA
                            ;BRANCH IF AN ERROR RECEIVED
        BNE
               BADCOM
        TYA
        BNE ACKREC
                           ;BRANCH IF ACK RECEIVED
;
BADCOM: DEC CRETRY ; A NACK OR TIME OUT OCCURED
        BPL COMFRM
                            ; SO BRANCH IF ANY RETRIES LEFT
        JMP DERR1
                        ;OTHERWISE, JUMP TO RETURN SECTION
ACKREC: LDA DSTATS
                        ; ACK WAS RECEIVED
        BPL WATCOM ;BRANCH TO WAIT FOR COMPLETE
; IF THERE IS NO DATA TO BE SENT
; SEND A DATA FRAME TO PERIPHERAL
              #CRETRI ;SET NUMBER OF RETRIES
        LDA
        STA
              CRETRY
```

```
;LOAD BUFFER POINTER WITH DCB INFORMATION
        JSR
              LDPNTR
        JSR SENDIN
                            ;GO SEND THE DATA FRAME TO A SMART DEVICE
        BEQ BADCOM ; BRANCH IF BAD
; WAIT FOR COMPLETE SIGNAL FROM PERIPHERAL
WATCOM: JSR STTMOT ;SET DDEVICE TIME OUT VALUES IN Y,X
              #$00
        LDA
        STA
                           ;CLEAR ERROR FLAG
               ERRFLG
                         ;SET UP TIMER AND WAIT;BRANCH IF TIME OUT
        JSR WAITER
        BEQ
               DERR
; DEVICE DID NOT TIME OUT
        BIT DSTATS
        BVS
              MODATA
                       BRANCH IF MORE DATA FOLLOWS
              ERRFLG
        LDA
        BNE DERR1 ;BRANCH IF AN ERROR OCCURRED BEQ RETURN ;OTHERWISE RETURN
;
; RECEIVE A DATA FRAME FROM PERIPHERAL
MODATA: JSR LDPNTR ;LOAD BUFFER POINTER WITH DCB INFORMATION
               RECEIV
        JSR
                            ;GO RECEIVE A DATA FRAME
DERR: LDA ERRFLG
        BEO NOTERR ; BRANCH IF NO ERROR PRECEEDED DATA
        LDA
              TSTAT ;GET TEMP STATUS
               STATUS
        STA
                            ;STORE IN REAL STATUS
NOTERR: LDA STATUS
        CMP #SUCCES
        BEQ
                            ; BRANCH IF COMPLETELY SUCCESSFUL
               RETURN
DERR1: DEC DRETRY
        BMI RETURN ; BRANCH IF OUT OF DEVICE RETRIES
        JMP COMMND
                           ;OTHERWISE ONE MORE TIME
RETURN: JSR SENDDS ; DISABLE POKEY INTERRUPTS
        LDA #0
        STA CRITIC
```

```
LDY
                STATUS
                              ; RETURN STATUS IN Y
         STY
                 DSTATS
                              ; AND THE DCB STATUS WORD
         RTS
                 RETURN
; WAIT SUBROUTINE
; WAITS FOR COMPLETE OR ACK
; RETURNS Y=$FF IF SUCCESSFUL, Y=$00 IF NOT
WAIT:
       LDA #$00
                             ;CLEAR ERROR FLAG
         STA
               ERRFLG
         CLC
                                 ;LOAD BUFFER POINTER WITH ADDRESS
         LDA
                 #TEMPLO
                           ;OF TEMPORARY RAM CELL
         STA
                BUFRLO
         ADC
                #1
         STA
                              ;ALSO SET BUFFER END +1 ADDRESS
                BFENLO
         LDA
                #TEMPHI
               BUFRHI
         STA
         STA
               BFENHI
                              ; DONE LOADING POINTER
         LDA
                #$FF
         STA
                NOCKSM
                              ;SET NO CHECKSUM FOLLOWS DATA FLAG
         JSR RECEIV
                              ;GO RECEIVE A BYTE
                #$FF
                            ; ASSUME SUCCESS
         LDY
         LDA
                 STATUS
         CMP
                #SUCCES
         BNE
                NWOK
                            ;BRANCH IF IT DID NOT WORK OK
ï
                           ; MAKE SURE THE BYTE SUCCESSFULLY RECEIVED
WOK:
       LDA
                TEMP
         CMP
                #ACK
                            ; WAS ACTUALLY AN ACK OR COMPLETE
         BEQ
                GOOD
         CMP
                #COMPLT
         BEO
                 GOOD
         CMP
                #ERROR
                              ;BRANCH IF DEVICE DID NOT SEND BACK
         BNE
                 NOTDER
; A DEVICE ERROR CODE
         LDA
                #DERROR
         STA
                 STATUS
                              ;SET DEVICE ERROR STATUS
         BNE
                 NWOK
NOTDER: LDA #DNACK
                         OTHERWISE SET HACK STATUS
         STA
                STATUS
NWOK:
       LDA STATUS
         CMP
               #TIMOUT
         BEQ
                 BAD
                              ;BRANCH IF TIME OUT
;
         LDA
                #$FF
         STA
                 ERRFLG
                              ;SET SOME ERROR FLAG
```

```
; RETURN WITH OUT SETTING Y = 0
         BNE
               GOOD
                #0
BAD:
       LDY
GOOD:
     LDA STATUS
        STA TSTAT
                               ; RETURN
         RTS
; SEND SUBROUTINE
; SENDS A BUFFER OF BYTES OUT OVER THE SERIAL BUS
SEND: LDA #SUCCES
                       ; ASSUME SUCCESS
         STA
               STATUS
         JSR
               SENDEN
                             ; ENABLE SENDING
         LDY
              #0
         STY
                             ;CLEAR CHECK SUM
                CHKSUM
         STY
                CHKSNT
                             ; CHECKSUM SENT FLAG
                             ;TRANSMISSION DONE FLAG
         STY
               XMTDON
         LDA
                (BUFRLO), Y ; PUT FIRST BYTE FROM BUFFER
         STA
                             ; INTO THE SERIAL OUTPUT REGISTER
                SEROUT
         STA CHKSUM ; PUT IT IN CHECKSUM
NOTDON: LDA BRKKEY
         BNE
               NTBRKO
                          ;JUMP IF BREAK KEY PRESSED
         JMP
               BROKE
NTBRKO: LDA XMTDON
                         ;LOOP UNTIL TRANSMISSION IS DONE
         BEO NOTDON
         JSR
                             ; DISABLE SENDING
               SENDDS
         RTS RETURN
; OUTPUT DATA NEEDED INTERRUPT SERVICE ROUTINE
ISRODN: TYA
                               ; SAVE Y REG ON STACK
         PHA
;
         INC
              BUFRLO
                          ; INCREMENT DUFFER POINTER
         BNE
               NOWRPO
         INC
               BUFRHI
```

```
; CHECK IF PAST END OF BUFFER
NOWRPO: LDA BUFRLO
         CMP
               BFENLO
               BUFRHI
         LDA
                            ;HIGH PART
         SBC
               BFENHI
             NOTEND
         BCC
                            BRANCH IF NOT PAST END OF BUFFER
         LDA
               CHKSNT
         BNE
               RELONE
                             ; BRANCH IF CHECKSUM ALREADY SENT
             CHKSUM
         LDA
               SEROUT
                             ; SEND CHECK SUM
         STA
         LDA
               #$FF
         STA
                CHKSNT
                             ;SET CHECKSUM SENT FLAG
         BNE
               CHKDON
RELONE: LDA POKMSK
                        ; ENABLE TRANSMIT DONE INTERRUPT
         ORA
               #$08
         STA
               POKMSK
         STA
               IROEN
CHKDON: PLA
         TAY
                               ; RESTORE Y REG
         PLA
                               ; RETURN FROM INTERRUPT
         RTI
;
NOTEND: LDY #0
               (BUFRLO),Y ; PUT NEXT BYTE FROM BUFFER
         LDA
         STA
               SEROUT
                            ; INTO THE SERIAL OUTPUT REGISTER
         CLC
                               ; ADD IT TO CHECKSUM
              CHKSUM
         ADC
         ADC
                #0
              CHKSUM
         STA
         JMP
               CHKDON
                            GO RETURN
; TRANSMIT DONE INTERRUPT SERVICE ROUTINE
ISRTD: LDA CHKSNT
         BEQ FOOEY
                         BRANCH IF CHECKSUM NOT YET SENT
         STA
               XMTDON
                             ;OTHERWISE SET TRANSMISSION DONE FLAG
              POKMSK
         LDA
                          ; DISABLE TRANSMIT DONE INTERRUPT
         AND
               #$F7
         STA
               POKMSK
         STA
               IRQEN
FOOEY: PLA
                          ; RETURN FROM INTERRUPT
         RTI
;
```

```
; RECEIVE SUBROUTINE
RECEIV: LDA #0
         LDY CASFLG
               NOCLR ; BRANCH IF CASSETTE
         BNE
                             ; CLEAR CHKSUM
        STA CHKSUM
NOCLR: STA BUFRFL ;
STA RECVDN
                        ;BUFFER FULL FLAG
                            ; RECEIVE DONE FLAG
         LDA #SUCCES
               STATUS
RECVEN
         STA
                            ;SET GOOD STATUS FOR DEFAULT CASE.
         JSR
                            ;DO RECEIVE ENABLE
              #NCOMHI ; COMMAND FRAME HI COMMAND
PBCTL ; STORE IN PIA
         LDA
         STA
CHKTIM: LDA BRKKEY
         BNE
               NTBRK1
         JMP
                         ;JUMP IF BREAK KEY PRESSED
               BROKE
NTBRK1: LDA TIMFLG
                        ;NO,
         BEO TOUT
                         ; IF TIMEOUT, GO SET ERROR STATUS
               RECVDN
         LDA
         BEQ
               CHKTIM
                          ;DONE ?
GOBACK: RTS
TOUT: LDA #TIMOUT ;YES,
        STA STATUS ;SET TIMEOUT STATUS
RRETRN: RTS
                          ; RETURN
; SERIAL INPUT READY INTERRUPT SERVICE ROUTINE
ISRSIR: TYA
        PHA
                               ; SAVE Y REG ON STACK
             SKSTAT
         LDA
               SKRES ; RESET STATUS REGISTER
         STA
  ****** THIS MAY NOT BE THE PLACE TO DO IT *******
```

```
BMI
               NTFRAM
                            BRANCH IF NO FRAMING ERROR
         LDY
               #FRMERR
         STY
               STATUS
                            ;SET FRAME ERRORR STATUS
NTFRAM: AND #$20
         BNE
                            ;BRANCH IF NO OVERRUN ERROR
               NTOVRN
;
         LDY
               #OVRRUN
         STY
                            ;SET OVERRUN ERROR STATUS
               STATUS
;
NTOVRN: LDA BUFRFL
        BEQ
               NOTYET
                            ; BRANCH IF BUFFER WAS NOTYET FILLED
        LDA
                        THIS INPUT BYTE 15 THE CHECKSUM
               SERIN
         CMP
               CHKSUM
                            ; BRANCH IF CHECKSUMS MATCH
         BEQ
               SRETRN
;
         LDY
               #CHKERR
         STY
               STATUS
                            ;SET CHECKSUM ERROR STATUS
SRETRN: LDA #$FF ;SET RECEIVE DONE FLAG
        STA RECVDN
SUSUAL: PLA
         TAY
                              ; RESTORE Y REG
         PLA
                              ; RETURN FROM INTERRUPT
         RTI
NOTYET: LDA SERIN
         LDY
              #0
               (BUFRLO), Y ; STORE INPUT REGISTER INTO BUFFER
         STA
;
                              ; ADD IT TO CHECKSUM
         CLC
         ADC
              CHKSUM
         ADC
               #0
             CHKSUM
         STA
         INC
               BUFRLO
                            ; INCREMENT BUFFER POINTER
         BNE
               NTWRP1
         INC
               BUFRHI
NTWRP1: LDA BUFRLO
         CMP BFENLO
         LDA
               BUFRHI
         SBC
               BFENHI
         DCC
               SUSUAL
                            ; BRANCH IF NEW BUFFER ADDRESS IS IN BUFFER L
         LDA
             NOCKSM
         BEQ
               GOON
                          ;BRANCH IF A CHECKSUM WILL FOLLOW DATA
         LDA
               #0
                            CLEAR NO CHECKSUM FLAG
         STA
               NOCKSM
         BEQ SRETRN
                       GO RETURN AND SET RECEIVE DONE FLAG
```

```
GOON: LDA #$FF
        STA
              BUFRFL
                           ;SET BUFFER FULL FLAG
        BNE SUSUAL
                      ;GO RETURN
; LOAD BUFFER POINTER SUBROUTINE
; LOAD BUFFER POINTER WITH DCB BUFFER INFORMATION
LDPNTR: CLC
        LDA
              DBUFLO
        STA
               BUFRLO
        ADC
              DBYTLO
        STA BFENLO
                      ;ALSO SET SUFFER END + 1 ADDRESS
              DBUFHI
        LDA
        STA
               BUFRHI
              DBYTHI
        ADC
        STA
              BFENHI
        RTS
                             ; RETURN
; CASSETTE HANDLING CODE
CASENT: LDA DSTATS
                      ; BRANCH IF INPUT FROM CASSETTE
       BPL CASRED
; WRITE A RECORD
            #B600LO ;SET BAUD RATE TO 600 AUDF3
        LDA
        STA
        LDA
               #B600HI
        STA
               AUDF4
        JSR SENDEN
                           ;TURN ON POKEY MARK TONE
        LDY #WSIRG
                         ;LOAD SHORT WRITE INTER RECORD GAP TIME
        LDA
               DAUX2
                            ; BRANCH IF SHORT GAP IS DESIRED
        BMI
               SRTIR0
               #WIRGLO ;SET WRITE IRQ TIME
        LDY
SRTIRO: LDX #WIRGHI
               SETVBX
        JSR
        LDA
               #MOTRGO
```

```
STA
             PACTL
                      ;TURN ON MOTOR
TIMIT: LDA TIMFLG
                     ;LOOP UNTIL DONE
        BNE TIMIT
;
        JSR LDPNTR
                        ;LOAD BUFFER POINTER WITH DCB INFORMATION
             SEND
                       ;SEND A BUFFER
        JSR
        JMP CRETRN ;GO, RETURN
; RECEIVE A RECORD
CASRED: LDA #$FF
        STA CASFLG ;SET SET CASSETTE FLAG
;
        LDY
             #RSIRG
                        ;LOAD SHORT READ INTER RECORD GAP TIME
             DAUX2
        LDA
        BMI SRTIR1
                        ;BRANCH IF SHORT GAP IS DESIRED
;
       LDY #RIRGLO ;SET TIME OUT FOR READ IRQ
SRTIR1: LDX #RIRGHI
        JSR
             SETVBX
        LDA #MOTRGO
STA PACTL ;TURN ON MOTOR
TIMIT1: LDA TIMFLG
                     ;LOOP UNTIL DONE
       BNE TIMIT1
        JSR LDPNTR ;LOAD BUFFER POINTER WITH DOS INFORMATION
        JSR
            STTMOT
                        ;SET DEVICE TIME OUT IN Y,X
        JSR
             SETVBX
ï
        JSR BEGIN ;SET INITIAL BAUD RATE
        JSR RECEIV ;GO RECEIVE A BLOCK
CRETRN: LDA DAUX2
                        ; BRANCH IF DOING SHORT INTER RECORD GAPS
       BMI
             SRTIR2
; DON'T TURN OFF CASSETTE MOTOR
        LDA
             #MOTRST
             PACTL
                      ;TURN OFF MOTOR
SRTIR2: JMP RETURN
                     ;GO RETURN
RTS
```

```
; SEND ENABLE SUBROUTINE
                       ; MASK OFF PREVIOUS SERIAL BUS CONTROL BITS
SENDEN: LDA #$07
         AND
               SSKCTL
               #$20
                        ;SET TRANSMIT MODE
         ORA
         LDY
               DDEVIC
         CPY
                #CASET
               NOTCAS
                             ;BRANCH IF NOT CASSETTE
         BNE
             #$08
                         ;SET THE FSK OUTPUT BIT
         ORA
         LDY
               #LOTONE
                           ;SET FSK TONE FREQUENCIES
         STY
                AUDF2
         LDY
                #HITONE
         STY
               AUDF1
                         ;STORE NEW VALUE TO SYSTEM MASK
NOTCAS: STA SSKCTL
                          ;STORE TO ACTUAL REGISTER
         STA
                SKCTL
                #$C7
         LDA
                           ; MASK OFF PREVIOUS SERIAL BUS INTERRUPT BITS
         AND
                POKMSK
                           ; ENABLE OUTPUT DATA NEEDED INTERRUPT
         ORA
                #$10
         JMP
               CONTIN
                             GO CONTINUE IN RECEIVE ENABLE SUBROUTINE
; RECEIVE ENABLE SUBROUTINE
                  ; MASK OFF PREVIOUS SERIAL BUS CONTROL BITS
RECVEN: LDA
           #$07
         AND
               SSKCTL
         ORA
               #$10
                           ;SET RECEIVE MODE ASYNCH.
                             ;STORE NEW VALUE TO SYSTEM MASK
         STA
                SSKCTL
                           ;STORE TO ACTUAL REGISTER
         STA
                SKCTL
         STA
              SKRES ; RESET SERIAL PORT/KEYBOARD STATUS REGISTER
         LDA
                #$C7
                           ; MASK OFF PREVIOUS SERIAL BUS INTERRUPTBITS
         AND
                POKMSK
                #$20
                           ; ENABLE RECEIVE INTERRUPT
         ORA
                         ;STORE NEW VALUE TO SYSTEM MASK
CONTIN: STA POKMSK
         STA
                IRQEN
                          ;STORE TO ACTUAL. REGISTER
         LDA
                #$28
                          ;CLOCK CH.3 WITH 1.79 MHZ
```

```
;CLOCK CH.4 WITH CH. 3
         STA
                AUDCTL
                #6
         LDX
                              ;SET PURE TONES, NO VOLUME
         LDA
                #$A8
                             ;TEST QUIET I/O FLAG
         LDY
                SOUNDR
                             ; NE IS NORMAL (NOISY)
         BNE
                NOISE1
         LDA
                #$A0
NOISE1: STA AUDC1,X
         DEX
         DEX
         BPL
             NOISE1
         LDA
                #$A0
         STA
                AUDC3
                           ;TURN OFF SOUND ON CHANNEL 3
               DDEVIC
         LDY
               #CASET
CAS31
         CPY
                           ; BRANCH IF CASSETTE IS DESIRED
         BEQ
         STA
               AUDC1
                           ;OTHERWISE TURN OFF CHANNELS 1 AND 2
         STA
                AUDC2
CAS31: RTS
                           ; RETURN
; DISABLE SEND AND DISABLE RECEIVE SUBROUTINES
SENDDS: NOP
RECVDS: LDA #$C7
                       ; MASK OFF SERIAL BUS INTERRUPTS
         AND POKMSK
                             ;STORE NEW VALUE TO SYSTEM MASK
         STA
               POKMSK
               IROEN ;STORE TO ACTUAL REGISTER
         STA
;
         LDX
                #6
         LDA
                 #0
ZERIT: STA AUDC11X
         DEX
         DEX
         \mathtt{BPL}
               ZERIT ;TURN OFF AUDIO VOLUME
         RTS
                               ; RETURN
```

; SET DDEVICE TIME OUT VALUES IN Y,X SUBROUTINE

```
STTMOT: LDA DTIMLO
                         GET DEVICE TIME OUT IN 1 SECOND INCR
                           ; PUT 6 HI BITS IN X, LO 2 BITS IN Y
         ROR A
         ROR
               Α
         TAY
                                ;TEMP SAVE
         AND #$3F ;MASK OFF 2 HI BITS
                               ;THIS IS HZ BYTE OF TIME OUT
         TAX
         TYA
                               ; RESTORE
               A
         ROR
         AND
               #$C0
                         ; MASK OFF ALL BUT 2 HI BITS
         TAY
                               ;THIS IS LO BYTE OF TIME OUT
         RTS
INTTBL: .WORD ISRSIR ;SERIAL INPUT READY .WORD ISRODN ;OUTPUT DATA NEEDED
         .WORD ISRTD
                          ;TRANSMISSION DONE
;
SIRHI =
              ISRSIR/256 ; SERIAL INPUT READY ISR ADDRESS
SIRLO =
               (-256)*SIRHI+ISRSIR
ODNHI =
              ISRODN/256 ;OUTPUT DATA NEEDED ISR ADDRESS
ODNLO =
               (-256)*ODNHI+ISRODN
     =
TDHI
               ISRTD/256 ;TRANSMISSION DONE ISR ADDRESS
IDLO
               (-256)*TDHI+ISRTD
      =
; SEND A DATA FRAME TO AN INTELLIGENT PERIPHERAL SUBROUTINE
SENDIN: LDX #$01
DELAY0: LDY #$FF
DELAY1: DEY
         BNE DELAY1
         DEX
         BNE
               DELAY0
         JSR
               SEND
                       GO SEND THE DATA FRAME
         LDY #CTIMLO ;SET ACK TIME OUT
         LDX
                #CTIMHI
WAITER: JSR SETVBX
         JSR WAIT ; WAIT FOR ACK
                                ; IF Y=0, A TIME OUT OR NACK OCCURED
         TYA
         RTS
                                ; RETURN
```

```
COMPUTE VALUE FOR POKEY FREQ REGS FOR THE BAUD RATE AS
; MEASURED BY AN INTERVAL OF THE 'VCOUNT' TIMER.
COMPUT: STA
            TIMER2
                  TIMER2+1 ;SAVE FINAL TIMER VALUE
          STY
                                ; ADJUST VCOUNT VALUE
          JSR
                  ADJUST
                                ; SAVE ADJUSTED VALUE
          STA
                 TIMER2
          LDA
                  TIMER1
          JSR
                  ADJUST
                                ; ADJUST
          STA
                  TIMER1
                                ; SAVE ADJUSTED TIMER1 VALUE
          LDA
                  TIMER2
          SEC
          SBC
                  TIMER1
          STA
                  TEMP1
                             ;FIND VCOUNT DIFFERENCE
          LDA
                  TIMER2+1
          SEC
               TIMER1+1
          SBC
          TAY
                                  ;FIND VBLANK COUNT DIFFERENCE
          .IF
                  PALFLG
          LDA
                  #-$9C
HITIMR: CLC
                 #$9C
          ADC
          .ENDIF
                 PALFLG-1
          .IF
          LDA
                  #-$83
HITIMR: CLC
                  #$83
          ADC
                             ; ACCUMULATE MULTIPLICATION
          .ENDIF
          DEY
          BPL
                  HITIMR
                                ; DONE?
          CLC
                  TEMP1
          ADC
                            ;TOTAL VCOUNT DIFFERENCE
                              ;SAVE ACCUM
FINDX: TAY
          LSR
                  Α
          LSR
                  Α
          LSR
                  Α
          ASL
                  Α
          SEC
          SBC
                  #22
                                ; ADJUST TABLE INDEX
                                  ; DIVIDE INTERVAL BY 4 TO CET TABLE INDEX
          TAX
          TYA
                                  ; RESTORE ACCUM
                  #7
          AND
          TAY
                                  ; PULL OFF 3 LO BITS OF INTERVAL
          LDA
                 #-11
DOINTP: CLC
                 #11
                                ; ACCUMULATE INTERPOLATION CONSTANT
          ADC
          DEY
          BPL
                  DOINTP
                                ; INTERPOLATION CONSTANT COMPUTATION DONE?
```

```
ENINTP: LDY #0
         STY ADDCOR ; CLEAR ADDITION CORRECTION FLAG
         SEC
         SBC #7
                            ; ADJUST INTERPOLATION CONSTANT
              PLUS
ADDCOR
         BPL
         DEC
PLUS: CLC
         ADC POKTAB, X ; ADD CONSTANT TO LO BYTE TABLE VALUE
         TAY
                               ;LO BYTE POKEY FREQ VALUE
         LDA ADDCOR
         ADC
               POKTAB+1,X ; ADD CARRY TO HI BYTE TABLEVALUE
; HI BYTE POKEY FREQ VALUE
        RTS
       ROUTINE TO ADJUST VCOUNT VALUE
ADJUST: CMP #$7C
        BMI ADJ1 ;LAROER THAN '7C' ?
         SEC
                              ;YES
         SBC #$7C
        RTS
ADJ1: CLC
        .IF PALFLG ADC #$20
         .ENDIF
         .IF PALFLG-1
         ADC
               #$7
         .ENDIF
         RTS
       INITIAL BAUD RATE MEASUREMENT -- USED TO SET THE
                  BAUD RATE AT THE START OF A RECORD.
                  IT IS ASSUMED THAT THE FIRST TWO BYTES OF EVERY
; RECORD ARE 'AA' HEX.
BEGIN: LDA BRKKEY
        BNE NTBRK2
               BROKE ; JUMP IF BREAK KEY PRESSED
         JMP
NTBRK2: SEI
         LDA
              TIMFLG
                -
OKTIM1
               OKTIM1 ;BRANCH IF NOT TIMEDOUT TOUT1 ;BRANCH IF TIME OUT
         BNE
         BEQ
OKTIM1: LDA SKSTAT
         AND #$10
                          ; READ SERIAL PORT
         BNE
               BEGIN
                         ;START BIT?
               SAVIO
                        ;SAVE SER. DATA IN
         STA
```

```
VCOUNT
               RTCLOK+2 ; READ LO BYTE OF VBLANK CLOCK
        LDY
        STX
               TIMER1
        STY TIMER1+1 ; SAVE INITIAL TIMER VALUE
;
        LDX
              #1
                            ;SET MODE FLAG
        STX
               TEMP3
        LDY
               #10
                           ;SET BIT COUNTER FOR 10 BITS
COUNT: LDA BRKKEY
               BROKE ; BRANCH IF BREAK KEY PRESSED
        BEQ
;
        LDA
              TIMFLG
        BNE
               OKTIMR
                           ;BRANCH IF NOT TIMED OUT
TOUT1: CLI
        JMP
              TOUT ; BRANCH IF TIME OUT
OKTIMR: LDA SKSTAT
        AND #$10
                         ; READ SERIAL PORT
        CMP
               SAVIO
                         ; DATA IN CHANGED YET?
        BEO
               COUNT
        STA
              SAVIO ; YES, SAVE SER. DATA IN
        DEY
                             ; DECR. BIT COUNTER
        BNE COUNT
                        ; DONE?
        DEC
              TEMP3
                        ;YES,
        BMI
              GOREAD
                           ; DONE WITH BOTH MODES?
        LDA
              VCOUNT
        LDY
              RTCLOK+2 ; READ TIMER LO & HI BYTES
        JSR
              COMPUT
                           ; NO, COMPUTE SAUD RATE
        STY
              CBAUDL
                           ;SET BAUD RATE INTO RAM CELLS
        STA
              CBAUDH
        LDY
              #9
                            ;SET BIT COUNTER FOR 9 BITS
        BNE COUNT
GOREAD: LDA CBAUDL
        STA
              AUDF3
        LDA
               CBAUDH
        STA
              AUDF4
                       ; SET POKEY FREQ REGS FOR BAUD RATE
        LDA
               #0
        STA
              SKSTAT
        LDA
              SSKCTL
        STA
              SKSTAT
                           ; INIT. POKEY SERIAL PORT
        LDA
               #$55
            (BUFRLO),Y ;STORE '$55' AS FIRST RCV. BUFFER
        STA
        INY
        STA
              (BUFRLO),Y
        LDA
               #$AA
        STA
               CHKSUM
                          ;STORE CHECKSUM FOR 2 BYTES OF '$AA'
        CLC
        LDA
             BUFRLO
        ADC
               #2
        STA
              BUFRLO
               BUFRHI
        LDA
        ADC
               #0
              BUFRHI ; INCR. BUFFER POINTER BY 1
        STA
        CLI
        RTS
```

; READ VERTICAL LINECOUNTER

LDX

```
BREAK KEY WAS PRESSED, SO PREPARE
BROKE: JSR SENDDS
        LDA #MOTRST
                           ;TO RETURN
         STA PACTL
STA PBCTL
                        ;TURN OFF MOTOR ;RAISE NOT COMMAND LINE
               #BRKABT
         LDA
         STA
                STATUS
                             ;STORE BREAK ABORT STATUS CODE
         LDX STACKP
         TXS
                               ; RESTORE STACK POINTER
               BRKKEY
         DEC
                             ;SET BREAK KEY FLAG TO NONZERO
         CLI
                               ; ALLOW IRQ'S
         JMP RETURN
                       ; GO RETURN
SETVBX: LDA #JTADRL ;STORE TIME OUT ROUTINE ADDRESS
         STA CDTMA1
         LDA
               #JTADRH
         STA
                CDTMA1+1
         LDA #1
                             ;SET FOR TIMER 1
         SEI
                               ;THE SETVBL ROUTINE NEEDS THIS TO CUT SHORT
               SETVBV
                             ; ANY VBLANKS THAT OCCUR
         JSR
         LDA
               #1
                              ;SET FOR TIMER 1
               TIMFLG
         STA
                          ;SET FLAG TO NOT TIMED OUT
         CLI
         RTS
 'VCOUNT' INTERVAL TIMER MEASUREMENT -- TO -- POKEY FREQ REG VALUE
                  CONVERSION TABLE
 THE VALUES STORED IN THE TABLE ARE 'AUDF+7'.
        THE FOLLOWING FORMULAS WERE USED TO DETERMINE THE TABLE VALUES:
;
                  F OUT F IN/(2*(AUDF+M)) , WHERE F IN=1.78979 MHZ. & M=7
         FROM THIS WAS DERIVED THE FORMULA USED TO COMPUTE THE
         TABLE VALUES BASED ON A MEASUREMENT OF THE PERIOD BY
         AN INTERVAL OF THE 'VCOUNT' TIMER.
                  AUDF+7=(11.365167)*T OUT, WHERE T OUT=# OF COUNTS
                  (127 USEC. RESOLUTION) OF 'VCOUNT' FOR 1
                  CHARACTER TIME (10 BIT TIMES).
```

```
BAUD RATE
                                                VCOUNT INTERVAL
                  AUDF+7
                                 -----
;
       .WORD $27C
                                 ;1407
                                                            56
        .WORD $2D7
                                                            64
                                ;1231
                                                            72
        .WORD $332
                                ;1094
       .WORD $38D
                                ;985
                                                             80
POKTAB: .WORD $3E8
                                                              88
                                 ;895
         .WORD $443
                                 ;820
                                                              96
         .WORD $49E
                                ;757
                                                              104
         .WORD $4F9
                                 ;703
                                                              112
         .WORD $554
                                                              120
                                ;656
         .WORD $5AF
                                                              128
                                ;615
         .WORD $60A
                                 ;579
                                                              136
         .WORD $665
                                ;547
                                                              144
         .WORD $6C0
                                ;518
                                                              152
         .WORD $71A
                                 ;492
                                                              160
         .WORD $775
                                ;469
                                                              168
         .WORD $7D0
                                ;447
                                                              176
        .WORD $828
                                 ;428
                                                              184
;
       .WORD $886
                                ;410
                                                              192
        .WORD $8E1
                                                              200
                                ;394
        .WORD $93C
                                 ;379
                                                              208
;
        .WORD $997
                                ;365
                                                              216
        .WORD $9F2
                                                              224
                                 ;352
        .WORD $A4D
                                ;339
                                                              232
;
        .WORD $AA8
                                ;328
                                                              240
        .WORD $803
                                                              248
                                ;318
;
CRNTP3 =*
        *=$14
SIOSPR: .BYTE DSKORG-CRNTP3 ; GSIOL IS TOO LONG
        .TITLE 'DISK ***** DISKP.SRC ***** 3/9/79 ***** 4:00:00 P.M.'
;
STATVL = ;
STATVH =
              DVSTAT/256
              (-256)*STATVH+DVSTAT ;STATUS POINTER
       CONSTANT EQUATES
                            ;SERIAL BUS DISK I.D.
DISKID =
              $31
                            ;DISK PUT SECTOR DCB COMMAND
PUTSEC =
               $50
              $52
                            ; DISK GET SECTOR DCB COMMAND
; READ =
; WRITE =
              $57
                            ; DISK PUT SECTOR WITH READ CHECK DCB COMMAND
STATC =
               $53
                            ; DISK STATUS DCB COMMAND
```

```
;DISK FORMAT DCB COMMAND !!!!! *****
FOMAT =
              $21
NODAT
              0
                        ;SIO COMMAND FOR "NO DATA" OPERATION
                          ;SIO COMMAND FOR "DATA FROM DEVICE"
GETDAT =
              $40
PUTDAT =
              $80
                          ;SIO COMMAND FOR "DATA TO DEVICE"
       VECTORS
        *=$E450
             DINIT ; DISK INIT. VECTOR
        JMP
        JMP
              DSKIF
                       ; DISK INTERFACE ENTRY POINT
       CONSTANTS
        *=DSKORG
 ********************
       DISK INTERFACE ROUTINE STARTS HERE
 DISK INTERFACE INITIALIZATION ROUTINE
DINIT: LDA #160
        STA
                          ;SET INITIAL DISK TIMEOUT TO 160 SEC
             DSKTIM
        RTS
      DISK INTERFACE ENTRY POINT
DSKIF: LDA #DISKID
        STA
                          ;SET SERIAL BUS ID IN DCB
             DDEVIC
        LDA
              DSKTIM
        LDX
              DCOMND
        CPX
              #FOMAT
                          ; IS COMMAND A FORMAT COMMAND?
        BEQ
              PUTDTO
              #7
                           ; NO, SET TIMEOUT TO 7 SECS.
        LDA
PUTDTO: STA DTIMLO
                      ; PUT DISK TIMEOUT IN DCB
        LDX
              #GETDAT
                        ;SET "GET DATA" COMMAND FOR SIO
              #$80
                        ;SET BYTE COUNT TO 128
        LDY
        LDA
              DCOMND
                          ; READ COMMAND IN DCB
        CMP
              #WRITE
                          ; IS COMMAND A "PUT SECTOR" COMMAND?
```

```
LDX
              #PUTDAT
                          ; YES, SET "PUT DATA" COMMAND FOR 610
CKSTC: CMP #STATC ; IS COMMAND A STATUS COMMAND?
         BNE PUTCNT
         LDA
               #STATVL
         STA
               DBUFLO
         LDA
               #STATVH
         STA DBUFHI
LDY #4
                            ;SET BUFFER ADDR TO GLOBAL STATUS BUFFER
               DBUFHI
                             ; YES, SET BYTE COUNT TO 4
PUTCNT: STX DSTATS ;PUT STATUS COMMAND FOR SIO IN DCB
         STY DBYTLO
         LDA
               #0
        STA DBYTHI
JSR SIOV
BPL GOODST
RTS
                            ; PUT BYTE COUNT IN DCB
                         ;CALL SERIAL I/O.
                            ; NO ERROR
                              ; NO, GO BACK
         RTS
GOODST: LDA DCOMND ; READ THE COMMAND
         CMP #STATC
                        ; WAS IT A STATUS COMMAND?
         BNE
               PUTBC
               PUTADR
         JSR
                            ; PUT BUFFER ADDR IN TEMP REQ.
        LDA (BUFADR),Y ;READ DISK TIMEOUT VALUE BYTE OF STATUS
STA DSKTIM :DITT TO TOO ---
PUTBC: LDA DCOMND
                            ; WAS COMMAND A FORMAT COMMAND?
        CMP #FOMAT
        SK PUTADR ;YES PUT BUFFER, ADDR INTO TEMP REC
        BNE ENDDIF
       JSR PUTADR
FMTD:
TWICE: INY
                              ; INCR BUFFER POINTER BY 2
        TNY
RDBAD: LDA (BUFADR), Y ; READ LO BYTE BAD SECTOR DATA
        CMP #$FF
         BNE TWICE ; IS IT "FF" ?
         INY
                               ;YES,
              (BUFADR),Y ; READ HI BYTE BAD SECTOR DATA
         LDA
         INY
         CMP
               #$FF
         BNE RDBAD ; IS IT "FF" ?
         DEY
                              ;YES
         DEY
         STY
               DBYTLO
                            ; PUT BAD SECTOR BYTE COUNT INTO DCB
         LDA
               #0
         STA DBYTHI
ENDDIF: LDY DSTATS
        RTS
       SUBROUTINES
       PUT BUFFER ADDR FROM DCB INTO TEMP REQ
PUTADR: LDA DBUFLO
         STA BUFADR
         LDA
               DBUFHI
         STA
               BUFADR+1 ; PUT BUFFER ADDR IN TEMP REQ
         RTS
```

BNE CKSTC

```
SPARE BYTE OR MODULE TOO LONG FLAG
CRNTP4 = *
;
DSKSPR: .BYTE PRNORG-CRNTP4 ; GDISKP TOO LONG
         .PAGE
         .TITLE 'PRINTER **** PRINTP.SRC **** 3/9/79 **** 4:00:00 P
       DEVICE NUMBER OR CODE EQUATES
              $2
                          ; IOCB OPEN FOR OUTPUT COMMAND
OPNOUT =
NBUFSZ =
              40
                          ; PRINT NORMAL BUFFER SIZE
                         ; PRINT DOUBLE BUFFFER SIZE
              20
DBUFSZ =
             29
$40
$53
SBUFSZ =
                         ; PRINT SIDEWAYS BUFFER SIZE
PDEVN =
                             ; PRINTER DEVICE NUMBER
; STATC =
                            ; DCB STATUS COMMAND CODE
WRITEC =
              $57
                            ;DCB WRITE COMMAND
             $20
$4E
SPACE =
                             ; ASCII SPACE CHAR.
N
                            ;ASCII "N" CHAR.
   =
              $44
                            ;ASCII "D" CHAR.
D
       =
                             ;ASCII "S" CHAR.
S
               $53
;
    PRINTER HANDLER ENTRY POINTS
         *=$E430
         .WORD PHOPEN-1 ; PRINTER HANDLER OPEN
         .WORD PHCLOS-1
                         ; PH CLOSE
         .WORD BADST-1 ; PH READ
         .WORD PHWRIT-1
                         ; PH WRITE
         .WORD PHSTAT-1
                         ; PH STATUS
         .WORD BADST-1 ; PH SPECIAL
         JMP PHINIT
                        ;PH INIT.
         .BYTE 0
                          ;ROM FILLER
```

*=PRNORG

```
PRINTER HANDLER INITIALIZATION ROUTINE
PHINIT: LDA #30
        STA PTIMOT ;SET UP INITIAL PRINTER TIMEOUT OF 30 SEC.
        RTS
      PRINTER HANDLER CONSTANTS
PHSTLO: .WORD DVSTAT
                       ;STATUS BUFFER POINTER
PHCHLO: .WORD PRNBUF
                      ; CHAR. BUFFER POINTER
       *******
       PRINTER HANDLER ROUTINES
       *******
      PRINTER HANDLER STATUS ROUTINE
PHSTAT: LDA #4
              PBUFSZ
                           ;SET BUFFER SIZE TO 4 BYTES
        STA
        LDX
              PHSTLO
        LDY
              PHSTLO+1 ;SET POINTER TO STATUS BUFFER
              #STATC
                           ;SETCOMMAND TO "STATUS"
        LDA
                           ;SET STATUS COMMAND
        STA
              DCOMND
        STA
              DAUX1
                           ;GO SETUP DCH
        JSR
              SETDCB
        JSR
              SIOV
                         ; SEND STATUS COMMAND
              BADST
                        ;GO IF ERROR
        BMI
              PHPUT
        JSR
                         ; YES, PUT STATUS INTO GLOBAL BUFFER.
BADST: RTS
       PRINTER HANDLER OPEN ROUTINE
PHOPEN: JSR PHSTAT
                       ; DO STATUS COMMAND TO SIO
        LDA
             #0
        STA
              PBPNT
                       ;CLEAR PRINT BUFFER POINTER
        RTS
      PRINTER HANDLER WRITE ROUTINE
PHWRIT: STA PTEMP ; SAVE ACCUM
        JSR
              PRMODE
                         GO DETERMINE PRINTMODE
        LDX
              PBPNT
```

```
PRNBUF,X ; PUT CHAR. IN PRINT BUFFER
        STA
        INX
                             ; INCR. BUFFER POINTER
             PBUFSZ
        CPX
                           ;BUFFER POINTERBUFFER SIZE?
              BUFFUL
        BEO
                        ;SAVE SUFFER POINTER
        STX
              PBPNT
                           ; IS CHAR. = EOL ?
        CMP
              #CR
              BLFILL
        BEQ
                           ; IF YES, GO DO BLANK FILL.
              #SUCCES ; PUT GOOD STATUS IN Y REQ FOR CIO.
        LDY
        RTS
BLFILL: LDA #SPACE ; PUT BLANK IN ACCUM.
FILLBF: STA PRNBUF, X ; STORE IT IN PRINT BUFFER.
        INX
        CPX
              PBUFSZ
        BNE
              FILLBF ;BUFFER BLANK FILLED?
BUFFUL: LDA #0
              PBPNT ; CLEAR PRINT BUFFER POINTER
        STA
        LDX
              PHCHLO
        LDY
              PHCHLO+1 ;SET POINTER TO PRINT BUFFER
        JSR SETDCB
JSR SIOV
                           ;GO SETUP OCR
                        ;SEND PRINT COMMAND
        RTS
                             ;YES.
       PRINTER HANDLER CLOSE ROUTINE
PHCLOS: JSR PRMODE
                       GO DETERMINE PRINT MODE
        LDX PBPNT
              BLFILL
        BNE
        LDY
              #SUCCES
        RTS
       SUBROUTINES
       SET UP DCB TO CALL SIO
SETDCB: STX DBUFLO
        STY DBUFHI ;SET BUFFER POINTER
        LDA
              #PDEVN
        STA
               DDEVIC
                           ;SET PRINTER BUS I.D. FOR DCB
        LDA
               #1
              DUNIT
                        ;SET UNIT NUMBER TO 1
        STA
        LDA
              #$80
                         ;DEVICE WILL EXPECT DATA
              DCOMND
        LDX
        CPX
              #STATC
                           ;STATUS COMMAND?
        BNE
              PSIOC
```

PTEMP ;GET CHAR. SENT BY CID

LDA

```
;SET SIO MODE COMMAND
PSIOC: STA DSTATS
        LDA
              PBUFSZ
        STA
              DBYTLO ;SET LO BYTE COUNT
              #0
        LDA
              DBYTHI
        STA
                           ;SET HI BYTE COUNT
              PTIMOT
        LDA
              DTIMLO
        STA
                           ;SET DEVICE TIMEOUT COUNT
        RTS
; GET DEVICE TIMEOUT FROM STATUS & SAVE IT
PHPUT: LDA DVSTAT+2
        STA PTIMOT ;SAVE DEVICE TIMEOUT
        RTS
;
; DETERMINE PRINT MODE & SETUP PRINT BUFFER SIZE, DCB PRINT
; COMMAND, &. DCB AUX1 FOR PRINT MODE
PRMODE: LDY #WRITEC ; PUT WRITE COMMAND IN Y REG
        LDA ICAX2Z
                           ; READ PRINT MODE
CMODE: CMP #N
        BNE CDUBL ; PRINT NORMAL ?
              #NBUFSZ
        LDX
                         ;YES, SET NORMAL CHAR. BUFFER SIZE
        BNE
              SETBSZ
CDUBL: CMP #D
             CSIDE ; PRINT DOUBLE?
        BNE
        LDX
                         ;YES, SET DOUBLE CHAR. BUFFER SIZE
              #DBUFSZ
        BNE
              SETBSZ
CSIDE: CMP #S
                         ; PRINT SIDEWAYS ?
        BNE GOERR ; IF NOT, GO TO ERROR ROUTINE LDX #SBUFSZ ; YES, SET SIDEWAYS BUFFER S
                         ;YES, SET SIDEWAYS BUFFER SIZE
SETBSZ: STX PBUFSZ ;STORE PRINT BUFFER SIZE STY DCOMND ;STORE DCB COMMAND
        STA
              DAUX1
                        ;STORE DCB AUX1 PRINT MODE
        RTS
GOERR: LDA #N
                         ;SET DEFAULT PRINT MODE TO NORMAL
        BNE CMODE
SPARE BYTE OR MODULE TOO LONG FLAG
CRNTP5 = *
PRNSPR: .BYTE CASORG-CRNTP5 ; GPRINTP TOO LONG
;
        .PAGE
        .TITLE 'CASSET HANDLER 3/12 (DK1:CASCV)'
          CASBUF/256
CBUFH =
CBUFL =
             (-256)*CBUFH+CASBUF
```

; EXPECT DATA FROM DEVICE

#\$40

LDA

```
SRSTA =
               $40
                           ;SIO READ STATUS
SWSTA =
               $80
                            ;SIO WRITE STATUS
              $34
;MOTRGO =
;MOTRST =
             $3C
                        ;DATA RECORD TYPE BYTE
          $FC
DTA =
           $FA
                        ;LAST DATA RECORD
DT1 =
EOT =
          $FE
                        ;END OF TAPE
          $FB
HDR =
                        ;HEADER
                         ; CHANGE TO RECORD MODE TONE
TONE1 =
          2
TONE 2 =
             1
                         ; PRESS PLAY TONE
;
         *=CASETV
         .WORD OPENC-1, CLOSEC-1, GBYTE-1, PBYTE-1, STATU-1, SPECIAL-1
        JMP INIT
         .BYTE 0
                         ;ROM FILLER BYTE
; USED IN MONITP FOR CASSETTE BOOT
ï
         *=RBLOKV
        JMP RBLOK
        *=CSOPIV
        JMP OPINP
         *=CASORG
; INIT ROUTINE
     LDA #$CC
INIT:
        STA CBAUDL
        LDA
               #$05
        STA
               CBAUDH
                           ;SET CASSET BAUD RATE TO 600
                              ;THATS ALL FOLKS
SPECIAL:
        RTS
         .PAGE
; OPEN FUNCTION - WITH NO TIMING ADJUST
OPENC: LDA ICAX2Z
                        GET AX2
                        ;SAVE IT FOR FUTURE REFERENCE
        STA FTYPE
        LDA
               ICAX1Z
        AND
               #$0C
                         ; IN AND OUT BITS
               #$04
        CMP
              OPINP
        BEQ
                         ;SEE IF OPEN FOR OUTPUT
        CMP
               #$08
        BEQ
               OPOUT
        RTS
                              ; IF ALREADY OPEN, RETURN LEAVING STATUS=$84
OPINP: LDA #0
        STA
               WMODE ;SET READ MODE
```

```
#TONE 2
SFH:
        LDA
                             ; TONE FOR PRESS PLAY
         JSR
               BEEP
                           ;GO BEEP
              OPNRTN
#MOTRGO
                            ; IF ERROR DURING BEEP
         BMI
         LDA
                        ;TURN MOTOR ON
         STA
               PACTL
         .IF
               PALFLG
               #$E0
         LDY
         LDX
               #1
         .ENDIF
              PALFLG-1
         .IF
         LDY
               #$40
                          ;5-31-79 9 SEC READ LEADER
         LDX
                #2
         .ENDIF
         LDA
               #3
         STA
                CDTMF3
                            ;SET UP YBLANK TIMER
         JSR
               SETVBV
WAITTM: LDA CDTMF3
         BNE WAITTM
                             ; WAIT FOR MOTOR TO COME UP TO SPEED
               #$80
                           ; NEXT BYTE=NO BYTES IN BUFFER
         LDA
         STA
               BPTR
               BLIM
         STA
         JMP
               OPOK
                          ;OPEN OK
; OPEN FOR OUTPUT
       LDY #BRKABT ; BREAK KEY ABORT STATUS
PBRK:
         DEC BRKKEY ; RESET BREAK KEY
OPNRTN: LDA #0
                          ;CLEAR WRITE MODE FLAG
         STA WMODE
         RTS
                               ; AND EXIT.
OPOUT: LDA #$80
         STA WMODE ;SET WRITE MODE
         LDA
                            ;TELL USER TO TURN ON RECORD MODE
               #TONE1
         JSR
             BLLF
OPNRTN
#$CC
AUDF3
#$05
               BEEP
                             ; IF ERROR DURING BEEP
         BMI
         LDA
                           ;SET BAUD RATE
         STA
                          ; WHICH SEEMS TO BE NESSECARY
                           ; FOR SOME OBSCURE REASON
         LDA
         STA
               AUDF4
         LDA
               #$60
              DDEVIC
SENDEV ;TELL POKEY TO WRITE N
#MOTRGO ;WRITE 5 SEC BLANK TAPE
         STA
                             ;TELL POKEY TO WRITE MARKS
         JSR
         LDA
         STA
               PACTL
         LDA
               #3
         .IF
               PALFLG
         LDX
               #$3
             #$C0
         LDY
         .ENDIF
         .IF
               PALFLG-1
                              ;5/30/79 20 SEC LEADER
         LDX
                #4
         LDY
               #$80
         .ENDIF
         JSR
                SETVBV
                #$FF
         LDA
         STA
               CDTMF3
WDLR:
       LDA BRKKEY
```

; NO EOF YET

STA

FEOF

```
PBRK
         BEO
                          ; IF BREAK DURING WRITE LEADER
         LDA
               CDTMF3
         BNE
               WDLR
         LDA
               #0
                            ; INIT BUFFER POINTER
               BPTR
         STA
OPOK: LDY #SUCCES
        RTS
         .PAGE
; GET BYTE
GBYTE: LDA FEOF ; IF AT EOF ALREADY
        BMI ISEOF
                        ; RETURN EOF STATUS
         LDX
               BPTR
                          ;BUFFER POINTER
         CPX
                          ; IF END OF BUFFER
               BLIM
              RBLOK ;READ ANOTHER BLOCK CASBUF+3,X ;GET NEXT BYTE
         BEO
         LDA
        INC
               BPTR ; DUMP POINTER
        LDY
               #SUCCES
                          ;OK STATUS
       RTS
GBX:
RBLOK: LDA #'R
                        ; READ OPCODE
                       ;SIO ON SYS BUF
         JSR SIOSB
         TYA
              GBX
                            ; IF SIO ERRORS, RETURN
         BMI
         LDA
               #0
        STA BPTR
LDX #$80
LDA CASBUF+2
                          ; RESET POINTER
                          ;DEFAULT # BYTES
         CMP
               #EOT
              ATEOF
                        ; IF HEADER, GO READ AGAIN
         BEQ
         CMP
               #DT1
                          ; IF LAST DATA REC
        BNE
               NLR
        LDX
               CASBUF+130 ;LAST DATA RECORD, GET # BYTES
       STX
               BLIM
NLR:
              GBYTE
                        GET NEXT BYTE
        JMP
ATEOF: DEC FEOF ;SET FEOF ISEOF: LDY #EOFERR ;ENDFILE STATUS
        RTS
        .PAGE
; PUT BYTE TO BUFFER
PBYTE: LDX BPTR ;BUFFER POINTER
         STA CASBUF+3,X ;STORE CHAR AWAY
               BPTR
         INC
                      ;BUMP POINTER
         LDY
               #SUCCES
                         ;OK STATUS
                          ; IF BUFFER FULL
         CPX
               #127
         BEQ
                *+3
         RTS
; WRITE OUT THE BUFFER
            #DTA
         LDA
                        ;RECORD TYPE = DATA
         JSR
               WSIOSB
                           ; DO WRITE ON SYSTEM BUFFER
         LDA
               #0
         STA
               BPTR
                          ; RESET BUFFER POINTER
         RTS
                              ;EXIT.
         .PAGE
; STATUS - RETURN STATUS INFO THRU DVSTAT
```

```
RTS
        .PAGE
; CLOSE
CLOSEC: LDA WMODE ;SEE IF WRITING
       BMI CLWRT ;300 CLOSE FOR WRITE
; CLOSE FOR READ - FLAG CLOSED
        LDY #SUCCES ;SUCCESSFULL
FCAX: LDA #MOTRST ;STOP THE MOTOR IN CASE WAS SHORT IRQ MODE
       STA PACTL
        RTS
CLWRT: LDX BPTR ;BUFFER POINTER
       BEO WTLR ; IF NO DATA BYTES IN BUFFER, NO DT1 REC
             CASBUF+130 ; WRITE TO LAST RECORD
        STX
        LDA
              #DT1 ;REC TYPE
              WSIOSB
        JSR
                          ;WRITE OUT USER BUFFER
        BMI FCAX ;GO IF ERROR
WTLR: LDX #127 ;ZERO BUFFER
       LDA #0
ZTBUF: STA CASBUF+3,X
       DEX
        BPL
              ZTBUF
        LDA
                        ;WRITE EOT RECORD
              #EOT
        JSR WSIOSB
JMP FCAX ;FLAG CLOSED AND EXIT
        .PAGE
; SUBROUTINES
; BEEP - GENERATE TONE ON KEYBOARD SPEAKER
; ON ENTRY A= FREQ
BEEP: STA FREO
BEEP1: LDA RTCLOK+2 ; CURRENT CLOCK
        CLC
        .IF PALE
              PALFLG
        .ENDIF
        .IF PALFLG-1
        ADC #30
                          ; 1 SEC TONE
        .ENDIF
        TAX
            #$FF
CONSOL
WFL:
      LDA
        STA
                          ;TURN ONSPEAKER
        LDA
              #0
        LDY
              #$F0
        DEY
            *-1
CONSOL
        BNE
        STA
                        ;TURN OFF SPEAKER
        LDY
              #$F0
        DEY
        BNE
              *-1
              RTCLOK+2 ;SEE IF 1 SEC IS UP YET
        CPX
        BNE
              {	t WFL}
                        ; COUNT BEEPS
        DEC
              FREQ
        BEO
              WFAK
                        ; IF ALL DONE GO WAIT FOR KEY
        TXA
```

STATU: LDY #SUCCES

```
.IF
              PALFLG
        ADC
              #8
         .ENDIF
              PALFLG-1
         .IF
        ADC
               #10
         .ENDIF
        TAX
        CPX
               RTCLOK+2
        BNE
               *-2
        BEQ
               BEEP1 ; UNCOND DO BEEP AGIN
WEAK:
      JSR WFAK1 ; USE SIMULATED "JMP (KGETCH)"
        TYA
        RTS
WFAK1: LDA KEYBDV+5
        PHA
        LDA KEYBDV+4 ;SIMULATE "JMP (KGETCH)"
        PHA
        RTS
; SIOBS - CALL SIO ON SYSTEM BUFFER
SIOSB: STA DCOMND ; SAVE COMMAND
        LDA
              #0
               DBYTHI ;SET BUFFER LENGTH
        STA
        LDA
               #131
        STA DBYTLO LDA #CBUFH
        STA
              DBUFHI
                            ;SET BUFFER ADDRESS
        LDA
               #CBUFL
             DBUFLO
        STA
CSIO:
      LDA #$60 ; CASSET PSEUDO DEVICE
        STA DDEVIC
        LDA
               #0
        STA
              DUNIT
        LDA
                            ;DEVICE TIMEOUT (5/30/79)
               #35
              #35
DTIMLO
DCOMND
#SRSTA
        STA
        LDA
                            GET COMMAND SACK
        LDY
                            ;SIO READ STATUS COMMAND
        CMP
               # ' R
        BEQ
               *+4
        LDY
               #SWSTA
                           ;SIC WRITE STATUS COMMAND
        STY
                            ;SET STATUS FOR SIO
              DSTATS
        LDA
              FTYPE
                        ; INDICATE IF SHORT IRQ MODE
        STA
              DAUX2
        JSR
               SIOV
                         GO CALL SIO
        RTS
; WSIOSB - WRITE SIC SYSTEM SUFFER
WSIOSB: STA CASBUF+2 ;STORE TYPE BYTE
        LDA #$55
         STA
               CASBUF+0
        STA
               CASBUF+1
        LDA
               # ' W
                            ;WRITE
              SIOSB ; CALL SIO ON SYSTEM BUFFER
        JSR
        RTS
               AND
                            ; RETURN
CRNTP6 =*
        *=$14
```

CLC

```
CASSPR: .BYTE MONORG-CRNTP6 ; GCASCV IS TOO LONG
          .TITLE 'MONITOR ***** MONITP.SRC ***** 3/9/79 ***** 4:00:00 P
        CONSTANT EQUATES
PUTTXT =
                $9
                              ; "PUT TEXT RECORD" CIO COMMANDCODE
                              ; "GET CHARACTER" CIO COMMAND CODE
GETCAR =
                $7
PUTCAR =
                $В
                             ; "PUT CHARACTER" CIO COMMAND CODE
INIMLL =
               $00
                                ; INITIAL HEM LO LOW BYTE
INIMLH =
               $07
                                 ; INITIAL HEM LO HIGH BYTE
                              ;GOOD STATUS CODE
; GOOD =
               $1
; WRITE =
               $57
                                ;WRITE COMMAND
; READ = $52
; STATC = $53
SEX = $0
CLS = $7D
CTRLC = $92
EOF = $88
                                 ; READ COMMAND
                                 ;STATUS COMMAND
                          SCREEN EDITOR 10CR INDEX
                           CLEAR SCREEN CODE
                                 ;KEYBOARD CODE FOR 'CONTROL C'
                            ; CASSETTE END OF FILE CODE
      = $0
LIRQ
                               ;LONG IRQ TYPE CODE
BUFFH =
               (CASBUF+3)/256
               (-256)*BUFFH+CASBUF+3;BUFFER POINTER
BUFFL =
; THE FOLLOWING EOUATES ARE IN THE CARTRIDGE ADDRESS SPACE.
; "B" CARTRIDGE ADDR'S ARE 8000-9FFF (36K CONFIG. ONLY)
; "A" CART. ADDR'S ARE A000-BFFF (36K CONFIG. ONLY)
; "A" CART. ADDR'S ARE B000-BFFF (48K CONFIG. ONLY)
         *=$BFFA
CARTCS: .RES 2
                               ; CARTRIDGE COLD START ADDRESS.
CART: .RES
                               ; CARTRIDGE AVAILABLE FLAG BYTE.
               1
CARTFG: .RES
                               ; CARTRIDGE FLAG BYTE. BIT 0=FLAG1,
CARTAD: .RES
               2
                               ; 2-BYTE CARTRIDGE START VECTOR
        CARTRIDGE FLAG ACTION DEFINITIONS
        BIT
                            ACTION IF SET
                           SPECIAL -- DON'T POWER-UP, JUST RUN CARTRIDGE
        6-3
                           NONE
        2
                           RUN CARTRIDGE
        1
                           NONE
        0
                           BOOT DOS
         ****
        NOTE
```

```
1.IF BIT2 IS 0, GOTO BLACKBOARD MODE.
        2.IF BITO SET THE DISK WILL BE BOOTED BEFORE ANY
          OTHER ACTION.
       POWER-UP VECTOR
; *********
       *=$FFFC
; PVECT .WORD PWRUP
                              POWER-UP VECTOR
ENTRY POINT VECTOR
        *=BLKBDV
        JMP SIGNON ; BLACK BOARD VECTOR
        *=WARMSV
        JMP RESET ; WARM START VECTOR
        *=COLDSV
        JMP PWRUP ; COLD START VECTOR (9000 FOR RAM VECTOR WRIT
        *=$9000
        JSR $900C
        JMP
              PWRUP
                         ;(TO HANDLE RAM VECTOR WRITING)
        JSR $900C
JMP RESET
        *=MONORG
       HANDLER TABLE ENTRIES
TBLENT: .BYTE 'P'
        .WORD PRINTV
         .BYTE 'C'
        .WORD CASETV
        .BYTE 'E'
         .WORD EDITRV
```

```
.BYTE 'S'
       .WORD SCRENV
       .BYTE 'K'
       .WORD KEYBDV
;
;TBLLEN = IDENT-TBLENT-1 HANDLER TABLE LENGTH. "MOVED TO LINE 8
      **** PRINT MESSAGES ****
IDENT: .BYTE CLS, 'ATARI COMPUTER - MEMO PAD', CR
           IDENT/256
IDENTH =
           (-256)*IDENTH+IDENT ;SYSTEM I.D. MSG POINTER
IDENTL =
TBLLEN =
           IDENT-TBLENT-1 ; HANDLER TABLE LENGTH
DERR5: .BYTE 'BOOT ERROR', CR
DERRH = DERR5/256
DERRL =
            (-256)*DERRH+DERR5 ; DISK ERROR MSG POINTER
     DEVICE/FILENAME SPECIFICATIONS
OPNEDT: .BYTE 'E:',CR ;"OPEN SCREEN EDITOR" DEVICE SPEC.
OPNH
     = OPNEDT/256
OPNL
            (-256)*OPNH+OPNEDT ;SCREEN EDITOR OPEN POINTER
RESET BUTTON ROUTINE STARTS HERE
RESET: SEI
                      ; DISABLE IRO INTERRUPTS
           COLDST
       LDA
                       ; WERE WE IN MIDDLE OF COLDSTART?
            PWRUP
                     ; YES, GO TRY IT AGAIN
       BNE
       LDA
            #$FF
            PWRUP1
                       ;SET WARM START FLAG
       BNE
POWER UP ROUTINES START HERE
```

```
PWRUP: SEI
                                 ; DISABLE IRQ INTERRUPTS
         LDA #0
                                    ;CLEAR WARMSTART FLAG
PWRUP1: STA WARMST
                                      ;CLEAR DECIMAL FLAG.
           CLD
           LDX #$FF
           TXS
                                       ;SET STACK POINTER
           JSR SPECL ; CARTRIDGE SPECIAL CASE?

JSR HARDI ; DO HARDWARE INITIALIZAT

LDA WARMST ; IS IT WARMSTART?

BNE ZOSRAM ; YES, ONLY ZERO OS RA
                               ; DO HARDWARE INITIALIZATION
                                    ;YES, ONLY ZERO OS RAM
ZERORM: LDA #0
           LDY
                  #WARMST
           STA
                   RAMLO
           STA RAMLO+1 ; INITIALIZE RAM POINTER
CLRRAM: STA (RAMLO), Y ; CLEAR MEMORY LOC.
           INY
                  #0
           CPY
                                     ;AT END OF PAGE?
           BNE CLRRAM
INC RAMLO+1 ;YES. INCR PAGE POINTER
LDX RAMLO+1
CPX TRAMSZ ;AT END OF MEM?
BNE CLRRAM ;NO.
; INITIALIZE DOSVEC TO POINT TO SIGNON (BLACKBOARD)
           LDA BLKBDV+1
STA DOSVEC ; USE BLACKBOARD VECTOR
LDA BLKBDV+2 ; FOR DOSVEC
STA DOSVEC+1
LDA #$FF
STA COLDST ; SET TO SHOW IN MIDDLE
BNE ESTSCM ; GO AROUND ZOSRAM
                                   ; SET TO SHOW IN MIDDLE OF COLDSTART
; CLEAR OS RAM (FOR WARMSTART)
ZOSRAM: LDX #0
          TXA
ZOSRM2: STA $200,X ;CLEAR PAGES 2 AND 3
           STA $300,X
           DEX
           BNE
                   ZOSRM2
           LDX #INTZBS
ZOSRM3: STA 0,X ;CLEAR ZERO PAGE LOCATIONS INTZBS-7F
           INX
           BPL ZOSRM3
; ESTABLISH SCREEN MARGINS
ESTSCM: LDA #LEDGE
           STA LMARGN
           LDA
                   #REDGE
           STA RMARGN
;
; MOVE VECTOR TABLE FROM ROM TO RAM
OPSYS: LDX #$25
MOVVEC: LDA VCTABL, X ; ROM TABLE
           STA INTABS, X ; TO RAM
           DEX
           BPL MOVVEC
```

;

```
JSR
               OSRAM ;DO O.S. RAM SETUP
         CLI
                                ; ENABLE IRO INTERRUPTS
       LINK HANDLERS
        LDX
                #TBLLEN
NXTENT: LDA TBLENT,X ; READ HANDLER TABLE ENTRY
         STA HATABS, X ; PUT IN TABLE
         DEX
         BPL NXTENT ; DONE WITH ALL ENTRIES?
; INTERROGATE CARTRIDGE ADDR. SPACE TO SEE WHICH CARTRIDGES THERE ARE
         LDX
                #0
         STX
                TSTDAT
                             ;CLEAR "B" CART. FLAG
         STX
               TRAMSZ
                             ;CLEAR "A" CART. FLAG
              RAMSIZ
#$90
         LDX
         CPX
                           ; RAM IN "B" CART. SLOT?
               ENDBCK
CART-$2000 ;NO.
ENDBCK ;C.
         BCS
         LDA
                            ; CART. PLUGGED INTO "B" SLOT'?
         BNE
         INC
               TSTDAT
                             ;YES, SET "B" CART, FLAG
         JSR CBINI ; INITIALIZE CARTRIDGE "B"
ENDBCK: LDX RAMSIZ
         CPX #$B0
                           ; RAM IN "A" CART. SLOT?
         BCS ENDACK
LDX CART
BNE ENDACK
                           ;NO,
                           ; CART. PLUGGED INTO "A" SLOT?
                             ; YES, SET "A" CART, FLAG
         INC
               TRAMSZ
                         ; INITIALIZE CARTRIDGE "A"
         JSR
               CAINI
; OPEN SCREEN EDITOR
ENDACK: LDA #3
         LDX
               #SEX
         STA
                            ;OPEN I/O COMMAND
               ICCOM,X
         LDA
                #OPNL
               ICBAL,X
         STA
         LDA
               #OPNH
               ICBAH,X
         STA
                           ;SET BUFFER POINTER TO OPEN SCREEN EDITOR
         LDA
               #$C
         STA ICAX1,X
JSR CIOV
                           ;SET UP OPEN FOR INPUT/OUTPUT
                           ;GO TO CIO
         BPL
               SCRNOK
                             ;BR IF NO ERROR
         JMP
               PWRUP
                          ; RETRY PWRUP IF ERROR (SHOULD NEVER HAPPEN!)
                           ;SCREEN OK, SO WAIT FOR YBLANK TO
SCRNOK: INX
         BNE SCRNOK
                             BRING UP THE DISPLAY
         INY
             SCRNOK
         BPL
```

```
JSR CSBOOT
                       ; CHECK, BOOT, AND INIT
; CHECK TO SEE IF EITHER CARTRIDGE WANTS DISK BOOT
            TRAMSZ
                           ; CHECK BOTH CARTRIDGES
        LDA
        ORA
              TSTDAT
                           ; NEITHER CARTRIDGE LIVES
              NOCART
        BEO
        LDA
                       ; "
; NO
; G
              TRAMSZ
                           ;"A" CART?
        BEQ
              NOA1
        LDA CARTFG
                           GET CARTRIDGE MODE FLAG
NOA1: LDX TSTDAT ; "B" CART?
       BEQ NOB1
                         ; NO
        ORA
               CARTFG-$2000 ; ADD OTHER FLAG
      AND #1
                        ; DOES EITHER CART WANT BOOT?
NOB1:
       BEQ NOBOOT
                           ; NO
; DO DISK BOOT
NOCART: JSR BOOT ; CHECK. BOOT. AND INIT
; GO TO ONE OF THE CARTRIDGES IF THEY SO DESIRE
NOBOOT: LDA #0
        STA COLDST ;RESET TO SHOW DONE WITH COLDSTART LDA TRAMSZ ;"A" CART?
        BEQ NOA2
LDA CARTFG
                         ;NO
                         GET CARTRIDGE MODE FLAG
        AND
              #4
                           ; DOES IT WANT TO RUN?
                        ;NO
              NOA2
        BEQ
        JMP (CARTCS) ; RUN "A" CARTRIDGE
     LDA TSTDAT ; "B" CART?
NOA2:
        BEQ NOCAR2
                            ; NO
        LDA
              CARTFG-$2000 ;GET "B" MODE FLAG
        AND
                            ; DOES IT WANT TO RUN?
              NOCART ; NO
        BEQ
             (CARTCS-$2000) ; RUN "B" CARTRIDGE
        JMP
; NO CARTRIDGES, OR NEITHER WANTS TO RUNS
; SO GO TO DOSVEC (DOS, CASSETTE, OR BLACKBOARD)
NOCAR2: JMP (DOSVEC)
; PRINT SIGN-ON MESSAGE
SIGNON: LDX #IDENTL
        LDY #IDENTH
        JSR PUTLIN
                           ;GO PUT SIGN-ON MSG ON SCREEN
       BLACKBOARD ROUTINE
BLACKB: JSR BLKB2 ; "JSR EGETCH"
       JMP BLACKB ; FOREVER
BLKB2: LDA EDITRV+5 ; HIGH BYTE
        PHA
        LDA
              EDITRV+4 ;LOW BYTE
        PHA
        RTS
                              ;SIMULATES "JMP (EDITRV)"
; CARTRIDGE INITIALIZATION INDIRECT JUMPS
CAINI: JMP (CARTAD)
CBINI: JMP (CARTAD-$2000)
```

; DO CASSETTE BOOT

```
.PAGE
                   SUBROUTINES
; CHECK FOR HOW MUCH RAM & SPECIAL CARTRIDGE CASE.
; IF SPECIAL CARTRIDGE CASE, DON'T GO BACK -- GO TO CART.
SPECL: LDA CART
                        ; CHECK FOR RAM OR CART
         BNE ENSPE2
                            GO IF NOTHING OR MAYBE RAM
         INC
                CART
                            ; NOW DO RAM CHECK
                CART
                            ; IS IT ROM?
         LDA
                ENSPEC
         BNE
                              ; NO
         LDA
                CARTFG
                              ;YES,
         \mathtt{BPL}
                ENSPEC
                              ;BIT SET?
                 (CARTAD) ; YES, GO RUN CARTRIDGE
         JMP
; CHECK FOR AMOUNT OF RAM
ENSPEC: DEC CART
                        ; RESTORE RAM IF NEEDED
ENSPE2: LDY #0
         STY
                RAMLO+1
                 #$10
         LDA
         STA
                 TRAMSZ
                              ;SET RAM POINTER TO 4K.
HOWMCH: LDA (RAMLO+1), Y ; READ RAM LOCATION
         EOR
                #$FF
                       ; INVERT IT.
         STA
                 (RAMLO+1), Y ; WRITE INVERTED DATA.
                 (RAMLO+1),Y ; READ RAM AGAIN
         CMP
         BNE
                ENDRAM
         EOR
                #$FF
                            ; CONVERT IT BACK
                 (RAMLO+1), Y ; RESTORE ORIGINAL RAMDATA
         STA
                 TRAMSZ
         LDA
         CLC
         ADC
                #$10
                              ; INCR. RAM POINTER BY 4K.
         STA
                TRAMSZ
```

ENDRAM: RTS

BNE

HOWMCH

;GO FIND HOW MUCH RAM.

```
HARDWARE INITIALIZATION
HARDI: LDA
           #0
        TAX
CLRCHP: STA $D000,X
         STA
              $D400,X
         STA
               $D200,X
         STA
                $D300,X
         INX
         BNE
               CLRCHP
         RTS
       O.S. RAM SETUP
OSRAM: DEC BRKKEY
                      ;TURN OFF BREAK KEY FLAG
         LDA
               #.LOW.BRKKY2
         STA
               BRKKY
         LDA
               #.HIGH.BRKKY2
         STA
               BRKKY+1
                             ; READ RAM SIZE IN TEMP. REG.
         LDA
               TRAMSZ
         STA
               RAMSIZ
                             ; SAVE IT IN RAM SIZE.
               MEMTOP+1 ; INIT. MEMTOP ADDR HI BYTE
         STA
         LDA
               #0
               MEMTOP
         STA
                             ; INIT. MEMTOP ADDR LO BYTE
         LDA
               #INIMLL
         STA
               MEMLO
               #INIMLH
         LDA
                           ; INITIALIZE MEMLO ADDR VECTOR
         STA
               MEMLO+1
               EDITRV+$C ; EDITOR INIT.
         JSR
                SCRENV+$C ; SCREEN INIT.
         JSR
         JSR
               KEYBDV+$C ; KEYBOARD INIT.
         JSR
               PRINTV+$C ; PRINTER HANDLER INIT
               CASETV+$C ; CASSETTE HANDLER INIT
         JSR
         JSR
               CIOINV
                             ;CIO INIT.
         JSR
               SIOINV
                             ;SIO INIT.
         JSR
               INTINV
                             ; INTERRUPT HANDLER INIT.
         LDA
               CONSOL
         AND
               #$1
                        GAME START KEY DEPRESSED?
         BNE
               NOKEY
         INC
               CKEY
                          ;YES. SET KEY FLAG.
NOKEY: RTS
; DO BOOT OF DISK
BOOT:
       LDA WARMST
         BEQ NOWARM
                             ; WARMSTART?
         LDA
               BOOT?
                         ;YES,
         AND
                #1
         BEQ
                NOINIT
                             ; VALID BOOT?
         JSR
               DINI
                          ;YES, RE-INIT. DOS SOFTWARE
NOINIT: RTS
NOWARM: LDA #1
```

```
STA
                DUNIT
                           ; ASSIGN DISK DRIVE NO.
                 #STATC
         LDA
         STA
                DCOMND
                              ;SET UPSTATUS COMMAND
         JSR DSKINV
BPL DOBOOT
                              ;GO DO DISK STATUS
                              ; IS STATUS FROM 510 GOOD?
                                ; NO, GO BACK WITH BAD BOOT STATUS
         RTS
DOBOOT: LDA #0
         STA DAUX2
         LDA
                #1
         STA
                         ;SET SECTOR # TO 1.
               DAUX1
         LDA
                #BUFFL
                DBUFLO
         STA
         LDA
                #BUFFH
         STA DBUFHI
                             ;SET UP BUFFER ADDR
SECT1: JSR GETSEC ;GET SECTOR

BPL ALLSEC ;STATUS O.K.?

BADDSK: JSR DSKRDE ;NO, GO PRINT DISK READ ERROR
         LDA CASSBT
         BEQ
                DOBOOT
                              ; CASSETTE BOOT?
         RTS
                                 ;YES, QUIT
ALLSEC: LDX #3
RDBYTE: LDA CASBUF+3,X ; READ A BUFFER BYTE
                DFLAGS,X ;STORE IT
         STA
         DEX
               RDBYTE
BOOTAD
RAMLO
BOOTAD+1
         BPL
                              ; DONE WITH 4 BYTE TRANSFER
         LDA
                              ;YES.
         STA
         LDA
               RAMLO+1
CASBUF+7
         STA
                           ; PUT BOOT ADDR INTO Z. PAGE RAM
         LDA
         STA
                DOSINI
                            ;ESTABLISH DOS INIT ADDRESS
         LDA
                CASBUF+8
         STA DOSINI+1
MVBUFF: LDY #$7F ;YES, SET BYTE COUNT
MVNXB: LDA CASBUF+3,Y
         STA (RAMLO), Y ; MOVE A BYTE FROM SECTOR BUFFER TO BOOT ADDR
         DEY
         BPL MVNXB
                         ;DONE ?
         CLC
                               ;YES,
               RAMLO
         LDA
         ADC
                #$80
               RAMLO
RAMLO+1
         STA
         LDA
               #0
RAMLO+1 ;INCR BOOT LOADER BUFFER POINTER
         ADC
         STA
                DBSECT
                             ;DECR # OF SECTORS.
         DEC
         BEQ
                ENBOOT
                              ; MORE SECTORS ?
INC DAUX1 ;YES INCR SECTOR #

SECTX: JSR GETSEC ;GO GET SECTOR.

BPL MVBUFF ;STATUS O.K. ?
         JSR
                DSKRDE
                               ; NO, GO PRINT DISK READ ERROR
                 CASSBT
         LDA
         BNE
                              ; IF CASSETTE, QUIT.
                BADDSK
                SECTX ; IF DISK, TRY SECTOR AGAIN.
         BEO
ENBOOT: LDA CASSBT
         BEQ XBOOT ; A CASSETTE BOOT ?
JSR GETSEC ; YES, GET EOF R
                            ; YES, GET EOF RECORD, BUT DON'T USE IT.
XBOOT: JSR BLOAD ;GO EXECUVE BOOT LOADER
```

```
BCS
              BADDSK
                            ; IF BAD BOOT, DO IT OVER AGAIN
         DINI
INC POT
                          ;GO INIT. SOFTWARE
               BOOT?
                        ;SHOW BOOT SUCCESS
         RTS
BLOAD: CLC
         LDA
              BOOTAD
         ADC
               #6
               RAMLO
         STA
         LDA
               BOOTAD+1
         ADC
               #0
               RAMLO+1 ; PUT START ADDR OF BOOTLOADER INTO RAM
         STA
        JMP
               (RAMLO)
DINI: JMP (DOSINI)
; DISPLAY DISK READ ERROR MSG
DSKRDE: LDX #DERRL
        LDY #DERRH
; PUT LINE ON SCREEN AT PRESENT CURSOR POSITION
   X-REG -- LO BYTE, BEGIN ADDR OF LINE
   Y-REG -- HI BYTE, BEGIN ADDR OF LINE
PUTLIN: TXA
         LDX
               #SEX
         STA
               ICBAL,X
         TYA
         STA ICBAH,X ;SET UP ADDR OF BEGIN OF LINE
         LDA
               #PUTTXT
              ICCOM,X
         STA
                          ; "PUT TEXT RECORD" COMMAND
         LDA
               #$FF
             ICBLL,X ;SET BUFFER LENGTH
CIOV ;PUT LINE ON SCREEN
         STA
         JSR
                           ; PUT LINE ON SCREEN
         RTS
; GET SECTOR FROM DISK 0
GETSEC: LDA CASSBT
         BEQ DISKM ; CASSETTE BOOT?
         JMP RBLOKV
                            ; YES, GO TO READ BLOCK ROUTINE
DISKM: LDA #READ
                           ;SET READ SECTOR COMMAND
         STA DCOMND
         LDA
               #1
         STA
JSR
               DUNIT
                        ;SET DRIVE NO. TO DRIVE 0
               DSKINV
                            GET SECTOR
         RTS
```

; DO CHECK FOR CASSETTE BOOT & IF SO DO BOOT

```
CSBOOT: LDA WARMST
                              ; WARMSTART?
           BEQ CSBOT2
                                      ; NO
            LDA BOOT? ;GET BOOT FLAG
AND #2 ;WAS CASSET
BEQ NOCSB2 ;NO
                                      ; WAS CASSETTE BOOT SUCCESFULL?
                    CINI
                                  ;YES, INIT CASSETTE SOFTWARE
            JSR
NOCSB2: RTS
CSBOT2: LDA CKEY
            BEQ NOCSBT ; "C" KEY FLAG SET ? LDA #$80 ; YES,
           LDA #$80 ;YES,

STA FTYPE ;SET LONG IRQ TYPE

INC CASSBT ;SET CASSETTE BOOT FLAG

JSR CSOPIV ;OPEN CASSETTE FOR INPUT

JSR SECT1 ;DO BOOT & INIT.

LDA #0

STA CASSBT ;RESET CASSETTE BOOT FLAG

STA CKEY ;CLEAR KEY FLAG

ASL BOOT? ;SHIFT BOOT FLAG (NOW=2 IF SUCCESS)

LDA DOSINI

STA CASINI ;MOVE INIT ADDRESS FOR CASSETTE

LDA DOSINI+1

STA CASINI+1
                    CASINI+1
            STA
NOCSBT: RTS
CINI: JMP (CASINI) ; INIT CASSETTE
; SPARE BYTE OR MODULE TOO LONG FLAG
CRNTP7 =*
           *=$14
MONSPR: .BYTE KBDORG-CRNTP7 ; GMONITP TOO LONG
            .PAGE
            .TITLE 'DISPLAY HANDLER -- 10-30-78 -- DISPLC'
; HANDLER DEPENDENT EQUATES
                  $7D
$9F
                                     ;CLEAR SCREEN ATASCI CODE
CLRCOD =
CNTL1 =
                                       ; POKEY KEY CODE FOR ^1
FRMADR =
                  SAVADR
TOADR =
                   MLTTMP
            .PAGE
            *=EDITRV
; SCREEN EDITOR HANDLER ENTRY POINT
EDITOR: .WORD EOPEN-1
            .WORD RETUR1-1 ; (CLOSE)
            .WORD EGETCH-1
            .WORD EOUTCH-1
```

```
.WORD RETUR1-1 ;(STATUS)
           .WORD NOFUNC-1
                            ;(SPECIAL)
          JMP PWRONA
          .BYTE 0
                              ;ROM FILLER BYTE
;
          *=SCRENV
; DISPLAY HANDLER ENTRY POINT
DISPLA: .WORD DOPEN-1
          .WORD RETUR1-1 ; (CLOSE)
          .WORD GETCH-1
           .WORD OUTCH-1
          .WORD RETUR1-1 ; (STATUS)
.WORD DRAW-1 ; (SPECIAL)
JMP PWRONA
          .BYTE 0
                              ;ROM FILLER BYTE
; KEYBOARD HANDLER ENTRY POINT
KBDHND: .WORD RETUR1-1
          .WORD RETUR1-1 ; (CLOSE)
          .WORD KGETCH-1
          .WORD NOFUNC-1 ;(OUTCH)
                             ;(STATUS);(SPECIAL)
          .WORD RETUR1-1
          .WORD NOFUNC-1
          JMP PWRONA
          .BYTE 0
                              ;ROM FILLER BYTE
;
; INTERRUPT VECTOR TABLE ENTRY
          *=VCTABL-INTABS+VKEYBD
          .WORD PIRQ5 ;KEYBOARD IRQ INTERRUPT VECTOR
          *=KBDORG
PWRONA: LDA #$FF
          STA CH
          LDA MEMTOP+1
AND #$F0 ;INSURE 4K PAGE BOUNDARY
STA RAMTOP
LDA #$40 ;DEFAULT TO UPPER CASE AI
STA SHFLOK
                           ;DEFAULT TO UPPER CASE ALPHA AT PWRON
          RTS
                                   ; POWER ON COMPLETED
          .PAGE
; BEGIN DISPLAY HANDLER OPEN PROCESSING
DOPEN: LDA ICAX2Z
                            GET AUX 2 BYTE
         AND #$F
         BNE OPNCOM ; IF MODE ZERO, CLEAR ICAX1Z

DA ICAX1Z ; CLEAR "CLR INHIBIT" AND "MXD MODE" BITS
EOPEN: LDA ICAX1Z
          AND #$F
          STA
                 ICAX1Z
          LDA
                 #0
```

```
OPNCOM: STA DINDEX
         LDA #$E0
                          ; INITIALIZE GLOBAL VBLANK RAM
               CHBAS
         STA
         LDA
               #2
              CHACT
SDMCTL
         STA
                             ;TURN OFF DMA NEXT VBLANK
         STA
         LDA
               #SUCCES
         STA DSTAT
LDA #$CO
ORA POKMSK
STA POKMSK
                          ;CLEAR STATUS
                           ;DO IRQEN
         STA
               IRQEN
         LDA
               #0
         STA
               TINDEX
                             ;TEXT INDEX MUST ALWAYS BE 0
               ADRESS
         STA
               SWPFLG
         STA
                             ;TURN CURSOR ON ATOPEN
         STA
               CRSINH
         LDY
               #14
                             CLEAR TAB STOPS
         LDA
                #1
                              ; INIT TAB STOPS TO EVERY 8 CHARACTERS
CLRTBS: STA TABMAP, Y
         DEY
         BPL
               CLRTBS
         LDX
                #4
                             ;LOAD COLOR REGISTERS
DOPEN8: LDA COLRTB,X
         STA COLORO, X
         DEX
         BPL DOPEN8
LDY RAMTOP ;DO TXTMSC=$2C40 (IF MEMTOP=3000)
         DEY
               TXTMSC+1
         STY
               #$60
         LDA
         STA
               TXTMSC
         LDX
               DINDEX
              ANCONV,X ; CONVERT IT TO ANTIC CODE DOPENA ; IF ZERO, IT IS ILLEGAL
         LDA
         BNE
OPNERR: LDA #BADMOD ;SET ERROR STATUS
         STA
               DSTAT
DOPENA: STA HOLD1
         LDA RAMTOP ;SET UP AN INDIRECT POINTER
         STA
               ADRESS+1
         LDY
               ALOCAT, X ; ALLOCATE N BLOCKS OF 40 BYTES
DOPEN1: LDA #40
         JSR DBSUB
         DEY
             DOPEN1
         BNE
         LDA
               GPRIOR
                            ; ICLEAR GTIA MODES
         AND
               #$3F
         STA
               OPNTMP+1
         TAY
         CPX #8
BCC NOT8
                             ;TEST IF 320X1
         TXA
                               GET 2 LOW BITS
               Α
         ROR
         ROR
               Α
         ROR
               Α
         AND
               #$C0
                          ;NOW 2 TOP BITS
         ORA
               OPNTMP+1
         TAY
         LDA
               #16
                             ;SUBTRACT 16 MORE FOR PAGE BOUNDARY
```

```
JSR
               DBSUB
         CPX
                #11
                             ;TEST MODE 11
                          ; IF MODE = 11
               NOT8
         BNE
         LDA
               #6
                             ; PUT GTIA LUM VALUE INTO BACKGROUND REGISTER
                COLOR4
         STA
     STY GPRIOR
                        ;STORE NEW PRIORITY
NOT8:
              ADRESS
                            ; SAVE MEMORY SCAN COUNTER ADDRESS
         LDA
         STA
               SAVMSC
         LDA
               ADRESS+1
         STA SAVMSC+1
VBWAIT: LDA VCOUNT ; WAIT FOR NEXT VBLANK BEFORE MESSING
         CMP
              #$7A
                          ; WITH THE DISPLAY LIST
                VBWAIT
         BNE
         JSR
               DBDEC
                         ;START PUTTING DISPLAY LIST RIGHT UNDER RAM
              PAGETB,X ;TEST IF DISPLAY LIST WILL BE IN TROUBLE NOMOD ;OF CROSSING A 256 BYTE PAGE BOUNDARY
         LDA
         BEO
                          ; IF SO, DROP DOWN A PAGE
         LDA
               #$FF
         STA
               ADRESS
         DEC
               ADRESS+1
NOMOD: LDA ADRESS
                        ; SAVE END OF DISPLAY LIST FOR LATER
         STA SAVADR
               ADRESS+1
         LDA
         STA
               SAVADR+1
               DBDDEC
                             ; (DOUBLE BYTE DOUBLE DECREMENT)
         JSR
               #$41
                          ;(ANTIC) WAIT FOR VBLANK AND JMP TO TOP
         LDA
         JSR
               STORE
              OPNTMP
         STX
               #24
         LDA
                            ; INITIALIZE BOTSCR
         STA
               BOTSCR
         LDA
               DINDEX
                            ; DISALLOW MIXED MODE IF MODE.GE.9
         CMP
               #9
         BCS
               NOTMXD
              ICAX1Z
                          ;TEST MIXED MODE
         LDA
         AND
               #$10
               NOTMXD
         BEO
         LDA
                #4
         STA
               BOTSCR
                             ; ADD 4 LINES OF TEXT AT BOTTOM OF SCREEN
         LDX
               #2
DOPEN2: LDA #2
              STORE
         JSR
         DEX
               DOPEN2
         BPL
         LDY
               RAMTOP
                            ; RELOAD MSC FOR TEXT
         DEY
         TYA
              STORE
         JSR
         LDA
               #$60
         JSR
               STORE
         LDA
               #$42
             STORE
         JSR
         CLC
         LDA
               #MXDMDE-NUMDLE ; POINT X AT MIXED MODE TABLE
         ADC
                OPNTMP
         STA
                OPNTMP
NOTMXD: LDY OPNTMP
         LDX
                NUMDLE, Y GET NUMBER OF DISPLAY LIST ENTRIES
DOPEN3: LDA HOLD1 ;STORE N DLE'S
         JSR
               STORE
```

DEX

```
BNE
               DOPEN3
         LDA
                DINDEX
                             ; DO THE MESSY 320X1 PROBLEM
         CMP
                #8
         BCC
               DOPEN5
               #93
                             GET REMAINING NUMBER OF DLE'S
         LDX
                            ; RELOAD MEMORY SCAN COUNTER
         LDA
               RAMTOP
         SEC
               #$10
         SBC
         JSR
               STORE
         LDA
               #0
         JSR
               STORE
         LDA
               #$4F
                           ; (ANTIC) RELOAD MSC CODE
         JSR
                STORE
DOPEN4: LDA HOLD1 ; DO REMAINING DLE'S
         JSR STORE
         DEX
         BNE
               DOPEN4
DOPEN5: LDA SAVMSC+1 ; POLISH OFF DISPLAY LIST
         JSR
              STORE
         LDA
               SAVMSC
         JSR
               STORE
              HOLD1
         LDA
         ORA
               #$40
         JSR
               STORE
         LDA
               #$70
                          ;24 BLANK LINES
         JSR
               STORE
         LDA
               #$70
              STORE
         JSR
         LDA
               ADRESS
                            ; SAVE DISPLAY LIST ADDRESS
         STA
               SDLSTL
         LDA
               ADRESS+1
         STA
               SDLSTL+1
              #$70
STORE
                          ; ADD LAST BLANK LINE ENTRY
         LDA
                         ; POSITION ADRESS=SDLSTL-1
         JSR
                            ;STORE NEW MEMTOP
         LDA
               ADRESS
         STA
               MEMTOP
         LDA
               ADRESS+1
         STA
               MEMTOP+1
              SAVADR
         LDA
         STA
               ADRESS
         LDA
               SAVADR+1
         STA
               ADRESS+1
         LDA
               SDLSTL+1
         JSR
               STORE
               SDLSTL
         LDA
         JSR
               STORE
         LDA
               DSTAT
                          ; IF ERROR OCURRED ON ALLOCATION, OPEN THE ED
               DOPEN9
         BPL
         PHA
                               ; SAVE STATUS
         JSR
             EOPEN
                          ;OPEN THE EDITOR
         PLA
                               ; RESTORE STATUS
         TAY
                               ; AND RETURN IT TO CIO
         RTS
DOPEN9: LDA ICAX1Z
                         ;TEST CLEAR INHIBIT BIT
         AND
               #$20
         BNE
                DOPEN7
         JSR
                             ;CLEAR SCREEN
               CLRSCR
         STA
               TXTROW
                             ; AND HOME TEXT CURSOR (AC IS ZERO)
         LDA
                LMARGN
```

```
STA TXTCOL
DOPEN7: LDA #$22 ; EVERYTHING ELSE IS SET UP
        ORA SDMCTL ;SO TURN ON DMACTL
        STA
              SDMCTL
        JMP RETUR2
GETCH: JSR RANGE ;GETCH DOES INCRSR. GETPLT DOESN'T
        JSR GETPLT
        JSR
              INATAC ; CONVERT INTERNAL CODE TO ATASCII
        JSR
              INCRSB
        JMP
              RETUR1
GETPLT: JSR CONVRT ; CONVERT ROW/COLUMN TO ADRESS
        LDA (ADRESS),Y
        AND DMASK
SHIFTD: LSR SHFAMT ;SHIFT DATA DOWN TO LOW BITS
        BCS SHIFT1
        LSR
              Α
        BPL
              SHIFTD
                          ; (UNCONDITIONAL)
SHIFT1: STA CHAR
        CMP #0
                      ; RESTORE FLAGS ALSO
        RTS
;
;
OUTCH: STA ATACHR
       JSR RANGE
       JSR
              OFFCRS
OUTCHA: LDA ATACHR
                       ;TEST FOR CLEAR SCREEN
        CMP
             #CLRCOD
        BNE
              OUTCHE
        JSR
              CLRSCR
              RETUR2
        JMP
                      ;TEST FOR CARRIAGE RETURN
OUTCHE: LDA ATACHR
        CMP
             #CR
              OUTCHB
        BNE
                          ;DO CR
        JSR
              DOCRWS
        JMP
              RETUR2
OUTCHB: JSR OUTPLT
        JSR INCRSR
        JMP
              RETUR2
;
OUTPLT: LDA SSFLAG ;*****LOOP HERE IF START/STOP FLAG ISNON-0
        BNE OUTPLT
        LDX
              #2
CRLOOP: LDA ROWCRS,X ;SAVE CURSOR LOCATION FOR DRAW LINE TO DRAW
        STA OLDROW, X
        DEX
        BPL CRLOOP
LDA ATACHR
                          ; CONVERT ATASCII (ATACHR) TO INTERNAL (CHAR)
        TAY
                            ;SAVE ATACHR
              Α
        ROL
        ROL
              Α
        ROL
              Α
        ROL
              #3
        AND
        TAX
                            ;X HAS INDEX INTO ATAINT
        TYA
                             ; RESTORE ATACHR
```

```
AND #$9F ;STRIP OFF COLUMN ADDRESS
               ATAINT,X ;OR IN NEW COLUMN ADDRESS
         ORA
OUTCH2: STA CHAR
         JSR CONVRT
         LDA
                CHAR
SHIFTU: LSR SHFAMT
                         ;SHIFT UP TO PROPER POSITION
         BCS SHIFT2
         ASL
                Α
         JMP
               SHIFTU
SHIFT2: AND DMASK
         STA TMPCHR ;SAVE SHIFTED DATA
         LDA
               DMASK
                          ; INVERT MASK
         EOR #$FF
AND (ADRESS),Y ;MASK OFF OLD DATA
ORA TMPCHR ;OR IN NEW DATA
STA (ADRESS),Y
         RTS
;
RETUR2: JSR GETPLT ; DO CURSOR ON THE WAY OUT
         STA OLDCHR
         LDX
               DINDEX ;GRAPHICS HAVE INVISIBLE CURSOR
         BNE
               RETUR1
               CRSINH
                             ;TEST CURSOR INHIBIT
         LDX
         BNE
               RETUR1
         EOR
               #$80
                           ;TOGGLE MSB
         JSR OUTCH2
                             ;DISPLAY IT
RETUR1: LDY DSTAT ; RETURN TO CIO WITH STATUS IN Y
         LDA #SUCCES
         STA
               DSTAT ;SET STATUS= SUCCESSFUL COMPLETION
ATACHR ;PUT ATACHR IN AC FOR RETURN TO
         LDA
                            ; PUT ATACHR IN AC FOR RETURN TO CIO
NOFUNC: RTS
                          ; (NON-EXISTENT FUNCTION RETURN POINT)
; END OF DISPLAY HANDLER
         .PAGE
EGETCH: JSR SWAP
         JSR ERANGE
                            ; ANYTHING IN THE BUFFER?
         LDA
               BUFCNT
         BNE
               EGETC3
                             ;YES
               ROWCRS
                             ; NO, SO SAVE BUFFER START ADDRESS
         LDA
         STA
               BUFSTR
         LDA
               COLCRS
         STA BUFSTR+1
EGETC1: JSR KGETCH ;LET'S FILL OUR BUFFER
         STY DSTAT
                         ;SAVE KEYBOARD STATUS
               ATACHR
         LDA
                            ;TEST FOR CR
         CMP
               #CR
              EGETC2
DOSS
SWAP
         BEQ
                          ;NO, GO PRINT IT ;JSR DOSS DID SWAP SO SWAP BACK
         JSR
         JSR
         LDA
               LOGCOL
                           ;BEEP IF NEARING LOGICAL COL 120
         CMP
               #113
```

```
JSR
               BELL
EGETC6: JMP EGETC1
                      GET BUFFER COUNT
EGETC2: JSR OFFCRS
        JSR
               DOBUFC
        LDA
               BUFSTR
                           ; RETURN A CHARACTER
        STA
               ROWCRS
        LDA
               BUFSTR+1
        STA
               COLCRS
EGETC3: LDA BUFCNT
        BEQ
               EGETC5
EGETC7: DEC BUFCNT
                        ;AND RETURN TILL BUFCNT=0
               EGETC5
        BEQ
        LDA
              DSTAT
                        ; IF ERR, LOOP ON EGETC7 UNTIL BUFR IS E11PTIE
        BMI
              EGETC7
        JSR
               GETCH
        STA
              ATACHR
               SWAP
                      ;DO REAL CARRIAGE RETURN
        JMP
                          ; AND RETURN WITHOUT TURNING CURSOR BACK ON
EGETC5: JSR DOCRWS
                            ; AND RETURN EOL
        LDA #CR
        STA
              ATACHR
        JSR
              RETUR2
                            ;TURN ON CURSOR THEN SWAP
        STY
              DSTAT
                         ; SAVE KEYBOARD STATUS
        JMP
              SWAP
                          ; AND RETURN THROUGH RETUR1
JSRIND: JMP (ADRESS) ; J5R TO THIS CAUSES JSR INDIRECT
EOUTCH: STA ATACHR
                    ;SAVE ATASCII VALUE
        JSR SWAP
        JSR
               ERANGE
      JSR OFFCRS ;TURN OFF CURSOR JSR TSTCTL ;TEST FOR CON
DOSS:
                         TEST FOR CONTROL CHARACTERS (Z=1 IF CTL)
        BEQ
               EOUTC5
                        ; ESCFLG ONLY WORKS ONCE
EOUTC6: ASL ESCFLG
        JSR OUTCHE
ERETN: JMP SWAP ; AND RETURN THROUGH RETUR1
EOUTC5: LDA DSPFLG
                        ;DO DSPFLG AND ESCFLC
        ORA ESCFLG
              EOUTC6
        BNE
                            ; IF NON-0 DISPLAY RATHER THAN EXECUTE IT
              ESCFLG
        ASL
        INX
                              ; PROCESS CONTROL CHARACTERS
              CNTRLS,X ;GET DISPLACEMENT INTO ROUTINE
        LDA
        STA
              ADRESS
              CNTRLS+1,X ;GET HIGH BYTE ADRESS+1
        LDA
        STA
                           ;DO COMPUTED JSR
              JSRIND
        JSR
        JSR RETUR2 ;DO CURSOR
JMP SWAP ;ALL DONE SO RETURN THROUGH RETUR1
; END SCREEN EDITOR.
; BEGIN KEYBOARD HANDLER
```

BNE EGETC6

```
KGETC2: LDA #$FF
       STA CH
KGETCH: LDA ICAX1Z ;TEST LSB OF AUX1 FOR SPECIAL EDITOR READ MO
         LSR A
         BCS
               GETOUT
         LDA
               #BRKABT
              BRKKEY
K7
CH
#$FF
KGETCH
         LDX
                            ;TEST BREAK
         BEQ
                            ; IF BREAK, PUT BRKABT IN DSTAT AND CRIN ATA
         LDA
         CMP
         BEO
         STA
              HOLDCH
#$FF
                            ; SAVE CH FOR SHIFT LOCK PROC
                         ;"CLEAR" CH
         LDX
         STX
               CH
         JSR CLICK ; DO KEYBOARD AUDIO FEEDBACK (A IS OK)
                          ;DO ASCCON
KGETC3: TAX
              #$C0
         CPX
                         TEST FOR CTL & SHIFT TOGETHER
              ASCC01
         BCC
         LDX
               #3
                            ;BAD CODE
ASCCO1: LDA ATASCI,X
                         ;DONE;DO NULLS
         STA ATACHR
         CMP
               #$80
         BEQ KGETC2
CMP #$81
                         ; CHECK ATARI KEY
              KGETC1
INVFLG
         BNE
         LDA
               #$80
         EOR
         STA
              INVFLG
JMP KGETC2 ;DONT : KGETC1: CMP #$82 ;CAPS/LOWER.
                        ;DONT RETURN A VALUE
         BNE K1
               #0
                            ;CLEAR SHFLOK
         LDA
         STA SHFLOK
BEQ KGETC2
         STA
K1: CMP #$83 ;SHIFT CAPS/LOWER
         BNE K2
         LDA
               #$40
         STA SHFLOK ;SHIFT BIT BNE KGETC2
K2: CMP #$84 ;CNTL CAPS/LOWER
         BNE K3
               #$80
         LDA
                         ; CNTL BIT
               SHFLOK
         STA
         BNE KGETC2
     CMP #$85 ;DO EOF
K3:
         BNE K6
         LDA #EOFERR
K7:
     STA DSTAT
         STA BRKKEY ;RESTORE BREAK
GETOUT: LDA #CR
K6: LDA HOLDCH ;PROCESS SHIFT LOCKS

CMP #$40 ;REGULAR SHIFT AND C
BCS K5 ;OVER LOCK

LDA -
                        ; PUT CR IN ATACHR
                        ; REGULAR SHIFT AND CONTROL TAKE PRECEDENCE
              ATACHR ;TEST FOR ALPHA #$61 ;LOWER CASE A
         LDA
         лэо1
эсс K5
СМР
         CMP
                         ;NOT ALPHA IF LT
```

;LOWER CASE Z+1

#\$7B

;

```
BCS
              K5
                           ;NOT ALPHA IF GE
              SHFLOK
K5
                        ;DO SHIFT/CONTROL LOCK
        LDA
                           ; IF NO LOCK. DONT RE-DO IT
        BEQ
        ORA
              HOLDCH
        JMP KGETC3
                        ;DO RETRY
K5: JSR TSTCTL
                       ; DONT INVERT M58 OF CONTROL CHARACTERS
        BEO
              Κ4
        LDA
              ATACHR
        EOR INVFLG
   STA ATACHR
K8:
   JMP
K4:
          RETUR1 ; ALL DONE
;
        .PAGE
; CONTROL CHARACTER PROCESSORS
ESCAPE: LDA #$80
                     ;SET ESCAPE FLAG
        STA ESCFLG
        RTS
CRSRUP: DEC ROWCRS
        BPL COMRET
        LDX
              BOTSCR
                          ; WRAPAROUND
        DEX
UPDNCM: STX ROWCRS
COMRET: JMP STRBEG
                    ; CULVERT ROW AND COL TO LOGCOL AND RETURN
CRSRDN: INC ROWCRS
        LDA ROWCRS
        CMP
              BOTSCR
        BCC
              COMRET
        LDX
              #0
        BEQ UPDNCM
                     ; (UNCONDITIONAL)
CRSRLF: DEC COLCRS
        LDA COLCRS
                          ; (IF LMARGN=0, THIS ELIMINATES PROBLEM CASE)
        BMI
               CRSRL1
        CMP
              LMARGN
        BCS COMRE1
CRSRL1: LDA RMARGN
LFRTCM: STA COLCRS
COMRE1: JMP DOLCOL
                    COLVERT ROW AND COL TO LOGCOL AND RETURN
CRSRRT: INC COLCRS
        LDA COLCRS
        CMP
              RMARGN
              COMRE1
        BCC
        BEQ
              COMRE1
                          ; (CAUSE OLE)
        LDA
              LMARGN
              LFRTCM
        JMP
                          ;UNCONDITIONAL TO COMMON STORE
CLRSCR: JSR PUTMSC
        LDY #0
        TYA
                             ; PUT 0 IN THEAC
CLRSC2: STA (ADRESS),Y ;(AC IS ZERO)
        INY
             CLRSC2
        BNE
        INC
              ADRESS+1
             ADRESS+1
RAMTOP
CLRSC2
        LDX
        CPX
        BCC
        LDA
              #$FF
                        CLEAN UP LOGICAL LINE BITMAP
```

```
CLRSC3: STA LOGMAP, Y ; (Y IS ZERO AFTER CLRSC2 LOOP)
         INY
         CPY
               #4
         BCC CLRSC3
       JSR COLCR ; PLACE COLCRS AT LEFT EDGE
HOME:
         STA LOGCOL
               BUFSTR+1
         STA
         LDA
               #0
         STA ROWCRS
STA COLCRS+1
STA BUFSTR
         RTS
;
BS: LDA LOGCOL
                    ; BACKSPACE
        CMP LMARGN
               BS1
         BEO
                            ;LEFT EDGE?
BSA:
        LDA
               COLCRS
         CMP
               LMARGN
         BNE
               BS3
                             ; NO
         JSR
                             ;YES, SEE IF LINE SHOULD BE DELETED
               DELTIM
BS3:
       JSR
               CRSRLF
              COLCRS
         LDA
         CMP
               RMARGN
               BS2
         BNE
               ROWCRS
         LDA
         BEO
               BS2
              CRSRUP
#$20
ATACHR
        JSR
                          ; MAKE BACKSPACE DESTRUCTIVE
BS2: LDA
        STA
        JSR
               OUTPLT
                            ; AND RETURN
BS1:
       JMP
               DOLCOL
TAB:
       JSR
               CRSRRT
                            ;BEGIN SEARCH
              COLCRS
LMARGN
              ABI ;NO
DOCR ;DO CARRIAGE RETURN
LOGGET ;CHECK IF END ^-
TAB1 ;NO ~
                            ;TEST FOR NEW LINE
         LDA
        CMP
         BNE
         JSR
                            ; CHECK IF END OF LOGICAL LINE
         JSR
         BCC
              TAB2
         BCS
                           ; (UNCONDITIONAL)
                        ; CHECK FOR TAB STOP
TAB1: LDA LOGCOL
               BITGET
         JSR
         BCC
                TAB
                             ; NO, SO KEEP LOOKING
                       ; CULVERT ROW AND COL TO LOGCOL AND RETURN
TAB2: JMP DOLCOL
SETTAB: LDA LOGCOL
         JMP
               BITSET ;SET BIT IN MAP AND RETURN
CLRTAB: LDA LOGCOL
         JMP
                             ;CLEAR " " " " "
               BITCLR
INSCHR: JSR PHACRS
         JSR GETPLT
                            GET CHARACTER UNDER CURSOR
         STA
               INSDAT
               #0
         LDA
         STA
               SCRFLG
INSCH4: JSR OUTCH2
                         ;STORE DATA
         LDA LOGCOL
                             ; SAVE LOGCOL: IF AFTER INCRSA LOGCOL IS
                              ; < THAN IT IS NOW, END LOOP
         PHA
         JSR INCRSA
                             ;SPECIAL INCRSR ENTRY POINT
         PLA
         CMP
               LOGCOL
         BCS
               INSCH3
                             ;OUIT
```

```
INSCH1: LDA INSDAT ; KEEP GOING
        PHA
        JSR
              GETPLT
        STA
              INSDAT
        PLA
        JMP INSCH4
INSCH3: JSR PLACRS
INSCH6: DEC SCRFLG
        BMI INSCH5
                           ; IF SCROLL OCCURRED
        DEC
               ROWCRS
                            ; MOVE CURSOR UP
        BNE
               INSCH6
                            ; (UNCOND) CONTINUE UNTIL SCRFLG IS MINUS
INSCH5: JMP DOLCOL ; CULVERT ROW AND COL TO LOGCOL AND RETURN
DELCHR: JSR PHACRS
DELCH1: JSR CONVRT ;GET DATA TO THE RIGHT OF THE CURSOR
        LDA ADRESS
        STA
               SAVADR
                           ; SAVE ADRESS TO KNOW WHERE TO PUT DATA
        LDA
               ADRESS+1
        STA
              SAVADR+1
        LDA LOGCOL
        PHA
             INCRSB ; PUT CURSOR OVER NEXT CHARACTER
        JSR
        PLA
              LOGCOL
        CMP
                           ;TEST NEW LOGCOL AGAINST OLD LOGCOL
        BCS
              DELCH2
                           ; IF OLD.GE.NEW THEN QUIT
              ROWCRS
BOTSCR
DELCH2
        LDA
                            ; IS ROW OFF SCREEN?
        CMP
        BCS
                           ;YES, SO OUIT
              GETPLT
                           GET DATA UNDER CURSOR
        JSR
        LDY
               #0
               (SAVADR),Y ; PUT IT IN PREVIOUS POSITION
        STA
        BEQ DELCH1 ;AND LOOP (UNCONDITIONAL)
DELCH2: LDY #0
        TYA
               (SAVADR),Y ;CLEAR THE LAST POSITION
         STA
               DELTIA
                           TRY TO DELETE A LINE
        JSR
        JSR
              PLACRS
        JMP DOLCOL
                           ; AND RETURN
                         ; INSLIN PUTS "1" INTO BIT MAP
INSLIN: SEC
INSLIA: JSR EXTEND ;ENTRY POINT FOR C=0
        LDA LMARGN
                           ; DO CARRIAGE RETURN (NO LF)
        STA
              COLCRS
              COLCRS
CONVRT
ADRESS
                           GET ADDRESS
        JSR
        LDA
                           ;SET UP TO=40+FROM (FROM = CURSOR)
        STA
              FRMADR
        CLC
        ADC
              #40
              TOADR
ADRESS+1
FRMADR+1
        STA
        LDA
        STA
        ADC
               #0
               TOADR+1
        STA
        LDX
                           ;SET UP LOOP COUNTER
              ROWCRS
        CPX
               #23
        BEQ
                INSLI2
INSLI1: JSR MOVLIN
        INX
        CPX
              #23
```

```
INSLI1
INSLI2: JSR CLRLIN ;CLEAR CURRENT LINE

JMP DOLCOL ;COLVERT ROW AND COL TO LOGCOL AND RETURN

DELLIN: JSR DOLCOL ;GET BEGINNING OF LOG LINE (HOLD1)
DELLIA: LDY HOLD1 ;SQUEEZE BIT MAP
       STY ROWCRS ; PUT CURSOR THERE
DELLIB: LDY ROWCRS
DELLI1: TYA
         SEC
         JSR LO2GET ;GET NEXT BIT
         PHP
         TYA
         CLC
         ADC
               #120
         PLP
         JSR BITPUT ; WRITE IT OVER PRESENT BIT
         INY
         CPY
               #24
              DELLI1 ;LOOP
LOGMAP+2 ;SET LSB
         BNE
         LDA
         ORA
               #1
         STA LOGMAP+2
DELLI2: LDA LMARGN ; DELETE LINE OF DATA USING PART OF SCROLL
              COLCRS
         STA
                            ;CR NO LF
         JSR
               CONVRT
         JSR
               SCROL1
         JSR LOGGET
                          ;TEST NEXT LINE FOR CONTINUATION
; IS IT A NEW LOG LINE?
               DOPCOP
        BCC DELLIB
                            ; NO SO DELETE ANOTHER
         JMP
                             ;YES SO DOLCOL AND RETURN
BELL: LDY #$20
BELL1: JSR CLICK
        DEY
         BPL BELL1
        RTS
         .PAGE
; ROUTINES
; DOUBLE BYTE DECREMENT OF INDIRECT POINTER
; INCLUDING DB SUBTRACT AND DB DOUBLE DECREMENT
DBDDEC: LDA #2
       BNE DBSUB ; (UNCONDITIONAL)
; STORE DATA INDIRECT AND DECREMENT POINTER
; (PLACED HERE TO SAVE JMP DBDEC AFTER STORE)
STORE: LDY DSTAT ; RETURN ON ERROR
        BMI STROK
        LDY
               #0
STOREI: STA (ADRESS),Y
       JMP
             DBDEC ; DECREMENT AND RETURN
DBDEC: LDA #1
DBSUB: STA SUBTMP
        LDA DSTAT
                          ; RETURN ON ERROR
```

BNE

BMI STROK

```
LDA
               ADRESS
         SEC
         SBC
               SUBTMP
         STA
               ADRESS
         BCS
               DBSUB1
         DEC
               ADRESS+1
DBSUB1: LDA APPMHI+1 ; MARE SURE NOTHING EVER OVERWRITES APPMHI
         CMP
              ADRESS+1
               STROK ;OK
STRERR ;
         BCC
         BNE
                         ; ERROR
               APPMHI
         LDA
         CMP
               ADRESS
         BCC
               STROK
STRERR: LDA #SCRMEM ; SHOW MEM TOO SMALL FOR SCREEN ERROR
        STA DSTAT
STROK: RTS
; CONVERT ROW/COLUMN CURSOR INTO REAL ADDRESS (FROM SAVMSC ON UP)
CONVRT: LDA ROWCRS ; SAVE CURSOR
         PHA
               COLCRS
         LDA
         PHA
         LDA
               COLCRS+1
         PHA
         JSR PUTMSC
         LDA
               ROWCRS
                            ; PUT 10*ROWCRS INTO MLTTMP
         STA
               MLTTMP
         LDA
               #0
         STA
               MLTTMP+1
         LDA MLTTMP+1 ;QUICK X8
         ASL
               Α
         ROL
               MLTTMP+1
               HOLD1
                         ; (SAVE 2X VALUE)
         STA
               MLTTMP+1
         LDY
         STY
               HOLD2
         ASL
              MLTTMP+1
         ROL
         ASL
               Α
               MLTTMP+1
         ROL
                              ;ADD IN 2X
         CLC
         ADC HOLD1 STA MLTTMP
         LDA
               MLTTMP+1
               HOLD2
         ADC
               MLTTMP+1
         STA
         LDX DINDEX ; NOW SILDY DHLINE, X ; MULTIPLY
                            ; NOW SHIFT MLTTMP LEFT DHLINE TIMES TO FINIS
CONVR1: DEY
                          ;LOOP N TIMES
         BMI
               CONVR2
         ASL
               MLTTMP
         ROL
               MLTTMP+1
         JMP
               CONVR1
CONVR2: LDY DIV2TB,X ; NOW DIVIDE HCRSR TO ACCOUNT FOR PARTIAL BYT
              COLCRS
         LDA
         LDX
               #7
                             ; * TRICKY *
CONVR3: DEY
```

```
BMI
               CONVR4
         DEX
         LSR
               COLCRS+1
         ROR
               A
              TMPLBT
                         ; SAVE LOW BITS FOR MASK
         ROR
         JMP
               CONVR3
CONVR4: INY
                          ;SO Y IS ZERO UPON RETURN FROM THIS ROUTINE
         CLC
               MLTTMP
         ADC
                            ; ADD SHIFTED COLCRS TO MLTThP
               MLTTMP
         STA
         BCC
               CONVR5
         INC
               MLTTMP+1
CONVR5: SEC
                          ; * TRICKY *
CONVR6: ROR TMPLBT ;SLIDE A "1" UP AGAINST LOW BITS (CONTINUE T
         CLC
         DEX
                               ; AND FINISH SHIFT SO LOW BITS ARE
         BPL
              CONVR6
                             ; RIGHT JUSTIFIED.
         LDX
               TMPLBT
                             ;TMPLBT IS NOW THE INDEX INTO DMASKTB
         LDA
               MLTTMP
                             ; PREPARE FOR RETURN
         CLC
              ADRESS
ADRESS
         ADC
         STA
         STA
               OLDADR
                            ; REMEMBER THIS ADDRESS FOR CURSOR
               MLTTMP+1
         LDA
         ADC
               ADRESS+1
         STA
               ADRESS+1
              OLDADR+1
DMASKT,X
         STA
         LDA
         STA
               DMASK
         STA
               SHFAMT
         PLA
         STA COLCRS+1
         PLA
         STA COLCRS
         PLA
         STA
               ROWCRS
         RTS
; INCREMENT CURSOR AND DETECT BOTH END OF LINE AND END OF SCREEN
                          ; NON-EXTEND ENTRY POINT
INCRSB: LDA #0
        BEQ INCREC
INCRSR: LDA #$9B
                     ;SPECIAL CASE ELIMINATOR
INCREC: STA INSDAT
INCRSA: INC LOGCOL
                        ; (INSCHR ENTRY POINT)
         INC
               COLCRS
               INCRS2
                            ;DO HIGH BYTE
         BNE
         INC
               COLCRS+1
INCRS2: LDA COLCRS
                        ;TEST END OF LINE
         LDX DINDEX
         CMP
               COLUMN, X ; TEST TABLED VALUE FOR ALL SCREEN MODES
               INC2A
                          ;DO CR IF EQUAL
         BEO
         CPX
               #0
                             ; MODE 0?
              INCRS3
RMARGN
INCRS3
                            ; IF NOT. JUST RETURN
         BNE
         CMP
                             ;TEST AGAINST RMARGN
                            ;EGUAL IS OK
         BEQ
               INC2A
        BCS
                        ; IF GREATER THAN, DO CR
INCRS3: RTS
```

```
;CHECK MODE
DOCR1 ;NOT 320X1 $0 DO IT
INC2A: CPX #8
        BCC
        LDA
               COLCRS+1 ; TEST MED
        BEQ INCRS3 ;ONLY AT 64 SO DON'T DO IT
DOCR1: LDA DINDEX ;DON'T MESS WITH LOGMAP IF NO MODE ZERO
        BNE DOCR
                            ;TEST LINE OVERRUN
               LOGCOL
        LDA
               #81
        CMP
        BCC DOCR1B
LDA INSDAT
                           ; IF LESS THAN 81 IT IS DEFINITELY NOT LINE 3
        LDA
                         ;ONLY DO LOG LINE OVERFLOW IF INSDAT <>0
              DOCR
        BEQ
        JSR
              DOCRWS
                           ;LOG LINE OVERFLOW IS SPECIAL CASE
        JMP
              INCRS1
                            ; RETURN
DOCR1B: JSR DOCR ;GET IT OVER WITH
        LDA ROWCRS
        CLC
                             ;TEST LOGICAL LINE BIT MAP
        ADC #120
        JSR
              BITGET
        BCC DOCR1A
LDA INSDAT
BEQ DOCR1A
                            ;DON'T EXTEND IF OVERRUN IS INTO MIDDLE OF L
                            ;DON'T EXTEND IF INSDAT IS ZERO
                           ; (INSCHR SPECIAL CASE)
        CLC
                             ; INSERT "0" INTO BIT MAP
        JSR INSLIA
DOCR1A: JMP DOLCOL ; CONVERT ROW AND COL TO LOGCOL AND RETURN
NOSCRI: LDA #0 : DOCR WITHOUT SCROUT
NOSCRL: LDA #0
                        ;DOCR WITHOUT SCROLL
       BEQ NOSCR1 ; (UNCONDITIONAL)
DOCRWS: LDA #$9B ;DOCR WITH SCROLLING (NORMAL MODE)
NOSCR1: STA INSDAT
DOCR: JSR COLCR ; PLACE COLCRS AT LEFT EDGE
        LDA
              #0
        STA
               COLCRS+1
        INC ROWCRS
DOCR2: LDX DINDEX
                           ;SET UP SCROLL LOOP COUNTER
        LDY #24
        BIT
              SWPFLG
        BPL
              DOCR2A
                            ; BRANCH IF NORMAL
        LDY
               #4
        TYA
        BNE DOCR2B
                      ; (UNCONDITIONAL)
DOCR2A: LDA NOROWS, X ; GET NO OF ROWS
DOCR2B: CMP ROWCRS
        BNE INCRS1
        STY
              HOLD3
                            ;DON'T SCROLL IF MODE <> 0
        TXA
        BNE INCRS1
        LDA
              INSDAT
                           ; OR IF INSDAT = 0
              INCRS1
        BEQ
              INSDAT
       LDA
                           IF INSDAT <> $9B THEN ROLL IN A 0
               #$9B ;TO EXTEND BOTTOM LOGICAL LINE
        CMP
        SEC
        BEQ DOCR4B
        CLC
DOCR4B: JSR SCROLL
                        ;LOOP SACK TO HERE IF >1 SCROLLS
        INC SCRFLG
                           ; ROWS MOVE UP SO BUFSTR SHOULD TOO
        DEC
              BUFSTR
              HOLD3
        DEC
              LOGMAP
        LDA
        SEC
                             FOR PARTIAL LINES ROLL IN A "1"
        BPL DOCR4B
                           ; AGAIN IF PARTIAL LOGICAL LINE
```

```
ROWCRS
         STA
                        COLVERT ROW AND COL TO LOGCDL AND RETURN
INCRS1: JMP DOLCOL
; SUBEND: SUBTRACT ENDPT FROM ROWAC OR COLAC. (X=0 OR 2)
SUBEND: SEC
         LDA ROWAC, X
SBC ENDPT
         SBC
         SBC ENDPT
STA ROWAC, X
LDA ROWAC+1, X
SBC ENDPT+1
STA ROWAC+1, X
         RTS
; RANGE: DO CURSOR RANGE TEST. IF ERROR, POP STACK TWICE AND JMP RETURN
         (ERANGE IS EDITOR ENTRY POINT AND TEST IF EDITOR IS OPEN.
          IF IT ISNT IT OPENS THE EDITOR AND CONTINUES)
ERANGE: LDA BOTSCR
                       ; IF BOTSCR=4
         CMP #4
                RANGE ;THEN IT IS IN MIXED NODE AND OK
DINDEX ;IF MODE = 0
         BEO
         LDA
                RANGE
                         ;THEN IT IS INEDITOR MODE AND OK; IF NOT, OPEN EDITOR
         BEO
         JSR EOPEN
                          ;**** RANGE CHECK RMARGN **** SET UP AC
RANGE: LDA #39
         CMP RMARGN
                              ; **** RANGE CHECK RMARGN **** COMPARE
                RANGE3
                              ; **** RANGE CHECK RMARGN **** BRANCH GE
         BCS
         STA
                RMARGN
                              ; **** RANGE CHECK RMARGN **** BAD SO STORE
RANGE3: LDX DINDEX
         LDA NOROWS, X ; CHECK ROWS
                ROWCRS
         CMP
                              ; (ERROR IF TABLE.GE.ROWCRS)
         BCC
                RNGERR
         BEQ
               RNGERR
         CPX
                #8
                               ; CHECK FOR 320X1
               RANGE1 ;SPECIAL CASE IT
COLCRS+1
RNGOK ;IF HIGH BYTE IS 0, COL IS OK
         BNE
         LDA
         BEO
         CMP
                #1
                              ;IF >1, BAD
         BNE
                RNGERR
         BEO RANGE 2
                              ; IF 1, GO CHECK LOWBYTE
RANGE1: LDA COLCRS+1 ; FOR OTHERS, NON-ZERO HIGH BYTE IS BAD
         BNE
                RNGERR
RANGE2: LDA COLUMN, X ; CHECK LOWBYTE
         CMP
                COLCRS
         BCC
                RNGERR
         BEQ
                RNGERR
RNGOK: LDA #SUCCES ;SET STATUS OK
         STA DSTAT
         LDA
                #BRKABT
                           ; PREPARE BREAK ABORT STATUS
                BRKKEY
         LDX
                              ; CHECK BREAK KEY FLAG
         STA
                              ; 'CLEAR' BREAK
                BRKKEY
         BEQ
                              ; IF BREAK, QUIT IMMEDIATELY AND RETURN TO CI
                RNGER2
         RTS
RNGERR: JSR HOME ;ON RANGE ERROR, BRING CURSOR BACK
         LDA #CRSROR ;SHOW CURSOR OVERRANGE ERROR
RNGER2: STA DSTAT
```

; PLACE CURSOR AT NEW LINE NEAR THE BOTTOM

HOLD3

 $L^{1}DA$

```
RNGER1: PLA
                            ; RESTORE STACK (THIS ROUTINE IS ALWAYS 1 LEV
                                ; AWAY FROM RETURN TO CIO)
                              ; IF SWAPPED. SWAP BACK
         LDA
                SWPFLG
         BPL
                RETUR3
         JSR SWAPA
                           ; AND DONT DO RETUR1
                         ; RETURN TO CIO
RETUR3: JMP RETUR1
; OFFCRS: RESTORE OLD DATA UNDER CURSOR SO IT CAN BE MOVED
OFFCRS: LDY #0
         LDA OLDCHR
STA (OLDADR),Y
         RTS
; BITMAP ROUTINES:
; BITCON: PUT MASK IN BITMSK AND INDEX IN X
; BITPUT: PUT CARRY INTO BITMAP
; BITROL: ROL CARRY INTO BOTTOM OF BITMAP (SCROLL)
; BITSET: SET PROPER BIT
; BITCLR: CLEAR PROPER BIT
; BITGET: RETURN CARRY SET IF BIT IS THERE
; LOGGET: DO BITGET FOR LOGMAP INSTEAD OF TABMAP
BITCON: PHA
         AND
                #7
         TAX
                               GET MASK
         LDA MASKTB,X
STA BITMSK
                               ; PROCESS INDEX
         PLA
               A
         LSR
         LSR
LSR
               Α
               Α
         TAX
         RTS
BITROL: ROL LOGMAP+2
        ROL LOGMAP+1
         ROL
               LOGMAP
         RTS
BITPUT: BCC BITCLR ; AND RETURN
; OTHERWISE FALL THROUGH TO BITSET AND RETURN
BITSET: JSR BITCON
         LDA TABMAP,X
         ORA
               BITMSK
         STA
               TABMAP,X
         RTS
BITCLR: JSR BITCON
         LDA BITMSK
         EOR
                #$FF
         AND TABMAP, X
```

```
RTS
LOGGET: LDA ROWCRS
LO1GET: CLC
LO2GET: ADC #120
BITGET: JSR BITCON
         CLC
         LDA TABMAP,X
AND BITMSK
               BITMSK
         BEQ BITGE1
         SEC
BITGE1: RTS
; INATAC: INTERNAL(CHAR) TO ATASCII(ATACHR) CONVERSION
INATAC: LDA CHAR
         LDY DINDEX ; IF GRAPHICS MODES
         CPY
                #3
         BCS
               INATA1 ;THEN DON'T CHANGE CHAR
         ROL
         ROL
                Α
         ROL
               Α
         ROL
               A
             #3
         AND
         TAX
               CHAR
         LDA
         AND
               #$9F
         ORA INTATA, X
INATA1: STA ATACHR
        RTS
; MOVLLN: MOVE 40 BYTES AT FRMADR TO TOADR SAVING OLD TOAOR
        DATA IN THE LINBUF. THEN MAKE NEXT FRMADR
        BE AT LINBUF FOR NEXT TRANSFER & TOADR=TOADR+40
MOVLIN: LDA #LINBUF/256 ;SET UP ADRESS=LINBUF$=247
         STA ADRESS+1
               #LINBUF.AND.$FF
         LDA
                ADRESS
         STA
         LDY
                #39
MOVLI1: LDA (TOADR), Y ; SAVE TO DATA
         STA TMPCHR
         LDA
               (FRMADR),Y ;STORE DATA
               (FRMADR),Y
(TOADR),Y
TMPCHR
         STA
         LDA
         STA
               (ADRESS),Y
         DEY
               MOVLI1
ADRESS+1 ;SET UP FRMADR=LAST LINE
         BPL
         LDA
               FRMADR+1
         STA
         LDA
               ADRESS
         STA
               FRMADR
         CLC
                                ; ADD 40 TO TOADR
```

STA TABMAP, X

```
LDA
               TOADR
         ADC
               #40
         STA TOADR
BCC MOVLI2
INC TOADR+1
MOVLI2: RTS
; EXTEND: EXTEND BIT MAP FROM ROWCRS (EXTEND LOGICAL LINE
EXTEND: PHP
                          ;SAVE CARRY
        LDY
               #23
EXTEN1: TYA
         JSR LO1GET
         PHP
         TYA
         CLC
         ADC
               #121
         PLP
         JSR BITPUT
EXTEN3: DEY
        BMI
              EXTEN4
         CPY
               ROWCRS
         BCS
               EXTEN1
EXTEN4: LDA ROWCRS
         CLC
         ADC #120
         PLP
         JMP BITPUT ;STORE NEW LINE'S BIT AND RETURN
; CLRLIN: CLEAR LINE CURSOR IS ON
CLRLIN: LDA LMARGN
         STA COLCRS
         JSR CONVRT
         LDY
               #39
         LDA
               #0
CLRLI1: STA (ADRESS), Y
         DEY
         BPL CLRLI1
         RTS
;
; SCROLL: SCROLL SCREEN
SCROLL: JSR BITROL ;ROLL IN CARRY LDA SAVMSC ;SET UP WOR
                         ;SET UP WORKING REGISTERS
         STA
               ADRESS
         LDA
               SAVMSC+1
         STA ADRESS+1
SCROL1: LDY #40
                         ;LOOP
         LDA
              (ADRESS),Y
         LDX
               RAMTOP ; TEST FOR LAST LINE
         DEX
```

```
ADRESS+1
         CPX
         BNE
               SCROL2
         LDX
               #$D7
         CPX
               ADRESS
              SCROL2
         BCS
         LDA
               #0
                              ;YES SO STORE ZERO DATA FOR THIS ENTIRE LINE
SCROL2: LDY #0
              (ADRESS),Y
ADRESS
SCROL1
         STA
         INC
         BNE
              ADRESS+1
ADRESS+1
         INC
         LDA
               RAMTOP
         CMP
         BNE SCROL1
JMP DOLCOL
                          ; AND RETURN
; DOLCOL: DO LOGICAL COLUMN FROM BITMAP AND COLCRS
                         ;START WITH ZERO
DOLCOL: LDA #0
         STA
               LOGCOL
         LDA
                ROWCRS
         STA
               HOLD1
DOLCO1: LDA HOLD1
                      ; ADD IN ROW COMPONENT
         JSR LO1GET
         BCS DOLCO2
LDA LOGCOL
                             ; FOUND BEGINNING OF LINE
                             ; ADD 40 AND LOOK BAC ONE
         CLC
         ADC
               #40
               LOGCOL
         STA
         DEC HOLD1
JMP DOLCO1
                          ;UP ONE LINE
         LC
LDA LOGCOL
ADC COLCRS
DOLCO2: CLC
                           ; ADD IN COLCRS
         STA
               LOGCOL
         RTS
;
 DOBUFC: COMPUTE BUFFER COUNT AS THE NUMBER OF BYTES FROM
          BUFSTR TO END OF LOGICAL LINE WITH TRAILING SPACES REMOVED
DOBUFC: JSR PHACRS
         LDA LOGCOL
         PHA
                             ;START
               BUFSTR
         LDA
         STA
               ROWCRS
         LDA
               BUFSTR+1
               COLCRS
         STA
               #1
         LDA
         STA BUFCNT
DOBUF1: LDX #23
                         ;NORMAL
         LDA
              SWPFLG
                           ; IF SWAPPED, ROW 3 IS THE LAST LINE ON SCREE
         BPL
               DOB1
         LDX
                #3
       CPX ROWCRS
DOB1:
                         ;TEST IF CRSR IS AT LAST SCREEN POSITION
         BNE DOBU1A
         LDA
               COLCRS
```

```
CMP
               RMARGN
         BNE
                DOBU1A
         INC
               BUFCNT
                            ; YES, SO FAKE INCRSP TO AVOID SCROLLING
         JMP
               D0BUF2
DOBU1A: JSR INCRSB
         INC BUFCNT
         LDA
               LOGCOL
               LMARGN
         CMP
         BNE
               DOBUF1
                            ;NOT YET EOL
         DEC
               ROWCRS
                            ;BACK UP ONE INCRSR
            CRSRLF
         JSR
DOBUF2: JSR GETPLT
                        ;TEST CURRENT COLUMN FOR NON-ZERO DATA
              DOBUF4
         BNE
                            ;QUIT IF NON-ZERO
              BUFCNT
LOGCOL
LMARGN
DOBUF4
         DEC
                            ; DECREMENT COUNTER
                            ;BEGINNING OF LOGICAL LINE YET?
         LDA
         CMP
         BEQ
                            ;YES, SO QUIT
         JSR
               CRSRLF
                            ;BACK UP CURSOR
              COLCRS
         LDA
                            ; IF LOGCOL=RMARGN, GO UP 1 ROW
         CMP
               RMARGN
         BNE
               DOBUF3
             ROWCRS
         DEC
DOBUF3: LDA BUFCNT
         BNE DOBUF2
                            ;LOOP UNLESS BUFCNT JUST WENT TO ZERO
DOBUF4: PLA
         STA LOGCOL
JSR PLACRS
         RTS
; STRBEG: MOVE BUFSTR TO BEGINNING OF LOGICAL LINE.
STRBEG: JSR DOLCOL ;USE DOLCOL TO POINT HOLD1 AT BOL
         LDA HOLD1
         STA BUFSTR
         LDA
               LMARGN
         STA BUFSTR+1
         RTS
; DELTIM: TIME TO DELETE A LINE IF IT IS EMPTY AND AN EXTENSION
DELTIA: LDA LOGCOL ; IF LOGCOL<>LMARGN
         CMP LMARGN
                            ;THEN DONT MOVE UP ONE
               DELTIG
                            ;LINE BEFORE TESTING DELTIM
         BNE
         DEC
                ROWCRS
DELTIG: JSR DOLCOL
                        ;TEST FOR EXTENSION
DELTIM: LDA LOGCOL
         CMP LMARGN
              DELTI3
CONVRT
RMARGN
                            ; NO
         BEO
         JSR
                           ;SET UP COUNT
         LDA
         SEC
         SBC LMARGN
```

```
TAY
DELTI1: LDA (ADRESS), Y
        BNE DELTI3 ;FOUND A NON-0 SD QUIT AND RETURN
        DEY
        BPL DELTI1
DELTI2: JMP DELLIB ;DELETE A LINE AND RETURN
DELTI3: RTS
; TSTCTL: SEARCH CNTRLS TABLE TO SEE IF ATACHR IS A CNTL CHAR
TSTCTL: LDX #45
                        ; PREPARE TO SEARCH TABLE
TSTCT1: LDA CNTRLS,X
        CMP ATACHR
        BEQ
              TSTCT2
        DEX
        DEX
        DEX
        BPL TSTCT1
TSTCT2: RTS
; PUSH ROWCRS, COLCRS AND COLCRS+1
PHACRS: LDX #2
PHACR1: LDA ROWCRS,X
        STA TMPROW, X
        DEX
        BPL PHACR1
        RTS
; PULL COLCRS+1, COLCRS AND ROWCRS
PLACRS: LDX #2
PLACR1: LDA TMPROW, X
         STA ROWCRS, X
        DEX
        BPL PLACR1
        RTS
; SWAP: IF MIXED MODE, SWAP TEXT CURSORS WITH REGULAR CURSORS
SWAP: JSR SWAPA ;THIS ENTRY POINT DOESRETUR1
       JMP RETURI
SWAPA: LDA BOTSCR
        CMP
              #24
        BEO
               SWAP3
        LDX
               #11
SWAP1: LDA ROWCRS,X
        PHA
        LDA
              TXTROW,X
ROWCRS,X
         STA
         PLA
         STA TXTROW, X
```

```
BPL SWAP1
LDA SWPFLG
EOR #$FF
STA SWPFLG
SWAP3: RTS
; CLICK: MAKE CLICK THROUGH KEYBOARD SPEAKER
CLICK: LDX #$7F
CLICK1: STX CONSOL
          STX WSYNC
          DEX
          BPL CLICK1
          RTS
;
; COLCR: PUTS EITHER 0 OR LMARON INTO COLCRS BASED ON MODE AND SWPFLG
COLCR: LDA #0
          LDX SWPFLG
          BNE
                 COLCR1
          LDX
                 DINDEX
          BNE COLCR2
COLCR1: LDA LMARGN
COLCR2: STA COLCRS
         RTS
; PUTMSC: PUT SAVMSC INTO ADRESS
PUTMSC: LDA SAVMSC ;SETUP ADDRESS
          STA ADRESS
          LDA
                 SAVMSC+1
                 ADRESS+1
          STA
          RTS
          .PAGE
; DRAW -- DRAW A LINE FROM OLDROW, OLDCOL TO NEWROW, NEWCOL
; (THE AL MILLER METHOD FROM BASKETBALL)
DRAW: LDX #0
          LDA ICCOMZ ;TEST COMMAND: $11=DRAW $12=FILL
          CMP
                 #$11
         BEQ DRAWA
CMP #$12 ;TEST FILL
BEQ DRAWB ;YES
LDY #NVALID ;NO, SO RETURN INVALID COMMAND
         RTS
DRAWB: INX
DRAWA: STX FILFLG
         LDA ROWCRS
                                ; PUT CURSOR INTO NEWROW, NEWCOL
          STA NEWROW
LDA COLCRS
STA NEWCOL
LDA COLCRS+1
STA NEWCOL+1
```

DEX

	LDA		#1	
	STA		ROWINC	;SET UP INITIAL DIRECTIONS
	STA		COLINC	
	SEC LDA		MENDON	DEMEDMINE DELET DOM
	SBC		NEWROW	; DETERMINE DELTA ROW
	SBC		OLDROW DELTAR	
	BCS		DELIAR DRAW1	;DO DIRECTION AND ABSOLUTE VALUE
	LDA		#\$FF	;BORROW WAS ATTEMPTED
	STA		ROWINC	;SET DIRECTION DOWN
	LDA		DELTAR	/SEI DIRECTION DOWN
	EOR		#\$FF	;DELTAR = DELTAR
	CLC		11 4	, , , , , , , , , , , , , , , , , , , ,
	ADC		#1	
	STA		DELTAR	
DRAW1:	SEC			
	LDA		NEWCOL	; NOW DELTA COLUMN
	SBC		OLDCOL	
	STA		DELTAC	
	LDA		NEWCOL+1	;TWO-BYTE QUANTITY
	SBC		OLDCOL+1	
	STA		DELTAC+1	
	BCS		DRAW2	;DIRECTION AND ABSOLUTE VALUE
	LDA		#\$FF	;BORROW WAS ATTEMPTED
	STA		COLINC	;SET DIRECTION = LEFT
	LDA		DELTAC	
	EOR		#\$FF	;DELTAC = DELTAC
	STA		DELTAC	
	LDA		DELTAC+1	
	EOR		#\$FF -	
	STA		DELTAC+1	
	INC		DELTAC	; ADD ONE FOR TWOS COMPLEMENT
	BNE		DRAW2	
י מיז גיים	INC	що	DELTAC+1	·ZEDO DAM EOD DDAM LOOD
DRAW2:	LDX LDY	#2	#0	;ZERO RAM FOR DRAW LOOP
	STY		#0 COLAC+1	
DRAW3A:			COLACTI	
DIAW 3A •	STA		ROWAC,X	
	LDA		OLDROW, X	
	STA		ROWCRS,X	
	DEX		, , ,	
	BPL		DRAW3A	
	LDA		DELTAC	;FIND LARGER ONE (ROW OR COL)
;	STA		COUNTR	(PREPARE COUNTR AND ENDPT)
;	STA		ENDPT	
	INX			; MAKE X 0
	TAY			
	LDA		DELTAC+1	
	STA		COUNTR+1	
	STA		ENDPT+1	
	BNE		DRAW3	;AUTOMATICALLY LARGER IF MSD>0
	LDA		DELTAC	
	CMP		DELTAR	;LOW COL >LOW ROW?
	BCS		DRAW3	; YES
	LDA		DELTAR	
	LDX		#2	
י כיזיע מע	TAY			DITE THE INTERTAL CONDITIONS
DRAW3:	TYA			; PUT IN INITIAL CONDITIONS

```
STA
              COUNTR
         STA
               ENDPT
         PHA
                              ;SAVE AC
        LDA ENDPT+1 ; PUT LSB OF HIGH BYTE LSR A ; INTO CARRY
         PLA
                              ; RESTORE AC
               A
                          ; ROR THE 9 BIT ACUMULATOR
         ROR
               ROWAC,X
         STA
DRAW4A: LDA COUNTR ;TEST ZERO
               COUNTR+1
         ORA
               DRAWI1
                           ; IF COUNTER IS ZERO, LEAVE DRAW
         BNE
         JMP
               DRAW10
                          ; ADD ROW TO ROWAC (PLOT LOOP)
DRAWI1: CLC
        LDA
              ROWAC
         ADC
               DELTAR
         STA
               ROWAC
        BCC
              DRAW5
         INC
               ROWAC+1
DRAW5: LDA ROWAC+1 ; COMPARE ROW TO ENDPOINT
                        ; IF HIGH BYTE OF ROW IS .LT. HIGH
        CMP ENDPT+1
         BCC
               DRAW6
                         ;BYTE OF ENDPT, BLT TO COLUMN
         BNE
               DRAW5A
         LDA
              ROWAC
                         ;LOW BYTE
         CMP
               ENDPT
         BCC
               DRAW6
                         ;ALSO TILT
DRAW5A: CLC
                          GE SO MOVE POINT
        LDA ROWCRS
ADC ROWINC
         STA
              ROWCRS
                            ; AND SUBTRACT ENDPT FROM ROWAC
         LDX
               #0
        JSR
               SUBEND
DRAW6: CLC
                          ; DO SAME FOR COLUMN (DOUBLE BYTE ADD)
        LDA COLAC ; ADD
         ADC
              DELTAC
         STA
               COLAC
         LDA
               COLAC+1
         ADC
               DELTAC+1
         STA
               COLAC+1
         CMP
              ENDPT+1 ; COMPARE HIGH BYTE
         BCC
              DRAW8
         BNE
               DRAW6A
         LDA
               COLAC
                         ; COMPARE LOW BYTE
         CMP
               ENDPT
        BCC
               DRAW8
                        ; + OR - ?
DRAW6A: BIT COLINC
         BPL DRAW6B
                            ; DO DOUBLE BYTE DECREMENT
         DEC
               COLCRS
        LDA
               COLCRS
         CMP
               #$FF
         BNE
               DRAW7
         LDA
              COLCRS+1
         BEO
               DRAW7
                        ;DON'T DEC IF ZERO
               COLCRS+1
         DEC
         BPL
               DRAW7
                         ; (UNCONDITIONAL)
                        ; DO DOUBLE BYTE INCREMENT
DRAW6B: INC COLCRS
               DRAW7
         BNE
         INC
               COLCRS+1
DRAW7: LDX #2
                         ; AND SUBTRACT ENDPT FROM COLAC
         JSR
               SUBEND
```

```
DRAW8: JSR RANGE
         JSR
               OUTPLT
                              ; PLOT POINT
         LDA
                {	t FILFLG}
                             ;TEST RIGHT FILL
         BEQ
                DRAW9
                PHACRS
         JSR
         LDA
                ATACHR
         STA
                \mathtt{HOLD4}
DRAW8A: LDA ROWCRS
                         ; SAVE ROW IN CASE OF CR
         PHA
               INCRSA
                              ; POSITION CURSOR ONE PAST DOT
         JSR
         PLA
                                ; RESTORE ROWCRS
         STA
                ROWCRS
DRAW8C: JSR RANGE
         JSR
                              GET DATA
               GETPLT
         BNE
                              ;STOP IF NON-ZERO DATA IS ENCOUNTERED
                DRAW8B
         LDA
                FILDAT
                              ;FILL DATA
         STA
                ATACHR
         JSR
                OUTPLT
                              ;DRAW IT
         JMP
                DRAW8A
                              ;LOOP
DRAW8B: LDA HOLD4
         STA
               ATACHR
         JSR
               PLACRS
DRAW9: SEC
                            ; DO DOUBLE BYTE SUBTRACT
         LDA
               COUNTR
         SBC
                #1
         STA
               COUNTR
               COUNTR+1
         LDA
         SBC
                #0
         STA
                COUNTR+1
         BMI
                DRAW10
         JMP
                DRAW4A
DRAW10: JMP RETUR1
         .PAGE
; TABLES
; MEMORY ALLOCATION
ALOCAT: .BYTE 24,16,10,10,16,28,52,100,196,196,196,196
;
; NUMBER OF DISPLAY LIST ENTRIES
NUMDLE: .BYTE 23,23,11,23,47,47,95,95,97,97,97
MXDMDE: .BYTE 19,19,9,19,39,39,79,79,65,65,65,65; (EXT OF NUMDLE)
; ANTIC CODE FROM INTERNAL MODE CONVERSION TABLE
   INTERNAL
                      ANTIC CODE
                                              DESCRIPTION
        0
                           2
                                              40X2X8 CHARACTERS
```

```
1
                            6
                                                20X5X8
         2
                            7
                                                20X5X16
                                                         11 11
                                                40X4X8 GRAPHICS
        3
                            8
                            9
                                                80X2X4
        5
                                                80X4X4
                           Α
                           В
                                                160X2X2 ""
                                                160X4X2 ""
        7
                           D
        8
                           F
                                                320X2X1
        9
                           SAME AS 8 BUT GTIA 'LUM' MODE
                         SAME AS 8 BUT GTIA 'COL/LUM REGISTER' MODE
        10
        11
                         SAME AS 8 BUT GTIA 'COLOR' MODE
ANCONV: .BYTE 2,6,7,8,9,$A,$B,$D,$F,$F,$F,$F;ZEROS FOR RANGE TEST IN
;
; PAGE TABLE TELLS WHICH DISPLAY LISTS ARE IN DANGER OF
; CROSSING A 256 BYTE PAGE BOUNDARY
PAGETB: .BYTE 0,0,0,0,0,0,1,1,1,1,1
; THIS IS THE NUMBER OF LEFT SHIFTS NEEDED TO MULTIPLY
; COLCRS BY 10,20, OR 40. (ROWCRS*10)/(2**DHLINE)
DHLINE: .BYTE 2,1,1,0,0,1,1,2,2,2,2
;
; COLUMN: NUMBER OF COLUMNS
COLUMN: .BYTE 40,20,20,40,80,80,160,160,64,80,80,80; MODE 8 IS SPECIAL
;
; NOROWS: NUMBER OF ROWS
NOROWS: .BYTE 24,24,12,24,48,48,96,96,192,192,192,192
; DIV2TB: HOW MANY RIGHT SHIFTS FOR HCRSR FOR PARTIAL BYTE MODES
DIV2TB: .BYTE 0,0,0,2,3,2,3,2,3,1,1,1
; DMASKT: DISPLAY MASK TABLE
```

```
DMASKT: .BYTE $00,$FF,$F0,$0F
          .BYTE $C0,$30,$0C,$03
; MASKTB: BIT MASK. (ALSO PART OF DMASKTB DO NOT SEPARATE)
MASKTB: .BYTE $80,$40,$20,$10,$08,$04,$02,$01
;
COLRTB: .BYTE $28,$CA,$94,$46,$00
;CNTRLS: CONTROL CODES AND THEIR DISPLACEMENTS INTO THE
          CONTROL CHARACTER PROCESSORS
CNTRLS: .BYTE $1B
          .WORD ESCAPE
          .BYTE $1C
          .WORD CRSRUP
          .BYTE $1D
          .WORD CRSRDN
          .BYTE $1E
          .WORD CRSRLF
          .BYTE $1F
          .WORD CRSRRT
          .BYTE $7D
          .WORD CLRSCR
          .BYTE $7E
          .WORD BS
          .BYTE $7F
          .WORD TAB
          .BYTE $9B
          .WORD DOCRWS
          .BYTE $9C
          .WORD DELLIN
          .BYTE $9D
          .WORD INSLIN
          .BYTE $9E
          .WORD CLRTAB
          .BYTE $9F
          .WORD SETTAB
          .BYTE $FD
          .WORD BELL
          .BYTE $FE
          .WORD DELCHR
          .BYTE $FF
          .WORD INSCHR
; ATAINT: ATASCI TO INTERNAL TABLE
```

```
ATAINT: .BYTE $40,$00,$20,$60
; INTATA: INTERNAL TO ATASCI TABLE
INTATA: .BYTE $20,$40,$00,$60
; ATASCI: ATASCII CONVERSION TABLE
ATASCI: .BYTE
                $6C,$6A,$3B,$80,$80,$6B,$2B,$2A ;LOWER CASE
          .BYTE $6F,$80,$70,$75,$9B,$69,$2D,$3D
          .BYTE $76,$80,$63,$80,$80,$62,$78,$7A
          .BYTE $34,$80,$33,$36,$1B,$35,$32,$31
          .BYTE $2C,$20,$2E,$6E,$80,$6D,$2F,$81
          .BYTE $72,$80,$65,$79,$7F,$74,$77,$71
          .BYTE $39,$80,$30,$37,$7E,$38,$3C,$3E
          .BYTE $66,$68,$64,$80,$82,$67,$73,$61
          .BYTE $4C,$4A,$3A,$80,$80,$4B,$5C,$5E ;UPPER CASE
          .BYTE $4F,$80,$50,$55,$9B,$49,$5F,$7C
          .BYTE $56,$80,$43,$80,$80,$42,$58,$5A
          .BYTE $24,$80,$23,$26,$1B,$25,$22,$21
          .BYTE $5B,$20,$5D,$4E,$80,$4D,$3F,$81
          .BYTE $52,$80,$45,$59,$9F,$54,$57,$51
          .BYTE $28,$80,$29,$27,$9C,$40,$7D,$9D
          .BYTE $46,$48,$44,$80,$83,$47,$53,$41
          .BYTE $0C,$0A,$7B,$80,$80,$0B,$1E,$1F ; CONTROL
          .BYTE $0F,$80,$10,$15,$9B,$09,$1C,$1D
```

.BYTE \$16,\$80,\$03,\$80,\$80,\$02,\$18,\$1A

```
.BYTE $80,$80,$85,$80,$1B,$80,$FD,$80
          .BYTE $00,$20,$60,$0E,$80,$0D,$80,$81
          .BYTE $12,$80,$05,$19,$9E,$14,$17,$11
          .BYTE $80,$80,$80,$80,$FE,$80,$7D,$FF
          .BYTE $06,$08,$04,$80,$84,$07,$13,$01
PIRQ5: LDA KBCODE
         CMP
                 CH1
                               ;TEST AGAINST LAST KEY PRESSED
                 PIRQ3
         BNE
                           ; IF NOT, GO PROCESS KEY
         LDA
                 KEYDEL
                               ; IF KEY DELAY BYTE > 0
                 PIRQ4
                            ; IGNORE KEY AS BOUNCE
PIRQ3: LDA KBCODE
                           ; RESTORE AC
         CMP
                              ;TEST CONTROL 1 (SSFLAG)
                #CNTL1
         BNE
                PIRQ1
         LDA
                SSFLAG
         EOR
                 #$FF
         STA
                 SSFLAG
         BCS
                PIRO4
                           ; (UNCONDITIONAL) MAKE ^1 INVISIBLE
PIRQ1: STA CH
         STA
                 CH1
         LDA
                 #3
         STA
                              ; INITIALIZE KEY DELAY FOR DEBOUNCE
                 KEYDEL
         LDA
                 #0
                               ;CLEAR COLOR SHIFT BYTE
         STA
                 ATRACT
       LDA #$30
PIRQ4:
                SRTIMR
         STA
PIRQ2: PLA
         RTI
;
         .BYTE $FF,$FF,$FF,$FF,$FF
CRNTPC =*
         *=$14
KBDSPR: .BYTE
               $FFF8-CRNTPC ; GDISPLC IS TOO LONG
```

.END