

## 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
;
CHRORG  =      $E000      ;CHARACTER SET
VECTBL  =      $E400      ;VECTOR TABLE
VCTABL  =      $E480      ;RAM VECTOR INITIAL VALUE TABLE
CIOORG  =      $E4A6      ;CENTRAL I/O HANDLER
INTORG  =      $E6D5      ;INTERRUPT HANDLER
SIOORG  =      $E944      ;SERIAL I/O DRIVER
DSKORG  =      $EDEA      ;DISK HANDLER
PRNORG  =      $EE78      ;PRINTER HANDLER
CASORG  =      $EF41      ;CASSETTE HANDLER
MONORG  =      $F0E3      ;MONITOR/POWER UP MODULE
KBDORG  =      $F3E4      ;KEYBOARD/DISPLAY HANDLER
;
;
;
;
;      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
;      6        PUT
;      8        STATUS
;      A        SPECIAL
;      C        JUMP TO POWER ON INITIALIZATION ROUTINE
;      F        NOT USED
;
;
;
EDITRV  =      $E400      ;EDITOR
SCRENV  =      $E410      ;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.
;
DISKIW  =          $E450          ;DISK INITIALIZATION
DSKINV  =          $E453          ;DISK INTERFACE
CIOV    =          $E456          ;CENTRAL INPUT OUTPUT ROUTINE
SIOV    =          $E459          ;SERIAL INPUT OUTPUT ROUTINE
SETVBV  =          $E45C          ;SET SYSTEM TIMERS ROUTINE
SYSVBV  =          $E45F          ;SYSTEM VERTICAL BLANK CALCULATIONS
XITVBV  =          $E462          ;EXIT VERTICAL BLANK CALCULATIONS
SIOINV  =          $E465          ;SERIAL, INPUT OUTPUT INITIALIZATION
SENDEV  =          $E468          ;SEND ENABLE ROUTINE
INTINV  =          $E46B          ;INTERRUPT HANDLER INITIALIZATION
CIOINV  =          $E46E          ;CENTRAL INPUT OUTPUT INITIALIZATION
BLKBDV  =          $E471          ;BLACKBOARD MODE
WARMSV  =          $E474          ;WARM START ENTRY POINT
COLDSV  =          $E477          ;COLD START ENTRY POINT
RBLOKV  =          $E47A          ;CASSETTE READ BLOCK ENTRY POINT VECTOR
CSOPIV  =          $E47D          ;CASSETTE OPEN FOR INPUT VECTOR
;VCTABL = $E480
;
;
; OPERATING SYSTEM EQUATES
;
; COMMAND CODES FOR IOCB
OPEN     =          3              ;OPEN FOR INPUT/OUTPUT
GETREC   =          5              ;GET RECORD (TEXT)
GETCHR   =          7              ;GET CHARACTER(S)
PUTREC   =          9              ;PUT RECORD (TEXT)
PUTCHR   =          $B             ;PUT CHARACTER(S)
CLOSE    =          $C             ;CLOSE DEVICE
STATIS   =          $D             ;STATUS REQUEST
SPECIL   =          $E             ;BEGINNING OF SPECIAL ENTRY COMMANDS
;
; SPECIAL ENTRY COMMANDS
DRAWLN   =          $11            ;DRAW LINE
FILLIN   =          $12            ;DRAW LINE WITH RIGHT FILL
RENAME   =          $20            ;RENAME DISK FILE
DELETE   =          $21            ;DELETE DISK FILE
FORMAT   =          $22            ;FORMAT
LOCKFL   =          $23            ;LOCK FILE TO READ ONLY
UNLOCK   =          $24            ;UNLOCK LOCKED FILE
POINT    =          $25            ;POINT SECTOR
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 (
DIRECT   =          $2             ;OPEN FOR DIRECTORY ACCESS (D)
OPNIN    =          $4             ;OPEN FOR INPUT (ALL DEVICES)
OPNOT    =          $8             ;OPEN FOR OUTPUT (ALL DEVICES)
OPNINO   =          OPNIN+OPNOT    ;OPEN FOR INPUT AND OUTPUT (ALL DEVICES)
MXDMOD   =          $10            ;OPEN FOR MIXED MODE (E,S)

```

```

INSCLR  =          $20              ;OPEN WITHOUT CLEARING SCREEN (E,S)
;
; DEVICE NAMES
SCREDT  =          'E              ;SCREEN EDITOR (R/W)
KBD     =          'K              ;KEYBOARD (R ONLY)
DISPLY  =          'S              ;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      =          $9B
;
;
;      OPERATING SYSTEM STATUS CODES
;
SUCCESS =          $01              ;SUCCESSFUL OPERATION
;
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
OVRRUN  =          $8E              ;SERIAL BUS DATA OVERRUN
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                ;TEMPORARY REGISTER FOR RAM SIZE
TSTDAT: .RES      1                ;RAM TEST DATA REGISTER
;
; CLEARED ON COLOSTART ONLY
WARMST: .RES      1                ;WARM START FLAG

```

```

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
POKMSK: .RES      1          ;SYSTEM MASK FOR POKEY IRQ ENABLE
BRKKEY: .RES      1          ;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
;
DSKFMS: .RES      2          ;DISK FILE MANAGER POINTER
DSKUTL: .RES      2          ;DISK UTILITIES POINTER
;
PTIMOT: .RES      1          ;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
IOCBSZ  =              16   ;NUMBER OF BYTES PER IOCB
MAXIOC  =              8*IOCBSZ ;LENGTH OF THE IOCB AREA
IOCBAS  =*
ICHIDZ: .RES      1          ;HANDLER INDEX NUMBER (FF = IOCB FREE)
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          ;
ICPTLZ: .RES      1          ;PUT BYTE ROUTINE ADDRESS - 1
ICPTHZ: .RES      1          ;
ICBL LZ: .RES      1          ;BUFFER LENGTH LOW BYTE
ICBLHZ: .RES      1          ;
ICAX1Z: .RES      1          ;AUXILIARY INFORMATION FIRST BYTE
ICAX2Z: .RES      1          ;
ICSPRZ: .RES      4          ;TWO SPARE BYTES (CIO LOCAL USE)
ICIDNO  =              ICSPRZ+2 ;IOCB NUMBER X 16
CIOCHR  =              ICSPRZ+3 ;CHARACTER BYTE FOR CURRENT OPERATION
;
STATUS: .RES      1          ;INTERNAL STATUS STORAGE
CHKSUM: .RES      1          ;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
CRETRY: .RES      1          ;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      1

```

```

FTYPE:  .RES      1
FEOF:    .RES      1
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
;
ATTRACT: .RES      1      ;ATTRACT FLAG
DRKMSK:  .RES      1      ;DARK ATTRACT MASK
COLRSH:  .RES      1      ;ATTRACT 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      1
SAVMSC:  .RES      2
OLDROW:  .RES      1
OLDCOL:  .RES      2
OLDCHR:  .RES      1      ;DATA UNDER CURSOR
OLDADR:  .RES      2
NEWROW:  .RES      1      ;POINT DRAW GOES TO
NEWCOL:  .RES      2
LOGCOL:  .RES      1      ;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
BUFCNT:  .RES      1      ;BUFFER COUNT
BUFSTR:  .RES      2      ;EDITOR GETCH POINTER
BITMSK:  .RES      1      ;BIT MASK
SHFAMT:  .RES      1
ROWAC:   .RES      2
COLAC:   .RES      2
ENDPT:   .RES      2
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
INSDAT:  .RES      1
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
VDSLST: .RES      2      ;DISPLAY LIST NMI VECTOR
VPRCED: .RES      2      ;PROCEED LINE IRQ VECTOR
VINTER: .RES      2      ;INTERRUPT LINE IRQ VECTOR
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
VSEROC: .RES      2      ;POKEY SERIAL OUTPUT COMPLETE IRQ
VTIMR1: .RES      2      ;POKEY TIMER 1 IRQ
VTIMR2: .RES      2      ;POKEY TIMER 2 IRQ
VTIMR4: .RES      2      ;POKEY TIMER 4 IRQ
VIMIRU: .RES      2      ;IMMEDIATE IRQ VECTOR
CDTMV1: .RES      2      ;COUNT DOWN TIMER 1
CDTMV2: .RES      2      ;COUNT DOWN TIMER 2
CDTMV3: .RES      2      ;COUNT DOWN TIMER 3
CDTMV4: .RES      2      ;COUNT DOWN TIMER 4
CDTMV5: .RES      2      ;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      2      ;COUNT DOWN TIMER 2 JSR ADDRESS
CDTMF3: .RES      1      ;COUNT DOWN TIMER 3 FLAG
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
CDTMF5: .RES      1      ;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
DBSECT: .RES 1 ;NUMBER OF DISK BOOT SECTORS
BOOTAD: .RES 2 ;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 1
PADDL4: .RES 1
PADDL5: .RES 1
PADDL6: .RES 1
PADDL7: .RES 1
STICK0: .RES 1 ;JOYSTICK 0 RAM CELL
STICK1: .RES 1
STICK2: .RES 1
STICK3: .RES 1
PTRIG0: .RES 1 ;PADDLE TRIGGER 0
PTRIG1: .RES 1
PTRIG2: .RES 1
PTRIG3: .RES 1
PTRIG4: .RES 1
PTRIG5: .RES 1
PTRIG6: .RES 1
PTRIG7: .RES 1
STRIG0: .RES 1 ;JOYSTICK TRIGGER 0
STRIG1: .RES 1
STRIG2: .RES 1
STRIG3: .RES 1
;
CSTAT: .RES 1
WMODE: .RES 1
BLIM: .RES 1
IMASK: .RES 1
JVECK: .RES 2
;
; .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 & OLD COL FOR TEXT (AND THEN SOME)

```

```

TMPX1:  .RES      1
HOLD3:  .RES      1
SUBTMP: .RES      1
HOLD2:  .RES      1
DMASK:  .RES      1
TMPLBT: .RES      1
ESCFLG: .RES      1      ;ESCAPE FLAG
TABMAP: .RES     15
LOGMAP: .RES      4      ;LOGICAL LINE START BIT MAP
INVFLG: .RES      1      ;INVERSE VIDEO FLAG (TOGGLED BY ATARI KEY)
FILFLG: .RES      1      ;RIGHT FILL FLAG FOR DRAW
TMPROW: .RES      1
TMPCOL: .RES      2
SCRFLG: .RES      1      ;SET IF SCROLL OCCURS
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
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      1
COLOR4: .RES      1
;
;
      .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
CBAUDL: .RES      1      ;CASSETTE BAUD RATE LOW BYTE
CBAUDH: .RES      1
;
CRSINH: .RES      1      ;CURSOR INHIBIT (00 = CURSOR ON)
KEYDEL: .RES      1      ;KEY DELAY
CH1:    .RES      1
;
CHACT:  .RES      1      ;CHACTL REGISTER RAM
CHBAS:  .RES      1      ;CHBAS REGISTER RAM
;
      .RES      5      ;SPARE BYTES
;
CHAR:   .RES      1
ATACHR: .RES      1      ;ATASCII CHARACTER
CH:     .RES      1      ;GLOBAL VARIABLE FOR KEYBOARD

```



```

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      1          ;PERIPHERAL UNIT 1 BUS ID. NUMBER
DUNIT:   .RES      1          ;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      1          ;DEVICE TIME OUT IN 1 SECOND UNITS
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
TIMFLG:  .RES      1          ;TIME OUT FLAG FOR BAUD RATE CORRECTION
STACKP:  .RES      1          ;SIO STACK POINTER SAVE CELL
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      1          ;HANDLER INDEX NUMBER (FF = IOCB FREE)
ICDNO:   .RES      1          ;DEVICE NUMBER (DRIVE NUMBER)
ICCOM:   .RES      1          ;COMMAND CODE
ICSTA:   .RES      1          ;STATUS OF LAST IOCB ACTION
ICBAL:   .RES      1          ;BUFFER ADDRESS LOW BYTE
ICBAH:   .RES      1
ICPTL:   .RES      1          ;PUT BYTE ROUTINE ADDRESS - 1
ICPTH:   .RES      1
ICBLN:   .RES      1          ;BUFFER LENGTH LOW BYTE
ICBLH:   .RES      1
ICAX1:   .RES      1          ;AUXILIARY INFORMATION FIRST BYTE
ICAX2:   .RES      1

```

```

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
;
FPREC   =          6          ;FLOATING PT PRECISION (# OF BYTES)
; IF CARRY USED THEN CARRY CLEAR => NO ERROR, CARR
AFP     =          $D800      ;ASCII->FLOATING POINT(FP)
;
;                               INBUFF+CIX -> FR0, CIX, CARRY
FASC    =          $D8E6      ;FP -> ASCII FR0 -> LBUFF (INBUFF)
IFP     =          $D9AA      ;INTEGER -> FP
;
;                               0-$FFFF (LSB,MSB) IN FR0,FR0+1->FR0
FPI     =          $D9D2      ;FP -> INTEGER FR0 -> FR0,FR0+1, CARRY
FSUB    =          $DA60      ;FR0 <- FR0 - FR1 ,CARRY
FADD    =          $DA66      ;FR0 <- FR0 + FR1 ,CARRY
FMUL    =          $DADB      ;FR0 <- FR0 * FR1 ,CARRY
FDIV    =          $DB28      ;FR0 <- FR0 / FR1 ,CARRY
FLD0R   =          $DD89      ;FLOATING LOAD REG0      FR0  <- (X,Y)
FLD0P   =          $DD80      ;  "      "      "      FR0  <- (FLFTR)
FLD1R   =          $DD98      ;  "      "      REG1      FR1  <- (X,Y)
FLD1P   =          $DD9C      ;  "      "      "      FR1  <- (FLPTR)
FSTOR   =          $DDA7      ;FLOATING STORE REG0 (X,Y) <- FR0

```

```

FSTOP    =          $DDAB          ;   "   "   "   (FLPTR)  <- FR0
FMOVE    =          $DDB6          ;FR1 <- FR0
PLYEVL   =          $DD40          ;FR0 <- P(Z) = SUM(I=N TO 0) (A(I)*Z**I) CAR
;
;                               INPUT:  (X,Y) = A(N),A(N-1)...A(0) -> PLYARG
;                               ACC      = # OF COEFFICIENTS = DEGREE+1
;                               FR0      = Z
EXP       =          $DDC0          ;FR0 <- E**FR0 = EXP10(FR0 * LOG10(E)) CARRY
EXP10    =          $DDCC          ;FR0 <- 10**FR0 CARRY
LOG       =          $DECD          ;FR0 <- 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
COS       =          $BD73          ;FR0 <- COS(FR0) CARRY
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
FR0:      .RES      FPREC          ;FP REG0
FRE:      .RES      FPREC
FR1:      .RES      FPREC          ;FP REG1
FR2:      .RES      FPREC
FRX:      .RES      1              ;FP SPARE
EEXP:     .RES      1              ;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
CIX:      .RES      1              ;CURRENT INPUT INDEX
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
DEGON     =          6              ;INDICATES DEGREES
FLPTR:    .RES      2              ;POINTS TO USER'S FLOATING PT NUMBER
FPTR2:    .RES      2
; FLOATING PT ROUTINES' NON-ZERO PAGE RAM
; (NEEDED ONLY IF F.P. ROUTINES CALLED)
      *=$57E
LBPR1:    .RES      1              ;LBUFF PREFIX 1
LBPR2:    .RES      1              ;LBUFF PREFIX 2
LBUFF:    .RES      128           ;LINE BUFFER
PLYARG    =          LBUFF+$60     ;POLYNOMIAL ARGUMENTS
FPSCR     =          PLYARG+FPREC
FPSCR1    =          FPSCR+FPREC
FSCR      =          FPSCR
FSCR1     =          FPSCR1
LBFEND    =          *-1           ;END OF LBUFF
;
;
;
;
;
;
;
;

```

# COLLEEN MNEMONICS

;			
;			
POKEY	=	\$D200	;VBLANK ACTION: DESCRIPTION:
POT0	=	POKEY+0	;POT0-->PADDL0 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	=	POKEY+3	;POT3-->PADDL3 0-227 IN RAM CELL
POT4	=	POKEY+4	;POT4-->PADDL4 0-227 IN RAM CELL
POT5	=	POKEY+5	;POT5-->PADDL5 0-227 IN RAM CELL
POT6	=	POKEY+6	;POT6-->PADDL6 0-227 IN RAM CELL
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	=	POKEY+8	;NONE AUDCTL<--[SIO]
STIMER	=	POKEY+9	
SKRES	=	POKEY+10	;NONE SKRES<--[SIO]
SEROUT	=	POKEY+13	;NONE SEROUT<--[SIO]
IRQEN	=	POKEY+14	;POKMSK-->IRQEN (AFFECTED BY OPEN S: OR E:)
SKCTL	=	POKEY+15	;SSKCTL-->SKCTL SSKCTL<--[SIO]
;			
CTIA	=	\$D000	;VBLANK ACTION: DESCRIPTION:
HPOSP0	=	CTIA+0	
HPOSP1	=	CTIA+1	
HPOSP2	=	CTIA+2	
HPOSP3	=	CTIA+3	
HPOSM0	=	CTIA+4	
HPOSM1	=	CTIA+5	
HPOSM2	=	CTIA+6	
HPOSM3	=	CTIA+7	
SIZEP0	=	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	
COLPM0	=	CTIA+18	;PCOLR0-->COLPM0 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
COLPF1	=	CTIA+23	;COLOR1-->COLPF1 WITH ATTRACT MODE
COLPF2	=	CTIA+24	;COLOR2-->COLPF2 WITH ATTRACT MODE

```

COLPF3 = CTIA+25 ;COLOR3-->COLPF3 WITH ATTRACT MODE
COLBK = CTIA+26 ;COLOR4-->COLBK WITH ATTRACT MODE
PRIOR = CTIA+27 ;(ON OPEN S: OR E:) GPRIOR-->PRIOR
VDELAY = CTIA+28
GRCTL = CTIA+29
HITCLR = CTIA+30
CONSOL = CTIA+31 ;$08-->CONSOL TURN OFF SPEAKER
M0PF = CTIA+0
M1PF = CTIA+1
M2PF = CTIA+2
M3PF = CTIA+3
P0PF = CTIA+4
P1PF = CTIA+5
P2PF = CTIA+8
P3PF = CTIA+7
M0PL = CTIA+8
M1PL = CTIA+9
M2PL = CTIA+10
M3PL = CTIA+11
P0PL = CTIA+12
P1PL = CTIA+13
P2PL = CTIA+14
P3PL = CTIA+15
TRIG0 = CTIA+16 ;TRIG0-->STRIG0
TRIG1 = CTIA+17 ;TRIG1-->STRIG1
TRIG2 = CTIA+18 ;TRIG2-->STRIG2
TRIG3 = CTIA+19 ;TRIG3-->STRIG3
;
ANTIC = $D400 ;VBLANK ACTION DESCRIPTION
DMACTL = ANTIC+0 ;DMACTL<--SDMCTL ON OPEN S: OR E:
CHACTL = ANTIC+1 ;CHACTL<--CHACT ON OPEN S: OR E:
DLISTL = ANTIC+2 ;DLISTL<--SDLSTL ON OPEN S: OR E:
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 = ANTIC+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
PIA = $D300 ;VBLANK ACTION DESCRIPTION
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      =      $9B                      ;END OF RECORD
        .PAGE
;
; CIO JUMP VECTOR FOR USERS
        *=CIOV
        JMP      CIO                      ;GO TO CIO
;
; CIO INIT JUMP VECTOR FOR POWER UP
        *=CIOINV
        JMP      CIOINT                  ;GO TO INIT
;
;
; ERROR ROUTINE ADDRESS EQUATE
; ERRTNH =ERRTN/256                      "MOVED TO LINE 788"
; ERRTNL =-ERRTNH*256+ERRTN "MOVED TO LINE 789"
;
;
        *=CIOORG
;
; CIO INITIALIZATION (CALLED BY MONITOR AT POWER UP)
CIOINT:  LDX      #0
CIOI1:   LDA      #IOCFRE                ;SET ALL IOCB'S TO FREE
        STA      ICHID,X                ;BY SETTING HANDLER ID'S=$FF
        LDA      #ERRTNL
        STA      ICPTL,X                ;POINT PUT TO ERROR ROUTINE
        LDA      #ERRTNH
        STA      ICPTH,X
        TXA
        CLC
        ADC      #IOCBSZ                ;BUMP INDEX BY SIZE
        TAX
        CMP      #MAXIOC                ;DONE?
        BCC      CIOI1                  ;NO
        RTS                                ;YES, RETURN
;
; 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
        AND      #$F                    ;IS IOCB MULTIPLE OF 16?
        BNE      CIERR1                ;NO, ERROR
        CPX      #MAXIOC                ;IS INDEX TOO LARGE?
        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
        CPY    #12          ;12 BYTES
        BCC    IOC1A
;
; COMPUTE CIO INTERNAL VECTOR FOR COMMAND
        LDY    #NVALID      ;ASSUME INVALID CODE
        LDA    ICCOMZ        ;COMMAND CODE TO INDEX
        CMP    #OPEN        ;IS COMMAND LEGAL?
        BCC    CIERR4        ;NO
        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
        CMP    #2            ;IS IT CLOSE?
        BEQ    CICLOS        ;YES
        CMP    #8            ;IS IT STATUS OR SPECIAL?
        BCS    CISTSP        ;YES
        CMP    #4            ;IS IT READ?
        BEQ    CIREAD        ;YES
        JMP    CIWRIT        ;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

```

```

;
; 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
    JSR    COMENT        ;COMPUTE HANDLER ENTRY POINT
    BCS    CICLO2        ;GO IF ERROR IN COMPUTE
    JSR    GOHAND        ;GO TO HANDLER TO CLOSE DEVICE
CICLO2: LDA    #IOCFRE    ;GET IOCB "FREE" VALUE
    STA    ICHIDZ        ;SET HANDLER ID
    LDA    #ERRTNH
    STA    ICPTHZ        ;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
    JSR    GOHAND        ;GO TO HANDLER

;
; RESTORE HANDLER INDEX (DO IMPLIED CLOSE)
    LDX    ICIDNO        ;IOCB INDEX
    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  ICBL LZ
        ORA  ICBL LZ+1   ;IS BUFFER LENGTH ZERO?
        BNE  RCI3        ;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
        JSR  INCBFP      ;INCREMENT BUFFER POINTER
        LDA  ICCOMZ      ;GET COMMAND CODE
        AND  #2          ;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
RCI1:   JSR  DECBFL      ;DECREMENT BUFFER LENGTH
        BNE  RCI3        ;CONTINUE IF NON ZERO
        .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
;
; LOOP TO WAIT FOR EOL
RCI6:   JSR  GOHAND      ;GET BYTE FROM HANDLER
        STA  CIOCHR      ;SAVE CHARACTER
        BMI  RCI4        ;GO IF ERROR
;
; 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
RCI11:  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     WCI1A          ;YES
;
; ILLEGAL WRITE -- DEVICE OPENED FOR READ ONLY
        LDY     #RDONLY        ;ERROR CODE
WCI1B:  JMP     CIRTN1          ;RETURN
;
; COMPUTE AND CHECK ENTRY POINT
WCI1A:  JSR     COMENT          ;COMPUTE HANDLER ENTRY POINT
        BCS     WCI1B          ;GO IF ERROR IN COMPUTE
;
; PUT RECORD OR CHARACTERS
        LDA     ICBLLZ
        ORA     ICBLLZ+1      ;IS BUFFER LENGTH ZERO?
        BNE     WCI3          ;NO
        LDA     CIOCHR          ;GET CHARACTER
        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 BUFFER POINTER
;
; CHECK FOR TEXT RECORD
        LDA     ICCOMZ          ;GET COMMAND BYTE
        AND     #2             ;IS IT PUT RECORD?
        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
        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    ICBAHZ
        LDX    #0                ;LOOP COUNT AND INDEX
CIRTN3: LDA    IOCBAS,X          ;ZERO PAGE
        STA    IOCB,Y            ;TO USER AREA
        INX
        INY
        CPX    #12              ;12 BYTES
        BCC    CIRT3
;
; RESTORE A,X, & Y
        LDA    CIOCHR            ;GET LAST CHARACTER
        LDX    ICIDNO            ;IOCB INDEX
        LDY    ICSTAZ            ;GET STATUS AND SET FLAGS
        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
;
; 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
        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
        LDA    (ICSPRZ),Y        ;GET LOW BYTE OF VECTOR FROM
        TAX                    ;HANDLER ITSELF AND SAVE
        INY
        LDA    (ICSPRZ),Y        ;GET HI BYTE OF VECTOR
        STA    ICSPRZ+1
        STX    ICSPRZ            ;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 BELOW?
        BNE    DECBF1         ;NO
        DEC    ICBLLZ+1       ;DECREMENT HI BYTE
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    ICBLL,X        ;GET LOW BYTE OF INITIAL LENGTH
        SBC    ICBLLZ         ;SUBTRACT FINAL LOW BYTE
        STA    ICBLLZ         ;AND SAVE BACK
        LDA    ICBLLH,X       ;GET HI BYTE
        SBC    ICBLLZ+1
        STA    ICBLLHZ
        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
        PHA
        ;PUT ON STACK
        LDA    ICSPRZ         ;GET JUMP ADDRESS LO BYTE
        PHA
        ;PUT ON STACK
        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
        LDA    (ICBALZ),Y     ;GET DEVICE NAME FROM USER
        BEQ    CIERR2
        LDY    #MAXDEV        ;INITIAL COMPARE INDEX
DEVS1:  CMP    HATABS,Y        ;IS THIS THE DEVICE?
        BEQ    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
        BCS      DEVS4      ;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
        CMP      #$A         ;IS NUMBER IN RANGE?
        BCC      DEVS3      ;YES
        LDA      #1         ;NO. DEFAULT TO ONE
DEVS3:  STA      ICDNOZ      ;SAVE DEVICE NUMBER
        CLC
        BCS      DEVS4      ;SHOW NO ERROR
;
; RETURN
DEVS4:  RTS
        .PAGE
;
;
; CIO ROM TABLES
;
; COMMAND TABLE
; MAPS EACH COMMAND TO OFFSET FOR APPROPRIATE VECTOR IN HANDLER
COMTAB: BYTE      0,4,4,4,4,6,6,6,6,2,8,10

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  SYIRQB     ;VINTER
        .WORD  SYIRQB     ;VBREAK

```

```

;
        .RES      8
        .WORD SYIRQB      ;VTMIR1
        .WORD SYIRQB      ;VTIMR2
        .WORD SYIRQB      ;VTMIR4
        .WORD SYIRQ       ;VIMIRQ
        .WORD 0,0,0,0,0    ;CDTMV1-4

        .WORD SYSVBL      ;VVBLKI
        .WORD XITVBL      ;VVSLKD
;
        *=$900C
;
        LDA      #PIRQH      ;SET UP RAM VECTORS FOR LINBUG VERSION
        STA      $FFF9
        LDA      #PIRQL
        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      NMIEH        ;ENABLE DISPLAY LIST, VERTICAL BLANK
        LDA      #$38          ;LOOK AT DATA DIRECTION REGISTERS IN PIA
        STA      PACTL
        STA      PBCTL
        LDA      #0            ;MAKE ALL INPUTS
        STA      PORTA
        STA      PORTB
        LDA      #$3C          ;BACK TO PORTS
        STA      PACTL
        STA      PBCTL
        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 VSEROC-INTABS
        .BYTE VSEROR-INTABS
        .BYTE VSERIN-INTABS

SYIRQ:  PHA                                ;SAVE ACCUMULATOR
        LDA      IRQST                     ; CHECK FOR SERIAL IN
        AND      #$20
        BNE      SYIRQ2
        LDA      #$DF                      ; MASK ALL OTHERS
        STA      IRQEN
        LDA      POKMSK
        STA      IRQEN
        JMP      (VSERIN)

SYIRQ2:  TXA                                ;PUT X INTO ACC
        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 INTERRUPT ENABLED?
        BEQ      LL

LOOPM2:  BIT      IRQST                   ; IS IT THE INTERRUPT?
        BEQ      JMPP

LL:      DEX                                ;NO DEC X AND TRY NEXT MASK
        BPL      LOOPM                   ;IF NOT NEG 0010 LOOPH
        JMP      SYIRQ8                   ;DONE BUT NO INTERRUPT

JMPP:    EOR      #$FF                    ;COMPLEMENT MASK
        STA      IRQEN                   ;ENABLE ALL OTHERS
        LDA      POKMSK                   ; GET POKE MASK
        STA      IRQEN                   ; 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
        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      ATTRACT                   ; TURN OFF ATTRACT MODE
        PLA
        RTI                                ;EXIT FROM INT

SYIRQ8:  PLA
        TAX
        BIT      PACTL                    ;PROCEED ***I GUESS***
        BPL      SYIRQ9
        LDA      PORTA                    ;CLEAR INT STATUS BIT
        JMP      (VPRCED)

SYIRQ9:  BIT      PBCTL                    ;INTERRUPT ***I GUESS***
        BPL      SYIRQA
        LDA      PORTB                    ;CLEAR INT STATUS
        JMP      (VINTER)

SYIRQA:  PLA
        STA      JVECK
        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
;
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
        PHA
        TYA
        PHA
        STA     NMIRES         ;RESET INTERRUPT STATUS
        JMP     (VVBLKI)       ;JUMP THRU VECTOR
        .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
        INC     ATRACT         ;INCREMENT ATRACT (CAUSES ATRACT WHEN MINUS)
        INC     RTCLOK+1
        BNE     SYSVB1
        INC     RTCLOK
SYSVB1: LDA     #$FE           ;{ATTRACT} SET DARK MASK TO NORMAL
        LDX     #0             ;SET COLRSH TO NORMAL
        LDY     ATRACT         ;TEST ATRACT FOR NEGATIVE
        BPL     VBATRA         ;WHILE POSITIVE DONT GO INTO ATRACT
        STA     ATRACT         ;IN ATRACTI SO STAY BY STA $FE
        LDX     RTCLOK+1       ;COLOR SHIFT FOLLOWS RICLOK+1
        LDA     #$F6           ;SET DARK MASK TO DARK
VBATRA: STA     DRKMSK
        STX     COLRSH
        LDX     #0             ;POINT TO TIMER1
        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

```



```

        LDA    $104,X          ;GET STACKED P
        AND    #$04            ;I BIT
        BEQ    SYSVB3          ;BRANCH IF OK
XXIT:   JMP    XITVBL          ;I WAS SET, EXIT
SYSVB3: LDA    PENV
        STA    LPENV
        LDA    PENH
        STA    LPENH
        LDA    SDLSTH
        STA    DLISTH
        LDA    SDLSTL
        STA    DLISTL
        LDA    SDMCTL
        STA    DMACTL
        LDA    GPRIOR          ;GLOBAL PRIOR
        STA    PRIOR
        LDX    #$08            ;TURN OFF KEYBOARD SPEAKER
        STX    CONSOL
SCOLLP: CLI                    ;DISABLE INTERRUPTS
        LDA    PCOLR0,X        ;LOAD COLOR REGISTERS FROMRAM
        EOR    COLRSH          ;DO COLOR SHIFT
        AND    DRKMSK          ;AND DARK ATTRACT
        STA    COLPM0,X
        DEX
        BPL    SCOLLP
        LDA    CHBAS
        STA    CHBASE
        LDA    CHACT
        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
        BEQ    SYSVBA
        JSR    DCTIMR          ;DECREMENT AND SET FLAG IF NONZERO
        STA    CDTMF3-4,X
SYSVBA: CPX    #8              ;SEE IF DONE ALL 3
        BNE    SYSVBB          ;LOOP
; CHECK DEBOUNCE COUNTER
        LDA    SKSTAT
        AND    #$04            ;KEY DOWN BIT
        BEQ    SYVB6A          ;IF KEY DOWN
; KEY UP SO COUNT IT
        LDA    KEYDEL          ;KEY DELAY COUNTER
        BEQ    SYVB6A          ;IF COUNTED DOWN ALREADY
        DEC    KEYDEL          ;COUNT IT
; CHECK SOFTWARE REPEAT TIMER
SYVB6A: LDA    SRTIMR
        BEQ    SYSVB7          ;DOESN'T COUNT
        LDA    SKSTAT
        AND    #$04            ;CHECK KEY DOWN BIT
        BNE    SYSVB6          ;BRANCH IF NO LONGER DOWN
        DEC    SRTIMR          ;COUNT FRAME OF KEY DOWN
        BNE    SYSVB7          ;BRANCH IF NOT RUN OUT

```

```

; TIMER RAN OUT - RESET AND SIMULATE KEYBOARD IRQ
        LDA     #SRTIM2           ;TIMER VALUE
        STA     SRTIMR           ;SET TIMER
        LDA     KBCODE           ;GET THE KEY
        STA     CH               ;PUT INTO CH
; READ GAME CONTROLLERS
SYSVB7: LDY     #1
        LDX     #3
STLOOP: LDA     PORTA,Y
        LSR     A
        LSR     A
        LSR     A
        LSR     A
        STA     STICK0,X        ;STORE JOYSTICK
        DEX
        LDA     PORTA,Y
        AND     #$F
        STA     STICK0,X        ;STORE JOYSTICK
        DEX
        DEY
        BPL     STLOOP
;
        LDX     #3
STRL:   LDA     TRIG0,X          ;MOVE JOYSTICK TRIGGERS
        STA     STRIG0,X
        LDA     POT0,X          ;MOVE POT VALUES
        STA     PADDL0,X
        LDA     POT4,X
        STA     PADDL4,X
        DEX
        BPL     STRL
        STA     POTGO           ;START POTS FOR NEXT TIME
;
        LDX     #6
        LDY     #3
PTRLP:  LDA     STICK0,Y        ;TRANSFER BITS FROM JOYSTICKS
        LSR     A               ;TO PADDLE TRIGGERS
        LSR     A
        LSR     A
        STA     PTRIG1,X
        LDA     #0
        ROL     A
        STA     PTRIG0,X
        DEX
        DEX
        DEY
        BPL     PTRLP
;
        JMP     (VVBLKD)        ;GO TO DEFERRED VBLANK ROUTINE
SV7H    =       SYSVB7/256
SV7L    =       (-256)*SV7H+SYSVB7
SYSVB6: LDA     #0
        STA     SRTIMR          ;ZERO TIMER
        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
        LDY    CDTMV1+1,X  ;SEE IF BOTH ZERO
        BEQ    DCTXF       ;YES, EXIT NONZERO
        DEC    CDTMV1+1,X  ;DEC HI BYTE
DCTIM1: DEC    CDTMV1,X    ;DEC LO BYTE
        BNE    DCTXF
        LDY    CDTMV1+1,X
        BNE    DCTXF
        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 ALLOW VBLANK TO HAPPEN
        BNE    SETLOP      ;IF THIS IS LINE "7C"
        LDX    INTEMP
        STA    CDTMV1-1,X
        TYA
        STA    CDTMV1-2,X
        RTS

;
; EXIT FROM VERTICAL BLANK
;
XITVBL: PLA                ;UNSTACK Y
        TAY
        PLA                ;UNSTACK X
        TAX
        PLA                ;UNSTACK A
        RTI                ;AND GO BACK FROM WHENCE.

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
;

```

```

;
;
;
;
; 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 ;!!!!!! *****
CASET = $60 ;!!!!!! *****
;
;
;BUS COMMANDS
;
READ = 'R
WRITE = 'W
;STATIS = 'S
;FORMAT = '!'
;
;
; COMMAND AUX BYTES
;
SIDWAY = 'S ;PRINT 18 CHARACTERS SIDEWAYS
NORMAL = 'N ;PRINT 40 CHARACTERS NORMALLY
DOUBLE = 'D ;PRINT 20 CHARACTERS DOUBLE WIDE
PLOT = 'P ;PLOT MODE
;
;
; BUS RESPONSES
;
ACK = 'A ;DEVICE ACKNOWLEDGES INFORMATION
NACK = 'N ;DEVICE DID NOT UNDERSTAND
COMPLT = 'C ;DEVICE SUCCESSFULLY COMPLETED OPERATION
ERROR = 'E ;DEVICE INCURRED AN ERROR IN AN ATTEMPTED OP
;
;
; MISCELLANEOUS EQUATES
;
B192LO = $28 ;19200 BAUD RATE POKEY COUNTER VALUES (LO BY
B192HI = $00 ;(HI BYTE)
B600LO = $CC ;600 BAUD (LO BYTE)
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 PALFLG
WIRGLO = 150 ;WRITE INTER RECORD GAP (IN 1/60 SEC)
RIRGLO = 100 ;READ INTER RECORD GAP (IN 1/60 SEC)
WSIRG = 13 ;SHORT WRITE INTER RECORD GAP

```

```

RSIRG      =          8          ;SHORT READ INTER RECORD GAP
        .ENDIF
        .IF      PALFLG-1
WIRGLO     =          180        ;WRITE INTER RECORD GAP (IN 1/60 SEC)
RIRGLO     =          120        ;READ INTER RECORD GAP (IN 1/60 SEC)
WSIRG      =          15        ;SHORT WRITE INTER RECORD GAP
RSIRG      =          10        ;SHORT READ INTER RECORD GAP
        .ENDIF
WIRGHI     =          0
RIRGHI     =          0
;
NCOMLO     =          $34        ;PIA COMMAND TO LOWER NOT COMMAND LINE
NCOMHI     =          $3C        ;PIA COMMAND TO RAISE NOT COMMAND LINE
MOTRGO     =          $34        ;PIA COMMAND TO TURN ON CASSETTE MOTOR
MOTRST     =          $3C        ;PIA COMMAND TO TURN OFF MOTOR
;
TEMPHI     =          TEMP/256    ;ADDRESS OF TEMP CELL (HI BYTE)
TEMPLO     =          (-256)*TEMPHI+TEMP ; (LO BYTE)
CBUFHI     =          CDEVIC/256  ;ADDRESS OF COMMAND BUFFER (HI BYTE)
CBUFLO     =          (-256)*CBUFHI+CDEVIC ; (LO BYTE)
;
CRETRI     =          13         ;NUMBER OF COMMAND FRAME RETRIES
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
        STA      SOUNDRL        ;INIT POKE ADDRESS FOR QUIET I/O
        STA      SKCTL
;
;
        RTS                      ;RETURN
;
;
;
;
;
;
SIO:    TSX
        STX      STACKP        ;SAVE STACK POINTER
        LDA      #1
        STA      CRITIC
;
        LDA      DDEVIC
        CMP      #CASET
        BNE      NOTCST        ;BRANCH IF NOT CASSETTE
        JMP      CASENT        ;OTHERWISE JUMP TO CASSETTE ENTER
;
; ALL DEVICES EXCEPT CASSETTE ARE INTELLIGENT
;
NOTCST: LDA      #0
        STA      CASFLG        ; INIT CASSETTE FLAG TO NO CASSETTE
;
        LDA      #DRETRI        ;SET NUMBER OF DEVICE RETRIES
        STA      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
        LDA      DDEVIC
        ADC      DUNIT
        ADC      #$FF          ;SUBTRACT 1
        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
        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
        STA      BUFRHI
        STA      BFENHI          ;DONE SETTING UP BUFFER POINTER
;
        LDA      #NCOMLO        ;LOWER NOT COMMAND LINE
        STA      PBCTL
;
        JSR      SENDIN          ;SEND THE COMMAND FRAME TO A SMART DEVICE
;
        LDA      ERRFLG
        BNE      BADCOM          ;BRANCH IF AN ERROR RECEIVED
;
        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
;
        LDA      #CRETRI          ;SET NUMBER OF RETRIES
        STA      CRETRY
;
        JSR      LDPNTR          ;LOAD BUFFER POINTER WITH DCB INFORMATION
;
        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
;
        LDA      #$00
        STA      ERRFLG          ;CLEAR ERROR FLAG
;
        JSR      WAITER          ;SET UP TIMER AND WAIT
        BEQ      DERR            ;BRANCH IF TIME OUT
;
;
; DEVICE DID NOT TIME OUT
;
        BIT      DSTATS
        BVS      MODATA          ;BRANCH IF MORE DATA FOLLOWS

```

```

;
        LDA      ERRFLG
        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 PRECEDED DATA
;
        LDA      TSTAT      ;GET TEMP STATUS
        STA      STATUS      ;STORE IN REAL STATUS
;
;
NOTERR: LDA      STATUS
        CMP      #SUCCESS
        BEQ      RETURN      ;BRANCH IF COMPLETELY SUCCESSFUL
;
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
        STA      ERRFLG      ;CLEAR ERROR FLAG
;
        CLC                  ;LOAD BUFFER POINTER WITH ADDRESS
        LDA      #TEMPLO      ;OF TEMPORARY RAM CELL
        STA      BUFRLO
        ADC      #1
        STA      BFENLO      ;ALSO SET BUFFER END +1 ADDRESS
        LDA      #TEMPHI
        STA      BUFRHI
        STA      BFENHI      ;DONE LOADING POINTER

```



```

;
    LDA    #$FF
    STA    NOCKSM        ;SET NO CHECKSUM FOLLOWS DATA FLAG
;
    JSR     RECEIV        ;GO RECEIVE A BYTE
;
    LDY     #$FF        ;ASSUME SUCCESS
    LDA     STATUS
    CMP     #SUCCES
    BNE     NWOK        ;BRANCH IF IT DID NOT WORK OK
;
;
;
;
WOK:    LDA     TEMP        ;MAKE SURE THE BYTE SUCCESSFULLY RECEIVED
        CMP     #ACK        ;WAS ACTUALLY AN ACK OR COMPLETE
        BEQ     GOOD
        CMP     #COMPLT
        BEQ     GOOD
;
        CMP     #ERROR
        BNE     NOTDER        ;BRANCH IF DEVICE DID NOT SEND BACK
; 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
        BNE     GOOD        ;RETURN WITH OUT SETTING Y = 0
;
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
        STY      XMTDON         ;TRANSMISSION DONE FLAG
;
;
        LDA      (BUFRLO),Y      ;PUT FIRST BYTE FROM BUFFER
        STA      SEROUT          ;INTO THE SERIAL OUTPUT REGISTER
;
;
        STA      CHKSUM          ;PUT IT IN CHECKSUM
;
NOTDON:  LDA      BRKKEY
        BNE      NTBRKO
        JMP      BROKE           ;JUMP IF BREAK KEY PRESSED
;
NTBRKO:  LDA      XMTDON          ;LOOP UNTIL TRANSMISSION IS DONE
        BEQ      NOTDON
;
        JSR      SENDDS          ;DISABLE SENDING
;
        RTS      RETURN
;
;
;
;
; OUTPUT DATA NEEDED INTERRUPT SERVICE ROUTINE
;
ISRDN:   TYA
        PHA                      ;SAVE Y REG ON STACK
;
        INC      BUFRLO          ;INCREMENT DUFFER POINTER
        BNE      NOWRPO
        INC      BUFRHI
;
NOWRPO:  LDA      BUFRLO          ;CHECK IF PAST END OF BUFFER
        CMP      BFENLO
        LDA      BUFRHI          ;HIGH PART
        SBC      BFENHI
        BCC      NOTEND          ;BRANCH IF NOT PAST END OF BUFFER
;
        LDA      CHKSNT
        BNE      RELONE          ;BRANCH IF CHECKSUM ALREADY SENT
;
        LDA      CHKSUM
        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

```

```

        RTI
;
;
NOTEND: LDY    #0
        LDA    (BUFRLO),Y    ;PUT NEXT BYTE FROM BUFFER
        STA    SEROUT        ;INTO THE SERIAL OUTPUT REGISTER
;
        CLC                    ;ADD IT TO CHECKSUM
        ADC    CHKSUM
        ADC    #0
        STA    CHKSUM
;
        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        ;BUFFER FULL FLAG
        STA    RECVDN        ;RECEIVE DONE FLAG
;
;
;
        LDA    #SUCCES
        STA    STATUS        ;SET GOOD STATUS FOR DEFAULT CASE.
        JSR    RECVEN        ;DO RECEIVE ENABLE
        LDA    #NCOMHI        ;COMMAND FRAME HI COMMAND
        STA    PBCTL        ;STORE IN PIA

```

```

CHKTIM: LDA    BRKKEY
        BNE     NTBRK1
        JMP     BROKE          ;JUMP IF BREAK KEY PRESSED
;
NTBRK1: LDA     TIMFLG          ;NO,
        BEQ     TOUT           ;IF TIMEOUT, GO SET ERROR STATUS
        LDA     RECVDN
        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
;
;
;
        LDA     SKSTAT
        STA     SKRES          ;RESET STATUS REGISTER
; ***** 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     #$20
        BNE     NTOVRN        ;BRANCH IF NO OVERRUN ERROR
;
        LDY     #OVRRUN
        STY     STATUS         ;SET OVERRUN ERROR STATUS
;
NTOVRN: LDA     BUFRFL
        BEQ     NOTYET         ;BRANCH IF BUFFER WAS NOTYET FILLED
;
        LDA     SERIN          ;THIS INPUT BYTE 15 THE CHECKSUM
        CMP     CHKSUM
        BEQ     SRETRN         ;BRANCH IF CHECKSUMS MATCH
;
        LDY     #CHKERR
        STY     STATUS         ;SET CHECKSUM ERROR STATUS
;
SRETRN: LDA     #$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
        STA     CHKSUM
;
        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
        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     BFENLO            ;ALSO SET SUFFER END + 1 ADDRESS
;
        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
;
        LDY     #WSIRG                   ;LOAD SHORT WRITE INTER RECORD GAP TIME
        LDA     DAUX2
        BMI     SRTIR0                   ;BRANCH IF SHORT GAP IS DESIRED
;
        LDY     #WIRGLO                  ;SET WRITE IRQ TIME
SRTIR0: LDX     #WIRGHI
        JSR     SETVBX
;
        LDA     #MOTRGO
        STA     PACTL                    ;TURN ON MOTOR
;
TIMIT:  LDA     TIMFLG                    ;LOOP UNTIL DONE
        BNE     TIMIT
;
        JSR     LDPNTR                   ;LOAD BUFFER POINTER WITH DCB INFORMATION
;
        JSR     SEND                     ;SEND A BUFFER
;
        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
        LDA     DAUX2
        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
;
        JSR     BEGIN      ;SET INITIAL BAUD RATE
;
        JSR     RECEIV      ;GO RECEIVE A BLOCK
;
CRETRN: LDA     DAUX2
        BMI     SRTIR2      ;BRANCH IF DOING SHORT INTER RECORD GAPS
; DON'T TURN OFF CASSETTE MOTOR
        LDA     #MOTRST
        STA     PACTL      ;TURN OFF MOTOR
;
SRTIR2: JMP     RETURN      ;GO RETURN
;
;
;
;
;
JTIMER: LDA     #$00
JTADRH  =        JTIMER/256  ;HI BYTE OF JUMP TIMER ROUTINE ADDR
JTADRL  =        (-256)*JTADRH+JTIMER
        STA     TIMFLG      ;SET TIME OUT FLAG
        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
        BNE     NOTCAS      ;BRANCH IF NOT CASSETTE
;
        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
;
        LDA      #$C7        ;MASK OFF PREVIOUS SERIAL BUS INTERRUPT BITS
        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
        AND      SSKCTL
        ORA      #$10        ;SET RECEIVE MODE ASYNCH.
        STA      SSKCTL      ;STORE NEW VALUE TO SYSTEM MASK
        STA      SKCTL      ;STORE TO ACTUAL REGISTER
;
        STA      SKRES      ;RESET SERIAL PORT/KEYBOARD STATUS REGISTER
;
        LDA      #$C7        ;MASK OFF PREVIOUS SERIAL BUS INTERRUPTBITS
        AND      POKMSK
        ORA      #$20        ;ENABLE RECEIVE INTERRUPT
CONTIN: STA      POKMSK      ;STORE NEW VALUE TO SYSTEM MASK
        STA      IRQEN      ;STORE TO ACTUAL. REGISTER
;
;
        LDA      #$28        ;CLOCK CH.3 WITH 1.79 MHZ
        STA      AUDCTL      ;CLOCK CH.4 WITH CH. 3
;
        LDX      #6          ;SET PURE TONES, NO VOLUME
        LDA      #$A8
        LDY      SOUNDNR      ;TEST QUIET I/O FLAG
        BNE      NOISE1      ;NE IS NORMAL (NOISY)
        LDA      #$A0
NOISE1: STA      AUDC1,X
        DEX
        DEX
        BPL      NOISE1
;
        LDA      #$A0
        STA      AUDC3      ;TURN OFF SOUND ON CHANNEL 3
        LDY      DDEVIC
        CPY      #CASET
        BEQ      CAS31      ;BRANCH IF CASSETTE IS DESIRED
        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
;
STTMOT: LDA     DTIMLO         ;GET DEVICE TIME OUT IN 1 SECOND INCR
        ROR     A              ;PUT 6 HI BITS IN X, LO 2 BITS IN Y
        ROR     A
        TAY                      ;TEMP SAVE
        AND     #$3F           ;MASK OFF 2 HI BITS
        TAX                      ;THIS IS HZ BYTE OF TIME OUT
;
        TYA                      ;RESTORE
        ROR     A
        AND     #$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
;
        TYA                      ;IF Y=0, A TIME OUT OR NACK OCCURED
;
        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      ;SAVE ADJUSTED VALUE
        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
SBC      TIMER1+1
TAY                      ;FIND VBLANK COUNT DIFFERENCE
.IF      PALFLG
LDA      #-$9C
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
        LSR      A
        LSR      A
        LSR      A
        ASL      A
        SEC
        SBC      #22      ;ADJUST TABLE INDEX
        TAX      ;DIVIDE INTERVAL BY 4 TO GET TABLE INDEX
        TYA      ;RESTORE ACCUM
        AND      #7
        TAY      ;PULL OFF 3 LO BITS OF INTERVAL
        LDA      #-11
DOINTP:  CLC
        ADC      #11      ;ACCUMULATE INTERPOLATION CONSTANT
        DEY
        BPL      DOINTP      ;INTERPOLATION CONSTANT COMPUTATION DONE?
;
ENINTP:  LDY      #0
        STY      ADDCOR      ;CLEAR ADDITION CORRECTION FLAG
        SEC
        SBC      #7      ;ADJUST INTERPOLATION CONSTANT
        BPL      PLUS
        DEC      ADDCOR
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      ;LARGER 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
        JMP      BROKE          ;JUMP IF BREAK KEY PRESSED
;
NTBRK2: SEI
;
        LDA      TIMFLG
        BNE      OKTIM1          ;BRANCH IF NOT TIMEDOUT
        BEQ      TOUT1          ;BRANCH IF TIME OUT
;
OKTIM1: LDA      SKSTAT
        AND      #$10            ;READ SERIAL PORT
        BNE      BEGIN          ;START BIT?
        STA      SAVIO          ;SAVE SER. DATA IN
        LDX      VCOUNT          ;READ VERTICAL LINECOUNTER
        LDY      RTCLOCK+2      ;READ LO BYTE OF VBLANK CLOCK
        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
        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
        CMP      SAVIO          ;DATA IN CHANGED YET?
        BEQ      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     RTCLOCK+2      ;READ TIMER LO & HI BYTES
    JSR     COMPUT        ;NO, COMPUTE SAUD RATE
    STY     CBAUDL
    STA     CBAUDH         ;SET BAUD RATE INTO RAM CELLS
    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
        STA     (BUFRLO),Y ;STORE '$55' AS FIRST RCV. BUFFER
        INY
        STA     (BUFRLO),Y
        LDA     #$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

;
        LDX     STACKP
        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
;
SEI                    ;THE SETVBL ROUTINE NEEDS THIS TO CUT SHORT
JSR    SETVBV           ;ANY VBLANKS THAT OCCUR
LDA    #1                ;SET FOR TIMER 1
STA    TIMFLG           ;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).
;
;
;
;
;      AUDF+7          BAUD RATE          VCOUNT INTERVAL
;      -----          -
;
;      .WORD $27C      ;1407              56
;      .WORD $2D7      ;1231              64
;      .WORD $332      ;1094              72
;      .WORD $38D      ;985               80
POKTAB: .WORD $3E8      ;895               88
;      .WORD $443      ;820               96
;      .WORD $49E      ;757              104
;      .WORD $4F9      ;703              112
;      .WORD $554      ;656              120
;      .WORD $5AF      ;615              128
;      .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	;394	200
;	.WORD	\$93C	;379	208
;	.WORD	\$997	;365	216
;	.WORD	\$9F2	;352	224
;	.WORD	\$A4D	;339	232
;	.WORD	\$AA8	;328	240
;	.WORD	\$803	;318	248

;  
;  
;  
;  
;

CRNTP3 =\*

\*=\$14

SIOSPR: .BYTE DSKORG-CRNTP3 ;^GSIOL IS TOO LONG

;

.TITLE 'DISK \*\*\*\*\* DISKP.SRC \*\*\*\*\* 3/9/79 \*\*\*\*\* 4:00:00 P.M.'

;  
;  
;  
;  
;  
;

STATVH = DVSTAT/256

STATVL = (-256)\*STATVH+DVSTAT ;STATUS POINTER

;  
;  
;  
;

CONSTANT EQUATES

;

DISKID	=	\$31	;SERIAL BUS DISK I.D.
PUTSEC	=	\$50	;DISK PUT SECTOR DCB COMMAND
; READ	=	\$52	;DISK GET SECTOR DCB COMMAND
; WRITE	=	\$57	;DISK PUT SECTOR WITH READ CHECK DCB COMMAND
STATC	=	\$53	;DISK STATUS DCB COMMAND
FOMAT	=	\$21	;DISK FORMAT DCB COMMAND !!!!! *****
NODAT	=	0	;SIO COMMAND FOR "NO DATA" OPERATION
GETDAT	=	\$40	;SIO COMMAND FOR "DATA FROM DEVICE"
PUTDAT	=	\$80	;SIO COMMAND FOR "DATA TO DEVICE"

;  
;  
;

VECTORS

;

\*=\$E450

;

JMP	DINIT	;DISK INIT. VECTOR
JMP	DSKIF	;DISK INTERFACE ENTRY POINT

;  
;  
;  
;  
;  
;

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
        LDA     DSKTIM
        LDX     DCOMND
        CPX     #FOMAT          ;IS COMMAND A FORMAT COMMAND?
        BEQ     PUTDTO
        LDA     #7               ;NO, SET TIMEOUT TO 7 SECS.
PUTDTO: STA     DTIMLO           ;PUT DISK TIMEOUT IN DCB
        LDX     #GETDAT         ;SET "GET DATA" COMMAND FOR SIO
        LDY     #$80            ;SET BYTE COUNT TO 128
        LDA     DCOMND          ;READ COMMAND IN DCB
        CMP     #WRITE          ;IS COMMAND A "PUT SECTOR" COMMAND?
        BNE     CKSTC
        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          ;SET BUFFER ADDR TO GLOBAL STATUS BUFFER
        LDY     #4              ;YES, SET BYTE COUNT TO 4
PUTCNT: STX     DSTATS          ;PUT STATUS COMMAND FOR SIO IN DCB
        STY     DBYTLO
        LDA     #0
        STA     DBYTHI          ;PUT BYTE COUNT IN DCB
        JSR     SIOV            ;CALL SERIAL I/O.
        BPL     GOODST          ;NO ERROR
        RTS                    ;NO, GO BACK
GOODST: LDA     DCOMND          ;READ THE COMMAND
        CMP     #STATC          ;WAS IT A STATUS COMMAND?
        BNE     PUTBC
        JSR     PUTADR          ;PUT BUFFER ADDR IN TEMP REQ.
        LDY     #2
        LDA     (BUFADR),Y      ;READ DISK TIMEOUT VALUE BYTE OF STATUS
        STA     DSKTIM          ;PUT IT IN DISK TIMEOUT REQ.

```



```

PUTBC:  LDA    DCOMND
        CMP     #FOMAT           ;WAS COMMAND A FORMAT COMMAND?
        BNE     ENDDIF
FMTD:   JSR     PUTADR           ;YES PUT BUFFER, ADDR INTO TEMP REC
        LDY     #$FE           ;SET BUFFER POINTER
TWICE:  INY
        INY                     ;INCR BUFFER POINTER BY 2
RDBAD:  LDA     (BUFADR),Y       ;READ LO BYTE BAD SECTOR DATA
        CMP     #$FF
        BNE     TWICE           ;IS IT "FF" ?
        INY                     ;YES,
        LDA     (BUFADR),Y       ;READ HI BYTE BAD SECTOR DATA
        INY
        CMP     #$FF
        BNE     RDBAD           ;IS IT "FF" ?
        DEY
        DEY                     ;YES
        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
;*****
;
;
;      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
;
OPNOUT  =      $2           ;IOCB OPEN FOR OUTPUT COMMAND
NBUFSZ  =      40           ;PRINT NORMAL BUFFER SIZE
DBUFSZ  =      20           ;PRINT DOUBLE BUFFFER SIZE
SBUFSZ  =      29           ;PRINT SIDEWAYS BUFFER SIZE
PDEVN   =      $40          ;PRINTER DEVICE NUMBER
; STATC  =      $53          ;DCB STATUS COMMAND CODE
WRITEC  =      $57          ;DCB WRITE COMMAND
SPACE   =      $20          ;ASCII SPACE CHAR.
N        =      $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
        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
        STA     PBUFSZ      ;SET BUFFER SIZE TO 4 BYTES
        LDX     PHSTLO
        LDY     PHSTLO+1    ;SET POINTER TO STATUS BUFFER
        LDA     #STATC      ;SETCOMMAND TO "STATUS"
        STA     DCOMND      ;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
        STA     PBPNT      ;CLEAR PRINT BUFFER POINTER
        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
        CPX     PBUFSZ      ;BUFFER POINTERBUFFER SIZE?
        BEQ     BUFFUL
        STX     PBPNT      ;SAVE SUFFER POINTER
        CMP     #CR         ;IS CHAR. = EOL ?
        BEQ     BLFILL      ;IF YES, GO DO BLANK FILL.
        LDY     #SUCCES     ;PUT GOOD STATUS IN Y REQ FOR CIO.
        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
        STA     PBPNT      ;CLEAR PRINT BUFFER POINTER
        LDX     PHCHLO
        LDY     PHCHLO+1    ;SET POINTER TO PRINT BUFFER
        JSR     SETDCB      ;GO SETUP OCR
        JSR     SIOV        ;SEND PRINT COMMAND
        RTS
        ;YES.
;

```

```

;
;
;
;     PRINTER HANDLER CLOSE ROUTINE
;
PHCLOS: JSR     PRMODE           ;GO DETERMINE PRINT MODE
        LDX     PBPNT
        BNE     BLFILL
        LDY     #SUCCES
        RTS

;
;
;
;
;
;
;
;
;
;     S U B R O U T I N E S
;
;
;
;
;     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
        STA     DUNIT           ;SET UNIT NUMBER TO 1
        LDA     #$80             ;DEVICE WILL EXPECT DATA
        LDX     DCOMND
        CPX     #STATC           ;STATUS COMMAND?
        BNE     PSIOC
        LDA     #$40             ;EXPECT DATA FROM DEVICE
PSIOC:  STA     DSTATS           ;SET SIO MODE COMMAND
        LDA     PBUFSZ
        STA     DBYTLO           ;SET LO BYTE COUNT
        LDA     #0
        STA     DBYTHI           ;SET HI BYTE COUNT
        LDA     PTIMOT
        STA     DTIMLO           ;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 ?
        LDX    #NBUFSZ     ;YES, SET NORMAL CHAR. BUFFER SIZE
        BNE    SETBSZ
CDUBL:  CMP    #D
        BNE    CSIDE       ;PRINT DOUBLE?
        LDX    #DBUFSZ     ;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
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)'
CBUFH   =          CASBUF/256
CBUFL   =          (-256)*CBUFH+CASBUF
SRSTA   =          $40      ;SIO READ STATUS
SWSTA   =          $80      ;SIO WRITE STATUS
;MOTRGO =          $34
;MOTRST =          $3C
;
;
DTA     =          $FC      ;DATA RECORD TYPE BYTE
DT1     =          $FA      ;LAST DATA RECORD
EOT     =          $FE      ;END OF TAPE
HDR     =          $FB      ;HEADER
TONE1   =          2        ;CHANGE TO RECORD MODE TONE
TONE2   =          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
;
        *=RBLOKV
        JMP      RBLOK
;
        *=CSOPIV
        JMP      OPINP
;
;
        *=CASORG
;
;
; INIT ROUTINE
;
INIT:    LDA      #$CC
        STA      CBAUDL
        LDA      #$05
        STA      CBAUDH      ;SET CASSET BAUD RATE TO 600
SPECIAL:                                ;THATS ALL FOLKS
        RTS
        .PAGE
;
; OPEN FUNCTION - WITH NO TIMING ADJUST
;
OPENC:   LDA      ICAX2Z      ;GET AX2
        STA      FTYPE      ;SAVE IT FOR FUTURE REFERENCE
        LDA      ICAX1Z
        AND      #$0C      ;IN AND OUT BITS
        CMP      #$04
        BEQ      OPINP
        CMP      #$08      ;SEE IF OPEN FOR OUTPUT
        BEQ      OPOUT
        RTS                ;IF ALREADY OPEN, RETURN LEAVING STATUS=$84
OPINP:   LDA      #0
        STA      WMODE      ;SET READ MODE
        STA      FEOF      ;NO EOF YET
SFH:     LDA      #TONE2      ;TONE FOR PRESS PLAY
        JSR      BEEP      ;GO BEEP
        BMI      OPNRTN      ;IF ERROR DURING BEEP
        LDA      #MOTRGO
        STA      PACTL      ;TURN MOTOR ON
        .IF      PALFLG
        LDY      #$E0
        LDX      #1
        .ENDIF
        .IF      PALFLG-1
        LDY      #$40      ;5-31-79 9 SEC READ LEADER
        LDX      #2
        .ENDIF
        LDA      #3
        STA      CDTMF3
        JSR      SETVBV      ;SET UP YBLANK TIMER
WAITTM:  LDA      CDTMF3
        BNE      WAITTM      ;WAIT FOR MOTOR TO COME UP TO SPEED
        LDA      #$80      ;NEXT BYTE=NO BYTES IN BUFFER
        STA      BPTR
        STA      BLIM
        JMP      OPOK      ;OPEN OK

```

```

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

```

```

        BMI      GBX          ;IF SIO ERRORS, RETURN
        LDA      #0
        STA      BPTR        ;RESET POINTER
        LDX      #$80        ;DEFAULT # BYTES
        LDA      CASBUF+2
        CMP      #EOT
        BEQ      ATEOF        ;IF HEADER, GO READ AGAIN
        CMP      #DT1        ;IF LAST DATA REC
        BNE      NLR
        LDX      CASBUF+130   ;LAST DATA RECORD, GET # BYTES
NLR:     STX      BLIM
        JMP      GBYTE        ;GET NEXT BYTE
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
        INC      BPTR        ;BUMP POINTER
        LDY      #SUCCES      ;OK STATUS
        CPX      #127        ;IF BUFFER FULL
        BEQ      *+3
        RTS
; WRITE OUT THE BUFFER
        LDA      #DTA        ;RECORD TYPE = DATA
        JSR      WSIO SB     ;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
FCAX:    LDA      #MOTRST      ;STOP THE MOTOR IN CASE WAS SHORT IRQ MODE
        STA      PACTL
        RTS
CLWRT:   LDX      BPTR        ;BUFFER POINTER
        BEQ      WTLR        ;IF NO DATA BYTES IN BUFFER, NO DT1 REC
        STX      CASBUF+130   ;WRITE TO LAST RECORD
        LDA      #DT1        ;REC TYPE
        JSR      WSIO SB     ;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      #EOT          ;WRITE EOT RECORD
        JSR      WSIOB
        JMP      FCAX          ;FLAG CLOSED AND EXIT
        .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
        ADC      #30          ; 1 SEC TONE
        .ENDIF
        TAX
WFL:    LDA      #$FF
        STA      CONSOL      ;TURN ONSPEAKER
        LDA      #0
        LDY      #$F0
        DEY
        BNE      *-1
        STA      CONSOL      ;TURN OFF SPEAKER
        LDY      #$F0
        DEY
        BNE      *-1
        CPX      RTCLOK+2     ;SEE IF 1 SEC IS UP YET
        BNE      WFL
        DEC      FREQ         ;COUNT BEEPS
        BEQ      WFAK         ;IF ALL DONE GO WAIT FOR KEY
        TXA
        CLC
        .IF      PALFLG
        ADC      #8
        .ENDIF
        .IF      PALFLG-1
        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
;
SIOB:   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
CSIO:   LDA      #$60            ;CASSET PSEUDO DEVICE
        STA      DDEVIC
        LDA      #0
        STA      DUNIT
        LDA      #35            ;DEVICE TIMEOUT (5/30/79)
        STA      DTIMLO
        LDA      DCOMND          ;GET COMMAND SACK
        LDY      #SRSTA          ;SIO READ STATUS COMMAND
        CMP      #'R
        BEQ      *+4
        LDY      #SWSTA          ;SIC WRITE STATUS COMMAND
        STY      DSTATS          ;SET STATUS FOR SIO
        LDA      FTYPE
        STA      DAUX2           ;INDICATE IF SHORT IRQ MODE
        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
        JSR      SIOSB          ;CALL SIO ON SYSTEM BUFFER
        RTS      AND            ;RETURN
CRNTP6  =*
        *=$14
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
GETCAR  =          $7            ;"GET CHARACTER" CIO COMMAND CODE
PUTCAR  =          $B            ;"PUT CHARACTER" CIO COMMAND CODE
INIMLL  =          $00          ;INITIAL HEM LO LOW BYTE
INIMLH  =          $07          ;INITIAL HEM LO HIGH BYTE
; GOOD  =          $1            ;GOOD STATUS CODE
; WRITE =          $57          ;WRITE COMMAND
; READ  =          $52          ;READ COMMAND
; STATC =          $53          ;STATUS COMMAND
SEX      =          $0            ;SCREEN EDITOR 10CR INDEX
CLS      =          $7D          ;CLEAR SCREEN CODE
CTRLC    =          $92          ;KEYBOARD CODE FOR 'CONTROL C'
EOF      =          $88          ;CASSETTE END OF FILE CODE
LIRQ     =          $0            ;LONG IRQ TYPE CODE
;

```

```

BUFFH   =      (CASBUF+3)/256
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)
;
;          *=$BFFA
CARTCS: .RES      2          ;CARTRIDGE COLD START ADDRESS.
CART:   .RES      1          ;CARTRIDGE AVAILABLE FLAG BYTE.
CARTFG: .RES      1          ;CARTRIDGE FLAG BYTE. BIT 0=FLAG1,
CARTAD: .RES      2          ;2-BYTE CARTRIDGE START VECTOR
;
;
;          CARTRIDGE FLAG ACTION DEFINITIONS
;
;
;          BIT          ACTION IF SET
;
;          7            SPECIAL -- DON'T POWER-UP, JUST RUN CARTRIDGE
;          6-3          NONE
;          2            RUN CARTRIDGE
;          1            NONE
;          0            BOOT DOS
;
;
;          *****
;          NOTE
;          *****
;
;          1.IF BIT2 IS 0, GOTO BLACKBOARD MODE.
;          2.IF BIT0 SET THE DISK WILL BE BOOTED BEFORE ANY
;             OTHER ACTION.
;
;
;
;
;
;
;
;          POWER-UP VECTOR
;
;          *****
;          *=$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
;
;
;TBLLLEN =          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
;
TBLLLEN =          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 IRQ INTERRUPTS
         LDA      COLDST          ;WERE WE IN MIDDLE OF COLDSTART?
         BNE      PWRUP           ;YES, GO TRY IT AGAIN
         LDA      #$FF
         BNE      PWRUP1          ;SET WARM START FLAG

;
;
;
;*****
;      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
         JSR      SPECL           ;CARTRIDGE SPECIAL CASE?
         JSR      HARDI          ;DO HARDWARE INITIALIZATION
         LDA      WARMST          ;IS IT WARMSTART?
         BNE      ZOSRAM          ;YES, ONLY ZERO OS RAM

;
ZERORM:  LDA      #0
         LDY      #WARMST
         STA      RAMLO
         STA      RAMLO+1        ;INITIALIZE RAM POINTER
CLRDRAM: STA      (RAMLO),Y ;CLEAR MEMORY LOC.
         INY
         CPY      #0              ;AT END OF PAGE?
         BNE      CLRDRAM
         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 OF COLDSTART
        BNE      ESTSCM        ;GO AROUND ZOSRAM
;
; 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 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
        STX      TSTDAT        ;CLEAR "B" CART. FLAG
        STX      TRAMSZ        ;CLEAR "A" CART. FLAG
        LDX      RAMSIZ
        CPX      #$90          ;RAM IN "B" CART. SLOT?

```

```

        BCS      ENDBCK
        LDA      CART-$2000    ;NO.
        BNE      ENDBCK        ;CART. PLUGGED INTO "B" SLOT'?
        INC      TSTDAT        ;YES, SET "B" CART, FLAG
        JSR      CBINI         ;INITIALIZE CARTRIDGE "B"
;
ENDBCK: LDX      RAMSIZ
        CPX      #$B0          ;RAM IN "A" CART. SLOT?
        BCS      ENDACK
        LDX      CART          ;NO,
        BNE      ENDACK        ;CART. PLUGGED INTO "A" SLOT?
        INC      TRAMSZ        ;YES, SET "A" CART. FLAG
        JSR      CAINI         ;INITIALIZE CARTRIDGE "A"
;
;
; OPEN SCREEN EDITOR
;
ENDACK: LDA      #3
        LDX      #SEX
        STA      ICCOM,X       ;OPEN I/O COMMAND
        LDA      #OPNL
        STA      ICBAL,X
        LDA      #OPNH
        STA      ICBAH,X       ;SET BUFFER POINTER TO OPEN SCREEN EDITOR
        LDA      #$C
        STA      ICAX1,X       ;SET UP OPEN FOR INPUT/OUTPUT
        JSR      CIOV          ;GO TO CIO
;
        BPL      SCRNOK        ;BR IF NO ERROR
        JMP      PWRUP         ;RETRY PWRUP IF ERROR (SHOULD NEVER HAPPEN!)
SCRNOK: INX
        BNE      SCRNOK        ;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
        ORA      TSTDAT        ;
        BEQ      NOCART        ;NEITHER CARTRIDGE LIVES
        LDA      TRAMSZ        ;"A" CART?
        BEQ      NOA1          ;NO
        LDA      CARTFG        ;GET CARTRIDGE MODE FLAG
NOA1:   LDX      TSTDAT         ;"B" CART?
        BEQ      NOB1          ;NO
        ORA      CARTFG-$2000  ;ADD OTHER FLAG
NOB1:   AND      #1             ;DOES EITHER CART WANT BOOT?
        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          ;NO
        LDA      CARTFG       ;GET CARTRIDGE MODE FLAG
        AND      #4           ;DOES IT WANT TO RUN?
        BEQ      NOA2         ;NO
        JMP      (CARTCS)     ;RUN "A" CARTRIDGE
NOA2:   LDA      TSTDAT       ;"B" CART?
        BEQ      NOCAR2       ;NO
        LDA      CARTFG-$2000 ;GET "B" MODE FLAG
        AND      #4           ;DOES IT WANT TO RUN?
        BEQ      NOCART       ;NO
        JMP      (CARTCS-$2000) ;RUN "B" CARTRIDGE
;
; 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)
        .PAGE
;
;
;
;
;
;
;
;
;
;
;
;
;
;
;
;
;
;
;
;
;

```



```

;
;
;
; 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
      LDA     CART            ;IS IT ROM?
      BNE     ENSPEC          ;NO
      LDA     CARTFG          ;YES,
      BPL     ENSPEC          ;BIT SET?
      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
      BNE     ENDRAM
      EOR     #$FF           ;CONVERT IT BACK
      STA     (RAMLO+1),Y ;RESTORE ORIGINAL RAMDATA
      LDA     TRAMSZ
      CLC
      ADC     #$10
      STA     TRAMSZ          ;INCR. RAM POINTER BY 4K.
      BNE     HOWMCH          ;GO FIND HOW MUCH RAM.
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
;
OSRAM: DEC     BRKKEY          ;TURN OFF BREAK KEY FLAG
      LDA     #.LOW.BRKKY2
      STA     BRKKY

```

```

        LDA        #.HIGH.BRKKY2
        STA        BRKKY+1
        LDA        TRAMSZ            ;READ RAM SIZE IN TEMP. REG.
        STA        RAMSIZ            ;SAVE IT IN RAM SIZE.
        STA        MEMTOP+1        ;INIT. MEMTOP ADDR HI BYTE
        LDA        #0
        STA        MEMTOP            ;INIT. MEMTOP ADDR LO BYTE
        LDA        #INIMLL
        STA        MEMLO
        LDA        #INIMLH
        STA        MEMLO+1        ;INITIALIZE MEMLO ADDR VECTOR
        JSR        EDITRV+$C        ;EDITOR INIT.
        JSR        SCRENV+$C        ;SCREEN INIT.
        JSR        KEYBDV+$C        ;KEYBOARD INIT.
        JSR        PRINTV+$C        ;PRINTER HANDLER INIT
        JSR        CASETV+$C        ;CASSETTE HANDLER INIT
        JSR        CIOINV            ;CIO INIT.
        JSR        SIOINV            ;SIO INIT.
        JSR        INTINV            ;INTERRUPT HANDLER INIT.
        LDA        CONSOL
        AND        #$1
        BNE        NOKEY            ;GAME START KEY DEPRESSED?
        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.
        LDA        #STATC
        STA        DCOMND            ;SET UPSTATUS COMMAND
        JSR        DSKINV            ;GO DO DISK STATUS
        BPL        DOBOOT            ;IS STATUS FROM 510 GOOD?
        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
        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

```

```

        STA      DFLAGS,X      ;STORE IT
        DEX
        BPL      RDBYTE          ;DONE WITH 4 BYTE TRANSFER
        LDA      BOOTAD          ;YES.
        STA      RAMLO
        LDA      BOOTAD+1
        STA      RAMLO+1        ;PUT BOOT ADDR INTO Z. PAGE RAM
        LDA      CASBUF+7
        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
        LDA      RAMLO+1
        ADC      #0
        STA      RAMLO+1        ;INCR BOOT LOADER BUFFER POINTER
        DEC      DBSECT          ;DECR # OF SECTORS.
        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
        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 RECORD, BUT DON'T USE IT.
XBOOT:  JSR      BLOAD          ;GO EXECUTE BOOT LOADER
        BCS      BADDSK          ;IF BAD BOOT, DO IT OVER AGAIN
        JSR      DINI           ;GO INIT. SOFTWARE
        INC      BOOT?          ;SHOW BOOT SUCCESS
        RTS
BLOAD:  CLC
        LDA      BOOTAD
        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
;
;

```

```

;
; 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      ;"PUT TEXT RECORD" COMMAND
        LDA      #$FF
        STA      ICBLI,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         ;SET DRIVE NO. TO DRIVE 0
        JSR      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 CASSETTE BOOT SUCCESFULL?
        BEQ      NOCSB2        ;NO
        JSR      CINI          ;YES, INIT CASSETTE SOFTWARE
NOCSB2: RTS
;
CSBOT2: LDA      CKEY
        BEQ      NOCSBT        ;"C" KEY FLAG SET ?
        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
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 ATASCII 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)
        .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)
        .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    #$FF
        STA    CH
        LDA    MEMTOP+1
        AND    #$F0          ;INSURE 4K PAGE BOUNDARY
        STA    RAMTOP
        LDA    #$40          ;DEFAULT TO UPPER CASE ALPHA AT PWRON
        STA    SHFLOK
        RTS                ;POWER ON COMPLETED
        .PAGE
;
;
; BEGIN DISPLAY HANDLER OPEN PROCESSING
;
DOPEN:  LDA    ICAX2Z          ;GET AUX 2 BYTE
        AND    #$F
        BNE    OPNCOM          ;IF MODE ZERO, CLEAR ICAX1Z
EOPEN:  LDA    ICAX1Z          ;CLEAR "CLR INHIBIT" AND "MXD MODE" BITS
        AND    #$F
        STA    ICAX1Z
        LDA    #0
OPNCOM: STA    DINDEX
        LDA    #$E0          ;INITIALIZE GLOBAL VBLANK RAM
        STA    CHBAS
        LDA    #2
        STA    CHACT
        STA    SDMCTL        ;TURN OFF DMA NEXT VBLANK
        LDA    #SUCCES
        STA    DSTAT        ;CLEAR STATUS
        LDA    #$C0          ;DO IRQEN
        ORA    POKMSK
        STA    POKMSK
        STA    IRQEN
        LDA    #0
        STA    TINDEX        ;TEXT INDEX MUST ALWAYS BE 0
        STA    ADRESS
        STA    SWPFLG
        STA    CRSINH        ;TURN CURSOR ON ATOPEN
        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    COLOR0,X
        DEX
        BPL    DOPEN8
        LDY    RAMTOP          ;DO TXTMSC=$2C40 (IF MEMTOP=3000)
        DEY
        STY    TXTMSC+1
        LDA    #$60
        STA    TXTMSC
        LDX    DINDEX
        LDA    ANCONV,X        ;CONVERT IT TO ANTIC CODE
        BNE    DOPENA          ;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
        JSR    DBSUB
        DEY
        BNE    DOPEN1
        LDA    GPRIOR          ;ICLEAR GTIA MODES
        AND    #$3F
        STA    OPNTMP+1
        TAY
        CPX    #8              ;TEST IF 320X1
        BCC    NOT8
        TXA                    ;GET 2 LOW BITS
        ROR    A
        ROR    A
        ROR    A
        AND    #$C0            ;NOW 2 TOP BITS
        ORA    OPNTMP+1
        TAY
        LDA    #16             ;SUBTRACT 16 MORE FOR PAGE BOUNDARY
        JSR    DBSUB
        CPX    #11             ;TEST MODE 11
        BNE    NOT8            ;IF MODE = 11
        LDA    #6              ;PUT GTIA LUM VALUE INTO BACKGROUND REGISTER
        STA    COLOR4
NOT8:   STY    GPRIOR          ;STORE NEW PRIORITY
        LDA    ADRESS          ;SAVE MEMORY SCAN COUNTER ADDRESS
        STA    SAVMSC
        LDA    ADRESS+1
        STA    SAVMSC+1
VBWAIT: LDA    VCOUNT          ;WAIT FOR NEXT VBLANK BEFORE MESSING
        CMP    #$7A            ;WITH THE DISPLAY LIST
        BNE    VBWAIT
        JSR    DBDEC           ;START PUTTING DISPLAY LIST RIGHT UNDER RAM
        LDA    PAGETB,X        ;TEST IF DISPLAY LIST WILL BE IN TROUBLE
        BEQ    NOMOD           ;OF CROSSING A 256 BYTE PAGE BOUNDARY
        LDA    #$FF            ;IF SO, DROP DOWN A PAGE
        STA    ADRESS
        DEC    ADRESS+1
NOMOD:  LDA    ADRESS          ;SAVE END OF DISPLAY LIST FOR LATER
        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
        STX      OPNTMP
        LDA      #24           ;INITIALIZE BOTSCR
        STA      BOTSCR
        LDA      DINDEX        ;DISALLOW MIXED MODE IF MODE.GE.9
        CMP      #9
        BCS      NOTMXD
        LDA      ICAX1Z        ;TEST MIXED MODE
        AND      #$10
        BEQ      NOTMXD
        LDA      #4
        STA      BOTSCR
        LDX      #2           ;ADD 4 LINES OF TEXT AT BOTTOM OF SCREEN
DOPEN2: LDA      #2
        JSR      STORE
        DEX
        BPL      DOPEN2
        LDY      RAMTOP        ;RELOAD MSC FOR TEXT
        DEY
        TYA
        JSR      STORE
        LDA      #$60
        JSR      STORE
        LDA      #$42
        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
        JSR      STORE
        DEX
        BNE      DOPEN3
        LDA      DINDEX        ;DO THE MESSY 320X1 PROBLEM
        CMP      #8
        BCC      DOPEN5
        LDX      #93           ;GET REMAINING NUMBER OF DLE'S
        LDA      RAMTOP        ;RELOAD MEMORY SCAN COUNTER
        SEC
        SBC      #$10
        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
        LDA      HOLD1
        ORA      #$40

```



```

        JSR      STORE
        LDA      #$70          ;24 BLANK LINES
        JSR      STORE
        LDA      #$70
        JSR      STORE
        LDA      ADRESS        ;SAVE DISPLAY LIST ADDRESS
        STA      SDLSTL
        LDA      ADRESS+1
        STA      SDLSTL+1
        LDA      #$70          ;ADD LAST BLANK LINE ENTRY
        JSR      STORE        ;POSITION ADRESS=SDLSTL-1
        LDA      ADRESS        ;STORE NEW MEMTOP
        STA      MEMTOP
        LDA      ADRESS+1
        STA      MEMTOP+1
        LDA      SAVADR
        STA      ADRESS
        LDA      SAVADR+1
        STA      ADRESS+1
        LDA      SDLSTL+1
        JSR      STORE
        LDA      SDLSTL
        JSR      STORE
        LDA      DSTAT        ;IF ERROR OCURRED ON ALLOCATION, OPEN THE ED
        BPL      DOPEN9
        PHA
        JSR      EOPEN        ;SAVE STATUS
        PLA                ;OPEN THE EDITOR
        TAY                ;RESTORE STATUS
        RTS                ;AND RETURN IT TO CIO

DOPEN9: LDA      ICAX1Z        ;TEST CLEAR INHIBIT BIT
        AND      #$20
        BNE      DOPEN7
        JSR      CLRSCR        ;CLEAR SCREEN
        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 ADDRESS
        LDA      (ADRESS),Y
        AND      DMASK

SHIFTD: LSR      SHFAMT        ;SHIFT DATA DOWN TO LOW BITS
        BCS      SHIFT1
        LSR      A
        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
        JMP  RETUR2
OUTCHE: LDA  ATACHR          ;TEST FOR CARRIAGE RETURN
        CMP  #CR
        BNE  OUTCHB
        JSR  DOCRWS          ;DO CR
        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  A
        ROL  A
        ROL  A
        ROL  A
        AND  #3
        TAX          ;X HAS INDEX INTO ATAIN
        TYA          ;RESTORE ATACHR
        AND  #$9F          ;STRIP OFF COLUMN ADDRESS
        ORA  ATAIN,X        ;OR IN NEW COLUMN ADDRESS
OUTCH2: STA  CHAR
        JSR  CONVRT
        LDA  CHAR
SHIFTU: LSR  SHFAMT          ;SHIFT UP TO PROPER POSITION
        BCS  SHIFT2
        ASL  A
        JMP  SHIFTU
SHIFT2: AND  DMASK
        STA  TMPCHR          ;SAVE SHIFTED DATA
        LDA  DMASK          ;INVERT MASK
        EOR  #$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  DINDEXT          ;GRAPHICS HAVE INVISIBLE CURSOR
        BNE  RETUR1

```

```

        LDX      CRSINH          ;TEST CURSOR INHIBIT
        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
        LDA      ATACHR         ;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
        LDA      BUFCNT         ;ANYTHING IN THE BUFFER?
        BNE      EGETC3        ;YES
        LDA      ROWCRS        ;NO, SO SAVE BUFFER START ADDRESS
        STA      BUFSTR
        LDA      COLCRS
        STA      BUFSTR+1
EGETC1: JSR      KGETCH         ;LET'S FILL OUR BUFFER
        STY      DSTAT          ;SAVE KEYBOARD STATUS
        LDA      ATACHR         ;TEST FOR CR
        CMP      #CR
        BEQ      EGETC2
        JSR      DOSS           ;NO, GO PRINT IT
        JSR      SWAP           ;JSR DOSS DID SWAP SO SWAP BACK
        LDA      LOGCOL        ;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
EGETC7: DEC      BUFCNT         ;AND RETURN TILL BUFCNT=0
        BEQ      EGETC5
        LDA      DSTAT          ;IF ERR, LOOP ON EGETC7 UNTIL BUFR IS ELIPTIE
        BMI      EGETC7
        JSR      GETCH
        STA      ATACHR
        JMP      SWAP           ;AND RETURN WITHOUT TURNING CURSOR BACK ON
EGETC5: JSR      DOCRWS         ;DO REAL CARRIAGE RETURN
        LDA      #CR            ;AND RETURN EOL
        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
DOSS:   JSR      OFFCRS        ;TURN OFF CURSOR
        JSR      TSTCTL       ;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
        BNE      EOUTC6       ;IF NON-0 DISPLAY RATHER THAN EXECUTE IT
        ASL      ESCFLG
        INX
        ;PROCESS CONTROL CHARACTERS
        LDA      CNTRLS,X      ;GET DISPLACEMENT INTO ROUTINE
        STA      ADRESS
        LDA      CNTRLS+1,X    ;GET HIGH BYTE
        STA      ADRESS+1
        JSR      JSRIND        ;DO COMPUTED JSR
        JSR      RETUR2        ;DO CURSOR
        JMP      SWAP          ;ALL DONE SO RETURN THROUGH RETUR1
;
;
;
;
; END SCREEN EDITOR.
;
;
; BEGIN KEYBOARD HANDLER
;
;
;
;
KGETC2: LDA      #$FF
        STA      CH
KGETCH: LDA      ICAX1Z        ;TEST LSB OF AUX1 FOR SPECIAL EDITOR READ MO
        LSR      A
        BCS      GETOUT
        LDA      #BRKABT
        LDX      BRKKEY        ;TEST BREAK
        BEQ      K7            ;IF BREAK, PUT BRKABT IN DSTAT AND CRIN ATA
        LDA      CH
        CMP      #$FF
        BEQ      KGETCH
        STA      HOLDCH        ;SAVE CH FOR SHIFT LOCK PROC
        LDX      #$FF         ;"CLEAR" CH
        STX      CH
        JSR      CLICK         ;DO KEYBOARD AUDIO FEEDBACK (A IS OK)
KGETC3: TAX
        CPX      #$C0          ;TEST FOR CTL & SHIFT TOGETHER
        BCC      ASCCO1
        LDX      #3            ;BAD CODE
ASCCO1: LDA      ATASCI,X
        STA      ATACHR        ;DONE
        CMP      #$80          ;DO NULLS

```

```

        BEQ      KGETC2
        CMP      #$81          ;CHECK ATARI KEY
        BNE      KGETC1
        LDA      INVFLG
        EOR      #$80
        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
        STA      SHFLOK          ;SHIFT BIT
        BNE      KGETC2
K2:      CMP      #$84          ;CNTL CAPS/LOWER
        BNE      K3
        LDA      #$80          ;CNTL BIT
        STA      SHFLOK
        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
        CMP      #$40          ;REGULAR SHIFT AND CONTROL TAKE PRECEDENCE
        BCS      K5              ;OVER LOCK
        LDA      ATACHR          ;TEST FOR ALPHA
        CMP      #$61          ;LOWER CASE A
        BCC      K5              ;NOT ALPHA IF LT
        CMP      #$7B          ;LOWER CASE Z+1
        BCS      K5              ;NOT ALPHA IF GE
        LDA      SHFLOK          ;DO SHIFT/CONTROL LOCK
        BEQ      K5              ;IF NO LOCK. DONT RE-DO IT
        ORA      HOLDCH
        JMP      KGETC3          ;DO RETRY
K5:      JSR      TSTCTL          ;DONT INVERT M58 OF CONTROL CHARACTERS
        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

```

```

        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
        BMI      CRSRL1          ;(IF LMARGN=0, THIS ELIMINATES PROBLEM CASE)
        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
        BCC      COMRE1
        BEQ      COMRE1          ;(CAUSE OLE)
        LDA      LMARGN
        JMP      LFRTCM          ;UNCONDITIONAL TO COMMON STORE
CLRSCR: JSR      PUTMSC
        LDY      #0
        TYA                      ;PUT 0 IN THEAC
CLRSC2: STA      (ADRESS),Y      ;(AC IS ZERO)
        INY
        BNE      CLRSC2
        INC      ADRESS+1
        LDX      ADRESS+1
        CPX      RAMTOP
        BCC      CLRSC2
        LDA      #$FF          ;CLEAN UP LOGICAL LINE BITMAP
CLRSC3: STA      LOGMAP,Y        ;(Y IS ZERO AFTER CLRSC2 LOOP)
        INY
        CPY      #4
        BCC      CLRSC3
HOME:   JSR      COLCR          ;PLACE COLCRS AT LEFT EDGE
        STA      LOGCOL
        STA      BUFSTR+1
        LDA      #0
        STA      ROWCRS
        STA      COLCRS+1
        STA      BUFSTR
        RTS
;
BS:     LDA      LOGCOL          ;BACKSPACE
        CMP      LMARGN
        BEQ      BS1
BSA:    LDA      COLCRS          ;LEFT EDGE?
        CMP      LMARGN
        BNE      BS3            ;NO
        JSR      DELTIM          ;YES, SEE IF LINE SHOULD BE DELETED
BS3:    JSR      CRSRLF
        LDA      COLCRS
        CMP      RMARGN

```

```

        BNE      BS2
        LDA      ROWCRS
        BEQ      BS2
        JSR      CRSRUP
BS2:    LDA      #$20          ;MAKE BACKSPACE DESTRUCTIVE
        STA      ATACHR
        JSR      OUTPLT
BS1:    JMP      DOLCOL        ;AND RETURN
TAB:    JSR      CRSRRT        ;BEGIN SEARCH
        LDA      COLCRS        ;TEST FOR NEW LINE
        CMP      LMARGN
        BNE      TAB1          ;NO
        JSR      DOCR          ;DO CARRIAGE RETURN
        JSR      LOGGET        ;CHECK IF END OF LOGICAL LINE
        BCC      TAB1          ;NO, CONTINUE
        BCS      TAB2          ;(UNCONDITIONAL)
TAB1:    LDA      LOGCOL        ;CHECK FOR TAB STOP
        JSR      BITGET
        BCC      TAB            ;NO, SO KEEP LOOKING
TAB2:    JMP      DOLCOL        ;CULVERT ROW AND COL TO LOGCOL AND RETURN
SETTAB: LDA      LOGCOL
        JMP      BITSET        ;SET BIT IN MAP AND RETURN
CLRTAB: LDA      LOGCOL
        JMP      BITCLR        ;CLEAR " " " " "
INSCHR: JSR      PHACRS
        JSR      GETPLT        ;GET CHARACTER UNDER CURSOR
        STA      INSDAT
        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
        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
        JSR      INCRSB        ;PUT CURSOR OVER NEXT CHARACTER

```

```

        PLA
        CMP     LOGCOL           ;TEST NEW LOGCOL AGAINST OLD LOGCOL
        BCS     DELCH2           ;IF OLD.GE.NEW THEN QUIT
        LDA     ROWCRS           ;IS ROW OFF SCREEN?
        CMP     BOTSCR
        BCS     DELCH2           ;YES, SO QUIT
        JSR     GETPLT           ;GET DATA UNDER CURSOR
        LDY     #0
        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
INSLIN: SEC                     ;INSLIN PUTS "1" INTO BIT MAP
INSLIA: JSR     EXTEND           ;ENTRY POINT FOR C=0
        LDA     LMARGN           ;DO CARRIAGE RETURN (NO LF)
        STA     COLCRS
        JSR     CONVRT           ;GET ADDRESS
        LDA     ADRESS           ;SET UP TO=40+FROM (FROM = CURSOR)
        STA     FRMADR
        CLC
        ADC     #40
        STA     TOADR
        LDA     ADRESS+1
        STA     FRMADR+1
        ADC     #0
        STA     TOADR+1
        LDX     ROWCRS           ;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 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
        BNE     DELLI1           ;LOOP
        LDA     LOGMAP+2         ;SET LSB
        ORA     #1
        STA     LOGMAP+2
DELLI2: LDA     LMARGN           ;DELETE LINE OF DATA USING PART OF SCROLL

```



```

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

```

```

        LDA        MLTTMP+1
        ADC        ADRESS+1
        STA        ADRESS+1
        STA        OLDADR+1
        LDA        DMASKT,X
        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
;
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
INCRS2: LDA        COLCRS            ;TEST END OF LINE
        LDX        DINDEX
        CMP        COLUMN,X          ;TEST TABLED VALUE FOR ALL SCREEN MODES
        BEQ        INC2A            ;DO CR IF EQUAL
        CPX        #0                ;MODE 0?
        BNE        INCRS3            ;IF NOT. JUST RETURN
        CMP        RMARGN            ;TEST AGAINST RMARGN
        BEQ        INCRS3            ;EQUAL IS OK
        BCS        INC2A            ;IF GREATER THAN, DO CR
INCRS3: RTS
INC2A:  CPX        #8                ;CHECK MODE
        BCC        DOCR1            ;NOT 320X1 $O 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
        LDA        LOGCOL            ;TEST LINE OVERRUN
        CMP        #81
        BCC        DOCR1B            ;IF LESS THAN 81 IT IS DEFINITELY NOT LINE 3
        LDA        INSDAT
        BEQ        DOCR              ;ONLY DO LOG LINE OVERFLOW IF INSDAT <>0
        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            ;DON'T EXTEND IF OVERRUN IS INTO MIDDLE OF L
        LDA        INSDAT            ;DON'T EXTEND IF INSDAT IS ZERO
        BEQ        DOCR1A            ;(INSCHR SPECIAL CASE)
        CLC                          ;INSERT "0" INTO BIT MAP
        JSR        INSLIA

```

```

DOCR1A: JMP     DOLCOL           ;CONVERT ROW AND COL TO LOGCOL AND RETURN
NOSCR1: 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
        LDY     #24              ;SET UP SCROLL LOOP COUNTER
        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 IN A 0
        CMP     #$9B           ;TO EXTEND BOTTOM LOGICAL LINE
        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
        DEC     HOLD3
        LDA     LOGMAP
        SEC
        ;FOR PARTIAL LINES ROLL IN A "1"
        BPL     DOCR4B          ;AGAIN IF PARTIAL LOGICAL LINE
        LDA     HOLD3           ;PLACE CURSOR AT NEW LINE NEAR THE BOTTOM
        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

```

```

        BEQ     RANGE      ;THEN IT IS IN MIXED NODE AND OK
        LDA     DINDEX          ;IF MODE = 0
        BEQ     RANGE      ;THEN IT IS INEDITOR MODE AND OK
        JSR     EOPEN          ;IF NOT, OPEN EDITOR
RANGE:  LDA     #39            ;***** RANGE CHECK RMARGN ***** SET UP AC
        CMP     RMARGN         ;***** RANGE CHECK RMARGN ***** COMPARE
        BCS     RANGE3        ;***** RANGE CHECK RMARGN ***** BRANCH GE
        STA     RMARGN         ;***** RANGE CHECK RMARGN ***** BAD SO STORE
RANGE3: LDX     DINDEX
        LDA     NOROWS,X      ;CHECK ROWS
        CMP     ROWCRS
        BCC     RNGERR        ; (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
        BNE     RNGERR        ;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
        LDX     BRKKEY        ;CHECK BREAK KEY FLAG
        STA     BRKKEY        ;'CLEAR' BREAK
        BEQ     RNGER2        ;IF BREAK, QUIT IMMEDIATELY AND RETURN TO CI
        RTS
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
        PLA                      ;AWAY FROM RETURN TO CIO)
        LDA     SWPFLG        ;IF SWAPPED. SWAP BACK
        BPL     RETUR3
        JSR     SWAPA         ;AND DONT DO RETUR1
RETUR3: JMP     RETUR1        ;RETURN TO CIO
;
;
;
; 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
        PLA
        ;PROCESS INDEX
        LSR     A
        LSR     A
        LSR     A
        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
        BEQ     BITGE1
        SEC
BITGE1: RTS

;
;
;
;
; INATAC: INTERNAL(CHAR) TO ATASCII(ATCHR) CONVERSION
;
INATAC: LDA     CHAR
        LDY     DINDEX          ;IF GRAPHICS MODES
        CPY     #3
        BCS     INATA1          ;THEN DON'T CHANGE CHAR

```

```

        ROL      A
        ROL      A
        ROL      A
        ROL      A
        AND      #3
        TAX
        LDA      CHAR
        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
        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
        LDA      TMPCHR
        STA      (ADRESS),Y
        DEY
        BPL      MOVLII1
        LDA      ADRESS+1 ;SET UP FRMADR=LAST LINE
        STA      FRMADR+1
        LDA      ADRESS
        STA      FRMADR
        CLC
        ;ADD 40 TO TOADR
        LDA      TOADR
        ADC      #40
        STA      TOADR
        BCC      MOVLII2
        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 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
        STA      (ADRESS),Y
        INC      ADRESS
        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 BACK ONE
        CLC
        ADC     #40
        STA     LOGCOL
        DEC     HOLD1        ;UP ONE LINE
        JMP     DOLCO1
DOLCO2: CLC                    ;ADD IN COLCRS
        LDA     LOGCOL
        ADC     COLCRS
        STA     LOGCOL
        RTS
;
;
;
; DOBUF1: COMPUTE BUFFER COUNT AS THE NUMBER OF BYTES FROM
;          BUFSTR TO END OF LOGICAL LINE WITH TRAILING SPACES REMOVED
;
DOBUF1: JSR     PHACRS
        LDA     LOGCOL
        PHA
        LDA     BUFSTR        ;START
        STA     ROWCRS
        LDA     BUFSTR+1
        STA     COLCRS
        LDA     #1
        STA     BUFCNT
DOBUF1: LDX     #23            ;NORMAL
        LDA     SWPFLG        ;IF SWAPPED, ROW 3 IS THE LAST LINE ON SCREEN
        BPL     DOB1
        LDX     #3
DOB1:   CPX     ROWCRS        ;TEST IF CURSR IS AT LAST SCREEN POSITION
        BNE     DOBU1A
        LDA     COLCRS
        CMP     RMARGN
        BNE     DOBU1A
        INC     BUFCNT        ;YES, SO FAKE INCRSP TO AVOID SCROLLING
        JMP     DOBUF2
DOBU1A: JSR     INCRSB
        INC     BUFCNT
        LDA     LOGCOL
        CMP     LMARGN
        BNE     DOBUF1        ;NOT YET EOL
        DEC     ROWCRS        ;BACK UP ONE INCRSR
        JSR     CRSRLF
DOBUF2: JSR     GETPLT        ;TEST CURRENT COLUMN FOR NON-ZERO DATA
        BNE     DOBUF4        ;QUIT IF NON-ZERO
        DEC     BUFCNT        ;DECREMENT COUNTER
        LDA     LOGCOL        ;BEGINNING OF LOGICAL LINE YET?
        CMP     LMARGN
        BEQ     DOBUF4        ;YES, SO QUIT
        JSR     CRSRLF        ;BACK UP CURSOR
        LDA     COLCRS        ;IF LOGCOL=RMARGN, GO UP 1 ROW
        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
        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
        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
DELTII1: LDA      (ADRESS),Y
        BNE      DELTI3          ;FOUND A NON-0 SD QUIT AND RETURN
        DEY
        BPL      DELTI1
DELTII2: JMP      DELLIB          ;DELETE A LINE AND RETURN
DELTII3: 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 DOES RETURN
        JMP    RETUR1
SWAPA:  LDA    BOTSCR
        CMP    #24
        BEQ    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

```

```

        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
        STA      ADRESS+1
        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      FILELG
        LDA      ROWCRS          ;PUT CURSOR INTO NEWROW,NEWCOL
        STA      NEWROW
        LDA      COLCRS
        STA      NEWCOL
        LDA      COLCRS+1
        STA      NEWCOL+1
        LDA      #1
        STA      ROWINC          ;SET UP INITIAL DIRECTIONS
        STA      COLINC
        SEC
        LDA      NEWROW          ;DETERMINE DELTA ROW
        SBC      OLDROW
        STA      DELTAR
        BCS      DRAW1          ;DO DIRECTION AND ABSOLUTE VALUE
        LDA      #$FF          ;BORROW WAS ATTEMPTED
        STA      ROWINC          ;SET DIRECTION DOWN
        LDA      DELTAR
        EOR      #$FF          ;DELTAR = |DELTAR|
        CLC
        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
        INC    DELTAC+1
DRAW2:   LDX    #2            ;ZERO RAM FOR DRAW LOOP
        LDY    #0
        STY    COLAC+1
DRAW3A:  TYA
        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
        TAY
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
        ROR    A            ;ROR THE 9 BIT ACUMULATOR
        STA    ROWAC,X
DRAW4A:  LDA    COUNTR        ;TEST ZERO
        ORA    COUNTR+1
        BNE    DRAWI1        ;IF COUNTER IS ZERO, LEAVE DRAW
        JMP    DRAW10
DRAWI1:  CLC                ;ADD ROW TO ROWAC (PLOT LOOP)
        LDA    ROWAC
        ADC    DELTAR
        STA    ROWAC
        BCC    DRAW5
        INC    ROWAC+1
DRAW5:   LDA    ROWAC+1      ;COMPARE ROW TO ENDPOINT
        CMP    ENDPT+1      ;IF HIGH BYTE OF ROW IS .LT. HIGH
        BCC    DRAW6        ;BYTE OF ENDPT, BLT TO COLUMN
        BNE    DRAW5A
        LDA    ROWAC

```

```

        CMP      ENDPT      ;LOW BYTE
        BCC      DRAW6      ;ALSO TILT
DRAW5A: CLC                      ;GE SO MOVE POINT
        LDA      ROWCRS
        ADC      ROWINC
        STA      ROWCRS
        LDX      #0          ;AND SUBTRACT ENDPT FROM ROWAC
        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
DRAW6A: BIT      COLINC      ;+ OR - ?
        BPL      DRAW6B
        DEC      COLCRS      ;DO DOUBLE BYTE DECREMENT
        LDA      COLCRS
        CMP      #$FF
        BNE      DRAW7
        LDA      COLCRS+1
        BEQ      DRAW7      ;DON'T DEC IF ZERO
        DEC      COLCRS+1
        BPL      DRAW7      ;(UNCONDITIONAL)
DRAW6B: INC      COLCRS      ;DO DOUBLE BYTE INCREMENT
        BNE      DRAW7
        INC      COLCRS+1
DRAW7:  LDX      #2          ;AND SUBTRACT ENDPT FROM COLAC
        JSR      SUBEND
DRAW8:  JSR      RANGE
        JSR      OUTPLT      ;PLOT POINT
        LDA      FILFLG      ;TEST RIGHT FILL
        BEQ      DRAW9
        JSR      PHACRS
        LDA      ATACHR
        STA      HOLD4
DRAW8A: LDA      ROWCRS      ;SAVE ROW IN CASE OF CR
        PHA
        JSR      INCRSA      ;POSITION CURSOR ONE PAST DOT
        PLA      ;RESTORE ROWCRS
        STA      ROWCRS
DRAW8C: JSR      RANGE
        JSR      GETPLT      ;GET DATA
        BNE      DRAW8B      ;STOP IF NON-ZERO DATA IS ENCOUNTERED
        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
        LDA    COUNTR+1
        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    23,23,11,23,47,47,95,95,97,97,97,97

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     " "
;      3              8          40X4X8  GRAPHICS
;      4              9          80X2X4      " "
;      5              A          80X4X4      " "
;      4              B          160X2X2     " "
;      7              D          160X4X2     " "
;      8              F          320X2X1     " "
;      9              SAME AS 8 BUT GTIA 'LUM' MODE
;     10              SAME AS 8 BUT GTIA 'COL/LUM REGISTER' MODE
;     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    0,0,0,0,0,0,0,1,1,1,1,1

```

```

;
;
; 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

```



```

        .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
;
;
;
;
;
; ATASCI: ATASCI TO INTERNAL TABLE
;
ATAINT: .BYTE    $40,$00,$20,$60
;
;
; INTATA: INTERNAL TO ATASCI TABLE
;
INTATA: .BYTE    $20,$40,$00,$60
;
;
; ATASCI: ATASCI 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
        BNE     PIRQ3              ;IF NOT, GO PROCESS KEY
        LDA     KEYDEL              ;IF KEY DELAY BYTE > 0
        BNE     PIRQ4              ;IGNORE KEY AS BOUNCE
PIRQ3:  LDA     KBCODE              ;RESTORE AC

```

```

        CMP     #CNTL1           ;TEST CONTROL 1 (SSFLAG)
        BNE     PIRQ1
        LDA     SSFLAG
        EOR     #$FF
        STA     SSFLAG
        BCS     PIRQ4           ;(UNCONDITIONAL) MAKE ^1 INVISIBLE
PIRQ1:  STA     CH
        STA     CH1
        LDA     #3
        STA     KEYDEL         ;INITIALIZE KEY DELAY FOR DEBOUNCE
        LDA     #0             ;CLEAR COLOR SHIFT BYTE
        STA     ATTRACT
PIRQ4:  LDA     #$30
        STA     SRTIMR
PIRQ2:  PLA
        RTI
;
;
        .BYTE   $FF,$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
;
CHORRG  =          $E000       ;CHARACTER SET
VECTBL  =          $E400       ;VECTOR TABLE
VCTABL  =          $E480       ;RAM VECTOR INITIAL VALUE TABLE
CIOORG  =          $E4A6       ;CENTRAL I/O HANDLER
INTORG  =          $E6D5       ;INTERRUPT HANDLER
SIOORG  =          $E944       ;SERIAL I/O DRIVER
DSKORG  =          $EDEA       ;DISK HANDLER
PRNORG  =          $EE78       ;PRINTER HANDLER
CASORG  =          $EF41       ;CASSETTE HANDLER
MONORG  =          $F0E3       ;MONITOR/POWER UP MODULE
KBDORG  =          $F3E4       ;KEYBOARD/DISPLAY HANDLER
;
;

```

```

;
;
;      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
;      6        PUT
;      8        STATUS
;      A        SPECIAL
;      C        JUMP TO POWER ON INITIALIZATION ROUTINE
;      F        NOT USED
;
;
EDITRV  =      $E400      ;EDITOR
SCRENV  =      $E410      ;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.
;
DISKIW  =      $E450      ;DISK INITIALIZATION
DSKINV  =      $E453      ;DISK INTERFACE
CIOV    =      $E456      ;CENTRAL INPUT OUTPUT ROUTINE
SIOV    =      $E459      ;SERIAL INPUT OUTPUT ROUTINE
SETVBV  =      $E45C      ;SET SYSTEM TIMERS ROUTINE
SYSVBV  =      $E45F      ;SYSTEM VERTICAL BLANK CALCULATIONS
XITVBV  =      $E462      ;EXIT VERTICAL BLANK CALCULATIONS
SIOINV  =      $E465      ;SERIAL, INPUT OUTPUT INITIALIZATION
SENDEV  =      $E468      ;SEND ENABLE ROUTINE
INTINV  =      $E46B      ;INTERRUPT HANDLER INITIALIZATION
CIOINV  =      $E46E      ;CENTRAL INPUT OUTPUT INITIALIZATION
BLKBDV  =      $E471      ;BLACKBOARD MODE
WARMSV  =      $E474      ;WARM START ENTRY POINT
COLDSV  =      $E477      ;COLD START ENTRY POINT
RBLOKV  =      $E47A      ;CASSETTE READ BLOCK ENTRY POINT VECTOR
CSOPIV  =      $E47D      ;CASSETTE OPEN FOR INPUT VECTOR
;VCTABL = $E480
;
;
; OPERATING SYSTEM EQUATES
;
; COMMAND CODES FOR IOCB
OPEN    =      3          ;OPEN FOR INPUT/OUTPUT
GETREC  =      5          ;GET RECORD (TEXT)
GETCHR  =      7          ;GET CHARACTER(S)
PUTREC  =      9          ;PUT RECORD (TEXT)
PUTCHR  =      $B         ;PUT CHARACTER(S)
CLOSE   =      $C         ;CLOSE DEVICE

```

```

STATIS  =          $D          ;STATUS REQUEST
SPECIL  =          $E          ;BEGINNING OF SPECIAL ENTRY COMMANDS
;
; SPECIAL ENTRY COMMANDS
DRAWLN  =          $11          ;DRAW LINE
FILLIN  =          $12          ;DRAW LINE WITH RIGHT FILL
RENAME  =          $20          ;RENAME DISK FILE
DELETE  =          $21          ;DELETE DISK FILE
FORMAT  =          $22          ;FORMAT
LOCKFL  =          $23          ;LOCK FILE TO READ ONLY
UNLOCK  =          $24          ;UNLOCK LOCKED FILE
POINT   =          $25          ;POINT SECTOR
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 (
DIRECT  =          $2           ;OPEN FOR DIRECTORY ACCESS (D)
OPNIN   =          $4           ;OPEN FOR INPUT (ALL DEVICES)
OPNOT   =          $8           ;OPEN FOR OUTPUT (ALL DEVICES)
OPNINO  =          OPNIN+OPNOT  ;OPEN FOR INPUT AND OUTPUT (ALL DEVICES)
MXDMOD  =          $10          ;OPEN FOR MIXED MODE (E,S)
INSCLR  =          $20          ;OPEN WITHOUT CLEARING SCREEN (E,S)
;
; DEVICE NAMES
SCREDT  =          'E           ;SCREEN EDITOR (R/W)
KBD     =          'K           ;KEYBOARD (R ONLY)
DISPLY  =          'S           ;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      =          $9B
;
;
; OPERATING SYSTEM STATUS CODES
;
SUCCES  =          $01          ;SUCCESSFUL OPERATION
;
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
RONLY   =          $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
OVRRUN  =          $8E          ;SERIAL BUS DATA OVERRUN
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 ;TEMPORARY REGISTER FOR RAM SIZE
TSTDAT: .RES 1 ;RAM TEST DATA REGISTER
;
; CLEARED ON COLOSTART ONLY
WARMST: .RES 1 ;WARM START FLAG
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
POKMSK: .RES 1 ;SYSTEM MASK FOR POKEY IRQ ENABLE
BRKKEY: .RES 1 ;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
;
DSKFMS: .RES 2 ;DISK FILE MANAGER POINTER
DSKUTL: .RES 2 ;DISK UTILITIES POINTER
;
PTIMOT: .RES 1 ;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
IOCBSZ = 16 ;NUMBER OF BYTES PER IOCB
MAXIOC = 8*IOCBSZ ;LENGTH OF THE IOCB AREA
IOCBAS =*
ICHIDZ: .RES 1 ;HANDLER INDEX NUMBER (FF = IOCB FREE)
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
ICPTLZ: .RES 1 ;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
ICSPRZ: .RES      4          ;TWO SPARE BYTES (CIO LOCAL USE)
ICIDNO  =          ICSPRZ+2  ;IOCB NUMBER X 16
CIOCHR  =          ICSPRZ+3  ;CHARACTER BYTE FOR CURRENT OPERATION
;
STATUS: .RES      1          ;INTERNAL STATUS STORAGE
CHKSUM: .RES      1          ;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
CRETRY: .RES      1          ;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      1
FTYPE:   .RES      1
FEOF:    .RES      1
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
;
ATTRACT: .RES      1          ;ATTRACT FLAG
DRKMSK:  .RES      1          ;DARK ATTRACT MASK
COLRSH:  .RES      1          ;ATTRACT 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      1
SAVMSC:  .RES      2
OLDROW:  .RES      1
OLDCOL:  .RES      2
OLDCHR:  .RES      1          ;DATA UNDER CURSOR
OLDADR:  .RES      2
NEWROW:  .RES      1          ;POINT DRAW GOES TO
NEWCOL:  .RES      2
LOGCOL:  .RES      1          ;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
BUFCNT:  .RES       1          ;BUFFER COUNT
BUFSTR:  .RES       2          ;EDITOR GETCH POINTER
BITMSK:  .RES       1          ;BIT MASK
SHFAMT:  .RES       1
ROWAC:   .RES       2
COLAC:   .RES       2
ENDPT:   .RES       2
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
INSDAT:  .RES       1
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
VDSLST: .RES       2          ;DISPLAY LIST NMI VECTOR
VPRCED: .RES       2          ;PROCEED LINE IRQ VECTOR
VINTER: .RES       2          ;INTERRUPT LINE IRQ VECTOR
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
VSEROC: .RES       2          ;POKEY SERIAL OUTPUT COMPLETE IRQ
VTIMR1: .RES       2          ;POKEY TIMER 1 IRQ
VTIMR2: .RES       2          ;POKEY TIMER 2 IRQ
VTIMR4: .RES       2          ;POKEY TIMER 4 IRQ
VIMIRU: .RES       2          ;IMMEDIATE IRQ VECTOR
CDTMV1: .RES       2          ;COUNT DOWN TIMER 1
CDTMV2: .RES       2          ;COUNT DOWN TIMER 2
CDTMV3: .RES       2          ;COUNT DOWN TIMER 3
CDTMV4: .RES       2          ;COUNT DOWN TIMER 4
CDTMV5: .RES       2          ;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      2          ;COUNT DOWN TIMER 2 JSR ADDRESS
CDTMF3: .RES      1          ;COUNT DOWN TIMER 3 FLAG
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
CDTMF5: .RES      1          ;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
DBSECT: .RES      1          ;NUMBER OF DISK BOOT SECTORS
BOOTAD: .RES      2          ;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      1
PADDL4: .RES      1
PADDL5: .RES      1
PADDL6: .RES      1
PADDL7: .RES      1
STICK0: .RES      1          ;JOYSTICK 0 RAM CELL
STICK1: .RES      1
STICK2: .RES      1
STICK3: .RES      1
PTRIG0: .RES      1          ;PADDLE TRIGGER 0
PTRIG1: .RES      1
PTRIG2: .RES      1
PTRIG3: .RES      1
PTRIG4: .RES      1
PTRIG5: .RES      1

```

```

PTRIG6: .RES      1
PTRIG7: .RES      1
STRIG0: .RES      1      ;JOYSTICK TRIGGER 0
STRIG1: .RES      1
STRIG2: .RES      1
STRIG3: .RES      1
;
CSTAT:  .RES      1
WMODE:  .RES      1
BLIM:   .RES      1
IMASK:  .RES      1
JVECK:  .RES      2
;
          .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 & OLD COL FOR TEXT (AND THEN SOME)
TMPX1:  .RES      1
HOLD3:  .RES      1
SUBTMP: .RES      1
HOLD2:  .RES      1
DMASK:  .RES      1
TMPLBT: .RES      1
ESCFLG: .RES      1      ;ESCAPE FLAG
TABMAP: .RES     15
LOGMAP: .RES      4      ;LOGICAL LINE START BIT MAP
INVFLG: .RES      1      ;INVERSE VIDEO FLAG (TOGGLED BY ATARI KEY)
FILFLG: .RES      1      ;RIGHT FILL FLAG FOR DRAW
TMPROW: .RES      1
TMPCOL: .RES      2
SCRFLG: .RES      1      ;SET IF SCROLL OCCURS
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
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      1
COLOR4: .RES      1
;
;
          .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
CBAUDL: .RES      1          ;CASSETTE BAUD RATE LOW BYTE
CBAUDH: .RES      1
;
CRSINH: .RES      1          ;CURSOR INHIBIT (00 = CURSOR ON)
KEYDEL: .RES      1          ;KEY DELAY
CH1:    .RES      1
;
CHACT:  .RES      1          ;CHACTL REGISTER RAM
CHBAS:  .RES      1          ;CHBAS REGISTER RAM
;
;          .RES      5          ;SPARE BYTES
;
CHAR:   .RES      1
ATACHR: .RES      1          ;ATASCII CHARACTER
CH:     .RES      1          ;GLOBAL VARIABLE FOR KEYBOARD
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      1          ;PERIPHERAL UNIT 1 BUS ID. NUMBER
DUNIT:   .RES      1          ;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      1          ;DEVICE TIME OUT IN 1 SECOND UNITS
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
TIMFLG:  .RES      1          ;TIME OUT FLAG FOR BAUD RATE CORRECTION

```

```

STACKP: .RES      1           ;SIO STACK POINTER SAVE CELL
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      1           ;HANDLER INDEX NUMBER (FF = IOCB FREE)
ICDNO:  .RES      1           ;DEVICE NUMBER (DRIVE NUMBER)
ICCOM:  .RES      1           ;COMMAND CODE
ICSTA:  .RES      1           ;STATUS OF LAST IOCB ACTION
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      1           ;
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
;
FPREC      =          6              ;FLOATING PT PRECISION (# OF BYTES)
; IF CARRY USED THEN CARRY CLEAR => NO ERROR, CARR
AFP        =          $D800          ;ASCII->FLOATING POINT(FP)
;
;                                INBUFF+CIX -> FR0, CIX, CARRY
FASC       =          $D8E6          ;FP -> ASCII FR0 -> LBUFF (INBUFF)
IFP        =          $D9AA          ;INTEGER -> FP
;
;                                0-$FFFF (LSB,MSB) IN FR0,FR0+1->FR0
FPI        =          $D9D2          ;FP -> INTEGER FR0 -> FR0,FR0+1, CARRY
FSUB       =          $DA60          ;FR0 <- FR0 - FR1 ,CARRY
FADD       =          $DA66          ;FR0 <- FR0 + FR1 ,CARRY
FMUL       =          $DADB          ;FR0 <- FR0 * FR1 ,CARRY
FDIV       =          $DB28          ;FR0 <- FR0 / FR1 ,CARRY
FLD0R     =          $DD89          ;FLOATING LOAD REG0      FR0 <- (X,Y)
FLD0P     =          $DD80          ; " " " " FR0 <- (FLFTR)
FLD1R     =          $DD98          ; " " " REG1      FR1 <- (X,Y)
FLD1P     =          $DD9C          ; " " " " FR1 <- (FLPTR)
FSTOR     =          $DDA7          ;FLOATING STORE REG0 (X,Y) <- FR0
FSTOP     =          $DDAB          ; " " " " (FLPTR) <- FR0
FMOVE     =          $DDB6          ;FR1 <- FR0
PLYEVL    =          $DD40          ;FR0 <- P(Z) = SUM(I=N TO 0) (A(I)*Z**I) CAR
;
;                                INPUT: (X,Y) = A(N),A(N-1)...A(0) -> PLYARG
;
;                                ACC      = # OF COEFFICIENTS = DEGREE+1
;
;                                FR0      = Z
EXP        =          $DDC0          ;FR0 <- E**FR0 = EXP10(FR0 * LOG10(E)) CARRY
EXP10     =          $DDCC          ;FR0 <- 10**FR0 CARRY
LOG        =          $DECD          ;FR0 <- 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
COS        =          $BD73          ;FR0 <- COS(FR0) CARRY
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
FR0:       .RES      FPREC          ;FP REG0
FRE:       .RES      FPREC
FR1:       .RES      FPREC          ;FP REG1
FR2:       .RES      FPREC
FRX:       .RES      1              ;FP SPARE
EEXP:     .RES      1              ;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
CIX:       .RES      1              ;CURRENT INPUT INDEX
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
DEGON      =          6              ;INDICATES DEGREES

```

```

FLPTR: .RES      2                ;POINTS TO USER'S FLOATING PT NUMBER
FPTR2: .RES      2
; FLOATING PT ROUTINES' NON-ZERO PAGE RAM
; (NEEDED ONLY IF F.P. ROUTINES CALLED)
      *=$57E
LBPR1: .RES      1                ;LBUFF PREFIX 1
LBPR2: .RES      1                ;LBUFF PREFIX 2
LBUFF: .RES     128                ;LINE BUFFER
PLYARG =          LBUFF+$60      ;POLYNOMIAL ARGUMENTS
FPSCR  =          PLYARG+FPREC
FPSCR1 =          FPSCR+FPREC
FSCR   =          FPSCR
FSCR1  =          FPSCR1
LBFEND =          *-1             ;END OF LBUFF
;
;
;
;
;
;
;
;
;      COLLEEN MNEMONICS
;
POKEY  =          $D200           ;VBLANK ACTION:      DESCRIPTION:
POT0   =          POKEY+0         ;POT0-->PADDL0      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   =          POKEY+3         ;POT3-->PADDL3      0-227 IN RAM CELL
POT4   =          POKEY+4         ;POT4-->PADDL4      0-227 IN RAM CELL
POT5   =          POKEY+5         ;POT5-->PADDL5      0-227 IN RAM CELL
POT6   =          POKEY+6         ;POT6-->PADDL6      0-227 IN RAM CELL
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 =          POKEY+8         ;NONE      AUDCTL<--[SIO]
STIMER =          POKEY+9
SKRES  =          POKEY+10        ;NONE      SKRES<--[SIO]
SEROUT =          POKEY+13        ;NONE      SEROUT<--[SIO]
IRQEN  =          POKEY+14        ;POKMSK-->IRQEN (AFFECTED BY OPEN S: OR E:)
SKCTL  =          POKEY+15        ;SSKCTL-->SKCTL      SSKCTL<--[SIO]
;
CTIA   =          $D000           ;VBLANK ACTION:      DESCRIPTION:
HPOSP0 =          CTIA+0
HPOSP1 =          CTIA+1

```

HPOSP2	=	CTIA+2		
HPOSP3	=	CTIA+3		
HPOSM0	=	CTIA+4		
HPOSM1	=	CTIA+5		
HPOSM2	=	CTIA+6		
HPOSM3	=	CTIA+7		
SIZEP0	=	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		
COLPM0	=	CTIA+18	;PCOLR0-->COLPM0	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
COLPF1	=	CTIA+23	;COLOR1-->COLPF1	WITH ATTRACT MODE
COLPF2	=	CTIA+24	;COLOR2-->COLPF2	WITH ATTRACT MODE
COLPF3	=	CTIA+25	;COLOR3-->COLPF3	WITH ATTRACT MODE
COLBK	=	CTIA+26	;COLOR4-->COLBK	WITH ATTRACT MODE
PRIOR	=	CTIA+27	;(ON OPEN S: OR E:)	GPRIOR-->PRIOR
VDELAY	=	CTIA+28		
GRCTL	=	CTIA+29		
HITCLR	=	CTIA+30		
CONSOL	=	CTIA+31	;\$08-->CONSOL	TURN OFF SPEAKER
M0PF	=	CTIA+0		
M1PF	=	CTIA+1		
M2PF	=	CTIA+2		
M3PF	=	CTIA+3		
P0PF	=	CTIA+4		
P1PF	=	CTIA+5		
P2PF	=	CTIA+8		
P3PF	=	CTIA+7		
M0PL	=	CTIA+8		
M1PL	=	CTIA+9		
M2PL	=	CTIA+10		
M3PL	=	CTIA+11		
P0PL	=	CTIA+12		
P1PL	=	CTIA+13		
P2PL	=	CTIA+14		
P3PL	=	CTIA+15		
TRIG0	=	CTIA+16	;TRIG0-->STRIG0	
TRIG1	=	CTIA+17	;TRIG1-->STRIG1	
TRIG2	=	CTIA+18	;TRIG2-->STRIG2	
TRIG3	=	CTIA+19	;TRIG3-->STRIG3	
;				
ANTIC	=	\$D400	;VBLANK ACTION	DESCRIPTION
DMACTL	=	ANTIC+0	;DMACTL<--SDMCTL	ON OPEN S: OR E:
CHACTL	=	ANTIC+1	;CHACTL<--CHACT	ON OPEN S: OR E:
DLISTL	=	ANTIC+2	;DLISTL<--SDLSTL	ON OPEN S: OR E:
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 = ANTIC+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
PIA = $D300 ;VBLANK ACTION DESCRIPTION
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 = $9B ;END OF RECORD
        .PAGE
;
; CIO JUMP VECTOR FOR USERS
        *=CIOV
        JMP CIO ;GO TO CIO
;
; CIO INIT JUMP VECTOR FOR POWER UP
        *=CIOINV
        JMP CIOINT ;GO TO INIT
;
;
; ERROR ROUTINE ADDRESS EQUATE
; ERRTNH =ERRTN/256 "MOVED TO LINE 788"
; ERRTNL =-ERRTNH*256+ERRTN "MOVED TO LINE 789"
;
;
        *=CIOORG
;
; CIO INITIALIZATION (CALLED BY MONITOR AT POWER UP)
CIOINT: LDX #0
CIOI1: LDA #IOCFRE ;SET ALL IOCB'S TO FREE
        STA ICHID,X ;BY SETTING HANDLER ID'S=$FF
        LDA #ERRTNL
        STA ICPTL,X ;POINT PUT TO ERROR ROUTINE
        LDA #ERRTNH
        STA ICPH,X
        TXA
        CLC
        ADC #IOCBSZ ;BUMP INDEX BY SIZE
        TAX
        CMP #MAXIOC ;DONE?
        BCC CIOI1 ;NO
        RTS ;YES, RETURN
;
; 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
            AND      #$F          ;IS IOCB MULTIPLE OF 16?
            BNE      CIERR1       ;NO, ERROR
            CPX      #MAXIOC      ;IS INDEX TOO LARGE?
            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
            CPY      #12          ;12 BYTES
            BCC      IOC1A
;
; COMPUTE CIO INTERNAL VECTOR FOR COMMAND
            LDY      #NVALID      ;ASSUME INVALID CODE
            LDA      ICCOMZ       ;COMMAND CODE TO INDEX
            CMP      #OPEN        ;IS COMMAND LEGAL?
            BCC      CIERR4       ;NO
            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
            CMP      #2           ;IS IT CLOSE?
            BEQ      CICLOS       ;YES
            CMP      #8           ;IS IT STATUS OR SPECIAL?
            BCS      CISTSP       ;YES
            CMP      #4           ;IS IT READ?
            BEQ      CIREAD       ;YES
            JMP      CIWRIT       ;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
;
; 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
        JSR     COMENT         ;COMPUTE HANDLER ENTRY POINT
        BCS     CICLO2         ;GO IF ERROR IN COMPUTE
        JSR     GOHAND         ;GO TO HANDLER TO CLOSE DEVICE
CICLO2: LDA     #IOCFRE        ;GET IOCB "FREE" VALUE
        STA     ICHIDZ         ;SET HANDLER ID
        LDA     #ERRTNH
        STA     ICPTHZ         ;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
        JSR      GOHAND          ;GO TO HANDLER
;
; RESTORE HANDLER INDEX (DO IMPLIED CLOSE)
        LDX      ICIDNO          ;IOCB INDEX
        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           ;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
        JSR      INCBFP          ;INCREMENT BUFFER POINTER
        LDA      ICCOMZ          ;GET COMMAND CODE
        AND      #2             ;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
RCI1:   JSR      DECBFL          ;DECREMENT BUFFER LENGTH
        BNE      RCI3           ;CONTINUE IF NON ZERO
        .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
;
; LOOP TO WAIT FOR EOL
RCI6:   JSR   GOHAND           ;GET BYTE FROM HANDLER
        STA   CIOCHR           ;SAVE CHARACTER
        BMI   RCI4             ;GO IF ERROR
;
; 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
RCI11:  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   WCI1A           ;YES
;
; ILLEGAL WRITE -- DEVICE OPENED FOR READ ONLY
        LDY   #RDONLY          ;ERROR CODE
WCI1B:  JMP   CIRTN1           ;RETURN
;
; COMPUTE AND CHECK ENTRY POINT
WCI1A:  JSR   COMENT           ;COMPUTE HANDLER ENTRY POINT
        BCS   WCI1B           ;GO IF ERROR IN COMPUTE
;
; PUT RECORD OR CHARACTERS
        LDA   ICBLLZ
        ORA   ICBLLZ+1         ;IS BUFFER LENGTH ZERO?
        BNE   WCI3             ;NO
        LDA   CIOCHR           ;GET CHARACTER
        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 BUFFER POINTER
;
; CHECK FOR TEXT RECORD
        LDA   ICCOMZ           ;GET COMMAND BYTE
        AND   #2               ;IS IT PUT RECORD?
        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
        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     ICBAHZ
        LDX     #0               ;LOOP COUNT AND INDEX
CIRT3:   LDA     IOCBAS,X         ;ZERO PAGE
        STA     IOCB,Y           ;TO USER AREA
        INX
        INY
        CPX     #12              ;12 BYTES
        BCC     CIRT3
;
; RESTORE A,X, & Y
        LDA     CIOCHR           ;GET LAST CHARACTER
        LDX     ICIDNO           ;IOCB INDEX
        LDY     ICSTAZ           ;GET STATUS AND SET FLAGS
        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
;

```

```

; 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
        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
        LDA      (ICSPRZ),Y    ;GET LOW BYTE OF VECTOR FROM
        TAX                        ;HANDLER ITSELF AND SAVE
        INY
        LDA      (ICSPRZ),Y    ;GET HI BYTE OF VECTOR
        STA      ICSPRZ+1
        STX      ICSPRZ        ;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      ICBL LZ      ;DECREMENT LOW BYTE
        LDA      ICBL LZ      ;CHECK IT
        CMP      #$FF        ;DID IT GO BELOW?
        BNE      DECBF1      ;NO
        DEC      ICBL LZ+1    ;DECREMENT HI BYTE
DECBF1: ORA      ICBL LZ+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      ICBL LZ,X    ;GET LOW BYTE OF INITIAL LENGTH
        SBC      ICBL LZ      ;SUBTRACT FINAL LOW BYTE
        STA      ICBL LZ      ;AND SAVE BACK
        LDA      ICBLH,X      ;GET HI BYTE
        SBC      ICBL LZ+1
        STA      ICBLH Z
        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
        PHA      ;PUT ON STACK
        LDA      ICSPRZ    ;GET JUMP ADDRESS LO BYTE
        PHA      ;PUT ON STACK
        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
        LDA      (ICBALZ),Y ;GET DEVICE NAME FROM USER
        BEQ      CIERR2
        LDY      #MAXDEV    ;INITIAL COMPARE INDEX
DEVS1:  CMP      HATABS,Y    ;IS THIS THE DEVICE?
        BEQ      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
        CMP      #$A        ;IS NUMBER IN RANGE?
        BCC      DEVS3      ;YES
        LDA      #1        ;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      0,4,4,4,4,6,6,6,6,2,8,10

LENGTH  =*-CIOINT
CRNTP1  =*
        *=$14

```

```

CIOSPR: BYTE      INTORG-CRNTPL ;^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    SYIRQB            ;VINTER
        .WORD    SYIRQB            ;VBREAK
;
        .RES      8
        .WORD    SYIRQB            ;VTMIR1
        .WORD    SYIRQB            ;VTIMR2
        .WORD    SYIRQB            ;VTMIR4
        .WORD    SYIRQ             ;VIMIRQ
        .WORD    0,0,0,0,0         ;CDTMV1-4

        .WORD    SYSVBL            ;VVBLKI
        .WORD    XITVBL            ;VVSLKD
;
        *=$900C
;
        LDA      #PIRQH            ;SET UP RAM VECTORS FOR LINBUG VERSION
        STA      $FFF9
        LDA      #PIRQL
        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      NMIEH            ;ENABLE DISPLAY LIST, VERTICAL BLANK
        LDA      #$38              ;LOOK AT DATA DIRECTION REGISTERS IN PIA
        STA      PACTL
        STA      PBCTL

```



```

        LDA        #0                ;MAKE ALL INPUTS
        STA        PORTA
        STA        PORTB
        LDA        #$3C              ;BACK TO PORTS
        STA        PACTL
        STA        PBCTL
        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      VSEROC-INTABS
        .BYTE      VSEROR-INTABS
        .BYTE      VSERIN-INTABS

SYIRQ:  PHA                        ;SAVE ACCUMULATOR
        LDA        IRQST            ; CHECK FOR SERIAL IN
        AND        #$20
        BNE        SYIRQ2
        LDA        #$DF              ; MASK ALL OTHERS
        STA        IRQEN
        LDA        POKMSK
        STA        IRQEN
        JMP        (VSERIN)
SYIRQ2: TXA                        ;PUT X INTO ACC
        PHA                        ;SAVE K ONTO STACK
        LDX        #$6                ;START WITH SIX OFFSET
LOOPM:  LDA        CMPTAB,X          ;LOAD MASK
        CPX        #5                ;CHECK TO SEE IF COMPLETE IS SET
        BNE        LOOPM2
        AND        POKMSK            ;IS THIS INTERRUPT ENABLED?
        BEQ        LL
LOOPM2: BIT        IRQST            ; IS IT THE INTERRUPT?
        BEQ        JMPP
LL:     DEX                        ;NO DEC X AND TRY NEXT MASK
        BPL        LOOPM            ;IF NOT NEC 0010 LOOPH
        JMP        SYIRQ8            ;DONE BUT NO INTERRUPT
JMPP:   EOR        #$FF              ;COMPLEMENT MASK
        STA        IRQEN            ;ENABLE ALL OTHERS
        LDA        POKMSK            ; GET POKE MASK
        STA        IRQEN            ; 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
        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
        JMP      (VPRCED)
SYIRQ9: BIT      PBCTL                     ;INTERRUPT ***I GUESS***
        BPL      SYIRQA
        LDA      PORTB                     ;CLEAR INT STATUS
        JMP      (VINTER)
SYIRQA: PLA
        STA      JVECK
        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
;
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
        STA      NMIRES                     ;RESET INTERRUPT STATUS
        JMP      (VVBLKI)                   ;JUMP THRU VECTOR
        .PAGE
;
; SYSTEM VBLANK ROUTINE
;
; INC FRAME COUNTER. PROCESS COUNTDOWN TIMERS. EXIT IF I WAS SET. CLEAR
; SET DLSTL, DLSTH, DMACTL FROM RAM CELLS. DO SOFTWARE REPEAT.

```

```

;
SYSVBL: INC    RTCLOCK+2    ;INC FRAME COUNTER
        BNE     SYSVB1
        INC     ATTRACT      ;INCREMENT ATTRACT (CAUSES ATTRACT WHEN MINUS)
        INC     RTCLOCK+1
        BNE     SYSVB1
        INC     RTCLOCK
SYSVB1: LDA     #$FE          ;{ATTRACT} SET DARK MASK TO NORMAL
        LDX     #0            ;SET COLRSH TO NORMAL
        LDY     ATTRACT      ;TEST ATTRACT FOR NEGATIVE
        BPL     VBATRA      ;WHILE POSITIVE DONT GO INTO ATTRACT
        STA     ATTRACT      ;IN ATTRACTI SO STAY BY STA $FE
        LDX     RTCLOCK+1    ;COLOR SHIFT FOLLOWS RICLOCK+1
        LDA     #$F6        ;SET DARK MASK TO DARK
VBATRA: STA     DRKMSK
        STX     COLRSH
        LDX     #0            ;POINT TO TIMER1
        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
        LDA     $104,X      ;GET STACKED P
        AND     #$04        ;I BIT
        BEQ     SYSVB3      ;BRANCH IF OK
XXIT:   JMP     XITVBL      ;I WAS SET, EXIT
SYSVB3: LDA     PENV
        STA     LPENV
        LDA     PENH
        STA     LPENH
        LDA     SDLSTH
        STA     DLISTH
        LDA     SDLSTL
        STA     DLISTL
        LDA     SDMCTL
        STA     DMACTL
        LDA     GPRIOR      ;GLOBAL PRIOR
        STA     PRIOR
        LDX     #$08        ;TURN OFF KEYBOARD SPEAKER
        STX     CONSOL
SCOLLP: CLI                ;DISABLE INTERRUPTS
        LDA     PCOLR0, X   ;LOAD COLOR REGISTERS FROMRAM
        EOR     COLRSH      ;DO COLOR SHIFT
        AND     DRKMSK      ;AND DARK ATTRACT
        STA     COLPM0,X
        DEX
        BPL     SCOLLP
        LDA     CHBAS
        STA     CHBASE
        LDA     CHACT
        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
        BEQ     SYSVBA
        JSR     DCTIMR           ;DECREMENT AND SET FLAG IF NONZERO
        STA     CDTMF3-4,X
SYSVBA: CPX     #8                ;SEE IF DONE ALL 3
        BNE     SYSVBB           ;LOOP
; CHECK DEBOUNCE COUNTER
        LDA     SKSTAT
        AND     #$04             ;KEY DOWN BIT
        BEQ     SYVB6A           ;IF KEY DOWN
; KEY UP SO COUNT IT
        LDA     KEYDEL           ;KEY DELAY COUNTER
        BEQ     SYVB6A           ;IF COUNTED DOWN ALREADY
        DEC     KEYDEL           ;COUNT IT
; CHECK SOFTWARE REPEAT TIMER
SYVB6A: LDA     SRTIMR
        BEQ     SYSVB7           ;DOESN'T COUNT
        LDA     SKSTAT
        AND     #$04             ;CHECK KEY DOWN BIT
        BNE     SYSVB6           ;BRANCH IF NO LONGER DOWN
        DEC     SRTIMR           ;COUNT FRAME OF KEY DOWN
        BNE     SYSVB7           ;BRANCH IF NOT RUN OUT
; TIMER RAN OUT - RESET AND SIMULATE KEYBOARD IRQ
        LDA     #SRTIM2         ;TIMER VALUE
        STA     SRTIMR           ;SET TIMER
        LDA     KBCODE           ;GET THE KEY
        STA     CH               ;PUT INTO CH
; READ GAME CONTROLLERS
SYSVB7: LDY     #1
        LDX     #3
STLOOP: LDA     PORTA,Y
        LSR     A
        LSR     A
        LSR     A
        LSR     A
        STA     STICK0,X        ;STORE JOYSTICK
        DEX
        LDA     PORTA,Y
        AND     #$F
        STA     STICK0,X        ;STORE JOYSTICK
        DEX
        DEY
        BPL     STLOOP
;
        LDX     #3
STRL:  LDA     TRIG0,X          ;MOVE JOYSTICK TRIGGERS
        STA     STRIG0,X
        LDA     POT0,X          ;MOVE POT VALUES
        STA     PADDL0,X
        LDA     POT4,X
        STA     PADDL4,X
        DEX
        BPL     STRL
        STA     POTGO           ;START POTS FOR NEXT TIME
;
        LDX     #6
        LDY     #3
PTRLP: LDA     STICK0,Y        ;TRANSFER BITS FROM JOYSTICKS

```

```

        LSR      A                ;TO PADDLE TRIGGERS
        LSR      A
        LSR      A
        STA      PTRIG1,X
        LDA      #0
        ROL      A
        STA      PTRIG0,X
        DEX
        DEX
        DEY
        BPL      PTRLP
;
        JMP      (VVBLKD)        ;GO TO DEFERRED VBLANK ROUTINE
SV7H    =        SYSVB7/256
SV7L    =        (-256)*SV7H+SYSVB7
SYSVB6: LDA      #0
        STA      SRTIMR          ;ZERO TIMER
        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
        LDY      CDTMV1+1,X      ;SEE IF BOTH ZERO
        BEQ      DCTXF           ;YES, EXIT NONZERO
        DEC      CDTMV1+1,X      ;DEC HI BYTE
DCTIM1: DEC      CDTMV1,X        ;DEC LO BYTE
        BNE      DCTXF
        LDY      CDTMV1+1,X
        BNE      DCTXF
        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
        BNE      SETLOP          ;IF THIS IS LINE "7C"
        LDX      INTEMP
        STA      CDTMV1-1,X
        TYA
        STA      CDTMV1-2,X
        RTS
;

```

```

; EXIT FROM VERTICAL BLANK
;
XITVBL: PLA                      ;UNSTACK Y
      TAY
      PLA                      ;UNSTACK X
      TAX
      PLA                      ;UNSTACK A
      RTI                     ;AND GO BACK FROM WHENCE.

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
FLOPPY = $30
;PRINTR = $40
;CASSET = $80 ;!!!!!! *****
CASET = $60 ;!!!!!! *****
;
;
;BUS COMMANDS
;
READ = 'R
WRITE = 'W
;STATIS = 'S
;FORMAT = '!'
;
;
; COMMAND AUX BYTES
;
SIDWAY = 'S ;PRINT 18 CHARACTERS SIDEWAYS
NORMAL = 'N ;PRINT 40 CHARACTERS NORMALLY
DOUBLE = 'D ;PRINT 20 CHARACTERS DOUBLE WIDE
PLOT = 'P ;PLOT MODE

```

```

;
;
; BUS RESPONSES
;
ACK      =      'A          ;DEVICE ACKNOWLEDGES INFORMATION
NACK     =      'N          ;DEVICE DID NOT UNDERSTAND
COMPLT   =      'C          ;DEVICE SUCCESSFULLY COMPLETED OPERATION
ERROR    =      'E          ;DEVICE INCURRED AN ERROR IN AN ATTEMPTED OP
;
;
; MISCELLANEOUS EQUATES
;
B192LO   =      $28          ;19200 BAUD RATE POKEY COUNTER VALUES (LO BY
B192HI   =      $00          ;(HI BYTE)
B600LO   =      $CC          ;600 BAUD (LO BYTE)
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      PALFLG
WIRGLO   =      150          ;WRITE INTER RECORD GAP (IN 1/60 SEC)
RIRGLO   =      100          ;READ INTER RECORD GAP (IN 1/60 SEC)
WSIRG    =      13           ;SHORT WRITE INTER RECORD GAP
RSIRG    =      8            ;SHORT READ INTER RECORD GAP
      .ENDIF
      .IF      PALFLG-1
WIRGLO   =      180          ;WRITE INTER RECORD GAP (IN 1/60 SEC)
RIRGLO   =      120          ;READ INTER RECORD GAP (IN 1/60 SEC)
WSIRG    =      15           ;SHORT WRITE INTER RECORD GAP
RSIRG    =      10           ;SHORT READ INTER RECORD GAP
      .ENDIF
WIRGHI   =      0
RIRGHI   =      0
;
NCOMLO   =      $34          ;PIA COMMAND TO LOWER NOT COMMAND LINE
NCOMHI   =      $3C          ;PIA COMMAND TO RAISE NOT COMMAND LINE
MOTRGO   =      $34          ;PIA COMMAND TO TURN ON CASSETTE MOTOR
MOTRST   =      $3C          ;PIA COMMAND TO TURN OFF MOTOR
;
TEMPHI   =      TEMP/256     ;ADDRESS OF TEMP CELL (HI BYTE)
TEMPLO   =      (-256)*TEMPHI+TEMP ;(LO BYTE)
CBUFHI   =      CDEVIC/256   ;ADDRESS OF COMMAND BUFFER (HI BYTE)
CBUFLO   =      (-256)*CBUFHI+CDEVIC ;(LO BYTE)
;
CRETRI   =      13           ;NUMBER OF COMMAND FRAME RETRIES
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
        STA    SOUNDN          ;INIT POKE ADDRESS FOR QUIET I/O
        STA    SKCTL
;
;
        RTS                    ;RETURN
;
;
;
;
;
SIO:    TSX
        STX    STACKP          ;SAVE STACK POINTER
        LDA    #1
        STA    CRITIC
;
        LDA    DDEVIC
        CMP    #CASET
        BNE    NOTCST          ;BRANCH IF NOT CASSETTE
        JMP    CASENT          ;OTHERWISE JUMP TO CASSETTE ENTER
;
; ALL DEVICES EXCEPT CASSETTE ARE INTELLIGENT
;
NOTCST: LDA    #0
        STA    CASFLG          ; INIT CASSETTE FLAG TO NO CASSETTE
;
        LDA    #DRETRI          ;SET NUMBER OF DEVICE RETRIES
        STA    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
        LDA    DDEVIC
        ADC    DUNIT
        ADC    #$FF                ;SUBTRACT 1
        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
        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
        STA    BUFRHI
        STA    BFENHI                ;DONE SETTING UP BUFFER POINTER
;
        LDA    #NCOMLO                ;LOWER NOT COMMAND LINE
        STA    PBCTL
;
        JSR    SENDIN                ;SEND THE COMMAND FRAME TO A SMART DEVICE
;
        LDA    ERRFLG
        BNE    BADCOM                ;BRANCH IF AN ERROR RECEIVED
;
        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
;
        LDA    #CRETRI                ;SET NUMBER OF RETRIES
        STA    CRETRY
;

```

```

        JSR      LDPNTR          ;LOAD BUFFER POINTER WITH DCB INFORMATION
;
        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
;
        LDA      #$00
        STA      ERRFLG          ;CLEAR ERROR FLAG
;
        JSR      WAITER          ;SET UP TIMER AND WAIT
        BEQ      DERR            ;BRANCH IF TIME OUT
;
;
; DEVICE DID NOT TIME OUT
;
        BIT      DSTATS
        BVS      MODATA          ;BRANCH IF MORE DATA FOLLOWS
;
        LDA      ERRFLG
        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 PRECEDED DATA
;
        LDA      TSTAT           ;GET TEMP STATUS
        STA      STATUS          ;STORE IN REAL STATUS
;
;
NOTERR: LDA      STATUS
        CMP      #SUCCES
        BEQ      RETURN          ;BRANCH IF COMPLETELY SUCCESSFUL
;
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     #$00
        STA     ERRFLG          ;CLEAR ERROR FLAG
;
        CLC                     ;LOAD BUFFER POINTER WITH ADDRESS
        LDA     #TEMPLO         ;OF TEMPORARY RAM CELL
        STA     BUFRLO
        ADC     #1
        STA     BFENLO          ;ALSO SET BUFFER END +1 ADDRESS
        LDA     #TEMPHI
        STA     BUFRHI
        STA     BFENHI          ;DONE LOADING POINTER
;
        LDA     #$FF
        STA     NOCKSM          ;SET NO CHECKSUM FOLLOWS DATA FLAG
;
        JSR     RECEIV          ;GO RECEIVE A BYTE
;
        LDY     #$FF            ;ASSUME SUCCESS
        LDA     STATUS
        CMP     #SUCCES
        BNE     NWOK            ;BRANCH IF IT DID NOT WORK OK
;
;
;
;
WOK:    LDA     TEMP            ;MAKE SURE THE BYTE SUCCESSFULLY RECEIVED
        CMP     #ACK            ;WAS ACTUALLY AN ACK OR COMPLETE
        BEQ     GOOD
        CMP     #COMPLT
        BEQ     GOOD
;
        CMP     #ERROR
        BNE     NOTDER          ;BRANCH IF DEVICE DID NOT SEND BACK
; A DEVICE ERROR CODE
        LDA     #DERROR
        STA     STATUS          ;SET DEVICE ERROR STATUS
        BNE     NWOK
;
NOTDER: LDA     #DNACK          ;OTHERWISE SET HACK STATUS
        STA     STATUS
;
NWOK:   LDA     STATUS
        CMP     #TIMOUT
        BEQ     BAD             ;BRANCH IF TIME OUT
;
        LDA     #$FF
        STA     ERRFLG          ;SET SOME ERROR FLAG

```

```

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

```

```

NOWRPO: LDA    BUFRLO            ;CHECK IF PAST END OF BUFFER
        CMP     BFENLO
        LDA     BUFRHI            ;HIGH PART
        SBC     BFENHI
        BCC     NOTEND            ;BRANCH IF NOT PAST END OF BUFFER
;
        LDA     CHKSNT
        BNE     RELONE            ;BRANCH IF CHECKSUM ALREADY SENT
;
        LDA     CHKSUM
        STA     SEROUT            ;SEND CHECK SUM
        LDA     #$FF
        STA     CHKSNT            ;SET CHECKSUM SENT FLAG
        BNE     CHKDON
;
RELONE:  LDA     POKMSK            ;ENABLE TRANSMIT DONE INTERRUPT
        ORA     #$08
        STA     POKMSK
        STA     IRQEN
;
CHKDON:  PLA
        TAY                ;RESTORE Y REG
        PLA                ;RETURN FROM INTERRUPT
        RTI
;
;
NOTEND:  LDY     #0
        LDA     (BUFRLO),Y      ;PUT NEXT BYTE FROM BUFFER
        STA     SEROUT            ;INTO THE SERIAL OUTPUT REGISTER
;
        CLC                ;ADD IT TO CHECKSUM
        ADC     CHKSUM
        ADC     #0
        STA     CHKSUM
;
        JMP     CHKDON            ;GO RETURN
;
;
;
;
;
; TRANSMIT DONE INTERRUPT SERVICE ROUTINE
;
ISRTD:  LDA     CHKSNT
        BEQ     FOOEY            ;BRANCH IF CHECKSUM NOT YET SENT
;
        STA     XMTDON            ;OTHERWISE SET TRANSMISSION DONE FLAG
;
        LDA     POKMSK            ;DISABLE TRANSMIT DONE INTERRUPT
        AND     #$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        ;BUFFER FULL FLAG
;         STA    RECVDN        ;RECEIVE DONE FLAG
;
;
;
;         LDA    #SUCCES
;         STA    STATUS        ;SET GOOD STATUS FOR DEFAULT CASE.
;         JSR    RECVEN        ;DO RECEIVE ENABLE
;         LDA    #NCOMHI        ;COMMAND FRAME HI COMMAND
;         STA    PBCTL        ;STORE IN PIA
CHKTIM: LDA    BRKKEY
;         BNE    NTBRK1
;         JMP    BROKE        ;JUMP IF BREAK KEY PRESSED
;
NTBRK1: LDA    TIMFLG        ;NO,
;         BEQ    TOUT        ;IF TIMEOUT, GO SET ERROR STATUS
;         LDA    RECVDN
;         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
;
;
;
;         LDA    SKSTAT
;         STA    SKRES        ;RESET STATUS REGISTER
; ***** 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      #$20
        BNE      NTOVRN          ;BRANCH IF NO OVERRUN ERROR
;
        LDY      #OVRRUN
        STY      STATUS          ;SET OVERRUN ERROR STATUS
;
NTOVRN: LDA      BUFRFL
        BEQ      NOTYET          ;BRANCH IF BUFFER WAS NOTYET FILLED
;
        LDA      SERIN           ;THIS INPUT BYTE 15 THE CHECKSUM
        CMP      CHKSUM
        BEQ      SRETRN          ;BRANCH IF CHECKSUMS MATCH
;
        LDY      #CHKERR
        STY      STATUS          ;SET CHECKSUM ERROR STATUS
;
SRETRN: LDA      #$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
        STA      CHKSUM
;
        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
        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    BFENLO          ;ALSO SET SUFFER END + 1 ADDRESS
;
        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
;
        LDY    #WSIRG          ;LOAD SHORT WRITE INTER RECORD GAP TIME
        LDA    DAUX2
        BMI    SRTIR0          ;BRANCH IF SHORT GAP IS DESIRED
;
        LDY    #WIRGLO        ;SET WRITE IRQ TIME
SRTIR0: LDX    #WIRGHI
        JSR    SETVBX
;
        LDA    #MOTRGO

```



```

        STA      PACTL      ;TURN ON MOTOR
;
TIMIT:  LDA      TIMFLG      ;LOOP UNTIL DONE
        BNE      TIMIT
;
        JSR      LDPNTR      ;LOAD BUFFER POINTER WITH DCB INFORMATION
;
        JSR      SEND        ;SEND A BUFFER
;
        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
        LDA      DAUX2
        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
;
        JSR      BEGIN      ;SET INITIAL BAUD RATE
;
        JSR      RECEIV      ;GO RECEIVE A BLOCK
;
CRETRN: LDA      DAUX2
        BMI      SRTIR2      ;BRANCH IF DOING SHORT INTER RECORD GAPS
; DON'T TURN OFF CASSETTE MOTOR
        LDA      #MOTRST
        STA      PACTL      ;TURN OFF MOTOR
;
SRTIR2: JMP      RETURN      ;GO RETURN
;
;
;
;
JTIMER: LDA      #$00
JTADRH  =          JTIMER/256 ;HI BYTE OF JUMP TIMER ROUTINE ADDR
JTADRL  =          (-256)*JTADRH+JTIMER
        STA      TIMFLG      ;SET TIME OUT FLAG
        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
        BNE     NOTCAS          ;BRANCH IF NOT CASSETTE
;
        ORA     #$08            ;SET THE FSK OUTPUT BIT
;
        LDY     #LOTONE         ;SET FSK TONE FREQUENCIES
        STY     AUDF2
        LDY     #HITONE
        STY     AUDF1
;
NOTCAS: STA     SSKCTL          ;STORE NEW VALUE TO SYSTEM MASK
        STA     SKCTL           ;STORE TO ACTUAL REGISTER
;
        LDA     #$C7            ;MASK OFF PREVIOUS SERIAL BUS INTERRUPT BITS
        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
        AND     SSKCTL
        ORA     #$10            ;SET RECEIVE MODE ASYNCH.
        STA     SSKCTL          ;STORE NEW VALUE TO SYSTEM MASK
        STA     SKCTL           ;STORE TO ACTUAL REGISTER
;
        STA     SKRES          ;RESET SERIAL PORT/KEYBOARD STATUS REGISTER
;
        LDA     #$C7            ;MASK OFF PREVIOUS SERIAL BUS INTERRUPTBITS
        AND     POKMSK
        ORA     #$20            ;ENABLE RECEIVE INTERRUPT
CONTIN: STA     POKMSK          ;STORE NEW VALUE TO SYSTEM MASK
        STA     IRQEN           ;STORE TO ACTUAL. REGISTER
;
;
        LDA     #$28            ;CLOCK CH.3 WITH 1.79 MHZ

```

```

STA      AUDCTL              ;CLOCK CH.4 WITH CH. 3
;

LDX      #6                  ;SET PURE TONES, NO VOLUME
LDA      #$A8
LDY      SOUNDNR             ;TEST QUIET I/O FLAG
BNE      NOISE1              ;NE IS NORMAL (NOISY)
LDA      #$A0
NOISE1: STA    AUDC1,X
DEX
DEX
BPL      NOISE1
;

LDA      #$A0
STA      AUDC3               ;TURN OFF SOUND ON CHANNEL 3
LDY      DDEVIC
CPY      #CASET
BEQ      CAS31               ;BRANCH IF CASSETTE IS DESIRED
STA      AUDC1               ;OTHERWISE TURN OFF CHANNELS 1 AND 2
STA      AUDC2
;
;
CAS31:   RTS                 ;RETURN
;
;
;
;
;
;
;
;
;
;
;
;
; DISABLE SEND AND DISABLE RECEIVE SUBROUTINES
;
SENDDS: NOP
RECVDS: LDA    #$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    AUDC1,X
DEX
DEX
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
        ROR     A            ;PUT 6 HI BITS IN X, LO 2 BITS IN Y
        ROR     A
        TAY                     ;TEMP SAVE
        AND     #$3F          ;MASK OFF 2 HI BITS
        TAX                     ;THIS IS HZ BYTE OF TIME OUT
;
        TYA                     ;RESTORE
        ROR     A
        AND     #$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
;
        TYA                     ;IF Y=0, A TIME OUT OR NACK OCCURED
;
        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        ;SAVE ADJUSTED VALUE
        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
        SBC     TIMER1+1
        TAY                     ;FIND VBLANK COUNT DIFFERENCE
        .IF     PALFLG
        LDA     #-$9C
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
        LSR     A
        LSR     A
        LSR     A
        ASL     A
        SEC
        SBC     #22           ;ADJUST TABLE INDEX
        TAX                     ;DIVIDE INTERVAL BY 4 TO CET TABLE INDEX
        TYA                     ;RESTORE ACCUM
        AND     #7
        TAY                     ;PULL OFF 3 LO BITS OF INTERVAL
        LDA     #-11
DOINTP: CLC
        ADC     #11           ;ACCUMULATE INTERPOLATION CONSTANT
        DEY
        BPL     DOINTP        ;INTERPOLATION CONSTANT COMPUTATION DONE?

```

```

;
ENINTP: LDY    #0
        STY    ADDCOR          ;CLEAR ADDITION CORRECTION FLAG
        SEC
        SBC    #7              ;ADJUST INTERPOLATION CONSTANT
        BPL    PLUS
        DEC    ADDCOR
PLUS:   CLC
        ADC    POKTAB,X        ;ADD CONSTANT TO LO BYTE TABLE VALUE
        TAY
        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
        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
        JMP    BROKE          ;JUMP IF BREAK KEY PRESSED
;
NTBRK2: SEI
;
        LDA    TIMFLG
        BNE    OKTIM1          ;BRANCH IF NOT TIMEDOUT
        BEQ    TOUT1           ;BRANCH IF TIME OUT
;
OKTIM1: LDA    SKSTAT
        AND    #$10            ;READ SERIAL PORT
        BNE    BEGIN          ;START BIT?
        STA    SAVIO           ;SAVE SER. DATA IN

```

```

        LDX      VCOUNT          ;READ VERTICAL LINECOUNTER
        LDY      RTCLOK+2        ;READ LO BYTE OF VBLANK CLOCK
        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
        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
        CMP      SAVIO           ;DATA IN CHANGED YET?
        BEQ      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
        STA      CBAUDH          ;SET BAUD RATE INTO RAM CELLS
        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
        STA      (BUFRLO),Y      ;STORE '$55' AS FIRST RCV. BUFFER
        INY
        STA      (BUFRLO),Y
        LDA      #$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
;
        LDX  STACKP
        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
;
        SEI              ;THE SETVBL ROUTINE NEEDS THIS TO CUT SHORT
        JSR  SETVBV     ;ANY VBLANKS THAT OCCUR
        LDA  #1         ;SET FOR TIMER 1
        STA  TIMFLG     ;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  $F_{IN} = 1.78979 \text{ MHz}$ . &  $M = 7$ 
;
;      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  $T_{OUT} = \# \text{ OF COUNTS}$ 
;      (127 USEC. RESOLUTION) OF 'VCOUNT' FOR 1
;      CHARACTER TIME (10 BIT TIMES).
;

```



```

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

```

```

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

```

```

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

```

```
;*****  
;  
;  
;      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  
;  
OPNOUT   =          $2           ;IOCB OPEN FOR OUTPUT COMMAND  
NBUSFSZ  =          40           ;PRINT NORMAL BUFFER SIZE  
DBUSFSZ  =          20           ;PRINT DOUBLE BUFFFER SIZE  
SBUSFSZ  =          29           ;PRINT SIDEWAYS BUFFER SIZE  
PDEVN    =          $40          ;PRINTER DEVICE NUMBER  
; STATC  =          $53          ;DCB STATUS COMMAND CODE  
WRITEC   =          $57          ;DCB WRITE COMMAND  
SPACE    =          $20          ;ASCII SPACE CHAR.  
N         =          $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
        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
        STA     PBUFSZ      ;SET BUFFER SIZE TO 4 BYTES
        LDX     PHSTLO
        LDY     PHSTLO+1    ;SET POINTER TO STATUS BUFFER
        LDA     #STATC      ;SETCOMMAND TO "STATUS"
        STA     DCOMND      ;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
        STA     PBPNT      ;CLEAR PRINT BUFFER POINTER
        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
        CPX      PBUFSZ        ;BUFFER POINTERBUFFER SIZE?
        BEQ      BUFFUL
        STX      PBPNT        ;SAVE SUFFER POINTER
        CMP      #CR           ;IS CHAR. = EOL ?
        BEQ      BLFILL        ;IF YES, GO DO BLANK FILL.
        LDY      #SUCCES       ;PUT GOOD STATUS IN Y REQ FOR CIO.
        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
        STA      PBPNT        ;CLEAR PRINT BUFFER POINTER
        LDX      PHCHLO
        LDY      PHCHLO+1      ;SET POINTER TO PRINT BUFFER
        JSR      SETDCB        ;GO SETUP OCR
        JSR      SIOV          ;SEND PRINT COMMAND
        RTS                    ;YES.

;
;
;
;
;      PRINTER HANDLER CLOSE ROUTINE
;
PHCLOS: JSR      PRMODE        ;GO DETERMINE PRINT MODE
        LDX      PBPNT
        BNE      BLFILL
        LDY      #SUCCES
        RTS

;
;
;
;
;
;
;
;
;
;      S U B R O U T I N E S
;
;
;
;
;      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
        STA      DUNIT          ;SET UNIT NUMBER TO 1
        LDA      #$80          ;DEVICE WILL EXPECT DATA
        LDX      DCOMND
        CPX      #STATC        ;STATUS COMMAND?
        BNE      PSIOC

```

```

        LDA      #$40          ;EXPECT DATA FROM DEVICE
PSIOC:  STA      DSTATS        ;SET SIO MODE COMMAND
        LDA      PBUFSZ
        STA      DBYTLO        ;SET LO BYTE COUNT
        LDA      #0
        STA      DBYTHI        ;SET HI BYTE COUNT
        LDA      PTIMOT
        STA      DTIMLO        ;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 ?
        LDX      #NBUFSZ      ;YES, SET NORMAL CHAR. BUFFER SIZE
        BNE      SETBSZ
CDUBL:  CMP      #D
        BNE      CSIDE        ;PRINT DOUBLE?
        LDX      #DBUFSZ      ;YES, SET DOUBLE CHAR. BUFFER SIZE
        BNE      SETBSZ
CSIDE:  CMP      #S
        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
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)'
CBUFH   =          CASBUF/256
CBUFL   =          (-256)*CBUFH+CASBUF

```

```

SRSTA    =          $40                ;SIO READ STATUS
SWSTA    =          $80                ;SIO WRITE STATUS
;MOTRGO  =          $34
;MOTRST  =          $3C
;
;
DTA      =          $FC                ;DATA RECORD TYPE BYTE
DT1      =          $FA                ;LAST DATA RECORD
EOT      =          $FE                ;END OF TAPE
HDR      =          $FB                ;HEADER
TONE1    =          2                  ;CHANGE TO RECORD MODE TONE
TONE2    =          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
;
*=RBLOKV
JMP      RBLOK
;
*=CSOPIV
JMP      OPINP
;
;
*=CASORG
;
;
; INIT ROUTINE
;
INIT:    LDA      #$CC
        STA      CBAUDL
        LDA      #$05
        STA      CBAUDH                ;SET CASSET BAUD RATE TO 600
SPECIAL:                                ;THATS ALL FOLKS
        RTS
        .PAGE
;
; OPEN FUNCTION - WITH NO TIMING ADJUST
;
OPENC:   LDA      ICAX2Z                ;GET AX2
        STA      FTYPE                ;SAVE IT FOR FUTURE REFERENCE
        LDA      ICAX1Z
        AND      #$0C                ;IN AND OUT BITS
        CMP      #$04
        BEQ      OPINP
        CMP      #$08                ;SEE IF OPEN FOR OUTPUT
        BEQ      OPOUT
        RTS                          ;IF ALREADY OPEN, RETURN LEAVING STATUS=$84
OPINP:   LDA      #0
        STA      WMODE                ;SET READ MODE

```



```

        STA      FEOF          ;NO EOF YET
SFH:    LDA      #TONE2        ;TONE FOR PRESS PLAY
        JSR      BEEP          ;GO BEEP
        BMI      OPNRTN        ;IF ERROR DURING BEEP
        LDA      #MOTRGO
        STA      PACTL        ;TURN MOTOR ON
        .IF      PALFLG
        LDY      #$E0
        LDX      #1
        .ENDIF
        .IF      PALFLG-1
        LDY      #$40          ;5-31-79 9 SEC READ LEADER
        LDX      #2
        .ENDIF
        LDA      #3
        STA      CDTMF3
        JSR      SETVBV        ;SET UP YBLANK TIMER
WAITTM: LDA      CDTMF3
        BNE      WAITTM        ;WAIT FOR MOTOR TO COME UP TO SPEED
        LDA      #$80          ;NEXT BYTE=NO BYTES IN BUFFER
        STA      BPTR
        STA      BLIM
        JMP      OPOK          ;OPEN OK
;
; OPEN FOR OUTPUT
;
PBRK:   LDY      #BRKABT        ;BREAK KEY ABORT STATUS
        DEC      BRKKEY        ;RESET BREAK KEY
OPNRTN: LDA      #0            ;CLEAR WRITE MODE FLAG
        STA      WMODE
        RTS                  ;AND EXIT.
;
OPOUT:  LDA      #$80
        STA      WMODE        ;SET WRITE MODE
        LDA      #TONE1        ;TELL USER TO TURN ON RECORD MODE
        JSR      BEEP
        BMI      OPNRTN        ;IF ERROR DURING BEEP
        LDA      #$CC          ;SET BAUD RATE
        STA      AUDF3        ;WHICH SEEMS TO BE NESSECARY
        LDA      #$05          ;FOR SOME OBSCURE REASON
        STA      AUDF4
        LDA      #$60
        STA      DDEVIC
        JSR      SENDEV        ;TELL POKEY TO WRITE MARKS
        LDA      #MOTRGO        ;WRITE 5 SEC BLANK TAPE
        STA      PACTL
        LDA      #3
        .IF      PALFLG
        LDX      #$3
        LDY      #$C0
        .ENDIF
        .IF      PALFLG-1
        LDX      #4            ;5/30/79 20 SEC LEADER
        LDY      #$80
        .ENDIF
        JSR      SETVBV
        LDA      #$FF
        STA      CDTMF3
WDLR:   LDA      BRKKEY

```

```

        BEQ      PBRK            ;IF BREAK DURING WRITE LEADER
        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
        CPX      BLIM           ;IF END OF BUFFER
        BEQ      RBLOK          ;READ ANOTHER BLOCK
        LDA      CASBUF+3,X     ;GET NEXT BYTE
        INC      BPTR           ;DUMP POINTER
        LDY      #SUCCES        ;OK STATUS
GBX:    RTS
RBLOK:  LDA      #'R            ;READ OPCODE
        JSR      SIOSB          ;SIO ON SYS BUF
        TYA
        BMI      GBX            ;IF SIO ERRORS, RETURN
        LDA      #0
        STA      BPTR          ;RESET POINTER
        LDX      #$80           ;DEFAULT # BYTES
        LDA      CASBUF+2
        CMP      #EOT
        BEQ      ATEOF          ;IF HEADER, GO READ AGAIN
        CMP      #DT1           ;IF LAST DATA REC
        BNE      NLR
        LDX      CASBUF+130     ;LAST DATA RECORD, GET # BYTES
NLR:    STX      BLIM
        JMP      GBYTE          ;GET NEXT BYTE
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
        INC      BPTR           ;BUMP POINTER
        LDY      #SUCCES        ;OK STATUS
        CPX      #127           ;IF BUFFER FULL
        BEQ      *+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
FCAX:   LDA    #MOTRST         ;STOP THE MOTOR IN CASE WAS SHORT IRQ MODE
        STA    PACTL
        RTS
CLWRT:  LDX    BPTR            ;BUFFER POINTER
        BEQ     WTLR           ;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
WTLR:   LDX    #127            ;ZERO BUFFER
        LDA     #0
ZTBUF:  STA    CASBUF+3,X
        DEX
        BPL     ZTBUF
        LDA     #EOT           ;WRITE EOT RECORD
        JSR     WSIOSB
        JMP     FCAX           ;FLAG CLOSED AND EXIT
        .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
        ADC     #30           ; 1 SEC TONE
        .ENDIF
        TAX
WFL:    LDA     #$FF
        STA     CONSOL        ;TURN ONSPEAKER
        LDA     #0
        LDY     #$F0
        DEY
        BNE     *-1
        STA     CONSOL        ;TURN OFF SPEAKER
        LDY     #$F0
        DEY
        BNE     *-1
        CPX     RTCLOK+2      ;SEE IF 1 SEC IS UP YET
        BNE     WFL
        DEC     FREQ          ;COUNT BEEPS
        BEQ     WFAK          ;IF ALL DONE GO WAIT FOR KEY
        TXA

```

```

        CLC
        .IF      PALFLG
        ADC      #8
        .ENDIF
        .IF      PALFLG-1
        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
;
SIOBS:   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
CSIO:    LDA      #$60          ;CASSET PSEUDO DEVICE
        STA      DDEVIC
        LDA      #0
        STA      DUNIT
        LDA      #35          ;DEVICE TIMEOUT (5/30/79)
        STA      DTIMLO
        LDA      DCOMND          ;GET COMMAND SACK
        LDY      #SRSTA          ;SIO READ STATUS COMMAND
        CMP      #'R
        BEQ      *+4
        LDY      #SWSTA          ;SIC WRITE STATUS COMMAND
        STY      DSTATS          ;SET STATUS FOR SIO
        LDA      FTYPE
        STA      DAUX2          ;INDICATE IF SHORT IRQ MODE
        JSR      SIOV          ;GO CALL SIO
        RTS
;
; WSIOBS - WRITE SIC SYSTEM SUFFER
;
WSIOBS:  STA      CASBUF+2      ;STORE TYPE BYTE
        LDA      #$55
        STA      CASBUF+0
        STA      CASBUF+1
        LDA      #'W          ;WRITE
        JSR      SIOBS          ;CALL SIO ON SYSTEM BUFFER
        RTS      AND          ;RETURN
CRNTP6   =*
        *=$14

```

```

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
GETCAR  =          $7              ;"GET CHARACTER" CIO COMMAND CODE
PUTCAR  =          $B              ;"PUT CHARACTER" CIO COMMAND CODE
INIMLL  =          $00             ;INITIAL HEM LO LOW BYTE
INIMLH  =          $07             ;INITIAL HEM LO HIGH BYTE
; GOOD  =          $1              ;GOOD STATUS CODE
; WRITE =          $57             ;WRITE COMMAND
; READ  =          $52             ;READ COMMAND
; STATC =          $53             ;STATUS COMMAND
SEX      =          $0              ;SCREEN EDITOR 10CR INDEX
CLS      =          $7D             ;CLEAR SCREEN CODE
CTRLC   =          $92             ;KEYBOARD CODE FOR 'CONTROL C'
EOF      =          $88             ;CASSETTE END OF FILE CODE
LIRQ     =          $0              ;LONG IRQ TYPE CODE
;
BUFFH    =          (CASBUF+3)/256
BUFFL    =          (-256)*BUFFH+CASBUF+3 ;BUFFER POINTER
;
;
;
; THE FOLLOWING EQUATES ARE IN THE CARTRIDGE ADDRESS SPACE.
;
;
; "B" CARTRIDGE ADDR'S ARE 8000-9FFF (36K CONFIG. ONLY)
; "A" CART. ADDR'S ARE A000-BFFF (36K CONFIG. ONLY)
;
; "A" CART. ADDR'S ARE B000-BFFF (48K CONFIG. ONLY)
;
        *=$BFFA
CARTCS:  .RES      2                ;CARTRIDGE COLD START ADDRESS.
CART:    .RES      1                ;CARTRIDGE AVAILABLE FLAG BYTE.
CARTFG:  .RES      1                ;CARTRIDGE FLAG BYTE. BIT 0=FLAG1,
CARTAD:  .RES      2                ;2-BYTE CARTRIDGE START VECTOR
;
;
;
        CARTRIDGE FLAG ACTION DEFINITIONS
;
;
;
        BIT              ACTION IF SET
;
;
        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 BIT0 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
;
;
;TBLLN = 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
;
TBLLN = 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 IRQ INTERRUPTS
        LDA      COLDST          ;WERE WE IN MIDDLE OF COLDSTART?
        BNE      PWRUP           ;YES, GO TRY IT AGAIN
        LDA      #$FF
        BNE      PWRUP1          ;SET WARM START FLAG
;
;
;
;*****
;      POWER UP ROUTINES START HERE
;*****

```

```

;
PWRUP:  SEI                      ;DISABLE IRQ INTERRUPTS
        LDA      #0              ;CLEAR WARMSTART FLAG
PWRUP1:  STA      WARMST
        CLD                      ;CLEAR DECIMAL FLAG.
        LDX      #$FF
        TXS                      ;SET STACK POINTER
        JSR      SPECL           ;CARTRIDGE SPECIAL CASE?
        JSR      HARDI          ;DO HARDWARE INITIALIZATION
        LDA      WARMST          ;IS IT WARMSTART?
        BNE      ZOSRAM          ;YES, ONLY ZERO OS RAM
;
ZERORM:  LDA      #0
        LDY      #WARMST
        STA      RAMLO
        STA      RAMLO+1         ;INITIALIZE RAM POINTER
CLRRAM:  STA      (RAMLO),Y ;CLEAR MEMORY LOC.
        INY
        CPY      #0              ;AT END OF PAGE?
        BNE      CLRRAM
        INC      RAMLO+1         ;YES. INCR PAGE POINTER
        LDX      RAMLO+1
        CPX      TRAMSZ          ;AT END OF MEM?
        BNE      CLRRAM          ;NO.
;
; INITIALIZE DOSVEC TO POINT TO SIGNON (BLACKBOARD)
        LDA      BLKBDV+1
        STA      DOSVEC          ;USE BLACKBOARD VECTOR
        LDA      BLKBDV+2        ;FOR DOSVEC
        STA      DOSVEC+1
        LDA      #$FF
        STA      COLDST          ;SET TO SHOW IN MIDDLE OF COLDSTART
        BNE      ESTSCM          ;GO AROUND ZOSRAM
;
; 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 IRQ INTERRUPTS
;
;
;   LINK HANDLERS
;
        LDX      #TBLLLEN
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
        LDX      RAMSIZ
        CPX      #$90        ;RAM IN "B" CART. SLOT?
        BCS      ENDBCK
        LDA      CART-$2000   ;NO.
        BNE      ENDBCK      ;CART. PLUGGED INTO "B" SLOT?
        INC      TSTDAT      ;YES, SET "B" CART. FLAG
        JSR      CBINI        ;INITIALIZE CARTRIDGE "B"
;
ENDBCK: LDX      RAMSIZ
        CPX      #$B0        ;RAM IN "A" CART. SLOT?
        BCS      ENDACK
        LDX      CART        ;NO,
        BNE      ENDACK      ;CART. PLUGGED INTO "A" SLOT?
        INC      TRAMSZ      ;YES, SET "A" CART. FLAG
        JSR      CAINI        ;INITIALIZE CARTRIDGE "A"
;
;
; OPEN SCREEN EDITOR
;
ENDACK: LDA      #3
        LDX      #SEX
        STA      ICCOM,X      ;OPEN I/O COMMAND
        LDA      #OPNL
        STA      ICBAL,X
        LDA      #OPNH
        STA      ICBAH,X      ;SET BUFFER POINTER TO OPEN SCREEN EDITOR
        LDA      #$C
        STA      ICAX1,X      ;SET UP OPEN FOR INPUT/OUTPUT
        JSR      CIOV         ;GO TO CIO
;
        BPL      SCRNOK      ;BR IF NO ERROR
        JMP      PWRUP       ;RETRY PWRUP IF ERROR (SHOULD NEVER HAPPEN!)
SCRNOK: INX
        BNE      SCRNOK      ;SCREEN OK, SO WAIT FOR YBLANK TO
        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
        ORA      TSTDAT          ;
        BEQ      NOCART          ;NEITHER CARTRIDGE LIVES
        LDA      TRAMSZ          ;"A" CART?
        BEQ      NOA1            ;NO
        LDA      CARTFG          ;GET CARTRIDGE MODE FLAG
NOA1:    LDX      TSTDAT          ;"B" CART?
        BEQ      NOB1            ;NO
        ORA      CARTFG-$2000    ;ADD OTHER FLAG
NOB1:    AND      #1              ;DOES EITHER CART WANT BOOT?
        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            ;NO
        LDA      CARTFG          ;GET CARTRIDGE MODE FLAG
        AND      #4              ;DOES IT WANT TO RUN?
        BEQ      NOA2            ;NO
        JMP      (CARTCS)        ;RUN "A" CARTRIDGE
NOA2:    LDA      TSTDAT          ;"B" CART?
        BEQ      NOCAR2          ;NO
        LDA      CARTFG-$2000    ;GET "B" MODE FLAG
        AND      #4              ;DOES IT WANT TO RUN?
        BEQ      NOCART          ;NO
        JMP      (CARTCS-$2000)  ;RUN "B" CARTRIDGE
;
; 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)

```

.PAGE

S U B R O U T I N E S

```
;
;
;
;
;
;
;
;
;
;
;
;
;
;
;
;
;
;
;
;
;
;
;
; 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
        LDA    CART            ;IS IT ROM?
        BNE    ENSPEC          ;NO
        LDA    CARTFG          ;YES,
        BPL    ENSPEC          ;BIT SET?
        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
        BNE    ENDRAM
        EOR    #$FF            ;CONVERT IT BACK
        STA    (RAMLO+1),Y ;RESTORE ORIGINAL RAMDATA
        LDA    TRAMSZ
        CLC
        ADC    #$10
        STA    TRAMSZ          ;INCR. RAM POINTER BY 4K.
        BNE    HOWMCH          ;GO FIND HOW MUCH RAM.
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
;
OSRAM:  DEC     BRKKEY           ;TURN OFF BREAK KEY FLAG
        LDA     #.LOW.BRKKY2
        STA     BRKKY
        LDA     #.HIGH.BRKKY2
        STA     BRKKY+1
        LDA     TRAMSZ           ;READ RAM SIZE IN TEMP. REG.
        STA     RAMSIZ          ;SAVE IT IN RAM SIZE.
        STA     MEMTOP+1        ;INIT. MEMTOP ADDR HI BYTE
        LDA     #0
        STA     MEMTOP          ;INIT. MEMTOP ADDR LO BYTE
        LDA     #INIMLL
        STA     MEMLO
        LDA     #INIMLH
        STA     MEMLO+1         ;INITIALIZE MEMLO ADDR VECTOR
        JSR     EDITRV+$C       ;EDITOR INIT.
        JSR     SCRENV+$C       ;SCREEN INIT.
        JSR     KEYBDV+$C       ;KEYBOARD INIT.
        JSR     PRINTV+$C       ;PRINTER HANDLER INIT
        JSR     CASETV+$C       ;CASSETTE HANDLER INIT
        JSR     CIOINV          ;CIO INIT.
        JSR     SIOINV          ;SIO INIT.
        JSR     INTINV          ;INTERRUPT HANDLER INIT.
        LDA     CONSOL
        AND     #$1
        BNE     NOKEY           ;GAME START KEY DEPRESSED?
        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.
        LDA      #STATC
        STA      DCOMND        ;SET UPSTATUS COMMAND
        JSR      DSKINV        ;GO DO DISK STATUS
        BPL      DOBOOT        ;IS STATUS FROM 510 GOOD?
        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
        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
        STA      DFLAGS,X     ;STORE IT
        DEX
        BPL      RDBYTE        ;DONE WITH 4 BYTE TRANSFER
        LDA      BOOTAD        ;YES.
        STA      RAMLO
        LDA      BOOTAD+1
        STA      RAMLO+1      ;PUT BOOT ADDR INTO Z. PAGE RAM
        LDA      CASBUF+7
        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
        LDA      RAMLO+1
        ADC      #0
        STA      RAMLO+1      ;INCR BOOT LOADER BUFFER POINTER
        DEC      DBSECT        ;DECR # OF SECTORS.
        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
        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 RECORD, BUT DON'T USE IT.
XBOOT:  JSR      BLOAD        ;GO EXECUVE BOOT LOADER

```

```

        BCS      BADDSK          ;IF BAD BOOT, DO IT OVER AGAIN
        JSR      DINI            ;GO INIT. SOFTWARE
        INC      BOOT?           ;SHOW BOOT SUCCESS
        RTS

BLOAD:   CLC
        LDA      BOOTAD
        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
;
;
;
; 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        ;"PUT TEXT RECORD" COMMAND
        LDA      #$FF
        STA      ICBLI,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          ;SET DRIVE NO. TO DRIVE 0
        JSR      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 CASSETTE BOOT SUCCESFULL?
        BEQ     NOCSB2          ;NO
        JSR     CINI            ;YES, INIT CASSETTE SOFTWARE
NOCSB2: RTS
;
CSBOT2: LDA     CKEY
        BEQ     NOCSBT          ;"C" KEY FLAG SET ?
        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
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)
        .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)
        .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    #$FF
        STA    CH
        LDA    MEMTOP+1
        AND    #$F0        ;INSURE 4K PAGE BOUNDARY
        STA    RAMTOP
        LDA    #$40        ;DEFAULT TO UPPER CASE ALPHA AT PWRON
        STA    SHFLOK
        RTS                ;POWER ON COMPLETED
        .PAGE
;
;
; BEGIN DISPLAY HANDLER OPEN PROCESSING
;
DOPEN:  LDA    ICAX2Z        ;GET AUX 2 BYTE
        AND    #$F
        BNE    OPNCOM        ;IF MODE ZERO, CLEAR ICAX1Z
EOPEN:  LDA    ICAX1Z        ;CLEAR "CLR INHIBIT" AND "MXD MODE" BITS
        AND    #$F
        STA    ICAX1Z
        LDA    #0

```



```

OPNCOM: STA    DINDEX
        LDA     #$E0          ;INITIALIZE GLOBAL VBLANK RAM
        STA     CHBAS
        LDA     #2
        STA     CHACT
        STA     SDMCTL        ;TURN OFF DMA NEXT VBLANK
        LDA     #SUCCES
        STA     DSTAT        ;CLEAR STATUS
        LDA     #$C0          ;DO IRQEN
        ORA     POKMSK
        STA     POKMSK
        STA     IRQEN
        LDA     #0
        STA     TINDEX        ;TEXT INDEX MUST ALWAYS BE 0
        STA     ADRESS
        STA     SWPFLG
        STA     CRSINH        ;TURN CURSOR ON ATOPEN
        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     COLOR0,X
        DEX
        BPL     DOPEN8
        LDY     RAMTOP        ;DO TXTMSC=$2C40 (IF MEMTOP=3000)
        DEY
        STY     TXTMSC+1
        LDA     #$60
        STA     TXTMSC
        LDX     DINDEX
        LDA     ANCONV,X      ;CONVERT IT TO ANTIC CODE
        BNE     DOPENA        ;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
        JSR     DBSUB
        DEY
        BNE     DOPEN1
        LDA     GPRIOR        ;ICLEAR GTIA MODES
        AND     #$3F
        STA     OPNTMP+1
        TAY
        CPX     #8            ;TEST IF 320X1
        BCC     NOT8
        TXA
                                ;GET 2 LOW BITS
        ROR     A
        ROR     A
        ROR     A
        AND     #$C0          ;NOW 2 TOP BITS
        ORA     OPNTMP+1
        TAY
        LDA     #16            ;SUBTRACT 16 MORE FOR PAGE BOUNDARY

```

```

        JSR      DBSUB
        CPX      #11          ;TEST MODE 11
        BNE      NOT8        ;IF MODE = 11
        LDA      #6          ;PUT GTIA LUM VALUE INTO BACKGROUND REGISTER
        STA      COLOR4
NOT8:    STY      GPRIOR      ;STORE NEW PRIORITY
        LDA      ADRESS      ;SAVE MEMORY SCAN COUNTER ADDRESS
        STA      SAVMSC
        LDA      ADRESS+1
        STA      SAVMSC+1
VBWAIT: LDA      VCOUNT      ;WAIT FOR NEXT VBLANK BEFORE MESSING
        CMP      #$7A        ;WITH THE DISPLAY LIST
        BNE      VBWAIT
        JSR      DBDEC        ;START PUTTING DISPLAY LIST RIGHT UNDER RAM
        LDA      PAGETB,X     ;TEST IF DISPLAY LIST WILL BE IN TROUBLE
        BEQ      NOMOD        ;OF CROSSING A 256 BYTE PAGE BOUNDARY
        LDA      #$FF        ;IF SO, DROP DOWN A PAGE
        STA      ADRESS
        DEC      ADRESS+1
NOMOD:   LDA      ADRESS      ;SAVE END OF DISPLAY LIST FOR LATER
        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
        STX      OPNTMP
        LDA      #24          ;INITIALIZE BOTSCR
        STA      BOTSCR
        LDA      DINDEXT      ;DISALLOW MIXED MODE IF MODE.GE.9
        CMP      #9
        BCS      NOTMXD
        LDA      ICAX1Z      ;TEST MIXED MODE
        AND      #$10
        BEQ      NOTMXD
        LDA      #4
        STA      BOTSCR
        LDX      #2          ;ADD 4 LINES OF TEXT AT BOTTOM OF SCREEN
DOPEN2: LDA      #2
        JSR      STORE
        DEX
        BPL      DOPEN2
        LDY      RAMTOP      ;RELOAD MSC FOR TEXT
        DEY
        TYA
        JSR      STORE
        LDA      #$60
        JSR      STORE
        LDA      #$42
        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
        JSR      STORE
        DEX

```

```

BNE      DOPEN3
LDA      DINDEX          ;DO THE MESSY 320X1 PROBLEM
CMP      #8
BCC      DOPEN5
LDX      #93              ;GET REMAINING NUMBER OF DLE'S
LDA      RAMTOP           ;RELOAD MEMORY SCAN COUNTER
SEC
SBC      #$10
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
        LDA      HOLD1
        ORA      #$40
        JSR      STORE
        LDA      #$70        ;24 BLANK LINES
        JSR      STORE
        LDA      #$70
        JSR      STORE
        LDA      ADRESS      ;SAVE DISPLAY LIST ADDRESS
        STA      SDLSTL
        LDA      ADRESS+1
        STA      SDLSTL+1
        LDA      #$70        ;ADD LAST BLANK LINE ENTRY
        JSR      STORE        ;POSITION ADRESS=SDLSTL-1
        LDA      ADRESS      ;STORE NEW MEMTOP
        STA      MEMTOP
        LDA      ADRESS+1
        STA      MEMTOP+1
        LDA      SAVADR
        STA      ADRESS
        LDA      SAVADR+1
        STA      ADRESS+1
        LDA      SDLSTL+1
        JSR      STORE
        LDA      SDLSTL
        JSR      STORE
        LDA      DSTAT        ;IF ERROR OCURRED ON ALLOCATION, OPEN THE ED
        BPL      DOPEN9
        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      CLRSCR        ;CLEAR SCREEN
        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      (ADDRESS),Y
        AND      DMASK

SHIFTD: LSR      SHFAMT          ;SHIFT DATA DOWN TO LOW BITS
        BCS      SHIFT1
        LSR      A
        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
        JMP      RETUR2

OUTCHE: LDA      ATACHR          ;TEST FOR CARRIAGE RETURN
        CMP      #CR
        BNE      OUTCHB
        JSR      DOCRWS          ;DO CR
        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      A
        ROL      A
        ROL      A
        ROL      A
        AND      #3
        TAX                      ;X HAS INDEX INTO ATAIN
        TYA                      ;RESTORE ATACHR

```

```

        AND     #$9F           ;STRIP OFF COLUMN ADDRESS
        ORA     ATAINI,X       ;OR IN NEW COLUMN ADDRESS
OUTCH2: STA     CHAR
        JSR     CONVRT
        LDA     CHAR
SHIFTU: LSR     SHFAMT         ;SHIFT UP TO PROPER POSITION
        BCS     SHIFT2
        ASL     A
        JMP     SHIFTU
SHIFT2: AND     DMASK
        STA     TMPCHR         ;SAVE SHIFTED DATA
        LDA     DMASK         ;INVERT MASK
        EOR     #$FF
        AND     (ADDRESS),Y    ;MASK OFF OLD DATA
        ORA     TMPCHR         ;OR IN NEW DATA
        STA     (ADDRESS),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
        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
        LDA     ATACHR         ;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
        LDA     BUFCNT         ;ANYTHING IN THE BUFFER?
        BNE     EGETC3         ;YES
        LDA     ROWCRS         ;NO, SO SAVE BUFFER START ADDRESS
        STA     BUFSTR
        LDA     COLCRS
        STA     BUFSTR+1
EGETC1: JSR     KGETCH         ;LET'S FILL OUR BUFFER
        STY     DSTAT         ;SAVE KEYBOARD STATUS
        LDA     ATACHR         ;TEST FOR CR
        CMP     #CR
        BEQ     EGETC2
        JSR     DOSS           ;NO, GO PRINT IT
        JSR     SWAP           ;JSR DOSS DID SWAP SO SWAP BACK
        LDA     LOGCOL         ;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
EGETC7: DEC      BUFCNT          ;AND RETURN TILL BUFCNT=0
        BEQ      EGETC5
        LDA      DSTAT          ;IF ERR, LOOP ON EGETC7 UNTIL BUFR IS ELIPTIE
        BMI      EGETC7
        JSR      GETCH
        STA      ATACHR
        JMP      SWAP            ;AND RETURN WITHOUT TURNING CURSOR BACK ON
EGETC5: JSR      DOCRWS          ;DO REAL CARRIAGE RETURN
        LDA      #CR            ;AND RETURN EOL
        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
DOSS:   JSR      OFFCRS          ;TURN OFF CURSOR
        JSR      TSTCTL          ;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
        BNE      EOUTC6          ;IF NON-0 DISPLAY RATHER THAN EXECUTE IT
        ASL      ESCFLG
        INX                          ;PROCESS CONTROL CHARACTERS
        LDA      CNTRLS,X        ;GET DISPLACEMENT INTO ROUTINE
        STA      ADRESS
        LDA      CNTRLS+1,X      ;GET HIGH BYTE
        STA      ADRESS+1
        JSR      JSRIND          ;DO COMPUTED JSR
        JSR      RETUR2          ;DO CURSOR
        JMP      SWAP            ;ALL DONE SO RETURN THROUGH RETUR1
;
;
;
;
; END SCREEN EDITOR.
;
;
; BEGIN KEYBOARD HANDLER
;
;
;

```

```

;
KGETC2: LDA    #$FF
        STA    CH
KGETCH: LDA    ICAX1Z          ;TEST LSB OF AUX1 FOR SPECIAL EDITOR READ MO
        LSR    A
        BCS    GETOUT
        LDA    #BRKABT
        LDX    BRKKEY          ;TEST BREAK
        BEQ    K7              ;IF BREAK, PUT BRKABT IN DSTAT AND CRIN ATA
        LDA    CH
        CMP    #$FF
        BEQ    KGETCH
        STA    HOLDCH          ;SAVE CH FOR SHIFT LOCK PROC
        LDX    #$FF          ;"CLEAR" CH
        STX    CH
        JSR    CLICK          ;DO KEYBOARD AUDIO FEEDBACK (A IS OK)
KGETC3: TAX              ;DO ASCCON
        CPX    #$C0          ;TEST FOR CTL & SHIFT TOGETHER
        BCC    ASCCO1
        LDX    #3              ;BAD CODE
ASCCO1: LDA    ATASCI,X
        STA    ATACHR          ;DONE
        CMP    #$80          ;DO NULLS
        BEQ    KGETC2
        CMP    #$81          ;CHECK ATARI KEY
        BNE    KGETC1
        LDA    INVFLG
        EOR    #$80
        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
        STA    SHFLOK          ;SHIFT BIT
        BNE    KGETC2
K2:     CMP    #$84          ;CNTL CAPS/LOWER
        BNE    K3
        LDA    #$80          ;CNTL BIT
        STA    SHFLOK
        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
        CMP    #$40          ;REGULAR SHIFT AND CONTROL TAKE PRECEDENCE
        BCS    K5              ;OVER LOCK
        LDA    ATACHR          ;TEST FOR ALPHA
        CMP    #$61          ;LOWER CASE A
        BCC    K5              ;NOT ALPHA IF LT
        CMP    #$7B          ;LOWER CASE Z+1

```

```

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

```



```

CLRSC3: STA    LOGMAP,Y    ;(Y IS ZERO AFTER CLRSC2 LOOP)
        INY
        CPY    #4
        BCC    CLRSC3
HOME:   JSR    COLCR      ;PLACE COLCRS AT LEFT EDGE
        STA    LOGCOL
        STA    BUFSTR+1
        LDA    #0
        STA    ROWCRS
        STA    COLCRS+1
        STA    BUFSTR
        RTS

;
BS:     LDA    LOGCOL      ;BACKSPACE
        CMP    LMARGN
        BEQ    BS1
BSA:    LDA    COLCRS      ;LEFT EDGE?
        CMP    LMARGN
        BNE    BS3        ;NO
        JSR    DELTIM      ;YES, SEE IF LINE SHOULD BE DELETED
BS3:    JSR    CRSRLF
        LDA    COLCRS
        CMP    RMARGN
        BNE    BS2
        LDA    ROWCRS
        BEQ    BS2
        JSR    CRSRUP
BS2:    LDA    #$20        ;MAKE BACKSPACE DESTRUCTIVE
        STA    ATACHR
        JSR    OUTPLT
BS1:    JMP    DOLCOL      ;AND RETURN
TAB:    JSR    CRSRRT      ;BEGIN SEARCH
        LDA    COLCRS      ;TEST FOR NEW LINE
        CMP    LMARGN
        BNE    TAB1        ;NO
        JSR    DOCR        ;DO CARRIAGE RETURN
        JSR    LOGGET      ;CHECK IF END OF LOGICAL LINE
        BCC    TAB1        ;NO, CONTINUE
        BCS    TAB2        ;(UNCONDITIONAL)
TAB1:    LDA    LOGCOL      ;CHECK FOR TAB STOP
        JSR    BITGET
        BCC    TAB        ;NO, SO KEEP LOOKING
TAB2:    JMP    DOLCOL      ;CULVERT ROW AND COL TO LOGCOL AND RETURN
SETTAB: LDA    LOGCOL
        JMP    BITSET      ;SET BIT IN MAP AND RETURN
CLRTAB: LDA    LOGCOL
        JMP    BITCLR      ;CLEAR " " " " "
INSCHR: JSR    PHACRS
        JSR    GETPLT      ;GET CHARACTER UNDER CURSOR
        STA    INSDAT
        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
        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
        JSR     INCRSB        ;PUT CURSOR OVER NEXT CHARACTER
        PLA
        CMP     LOGCOL        ;TEST NEW LOGCOL AGAINST OLD LOGCOL
        BCS     DELCH2        ;IF OLD.GE.NEW THEN QUIT
        LDA     ROWCRS        ;IS ROW OFF SCREEN?
        CMP     BOTSCR
        BCS     DELCH2        ;YES, SO QUIT
        JSR     GETPLT        ;GET DATA UNDER CURSOR
        LDY     #0
        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
INSLIN: SEC                    ;INSLIN PUTS "1" INTO BIT MAP
INSLIA: JSR     EXTEND        ;ENTRY POINT FOR C=0
        LDA     LMARGN        ;DO CARRIAGE RETURN (NO LF)
        STA     COLCRS
        JSR     CONVRT        ;GET ADDRESS
        LDA     ADRESS        ;SET UP TO=40+FROM (FROM = CURSOR)
        STA     FRMADR
        CLC
        ADC     #40
        STA     TOADR
        LDA     ADRESS+1
        STA     FRMADR+1
        ADC     #0
        STA     TOADR+1
        LDX     ROWCRS        ;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 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
        BNE      DELLI1          ;LOOP
        LDA      LOGMAP+2        ;SET LSB
        ORA      #1
        STA      LOGMAP+2
DELLI2: LDA      LMARGN          ;DELETE LINE OF DATA USING PART OF SCROLL
        STA      COLCRS          ;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
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
        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
        BCC        STROK          ;OK
        BNE        STRERR          ;ERROR
        LDA        APPMHI
        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
        JSR        PUTMSC
        LDA        ROWCRS          ;PUT 10*ROWCRS INTO MLTTMP
        STA        MLTTMP
        LDA        #0
        STA        MLTTMP+1
        LDA        MLTTMP          ;QUICK X8
        ASL        A
        ROL        MLTTMP+1
        STA        HOLD1          ;(SAVE 2X VALUE)
        LDY        MLTTMP+1
        STY        HOLD2
        ASL        A
        ROL        MLTTMP+1
        ASL        A
        ROL        MLTTMP+1
        CLC                          ;ADD IN 2X
        ADC        HOLD1
        STA        MLTTMP
        LDA        MLTTMP+1
        ADC        HOLD2
        STA        MLTTMP+1
        LDX        DINDEXT        ;NOW SHIFT MLTTMP LEFT DHLINE TIMES TO FINIS
        LDY        DHLINE,X        ;MULTIPLY
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
        LDA        COLCRS
        LDX        #7              ;* TRICKY *
CONVR3: DEY

```

```

        BMI        CONVR4
        DEX
        LSR        COLCRS+1
        ROR        A
        ROR        TMPLBT        ;SAVE LOW BITS FOR MASK
        JMP        CONVR3
CONVR4: INY                ;SO Y IS ZERO UPON RETURN FROM THIS ROUTINE
        CLC
        ADC        MLTTMP        ;ADD SHIFTED COLCRS TO MLTThP
        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
        LDA        MLTTMP        ;PREPARE FOR RETURN
        CLC
        ADC        ADDRESS
        STA        ADDRESS
        STA        OLDADR        ;REMEMBER THIS ADDRESS FOR CURSOR
        LDA        MLTTMP+1
        ADC        ADDRESS+1
        STA        ADDRESS+1
        STA        OLDADR+1
        LDA        DMASKT,X
        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
;
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
INCRS2: LDA        COLCRS        ;TEST END OF LINE
        LDX        DINDEX
        CMP        COLUMN,X        ;TEST TABLED VALUE FOR ALL SCREEN MODES
        BEQ        INC2A        ;DO CR IF EQUAL
        CPX        #0                ;MODE 0?
        BNE        INCRS3        ;IF NOT. JUST RETURN
        CMP        RMARGN        ;TEST AGAINST RMARGN
        BEQ        INCRS3        ;EQUAL IS OK
        BCS        INC2A        ;IF GREATER THAN, DO CR
INCRS3: RTS

```

```

INC2A:  CPX    #8                ;CHECK MODE
        BCC    DOCR1            ;NOT 320X1 $O 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
        LDA    LOGCOL          ;TEST LINE OVERRUN
        CMP    #81
        BCC    DOCR1B          ;IF LESS THAN 81 IT IS DEFINITELY NOT LINE 3
        LDA    INSDAT
        BEQ    DOCR            ;ONLY DO LOG LINE OVERFLOW IF INSDAT <>0
        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          ;DON'T EXTEND IF OVERRUN IS INTO MIDDLE OF L
        LDA    INSDAT          ;DON'T EXTEND IF INSDAT IS ZERO
        BEQ    DOCR1A          ;(INSCHR SPECIAL CASE)
        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
        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
        LDY    #24              ;SET UP SCROLL LOOP COUNTER
        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 IN A 0
        CMP    #$9B            ;TO EXTEND BOTTOM LOGICAL LINE
        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
        DEC    HOLD3
        LDA    LOGMAP
        SEC                    ;FOR PARTIAL LINES ROLL IN A "1"
        BPL    DOCR4B          ;AGAIN IF PARTIAL LOGICAL LINE

```

```

        LDA        HOLD3            ;PLACE CURSOR AT NEW LINE NEAR THE BOTTOM
        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
        BEQ        RANGE            ;THEN IT IS IN MIXED NODE AND OK
        LDA        DINDEX            ;IF MODE = 0
        BEQ        RANGE            ;THEN IT IS INEDITOR MODE AND OK
        JSR        EOPEN            ;IF NOT, OPEN EDITOR
RANGE:  LDA        #39                ;***** RANGE CHECK RMARGN ***** SET UP AC
        CMP        RMARGN            ;***** RANGE CHECK RMARGN ***** COMPARE
        BCS        RANGE3            ;***** RANGE CHECK RMARGN ***** BRANCH GE
        STA        RMARGN            ;***** RANGE CHECK RMARGN ***** BAD SO STORE
RANGE3: LDX        DINDEX
        LDA        NOROWS,X          ;CHECK ROWS
        CMP        ROWCRS
        BCC        RNGERR            ; (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
        BNE        RNGERR            ;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
        LDX        BRKKEY            ;CHECK BREAK KEY FLAG
        STA        BRKKEY            ;'CLEAR' BREAK
        BEQ        RNGER2            ;IF BREAK, QUIT IMMEDIATELY AND RETURN TO CI
        RTS
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
      PLA                                ;AWAY FROM RETURN TO CIO)
      LDA      SWPFLG                    ;IF SWAPPED. SWAP BACK
      BPL      RETUR3
      JSR      SWAPA                    ;AND DONT DO RETUR1
RETUR3: JMP     RETUR1                    ;RETURN TO CIO
;
;
;
; 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
      PLA
      ;PROCESS INDEX
      LSR      A
      LSR      A
      LSR      A
      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
        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      A
        ROL      A
        ROL      A
        ROL      A
        AND      #3
        TAX
        LDA      CHAR
        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
        LDA      #LINBUF.AND.$FF
        STA      ADRESS
        LDY      #39
MOVL11: LDA      (TOADR),Y ;SAVE TO DATA
        STA      TMPCHR
        LDA      (FRMADR),Y ;STORE DATA
        STA      (TOADR),Y
        LDA      TMPCHR
        STA      (ADRESS),Y
        DEY
        BPL      MOVL11
        LDA      ADRESS+1 ;SET UP FRMADR=LAST LINE
        STA      FRMADR+1
        LDA      ADRESS
        STA      FRMADR
        CLC
                                ;ADD 40 TO TOADR

```

```

        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 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
        STA      (ADRESS),Y
        INC      ADRESS
        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 BACK ONE
        CLC
        ADC      #40
        STA      LOGCOL
        DEC      HOLD1            ;UP ONE LINE
        JMP      DOLCO1
DOLCO2: CLC                        ;ADD IN COLCRS
        LDA      LOGCOL
        ADC      COLCRS
        STA      LOGCOL
        RTS
;
;
;
; DOBUF: COMPUTE BUFFER COUNT AS THE NUMBER OF BYTES FROM
;         BUFSTR TO END OF LOGICAL LINE WITH TRAILING SPACES REMOVED
;
DOBUF: JSR      PHACRS
        LDA      LOGCOL
        PHA
        LDA      BUFSTR            ;START
        STA      ROWCRS
        LDA      BUFSTR+1
        STA      COLCRS
        LDA      #1
        STA      BUFCNT
DOBUF1: LDX      #23                ;NORMAL
        LDA      SWPFLG            ;IF SWAPPED, ROW 3 IS THE LAST LINE ON SCREE
        BPL      DOB1
        LDX      #3
DOB1:   CPX      ROWCRS            ;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
        CMP      LMARGN
        BNE      DOBUF1          ;NOT YET EOL
        DEC      ROWCRS          ;BACK UP ONE INCRSR
        JSR      CRSRLF
DOBUF2: JSR      GETPLT          ;TEST CURRENT COLUMN FOR NON-ZERO DATA
        BNE      DOBUF4          ;QUIT IF NON-ZERO
        DEC      BUFCNT          ;DECREMENT COUNTER
        LDA      LOGCOL          ;BEGINNING OF LOGICAL LINE YET?
        CMP      LMARGN
        BEQ      DOBUF4          ;YES, SO QUIT
        JSR      CRSRLF          ;BACK UP CURSOR
        LDA      COLCRS          ;IF LOGCOL=RMARGN, GO UP 1 ROW
        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
        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
        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
DELT11: LDA    (ADDRESS),Y
        BNE     DELT13          ;FOUND A NON-0 SD QUIT AND RETURN
        DEY
        BPL     DELT11
DELT12: JMP     DELLIB          ;DELETE A LINE AND RETURN
DELT13: 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 DOESRETURN
        JMP     RETUR1
SWAPA:  LDA     BOTSCR
        CMP     #24
        BEQ     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
        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
        STA      ADRESS+1
        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

```

```

        LDA     #1
        STA     ROWINC           ;SET UP INITIAL DIRECTIONS
        STA     COLINC
        SEC
        LDA     NEWROW           ;DETERMINE DELTA ROW
        SBC     OLDROW
        STA     DELTAR
        BCS     DRAW1           ;DO DIRECTION AND ABSOLUTE VALUE
        LDA     #$FF             ;BORROW WAS ATTEMPTED
        STA     ROWINC           ;SET DIRECTION DOWN
        LDA     DELTAR
        EOR     #$FF             ;DELTAR = |DELTAR|
        CLC
        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
        INC     DELTAC+1
DRAW2:   LDX     #2               ;ZERO RAM FOR DRAW LOOP
        LDY     #0
        STY     COLAC+1
DRAW3A:  TYA
        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
        TAY
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
        ROR     A          ;ROR THE 9 BIT ACUMULATOR
        STA     ROWAC,X
DRAW4A:  LDA     COUNTR      ;TEST ZERO
        ORA     COUNTR+1
        BNE     DRAWI1      ;IF COUNTER IS ZERO, LEAVE DRAW
        JMP     DRAW10
DRAWI1:  CLC
                ;ADD ROW TO ROWAC (PLOT LOOP)
        LDA     ROWAC
        ADC     DELTAR
        STA     ROWAC
        BCC     DRAW5
        INC     ROWAC+1
DRAW5:   LDA     ROWAC+1    ;COMPARE ROW TO ENDPOINT
        CMP     ENDPT+1    ;IF HIGH BYTE OF ROW IS .LT. HIGH
        BCC     DRAW6      ;BYTE OF ENDPT, BLT TO COLUMN
        BNE     DRAW5A
        LDA     ROWAC
        CMP     ENDPT      ;LOW BYTE
        BCC     DRAW6      ;ALSO TILT
DRAW5A:  CLC
                ;GE SO MOVE POINT
        LDA     ROWCRS
        ADC     ROWINC
        STA     ROWCRS
        LDX     #0          ;AND SUBTRACT ENDPT FROM ROWAC
        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
DRAW6A:  BIT     COLINC     ;+ OR - ?
        BPL     DRAW6B
        DEC     COLCRS      ;DO DOUBLE BYTE DECREMENT
        LDA     COLCRS
        CMP     #$FF
        BNE     DRAW7
        LDA     COLCRS+1
        BEQ     DRAW7      ;DON'T DEC IF ZERO
        DEC     COLCRS+1
        BPL     DRAW7      ;(UNCONDITIONAL)
DRAW6B:  INC     COLCRS     ;DO DOUBLE BYTE INCREMENT
        BNE     DRAW7
        INC     COLCRS+1
DRAW7:   LDX     #2
                ;AND SUBTRACT ENDPT FROM COLAC
        JSR     SUBEND

```



```

DRAW8:  JSR    RANGE
        JSR    OUTPLT      ;PLOT POINT
        LDA    FILFLG      ;TEST RIGHT FILL
        BEQ    DRAW9
        JSR    PHACRS
        LDA    ATACHR
        STA    HOLD4
DRAW8A:  LDA    ROWCRS      ;SAVE ROW IN CASE OF CR
        PHA
        JSR    INCRSA      ;POSITION CURSOR ONE PAST DOT
        PLA            ;RESTORE ROWCRS
        STA    ROWCRS
DRAW8C:  JSR    RANGE
        JSR    GETPLT      ;GET DATA
        BNE    DRAW8B      ;STOP IF NON-ZERO DATA IS ENCOUNTERED
        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
        LDA    COUNTR+1
        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    23,23,11,23,47,47,95,95,97,97,97,97

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     " "
;      3      8      40X4X8     GRAPHICS
;      4      9      80X2X4      " "
;      5      A      80X4X4      " "
;      4      B      160X2X2     " "
;      7      D      160X4X2     " "
;      8      F      320X2X1     " "
;      9      SAME AS 8 BUT GTIA 'LUM' MODE
;     10      SAME AS 8 BUT GTIA 'COL/LUM REGISTER' MODE
;     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    0,0,0,0,0,0,0,1,1,1,1,1

```

```

;
;
; 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
        .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    INCHR
;
;
;
;
; ATAINT: ATASCI TO INTERNAL TABLE
;

```

```

ATAINT: .BYTE    $40,$00,$20,$60
;
;
; INTATA: INTERNAL TO ATASCI TABLE
;
INTATA: .BYTE    $20,$40,$00,$60
;
;
; ATASCI: ATASCI 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
        BNE    PIRQ3          ;IF NOT, GO PROCESS KEY
        LDA    KEYDEL         ;IF KEY DELAY BYTE > 0
        BNE    PIRQ4          ;IGNORE KEY AS BOUNCE
PIRQ3:  LDA    KBCODE          ;RESTORE AC
        CMP    #CNTL1         ;TEST CONTROL 1 (SSFLAG)
        BNE    PIRQ1
        LDA    SSFLAG
        EOR    #$FF
        STA    SSFLAG
        BCS    PIRQ4          ;(UNCONDITIONAL) MAKE ^1 INVISIBLE
PIRQ1:  STA    CH
        STA    CH1
        LDA    #3
        STA    KEYDEL         ;INITIALIZE KEY DELAY FOR DEBOUNCE
        LDA    #0              ;CLEAR COLOR SHIFT BYTE
        STA    ATRACT
PIRQ4:  LDA    #$30
        STA    SRTIMR
PIRQ2:  PLA
        RTI
;
;
        .BYTE $FF,$FF,$FF,$FF,$FF,$FF

;
CRNTPC  =*
        *=$14
KBDSPR: .BYTE $FFF8-CRNTPC ;^GDISPLC IS TOO LONG
        .END

```