OS, Operating System, Revision B, NTSC, EPROM version, for 400/800, for Data General cross assembler

The document itself begins on the next page.

Document source:

Original backup tapes owned by Dutchman2000, obtained by Atarimania.
Documentary research and PDF layout by Laurent Delsarte.
Note that these backup tapes contain A LOT of information spread out in many folders, meaning it will take time to process the important bits.

Document identification:

Original file name:	REVBAMAC.ASM & REVBAMAC.PRN
Title of document:	OS, Operating System, Revision B, NTSC, EPROM version, for 400/800, for Data General cross assembler
Author(s):	(multiple, not listed)
Original file date:	1984-08-21 ('8/21/84' date found in document)
Type of document:	Software, commented source code
Target audience:	Internal
Status:	Ready
Reference (Atari):	(unknown)
Reference (Laurent Delsarte):	For any discussion, this PDF has been given the reference **BKUP-1984-08-21-SOFT-0010A-2** which should be quoted in any communication.
Tags:	#Atari #8bit #6502 #400 #800 #OS #REV.B #NTSC #EPROM #DataGeneral

Comments:

This is the printout of the original source code, fully commented.
It contains the .ASM version, followed by the. PRN version.

Interesting facts:

- This version is for the Data General cross assembler.
- "This is the Revision B EPROM version".
- The same source code is used for both the NTSC and PAL versions. Only a "PALFLG" flag is used to generate one version or the other at compile time.
- Differences:
  - in HITIMR:, "LDA #-$83" (PAL, Microtec cross assembler) vs. "LDA #$100-$83" (NTSC, Data General cross assembler). Is this a mistake? Is this a value specific to the compiler?
- GTIA is mentioned, even though the GTIA versions of the 400/800 were not available until 1982.
- See "This is to make DOS 2 work which used an absolute address".
- Al Miller [MILLER, Alan] 's name appears 3 times.
- The "draw" section uses the "Al Miller method from Basketball".

Present in this NTSC version but missing in the PAL (the other) version:
(See the end of the listing)

- Additions to Revision B to generate exact ROM image [CHARSET.ASM]
- Holes with random data bytes at $EDE8 & $E90B
- Checksum & hardware vectors from $FFF8 to $FFFF

Formatting:

- Comments are in grey, so that the black text is easier to read.
- Important 6502 instructions are highlighted in fluo.
  `JMP` `JSR` `RTS` `BEQ` `BNE` `BMI` `BPL` `BCS` `BCC` `BVS` `BVC`

What makes this document so interesting are all these valuable comments.

This page intentionally left blank

This page intentionally left blank

OS, Operating System, Revision B, NTSC, EPROM version, for 400/800, for Data General cross assembler [REVBAMAC.ASM]

```
;LIST X
; THIS IS THE MODIFIED SEPTEMBER ATARI 400/800 COMPUTER OPERATING
; SYSTEM LISTING, MODIFIED TO ASSEMBLE ON THE DATA GENERAL CROSS ASSEMBLER.
;
; 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
       EJECT
```

```
;
;
;       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.
;
DISKIV  =       $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 (E)
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
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 OVERRANGE
OVRRUN  =       $8E         ;SERIAL BUS DATA OVERRUN
CHKERR  =       $8F         ;SERIAL BUS CHECKSUM ERROR
;
DERROR  =       $90         ;PERIPHERAL DEVICE ERROR (OPERATION NOT COMPLETED)
BADMOD  =       $91         ;BAD SCREEN MODE NUMBER
FNCNOT  =       $92         ;FUNCTION NOT IMPLEMENTED IN HANDLER
SCRMEM  =       $93         ;INSUFICIENT MEMORY FOR SCREEN MODE
;
;
;
;
;
;
;       PAGE ZERO RAM ASSIGNMENTS
;
        ORG  $0000
LINZBS: DS      2           ;LINBUG RAM (WILL BE REPLACED BY MONITOR RAM)
;
; THESE LOCATIONS ARE NOT CLEARED
CASINI: DS      2           ;CASSETTE INIT LOCATION
RAMLO:  DS      2           ;RAM POINTER FOR MEMORY TEST
TRAMSZ: DS      1           ;TEMPORARY REGISTER FOR RAM SIZE
TSTDAT: DS      1           ;RAM TEST DATA REGISTER
;
; CLEARED ON COLDSTART ONLY
WARMST: DS      1           ;WARM START FLAG
BOOT?:  DS      1           ;SUCCESSFUL BOOT FLAG
DOSVEC: DS      2           ;DISK SOFTWARE START VECTOR
DOSINI: DS      2           ;DISK SOFTWARE INIT ADDRESS
APPMHI: DS      2           ;APPLICATIONS MEMORY HI LIMIT
;
; CLEARED ON COLD OR WARM START
INTZBS  =    *              ;INTERRUPT HANDLER
POKMSK: DS      1           ;SYSTEM MASK FOR POKEY IRQ ENABLE
BRKKEY: DS      1           ;BREAK KEY FLAG
RTCLOK: DS      3           ;REAL TIME CLOCK (IN 16 MSEC UNITS)
;
BUFADR: DS      2           ;INDIRECT BUFFER ADDRESS REGISTER
;
ICCOMT: DS      1           ;COMMAND FOR VECTOR
;
DSKFMS: DS      2           ;DISK FILE MANAGER POINTER
DSKUTL: DS      2           ;DISK UTILITIES POINTER
;
PTIMOT: DS      1           ;PRINTER TIME OUT REGISTER
PBPNT:  DS      1           ;PRINT BUFFER POINTER
PBUFSZ: DS      1           ;PRINT BUFFER SIZE
PTEMP:  DS      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: DS      1           ;HANDLER INDEX NUMBER (FF = IOCB FREE)
ICDNOZ: DS      1           ;DEVICE NUMBER (DRIVE NUMBER)
ICCOMZ: DS      1           ;COMMAND CODE
```

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

```
OPNTMP  =       MLTTMP      ;FIRST BYTE IS USED IN OPEN AS TEMP
SAVADR: DS      2
RAMTOP: DS      1           ;RAM SIZE DEFINED BY POWER ON LOGIC
BUFCNT: DS      1           ;BUFFER COUNT
BUFSTR: DS      2           ;EDITOR GETCH POINTER
BITMSK: DS      1           ;BIT MASK
SHFAMT: DS      1
ROWAC:  DS      2
COLAC:  DS      2
ENDPT:  DS      2
DELTAR: DS      1
DELTAC: DS      2
ROWINC: DS      1
COLINC: DS      1
SWPFLG: DS      1           ;NON-0 IF TXT AND REGULAR RAM IS SWAPPED
HOLDCH: DS      1           ;CH IS MOVED HERE IN KGETCH BEFORE CNTL & SHIFT PROC
INSDAT: DS      1
COUNTR: DS      2
;
;
;
;       80 - FF ARE RESERVED FOR USER APPLICATIONS
;
;
;
;       NOTE : SEE FLOATING POINT SUBROUTINE AREA FOR ZERO PAGE CELLS
;
;
;
;
;       PAGE 1   -   STACK
;
;
;
;
;       PAGE TWO RAM ASSIGNMENTS
;
        ORG  $0200
INTABS  =       *           ;INTERRUPT RAM
VDSLST: DS      2           ;DISPLAY LIST NMI VECTOR
VPRCED: DS      2           ;PROCEED LINE IRQ VECTOR
VINTER: DS      2           ;INTERRUPT LINE IRQ VECTOR
VBREAK: DS      2           ;SOFTWARE BREAK (00) INSTRUCTION IRQ VECTOR
VKEYBD: DS      2           ;POKEY KEYBOARD IRQ VECTOR
VSERIN: DS      2           ;POKEY SERIAL INPUT READY IRQ
VSEROR: DS      2           ;POKEY SERIAL OUTPUT READY IRQ
VSEROC: DS      2           ;POKEY SERIAL OUTPUT COMPLETE IRQ
VTIMR1: DS      2           ;POKEY TIMER 1 IRQ
VTIMR2: DS      2           ;POKEY TIMER 2 IRQ
VTIMR4: DS      2           ;POKEY TIMER 4 IRQ
VIMIRQ: DS      2           ;IMMEDIATE IRQ VECTOR
CDTMV1: DS      2           ;COUNT DOWN TIMER 1
CDTMV2: DS      2           ;COUNT DOWN TIMER2
CDTMV3: DS      2           ;COUNT DOWN TIMER 3
CDTMV4: DS      2           ;COUNT DOWN TIMER 4
CDTMV5: DS      2           ;COUNT DOWN TIMER 5
VVBLKI: DS      2           ;IMMEDIATE VERTICAL BLANK NMI VECTOR
VVBLKD: DS      2           ;DEFERRED VERTICAL BLANK NMI VECTOR
CDTMA1: DS      2           ;COUNT DOWN TIMER 1 JSR ADDRESS
CDTMA2: DS      2           ;COUNT DOWN TIMER 2 JSR ADDRESS
CDTMF3: DS      1           ;COUNT DOWN TIMER 3 FLAG
SRTIMR: DS      1           ;SOFTWARE REPEAT TIMER
CDTMF4: DS      1           ;COUNT DOWN TIMER 4 FLAG
INTEMP: DS      1           ;IAN'S TEMP (RENAMED FROM T1 BY POPULAR DEMAND)
```

```
CDTMF5: DS      1           ;COUNT DOWN TIMER FLAG 5
SDMCTL: DS      1           ;SAVE DMACTL REGISTER
SDLSTL: DS      1           ;SAVE DISPLAY LIST LOW BYTE
SDLSTH: DS      1           ;SAVE DISPLAY LIST HI BYTE
SSKCTL: DS      1           ;SKCTL REGISTER RAM
        DS      1           ;
;
LPENH:  DS      1           ;LIGHT PEN HORIZONTAL VALUE
LPENV:  DS      1           ;LIGHT PEN VERTICAL VALUE
BRKKY:  DS      2           ;BREAK KEY VECTOR
;
        DS      2           ;SPARE
;
CDEVIC: DS      1           ;COMMAND FRAME BUFFER - DEVICE
CCOMND: DS      1           ;COMMAND
CAUX1:  DS      1           ;COMMAND AUX BYTE 1
CAUX2:  DS      1           ;COMMAND AUX BYTE 2
;   NOTE: MAY NOT BE THE LAST WORD ON A PAGE
TEMP:   DS      1           ;TEMPORARY RAM CELL
;   NOTE: MAY NOT BE THE LAST WORD ON A PAGE
ERRFLG: DS      1           ;ERROR FLAG - ANY DEVICE ERROR EXCEPT TIME OUT
;
DFLAGS: DS      1           ;DISK FLAGS FROM SECTOR ONE
DBSECT: DS      1           ;NUMBER OF DISK BOOT SECTORS
BOOTAD: DS      2           ;ADDRESS WHERE DISK BOOT LOADER WILL BE PUT
COLDST: DS      1           ;COLDSTART FLAG (1=IN MIDDLE OF COLDSTART)
;
        DS      1           ;SPARE
;
DSKTIM: DS      1           ;DISK TIME OUT REGISTER
;
LINBUF: DS      40          ;CHAR LINE BUFFER
;
GPRIOR: DS      1           ;GLOBAL PRIORITY CELL
;
PADDL0: DS      1           ;POTENTIOMETER 0 RAM CELL
PADDL1: DS      1
PADDL2: DS      1
PADDL3: DS      1
PADDL4: DS      1
PADDL5: DS      1
PADDL6: DS      1
PADDL7: DS      1
STICK0: DS      1           ;JOYSTICK 0 RAM CELL
STICK1: DS      1
STICK2: DS      1
STICK3: DS      1
PTRIG0: DS      1           ;PADDLE TRIGGER 0
PTRIG1: DS      1
PTRIG2: DS      1
PTRIG3: DS      1
PTRIG4: DS      1
PTRIG5: DS      1
PTRIG6: DS      1
PTRIG7: DS      1
STRIG0: DS      1           ;JOYSTICK TRIGGER 0
STRIG1: DS      1
STRIG2: DS      1
STRIG3: DS      1
;
CSTAT:  DS      1
WMODE:  DS      1
BLIM:   DS      1
IMASK:  DS      1
JVECK:  DS      2
```

```
;
        DS      2           ;SPARE
;
;
;
;
TXTROW: DS      1           ;TEXT ROWCRS
TXTCOL: DS      2           ;TEXT COLCRS
TINDEX: DS      1           ;TEXT INDEX
TXTMSC: DS      2           ;FOOLS CONVRT INTO NEW MSC
TXTOLD: DS      6           ;OLDROW & OLDCOL FOR TEXT (AND THEN SOME)
TMPX1:  DS      1
HOLD3:  DS      1
SUBTMP: DS      1
HOLD2:  DS      1
DMASK:  DS      1
TMPLBT: DS      1
ESCFLG: DS      1           ;ESCAPE FLAG
TABMAP: DS      15
LOGMAP: DS      4           ;LOGICAL LINE START BIT MAP
INVFLG: DS      1           ;INVERSE VIDEO FLAG (TOGGLED BY ATARI KEY)
FILFLG: DS      1           ;RIGHT FILL FLAG FOR DRAW
TMPROW: DS      1
TMPCOL: DS      2
SCRFLG: DS      1           ;SET IF SCROLL OCCURS
HOLD4:  DS      1           ;TEMP CELL USED IN DRAW ONLY
HOLD5:  DS      1           ;DITTO
SHFLOK: DS      1
BOTSCR: DS      1           ;BOTTOM OF SCREEN : 24 NORM 4 SPLIT
;
;
PCOLR0: DS      1           ;P0 COLOR
PCOLR1: DS      1           ;P1 COLOR
PCOLR2: DS      1           ;P2 COLOR
PCOLR3: DS      1           ;P3 COLOR
COLOR0: DS      1           ;COLOR 0
COLOR1: DS      1
COLOR2: DS      1
COLOR3: DS      1
COLOR4: DS      1
;
;
        DS      23          ;SPARE
;
;
;
GLBABS  =    *              ;GLOBAL VARIABLES
;
        DS      4           ;SPARE
;
RAMSIZ: DS      1           ;RAM SIZE (HI BYTE ONLY)
MEMTOP: DS      2           ;TOP OF AVAILABLE USER MEMORY
MEMLO:  DS      2           ;BOTTOM OF AVAILABLE USER MEMORY
        DS      1           ;SPARE
DVSTAT: DS      4           ;STATUS BUFFER
CBAUDL: DS      1           ;CASSETTE BAUD RATE LOW BYTE
CBAUDH: DS      1
;
CRSINH: DS      1           ;CURSOR INHIBIT (00 = CURSOR ON)
KEYDEL: DS      1           ;KEY DELAY
CH1:    DS      1
;
CHACT:  DS      1           ;CHACTL REGISTER RAM
CHBAS:  DS      1           ;CHBAS REGISTER RAM
;
```

```
        DS      5           ;SPARE BYTES
;
CHAR:   DS      1
ATACHR: DS      1           ;ATASCII CHARACTER
CH:     DS      1           ;GLOBAL VARIABLE FOR KEYBOARD
FILDAT: DS      1           ;RIGHT FILL DATA (DRAW)
DSPFLG: DS      1           ;DISPLAY FLAG : DISPLAY CNTLS IF NON-ZERO
SSFLAG: DS      1           ;START/STOP FLAG FOR PAGING (CNTL 1). CLEARED BY BREAK
;
;
;
;
;
;
;
;       PAGE THREE RAM ASSIGNMENTS
;
DCB     =    *              ;DEVICE CONTROL BLOCK
DDEVIC: DS      1           ;PERIPHERAL UNIT 1 BUS I.D. NUMBER
DUNIT:  DS      1           ;UNIT NUMBER
DCOMND: DS      1           ;BUS COMMAND
DSTATS: DS      1           ;COMMAND TYPE/STATUS RETURN
DBUFLO: DS      1           ;DATA BUFFER POINTER LOW BYTE
DBUFHI: DS      1
DTIMLO: DS      1           ;DEVICE TIME OUT IN 1 SECOND UNITS
DUNUSE: DS      1           ;UNUSED BYTE
DBYTLO: DS      1           ;NUMBER OF BYTES TO BE TRANSFERRED LOW BYTE
DBYTHI: DS      1
DAUX1:  DS      1           ;COMMAND AUXILIARY BYTE 1
DAUX2:  DS      1
;
TIMER1: DS      2           ;INITIAL TIMER VALUE
ADDCOR: DS      1           ;ADDITION CORRECTION
CASFLG: DS      1           ;CASSETTE MODE WHEN SET
TIMER2: DS      2           ;FINAL TIMER VALUE. THESE TWO TIMER VALUES
; ARE USED TO COMPUTE INTERVAL FOR BAUD RATE
TEMP1:  DS      2           ;TEMPORARY STORAGE REGISTER
TEMP2:  DS      1           ;TEMPORARY STORAGE REGISTER
TEMP3:  DS      1           ;TEMPORARY STORAGE REGISTER
SAVIO:  DS      1           ;SAVE SERIAL IN DATA PORT
TIMFLG: DS      1           ;TIME OUT FLAG FOR BAUD RATE CORRECTION
STACKP: DS      1           ;SIO STACK POINTER SAVE CELL
TSTAT:  DS      1           ;TEMPORARY STATUS HOLDER
;
;
;
HATABS: DS      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:  DS      1           ;HANDLER INDEX NUMBER (FF = IOCB FREE)
ICDNO:  DS      1           ;DEVICE NUMBER (DRIVE NUMBER)
ICCOM:  DS      1           ;COMMAND CODE
ICSTA:  DS      1           ;STATUS OF LAST IOCB ACTION
ICBAL:  DS      1           ;BUFFER ADDRESS LOW BYTE
ICBAH:  DS      1
ICPTL:  DS      1           ;PUT BYTE ROUTINE ADDRESS - 1
ICPTH:  DS      1
ICBLL:  DS      1           ;BUFFER LENGTH LOW BYTE
ICBLH:  DS      1
ICAX1:  DS      1           ;AUXILIARY INFORMATION FIRST BYTE
ICAX2:  DS      1
ICSPR:  DS      4           ;FOUR SPARE BYTES
```

```
        DS      MAXIOC-IOCBSZ
;
PRNBUF: DS      40          ;PRINTER BUFFER
;
        DS      21          ;SPARE BYTES
;
;
;
;
;
;
;
;       PAGE FOUR RAM ASSIGNMENTS
;
CASBUF: DS      131         ;CASSETTE BUFFER
;
; USER AREA STARTS HERE AND GOES TO END OF PAGE FIVE
USAREA: DS      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   =       $DD8D       ;   "      "    "     FR0  <- (FLPTR)
FLD1R   =       $DD98       ;   "      "   REG1   FR1  <- (X,Y)
FLD1P   =       $DD9C       ;   "      "    "     FR1  <- (FLPTR)
FST0R   =       $DDA7       ;FLOATING STORE REG0 (X,Y) <- FR0
FST0P   =       $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     =       $BD81       ;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 F.P. ROUTINES ARE CALLED)
        ORG  $D4
FR0:    DS      FPREC       ;FP REG0
FRE:    DS      FPREC
FR1:    DS      FPREC       ;FP REG1
FR2:    DS      FPREC
FRX:    DS      1           ;FP SPARE
EEXP:   DS      1           ;VALUE OF E
NSIGN:  DS      1           ;SIGN OF #
ESIGN:  DS      1           ;SIGN OF EXPONENT
FCHRFLG:DS      1           ;1ST CHAR FLAG
DIGRT:  DS      1           ;# OF DIGITS RIGHT OF DECIMAL
CIX:    DS      1           ;CURRENT INPUT INDEX
INBUFF: DS      2           ;POINTS TO USER'S LINE INPUT BUFFER
ZTEMP1: DS      2
ZTEMP4: DS      2
ZTEMP3: DS      2
DEGFLG
RADFLG: DS      1           ;0=RADIANS, 6=DEGREES
RADON   =       0           ;INDICATES RADIANS
DEGON   =       6           ;INDICATES DEGREES
FLPTR:  DS      2           ;POINTS TO USER'S FLOATING PT NUMBER
FPTR2:  DS      2
; FLOATING PT ROUTINES' NON-ZERO PAGE RAM
; (NEEDED ONLY IF F.P. ROUTINES CALLED)
        ORG  $57E
LBPR1:  DS      1           ;LBUFF PREFIX 1
LBPR2:  DS      1           ;LBUFF PREFIX 2
LBUFF:  DS      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
GRACTL  =       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+6
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:
HSCROL  =       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]
;
;
;
; EJECT
        EJECT
```

```
        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
        EJECT
```

```
;
; CIO JUMP VECTOR FOR USERS
        ORG  CIOV
        JMP     CIO         ;GO TO CIO
;
; CIO INIT JUMP VECTOR FOR POWER UP
        ORG  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"
;
;
        ORG  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  =    HIGH ERRTN
ERRTNL  =    LOW ERRTN
        LDY     #NOTOPN     ;IOCB NOT OPEN
        RTS
        EJECT
```

```
;
; 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
        EJECT
```

```
;
; 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
        EJECT
```

```
;
;
; 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
        EJECT
```

```
;
; 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
        EJECT
```

```
;
; 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 IOCB
;
; TRANSFER DONE
RCI4:   JSR     SUBBFL      ;SET FINAL BUFFER LENGTH
        JMP     CIRTN2      ;RETURN
        EJECT
```

```
;
; 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 BUFFER LENGTH=1
        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
        EJECT
```

```
;
; 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
        EJECT
```

```
;
; 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
        EJECT
```

```
;
;
; 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     ICBLH,X     ;GET HI BYTE
        SBC     ICBLLZ+1
        STA     ICBLHZ
        RTS
;
;
; GOHAND -- GO INDIRECT TO A DEVICE HANDLER
; Y= STATUS ON RETURN, N FLAG=1 IF ERROR ON RETURN
GOHAND: LDY     #FNCNOT     ;PREPARE NO FUNCTION STATUS FOR HANDLER RTS
        JSR     CIJUMP      ;USE THE INDIRECT JUMP
```

```
        STY     ICSTAZ      ;SAVE STATUS
        CPY     #0          ;AND SET N FLAG
        RTS
;
; INDIRECT JUMP TO HANDLER BY PAUL'S METHOD
CIJUMP: TAX                 ;SAVE A
        LDA     ICSPRZ+1    ;GET JUMP ADDRESS HI BYTE
        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
        EJECT
```

```
;
; 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
        EJECT
```

```
;
;
; CIO ROM TABLES
;
; COMMAND TABLE
; MAPS EACH COMMAND TO OFFSET FOR APPROPRIATE VECTOR IN HANDLER
COMTAB: DB      0,4,4,4,4,6,6,6,6,2,8,10
LENGTH  =    *-CIOINT
CRNTP1  =    *
        ORG  $14
CIOSPR: DB      INTORG-CRNTP1 ;^GCIOL IS TOO LONG
        EJECT
```

```
        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
;
        ORG   $E912
        JMP     SETVBL
        ORG   SETVBV
        JMP     SETVBL
        JMP     SYSVBL
        JMP     XITVBL
        ORG   INTINV
        JMP     IHINIT
;
        ORG   VCTABL+INTABS-VDSLST
;
        DW      SYRTI       ;VDSLST
        DW      SYIRQB      ;VPRCED
        DW      SYIRQB      ;VINTER
        DW      SYIRQB      ;VBREAK
;
        DS      8
        DW      SYIRQB      ;VTIMR1
        DW      SYIRQB      ;VTIMR2
        DW      SYIRQB      ;VTIMR4
        DW      SYIRQ       ;VIMIRQ
        DW      0,0,0,0,0   ;CDTMV1-4
        DW      SYSVBL      ;VVBLKI
        DW      XITVBL      ;VVBLKD
;
        ORG   $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
        EJECT
```

```
;
; IRQ HANDLER
;
; JUMP THRU IMMEDIATE IRQ VECTOR, WHICH ORDINARILY POINTS TO
; SYSTEM IRQ: DETERMINE & CLEAR CAUSE, JUMP THRU SOFTWARE VECTOR.
;
        ORG  INTORG
IHINIT: LDA     #$40        ;VBL ON BUF DLIST OFF***FOR NOW***
        STA     NMIEN       ;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     (VIMIRQ)
CMPTAB: DB      $80         ;BREAK KEY
        DB      $40         ;KEY STROKE
        DB      $04         ;TIMER 4
        DB      $02         ;TIMER 2
        DB      $01         ;TIMER 1
        DB      $08         ;SERIAL OUT COMPLETE
        DB      $10         ;SERIAL OUT READY
        DB      $20         ;SERIAL  IN READY

; THIS IS A TABLE OF OFFSETS INTO PAGE 2.  THEY POINT TO
ADRTAB: DB      BRKKY-INTABS
        DB      VKEYBD-INTABS
        DB      VTIMR4-INTABS
        DB      VTIMR2-INTABS
        DB      VTIMR1-INTABS
        DB      VSEROC-INTABS
        DB      VSEROR-INTABS
        DB      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 X ONTO STACK
        LDX     #$6         ;START WITH SIX OFFSET
LOOPM:  LDA     CMPTAB,X    ;LOAD MASK
        CPX     #5          ;CHECK TO SEE IF COMPLETE IS SET
        BNE     LOOPM2
        AND     POKMSK      ;IS THIS INTERUPT ENABLED?
        BEQ     LL
LOOPM2: BIT     IRQST       ; IS IT THE INTERUPT?
        BEQ     JMPP
LL:     DEX                 ;NO DEC X AND TRY NEXT MASK
        BPL     LOOPM       ;IF NOT NEG GOTO LOOPM
        JMP     SYIRQ8      ;DONE BUT NO INTERUPT
JMPP:   EOR     #$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.
        EJECT
```

```
;
; 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      ;GO THRU WARM START JUMP
        TXA                 ;SAVE REGISTERS
        PHA
        TYA
        PHA
        STA     NMIRES      ;RESET INTERRUPT STATUS
        JMP     (VVBLKI)    ;JUMP THRU VECTOR
        EJECT
```

```
;
; SYSTEM VBLANK ROUTINE
;
; INC FRAME COUNTER. PROCESS COUNTDOWN TIMERS. EXIT IF I WAS SET, CLEAR I.
; 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        ;[ATRACT] 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 ATRACT, SO STAY BY STA $FE
        LDX     RTCLOK+1    ;COLOR SHIFT FOLLOWS RTCLOK+1
        LDA     #$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 INTERUPTS
        LDA     PCOLR0,X    ;LOAD COLOR REGISTERS FROM RAM
        EOR     COLRSH      ;DO COLOR SHIFT
        AND     DRKMSK      ;AND DARK ATRACT
        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    =       HIGH SYSVB7
SV7L    =       LOW 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
        EJECT
```

```
;
; SUBROUTINE TO SET VERTICAL BLANK VECTORS AND TIMERS
; ENTRY X=HI,Y=LO BYTE TO SET
;       A= 1-5 TIMERS 1-5
;          6 IMM VBLANK
;          7 DEF VBLANK
;
SETVBL: ASL     A           ;MUL BY 2
        STA     INTEMP
        TXA
        LDX     #5
        STA     WSYNC       ;WASTE 20 CPU CYCLES
SETLOP: DEX                 ;TO ALOWD VBLANK TO HAPPEN
        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   =       HIGH PIRQ
PIRQL   =       LOW PIRQ
PNMIH   =       HIGH PNMI
PNMIL   =       LOW PNMI
; SPARE BYTE OR MODULE TOO LONG FLAG
CRNTP2  =    *
        ORG  $14
INTSPR: DB      SIOORG-CRNTP2 ;^GINTHV IS TOO LONG
        EJECT
```

```
        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-79
;
;
; 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.
;
;
;
;
        EJECT
```

```
; EQUATES
;
; DCD DEVICE BUS ID NUMBERS
FLOPPY  =       $30
;PRINTR =       $40
;CASSET =       $60          ;!!!!! *****
CASET   =       $60          ;!!!!! *****
;
;
;BUS COMMANDS
;
READ    =       'R
WRITE   =       'W
;STATIS = 'S
;FORMAT = '!
;
;
; COMMAND AUX BYTES
;
SIDWAY  =       'S'          ;PRINT 16 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 OPERATION
;
;
; MISCELLANEOUS EQUATES
;
B192LO  =       $28          ;19200 BAUD RATE POKEY COUNTER VALUES (LO BYTE)
B192HI  =       $00          ;(HI BYTE)
B600LO  =       $CC          ;600 BAUD (LO BYTE)
B600HI  =       $05          ;(HI BYTE)
HITONE  =       $05          ;FSK HI FREQ POKEY COUNTE 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  =       HIGH TEMP    ;ADDRESS OF TEMP CELL (HI BYTE)
TEMPLO  =       LOW TEMP     ;(LO BYTE)
CBUFHI  =       HIGH CDEVIC ;ADDRESS OF COMMAND BUFFER (HI BYTE)
```

```
CBUFLO  =       LOW 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  =      HIGH JTIMER ;HI BYTE OF JUMP TIMER ROUTINE ADDR
                           ; "MOVED TO LINE 1427"
;JTADRL  =      LOW JTIMER  ; "MOVED TO LINE 1428"
;
        EJECT
```

```
;       SIO
;
;
        ORG  SIOV
        JMP     SIO         ;SIO ENTRY POINT
;
        ORG  SIOINV
        JMP     SIOINT      ;SIO INITIALIZATION ENTRY POINT
;
        ORG  SENDEV
        JMP     SENDEN      ;SEND ENABLE ENTRY POINT
;
        ORG  VCTABL-INTABS+VSERIN
;
        DW      ISRSIR      ;VSERIN
        DW      ISRODN      ;VSEROR
        DW      ISRTD       ;VSEROC
;
;
;
        ORG  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     SOUNDR      ;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 FRAME RETRIES
        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 DEVICE TIME OUT VALUES IN Y,X
;
        LDA     #$00
        STA     ERRFLG      ;CLEAR ERROR FLAG
;
        JSR     WAITER      ;SET UP TIMER AND WAIT
        BEQ     DERR        ;BRANCH IF TIME OUT
;
;
; DEVICE DID NOT TIME OUT
;
        BIT     DSTATS
        BVS     MODATA      ;BRANCH IF MORE DATA FOLLOWS
;
        LDA     ERRFLG
        BNE     DERR1       ;BRANCH IF AN ERROR OCCURRED
        BEQ     RETURN      ;OTHERWISE RETURN
;
;
;
;
; RECEIVE A DATA FRAME FROM PERIPHERAL
;
MODATA: JSR     LDPNTR      ;LOAD BUFFER POINTER WITH DCB INFORMATION
;
        JSR     RECEIV      ;GO RECEIVE A DATA FRAME
;
DERR:   LDA     ERRFLG
        BEQ     NOTERR      ;BRANCH IF NO ERROR PRECEEDED DATA
;
        LDA     TSTAT       ;GET TEMP STATUS
        STA     STATUS      ;STORE IN REAL STATUS
;
;
NOTERR: LDA     STATUS
        CMP     #SUCCES
        BEQ     RETURN      ;BRANCH IF COMPLETELY SUCCESSFUL
;
DERR1:  DEC     DRETRY
        BMI     RETURN      ;BRANCH IF OUT OF DEVICE RETRIES
;
        JMP     COMMND      ;OTHERWISE, ONE MORE TIME
;
;
;
;
RETURN: JSR     SENDDS      ;DISABLE POKEY INTERRUPTS
        LDA     #0
        STA     CRITIC
        LDY     STATUS      ;RETURN STATUS IN Y
        STY     DSTATS      ;AND THE DCB STATUS WORD
        RTS     RETURN
;
;
;
;
; WAIT SUBROUTINE
;
; WAITS FOR COMPLETE OR ACK
; RETURNS Y=$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 NACK 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     NTBRK0
        JMP     BROKE       ;JUMP IF BREAK KEY PRESSED
;
NTBRK0: 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 BUFFER POINTER
        BNE     NOWRP0
        INC     BUFRHI
;
NOWRP0: 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     #$20
        BNE     NTOVRN      ;BRANCH IF NO OVERRUN ERROR
;
        LDY     #OVRRUN
        STY     STATUS      ;SET OVERRUN ERROR STATUS
;
NTOVRN: LDA     BUFRFL
        BEQ     NOTYET      ;BRANCH IF BUFFER WAS NOT YET FILLED
;
        LDA     SERIN       ;THIS INPUT BYTE IS THE CHECKSUM
        CMP     CHKSUM
        BEQ     SRETRN      ;BRANCH IF CHECKSUMS MATCH
;
        LDY     #CHKERR
        STY     STATUS      ;SET CHECKSUM ERROR STATUS
;
SRETRN: LDA     #$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
        BCC     SUSUAL      ;BRANCH IF NEW BUFFER ADDRESS IS IN BUFFER LIMITS
;
        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 BUFFER 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 IRG 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 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 IRG
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 DCB 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  =       HIGH JTIMER ;HI BYTE OF JUMP TIMER ROUTINE ADDR
JTADRL  =       LOW 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 INTERRUPT BITS
        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     SOUNDR      ;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 DEVICE 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     #$3F        ;MASK OFF 2 HI BITS
        TAX                 ;THIS IS HI BYTE OF TIME OUT
;
        TYA                 ;RESTORE
        ROR     A
        AND     #$C0        ;MASK OFF ALL BUT 2 HI BITS
        TAY                 ;THIS IS LO BYTE OF TIME OUT
;
        RTS
;
;
;
;
;
;
;
;
;
;
INTTBL: DW      ISRSIR      ;SERIAL INPUT READY
        DW      ISRODN      ;OUTPUT DATA NEEDED
        DW      ISRTD       ;TRANSMISSION DONE
;
SIRHI   =       HIGH ISRSIR ;SERIAL INPUT READY ISR ADDRESS
SIRLO   =       LOW ISRSIR
ODNHI   =       HIGH ISRODN ;OUTPUT DATA NEEDED ISR ADDRESS
ODNLO   =       LOW ISRODN
TDHI    =       HIGH ISRTD  ;TRANSMISSION DONE ISR ADDRESS
TDLO    =       LOW 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     #$100-$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 TABLE VALUE
; 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 TIMED OUT
        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 LINE COUNTER
        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 BAUD 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 USECDS  OLUTION) OF 'VCOUNT' FOR 1
;               CHARACTER TIME (10 BIT TIMES).
;
;
;
;
;               AUDF+7          BAUD RATE       VCOUNT INTERVAL
;               ------          -----------     ---------------
;       DW      $27C            ;1407                   56
;       DW      $2D7            ;1231                   64
;       DW      $332            ;1094                   72
;       DW      $38D            ;985                    80
POKTAB: DW      $3E8            ;895                    88
        DW      $443            ;820                    96
        DW      $49E            ;757                    104
        DW      $4F9            ;703                    112
        DW      $554            ;656                    120
        DW      $5AF            ;615                    128
        DW      $60A            ;579                    136
        DW      $665            ;547                    144
        DW      $6C0            ;518                    152
        DW      $71A            ;492                    160
        DW      $775            ;469                    168
        DW      $7D0            ;447                    176
;       DW      $82B            ;428                    184
;       DW      $886            ;410                    192
;       DW      $8E1            ;394                    200
;       DW      $93C            ;379                    208
;       DW      $997            ;365                    216
;       DW      $9F2            ;352                    224
;       DW      $A4D            ;339                    232
;       DW      $AA8            ;328                    240
;       DW      $B03            ;318                    248
;
;
```

```
;
;
;*****************************************************************
CRNTP3  =    *
        ORG  $14
SIOSPR: DB      DSKORG-CRNTP3 ;^GSIOL IS TOO LONG
        EJECT
```

```
        TITLE   'DISK  ***** DISKP.SRC ***** 3/9/79 *****  4:00:00 P.M.'
;
;
;
;
;
;
STATVH  =       HIGH DVSTAT
STATVL  =       LOW 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
;
        ORG  $E450
;
        JMP     DINIT       ;DISK INIT. VECTOR
        JMP     DSKIF       ;DISK INTERFACE ENTRY POINT
;
;
;
;
;
;       CONSTANTS
;
        ORG  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 I.D 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     #$80        ;SET BYTE COUNT TO 128
        LDA     DCOMND      ;READ COMMAND IN DCB
        CMP     #WRITE      ;IS COMMAND A "PUT SECTOR" COMMAND?
        BNE     CKSTC
        LDX     #PUTDAT     ;YES, SET "PUT DATA" COMMAND FOR SIO
CKSTC:  CMP     #STATC      ;IS COMMAND A STATUS COMMAND?
        BNE     PUTCNT
        LDA     #STATVL
        STA     DBUFLO
        LDA     #STATVH
        STA     DBUFHI      ;SET BUFFER ADDR TO GLOBAL STATUS BUFFER
        LDY     #4          ;YES, SET BYTE COUNT TO 4
PUTCNT: STX     DSTATS      ;PUT STATUS COMMAND FOR SIO IN DCB
        STY     DBYTLO
        LDA     #0
        STA     DBYTHI      ;PUT BYTE COUNT IN DCB
        JSR     SIOV        ;CALL SERIAL I/O.
        BPL     GOODST      ;NO ERROR
        RTS                 ;NO, GO BACK
GOODST: LDA     DCOMND      ;READ THE COMMAND
        CMP     #STATC      ;WAS IT A STATUS COMMAND?
        BNE     PUTBC
        JSR     PUTADR      ;PUT BUFFER ADDR IN TEMP REG.
        LDY     #2
        LDA     (BUFADR),Y  ;READ DISK TIMEOUT VALUE BYTE OF STATUS
        STA     DSKTIM      ;PUT IT IN DISK TIMEOUT REG.
PUTBC:  LDA     DCOMND
        CMP     #FOMAT      ;WAS COMMAND A FORMAT COMMAND?
        BNE     ENDDIF
FMTD:   JSR     PUTADR      ;YES, PUT BUFFER ADDR INTO TEMP REG
        LDY     #$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
;
;
;
;       S U B R O U T I N E S
;
```

```
;
;       PUT BUFFER ADDR FROM DCB INTO TEMP REG
;
PUTADR: LDA     DBUFLO
        STA     BUFADR
        LDA     DBUFHI
        STA     BUFADR+1    ;PUT BUFFER ADDR IN TEMP REG
        RTS
;*****************************************************************
;
;
;       SPARE BYTE OR MODULE TOO LONG FLAG
;
CRNTP4  =       *
;
        ORG  $14
DSKSPR: DB      PRNORG-CRNTP4 ;^GDISKP TOO LONG
;
        EJECT
```

```
        TITLE   'PRINTER  ***** PRINTP.SRC ***** 3/9/79 *****  4:00:00 P.M.'
;
;
;
;
;
;
;
;
;
;
;       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
;
;
;
;
        ORG  $E430
;
        DW      PHOPEN-1    ;PRINTER HANDLER OPEN
        DW      PHCLOS-1    ;PH CLOSE
        DW      BADST-1     ;PH READ
        DW      PHWRIT-1    ;PH WRITE
        DW      PHSTAT-1    ;PH STATUS
        DW      BADST-1     ;PH SPECIAL
        JMP     PHINIT      ;PH INIT.
        DB      0           ;ROM FILLER
;
;
;
;
;
        ORG  PRNORG
;
;
;
;
;       PRINTER HANDLER INITIALIZATION ROUTINE
;
PHINIT: LDA     #30
        STA     PTIMOT      ;SET UP INITIAL PRINTER TIMEOUT OF 30 SEC.
        RTS
;
;
;       PRINTER HANDLER CONSTANTS
;
PHSTLO: DW      DVSTAT      ;STATUS BUFFER POINTER
PHCHLO: DW      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      ;SET COMMAND TO "STATUS"
        STA     DCOMND      ;SET STATUS COMMAND
        STA     DAUX1
        JSR     SETDCB      ;GO SETUP DCB
        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 PRINT MODE
        LDX     PBPNT
        LDA     PTEMP       ;GET CHAR. SENT BY CIO
        STA     PRNBUF,X    ;PUT CHAR. IN PRINT BUFFER
        INX                 ;INCR. BUFFER POINTER
        CPX     PBUFSZ      ;BUFFER POINTER=BUFFER SIZE?
        BEQ     BUFFUL
        STX     PBPNT       ;SAVE BUFFER POINTER
        CMP     #CR         ;IS CHAR. = EOL ?
        BEQ     BLFILL      ;IF YES, GO DO BLANK FILL.
        LDY     #SUCCES     ;PUT GOOD STATUS IN Y REG 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 DCB
        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  =       *
;
        ORG  $14
;
PRNSPR: DB      CASORG-CRNTP5 ;^GPRINTP TOO LONG
;
        EJECT
```

```
        TITLE   'CASSET HANDLER 3/12 (DK1:CASCV)'
CBUFH   =       HIGH CASBUF
CBUFL   =       LOW 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     =       $FB             ;HEADER
TONE1   =       2               ;CHANGE TO RECORD MODE TONE
TONE2   =       1               ;PRESS PLAY TONE
;
;
;
        ORG  CASETV
        DW      OPENC-1,CLOSEC-1,GBYTE-1,PBYTE-1,STATU-1,SPECIAL-1
        JMP     INIT
        DB      0               ;ROM FILLER BYTE
;
;
;
; USED IN MONITP FOR CASSETTE BOOT
;
        ORG  RBLOKV
        JMP     RBLOK
;
        ORG  CSOPIV
        JMP     OPINP
;
;
        ORG  CASORG
;
;
; INIT ROUTINE
;
INIT:   LDA     #$CC
        STA     CBAUDL
        LDA     #$05
        STA     CBAUDH          ;SET CASSET BAUD RATE TO 600
SPECIAL:                        ;THATS ALL FOLKS
        RTS
        EJECT
```

```
;
; 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 VBLANK 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
        EJECT
```

```
;
; 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        ;BUMP 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
        EJECT
```

```
;
; 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.
        EJECT
```

```
;
; STATUS - RETURN STATUS INFO THRU DVSTAT
;
STATU:  LDY     #SUCCES
        RTS
        EJECT
```

```
;
; CLOSE
;
CLOSEC: LDA     WMODE       ;SEE IF WRITING
        BMI     CLWRT       ;GO CLOSE FOR WRITE
; CLOSE FOR READ - FLAG CLOSED
        LDY     #SUCCES     ;SUCCESSFULL
FCAX:   LDA     #MOTRST     ;STOP THE MOTOR IN CASE WAS SHORT IRG 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
        EJECT
```

```
;
; 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     #$FF
        STA     CONSOL      ;TURN ON SPEAKER
        LDA     #0
        LDY     #$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 GO BEEP AGIN
WFAK:   JSR     WFAK1       ;USE SIMULATED "JMP (KGETCH)"
        TYA
        RTS
WFAK1:  LDA     KEYBDV+5
        PHA
        LDA     KEYBDV+4    ;SIMULATE "JMP (KGETCH)"
        PHA
        RTS
;
; SIOSB - CALL SIO ON SYSTEM BUFFER
;
SIOSB:  STA     DCOMND      ;SAVE COMMAND
        LDA     #0
        STA     DBYTHI      ;SET BUFFER LENGTH
        LDA     #131
        STA     DBYTLO
        LDA     #CBUFH
        STA     DBUFHI      ;SET BUFFER ADDRESS
        LDA     #CBUFL
        STA     DBUFLO
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 BACK
        LDY     #SRSTA      ;SIO READ STATUS COMMAND
        CMP     #'R
        BEQ     *+4
        LDY     #SWSTA      ;SIO WRITE STATUS COMMAND
        STY     DSTATS      ;SET STATUS FOR SIO
        LDA     FTYPE
        STA     DAUX2       ;INDICATE IF SHORT IRG MODE
        JSR     SIOV        ;GO CALL SIO
        RTS
;
; WSIOSB - WRITE SIO SYSTEM BUFFER
;
WSIOSB: STA     CASBUF+2    ;STORE TYPE BYTE
        LDA     #$55
        STA     CASBUF+0
        STA     CASBUF+1
        LDA     #'W'        ;WRITE
        JSR     SIOSB       ;CALL SIO ON SYSTEM BUFFER
        RTS                 ;RETURN
CRNTP6  =    *
        ORG  $14
CASSPR: DB      MONORG-CRNTP6 ;^GCASCV IS TOO LONG
        EJECT
```

```
        TITLE   'MONITOR  ***** MONITP.SRC ***** 3/9/79 *****  4:00:00 P.M.'
;
;
;
;       CONSTANT EQUATES
;
PUTTXT  =       $9          ;"PUT TEXT RECORD" CIO COMMAND CODE
GETCAR  =       $7          ;"GET CHARACTER" CIO COMMAND CODE
PUTCAR  =       $B          ;"PUT CHARACTER" CIO COMMAND CODE
INIMLL  =       $00         ;INITIAL MEM LO LOW BYTE
INIMLH  =       $07         ;INITIAL MEM LO HIGH BYTE
; GOOD  =       $1          ;GOOD STATUS CODE
; WRITE =       $57         ;WRITE COMMAND
; READ  =       $52         ;READ COMMAND
; STATC =       $53         ;STATUS COMMAND
SEX     =       $0          ;SCREEN EDITOR IOCB INDEX
CLS     =       $7D         ;CLEAR SCREEN CODE
CTRLC   =       $92         ;KEYBOARD CODE FOR 'CONTROL C'
EOF     =       $88         ;CASSETTE END OF FILE CODE
LIRG    =       $0          ;LONG IRG TYPE CODE
;
BUFFH   =       HIGH [CASBUF+3]
BUFFL   =       LOW [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)
;
        ORG  $BFFA
CARTCS: DS      2           ;CARTRIDGE COLD START ADDRESS.
CART:   DS      1           ;CARTRIDGE AVAILABLE FLAG BYTE.
CARTFG: DS      1           ;CARTRIDGE FLAG BYTE. BIT 0=FLAG1,
CARTAD: DS      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
;
;****************************
;       ORG  $FFFC
;
; PVECT DW      PWRUP           POWER-UP VECTOR
;************************
;
;
;
;
;
;       ENTRY POINT VECTOR
;
        ORG  BLKBDV
;
        JMP     SIGNON      ;BLACK BOARD VECTOR
;
        ORG  WARMSV
;
        JMP     RESET       ;WARM START VECTOR
;
        ORG  COLDSV
;
        JMP     PWRUP       ;COLD START VECTOR (9000 FOR RAM VECTOR WRITING)
;
        ORG  $9000
        JSR     $900C
        JMP     PWRUP       ;(TO HANDLE RAM VECTOR WRITING)
        JSR     $900C
        JMP     RESET
;
;
;
        ORG  MONORG
;
;
;
;
;       HANDLER TABLE ENTRIES
;
TBLENT: DB      'P'
        DW      PRINTV
        DB      'C'
        DW      CASETV
        DB      'E'
        DW      EDITRV
        DB      'S'
        DW      SCRENV
        DB      'K'
        DW      KEYBDV
;
;
;TBLLEN  =       IDENT-TBLENT-1  HANDLER TABLE LENGTH.  "MOVED TO LINE 885"
;
;       ***** PRINT MESSAGES *****
;
;
IDENT:  DB      CLS,'ATARI COMPUTER - MEMO PAD',CR
;
IDENTH  =       HIGH IDENT
```

```
IDENTL  =       LOW IDENT ;SYSTEM I.D. MSG POINTER
;
TBLLEN  =       IDENT-TBLENT-1 ;HANDLER TABLE LENGTH
DERR5:  DB      'BOOT ERROR',CR
;
DERRH   =       HIGH DERR5
DERRL   =       LOW DERR5 ;DISK ERROR MSG POINTER
;
;
;
;
;       DEVICE/FILENAME SPECIFICATIONS
;
OPNEDT: DB      'E:',CR ;"OPEN SCREEN EDITOR" DEVICE SPEC.
;
OPNH    =       HIGH OPNEDT
OPNL    =    LOW 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     #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                 ;SCREEN OK, SO WAIT FOR VBLANK TO
        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 RUN,
; 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)
        EJECT
```

```
;
;
;
;
;
;               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 RAM DATA
        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      ;WARM START?
        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 UP STATUS COMMAND
        JSR     DSKINV      ;GO DO DISK STATUS
        BPL     DOBOOT      ;IS STATUS FROM SIO 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       ;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     ICBLL,X     ;SET BUFFER LENGTH
        JSR     CIOV        ;PUT LINE ON SCREEN
        RTS
;
;
;
;
; GET SECTOR FROM DISK 0
;
GETSEC: LDA     CASSBT
        BEQ     DISKM       ;CASSETTE BOOT ?
        JMP     RBLOKV      ;YES, GO TO READ BLOCK ROUTINE
DISKM:  LDA     #READ
        STA     DCOMND      ;SET READ SECTOR COMMAND
        LDA     #1
        STA     DUNIT       ;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 IRG 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  =    *
;
        ORG  $14
MONSPR: DB      KBDORG-CRNTP7 ;^GMONITP TOO LONG
;
        EJECT
```

```
        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
;
        EJECT
```

```
;
;
        ORG   EDITRV
;
; SCREEN EDITOR HANDLER ENTRY POINT
;
EDITOR: DW      EOPEN-1
        DW      RETUR1-1    ;(CLOSE)
        DW      EGETCH-1
        DW      EOUTCH-1
        DW      RETUR1-1    ;(STATUS)
        DW      NOFUNC-1    ;(SPECIAL)
        JMP     PWRONA
        DB      0           ;ROM FILLER BYTE
;
        ORG   SCRENV
;
; DISPLAY HANDLER ENTRY POINT
;
DISPLA: DW      DOPEN-1
        DW      RETUR1-1    ;(CLOSE)
        DW      GETCH-1
        DW      OUTCH-1
        DW      RETUR1-1    ;(STATUS)
        DW      DRAW-1      ;(SPECIAL)
        JMP     PWRONA
        DB      0           ;ROM FILLER BYTE
;
        ORG   KEYBDV
;
;
; KEYBOARD HANDLER ENTRY POINT
;
KBDHND: DW      RETUR1-1
        DW      RETUR1-1    ;(CLOSE)
        DW      KGETCH-1
        DW      NOFUNC-1    ;(OUTCH)
        DW      RETUR1-1    ;(STATUS)
        DW      NOFUNC-1    ;(SPECIAL)
        JMP     PWRONA
        DB      0           ;ROM FILLER BYTE
;
;
; INTERRUPT VECTOR TABLE ENTRY
        ORG   VCTABL-INTABS+VKEYBD
        DW      PIRQ5       ;KEYBOARD IRQ INTERRUPT VECTOR
        EJECT
```

```
        ORG  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
        EJECT
```

```
;
;
; 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 AT OPEN
        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      ;CLEAR 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 EDITOR
        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     (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 IS NON-0
        BNE     OUTPLT
        LDX     #2
CRLOOP: LDA     ROWCRS,X    ;SAVE CURSOR LOCATION FOR DRAW LINE TO DRAW FROM
        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 ATAINT
        TYA                 ;RESTORE ATACHR
        AND     #$9F        ;STRIP OFF COLUMN ADDRESS
        ORA     ATAINT,X    ;OR IN NEW COLUMN ADDRESS
OUTCH2: STA     CHAR
        JSR     CONVRT
        LDA     CHAR
SHIFTU: LSR     SHFAMT      ;SHIFT UP TO PROPER POSITION
        BCS     SHIFT2
        ASL     A
        JMP     SHIFTU
SHIFT2: AND     DMASK
        STA     TMPCHR      ;SAVE SHIFTED DATA
        LDA     DMASK       ;INVERT MASK
        EOR     #$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     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
;
        EJECT
```

```
;
;
;
;
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, SO 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 EMPTIED
        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)    ;JSR 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 ESCFLG
        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     #$FF
        STA     CH
KGETCH: LDA     ICAX1Z      ;TEST LSB OF AUX1 FOR SPECIAL EDITOR READ MODE
        LSR     A
        BCS     GETOUT
        LDA     #BRKABT
        LDX     BRKKEY      ;TEST BREAK
        BEQ     K7          ;IF BREAK, PUT BRKABT IN DSTAT AND CR IN ATACHR
        LDA     CH
        CMP     #$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 MSB OF CONTROL CHARACTERS
        BEQ     K4
        LDA     ATACHR
        EOR     INVFLG
K8:     STA     ATACHR
K4:     JMP     RETUR1      ;ALL DONE
;
;
        EJECT
```

```
;
;
; CONTROL CHARACTER PROCESSORS
;
ESCAPE: LDA     #$80        ;SET ESCAPE FLAG
        STA     ESCFLG
        RTS
CRSRUP: DEC     ROWCRS
        BPL     COMRET
        LDX     BOTSCR      ;WRAPAROUND
        DEX
UPDNCM: STX     ROWCRS
COMRET: JMP     STRBEG      ;COLVERT ROW AND COL TO LOGCOL AND RETURN
CRSRDN: INC     ROWCRS
        LDA     ROWCRS
        CMP     BOTSCR
        BCC     COMRET
        LDX     #0
        BEQ     UPDNCM      ;(UNCONDITIONAL)
CRSRLF: DEC     COLCRS
        LDA     COLCRS
        BMI     CRSRL1      ;(IF LMARGN=0, THIS ELIMINATES PROBLEM CASE)
        CMP     LMARGN
        BCS     COMRE1
CRSRL1: LDA     RMARGN
LFRTCM: STA     COLCRS
COMRE1: JMP     DOLCOL      ;COLVERT ROW AND COL TO LOGCOL AND RETURN
CRSRRT: INC     COLCRS
        LDA     COLCRS
        CMP     RMARGN
        BCC     COMRE1
        BEQ     COMRE1      ;(CAUSE BLE)
        LDA     LMARGN
        JMP     LFRTCM      ;UNCONDITIONAL TO COMMON STORE
CLRSCR: JSR     PUTMSC
        LDY     #0
        TYA                 ;PUT 0 IN THE AC
CLRSC2: STA     (ADRESS),Y  ;(AC IS ZERO)
        INY
        BNE     CLRSC2
        INC     ADRESS+1
        LDX     ADRESS+1
        CPX     RAMTOP
        BCC     CLRSC2
        LDA     #$FF        ;CLEAN UP LOGICAL LINE BIT MAP
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      ;COLVERT 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      ;COLVERT 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                 ;NORMAL 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 T0=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
EJECT
```

```
;
;
; 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
STORE1: 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    ;MAKE 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 FINISH
         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 BYTES
         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 MLTTMP
         STA     MLTTMP
         BCC     CONVR5
         INC     MLTTMP+1
CONVR5:  SEC                ;* TRICKY *
CONVR6:  ROR     TMPLBT     ;SLIDE A "1" UP AGAINST LOW BITS (CONTINUE TILL X=-1)
         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     ADRESS
         STA     ADRESS
         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     INCRSC
INCRSR:  LDA     #$9B       ;SPECIAL CASE ELIMINATOR
INCRSC:  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 SO DO IT
        LDA     COLCRS+1    ;TEST MSD
        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 LOG LINE
        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 BACK 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 LOGCOL 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 MODE AND OK
        LDA     DINDEX      ;IF MODE = 0
        BEQ     RANGE       ;THEN IT IS IN EDITOR 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 39
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 LOW BYTE
RANGE1: LDA     COLCRS+1    ;FOR OTHERS, NON-ZERO HIGH BYTE IS BAD
        BNE     RNGERR
RANGE2: LDA     COLUMN,X    ;CHECK LOW BYTE
        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 CIO
        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 LEVEL
```

```
        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
;
;
;
; MOVLIN: MOVE 40 BYTES AT FRMADR TO TOADR SAVING OLD TOADR
;       DATA IN THE LINBUF. THEN MAKE NEXT FRMADR
;       BE AT LINBUF FOR NEXT TRANSFER & TOADR=TOADR+40
;
MOVLIN: LDA     #HIGH LINBUF;SET UP ADRESS=LINBUF=$247
        STA     ADRESS+1
        LDA     #LOW LINBUF
        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     MOVLI1
        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
;
;
;
; DOBUFC: COMPUTE BUFFER COUNT AS THE NUMBER OF BYTES FROM
;         BUFSTR TO END OF LOGICAL LINE WITH TRAILING SPACES REMOVED
;
DOBUFC: JSR     PHACRS
        LDA     LOGCOL
        PHA
        LDA     BUFSTR      ;START
        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 CRSR IS AT LAST SCREEN POSITION
        BNE     DOBU1A
        LDA     COLCRS
        CMP     RMARGN
        BNE     DOBU1A
        INC     BUFCNT      ;YES, SO FAKE INCRSR 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     DELTIB      ;LINE BEFORE TESTING DELTIM
        DEC     ROWCRS
DELTIB: JSR     DOLCOL
DELTIM: LDA     LOGCOL      ;TEST FOR EXTENSION
        CMP     LMARGN
        BEQ     DELTI3      ;NO
        JSR     CONVRT
        LDA     RMARGN      ;SET UP COUNT
        SEC
        SBC     LMARGN
        TAY
DELTI1: LDA     (ADRESS),Y
        BNE     DELTI3      ;FOUND A NON-0 SO QUIT AND RETURN
        DEY
        BPL     DELTI1
DELTI2: JMP     DELLIB      ;DELETE A LINE AND RETURN
DELTI3: RTS
;
;
;
; TSTCTL: SEARCH CNTRLS TABLE TO SEE IF ATACHR IS A CNTL CHAR
;
TSTCTL: LDX     #45         ;PREPARE TO SEARCH TABLE
TSTCT1: LDA     CNTRLS,X
        CMP     ATACHR
        BEQ     TSTCT2
        DEX
        DEX
        DEX
        BPL     TSTCT1
TSTCT2: RTS
;
;
;
; PUSH ROWCRS,COLCRS AND COLCRS+1
;
PHACRS: LDX     #2
PHACR1: LDA     ROWCRS,X
        STA     TMPROW,X
        DEX
        BPL     PHACR1
```

```
        RTS
;
;
; PULL COLCRS+1,COLCRS AND ROWCRS
;
PLACRS: LDX     #2
PLACR1: LDA     TMPROW,X
        STA     ROWCRS,X
        DEX
        BPL     PLACR1
        RTS
;
;
;
; SWAP: IF MIXED MODE, SWAP TEXT CURSORS WITH REGULAR CURSORS
;
SWAP:   JSR     SWAPA       ;THIS ENTRY POINT DOES RETUR1
        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 LMARGN 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      ;SET UP ADDRESS
        STA     ADRESS
        LDA     SAVMSC+1
        STA     ADRESS+1
        RTS
```

```
;
        EJECT
```

```
;
;
; 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     DRAW11      ;IF COUNTER IS ZERO, LEAVE DRAW
        JMP     DRAW10
DRAW11: 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 BLT
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
        EJECT
```

```
;
;
; TABLES
;
;
; MEMORY ALLOCATION
;
ALOCAT: DB      24,16,10,10,16,28,52,100,196,196,196,196
;
;
; NUMBER OF DISPLAY LIST ENTRIES
;
NUMDLE: DB      23,23,11,23,47,47,95,95,97,97,97,97
MXDMDE: DB      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      ""
;       6               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: DB      2,6,7,8,9,$A,$B,$D,$F,$F,$F,$F ;ZEROS FOR RANGE TEST IN PAGETB
;
;
; PAGE TABLE TELLS WHICH DISPLAY LISTS ARE IN DANGER OF
; CROSSING A 256 BYTE PAGE BOUNDARY
;
PAGETB: DB      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: DB      2,1,1,0,0,1,1,2,2,2,2,2
;
;
; COLUMN: NUMBER OF COLUMNS
;
COLUMN: DB      40,20,20,40,80,80,160,160,64,80,80,80 ;MODE 8 IS SPECIAL CASE
;
;
;
; NOROWS: NUMBER OF ROWS
;
NOROWS: DB      24,24,12,24,48,48,96,96,192,192,192,192
;
;
;
;
; DIV2TB: HOW MANY RIGHT SHIFTS FOR HCRSR FOR PARTIAL BYTE MODES
;
DIV2TB: DB      0,0,0,2,3,2,3,2,3,1,1,1
;
;
```

```
; DMASKT: DISPLAY MASK TABLE
;
DMASKT: DB      $00,$FF,$F0,$0F
        DB      $C0,$30,$0C,$03
;
; MASKTB: BIT MASK.   (ALSO PART OF DMASKTB! DO NOT SEPARATE)
;
MASKTB: DB      $80,$40,$20,$10,$08,$04,$02,$01
;
;
;
;
COLRTB: DB      $28,$CA,$94,$46,$00
;
;
;
;
; CNTRLS: CONTROL CODES AND THEIR DISPLACEMENTS INTO THE
;         CONTROL CHARACTER PROCESSORS
;
CNTRLS: DB      $1B
        DW      ESCAPE
        DB      $1C
        DW      CRSRUP
        DB      $1D
        DW      CRSRDN
        DB      $1E
        DW      CRSRLF
        DB      $1F
        DW      CRSRRT
        DB      $7D
        DW      CLRSCR
        DB      $7E
        DW      BS
        DB      $7F
        DW      TAB
        DB      $9B
        DW      DOCRWS
        DB      $9C
        DW      DELLIN
        DB      $9D
        DW      INSLIN
        DB      $9E
        DW      CLRTAB
        DB      $9F
        DW      SETTAB
        DB      $FD
        DW      BELL
        DB      $FE
        DW      DELCHR
        DB      $FF
        DW      INSCHR
;
;
;
;
; ATAINT: ATASCI TO INTERNAL TABLE
;
ATAINT: DB      $40,$00,$20,$60
;
;
; INTATA: INTERNAL TO ATASCI TABLE
;
INTATA: DB      $20,$40,$00,$60
```

```
;
;
; ATASCI: ATASCII CONVERSION TABLE
;
ATASCI: DB      $6C,$6A,$3B,$80,$80,$6B,$2B,$2A ;LOWER CASE
        DB      $6F,$80,$70,$75,$9B,$69,$2D,$3D

        DB      $76,$80,$63,$80,$80,$62,$78,$7A
        DB      $34,$80,$33,$36,$1B,$35,$32,$31

        DB      $2C,$20,$2E,$6E,$80,$6D,$2F,$81
        DB      $72,$80,$65,$79,$7F,$74,$77,$71

        DB      $39,$80,$30,$37,$7E,$38,$3C,$3E
        DB      $66,$68,$64,$80,$82,$67,$73,$61


        DB      $4C,$4A,$3A,$80,$80,$4B,$5C,$5E ;UPPER CASE
        DB      $4F,$80,$50,$55,$9B,$49,$5F,$7C

        DB      $56,$80,$43,$80,$80,$42,$58,$5A
        DB      $24,$80,$23,$26,$1B,$25,$22,$21

        DB      $5B,$20,$5D,$4E,$80,$4D,$3F,$81
        DB      $52,$80,$45,$59,$9F,$54,$57,$51

        DB      $28,$80,$29,$27,$9C,$40,$7D,$9D
        DB      $46,$48,$44,$80,$83,$47,$53,$41


        DB      $0C,$0A,$7B,$80,$80,$0B,$1E,$1F ;CONTROL
        DB      $0F,$80,$10,$15,$9B,$09,$1C,$1D

        DB      $16,$80,$03,$80,$80,$02,$18,$1A
        DB      $80,$80,$85,$80,$1B,$80,$FD,$80

        DB      $00,$20,$60,$0E,$80,$0D,$80,$81
        DB      $12,$80,$05,$19,$9E,$14,$17,$11

        DB      $80,$80,$80,$80,$FE,$80,$7D,$FF
        DB      $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
;
;
        DB      $FF,$FF,$FF,$FF,$FF,$FF
;
CRNTPC  =    *
        ORG  $14
KBDSPR: DB      $FFF8-CRNTPC ;^GDISPLC IS TOO LONG


;*********************************************************************
; ADDITIONS TO REV B TO GENERATE EXACT ROM IMAGE
; BY M.B. & V.W. 8/21/84
;

      LIST   I

      INCLUDE      D:CHARSET.ASM


;****
; HOLES WITH RANDOM DATA BYTES

      ORG    $EDE8
      DB     $24,$85

      ORG    $E90B
      DB     $66,$66,$7E,$66,$00,$00,$7C,$4C,$ED,$E8,$66,$7C,$00
      DB     $00,$3C,$66,$60,$60,$66,$3C,$00,$00,$78,$6C,$66,$66
      DB     $6C,$78,$00,$00,$7E,$60,$7C,$60,$60,$7E,$00,$00,$7E
      DB     $60,$7C,$60,$60,$60,$00,$00,$3E,$60,$60,$6E,$66,$3E
      DB     $00,$00,$66,$66,$7E


;****
;

      ORG    $FFF8  ;CHECKSUM & HARDWARE VECTORS
      DB     $F3,$E6,$91,$E7,$25,$F1,$F3,$E6
        END
```

This page intentionally left blank

OS, Operating System, Revision B, NTSC, EPROM version, for 400/800, for Data General cross assembler [REVBAMAC.PRN]

```
               ATARI CAMAC Assembler Ver  1.0A  Page   1
CENTRAL INPUT/OUTPUT (CIO) 2-7-79                  D1:REVBAMAC.AS


                    ;LIST X
                    ; THIS IS THE MODIFIED SEPTEMBER ATARI 400/800 COMPUTER OPERATING
                    ; SYSTEM LISTING, MODIFIED TO ASSEMBLE ON THE DATA GENERAL CROSS
                    ; ASSEMBLER.
                    ; 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
```

```
                ;
                ;
                ;       COLLEEN OPERATING SYSTEM EQUATE FILE
                ;
                ;       NTSC/PAL ASSEMBLY FLAG
                ;
= 0000  PALFLG  =       0           ;0 = NTSC   1 = PAL
                ;
                ;
                ;       MODULE ORIGIN TABLE
                ;
= E000  CHRORG  =       $E000       ;CHARACTER SET
= E400  VECTBL  =       $E400       ;VECTOR TABLE
= E480  VCTABL  =       $E480       ;RAM VECTOR INITIAL VALUE TABLE
= E4A6  CIOORG  =       $E4A6       ;CENTRAL I/O HANDLER
= E6D5  INTORG  =       $E6D5       ;INTERRUPT HANDLER
= E944  SIOORG  =       $E944       ;SERIAL I/O DRIVER
= EDEA  DSKORG  =       $EDEA       ;DISK HANDLER
= EE78  PRNORG  =       $EE78       ;PRINTER HANDLER
= EF41  CASORG  =       $EF41       ;CASSETTE HANDLER
= F0E3  MONORG  =       $F0E3       ;MONITOR/POWER UP MODULE
= F3E4  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
                ;
                ;
= E400  EDITRV  =       $E400       ;EDITOR
= E410  SCRENV  =       $E410       ;TELEVISION SCREEN
= E420  KEYBDV  =       $E420       ;KEYBOARD
= E430  PRINTV  =       $E430       ;PRINTER
= E440  CASETV  =       $E440       ;CASSETTE
                ;
                ;       JUMP VECTOR TABLE
                ;
                ;THE FOLLOWING IS A TABLE OF JUMP INSTRUCTIONS
                ;TO VARIOUS ENTRY POINTS IN THE OPERATING SYSTEM.
                ;
= E450  DISKIV  =       $E450       ;DISK INITIALIZATION
= E453  DSKINV  =       $E453       ;DISK INTERFACE
```

```
= E456      CIOV    =       $E456       ;CENTRAL INPUT OUTPUT ROUTINE
= E459      SIOV    =       $E459       ;SERIAL INPUT OUTPUT ROUTINE
= E45C      SETVBV  =       $E45C       ;SET SYSTEM TIMERS ROUTINE
= E45F      SYSVBV  =       $E45F       ;SYSTEM VERTICAL BLANK CALCULATIONS
= E462      XITVBV  =       $E462       ;EXIT VERTICAL BLANK CALCULATIONS
= E465      SIOINV  =       $E465       ;SERIAL INPUT OUTPUT INITIALIZATION
= E468      SENDEV  =       $E468       ;SEND ENABLE ROUTINE
= E46B      INTINV  =       $E46B       ;INTERRUPT HANDLER INITIALIZATION
= E46E      CIOINV  =       $E46E       ;CENTRAL INPUT OUTPUT INITIALIZATION
= E471      BLKBDV  =       $E471       ;BLACKBOARD MODE
= E474      WARMSV  =       $E474       ;WARM START ENTRY POINT
= E477      COLDSV  =       $E477       ;COLD START ENTRY POINT
= E47A      RBLOKV  =       $E47A       ;CASSETTE READ BLOCK ENTRY POINT VECTOR
= E47D      CSOPIV  =       $E47D       ;CASSETTE OPEN FOR INPUT VECTOR
                ;VCTABL = $E480
                ;
                ;
                ; OPERATING SYSTEM EQUATES
                ;
                ; COMMAND CODES FOR IOCB
= 0003      OPEN    =       3           ;OPEN FOR INPUT/OUTPUT
= 0005      GETREC  =       5           ;GET RECORD (TEXT)
= 0007      GETCHR  =       7           ;GET CHARACTER(S)
= 0009      PUTREC  =       9           ;PUT RECORD (TEXT)
= 000B      PUTCHR  =       $B          ;PUT CHARACTER(S)
= 000C      CLOSE   =       $C          ;CLOSE DEVICE
= 000D      STATIS  =       $D          ;STATUS REQUEST
= 000E      SPECIL  =       $E          ;BEGINNING OF SPECIAL ENTRY COMMANDS
                ;
                ; SPECIAL ENTRY COMMANDS
= 0011      DRAWLN  =       $11         ;DRAW LINE
= 0012      FILLIN  =       $12         ;DRAW LINE WITH RIGHT FILL
= 0020      RENAME  =       $20         ;RENAME DISK FILE
= 0021      DELETE  =       $21         ;DELETE DISK FILE
= 0022      FORMAT  =       $22         ;FORMAT
= 0023      LOCKFL  =       $23         ;LOCK FILE TO READ ONLY
= 0024      UNLOCK  =       $24         ;UNLOCK LOCKED FILE
= 0025      POINT   =       $25         ;POINT SECTOR
= 0026      NOTE    =       $26         ;NOTE SECTOR
= 00FF      IOCFRE  =       $FF         ;IOCB "FREE"
                ;
                ; AUX1 EQUATES
                ; () INDICATES WHICH DEVICES USE BIT
= 0001      APPEND  =       $1          ;OPEN FOR WRITE APPEND (D), OR SCREEN READ (E)
= 0002      DIRECT  =       $2          ;OPEN FOR DIRECTORY ACCESS (D)
= 0004      OPNIN   =       $4          ;OPEN FOR INPUT (ALL DEVICES)
= 0008      OPNOT   =       $8          ;OPEN FOR OUTPUT (ALL DEVICES)
= 000C      OPNINO  =       OPNIN+OPNOT ;OPEN FOR INPUT AND OUTPUT (ALL DEVICES)
= 0010      MXDMOD  =       $10         ;OPEN FOR MIXED MODE (E,S)
= 0020      INSCLR  =       $20         ;OPEN WITHOUT CLEARING SCREEN (E,S)
                ;
                ; DEVICE NAMES
= 0000      SCREDT  =       'E          ;SCREEN EDITOR (R/W)
= 0000      KBD     =       'K          ;KEYBOARD (R ONLY)
= 0000      DISPLY  =       'S          ;SCREEN DISPLAY (R/W)
= 0000      PRINTR  =       'P          ;PRINTER (W ONLY)
= 0000      CASSET  =       'C          ;CASSETTE
```

```
= 0000      MODEM   =       'M          ;MODEM
= 0000      DISK    =       'D          ;DISK (R/W)
                    ;
                    ; SYSTEM EOL (CARRIAGE RETURN)
        = 009B      CR      =       $9B
                    ;
                    ;
                    ;       OPERATING SYSTEM STATUS CODES
                    ;
        = 0001      SUCCES  =       $01         ;SUCCESSFUL OPERATION
                    ;
        = 0080      BRKABT  =       $80         ;BREAK KEY ABORT
        = 0081      PRVOPN  =       $81         ;IOCB ALREADY OPEN
        = 0082      NONDEV  =       $82         ;NON-EXISTANT DEVICE
        = 0083      WRONLY  =       $83         ;IOCB OPENED FOR WRITE ONLY
        = 0084      NVALID  =       $84         ;INVALID COMMAND
        = 0085      NOTOPN  =       $85         ;DEVICE OR FILE NOT OPEN
        = 0086      BADIOC  =       $86         ;INVALID IOCB NUMBER
        = 0087      RDONLY  =       $87         ;IOCB OPENED FOR READ ONLY
        = 0088      EOFERR  =       $88         ;END OF FILE
        = 0089      TRNRCD  =       $89         ;TRUNCATED RECORD
        = 008A      TIMOUT  =       $8A         ;PERIPHERAL DEVICE TIME OUT
        = 008B      DNACK   =       $8B         ;DEVICE DOES NOT ACKNOWLEDGE COMMAND
        = 008C      FRMERR  =       $8C         ;SERIAL BUS FRAMING ERROR
        = 008D      CRSROR  =       $8D         ;CURSOR OVERRANGE
        = 008E      OVRRUN  =       $8E         ;SERIAL BUS DATA OVERRUN
        = 008F      CHKERR  =       $8F         ;SERIAL BUS CHECKSUM ERROR
                    ;
        = 0090      DERROR  =       $90         ;PERIPHERAL DEVICE ERROR (OPERATION NOT COMPLETED)
        = 0091      BADMOD  =       $91         ;BAD SCREEN MODE NUMBER
        = 0092      FNCNOT  =       $92         ;FUNCTION NOT IMPLEMENTED IN HANDLER
        = 0093      SCRMEM  =       $93         ;INSUFICIENT MEMORY FOR SCREEN MODE
                    ;
                    ;
                    ;
                    ;
                    ;
                    ;
                    ;       PAGE ZERO RAM ASSIGNMENTS
                    ;
  0000  = 0000              ORG     $0000
  0000  = 0002      LINZBS: DS      2           ;LINBUG RAM (WILL BE REPLACED BY MONITOR RAM)
                    ;
                    ; THESE LOCATIONS ARE NOT CLEARED
  0002  = 0002      CASINI: DS      2           ;CASSETTE INIT LOCATION
  0004  = 0002      RAMLO:  DS      2           ;RAM POINTER FOR MEMORY TEST
  0006  = 0001      TRAMSZ: DS      1           ;TEMPORARY REGISTER FOR RAM SIZE
  0007  = 0001      TSTDAT: DS      1           ;RAM TEST DATA REGISTER
                    ;
                    ; CLEARED ON COLDSTART ONLY
  0008  = 0001      WARMST: DS      1           ;WARM START FLAG
  0009  = 0001      BOOT?:  DS      1           ;SUCCESSFUL BOOT FLAG
  000A  = 0002      DOSVEC: DS      2           ;DISK SOFTWARE START VECTOR
  000C  = 0002      DOSINI: DS      2           ;DISK SOFTWARE INIT ADDRESS
  000E  = 0002      APPMHI: DS      2           ;APPLICATIONS MEMORY HI LIMIT
                    ;
                    ; CLEARED ON COLD OR WARM START
```

```
      = 0010      INTZBS  =       *           ;INTERRUPT HANDLER
0010  = 0001      POKMSK: DS      1           ;SYSTEM MASK FOR POKEY IRQ ENABLE
0011  = 0001      BRKKEY: DS      1           ;BREAK KEY FLAG
0012  = 0003      RTCLOK: DS      3           ;REAL TIME CLOCK (IN 16 MSEC UNITS)
                  ;
0015  = 0002      BUFADR: DS      2           ;INDIRECT BUFFER ADDRESS REGISTER
                  ;
0017  = 0001      ICCOMT: DS      1           ;COMMAND FOR VECTOR
                  ;
0018  = 0002      DSKFMS: DS      2           ;DISK FILE MANAGER POINTER
001A  = 0002      DSKUTL: DS      2           ;DISK UTILITIES POINTER
                  ;
001C  = 0001      PTIMOT: DS      1           ;PRINTER TIME OUT REGISTER
001D  = 0001      PBPNT:  DS      1           ;PRINT BUFFER POINTER
001E  = 0001      PBUFSZ: DS      1           ;PRINT BUFFER SIZE
001F  = 0001      PTEMP:  DS      1           ;TEMPORARY REGISTER
                  ;
      = 0020      ZIOCB   =       *           ;ZERO PAGE I/O CONTROL BLOCK
      = 0010      IOCBSZ  =       16          ;NUMBER OF BYTES PER IOCB
      = 0080      MAXIOC  =       8*IOCBSZ    ;LENGTH OF THE IOCB AREA
      = 0020      IOCBAS  =       *
0020  = 0001      ICHIDZ: DS      1           ;HANDLER INDEX NUMBER (FF = IOCB FREE)
0021  = 0001      ICDNOZ: DS      1           ;DEVICE NUMBER (DRIVE NUMBER)
0022  = 0001      ICCOMZ: DS      1           ;COMMAND CODE
0023  = 0001      ICSTAZ: DS      1           ;STATUS OF LAST IOCB ACTION
0024  = 0001      ICBALZ: DS      1           ;BUFFER ADDRESS LOW BYTE
0025  = 0001      ICBAHZ: DS      1
0026  = 0001      ICPTLZ: DS      1           ;PUT BYTE ROUTINE ADDRESS - 1
0027  = 0001      ICPTHZ: DS      1
0028  = 0001      ICBLLZ: DS      1           ;BUFFER LENGTH LOW BYTE
0029  = 0001      ICBLHZ: DS      1
002A  = 0001      ICAX1Z: DS      1           ;AUXILIARY INFORMATION FIRST BYTE
002B  = 0001      ICAX2Z: DS      1
002C  = 0004      ICSPRZ: DS      4           ;TWO SPARE BYTES (CIO LOCAL USE)
      = 002E      ICIDNO  =       ICSPRZ+2    ;IOCB NUMBER X 16
      = 002F      CIOCHR  =       ICSPRZ+3    ;CHARACTER BYTE FOR CURRENT OPERATION
                  ;
0030  = 0001      STATUS: DS      1           ;INTERNAL STATUS STORAGE
0031  = 0001      CHKSUM: DS      1           ;CHECKSUM (SINGLE BYTE SUM WITH CARRY)
0032  = 0001      BUFRLO: DS      1           ;POINTER TO DATA BUFFER (LO BYTE)
0033  = 0001      BUFRHI: DS      1           ;POINTER TO DATA BUFFER (HI BYTE)
0034  = 0001      BFENLO: DS      1           ;NEXT BYTE PAST END OF THE DATA BUFFER (LO BYTE)
0035  = 0001      BFENHI: DS      1           ;NEXT BYTE PAST END OF THE DATA BUFFER (HI BYTE)
0036  = 0001      CRETRY: DS      1           ;NUMBER OF COMMAND FRAME RETRIES
0037  = 0001      DRETRY: DS      1           ;NUMBER OF DEVICE RETRIES
0038  = 0001      BUFRFL: DS      1           ;DATA BUFFER FULL FLAG
0039  = 0001      RECVDN: DS      1           ;RECEIVE DONE FLAG
003A  = 0001      XMTDON: DS      1           ;TRANSMISSION DONE FLAG
003B  = 0001      CHKSNT: DS      1           ;CHECKSUM SENT FLAG
003C  = 0001      NOCKSM: DS      1           ;NO CHECKSUM FOLLOWS DATA FLAG
                  ;
                  ;
003D  = 0001      BPTR:   DS      1
003E  = 0001      FTYPE:  DS      1
003F  = 0001      FEOF:   DS      1
0040  = 0001      FREQ:   DS      1
0041  = 0001      SOUNDR: DS      1           ;NOISY I/O FLAG. (ZERO IS QUIET)
```

```
0042  = 0001      CRITIC: DS      1           ;DEFINES CRITICAL SECTION (CRITICAL IF NON-ZERO)
                  ;
0043  = 0007      FMSZPG: DS      7           ;DISK FILE MANAGER SYSTEM ZERO PAGE
                  ;
                  ;
004A  = 0001      CKEY:   DS      1           ;FLAG SET WHEN GAME START PRESSED
004B  = 0001      CASSBT: DS      1           ;CASSETTE BOOT FLAG
004C  = 0001      DSTAT:  DS      1           ;DISPLAY STATUS
                  ;
004D  = 0001      ATRACT: DS      1           ;ATRACT FLAG
004E  = 0001      DRKMSK: DS      1           ;DARK ATRACT MASK
004F  = 0001      COLRSH: DS      1           ;ATRACT COLOR SHIFTER (EOR'ED WITH PLAYFIELD COLORS)
                  ;
      = 0002      LEDGE   =       2           ;LMARGN'S VALUE AT COLD START
      = 0027      REDGE   =       39          ;RMARGN'S VALUE AT COLD START
0050  = 0001      TMPCHR: DS      1
0051  = 0001      HOLD1:  DS      1
0052  = 0001      LMARGN: DS      1           ;LEFT MARGIN (SET TO 1 AT POWER ON)
0053  = 0001      RMARGN: DS      1           ;RIGHT MARGIN (SET TO 38 AT POWER ON)
0054  = 0001      ROWCRS: DS      1           ;CURSOR COUNTERS
0055  = 0002      COLCRS: DS      2
0057  = 0001      DINDEX: DS      1
0058  = 0002      SAVMSC: DS      2
005A  = 0001      OLDROW: DS      1
005B  = 0002      OLDCOL: DS      2
005D  = 0001      OLDCHR: DS      1           ;DATA UNDER CURSOR
005E  = 0002      OLDADR: DS      2
0060  = 0001      NEWROW: DS      1           ;POINT DRAW GOES TO
0061  = 0002      NEWCOL: DS      2
0063  = 0001      LOGCOL: DS      1           ;POINTS AT COLUMN IN LOGICAL LINE
0064  = 0002      ADRESS: DS      2
0066  = 0002      MLTTMP: DS      2
      = 0066      OPNTMP  =       MLTTMP      ;FIRST BYTE IS USED IN OPEN AS TEMP
0068  = 0002      SAVADR: DS      2
006A  = 0001      RAMTOP: DS      1           ;RAM SIZE DEFINED BY POWER ON LOGIC
006B  = 0001      BUFCNT: DS      1           ;BUFFER COUNT
006C  = 0002      BUFSTR: DS      2           ;EDITOR GETCH POINTER
006E  = 0001      BITMSK: DS      1           ;BIT MASK
006F  = 0001      SHFAMT: DS      1
0070  = 0002      ROWAC:  DS      2
0072  = 0002      COLAC:  DS      2
0074  = 0002      ENDPT:  DS      2
0076  = 0001      DELTAR: DS      1
0077  = 0002      DELTAC: DS      2
0079  = 0001      ROWINC: DS      1
007A  = 0001      COLINC: DS      1
007B  = 0001      SWPFLG: DS      1           ;NON-0 IF TXT AND REGULAR RAM IS SWAPPED
007C  = 0001      HOLDCH: DS      1           ;CH IS MOVED HERE IN KGETCH BEFORE CNTL & SHIFT PROC
007D  = 0001      INSDAT: DS      1
007E  = 0002      COUNTR: DS      2
                  ;
                  ;
                  ;
                  ;       80 - FF ARE RESERVED FOR USER APPLICATIONS
                  ;
                  ;
```

```
                        ;
                        ;       NOTE : SEE FLOATING POINT SUBROUTINE AREA FOR ZERO PAGE CELLS
                        ;
                        ;
                        ;
                        ;       PAGE 1   -   STACK
                        ;
                        ;
                        ;
                        ;       PAGE TWO RAM ASSIGNMENTS
                        ;
0080  = 0200                    ORG     $0200
      = 0200            INTABS  =       *               ;INTERRUPT RAM
0200  = 0002            VDSLST: DS      2               ;DISPLAY LIST NMI VECTOR
0202  = 0002            VPRCED: DS      2               ;PROCEED LINE IRQ VECTOR
0204  = 0002            VINTER: DS      2               ;INTERRUPT LINE IRQ VECTOR
0206  = 0002            VBREAK: DS      2               ;SOFTWARE BREAK (00) INSTRUCTION IRQ VECTOR
0208  = 0002            VKEYBD: DS      2               ;POKEY KEYBOARD IRQ VECTOR
020A  = 0002            VSERIN: DS      2               ;POKEY SERIAL INPUT READY IRQ
020C  = 0002            VSEROR: DS      2               ;POKEY SERIAL OUTPUT READY IRQ
020E  = 0002            VSEROC: DS      2               ;POKEY SERIAL OUTPUT COMPLETE IRQ
0210  = 0002            VTIMR1: DS      2               ;POKEY TIMER 1 IRQ
0212  = 0002            VTIMR2: DS      2               ;POKEY TIMER 2 IRQ
0214  = 0002            VTIMR4: DS      2               ;POKEY TIMER 4 IRQ
0216  = 0002            VIMIRQ: DS      2               ;IMMEDIATE IRQ VECTOR
0218  = 0002            CDTMV1: DS      2               ;COUNT DOWN TIMER 1
021A  = 0002            CDTMV2: DS      2               ;COUNT DOWN TIMER2
021C  = 0002            CDTMV3: DS      2               ;COUNT DOWN TIMER 3
021E  = 0002            CDTMV4: DS      2               ;COUNT DOWN TIMER 4
0220  = 0002            CDTMV5: DS      2               ;COUNT DOWN TIMER 5
0222  = 0002            VVBLKI: DS      2               ;IMMEDIATE VERTICAL BLANK NMI VECTOR
0224  = 0002            VVBLKD: DS      2               ;DEFERRED VERTICAL BLANK NMI VECTOR
0226  = 0002            CDTMA1: DS      2               ;COUNT DOWN TIMER 1 JSR ADDRESS
0228  = 0002            CDTMA2: DS      2               ;COUNT DOWN TIMER 2 JSR ADDRESS
022A  = 0001            CDTMF3: DS      1               ;COUNT DOWN TIMER 3 FLAG
022B  = 0001            SRTIMR: DS      1               ;SOFTWARE REPEAT TIMER
022C  = 0001            CDTMF4: DS      1               ;COUNT DOWN TIMER 4 FLAG
022D  = 0001            INTEMP: DS      1               ;IAN'S TEMP (RENAMED FROM T1 BY POPULAR DEMAND)
022E  = 0001            CDTMF5: DS      1               ;COUNT DOWN TIMER FLAG 5
022F  = 0001            SDMCTL: DS      1               ;SAVE DMACTL REGISTER
0230  = 0001            SDLSTL: DS      1               ;SAVE DISPLAY LIST LOW BYTE
0231  = 0001            SDLSTH: DS      1               ;SAVE DISPLAY LIST HI BYTE
0232  = 0001            SSKCTL: DS      1               ;SKCTL REGISTER RAM
0233  = 0001                    DS      1               ;
                        ;
0234  = 0001            LPENH:  DS      1               ;LIGHT PEN HORIZONTAL VALUE
0235  = 0001            LPENV:  DS      1               ;LIGHT PEN VERTICAL VALUE
0236  = 0002            BRKKY:  DS      2               ;BREAK KEY VECTOR
                        ;
0238  = 0002                    DS      2               ;SPARE
                        ;
023A  = 0001            CDEVIC: DS      1               ;COMMAND FRAME BUFFER - DEVICE
023B  = 0001            CCOMND: DS      1               ;COMMAND
023C  = 0001            CAUX1:  DS      1               ;COMMAND AUX BYTE 1
023D  = 0001            CAUX2:  DS      1               ;COMMAND AUX BYTE 2
```

```
                ;   NOTE: MAY NOT BE THE LAST WORD ON A PAGE
023E  = 0001    TEMP:   DS      1           ;TEMPORARY RAM CELL
                ;   NOTE: MAY NOT BE THE LAST WORD ON A PAGE
023F  = 0001    ERRFLG: DS      1           ;ERROR FLAG - ANY DEVICE ERROR EXCEPT TIME OUT
                ;
0240  = 0001    DFLAGS: DS      1           ;DISK FLAGS FROM SECTOR ONE
0241  = 0001    DBSECT: DS      1           ;NUMBER OF DISK BOOT SECTORS
0242  = 0002    BOOTAD: DS      2           ;ADDRESS WHERE DISK BOOT LOADER WILL BE PUT
0244  = 0001    COLDST: DS      1           ;COLDSTART FLAG (1=IN MIDDLE OF COLDSTART)
                ;
0245  = 0001            DS      1           ;SPARE
                ;
0246  = 0001    DSKTIM: DS      1           ;DISK TIME OUT REGISTER
                ;
0247  = 0028    LINBUF: DS      40          ;CHAR LINE BUFFER
                ;
026F  = 0001    GPRIOR: DS      1           ;GLOBAL PRIORITY CELL
                ;
0270  = 0001    PADDL0: DS      1           ;POTENTIOMETER 0 RAM CELL
0271  = 0001    PADDL1: DS      1
0272  = 0001    PADDL2: DS      1
0273  = 0001    PADDL3: DS      1
0274  = 0001    PADDL4: DS      1
0275  = 0001    PADDL5: DS      1
0276  = 0001    PADDL6: DS      1
0277  = 0001    PADDL7: DS      1
0278  = 0001    STICK0: DS      1           ;JOYSTICK 0 RAM CELL
0279  = 0001    STICK1: DS      1
027A  = 0001    STICK2: DS      1
027B  = 0001    STICK3: DS      1
027C  = 0001    PTRIG0: DS      1           ;PADDLE TRIGGER 0
027D  = 0001    PTRIG1: DS      1
027E  = 0001    PTRIG2: DS      1
027F  = 0001    PTRIG3: DS      1
0280  = 0001    PTRIG4: DS      1
0281  = 0001    PTRIG5: DS      1
0282  = 0001    PTRIG6: DS      1
0283  = 0001    PTRIG7: DS      1
0284  = 0001    STRIG0: DS      1           ;JOYSTICK TRIGGER 0
0285  = 0001    STRIG1: DS      1
0286  = 0001    STRIG2: DS      1
0287  = 0001    STRIG3: DS      1
                ;
0288  = 0001    CSTAT:  DS      1
0289  = 0001    WMODE:  DS      1
028A  = 0001    BLIM:   DS      1
028B  = 0001    IMASK:  DS      1
028C  = 0002    JVECK:  DS      2
                ;
028E  = 0002            DS      2           ;SPARE
                ;
                ;
                ;
                ;
0290  = 0001    TXTROW: DS      1           ;TEXT ROWCRS
0291  = 0002    TXTCOL: DS      2           ;TEXT COLCRS
0293  = 0001    TINDEX: DS      1           ;TEXT INDEX
```

```
0294  = 0002      TXTMSC: DS      2           ;FOOLS CONVRT INTO NEW MSC
0296  = 0006      TXTOLD: DS      6           ;OLDROW & OLDCOL FOR TEXT (AND THEN SOME)
029C  = 0001      TMPX1:  DS      1
029D  = 0001      HOLD3:  DS      1
029E  = 0001      SUBTMP: DS      1
029F  = 0001      HOLD2:  DS      1
02A0  = 0001      DMASK:  DS      1
02A1  = 0001      TMPLBT: DS      1
02A2  = 0001      ESCFLG: DS      1           ;ESCAPE FLAG
02A3  = 000F      TABMAP: DS      15
02B2  = 0004      LOGMAP: DS      4           ;LOGICAL LINE START BIT MAP
02B6  = 0001      INVFLG: DS      1           ;INVERSE VIDEO FLAG (TOGGLED BY ATARI KEY)
02B7  = 0001      FILFLG: DS      1           ;RIGHT FILL FLAG FOR DRAW
02B8  = 0001      TMPROW: DS      1
02B9  = 0002      TMPCOL: DS      2
02BB  = 0001      SCRFLG: DS      1           ;SET IF SCROLL OCCURS
02BC  = 0001      HOLD4:  DS      1           ;TEMP CELL USED IN DRAW ONLY
02BD  = 0001      HOLD5:  DS      1           ;DITTO
02BE  = 0001      SHFLOK: DS      1
02BF  = 0001      BOTSCR: DS      1           ;BOTTOM OF SCREEN : 24 NORM 4 SPLIT
                  ;
                  ;
02C0  = 0001      PCOLR0: DS      1           ;P0 COLOR
02C1  = 0001      PCOLR1: DS      1           ;P1 COLOR
02C2  = 0001      PCOLR2: DS      1           ;P2 COLOR
02C3  = 0001      PCOLR3: DS      1           ;P3 COLOR
02C4  = 0001      COLOR0: DS      1           ;COLOR 0
02C5  = 0001      COLOR1: DS      1
02C6  = 0001      COLOR2: DS      1
02C7  = 0001      COLOR3: DS      1
02C8  = 0001      COLOR4: DS      1
                  ;
                  ;
02C9  = 0017              DS      23          ;SPARE
                  ;
                  ;
                  ;
      = 02E0      GLBABS  =       *           ;GLOBAL VARIABLES
                  ;
02E0  = 0004              DS      4           ;SPARE
                  ;
02E4  = 0001      RAMSIZ: DS      1           ;RAM SIZE (HI BYTE ONLY)
02E5  = 0002      MEMTOP: DS      2           ;TOP OF AVAILABLE USER MEMORY
02E7  = 0002      MEMLO:  DS      2           ;BOTTOM OF AVAILABLE USER MEMORY
02E9  = 0001              DS      1           ;SPARE
02EA  = 0004      DVSTAT: DS      4           ;STATUS BUFFER
02EE  = 0001      CBAUDL: DS      1           ;CASSETTE BAUD RATE LOW BYTE
02EF  = 0001      CBAUDH: DS      1
                  ;
02F0  = 0001      CRSINH: DS      1           ;CURSOR INHIBIT (00 = CURSOR ON)
02F1  = 0001      KEYDEL: DS      1           ;KEY DELAY
02F2  = 0001      CH1:    DS      1
                  ;
02F3  = 0001      CHACT:  DS      1           ;CHACTL REGISTER RAM
02F4  = 0001      CHBAS:  DS      1           ;CHBAS REGISTER RAM
                  ;
02F5  = 0005              DS      5           ;SPARE BYTES
```

```
                        ;
02FA  = 0001    CHAR:   DS      1
02FB  = 0001    ATACHR: DS      1           ;ATASCII CHARACTER
02FC  = 0001    CH:     DS      1           ;GLOBAL VARIABLE FOR KEYBOARD
02FD  = 0001    FILDAT: DS      1           ;RIGHT FILL DATA (DRAW)
02FE  = 0001    DSPFLG: DS      1           ;DISPLAY FLAG : DISPLAY CNTLS IF NON-ZERO
02FF  = 0001    SSFLAG: DS      1           ;START/STOP FLAG FOR PAGING (CNTL 1). CLEARED BY BREAK
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;       PAGE THREE RAM ASSIGNMENTS
                        ;
      = 0300    DCB     =       *           ;DEVICE CONTROL BLOCK
0300  = 0001    DDEVIC: DS      1           ;PERIPHERAL UNIT 1 BUS I.D. NUMBER
0301  = 0001    DUNIT:  DS      1           ;UNIT NUMBER
0302  = 0001    DCOMND: DS      1           ;BUS COMMAND
0303  = 0001    DSTATS: DS      1           ;COMMAND TYPE/STATUS RETURN
0304  = 0001    DBUFLO: DS      1           ;DATA BUFFER POINTER LOW BYTE
0305  = 0001    DBUFHI: DS      1
0306  = 0001    DTIMLO: DS      1           ;DEVICE TIME OUT IN 1 SECOND UNITS
0307  = 0001    DUNUSE: DS      1           ;UNUSED BYTE
0308  = 0001    DBYTLO: DS      1           ;NUMBER OF BYTES TO BE TRANSFERRED LOW BYTE
0309  = 0001    DBYTHI: DS      1
030A  = 0001    DAUX1:  DS      1           ;COMMAND AUXILIARY BYTE 1
030B  = 0001    DAUX2:  DS      1
                        ;
030C  = 0002    TIMER1: DS      2           ;INITIAL TIMER VALUE
030E  = 0001    ADDCOR: DS      1           ;ADDITION CORRECTION
030F  = 0001    CASFLG: DS      1           ;CASSETTE MODE WHEN SET
0310  = 0002    TIMER2: DS      2           ;FINAL TIMER VALUE. THESE TWO TIMER VALUES
                ; ARE USED TO COMPUTE INTERVAL FOR BAUD RATE
0312  = 0002    TEMP1:  DS      2           ;TEMPORARY STORAGE REGISTER
0314  = 0001    TEMP2:  DS      1           ;TEMPORARY STORAGE REGISTER
0315  = 0001    TEMP3:  DS      1           ;TEMPORARY STORAGE REGISTER
0316  = 0001    SAVIO:  DS      1           ;SAVE SERIAL IN DATA PORT
0317  = 0001    TIMFLG: DS      1           ;TIME OUT FLAG FOR BAUD RATE CORRECTION
0318  = 0001    STACKP: DS      1           ;SIO STACK POINTER SAVE CELL
0319  = 0001    TSTAT:  DS      1           ;TEMPORARY STATUS HOLDER
                        ;
                        ;
                        ;
031A  = 0026    HATABS: DS      38          ;HANDLER ADDRESS TABLE
      = 0021    MAXDEV  =       *-HATABS-5  ;MAXIMUM HANDLER ADDRESS INDEX
                        ;
                        ;     NOTE : THE ENTIRE IOCB DEFINITIONS HAVE BEEN MODIFIED
                        ;
0340  = 0340    IOCB:   ORG     *           ;I/O CONTROL BLOCKS
0340  = 0001    ICHID:  DS      1           ;HANDLER INDEX NUMBER (FF = IOCB FREE)
0341  = 0001    ICDNO:  DS      1           ;DEVICE NUMBER (DRIVE NUMBER)
0342  = 0001    ICCOM:  DS      1           ;COMMAND CODE
0343  = 0001    ICSTA:  DS      1           ;STATUS OF LAST IOCB ACTION
0344  = 0001    ICBAL:  DS      1           ;BUFFER ADDRESS LOW BYTE
0345  = 0001    ICBAH:  DS      1
```

```
0346  = 0001      ICPTL:  DS      1           ;PUT BYTE ROUTINE ADDRESS - 1
0347  = 0001      ICPTH:  DS      1
0348  = 0001      ICBLL:  DS      1           ;BUFFER LENGTH LOW BYTE
0349  = 0001      ICBLH:  DS      1
034A  = 0001      ICAX1:  DS      1           ;AUXILIARY INFORMATION FIRST BYTE
034B  = 0001      ICAX2:  DS      1
034C  = 0004      ICSPR:  DS      4           ;FOUR SPARE BYTES
0350  = 0070              DS      MAXIOC-IOCBSZ
                  ;
03C0  = 0028      PRNBUF: DS      40          ;PRINTER BUFFER
                  ;
03E8  = 0015              DS      21          ;SPARE BYTES
                  ;
                  ;
                  ;
                  ;
                  ;
                  ;
                  ;       PAGE FOUR RAM ASSIGNMENTS
                  ;
03FD  = 0083      CASBUF: DS      131         ;CASSETTE BUFFER
                  ;
                  ; USER AREA STARTS HERE AND GOES TO END OF PAGE FIVE
0480  = 0080      USAREA: DS      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
                  ;
      = 0006      FPREC   =       6           ;FLOATING PT PRECISION (# OF BYTES)
                  ; IF CARRY USED THEN CARRY CLEAR => NO ERROR, CARR
      = D800      AFP     =       $D800       ;ASCII->FLOATING POINT (FP)
                  ;                               INBUFF+CIX -> FR0, CIX, CARRY
      = D8E6      FASC    =       $D8E6       ;FP -> ASCII FR0-> LBUFF (INBUFF)
      = D9AA      IFP     =       $D9AA       ;INTEGER -> FP
                  ;                               0-$FFFF (LSB,MSB) IN FR0,FR0+1->FR0
      = D9D2      FPI     =       $D9D2       ;FP -> INTEGER FR0 -> FR0,FR0+1, CARRY
      = DA60      FSUB    =       $DA60       ;FR0 <- FR0 - FR1 ,CARRY
```

```
      = DA66      FADD    =       $DA66       ;FR0 <- FR0 + FR1 ,CARRY
      = DADB      FMUL    =       $DADB       ;FR0 <- FR0 * FR1 ,CARRY
      = DB28      FDIV    =       $DB28       ;FR0 <- FR0 / FR1 ,CARRY
      = DD89      FLD0R   =       $DD89       ;FLOATING LOAD REG0   FR0  <- (X,Y)
      = DD8D      FLD0P   =       $DD8D       ;   "      "    "     FR0  <- (FLPTR)
      = DD98      FLD1R   =       $DD98       ;   "      "   REG1   FR1  <- (X,Y)
      = DD9C      FLD1P   =       $DD9C       ;   "      "    "     FR1  <- (FLPTR)
      = DDA7      FST0R   =       $DDA7       ;FLOATING STORE REG0 (X,Y) <- FR0
      = DDAB      FST0P   =       $DDAB       ;    "     "    " (FLPTR)  <- FR0
      = DDB6      FMOVE   =       $DDB6       ;FR1 <- FR0
      = DD40      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
      = DDC0      EXP     =       $DDC0       ;FR0 <- E**FR0 = EXP10(FR0 * LOG10(E)) CARRY
      = DDCC      EXP10   =       $DDCC       ;FR0 <- 10**FR0 CARRY
      = DECD      LOG     =       $DECD       ;FR0 <- LN(FR0) = LOG10(FR0)/LOG10(E) CARRY
      = DED1      LOG10   =       $DED1       ;FR0 <- LOG10 (FR0) CARRY
                  ; THE FOLLOWING ARE IN BASIC CARTRIDGE:
      = BD81      SIN     =       $BD81       ;FR0 <- SIN(FR0) DEGFLG=0 =>RADS, 6=>DEG. CA
      = BD73      COS     =       $BD73       ;FR0 <- COS(FR0) CARRY
      = BE43      ATAN    =       $BE43       ;FR0 <- ATAN(FR0) CARRY
      = BEB1      SQR     =       $BEB1       ;FR0 <- SQUAREROOT(FR0) CARRY
                  ; FLOATING POINT ROUTINES ZERO PAGE (NEEDED ONLY IF F.P. ROUTINES ARE CALLED)
0500  = 00D4              ORG     $D4
00D4  = 0006      FR0:    DS      FPREC       ;FP REG0
00DA  = 0006      FRE:    DS      FPREC
00E0  = 0006      FR1:    DS      FPREC       ;FP REG1
00E6  = 0006      FR2:    DS      FPREC
00EC  = 0001      FRX:    DS      1           ;FP SPARE
00ED  = 0001      EEXP:   DS      1           ;VALUE OF E
00EE  = 0001      NSIGN:  DS      1           ;SIGN OF #
00EF  = 0001      ESIGN:  DS      1           ;SIGN OF EXPONENT
00F0  = 0001      FCHRFLG:DS      1           ;1ST CHAR FLAG
00F1  = 0001      DIGRT:  DS      1           ;# OF DIGITS RIGHT OF DECIMAL
00F2  = 0001      CIX:    DS      1           ;CURRENT INPUT INDEX
00F3  = 0002      INBUFF: DS      2           ;POINTS TO USER'S LINE INPUT BUFFER
00F5  = 0002      ZTEMP1: DS      2
00F7  = 0002      ZTEMP4: DS      2
00F9  = 0002      ZTEMP3: DS      2
00FB              DEGFLG
00FB  = 0001      RADFLG: DS      1           ;0=RADIANS, 6=DEGREES
      = 0000      RADON   =       0           ;INDICATES RADIANS
      = 0006      DEGON   =       6           ;INDICATES DEGREES
00FC  = 0002      FLPTR:  DS      2           ;POINTS TO USER'S FLOATING PT NUMBER
00FE  = 0002      FPTR2:  DS      2
                  ; FLOATING PT ROUTINES' NON-ZERO PAGE RAM
                  ; (NEEDED ONLY IF F.P. ROUTINES CALLED)
0100  = 057E              ORG     $57E
057E  = 0001      LBPR1:  DS      1           ;LBUFF PREFIX 1
057F  = 0001      LBPR2:  DS      1           ;LBUFF PREFIX 2
0580  = 0080      LBUFF:  DS      128         ;LINE BUFFER
      = 05E0      PLYARG  =       LBUFF+$60   ;POLYNOMIAL ARGUMENTS
      = 05E6      FPSCR   =       PLYARG+FPREC
      = 05EC      FPSCR1  =       FPSCR+FPREC
      = 05E6      FSCR    =       FPSCR
      = 05EC      FSCR1   =       FPSCR1
```

```
= 05FF      LBFEND  =       *-1         ;END OF LBUFF
                ;
                ;
                ;
                ;
                ;
                ;
                ;
                ;
                ;       COLLEEN MNEMONICS
                ;
= D200      POKEY   =       $D200       ;VBLANK ACTION:           DESCRIPTION:
= D200      POT0    =       POKEY+0     ;POT0-->PADDL0           0-227 IN RAM CELL
= D201      POT1    =       POKEY+1     ;POT1-->PADDL1           0-227 IN RAM CELL
= D202      POT2    =       POKEY+2     ;POT2-->PADDL2           0-227 IN RAM CELL
= D203      POT3    =       POKEY+3     ;POT3-->PADDL3           0-227 IN RAM CELL
= D204      POT4    =       POKEY+4     ;POT4-->PADDL4           0-227 IN RAM CELL
= D205      POT5    =       POKEY+5     ;POT5-->PADDL5           0-227 IN RAM CELL
= D206      POT6    =       POKEY+6     ;POT6-->PADDL6           0-227 IN RAM CELL
= D207      POT7    =       POKEY+7     ;POT7-->PADDL7           0-227 IN RAM CELL
= D208      ALLPOT  =       POKEY+8     ;???
= D209      KBCODE  =       POKEY+9
= D20A      RANDOM  =       POKEY+10
= D20B      POTGO   =       POKEY+11    ;STROBED
= D20D      SERIN   =       POKEY+13
= D20E      IRQST   =       POKEY+14
= D20F      SKSTAT  =       POKEY+15
= D200      AUDF1   =       POKEY+0
= D201      AUDC1   =       POKEY+1
= D202      AUDF2   =       POKEY+2
= D203      AUDC2   =       POKEY+3
= D204      AUDF3   =       POKEY+4
= D205      AUDC3   =       POKEY+5
= D206      AUDF4   =       POKEY+6
= D207      AUDC4   =       POKEY+7
= D208      AUDCTL  =       POKEY+8     ;NONE               AUDCTL<--[SIO]
= D209      STIMER  =       POKEY+9
= D20A      SKRES   =       POKEY+10    ;NONE                SKRES<--[SIO]
= D20D      SEROUT  =       POKEY+13    ;NONE               SEROUT<--[SIO]
= D20E      IRQEN   =       POKEY+14    ;POKMSK-->IRQEN (AFFECTED BY OPEN S: OR E:)
= D20F      SKCTL   =       POKEY+15    ;SSKCTL-->SKCTL      SSKCTL<--[SIO]
                ;
= D000      CTIA    =       $D000       ;VBLANK ACTION:       DESCRIPTION:
= D000      HPOSP0  =       CTIA+0
= D001      HPOSP1  =       CTIA+1
= D002      HPOSP2  =       CTIA+2
= D003      HPOSP3  =       CTIA+3
= D004      HPOSM0  =       CTIA+4
= D005      HPOSM1  =       CTIA+5
= D006      HPOSM2  =       CTIA+6
= D007      HPOSM3  =       CTIA+7
= D008      SIZEP0  =       CTIA+8
= D009      SIZEP1  =       CTIA+9
= D00A      SIZEP2  =       CTIA+10
= D00B      SIZEP3  =       CTIA+11
= D00C      SIZEM   =       CTIA+12
```

```
= D00D      GRAFP0  =       CTIA+13
= D00E      GRAFP1  =       CTIA+14
= D00F      GRAFP2  =       CTIA+15
= D010      GRAFP3  =       CTIA+16
= D011      GRAFM   =       CTIA+17
= D012      COLPM0  =       CTIA+18     ;PCOLR0-->COLPM0      WITH ATTRACT MODE
= D013      COLPM1  =       CTIA+19     ;PCOLR1-->COLPM1      WITH ATTRACT MODE
= D014      COLPM2  =       CTIA+20     ;PCOLR2-->COLPM2      WITH ATTRACT MODE
= D015      COLPM3  =       CTIA+21     ;PCOLR3-->COLPM3      WITH ATTRACT MODE
= D016      COLPF0  =       CTIA+22     ;COLOR0-->COLPF0      WITH ATTRACT MODE
= D017      COLPF1  =       CTIA+23     ;COLOR1-->COLPF1      WITH ATTRACT MODE
= D018      COLPF2  =       CTIA+24     ;COLOR2-->COLPF2      WITH ATTRACT MODE
= D019      COLPF3  =       CTIA+25     ;COLOR3-->COLPF3      WITH ATTRACT MODE
= D01A      COLBK   =       CTIA+26     ;COLOR4-->COLBK       WITH ATTRACT MODE
= D01B      PRIOR   =       CTIA+27     ;(ON OPEN S: OR E:)   GPRIOR-->PRIOR
= D01C      VDELAY  =       CTIA+28
= D01D      GRACTL  =       CTIA+29
= D01E      HITCLR  =       CTIA+30
= D01F      CONSOL  =       CTIA+31     ;$08-->CONSOL         TURN OFF SPEAKER
= D000      M0PF    =       CTIA+0
= D001      M1PF    =       CTIA+1
= D002      M2PF    =       CTIA+2
= D003      M3PF    =       CTIA+3
= D004      P0PF    =       CTIA+4
= D005      P1PF    =       CTIA+5
= D006      P2PF    =       CTIA+6
= D007      P3PF    =       CTIA+7
= D008      M0PL    =       CTIA+8
= D009      M1PL    =       CTIA+9
= D00A      M2PL    =       CTIA+10
= D00B      M3PL    =       CTIA+11
= D00C      P0PL    =       CTIA+12
= D00D      P1PL    =       CTIA+13
= D00E      P2PL    =       CTIA+14
= D00F      P3PL    =       CTIA+15
= D010      TRIG0   =       CTIA+16     ;TRIG0-->STRIG0
= D011      TRIG1   =       CTIA+17     ;TRIG1-->STRIG1
= D012      TRIG2   =       CTIA+18     ;TRIG2-->STRIG2
= D013      TRIG3   =       CTIA+19     ;TRIG3-->STRIG3
            ;
= D400      ANTIC   =       $D400       ;VBLANK ACTION        DESCRIPTION
= D400      DMACTL  =       ANTIC+0     ;DMACTL<--SDMCTL      ON OPEN S: OR E:
= D401      CHACTL  =       ANTIC+1     ;CHACTL<--CHACT       ON OPEN S: OR E:
= D402      DLISTL  =       ANTIC+2     ;DLISTL<--SDLSTL      ON OPEN S: OR E:
= D403      DLISTH  =       ANTIC+3     ;DLISTH<--SDLSTH      ON OPEN S: OR E:
= D404      HSCROL  =       ANTIC+4
= D405      VSCROL  =       ANTIC+5
= D407      PMBASE  =       ANTIC+7
= D409      CHBASE  =       ANTIC+9     ;CHBASE<--CHBAS       ON OPEN S: OR E:
= D40A      WSYNC   =       ANTIC+10
= D40B      VCOUNT  =       ANTIC+11
= D40C      PENH    =       ANTIC+12
= D40D      PENV    =       ANTIC+13
= D40E      NMIEN   =       ANTIC+14    ;NMIEN<--40 POWER     ON AND [SETVBV]
= D40F      NMIRES  =       ANTIC+15    ;STROBED
= D40F      NMIST   =       ANTIC+15
= D300      PIA     =       $D300       ;VBLANK ACTION        DESCRIPTION
```

```
= D300      PORTA   =       PIA+0       ;PORTA-->STICK0,1     X-Y CONTROLLERS
= D301      PORTB   =       PIA+1       ;PORTB-->STICK2,3     X-Y CONTROLLERS
= D302      PACTL   =       PIA+2       ;NONE                 PACTL<--3C [INIT]
= D303      PBCTL   =       PIA+3       ;NONE                 PBCTL<--3C [INIT]
                ;
                ;
                ;
                ; EJECT
```

```
                        LIST    S
                ;               UPDATED BY AL MILLER 3-9-79
= 0030          ASCZER  =       '0'             ;ASCII ZERO
= 003A          COLON   =       $3A             ;ASCII COLON
= 009B          EOL     =       $9B             ;END OF RECORD
```

```
                    ;
                    ; CIO JUMP VECTOR FOR USERS
0600  = E456                ORG     CIOV
E456  4CC4E4                JMP     CIO         ;GO TO CIO
                    ;
                    ; CIO INIT JUMP VECTOR FOR POWER UP
E459  = E46E                ORG     CIOINV
E46E  4CA6E4                JMP     CIOINT      ;GO TO INIT
                    ;
                    ;
                    ; ERROR ROUTINE ADDRESS EQUATE
                    ; ERRTNH =ERRTN/256         "MOVED TO LINE 788"
                    ; ERRTNL =-ERRTNH*256+ERRTN "MOVED TO LINE 789"
                    ;
                    ;
E471  = E4A6                ORG     CIOORG
                    ;
                    ; CIO INITIALIZATION (CALLED BY MONITOR AT POWER UP)
E4A6  A200          CIOINT: LDX     #0
E4A8  A9FF          CIOI1:  LDA     #IOCFRE     ;SET ALL IOCB'S TO FREE
E4AA  9D4003                STA     ICHID,X     ;BY SETTING HANDLER ID'S=$FF
E4AD  A9C0                  LDA     #ERRTNL
E4AF  9D4603                STA     ICPTL,X     ;POINT PUT TO ERROR ROUTINE
E4B2  A9E4                  LDA     #ERRTNH
E4B4  9D4703                STA     ICPTH,X
E4B7  8A                    TXA
E4B8  18                    CLC
E4B9  6910                  ADC     #IOCBSZ     ;BUMP INDEX BY SIZE
E4BB  AA                    TAX
E4BC  C980                  CMP     #MAXIOC     ;DONE?
E4BE  90E8 ^E4A8            BCC     CIOI1       ;NO
E4C0  60                    RTS                 ;YES, RETURN
                    ;
                    ; ERROR ROUTINE FOR ILLEGAL PUT
      = E4C0        ERRTN   =       *-1
      = 00E4        ERRTNH  =       HIGH ERRTN
      = 00C0        ERRTNL  =       LOW ERRTN
E4C1  A085                  LDY     #NOTOPN     ;IOCB NOT OPEN
E4C3  60                    RTS
```

```
                        ;
                        ; CIO LOCAL RAM (USES SPARE BYTES IN ZERO PAGE IOCB)
      = 002C            ENTVEC  =       ICSPRZ
                        ;
                        ; CIO MAIN ROUTINE
                        ;
                        ; CIO INTERFACES BETWEEN USER AND INPUT/OUTPUT DE
E4C4  852F              CIO:    STA     CIOCHR      ;SAVE POSSIBLE OUTPUT CHARACTER
E4C6  862E                      STX     ICIDNO      ;SAVE IOCB NUMBER * N
                        ;
                        ; CHECK FOR LEGAL IOCB
E4C8  8A                        TXA
E4C9  290F                      AND     #$F         ;IS IOCB MULTIPLE OF 16?
E4CB  D004 ^E4D1                BNE     CIERR1      ;NO, ERROR
E4CD  E080                      CPX     #MAXIOC     ;IS INDEX TOO LARGE?
E4CF  9005 ^E4D6                BCC     IOC1        ;NO
                        ;
                        ; INVALID IOCB NUMBER -- RETURN ERROR
E4D1  A086              CIERR1: LDY     #BADIOC     ;ERROR CODE
E4D3  4C1BE6                    JMP     CIRTN1      ;RETURN
                        ;
                        ; MOVE USER IOCB TO ZERO PAGE
E4D6  A000              IOC1:   LDY     #0
E4D8  BD4003            IOC1A:  LDA     IOCB,X      ;USER IOCB
E4DB  992000                    STA     IOCBAS,Y    ;TO ZERO PAGE
E4DE  E8                        INX
E4DF  C8                        INY
E4E0  C00C                      CPY     #12         ;12 BYTES
E4E2  90F4 ^E4D8                BCC     IOC1A
                        ;
                        ; COMPUTE CIO INTERNAL VECTOR FOR COMMAND
E4E4  A084                      LDY     #NVALID     ;ASSUME INVALID CODE
E4E6  A522                      LDA     ICCOMZ      ;COMMAND CODE TO INDEX
E4E8  C903                      CMP     #OPEN       ;IS COMMAND LEGAL?
E4EA  9025 ^E511                BCC     CIERR4      ;NO
E4EC  A8                        TAY
                        ;
                        ; MOVE COMMAND TO ZERO BASE FOR INDEX
E4ED  C00E                      CPY     #SPECIL     ;IS COMMAND SPECIAL?
E4EF  9002 ^E4F3                BCC     IOC2        ;NO
E4F1  A00E                      LDY     #SPECIL     ;YES, SET SPECIAL OFFSET INDEX
E4F3  8417              IOC2:   STY     ICCOMT      ;SAVE COMMAND FOR VECTOR
E4F5  B9C6E6                    LDA     COMTAB-3,Y  ;GET VECTOR OFFSET FROM TABLE
E4F8  F00F ^E509                BEQ     CIOPEN      ;GO IF OPEN COMMAND
E4FA  C902                      CMP     #2          ;IS IT CLOSE?
E4FC  F035 ^E533                BEQ     CICLOS      ;YES
E4FE  C908                      CMP     #8          ;IS IT STATUS OR SPECIAL?
E500  B04C ^E54E                BCS     CISTSP      ;YES
E502  C904                      CMP     #4          ;IS IT READ?
E504  F063 ^E569                BEQ     CIREAD      ;YES
E506  4CC9E5                    JMP     CIWRIT      ;ELSE, MUST BE WRITE
```

```
                        ;
                        ; OPEN COMMAND
                        ;
                        ; FIND DEVICE HANDLER IN HANDLER ADDRESS TABLE
E509  A520      CIOPEN: LDA     ICHIDZ      ;GET HANDLER ID
E50B  C9FF              CMP     #IOCFRE     ;IS THIS IOCB CLOSED?
E50D  F005 ^E514        BEQ     IOC6        ;YES
                        ;
                        ; ERROR -- IOCB ALREADY OPEN
E50F  A081      CIERR3: LDY     #PRVOPN     ;ERROR CODE
E511  4C1BE6    CIERR4: JMP     CIRTN1      ;RETURN
                        ;
                        ; GO FIND DEVICE
E514  209EE6    IOC6:   JSR     DEVSRC      ;CALL DEVICE SEARCH
E517  B0F8 ^E511        BCS     CIERR4      ;GO IF DEVICE NOT FOUND
                        ;
                        ; DEVICE FOUND, INITIALIZE IOCB FOR OPEN
                        ;
                        ; COMPUTE HANDLER ENTRY POINT
E519  203DE6    IOC7:   JSR     COMENT
E51C  B0F3 ^E511        BCS     CIERR4      ;GO IF ERROR IN COMPUTE
                        ;
                        ; GO TO HANDLER FOR INITIALIZATION
E51E  2089E6            JSR     GOHAND      ;USE INDIRECT JUMP
                        ;
                        ; STORE PUT BYTE ADDRESS-1 INTO IOCB
E521  A90B              LDA     #PUTCHR     ;SIMULATE PUT CHARACTER
E523  8517              STA     ICCOMT
E525  203DE6            JSR     COMENT      ;COMPUTE ENTRY POINT
E528  A52C              LDA     ICSPRZ      ;MOVE COMPUTED VALUE
E52A  8526              STA     ICPTLZ      ;TO PUT BYTE ADDRESS
E52C  A52D              LDA     ICSPRZ+1
E52E  8527              STA     ICPTHZ
E530  4C1DE6            JMP     CIRTN2      ;RETURN TO USER
```

```
                        ;
                        ;
                        ; CLOSE COMMAND
E533  A001      CICLOS: LDY     #SUCCES     ;ASSUME GOOD CLOSE
E535  8423              STY     ICSTAZ
E537  203DE6            JSR     COMENT      ;COMPUTE HANDLER ENTRY POINT
E53A  B003 ^E53F        BCS     CICLO2      ;GO IF ERROR IN COMPUTE
E53C  2089E6            JSR     GOHAND      ;GO TO HANDLER TO CLOSE DEVICE
E53F  A9FF      CICLO2: LDA     #IOCFRE     ;GET IOCB "FREE" VALUE
E541  8520              STA     ICHIDZ      ;SET HANDLER ID
E543  A9E4              LDA     #ERRTNH
E545  8527              STA     ICPTHZ      ;SET PUT BYTE TO POINT TO ERROR
E547  A9C0              LDA     #ERRTNL
E549  8526              STA     ICPTLZ
E54B  4C1DE6            JMP     CIRTN2      ;RETURN
                        ;
                        ;
                        ; STATUS AND SPECIAL REQUESTS
                        ; DO IMPLIED OPEN IF NECESSARY AND GO TO DEVICE
E54E  A520      CISTSP: LDA     ICHIDZ      ;IS THERE A HANDLER ID?
E550  C9FF              CMP     #IOCFRE
E552  D005 ^E559        BNE     CIST1       ;YES
                        ;
                        ; IOCB IS FREE, DO IMPLIED OPEN
E554  209EE6            JSR     DEVSRC      ;FIND DEVICE IN TABLE
E557  B0B8 ^E511        BCS     CIERR4      ;GO IF ERROR IN COMPUTE
                        ;
                        ; COMPUTE AND GO TO ENTRY POINT IN HANDLER
E559  203DE6    CIST1:  JSR     COMENT      ;COMPUTER HANDLER ENTRY VECTOR
E55C  2089E6            JSR     GOHAND      ;GO TO HANDLER
                        ;
                        ; RESTORE HANDLER INDEX (DO IMPLIED CLOSE)
E55F  A62E              LDX     ICIDNO      ;IOCB INDEX
E561  BD4003            LDA     ICHID,X     ;GET ORIGINAL HANDLER ID
E564  8520              STA     ICHIDZ      ;RESTORE ZERO PAGE
E566  4C1DE6            JMP     CIRTN2      ;RETURN
```

```
                        ;
                        ; READ -- DO GET COMMANDS
E569  A522      CIREAD: LDA     ICCOMZ      ;GET COMMAND BYTE
E56B  252A              AND     ICAX1Z      ;IS THIS READ LEGAL?
E56D  D005 ^E574        BNE     RCI1A       ;YES
                        ;
                        ; ILLEGAL READ -- IOCB OPENED FOR WRITE ONLY
E56F  A083              LDY     #WRONLY     ;ERROR CODE
E571  4C1BE6    RCI1B:  JMP     CIRTN1      ;RETURN
                        ;
                        ; COMPUTE AND CHECK ENTRY POINT
E574  203DE6    RCI1A:  JSR     COMENT      ;COMPUTE ENTRY POINT
E577  B0F8 ^E571        BCS     RCI1B       ;GO IF ERROR IN COMPUTE
                        ;
                        ; GET RECORD OR CHARACTERS
E579  A528              LDA     ICBLLZ
E57B  0529              ORA     ICBLLZ+1    ;IS BUFFER LENGTH ZERO?
E57D  D008 ^E587        BNE     RCI3        ;NO
E57F  2089E6            JSR     GOHAND
E582  852F              STA     CIOCHR
E584  4C1DE6            JMP     CIRTN2
                        ;
                        ; LOOP TO FILL BUFFER OR END RECORD
E587  2089E6    RCI3:   JSR     GOHAND      ;GO TO HANDLER TO GET BYTE
E58A  852F              STA     CIOCHR      ;SAVE BYTE
E58C  3035 ^E5C3        BMI     RCI4        ;END TRANSFER IF ERROR
E58E  A000              LDY     #0
E590  9124              STA     (ICBALZ),Y  ;PUT BYTE IN USER BUFFER
E592  2070E6            JSR     INCBFP      ;INCREMENT BUFFER POINTER
E595  A522              LDA     ICCOMZ      ;GET COMMAND CODE
E597  2902              AND     #2          ;IS IT GET RECORD?
E599  D00C ^E5A7        BNE     RCI1        ;NO
                        ;
                        ; CHECK FOR EOL ON TEXT RECORDS
E59B  A52F              LDA     CIOCHR      ;GET BYTE
E59D  C99B              CMP     #EOL        ;IS IT AN EOL?
E59F  D006 ^E5A7        BNE     RCI1        ;NO
E5A1  2063E6            JSR     DECBFL      ;YES, DECREMENT BUFFER LENGTH
E5A4  4CC3E5            JMP     RCI4        ;END TRANSFER
                        ;
                        ; CHECK BUFFER FULL
E5A7  2063E6    RCI1:   JSR     DECBFL      ;DECREMENT BUFFER LENGTH
E5AA  D0DB ^E587        BNE     RCI3        ;CONTINUE IF NON ZERO
```

```
                    ;
                    ; BUFFER FULL, RECORD NOT ENDED
                    ; DISCARD BYTES UNTIL END OF RECORD
E5AC  A522          RCI2:   LDA     ICCOMZ      ;GET COMMAND BYTE
E5AE  2902                  AND     #2          ;IS IT GET CHARACTER?
E5B0  D011 ^E5C3            BNE     RCI4        ;YES, END TRANSFER
                    ;
                    ; LOOP TO WAIT FOR EOL
E5B2  2089E6        RCI6:   JSR     GOHAND      ;GET BYTE FROM HANDLER
E5B5  852F                  STA     CIOCHR      ;SAVE CHARACTER
E5B7  300A ^E5C3            BMI     RCI4        ;GO IF ERROR
                    ;
                    ; TEXT RECORD, WAIT FOR EOL
E5B9  A52F                  LDA     CIOCHR      ;GET GOT BYTE
E5BB  C99B                  CMP     #EOL        ;IS IT EOL?
E5BD  D0F3 ^E5B2            BNE     RCI6        ;NO, CONTINUE
                    ;
                    ; END OF RECORD, BUFFER FULL -- SEND TRUNCATED RECORD MESSAGE
E5BF  A989          RCI11:  LDA     #TRNRCD     ;ERROR CODE
E5C1  8523                  STA     ICSTAZ      ;STORE IN IOCB
                    ;
                    ; TRANSFER DONE
E5C3  2077E6        RCI4:   JSR     SUBBFL      ;SET FINAL BUFFER LENGTH
E5C6  4C1DE6                JMP     CIRTN2      ;RETURN
```

```
                        ;
                        ; WRITE -- DO PUT COMMANDS
E5C9  A522      CIWRIT: LDA     ICCOMZ      ;GET COMMAND BYTE
E5CB  252A              AND     ICAX1Z      ;IS THIS WRITE LEGAL?
E5CD  D005 ^E5D4        BNE     WCI1A       ;YES
                        ;
                        ; ILLEGAL WRITE -- DEVICE OPENED FOR READ ONLY
E5CF  A087              LDY     #RDONLY     ;ERROR CODE
E5D1  4C1BE6    WCI1B:  JMP     CIRTN1      ;RETURN
                        ;
                        ; COMPUTE AND CHECK ENTRY POINT
E5D4  203DE6    WCI1A:  JSR     COMENT      ;COMPUTE HANDLER ENTRY POINT
E5D7  B0F8 ^E5D1        BCS     WCI1B       ;GO IF ERROR IN COMPUTE
                        ;
                        ; PUT RECORD OR CHARACTERS
E5D9  A528              LDA     ICBLLZ
E5DB  0529              ORA     ICBLLZ+1    ;IS BUFFER LENGTH ZERO?
E5DD  D006 ^E5E5        BNE     WCI3        ;NO
E5DF  A52F              LDA     CIOCHR      ;GET CHARACTER
E5E1  E628              INC     ICBLLZ      ;SET BUFFER LENGTH=1
E5E3  D006 ^E5EB        BNE     WCI4        ;THEN JUST TRANSFER ONE BYTE
                        ;
                        ; LOOP TO TRANSFER BYTES FROM BUFFER TO HANDLER

E5E5  A000      WCI3:   LDY     #0
E5E7  B124              LDA     (ICBALZ),Y  ;GET BYTE FROM BUFFER
E5E9  852F              STA     CIOCHR      ;SAVE
E5EB  2089E6    WCI4:   JSR     GOHAND      ;GO PUT BYTE
E5EE  3025 ^E615        BMI     WCI5        ;END IF ERROR
E5F0  2070E6            JSR     INCBFP      ;INCREMENT BUFFER POINTER
                        ;
                        ; CHECK FOR TEXT RECORD
E5F3  A522              LDA     ICCOMZ      ;GET COMMAND BYTE
E5F5  2902              AND     #2          ;IS IT PUT RECORD?
E5F7  D00C ^E605        BNE     WCI1        ;NO
                        ;
                        ; TEXT RECORD -- CHECK FOR EOL TRANSFER
E5F9  A52F              LDA     CIOCHR      ;GET LAST CHARACTER
E5FB  C99B              CMP     #EOL        ;IS IT AN EOL?
E5FD  D006 ^E605        BNE     WCI1        ;NO
E5FF  2063E6            JSR     DECBFL      ;DECREMENT BUFFER LENGTH
E602  4C15E6            JMP     WCI5        ;END TRANSFER
                        ;
                        ; CHECK FOR BUFFER EMPTY
E605  2063E6    WCI1:   JSR     DECBFL      ;DECREMENT BUFFER LENGTH
E608  D0DB ^E5E5        BNE     WCI3        ;CONTINUE IF NON ZERO
```

```
                ;
                ; BUFFER EMPTY, RECORD NOT FILLED
                ; CHECK TYPE OF TRANSFER
E60A  A522      WCI2:   LDA     ICCOMZ      ;GET COMMAND CODE
E60C  2902              AND     #2          ;IS IT PUT CHARACTER?
E60E  D005 ^E615        BNE     WCI5        ;YES, END TRANSFER
                ;
                ; PUT RECORD (TEXT), BUFFER EMPTY, SEND EOL
E610  A99B              LDA     #EOL
E612  2089E6            JSR     GOHAND      ;GO TO HANDLER
                ;
                ; END PUT TRANSFER
E615  2077E6    WCI5:   JSR     SUBBFL      ;SET ACTUAL PUT BUFFER LENGTH
E618  4C1DE6            JMP     CIRTN2      ;RETURN
```

```
                        ;
                        ; CIO RETURNS
                        ; RETURNS WITH Y=STATUS
E61B  8423              CIRTN1: STY     ICSTAZ      ;SAVE STATUS
                        ;
                        ; RETURNS WITH STATUS STORED IN ICSTAZ
                        ; MOVE IOCB IN ZERO PAGE BACK TO USER AREA
E61D  A42E              CIRTN2: LDY     ICIDNO      ;GET IOCB INDEX
E61F  B94403                    LDA     ICBAL,Y
E622  8524                      STA     ICBALZ      ;RESTORE USER BUFFER POINTER
E624  B94503                    LDA     ICBAH,Y
E627  8525                      STA     ICBAHZ
E629  A200                      LDX     #0          ;LOOP COUNT AND INDEX
E62B  B520              CIRT3:  LDA     IOCBAS,X    ;ZERO PAGE
E62D  994003                    STA     IOCB,Y      ;TO USER AREA
E630  E8                        INX
E631  C8                        INY
E632  E00C                      CPX     #12         ;12 BYTES
E634  90F5 ^E62B                BCC     CIRT3
                        ;
                        ; RESTORE A,X, & Y
E636  A52F                      LDA     CIOCHR      ;GET LAST CHARACTER
E638  A62E                      LDX     ICIDNO      ;IOCB INDEX
E63A  A423                      LDY     ICSTAZ      ;GET STATUS AND SET FLAGS
E63C  60                        RTS                 ;RETURN TO USER
```

```
                        ;
                        ;
                        ; CIO SUBROUTINES
                        ;
                        ; COMENT -- CHECK AND COMPUTE HANDLER ENTRY POINT
E63D  A420      COMENT: LDY     ICHIDZ      ;GET HANDLER INDEX
E63F  C022              CPY     #MAXDEV+1   ;IS IT A LEGAL INDEX?
E641  9004 ^E647        BCC     COM1        ;YES
                        ;
                        ; ILLEGAL HANDLER INDEX MEANS DEVICE NOT OPEN FOR OPERATION
E643  A085              LDY     #NOTOPN     ;ERROR CODE
E645  B01B ^E662        BCS     COM2        ;RETURN
                        ;
                        ; USE HANDLER ADDRESS TABLE AND COMMAND TABLE TO GET VECTOR
E647  B91B03    COM1:   LDA     HATABS+1,Y  ;GET LOW BYTE OF ADDRESS
E64A  852C              STA     ICSPRZ      ;AND SAVE IN POINTER
E64C  B91C03            LDA     HATABS+2,Y  ;GET HI BYTE OF ADDRESS
E64F  852D              STA     ICSPRZ+1
E651  A417              LDY     ICCOMT      ;GET COMMAND CODE
E653  B9C6E6            LDA     COMTAB-3,Y  ;GET COMMAND OFFSET
E656  A8                TAY
E657  B12C              LDA     (ICSPRZ),Y  ;GET LOW BYTE OF VECTOR FROM
E659  AA                TAX                 ;HANDLER ITSELF AND SAVE
E65A  C8                INY
E65B  B12C              LDA     (ICSPRZ),Y  ;GET HI BYTE OF VECTOR
E65D  852D              STA     ICSPRZ+1
E65F  862C              STX     ICSPRZ      ;SET LO BYTE
E661  18                CLC                 ;SHOW NO ERROR
E662  60        COM2:   RTS
                        ;
                        ;
                        ; DECBFL -- DECREMENT BUFFER LENGTH DOUBLE BYTE
                        ; Z FLAG = 0 ON RETURN IF LENGTH = 0 AFTER DECREMENT
E663  C628      DECBFL: DEC     ICBLLZ      ;DECREMENT LOW BYTE
E665  A528              LDA     ICBLLZ      ;CHECK IT
E667  C9FF              CMP     #$FF        ;DID IT GO BELOW?
E669  D002 ^E66D        BNE     DECBF1      ;NO
E66B  C629              DEC     ICBLLZ+1    ;DECREMENT HI BYTE
E66D  0529      DECBF1: ORA     ICBLLZ+1    ;SET Z IF BOTH ARE ZERO
E66F  60                RTS
                        ;
                        ;
                        ; INCBFP -- INCREMENT WORKING BUFFER POINTER
E670  E624      INCBFP: INC     ICBALZ      ;BUMP LOW BYTE
E672  D002 ^E676        BNE     INCBF1      ;GO IF NOT ZERO
E674  E625              INC     ICBALZ+1    ;ELSE, BUMP HI BYTE
E676  60        INCBF1: RTS
                        ;
                        ;
                        ; SUBBFL -- SET BUFFER LENGTH = BUFFER LENGTH - WORKING BYTE COUNT
E677  A62E      SUBBFL: LDX     ICIDNO      ;GET IOCB INDEX
E679  38                SEC
E67A  BD4803            LDA     ICBLL,X     ;GET LOW BYTE OF INITIAL LENGTH
E67D  E528              SBC     ICBLLZ      ;SUBTRACT FINAL LOW BYTE
E67F  8528              STA     ICBLLZ      ;AND SAVE BACK
E681  BD4903            LDA     ICBLH,X     ;GET HI BYTE
E684  E529              SBC     ICBLLZ+1
```

```
E686  8529              STA     ICBLHZ
E688  60                RTS
                ;
                ;
                ; GOHAND -- GO INDIRECT TO A DEVICE HANDLER
                ; Y= STATUS ON RETURN, N FLAG=1 IF ERROR ON RETURN
E689  A092      GOHAND: LDY     #FNCNOT     ;PREPARE NO FUNCTION STATUS FOR HANDLER RTS
E68B  2093E6            JSR     CIJUMP      ;USE THE INDIRECT JUMP
E68E  8423              STY     ICSTAZ      ;SAVE STATUS
E690  C000              CPY     #0          ;AND SET N FLAG
E692  60                RTS
                ;
                ; INDIRECT JUMP TO HANDLER BY PAUL'S METHOD
E693  AA        CIJUMP: TAX                 ;SAVE A
E694  A52D              LDA     ICSPRZ+1    ;GET JUMP ADDRESS HI BYTE
E696  48                PHA                 ;PUT ON STACK
E697  A52C              LDA     ICSPRZ      ;GET JUMP ADDRESS LO BYTE
E699  48                PHA                 ;PUT ON STACK
E69A  8A                TXA                 ;RESTORE A
E69B  A62E              LDX     ICIDNO      ;GET IOCB INDEX
E69D  60                RTS                 ;GO TO HANDLER INDIRECTLY
```

```
                        ;
                        ; DEVSRC -- DEVICE SEARCH, FIND DEVICE IN HANDLER ADDRESS TABLE
                        ;
                        ; LOOP TO FIND DEVICE
E69E  A000          DEVSRC: LDY     #0
E6A0  B124                  LDA     (ICBALZ),Y  ;GET DEVICE NAME FROM USER
E6A2  F00C ^E6B0            BEQ     CIERR2
E6A4  A021                  LDY     #MAXDEV     ;INITIAL COMPARE INDEX
E6A6  D91A03        DEVS1:  CMP     HATABS,Y    ;IS THIS THE DEVICE?
E6A9  F00A ^E6B5            BEQ     DEVS2       ;YES
E6AB  88                    DEY
E6AC  88                    DEY                 ;ELSE, POINT TO NEXT DEVICE NAME
E6AD  88                    DEY
E6AE  10F6 ^E6A6            BPL     DEVS1       ;CONTINUE FOR ALL DEVICES
                        ;
                        ; NO DEVICE FOUND, DECLARE NON-EXISTENT DEVICE ERROR
E6B0  A082          CIERR2: LDY     #NONDEV     ;ERROR CODE
E6B2  38                    SEC                 ;SHOW ERROR
E6B3  B013 ^E6C8            BCS     DEVS4       ;AND RETURN
                        ;
                        ; FOUND DEVICE, SET ICHID,ICDNO, AND INIT DEVICE
E6B5  98            DEVS2:  TYA
E6B6  8520                  STA     ICHIDZ      ;SAVE HANDLER INDEX
E6B8  38                    SEC
E6B9  A001                  LDY     #1
E6BB  B124                  LDA     (ICBALZ),Y  ;GET DEVICE NUMBER (DRIVE NUMBER)
E6BD  E930                  SBC     #ASCZER     ;SUBTRACT ASCII ZERO
E6BF  C90A                  CMP     #$A         ;IS NUMBER IN RANGE?
E6C1  9002 ^E6C5            BCC     DEVS3       ;YES
E6C3  A901                  LDA     #1          ;NO, DEFAULT TO ONE
E6C5  8521          DEVS3:  STA     ICDNOZ      ;SAVE DEVICE NUMBER
E6C7  18                    CLC                 ;SHOW NO ERROR
                        ;
                        ; RETURN
E6C8  60            DEVS4:  RTS
```

```
                        ;
                        ;
                        ; CIO ROM TABLES
                        ;
                        ; COMMAND TABLE
                        ; MAPS EACH COMMAND TO OFFSET FOR APPROPRIATE VECTOR IN HANDLER
E6C9  0004040404  COMTAB: DB      0,4,4,4,4,6,6,6,6,2,8,10
      = 022F      LENGTH  =       *-CIOINT
      = E6D5      CRNTP1  =       *
E6D5  = 0014              ORG     $14
0014  00          CIOSPR: DB      INTORG-CRNTP1 ;^GCIOL IS TOO LONG
```

```
                ;LIVES ON DK1:INTHV.SRC
      = 0006    SRTIM2  =       6           ;SECOND REPEAT INTERVAL
                ;
                ; THIS IS TO MAKE DOS 2 WORK WHICH USED AN ABSOLUTE ADDRESS
                ;
0015  = E912            ORG     $E912
E912  4CEDE8            JMP     SETVBL
E915  = E45C            ORG     SETVBV
E45C  4CEDE8            JMP     SETVBL
E45F  4CAEE7            JMP     SYSVBL
E462  4C05E9            JMP     XITVBL
E465  = E46B            ORG     INTINV
E46B  4CD5E6            JMP     IHINIT
                ;
E46E  = E480            ORG     VCTABL+INTABS-VDSLST
                ;
E480  90E7              DW      SYRTI       ;VDSLST
E482  8FE7              DW      SYIRQB      ;VPRCED
E484  8FE7              DW      SYIRQB      ;VINTER
E486  8FE7              DW      SYIRQB      ;VBREAK
                ;
E488  = 0008            DS      8
E490  8FE7              DW      SYIRQB      ;VTIMR1
E492  8FE7              DW      SYIRQB      ;VTIMR2
E494  8FE7              DW      SYIRQB      ;VTIMR4
E496  06E7              DW      SYIRQ       ;VIMIRQ
E498  0000000000        DW      0,0,0,0,0   ;CDTMV1-4
E4A2  AEE7              DW      SYSVBL      ;VVBLKI
E4A4  05E9              DW      XITVBL      ;VVBLKD
                ;
E4A6  = 900C            ORG     $900C
                ;
900C  A9E6              LDA     #PIRQH      ;SET UP RAM VECTORS FOR LINBUG VERSION
900E  8DF9FF            STA     $FFF9
9011  A9F3              LDA     #PIRQL
9013  8DF8FF            STA     $FFF8
9016  A9E7              LDA     #PNMIH
9018  8DFBFF            STA     $FFFB
901B  A991              LDA     #PNMIL
901D  8DFAFF            STA     $FFFA
9020  60                RTS
```

```
                        ;
                        ; IRQ HANDLER
                        ;
                        ; JUMP THRU IMMEDIATE IRQ VECTOR, WHICH ORDINARILY POINTS TO
                        ; SYSTEM IRQ: DETERMINE & CLEAR CAUSE, JUMP THRU SOFTWARE VECTOR.
                        ;
9021  = E6D5                    ORG     INTORG
E6D5  A940              IHINIT: LDA     #$40            ;VBL ON BUF DLIST OFF***FOR NOW***
E6D7  8D0ED4                    STA     NMIEN           ;ENABLE DISPLAY LIST, VERTICAL BLANK
E6DA  A938                      LDA     #$38            ;LOOK AT DATA DIRECTION REGISTERS IN PIA
E6DC  8D02D3                    STA     PACTL
E6DF  8D03D3                    STA     PBCTL
E6E2  A900                      LDA     #0              ;MAKE ALL INPUTS
E6E4  8D00D3                    STA     PORTA
E6E7  8D01D3                    STA     PORTB
E6EA  A93C                      LDA     #$3C            ;BACK TO PORTS
E6EC  8D02D3                    STA     PACTL
E6EF  8D03D3                    STA     PBCTL
E6F2  60                        RTS
E6F3  6C1602            PIRQ:   JMP     (VIMIRQ)
E6F6  80                CMPTAB: DB      $80             ;BREAK KEY
E6F7  40                        DB      $40             ;KEY STROKE
E6F8  04                        DB      $04             ;TIMER 4
E6F9  02                        DB      $02             ;TIMER 2
E6FA  01                        DB      $01             ;TIMER 1
E6FB  08                        DB      $08             ;SERIAL OUT COMPLETE
E6FC  10                        DB      $10             ;SERIAL OUT READY
E6FD  20                        DB      $20             ;SERIAL  IN READY

                        ; THIS IS A TABLE OF OFFSETS INTO PAGE 2.  THEY POINT TO
E6FE  36                ADRTAB: DB      BRKKY-INTABS
E6FF  08                        DB      VKEYBD-INTABS
E700  14                        DB      VTIMR4-INTABS
E701  12                        DB      VTIMR2-INTABS
E702  10                        DB      VTIMR1-INTABS
E703  0E                        DB      VSEROC-INTABS
E704  0C                        DB      VSEROR-INTABS
E705  0A                        DB      VSERIN-INTABS

E706  48                SYIRQ:  PHA                     ;SAVE ACCUMULATOR
E707  AD0ED2                    LDA     IRQST           ; CHECK FOR SERIAL IN
E70A  2920                      AND     #$20
E70C  D00D ^E71B                BNE     SYIRQ2
E70E  A9DF                      LDA     #$DF            ; MASK ALL OTHERS
E710  8D0ED2                    STA     IRQEN
E713  A510                      LDA     POKMSK
E715  8D0ED2                    STA     IRQEN
E718  6C0A02                    JMP     (VSERIN)
E71B  8A                SYIRQ2: TXA                     ;PUT X INTO ACC
E71C  48                        PHA                     ;SAVE X ONTO STACK
E71D  A206                      LDX     #$6             ;START WITH SIX OFFSET
E71F  BDF6E6            LOOPM:  LDA     CMPTAB,X        ;LOAD MASK
E722  E005                      CPX     #5              ;CHECK TO SEE IF COMPLETE IS SET
E724  D004 ^E72A                BNE     LOOPM2
E726  2510                      AND     POKMSK          ;IS THIS INTERUPT ENABLED?
E728  F005 ^E72F                BEQ     LL
E72A  2C0ED2            LOOPM2: BIT     IRQST           ; IS IT THE INTERUPT?
```

```
E72D  F006 ^E735          BEQ     JMPP
E72F  CA          LL:     DEX                 ;NO DEC X AND TRY NEXT MASK
E730  10ED ^E71F          BPL     LOOPM       ;IF NOT NEG GOTO LOOPM
E732  4C62E7              JMP     SYIRQ8      ;DONE BUT NO INTERUPT
E735  49FF        JMPP:   EOR     #$FF        ;COMPLEMENT MASK
E737  8D0ED2              STA     IRQEN       ;ENABLE ALL OTHERS
E73A  A510                LDA     POKMSK      ; GET POKE MASK
E73C  8D0ED2              STA     IRQEN       ; ENABLE THOSE IN POKE MASK
E73F  BDFEE6              LDA     ADRTAB,X
E742  AA                  TAX
E743  BD0002              LDA     INTABS,X    ; GET ADDRESS LOW PART
E746  8D8C02              STA     JVECK       ; PUT IN VECTOR
E749  BD0102              LDA     INTABS+1,X  ; GET ADDRESS HIGH PART
E74C  8D8D02              STA     JVECK+1     ; PUT IN VECTOR HIGH PART
E74F  68                  PLA                 ; PULL X REGISTER FROM STACK
E750  AA                  TAX                 ; PUT IT INTO X
E751  6C8C02              JMP     (JVECK)     ; JUMP TO THE PROPER ROUTINE
E754  A900        BRKKY2: LDA     #0          ; BREAK KEY ROUTINE
E756  8511                STA     BRKKEY      ; SET BREAK KEY FLAG
E758  8DFF02              STA     SSFLAG      ; START/STOP FLAG
E75B  8DF002              STA     CRSINH      ; CURSOR INHIBIT
E75E  854D                STA     ATRACT      ; TURN OFF ATRACT MODE
E760  68                  PLA
E761  40                  RTI                 ;EXIT FROM INT
E762  68          SYIRQ8: PLA
E763  AA                  TAX
E764  2C02D3              BIT     PACTL       ;PROCEED ***I GUESS***
E767  1006 ^E76F          BPL     SYIRQ9
E769  AD00D3              LDA     PORTA       ;CLEAR INT STATUS BIT
E76C  6C0202              JMP     (VPRCED)
E76F  2C03D3      SYIRQ9: BIT     PBCTL       ;INTERRUPT ***I GUESS***
E772  1006 ^E77A          BPL     SYIRQA
E774  AD01D3              LDA     PORTB       ;CLEAR INT STATUS
E777  6C0402              JMP     (VINTER)
E77A  68          SYIRQA: PLA
E77B  8D8C02              STA     JVECK
E77E  68                  PLA
E77F  48                  PHA
E780  2910                AND     #$10        ;B BIT OF P REGISTER
E782  F007 ^E78B          BEQ     SYRTI2
E784  AD8C02              LDA     JVECK
E787  48                  PHA
E788  6C0602              JMP     (VBREAK)
E78B  AD8C02      SYRTI2: LDA     JVECK
E78E  48                  PHA
E78F  68          SYIRQB: PLA
E790  40          SYRTI:  RTI                 ;UNIDENTIFIED INTERRUPT, JUST RETURN.
```

```
                        ;
                        ; NMI HANDLER
                        ;
                        ; DETERMINE CAUSE AND JUMP THRU VECTOR
                        ;
E791  2C0FD4      PNMI:   BIT     NMIST
E794  1003 ^E799          BPL     PNMI1       ;SEE IF DISPLAY LIST
E796  6C0002              JMP     (VDSLST)
E799  48          PNMI1:  PHA
E79A  AD0FD4              LDA     NMIST
E79D  2920                AND     #$20        ;SEE IF RESET
E79F  F003 ^E7A4          BEQ     *+5
E7A1  4C74E4              JMP     WARMSV      ;GO THRU WARM START JUMP
E7A4  8A                  TXA                 ;SAVE REGISTERS
E7A5  48                  PHA
E7A6  98                  TYA
E7A7  48                  PHA
E7A8  8D0FD4              STA     NMIRES      ;RESET INTERRUPT STATUS
E7AB  6C2202              JMP     (VVBLKI)    ;JUMP THRU VECTOR
```

```
                        ;
                        ; SYSTEM VBLANK ROUTINE
                        ;
                        ; INC FRAME COUNTER. PROCESS COUNTDOWN TIMERS. EXIT IF I WAS SET, CLEAR I.
                        ; SET DLISTL, DLISTH, DMACTL FROM RAM CELLS. DO SOFTWARE REPEAT.
                        ;
E7AE  E614        SYSVBL: INC     RTCLOK+2    ;INC FRAME COUNTER
E7B0  D008 ^E7BA          BNE     SYSVB1
E7B2  E64D                INC     ATRACT      ;INCREMENT ATRACT (CAUSES ATRACT WHEN MINUS)
E7B4  E613                INC     RTCLOK+1
E7B6  D002 ^E7BA          BNE     SYSVB1
E7B8  E612                INC     RTCLOK
E7BA  A9FE        SYSVB1: LDA     #$FE        ;[ATRACT] SET DARK MASK TO NORMAL
E7BC  A200                LDX     #0          ;SET COLRSH TO NORMAL
E7BE  A44D                LDY     ATRACT      ;TEST ATRACT FOR NEGATIVE
E7C0  1006 ^E7C8          BPL     VBATRA      ;WHILE POSITIVE, DONT GO INTO ATRACT
E7C2  854D                STA     ATRACT      ;IN ATRACT, SO STAY BY STA $FE
E7C4  A613                LDX     RTCLOK+1    ;COLOR SHIFT FOLLOWS RTCLOK+1
E7C6  A9F6                LDA     #$F6        ;SET DARK MASK TO DARK
E7C8  854E        VBATRA: STA     DRKMSK
E7CA  864F                STX     COLRSH
E7CC  A200                LDX     #0          ;POINT TO TIMER1
E7CE  20D0E8              JSR     DCTIMR      ;GO DECREMENT TIMER1
E7D1  D003 ^E7D6          BNE     SYSVB2      ;BRANCH IF STILL COUNTING
E7D3  20CAE8              JSR     JTIMR1      ;GO JUMP TO ROUTINE
E7D6  A542        SYSVB2: LDA     CRITIC
E7D8  D008 ^E7E2          BNE     XXIT        ;GO IF CRITICAL SET
E7DA  BA                  TSX                 ;SEE IF I WAS SET
E7DB  BD0401              LDA     $104,X      ;GET STACKED P
E7DE  2904                AND     #$04        ;I BIT
E7E0  F003 ^E7E5          BEQ     SYSVB3      ;BRANCH IF OK
E7E2  4C05E9      XXIT:   JMP     XITVBL      ;I WAS SET, EXIT
E7E5  AD0DD4      SYSVB3: LDA     PENV
E7E8  8D3502              STA     LPENV
E7EB  AD0CD4              LDA     PENH
E7EE  8D3402              STA     LPENH
E7F1  AD3102              LDA     SDLSTH
E7F4  8D03D4              STA     DLISTH
E7F7  AD3002              LDA     SDLSTL
E7FA  8D02D4              STA     DLISTL
E7FD  AD2F02              LDA     SDMCTL
E800  8D00D4              STA     DMACTL
E803  AD6F02              LDA     GPRIOR      ;GLOBAL PRIOR
E806  8D1BD0              STA     PRIOR
E809  A208                LDX     #$08        ;TURN OFF KEYBOARD SPEAKER
E80B  8E1FD0              STX     CONSOL
E80E  58          SCOLLP: CLI                 ;DISABLE INTERUPTS
E80F  BDC002              LDA     PCOLR0,X    ;LOAD COLOR REGISTERS FROM RAM
E812  454F                EOR     COLRSH      ;DO COLOR SHIFT
E814  254E                AND     DRKMSK      ;AND DARK ATRACT
E816  9D12D0              STA     COLPM0,X
E819  CA                  DEX
E81A  10F2 ^E80E          BPL     SCOLLP
E81C  ADF402              LDA     CHBAS
E81F  8D09D4              STA     CHBASE
E822  ADF302              LDA     CHACT
E825  8D01D4              STA     CHACTL
```

```
E828  A202              LDX     #2          ;POINT TO TIMER 2
E82A  20D0E8            JSR     DCTIMR
E82D  D003 ^E832        BNE     SYSVB4      ;IF DIDNT GO ZERO
E82F  20CDE8            JSR     JTIMR2      ;GO JUMP TO TIMER2 ROUTINE
E832  A202      SYSVB4: LDX     #2          ;RESTORE X
E834  E8        SYSVBB: INX
E835  E8                INX
E836  BD1802            LDA     CDTMV1,X
E839  1D1902            ORA     CDTMV1+1,X
E83C  F006 ^E844        BEQ     SYSVBA
E83E  20D0E8            JSR     DCTIMR      ;DECREMENT AND SET FLAG IF NONZERO
E841  9D2602            STA     CDTMF3-4,X
E844  E008      SYSVBA: CPX     #8          ;SEE IF DONE ALL 3
E846  D0EC ^E834        BNE     SYSVBB      ;LOOP
                ; CHECK DEBOUNCE COUNTER
E848  AD0FD2            LDA     SKSTAT
E84B  2904              AND     #$04        ;KEY DOWN BIT
E84D  F008 ^E857        BEQ     SYVB6A      ;IF KEY DOWN
                ; KEY UP SO COUNT IT
E84F  ADF102            LDA     KEYDEL      ;KEY DELAY COUNTER
E852  F003 ^E857        BEQ     SYVB6A      ;IF COUNTED DOWN ALREADY
E854  CEF102            DEC     KEYDEL      ;COUNT IT
                ; CHECK SOFTWARE REPEAT TIMER
E857  AD2B02    SYVB6A: LDA     SRTIMR
E85A  F017 ^E873        BEQ     SYSVB7      ;DOESN'T COUNT
E85C  AD0FD2            LDA     SKSTAT
E85F  2904              AND     #$04        ;CHECK KEY DOWN BIT
E861  D060 ^E8C3        BNE     SYSVB6      ;BRANCH IF NO LONGER DOWN
E863  CE2B02            DEC     SRTIMR      ;COUNT FRAME OF KEY DOWN
E866  D00B ^E873        BNE     SYSVB7      ;BRANCH IF NOT RUN OUT
                ; TIMER RAN OUT - RESET AND SIMULATE KEYBOARD IRQ
E868  A906              LDA     #SRTIM2     ;TIMER VALUE
E86A  8D2B02            STA     SRTIMR      ;SET TIMER
E86D  AD09D2            LDA     KBCODE      ;GET THE KEY
E870  8DFC02            STA     CH          ;PUT INTO CH
                ; READ GAME CONTROLLERS
E873  A001      SYSVB7: LDY     #1
E875  A203              LDX     #3
E877  B900D3    STLOOP: LDA     PORTA,Y
E87A  4A                LSR     A
E87B  4A                LSR     A
E87C  4A                LSR     A
E87D  4A                LSR     A
E87E  9D7802            STA     STICK0,X    ;STORE JOYSTICK
E881  CA                DEX
E882  B900D3            LDA     PORTA,Y
E885  290F              AND     #$F
E887  9D7802            STA     STICK0,X    ;STORE JOYSTICK
E88A  CA                DEX
E88B  88                DEY
E88C  10E9 ^E877        BPL     STLOOP
                ;
E88E  A203              LDX     #3
E890  BD10D0    STRL:   LDA     TRIG0,X     ;MOVE JOYSTICK TRIGGERS
E893  9D8402            STA     STRIG0,X
E896  BD00D2            LDA     POT0,X      ;MOVE POT VALUES
E899  9D7002            STA     PADDL0,X
```

```
E89C  BD04D2              LDA     POT4,X
E89F  9D7402              STA     PADDL4,X
E8A2  CA                  DEX
E8A3  10EB ^E890          BPL     STRL
E8A5  8D0BD2              STA     POTGO       ;START POTS FOR NEXT TIME
                  ;
E8A8  A206                LDX     #6
E8AA  A003                LDY     #3
E8AC  B97802      PTRLP:  LDA     STICK0,Y    ;TRANSFER BITS FROM JOYSTICKS
E8AF  4A                  LSR     A           ;TO PADDLE TRIGGERS
E8B0  4A                  LSR     A
E8B1  4A                  LSR     A
E8B2  9D7D02              STA     PTRIG1,X
E8B5  A900                LDA     #0
E8B7  2A                  ROL     A
E8B8  9D7C02              STA     PTRIG0,X
E8BB  CA                  DEX
E8BC  CA                  DEX
E8BD  88                  DEY
E8BE  10EC ^E8AC          BPL     PTRLP
                  ;
E8C0  6C2402              JMP     (VVBLKD)    ;GO TO DEFERRED VBLANK ROUTINE
      = 00E8      SV7H    =       HIGH SYSVB7
      = 0073      SV7L    =       LOW SYSVB7
E8C3  A900        SYSVB6: LDA     #0
E8C5  8D2B02              STA     SRTIMR      ;ZERO TIMER
E8C8  F0A9 ^E873          BEQ     SYSVB7      ;UNCOND
E8CA  6C2602      JTIMR1: JMP     (CDTMA1)
E8CD  6C2802      JTIMR2: JMP     (CDTMA2)
                  ;
                  ; SUBROUTINE TO DECREMENT A COUNTDOWN TIMER
                  ; ENTRY X=OFFSET FROM TIMER 1
                  ; EXIT A,P=ZERO IF WENT ZERO, FF OTHERWISE
                  ;
E8D0  BC1802      DCTIMR: LDY     CDTMV1,X    ;LO BYTE
E8D3  D008 ^E8DD          BNE     DCTIM1      ;NONZERO, GO DEC IT
E8D5  BC1902              LDY     CDTMV1+1,X  ;SEE IF BOTH ZERO
E8D8  F010 ^E8EA          BEQ     DCTXF       ;YES, EXIT NONZERO
E8DA  DE1902              DEC     CDTMV1+1,X  ;DEC HI BYTE
E8DD  DE1802      DCTIM1: DEC     CDTMV1,X    ;DEC LO BYTE
E8E0  D008 ^E8EA          BNE     DCTXF
E8E2  BC1902              LDY     CDTMV1+1,X
E8E5  D003 ^E8EA          BNE     DCTXF
E8E7  A900                LDA     #0          ;WENT ZERO, RETURN ZERO
E8E9  60                  RTS
E8EA  A9FF        DCTXF:  LDA     #$FF        ;RETURN NONZERO
E8EC  60                  RTS
```

```
                        ;
                        ; 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
                        ;
E8ED  0A        SETVBL: ASL     A           ;MUL BY 2
E8EE  8D2D02            STA     INTEMP
E8F1  8A                TXA
E8F2  A205              LDX     #5
E8F4  8D0AD4            STA     WSYNC       ;WASTE 20 CPU CYCLES
E8F7  CA        SETLOP: DEX                 ;TO ALOWD VBLANK TO HAPPEN
E8F8  D0FD ^E8F7        BNE     SETLOP      ;IF THIS IS LINE "7C"
E8FA  AE2D02            LDX     INTEMP
E8FD  9D1702            STA     CDTMV1-1,X
E900  98                TYA
E901  9D1602            STA     CDTMV1-2,X
E904  60                RTS
                        ;
                        ; EXIT FROM VERTICAL BLANK
                        ;
E905  68        XITVBL: PLA                 ;UNSTACK Y
E906  A8                TAY
E907  68                PLA                 ;UNSTACK X
E908  AA                TAX
E909  68                PLA                 ;UNSTACK A
E90A  40                RTI                 ;AND GO BACK FROM WHENCE.
      = 00E6    PIRQH   =       HIGH PIRQ
      = 00F3    PIRQL   =       LOW PIRQ
      = 00E7    PNMIH   =       HIGH PNMI
      = 0091    PNMIL   =       LOW PNMI
                ; SPARE BYTE OR MODULE TOO LONG FLAG
      = E90B    CRNTP2  =       *
E90B  = 0014            ORG     $14
0014  39        INTSPR: DB      SIOORG-CRNTP2 ;^GINTHV IS TOO LONG
```

```
;       COLLEEN OPERATING SYSTEM
;
;       SIO  ( SERIAL BUS INPUT/OUTPUT CONTROLLER )
;       WITH SOFTWARE BAUD RATE CORRECTION ON CASSETTE
;
;
;               AL MILLER       3-APR-79
;
;
; 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.
;
;
;
;
```

```
                        ; EQUATES
                        ;
                        ; DCD DEVICE BUS ID NUMBERS
= 0030                  FLOPPY  =       $30
                        ;PRINTR =       $40
                        ;CASSET =       $60         ;!!!!! *****
= 0060                  CASET   =       $60         ;!!!!! *****
                        ;
                        ;
                        ;BUS COMMANDS
                        ;
= 0052                  READ    =       'R
= 0057                  WRITE   =       'W
                        ;STATIS = 'S
                        ;FORMAT = '!
                        ;
                        ;
                        ; COMMAND AUX BYTES
                        ;
= 0053                  SIDWAY  =       'S'         ;PRINT 16 CHARACTERS SIDEWAYS
= 004E                  NORMAL  =       'N'         ;PRINT 40 CHARACTERS NORMALLY
= 0044                  DOUBLE  =       'D'         ;PRINT 20 CHARACTERS DOUBLE WIDE
= 0050                  PLOT    =       'P'         ;PLOT MODE
                        ;
                        ;
                        ; BUS RESPONSES
                        ;
= 0041                  ACK     =       'A'         ;DEVICE ACKNOWLEDGES INFORMATION
= 004E                  NACK    =       'N'         ;DEVICE DID NOT UNDERSTAND
= 0043                  COMPLT  =       'C'         ;DEVICE SUCCESSFULLY COMPLETED OPERATION
= 0045                  ERROR   =       'E'         ;DEVICE INCURRED AN ERROR IN AN ATTEMPTED OPERATION
                        ;
                        ;
                        ; MISCELLANEOUS EQUATES
                        ;
= 0028                  B192LO  =       $28         ;19200 BAUD RATE POKEY COUNTER VALUES (LO BYTE)
= 0000                  B192HI  =       $00         ;(HI BYTE)
= 00CC                  B600LO  =       $CC         ;600 BAUD (LO BYTE)
= 0005                  B600HI  =       $05         ;(HI BYTE)
= 0005                  HITONE  =       $05         ;FSK HI FREQ POKEY COUNTE VALUE (5326 HZ)
= 0007                  LOTONE  =       $07         ;FSK LO FREQ POKEY COUNTER VALUE (3995 HZ)
                        ;
= 0000                          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
= FFFF                          IF      PALFLG-1
= 00B4                  WIRGLO  =       180         ;WRITE INTER RECORD GAP (IN 1/60 SEC)
= 0078                  RIRGLO  =       120         ;READ INTER RECORD GAP (IN 1/60 SEC)
= 000F                  WSIRG   =       15          ;SHORT WRITE INTER RECORD GAP
= 000A                  RSIRG   =       10          ;SHORT READ INTER RECORD GAP
                                ENDIF
= 0000                  WIRGHI  =       0
= 0000                  RIRGHI  =       0
                        ;
```

```
= 0034      NCOMLO  =       $34         ;PIA COMMAND TO LOWER NOT COMMAND LINE
= 003C      NCOMHI  =       $3C         ;PIA COMMAND TO RAISE NOT COMMAND LINE
= 0034      MOTRGO  =       $34         ;PIA COMMAND TO TURN ON CASSETTE MOTOR
= 003C      MOTRST  =       $3C         ;PIA COMMAND TO TURN OFF MOTOR
            ;
= 0002      TEMPHI  =       HIGH TEMP   ;ADDRESS OF TEMP CELL (HI BYTE)
= 003E      TEMPLO  =       LOW TEMP    ;(LO BYTE)
= 0002      CBUFHI  =       HIGH CDEVIC ;ADDRESS OF COMMAND BUFFER (HI BYTE)
= 003A      CBUFLO  =       LOW CDEVIC  ;(LO BYTE)
            ;
= 000D      CRETRI  =       13          ;NUMBER OF COMMAND FRAME RETRIES
= 0001      DRETRI  =       1           ;NUMBER OF DEVICE RETRIES
= 0002      CTIMLO  =       2           ;COMMAND FRAME ACK TIME OUT (LO BYTE)
= 0000      CTIMHI  =       0           ;COMMAND FRAME ACK TIME OUT (HI BYTE)
            ;
            ;
            ;JTADRH  =       HIGH JTIMER ;HI BYTE OF JUMP TIMER ROUTINE ADDR
                                        ;"MOVED TO LINE 1427"
            ;JTADRL  =       LOW JTIMER  ;"MOVED TO LINE 1428"
            ;
```

```
                        ;       SIO
                        ;
                        ;
0015  = E459            SIOV    ORG     SIOV
E459  4C59E9                    JMP     SIO         ;SIO ENTRY POINT
                        ;
E45C  = E465            SIOINV  ORG     SIOINV
E465  4C44E9                    JMP     SIOINT      ;SIO INITIALIZATION ENTRY POINT
                        ;
E468  = E468            SENDEV  ORG     SENDEV
E468  4CF2EB                    JMP     SENDEN      ;SEND ENABLE ENTRY POINT
                        ;
E46B  = E48A                    ORG     VCTABL-INTABS+VSERIN
                        ;
E48A  0FEB                      DW      ISRSIR      ;VSERIN
E48C  90EA                      DW      ISRODN      ;VSEROR
E48E  CFEA                      DW      ISRTD       ;VSEROC
                        ;
                        ;
                        ;
E490  = E944                    ORG     SIOORG
                        ;
                        ; SIO INITIALIZATION SUBROUTINE
                        ;
E944  A93C              SIOINT: LDA     #MOTRST
E946  8D02D3                    STA     PACTL       ;TURN OFF MOTOR
                        ;
E949  A93C                      LDA     #NCOMHI
E94B  8D03D3                    STA     PBCTL       ;RAISE NOT COMMAND LINE
                        ;
                        ;
E94E  A903                      LDA     #3
E950  8D3202                    STA     SSKCTL      ;GET POKEY OUT OF INITIALIZE MODE
E953  8541                      STA     SOUNDR      ;INIT POKE ADDRESS FOR QUIET I/O
E955  8D0FD2                    STA     SKCTL
                        ;
                        ;
E958  60                        RTS                 ;RETURN
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
E959  BA                SIO:    TSX
E95A  8E1803                    STX     STACKP      ;SAVE STACK POINTER
E95D  A901                      LDA     #1
E95F  8542                      STA     CRITIC
                        ;
E961  AD0003                    LDA     DDEVIC
E964  C960                      CMP     #CASET
E966  D003 ^E96B                BNE     NOTCST      ;BRANCH IF NOT CASSETTE
E968  4C80EB                    JMP     CASENT      ;OTHERWISE JUMP TO CASSETTE ENTER
                        ;
                        ; ALL DEVICES EXCEPT CASSETTE ARE INTELLIGENT
                        ;
E96B  A900              NOTCST: LDA     #0
```

```
E96D  8D0F03              STA     CASFLG      ;INIT CASSETTE FLAG TO NO CASSETTE
                  ;
E970  A901                LDA     #DRETRI     ;SET NUMBER OF DEVICE RETRIES
E972  8537                STA     DRETRY
E974  A90D        COMMND: LDA     #CRETRI     ;SET NUMBER OF COMMAND FRAME RETRIES
E976  8536                STA     CRETRY
                  ;
                  ; SEND A COMMAND FRAME
                  ;
E978  A928        COMFRM: LDA     #B192LO     ;SET BAUD RATE TO 19200
E97A  8D04D2              STA     AUDF3
E97D  A900                LDA     #B192HI
E97F  8D06D2              STA     AUDF4
                  ;
E982  18                  CLC                 ;SET UP COMMAND BUFFER
E983  AD0003              LDA     DDEVIC
E986  6D0103              ADC     DUNIT
E989  69FF                ADC     #$FF        ;SUBTRACT 1
E98B  8D3A02              STA     CDEVIC      ;SET BUS ID NUMBER
                  ;
E98E  AD0203              LDA     DCOMND
E991  8D3B02              STA     CCOMND      ;SET BUS COMMAND
                  ;
E994  AD0A03              LDA     DAUX1       ;STORE COMMAND FRAME AUX BYTES 1 AND 2
E997  8D3C02              STA     CAUX1
E99A  AD0B03              LDA     DAUX2
E99D  8D3D02              STA     CAUX2       ;DONE SETTING UP COMMAND BUFFER
                  ;
E9A0  18                  CLC                 ;SET BUFFER POINTER TO COMMAND FRAME BUFFER
E9A1  A93A                LDA     #CBUFLO
E9A3  8532                STA     BUFRLO      ;AND BUFFER END ADDRESS
E9A5  6904                ADC     #4
E9A7  8534                STA     BFENLO
E9A9  A902                LDA     #CBUFHI
E9AB  8533                STA     BUFRHI
E9AD  8535                STA     BFENHI      ;DONE SETTING UP BUFFER POINTER
                  ;
E9AF  A934                LDA     #NCOMLO     ;LOWER NOT COMMAND LINE
E9B1  8D03D3              STA     PBCTL
                  ;
E9B4  208AEC              JSR     SENDIN      ;SEND THE COMMAND FRAME TO A SMART DEVICE
                  ;
E9B7  AD3F02              LDA     ERRFLG
E9BA  D003 ^E9BF          BNE     BADCOM      ;BRANCH IF AN ERROR RECEIVED
                  ;
E9BC  98                  TYA
E9BD  D007 ^E9C6          BNE     ACKREC      ;BRANCH IF ACK RECEIVED
                  ;
                  ;
E9BF  C636        BADCOM: DEC     CRETRY      ;A NACK OR TIME OUT OCCURED
E9C1  10B5 ^E978          BPL     COMFRM      ;SO BRANCH IF ANY RETRIES LEFT
                  ;
E9C3  4C06EA              JMP     DERR1       ;OTHERWISE, JUMP TO RETURN SECTION
                  ;
                  ;
E9C6  AD0303      ACKREC: LDA     DSTATS      ;ACK WAS RECEIVED
E9C9  100C ^E9D7          BPL     WATCOM      ;BRANCH TO WAIT FOR COMPLETE ,
```

```
                        ; IF THERE IS NO DATA TO BE SENT
                        ;
                        ;
                        ;
                        ; SEND A DATA FRAME TO PERIPHERAL
                        ;
E9CB  A90D                      LDA     #CRETRI     ;SET NUMBER OF RETRIES
E9CD  8536                      STA     CRETRY
                        ;
E9CF  206AEB                    JSR     LDPNTR      ;LOAD BUFFER POINTER WITH DCB INFORMATION
                        ;
E9D2  208AEC                    JSR     SENDIN      ;GO SEND THE DATA FRAME TO A SMART DEVICE
                        ;
E9D5  F0E8 ^E9BF                BEQ     BADCOM      ;BRANCH IF BAD
                        ;
                        ;
                        ;
                        ; WAIT FOR COMPLETE SIGNAL FROM PERIPHERAL
                        ;
E9D7  2075EC            WATCOM: JSR     STTMOT      ;SET DEVICE TIME OUT VALUES IN Y,X
                        ;
E9DA  A900                      LDA     #$00
E9DC  8D3F02                    STA     ERRFLG      ;CLEAR ERROR FLAG
                        ;
E9DF  209BEC                    JSR     WAITER      ;SET UP TIMER AND WAIT
E9E2  F012 ^E9F6                BEQ     DERR        ;BRANCH IF TIME OUT
                        ;
                        ;
                        ; DEVICE DID NOT TIME OUT
                        ;
E9E4  2C0303                    BIT     DSTATS
E9E7  7007 ^E9F0                BVS     MODATA      ;BRANCH IF MORE DATA FOLLOWS
                        ;
E9E9  AD3F02                    LDA     ERRFLG
E9EC  D018 ^EA06                BNE     DERR1       ;BRANCH IF AN ERROR OCCURRED
E9EE  F01D ^EA0D                BEQ     RETURN      ;OTHERWISE RETURN
                        ;
                        ;
                        ;
                        ;
                        ; RECEIVE A DATA FRAME FROM PERIPHERAL
                        ;
E9F0  206AEB            MODATA: JSR     LDPNTR      ;LOAD BUFFER POINTER WITH DCB INFORMATION
                        ;
E9F3  20E0EA                    JSR     RECEIV      ;GO RECEIVE A DATA FRAME
                        ;
E9F6  AD3F02            DERR:   LDA     ERRFLG
E9F9  F005 ^EA00                BEQ     NOTERR      ;BRANCH IF NO ERROR PRECEEDED DATA
                        ;
E9FB  AD1903                    LDA     TSTAT       ;GET TEMP STATUS
E9FE  8530                      STA     STATUS      ;STORE IN REAL STATUS
                        ;
                        ;
EA00  A530              NOTERR: LDA     STATUS
EA02  C901                      CMP     #SUCCES
EA04  F007 ^EA0D                BEQ     RETURN      ;BRANCH IF COMPLETELY SUCCESSFUL
                        ;
```

```
EA06  C637        DERR1:  DEC     DRETRY
EA08  3003 ^EA0D          BMI     RETURN      ;BRANCH IF OUT OF DEVICE RETRIES
                  ;
EA0A  4C74E9              JMP     COMMND      ;OTHERWISE, ONE MORE TIME
                  ;
                  ;
                  ;
                  ;
EA0D  205FEC      RETURN: JSR     SENDDS      ;DISABLE POKEY INTERRUPTS
EA10  A900                LDA     #0
EA12  8542                STA     CRITIC
EA14  A430                LDY     STATUS      ;RETURN STATUS IN Y
EA16  8C0303              STY     DSTATS      ;AND THE DCB STATUS WORD
EA19  60                  RTS     RETURN
                  ;
                  ;
                  ;
                  ;
                  ; WAIT SUBROUTINE
                  ;
                  ; WAITS FOR COMPLETE OR ACK
                  ; RETURNS Y=$FF IF SUCCESSFUL, Y=$00 IF NOT
                  ;
EA1A  A900        WAIT:   LDA     #$00
EA1C  8D3F02              STA     ERRFLG      ;CLEAR ERROR FLAG
                  ;
EA1F  18                  CLC                 ;LOAD BUFFER POINTER WITH ADDRESS
EA20  A93E                LDA     #TEMPLO     ;OF TEMPORARY RAM CELL
EA22  8532                STA     BUFRLO
EA24  6901                ADC     #1
EA26  8534                STA     BFENLO      ;ALSO SET BUFFER END +1 ADDRESS
EA28  A902                LDA     #TEMPHI
EA2A  8533                STA     BUFRHI
EA2C  8535                STA     BFENHI      ;DONE LOADING POINTER
                  ;
EA2E  A9FF                LDA     #$FF
EA30  853C                STA     NOCKSM      ;SET NO CHECKSUM FOLLOWS DATA FLAG
                  ;
EA32  20E0EA              JSR     RECEIV      ;GO RECEIVE A BYTE
                  ;
EA35  A0FF                LDY     #$FF        ;ASSUME SUCCESS
EA37  A530                LDA     STATUS
EA39  C901                CMP     #SUCCES
EA3B  D019 ^EA56          BNE     NWOK        ;BRANCH IF IT DID NOT WORK OK
                  ;
                  ;
                  ;
                  ;
EA3D  AD3E02      WOK:    LDA     TEMP        ;MAKE SURE THE BYTE SUCCESSFULLY RECEIVED
EA40  C941                CMP     #ACK        ;WAS ACTUALLY AN ACK OR COMPLETE
EA42  F021 ^EA65          BEQ     GOOD
EA44  C943                CMP     #COMPLT
EA46  F01D ^EA65          BEQ     GOOD
                  ;
EA48  C945                CMP     #ERROR
EA4A  D006 ^EA52          BNE     NOTDER      ;BRANCH IF DEVICE DID NOT SEND BACK
                  ; A DEVICE ERROR CODE
```

```
EA4C  A990              LDA     #DERROR
EA4E  8530              STA     STATUS      ;SET DEVICE ERROR STATUS
EA50  D004 ^EA56        BNE     NWOK
                ;
EA52  A98B      NOTDER: LDA     #DNACK      ;OTHERWISE SET NACK STATUS
EA54  8530              STA     STATUS
                ;
EA56  A530      NWOK:   LDA     STATUS
EA58  C98A              CMP     #TIMOUT
EA5A  F007 ^EA63        BEQ     BAD         ;BRANCH IF TIME OUT
                ;
EA5C  A9FF              LDA     #$FF
EA5E  8D3F02            STA     ERRFLG      ;SET SOME ERROR FLAG
EA61  D002 ^EA65        BNE     GOOD        ;RETURN WITH OUT SETTING Y = 0
                ;
EA63  A000      BAD:    LDY     #0
                ;
EA65  A530      GOOD:   LDA     STATUS
EA67  8D1903            STA     TSTAT
EA6A  60                RTS                 ;RETURN
                ;
                ;
                ;
                ;
                ;
                ; SEND SUBROUTINE
                ;
                ; SENDS A BUFFER OF BYTES OUT OVER THE SERIAL BUS
                ;
                ;
EA6B  A901      SEND:   LDA     #SUCCES     ;ASSUME SUCCESS
EA6D  8530              STA     STATUS
                ;
EA6F  20F2EB            JSR     SENDEN      ;ENABLE SENDING
                ;
EA72  A000              LDY     #0
EA74  8431              STY     CHKSUM      ;CLEAR CHECK SUM
EA76  843B              STY     CHKSNT      ;CHECKSUM SENT FLAG
EA78  843A              STY     XMTDON      ;TRANSMISSION DONE FLAG
                ;
                ;
EA7A  B132              LDA     (BUFRLO),Y  ;PUT FIRST BYTE FROM BUFFER
EA7C  8D0DD2            STA     SEROUT      ;INTO THE SERIAL OUTPUT REGISTER
                ;
                ;
EA7F  8531              STA     CHKSUM      ;PUT IT IN CHECKSUM
                ;
EA81  A511      NOTDON: LDA     BRKKEY
EA83  D003 ^EA88        BNE     NTBRK0
EA85  4CA0ED            JMP     BROKE       ;JUMP IF BREAK KEY PRESSED
                ;
EA88  A53A      NTBRK0: LDA     XMTDON      ;LOOP UNTIL TRANSMISSION IS DONE
EA8A  F0F5 ^EA81        BEQ     NOTDON
                ;
EA8C  205FEC            JSR     SENDDS      ;DISABLE SENDING
                ;
EA8F  60                RTS                 ;RETURN
```

```
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ; OUTPUT DATA NEEDED INTERRUPT SERVICE ROUTINE
                        ;
EA90  98        ISRODN: TYA
EA91  48                PHA                 ;SAVE Y REG ON STACK
                        ;
EA92  E632              INC     BUFRLO      ;INCREMENT BUFFER POINTER
EA94  D002 ^EA98        BNE     NOWRP0
EA96  E633              INC     BUFRHI
                        ;
EA98  A532      NOWRP0: LDA     BUFRLO      ;CHECK IF PAST END OF BUFFER
EA9A  C534              CMP     BFENLO
EA9C  A533              LDA     BUFRHI      ;HIGH PART
EA9E  E535              SBC     BFENHI
EAA0  901C ^EABE        BCC     NOTEND      ;BRANCH IF NOT PAST END OF BUFFER
                        ;
EAA2  A53B              LDA     CHKSNT
EAA4  D00B ^EAB1        BNE     RELONE      ;BRANCH IF CHECKSUM ALREADY SENT
                        ;
EAA6  A531              LDA     CHKSUM
EAA8  8D0DD2            STA     SEROUT      ;SEND CHECK SUM
EAAB  A9FF              LDA     #$FF
EAAD  853B              STA     CHKSNT      ;SET CHECKSUM SENT FLAG
EAAF  D009 ^EABA        BNE     CHKDON
                        ;
EAB1  A510      RELONE: LDA     POKMSK      ;ENABLE TRANSMIT DONE INTERRUPT
EAB3  0908              ORA     #$08
EAB5  8510              STA     POKMSK
EAB7  8D0ED2            STA     IRQEN
                        ;
EABA  68        CHKDON: PLA
EABB  A8                TAY                 ;RESTORE Y REG
EABC  68                PLA                 ;RETURN FROM INTERRUPT
EABD  40                RTI
                        ;
                        ;
EABE  A000      NOTEND: LDY     #0
EAC0  B132              LDA     (BUFRLO),Y  ;PUT NEXT BYTE FROM BUFFER
EAC2  8D0DD2            STA     SEROUT      ;INTO THE SERIAL OUTPUT REGISTER
                        ;
EAC5  18                CLC                 ;ADD IT TO CHECKSUM
EAC6  6531              ADC     CHKSUM
EAC8  6900              ADC     #0
EACA  8531              STA     CHKSUM
                        ;
EACC  4CBAEA            JMP     CHKDON      ;GO RETURN
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
```

```
                        ; TRANSMIT DONE INTERRUPT SERVICE ROUTINE
                        ;
EACF  A53B      ISRTD:  LDA     CHKSNT
EAD1  F00B ^EADE        BEQ     FOOEY       ;BRANCH IF CHECKSUM NOT YET SENT
                        ;
EAD3  853A              STA     XMTDON      ;OTHERWISE SET TRANSMISSION DONE FLAG
                        ;
EAD5  A510              LDA     POKMSK      ;DISABLE TRANSMIT DONE INTERRUPT
EAD7  29F7              AND     #$F7
EAD9  8510              STA     POKMSK
EADB  8D0ED2            STA     IRQEN
                        ;
EADE  68        FOOEY:  PLA                 ;RETURN FROM INTERRUPT
EADF  40                RTI
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ; RECEIVE SUBROUTINE
                        ;
EAE0  A900      RECEIV: LDA     #0
                        ;
EAE2  AC0F03            LDY     CASFLG

EAE5  D002 ^EAE9        BNE     NOCLR       ;BRANCH IF CASSETTE
                        ;
EAE7  8531              STA     CHKSUM      ;CLEAR CHKSUM
EAE9  8538      NOCLR:  STA     BUFRFL      ;BUFFER FULL FLAG
EAEB  8539              STA     RECVDN      ;RECEIVE DONE FLAG
                        ;
                        ;
                        ;
EAED  A901              LDA     #SUCCES
EAEF  8530              STA     STATUS      ;SET GOOD STATUS FOR DEFAULT CASE.
EAF1  201BEC            JSR     RECVEN      ;DO RECEIVE ENABLE
EAF4  A93C              LDA     #NCOMHI     ;COMMAND FRAME HI COMMAND
EAF6  8D03D3            STA     PBCTL       ;STORE IN PIA
EAF9  A511      CHKTIM: LDA     BRKKEY
EAFB  D003 ^EB00        BNE     NTBRK1
EAFD  4CA0ED            JMP     BROKE       ;JUMP IF BREAK KEY PRESSED
                        ;
EB00  AD1703    NTBRK1: LDA     TIMFLG      ;NO,
EB03  F005 ^EB0A        BEQ     TOUT        ;IF TIMEOUT, GO SET ERROR STATUS
EB05  A539              LDA     RECVDN
EB07  F0F0 ^EAF9        BEQ     CHKTIM      ;DONE ?
EB09  60        GOBACK: RTS
EB0A  A98A      TOUT:   LDA     #TIMOUT     ;YES,
EB0C  8530              STA     STATUS      ;SET TIMEOUT STATUS
                        ;
                        ;
                        ;
                        ;
                        ;
```

```
                        ;
EB0E  60                RRETRN: RTS                 ;RETURN
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ; SERIAL INPUT READY INTERRUPT SERVICE ROUTINE
                        ;
EB0F  98                ISRSIR: TYA
EB10  48                        PHA                 ;SAVE Y REG ON STACK
                        ;
                        ;
                        ;
EB11  AD0FD2                    LDA     SKSTAT
EB14  8D0AD2                    STA     SKRES       ;RESET STATUS REGISTER
                        ; ********   THIS MAY NOT BE THE PLACE TO DO IT   *********
                        ;
EB17  3004 ^EB1D                BMI     NTFRAM      ;BRANCH IF NO FRAMING ERROR
                        ;
EB19  A08C                      LDY     #FRMERR
EB1B  8430                      STY     STATUS      ;SET FRAME ERRORR STATUS
                        ;
EB1D  2920              NTFRAM: AND     #$20
EB1F  D004 ^EB25                BNE     NTOVRN      ;BRANCH IF NO OVERRUN ERROR
                        ;
EB21  A08E                      LDY     #OVRRUN
EB23  8430                      STY     STATUS      ;SET OVERRUN ERROR STATUS
                        ;
EB25  A538              NTOVRN: LDA     BUFRFL
EB27  F013 ^EB3C                BEQ     NOTYET      ;BRANCH IF BUFFER WAS NOT YET FILLED
                        ;
EB29  AD0DD2                    LDA     SERIN       ;THIS INPUT BYTE IS THE CHECKSUM
EB2C  C531                      CMP     CHKSUM
EB2E  F004 ^EB34                BEQ     SRETRN      ;BRANCH IF CHECKSUMS MATCH
                        ;
EB30  A08F                      LDY     #CHKERR
EB32  8430                      STY     STATUS      ;SET CHECKSUM ERROR STATUS
                        ;
EB34  A9FF              SRETRN: LDA     #$FF        ;SET RECEIVE DONE FLAG
EB36  8539                      STA     RECVDN
                        ;
EB38  68                SUSUAL: PLA
EB39  A8                        TAY                 ;RESTORE Y REG
EB3A  68                        PLA                 ;RETURN FROM INTERRUPT
EB3B  40                        RTI
                        ;
                        ;
                        ;
EB3C  AD0DD2            NOTYET: LDA     SERIN
EB3F  A000                      LDY     #0
EB41  9132                      STA     (BUFRLO),Y  ;STORE INPUT REGISTER INTO BUFFER
                        ;
EB43  18                        CLC                 ;ADD IT TO CHECKSUM
EB44  6531                      ADC     CHKSUM
```

```
EB46  6900              ADC     #0
EB48  8531              STA     CHKSUM
                ;
EB4A  E632              INC     BUFRLO      ;INCREMENT BUFFER POINTER
EB4C  D002 ^EB50        BNE     NTWRP1
EB4E  E633              INC     BUFRHI
                ;
EB50  A532      NTWRP1: LDA     BUFRLO
EB52  C534              CMP     BFENLO
EB54  A533              LDA     BUFRHI
EB56  E535              SBC     BFENHI
EB58  90DE ^EB38        BCC     SUSUAL      ;BRANCH IF NEW BUFFER ADDRESS IS IN BUFFER LIMITS
                ;
EB5A  A53C              LDA     NOCKSM
EB5C  F006 ^EB64        BEQ     GOON        ;BRANCH IF A CHECKSUM WILL FOLLOW DATA
                ;
EB5E  A900              LDA     #0
EB60  853C              STA     NOCKSM      ;CLEAR NO CHECKSUM FLAG
                ;
EB62  F0D0 ^EB34        BEQ     SRETRN      ;GO RETURN AND SET RECEIVE DONE FLAG
                ;
                ;
EB64  A9FF      GOON:   LDA     #$FF
EB66  8538              STA     BUFRFL      ;SET BUFFER FULL FLAG
                ;
EB68  D0CE ^EB38        BNE     SUSUAL      ;GO RETURN
                ;
                ;
                ;
                ;
                ;
                ;
                ;
                ;
                ; LOAD BUFFER POINTER SUBROUTINE
                ;
                ; LOAD BUFFER POINTER WITH DCB BUFFER INFORMATION
                ;
EB6A  18        LDPNTR: CLC
EB6B  AD0403            LDA     DBUFLO
EB6E  8532              STA     BUFRLO
EB70  6D0803            ADC     DBYTLO
EB73  8534              STA     BFENLO      ;ALSO SET BUFFER END + 1 ADDRESS
                ;
EB75  AD0503            LDA     DBUFHI
EB78  8533              STA     BUFRHI
EB7A  6D0903            ADC     DBYTHI
EB7D  8535              STA     BFENHI
                ;
EB7F  60                RTS             ;RETURN
                ;
                ;
                ;
                ;
                ;
                ;
                ;
```

```
                        ;
                        ; CASSETTE HANDLING CODE
                        ;
EB80  AD0303    CASENT: LDA     DSTATS
EB83  102E ^EBB3        BPL     CASRED      ;BRANCH IF INPUT FROM CASSETTE
                        ;
                        ; WRITE A RECORD
                        ;
EB85  A9CC              LDA     #B600LO     ;SET BAUD RATE TO 600
EB87  8D04D2            STA     AUDF3
EB8A  A905              LDA     #B600HI
EB8C  8D06D2            STA     AUDF4
                        ;
EB8F  20F2EB            JSR     SENDEN      ;TURN ON POKEY MARK TONE
                        ;
EB92  A00F              LDY     #WSIRG      ;LOAD SHORT WRITE INTER RECORD GAP TIME
EB94  AD0B03            LDA     DAUX2
EB97  3002 ^EB9B        BMI     SRTIR0      ;BRANCH IF SHORT GAP IS DESIRED
                        ;
EB99  A0B4              LDY     #WIRGLO     ;SET WRITE IRG TIME
EB9B  A200      SRTIR0: LDX     #WIRGHI
EB9D  20B9ED            JSR     SETVBX
                        ;
EBA0  A934              LDA     #MOTRGO
EBA2  8D02D3            STA     PACTL       ;TURN ON MOTOR
                        ;
EBA5  AD1703    TIMIT:  LDA     TIMFLG      ;LOOP UNTIL DONE
EBA8  D0FB ^EBA5        BNE     TIMIT
                        ;
EBAA  206AEB            JSR     LDPNTR      ;LOAD BUFFER POINTER WITH DCB INFORMATION
                        ;
EBAD  206BEA            JSR     SEND        ;SEND A BUFFER
                        ;
EBB0  4CDFEB            JMP     CRETRN      ;GO RETURN
                        ;
                        ;
                        ;
                        ; RECEIVE A RECORD
                        ;
EBB3  A9FF      CASRED: LDA     #$FF
EBB5  8D0F03            STA     CASFLG      ;SET CASSETTE FLAG
                        ;
EBB8  A00A              LDY     #RSIRG      ;LOAD SHORT READ INTER RECORD GAP TIME
EBBA  AD0B03            LDA     DAUX2
EBBD  3002 ^EBC1        BMI     SRTIR1      ;BRANCH IF SHORT GAP IS DESIRED
                        ;
EBBF  A078              LDY     #RIRGLO     ;SET TIME OUT FOR READ IRG
EBC1  A200      SRTIR1: LDX     #RIRGHI
EBC3  20B9ED            JSR     SETVBX
                        ;
EBC6  A934              LDA     #MOTRGO
EBC8  8D02D3            STA     PACTL       ;TURN ON MOTOR
                        ;
EBCB  AD1703    TIMIT1: LDA     TIMFLG      ;LOOP UNTIL DONE
EBCE  D0FB ^EBCB        BNE     TIMIT1
                        ;
EBD0  206AEB            JSR     LDPNTR      ;LOAD BUFFER POINTER WITH DCB INFORMATION
```

```
                        ;
EBD3  2075EC            JSR     STTMOT      ;SET DEVICE TIME OUT IN Y,X
EBD6  20B9ED            JSR     SETVBX
                        ;
EBD9  2010ED            JSR     BEGIN       ;SET INITIAL BAUD RATE
                        ;
EBDC  20E0EA            JSR     RECEIV      ;GO RECEIVE A BLOCK
                        ;
EBDF  AD0B03    CRETRN: LDA     DAUX2
EBE2  3005 ^EBE9        BMI     SRTIR2      ;BRANCH IF DOING SHORT INTER RECORD GAPS
                        ; DON'T TURN OFF CASSETTE MOTOR
EBE4  A93C              LDA     #MOTRST
EBE6  8D02D3            STA     PACTL       ;TURN OFF MOTOR
                        ;
EBE9  4C0DEA    SRTIR2: JMP     RETURN      ;GO RETURN
                        ;
                        ;
                        ;
                        ;
                        ;
EBEC  A900      JTIMER: LDA     #$00
      = 00EB    JTADRH  =       HIGH JTIMER ;HI BYTE OF JUMP TIMER ROUTINE ADDR
      = 00EC    JTADRL  =       LOW JTIMER
EBEE  8D1703            STA     TIMFLG      ;SET TIME OUT FLAG
EBF1  60                RTS
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ; SEND ENABLE SUBROUTINE
                        ;
EBF2  A907      SENDEN: LDA     #$07        ;MASK OFF PREVIOUS SERIAL BUS CONTROL BITS
EBF4  2D3202            AND     SSKCTL
EBF7  0920              ORA     #$20        ;SET TRANSMIT MODE
                        ;
EBF9  AC0003            LDY     DDEVIC
EBFC  C060              CPY     #CASET
EBFE  D00C ^EC0C        BNE     NOTCAS      ;BRANCH IF NOT CASSETTE
                        ;
EC00  0908              ORA     #$08        ;SET THE FSK OUTPUT BIT
                        ;
EC02  A007              LDY     #LOTONE     ;SET FSK TONE FREQUENCIES
EC04  8C02D2            STY     AUDF2
EC07  A005              LDY     #HITONE
EC09  8C00D2            STY     AUDF1
                        ;
EC0C  8D3202    NOTCAS: STA     SSKCTL      ;STORE NEW VALUE TO SYSTEM MASK
EC0F  8D0FD2            STA     SKCTL       ;STORE TO ACTUAL REGISTER
                        ;
EC12  A9C7              LDA     #$C7        ;MASK OFF PREVIOUS SERIAL BUS INTERRUPT BITS
EC14  2510              AND     POKMSK
EC16  0910              ORA     #$10        ;ENABLE OUTPUT DATA NEEDED INTERRUPT
                        ;
                        ;
EC18  4C31EC            JMP     CONTIN      ;GO CONTINUE IN RECEIVE ENABLE SUBROUTINE
```

```
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ; RECEIVE ENABLE SUBROUTINE
                        ;
EC1B  A907      RECVEN: LDA     #$07        ;MASK OFF PREVIOUS SERIAL BUS CONTROL BITS
EC1D  2D3202            AND     SSKCTL
EC20  0910              ORA     #$10        ;SET RECEIVE MODE ASYNCH.
EC22  8D3202            STA     SSKCTL      ;STORE NEW VALUE TO SYSTEM MASK
EC25  8D0FD2            STA     SKCTL       ;STORE TO ACTUAL REGISTER
                        ;
EC28  8D0AD2            STA     SKRES       ;RESET SERIAL PORT/KEYBOARD STATUS REGISTER
                        ;
EC2B  A9C7              LDA     #$C7        ;MASK OFF PREVIOUS SERIAL BUS INTERRUPT BITS
EC2D  2510              AND     POKMSK
EC2F  0920              ORA     #$20        ;ENABLE RECEIVE INTERRUPT
EC31  8510      CONTIN: STA     POKMSK      ;STORE NEW VALUE TO SYSTEM MASK
EC33  8D0ED2            STA     IRQEN       ;STORE TO ACTUAL REGISTER
                        ;
                        ;
EC36  A928              LDA     #$28        ;CLOCK CH.3 WITH 1.79 MHZ
EC38  8D08D2            STA     AUDCTL      ;CLOCK CH.4 WITH CH. 3
                        ;
EC3B  A206              LDX     #6          ;SET PURE TONES, NO VOLUME
EC3D  A9A8              LDA     #$A8
EC3F  A441              LDY     SOUNDR      ;TEST QUIET I/O FLAG
EC41  D002 ^EC45        BNE     NOISE1      ;NE IS NORMAL (NOISY)
EC43  A9A0              LDA     #$A0
EC45  9D01D2    NOISE1: STA     AUDC1,X
EC48  CA                DEX
EC49  CA                DEX
EC4A  10F9 ^EC45        BPL     NOISE1
                        ;
EC4C  A9A0              LDA     #$A0
EC4E  8D05D2            STA     AUDC3       ;TURN OFF SOUND ON CHANNEL 3
EC51  AC0003            LDY     DDEVIC
EC54  C060              CPY     #CASET
EC56  F006 ^EC5E        BEQ     CAS31       ;BRANCH IF CASSETTE IS DESIRED
EC58  8D01D2            STA     AUDC1       ;OTHERWISE TURN OFF CHANNELS 1 AND 2
EC5B  8D03D2            STA     AUDC2
                        ;
                        ;
EC5E  60        CAS31:  RTS                 ;RETURN
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
```

```
                        ;
                        ;
                        ;
                        ; DISABLE SEND AND DISABLE RECEIVE SUBROUTINES
                        ;
EC5F  EA          SENDDS: NOP
EC60  A9C7        RECVDS: LDA     #$C7        ;MASK OFF SERIAL BUS INTERRUPTS
EC62  2510                AND     POKMSK
EC64  8510                STA     POKMSK      ;STORE NEW VALUE TO SYSTEM MASK
EC66  8D0ED2              STA     IRQEN       ;STORE TO ACTUAL REGISTER
                        ;
EC69  A206                LDX     #6
EC6B  A900                LDA     #0
EC6D  9D01D2      ZERIT:  STA     AUDC1,X
EC70  CA                  DEX
EC71  CA                  DEX
EC72  10F9 ^EC6D          BPL     ZERIT       ;TURN OFF AUDIO VOLUME
                        ;
EC74  60                  RTS                 ;RETURN
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ; SET DEVICE TIME OUT VALUES IN Y,X SUBROUTINE
                        ;
EC75  AD0603      STTMOT: LDA     DTIMLO      ;GET DEVICE TIME OUT IN 1 SECOND INCR
EC78  6A                  ROR     A           ;PUT 6 HI BITS IN X, LO 2 BITS IN Y
EC79  6A                  ROR     A
EC7A  A8                  TAY                 ;TEMP SAVE
EC7B  293F                AND     #$3F        ;MASK OFF 2 HI BITS
EC7D  AA                  TAX                 ;THIS IS HI BYTE OF TIME OUT
                        ;
EC7E  98                  TYA                 ;RESTORE
EC7F  6A                  ROR     A
EC80  29C0                AND     #$C0        ;MASK OFF ALL BUT 2 HI BITS
EC82  A8                  TAY                 ;THIS IS LO BYTE OF TIME OUT
                        ;
EC83  60                  RTS
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
EC84  0FEB        INTTBL: DW      ISRSIR      ;SERIAL INPUT READY
EC86  90EA                DW      ISRODN      ;OUTPUT DATA NEEDED
EC88  CFEA                DW      ISRTD       ;TRANSMISSION DONE
```

```
                        ;
      = 00EB    SIRHI   =       HIGH ISRSIR ;SERIAL INPUT READY ISR ADDRESS
      = 000F    SIRLO   =       LOW ISRSIR
      = 00EA    ODNHI   =       HIGH ISRODN ;OUTPUT DATA NEEDED ISR ADDRESS
      = 0090    ODNLO   =       LOW ISRODN
      = 00EA    TDHI    =       HIGH ISRTD  ;TRANSMISSION DONE ISR ADDRESS
      = 00CF    TDLO    =       LOW ISRTD
                        ;
                        ;
                        ;
                        ;
                        ; SEND A DATA FRAME TO AN INTELLIGENT PERIPHERAL SUBROUTINE
                        ;
                        ;
EC8A  A201      SENDIN: LDX     #$01
EC8C  A0FF      DELAY0: LDY     #$FF
EC8E  88        DELAY1: DEY
EC8F  D0FD ^EC8E        BNE     DELAY1
EC91  CA                DEX
EC92  D0F8 ^EC8C        BNE     DELAY0
                        ;
EC94  206BEA            JSR     SEND        ;GO SEND THE DATA FRAME
                        ;
EC97  A002              LDY     #CTIMLO     ;SET ACK TIME OUT
EC99  A200              LDX     #CTIMHI
EC9B  20B9ED    WAITER: JSR     SETVBX
                        ;
EC9E  201AEA            JSR     WAIT        ;WAIT FOR ACK
                        ;
ECA1  98                TYA                 ;IF Y=0, A TIME OUT OR NACK OCCURED
                        ;
ECA2  60                RTS                 ;RETURN
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ; COMPUTE VALUE FOR POKEY FREQ REGS FOR THE BAUD RATE AS
                        ; MEASURED BY AN INTERVAL OF THE 'VCOUNT' TIMER.
                        ;
ECA3  8D1003    COMPUT: STA     TIMER2
ECA6  8C1103            STY     TIMER2+1    ;SAVE FINAL TIMER VALUE
ECA9  2004ED            JSR     ADJUST      ;ADJUST VCOUNT VALUE
ECAC  8D1003            STA     TIMER2      ;SAVE ADJUSTED VALUE
ECAF  AD0C03            LDA     TIMER1
ECB2  2004ED            JSR     ADJUST      ;ADJUST
ECB5  8D0C03            STA     TIMER1      ;SAVE ADJUSTED TIMER1 VALUE
ECB8  AD1003            LDA     TIMER2
ECBB  38                SEC
ECBC  ED0C03            SBC     TIMER1
ECBF  8D1203            STA     TEMP1       ;FIND VCOUNT DIFFERENCE
```

```
ECC2  AD1103              LDA     TIMER2+1
ECC5  38                  SEC
ECC6  ED0D03              SBC     TIMER1+1
ECC9  A8                  TAY                 ;FIND VBLANK COUNT DIFFERENCE
      = 0000              IF      PALFLG
   -                      LDA     #-$9C
   -              HITIMR: CLC
   -                      ADC     #$9C
                          ENDIF
      = FFFF              IF      PALFLG-1
ECCA  A97D                LDA     #$100-$83
ECCC  18          HITIMR: CLC
ECCD  6983                ADC     #$83        ;ACCUMULATE MULTIPLICATION
                          ENDIF
ECCF  88                  DEY
ECD0  10FA ^ECCC          BPL     HITIMR      ;DONE?
ECD2  18                  CLC
ECD3  6D1203              ADC     TEMP1       ;TOTAL VCOUNT DIFFERENCE
ECD6  A8          FINDX:  TAY                 ;SAVE ACCUM
ECD7  4A                  LSR     A
ECD8  4A                  LSR     A
ECD9  4A                  LSR     A
ECDA  0A                  ASL     A
ECDB  38                  SEC
ECDC  E916                SBC     #22         ;ADJUST TABLE INDEX
ECDE  AA                  TAX                 ;DIVIDE INTERVAL BY 4 TO GET TABLE INDEX
ECDF  98                  TYA                 ;RESTORE ACCUM
ECE0  2907                AND     #7
ECE2  A8                  TAY                 ;PULL OFF 3 LO BITS OF INTERVAL
ECE3  A9F5                LDA     #-11
ECE5  18          DOINTP: CLC
ECE6  690B                ADC     #11         ;ACCUMULATE INTERPOLATION CONSTANT
ECE8  88                  DEY
ECE9  10FA ^ECE5          BPL     DOINTP      ;INTERPOLATION CONSTANT COMPUTATION DONE?
                  ;
ECEB  A000        ENINTP: LDY     #0
ECED  8C0E03              STY     ADDCOR      ;CLEAR ADDITION CORRECTION FLAG
ECF0  38                  SEC
ECF1  E907                SBC     #7          ;ADJUST INTERPOLATION CONSTANT
ECF3  1003 ^ECF8          BPL     PLUS
ECF5  CE0E03              DEC     ADDCOR
ECF8  18          PLUS:   CLC
ECF9  7DD0ED              ADC     POKTAB,X    ;ADD CONSTANT TO LO BYTE TABLE VALUE
ECFC  A8                  TAY                 ;LO BYTE POKEY FREQ VALUE
ECFD  AD0E03              LDA     ADDCOR
ED00  7DD1ED              ADC     POKTAB+1,X  ;ADD CARRY TO HI BYTE TABLE VALUE
                  ; HI BYTE POKEY FREQ VALUE
ED03  60                  RTS
                  ;
                  ;
                  ;
                  ;       ROUTINE TO ADJUST VCOUNT VALUE
                  ;
ED04  C97C        ADJUST: CMP     #$7C
ED06  3004 ^ED0C          BMI     ADJ1        ;LARGER THAN '7C' ?
ED08  38                  SEC                 ;YES,
ED09  E97C                SBC     #$7C
```

```
ED0B  60                RTS
ED0C  18          ADJ1: CLC
      = 0000            IF      PALFLG
   -                    ADC     #$20
                        ENDIF
      = FFFF            IF      PALFLG-1
ED0D  6907              ADC     #$7
                        ENDIF
ED0F  60                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.
                ;
ED10  A511        BEGIN:LDA     BRKKEY
ED12  D003 ^ED17        BNE     NTBRK2
ED14  4CA0ED            JMP     BROKE       ;JUMP IF BREAK KEY PRESSED
                ;
ED17  78          NTBRK2:SEI
                ;
ED18  AD1703            LDA     TIMFLG
ED1B  D002 ^ED1F        BNE     OKTIM1      ;BRANCH IF NOT TIMED OUT
ED1D  F025 ^ED44        BEQ     TOUT1       ;BRANCH IF TIME OUT
                ;
ED1F  AD0FD2      OKTIM1: LDA     SKSTAT
ED22  2910              AND     #$10        ;READ SERIAL PORT
ED24  D0EA ^ED10        BNE     BEGIN       ;START BIT?
ED26  8D1603            STA     SAVIO       ;SAVE SER. DATA IN
ED29  AE0BD4            LDX     VCOUNT      ;READ VERTICAL LINE COUNTER
ED2C  A414              LDY     RTCLOK+2    ;READ LO BYTE OF VBLANK CLOCK
ED2E  8E0C03            STX     TIMER1
ED31  8C0D03            STY     TIMER1+1    ;SAVE INITIAL TIMER VALUE
                ;
ED34  A201              LDX     #1          ;SET MODE FLAG
ED36  8E1503            STX     TEMP3
ED39  A00A              LDY     #10         ;SET BIT COUNTER FOR 10 BITS
ED3B  A511        COUNT:LDA     BRKKEY
ED3D  F061 ^EDA0        BEQ     BROKE       ;BRANCH IF BREAK KEY PRESSED
                ;
ED3F  AD1703            LDA     TIMFLG
ED42  D004 ^ED48        BNE     OKTIMR      ;BRANCH IF NOT TIMED OUT
ED44  58          TOUT1:CLI
ED45  4C0AEB            JMP     TOUT        ;BRANCH IF TIME OUT
                ;
ED48  AD0FD2      OKTIMR: LDA     SKSTAT
ED4B  2910              AND     #$10        ;READ SERIAL PORT
ED4D  CD1603            CMP     SAVIO       ;DATA IN CHANGED YET?
ED50  F0E9 ^ED3B        BEQ     COUNT
ED52  8D1603            STA     SAVIO       ;YES,SAVE SER. DATA IN
```

```
ED55  88                DEY                 ;DECR. BIT COUNTER
ED56  D0E3 ^ED3B        BNE     COUNT       ;DONE?
                ;
ED58  CE1503            DEC     TEMP3       ;YES,
ED5B  3012 ^ED6F        BMI     GOREAD      ;DONE WITH BOTH MODES?
ED5D  AD0BD4            LDA     VCOUNT
ED60  A414              LDY     RTCLOK+2    ;READ TIMER LO & HI BYTES
ED62  20A3EC            JSR     COMPUT      ;NO, COMPUTE BAUD RATE
ED65  8CEE02            STY     CBAUDL
ED68  8DEF02            STA     CBAUDH      ;SET BAUD RATE INTO RAM CELLS
ED6B  A009              LDY     #9          ;SET BIT COUNTER FOR 9 BITS
ED6D  D0CC ^ED3B        BNE     COUNT
                ;
ED6F  ADEE02    GOREAD: LDA     CBAUDL
ED72  8D04D2            STA     AUDF3
ED75  ADEF02            LDA     CBAUDH
ED78  8D06D2            STA     AUDF4       ;SET POKEY FREQ REGS FOR BAUD RATE
ED7B  A900              LDA     #0
ED7D  8D0FD2            STA     SKSTAT
ED80  AD3202            LDA     SSKCTL
ED83  8D0FD2            STA     SKSTAT      ;INIT. POKEY SERIAL PORT
ED86  A955              LDA     #$55
ED88  9132              STA     (BUFRLO),Y  ;STORE '$55' AS FIRST RCV. BUFFER
ED8A  C8                INY
ED8B  9132              STA     (BUFRLO),Y
ED8D  A9AA              LDA     #$AA
ED8F  8531              STA     CHKSUM      ;STORE CHECKSUM FOR 2 BYTES OF '$AA'
ED91  18                CLC
ED92  A532              LDA     BUFRLO
ED94  6902              ADC     #2
ED96  8532              STA     BUFRLO
ED98  A533              LDA     BUFRHI
ED9A  6900              ADC     #0
ED9C  8533              STA     BUFRHI      ;INCR. BUFFER POINTER BY 1
ED9E  58                CLI
ED9F  60                RTS
                ;
                ;
                ;
EDA0  205FEC    BROKE:  JSR     SENDDS      ;BREAK KEY WAS PRESSED, SO PREPARE
EDA3  A93C              LDA     #MOTRST     ;TO RETURN
EDA5  8D02D3            STA     PACTL       ;TURN OFF MOTOR
EDA8  8D03D3            STA     PBCTL       ;RAISE NOT COMMAND LINE
                ;
EDAB  A980              LDA     #BRKABT
EDAD  8530              STA     STATUS      ;STORE BREAK ABORT STATUS CODE
                ;
EDAF  AE1803            LDX     STACKP
EDB2  9A                TXS                 ;RESTORE STACK POINTER
                ;
EDB3  C611              DEC     BRKKEY      ;SET BREAK KEY FLAG TO NONZERO
EDB5  58                CLI                 ;ALLOW IRQ'S
                ;
EDB6  4C0DEA            JMP     RETURN      ;GO RETURN
                ;
                ;
                ;
```

```
                        ;
                        ;
EDB9  A9EC      SETVBX: LDA     #JTADRL     ;STORE TIME OUT ROUTINE ADDRESS
EDBB  8D2602            STA     CDTMA1
EDBE  A9EB              LDA     #JTADRH
EDC0  8D2702            STA     CDTMA1+1
                        ;
EDC3  A901              LDA     #1          ;SET FOR TIMER 1
                        ;
EDC5  78                SEI                 ;THE SETVBL ROUTINE NEEDS THIS TO CUT SHORT
EDC6  205CE4            JSR     SETVBV      ;ANY VBLANKS THAT OCCUR
EDC9  A901              LDA     #1          ;SET FOR TIMER 1
EDCB  8D1703            STA     TIMFLG      ;SET FLAG TO NOT TIMED OUT
EDCE  58                CLI
EDCF  60                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 USECDS  OLUTION) OF 'VCOUNT' FOR 1
                        ;               CHARACTER TIME (10 BIT TIMES).
                        ;
                        ;
                        ;
                        ;               AUDF+7          BAUD RATE       VCOUNT INTERVAL
                        ;               ------          -----------     ----------------
                        ;       DW      $27C            ;1407                   56
                        ;       DW      $2D7            ;1231                   64
                        ;       DW      $332            ;1094                   72
                        ;       DW      $38D            ;985                    80
EDD0  E803      POKTAB: DW      $3E8            ;895                    88
EDD2  4304              DW      $443            ;820                    96
EDD4  9E04              DW      $49E            ;757                    104
EDD6  F904              DW      $4F9            ;703                    112
EDD8  5405              DW      $554            ;656                    120
EDDA  AF05              DW      $5AF            ;615                    128
EDDC  0A06              DW      $60A            ;579                    136
EDDE  6506              DW      $665            ;547                    144
```

```
EDE0  C006                DW      $6C0            ;518                    152
EDE2  1A07                DW      $71A            ;492                    160
EDE4  7507                DW      $775            ;469                    168
EDE6  D007                DW      $7D0            ;447                    176
                  ;       DW      $82B            ;428                    184
                  ;       DW      $886            ;410                    192
                  ;       DW      $8E1            ;394                    200
                  ;       DW      $93C            ;379                    208
                  ;       DW      $997            ;365                    216
                  ;       DW      $9F2            ;352                    224
                  ;       DW      $A4D            ;339                    232
                  ;       DW      $AA8            ;328                    240
                  ;       DW      $B03            ;318                    248
                  ;
                  ;
                  ;
                  ;
                  ;*****************************************************************************
      = EDE8      CRNTP3  =       *
EDE8  = 0014              ORG     $14
0014  02          SIOSPR: DB      DSKORG-CRNTP3 ;^GSIOL IS TOO LONG
```

```
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
      = 0002    STATVH  =       HIGH DVSTAT
      = 00EA    STATVL  =       LOW DVSTAT  ;STATUS POINTER
                        ;
                        ;
                        ;
                        ;
                        ;       CONSTANT EQUATES
                        ;
      = 0031    DISKID  =       $31         ;SERIAL BUS DISK I.D.
      = 0050    PUTSEC  =       $50         ;DISK PUT SECTOR DCB COMMAND
                        ; READ  =       $52         ;DISK GET SECTOR DCB COMMAND
                        ; WRITE =       $57         ;DISK PUT SECTOR WITH READ CHECK DCB COMMAND
      = 0053    STATC   =       $53         ;DISK STATUS DCB COMMAND
      = 0021    FOMAT   =       $21         ;DISK FORMAT DCB COMMAND !!!!! *****
      = 0000    NODAT   =       0           ;SIO COMMAND FOR "NO DATA" OPERATION
      = 0040    GETDAT  =       $40         ;SIO COMMAND FOR "DATA FROM DEVICE"
      = 0080    PUTDAT  =       $80         ;SIO COMMAND FOR "DATA TO DEVICE"
                        ;
                        ;
                        ;       VECTORS
                        ;
0015  = E450            ORG     $E450
                        ;
E450  4CEAED            JMP     DINIT       ;DISK INIT. VECTOR
E453  4CF0ED            JMP     DSKIF       ;DISK INTERFACE ENTRY POINT
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;       CONSTANTS
                        ;
E456  = EDEA            ORG     DSKORG
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;*********************************************************************
                        ;       DISK INTERFACE ROUTINE STARTS HERE
                        ;*********************************************************************
                        ;
                        ;
                        ;
                        ;
                        ;       DISK INTERFACE INITIALIZATION ROUTINE
```

```
                        ;
EDEA  A9A0      DINIT:  LDA     #160
EDEC  8D4602            STA     DSKTIM      ;SET INITIAL DISK TIMEOUT TO 160 SEC
EDEF  60                RTS
                        ;
                        ;
                        ;
                        ;       DISK INTERFACE ENTRY POINT
                        ;
EDF0  A931      DSKIF:  LDA     #DISKID
EDF2  8D0003            STA     DDEVIC      ;SET SERIAL BUS I.D IN DCB
EDF5  AD4602            LDA     DSKTIM
EDF8  AE0203            LDX     DCOMND
EDFB  E021              CPX     #FOMAT      ;IS COMMAND A FORMAT COMMAND?
EDFD  F002 ^EE01        BEQ     PUTDTO
EDFF  A907              LDA     #7          ;NO, SET TIMEOUT TO 7 SECS.
EE01  8D0603    PUTDTO: STA     DTIMLO      ;PUT DISK TIMEOUT IN DCB
EE04  A240              LDX     #GETDAT     ;SET "GET DATA" COMMAND FOR SIO
EE06  A080              LDY     #$80        ;SET BYTE COUNT TO 128
EE08  AD0203            LDA     DCOMND      ;READ COMMAND IN DCB
EE0B  C957              CMP     #WRITE      ;IS COMMAND A "PUT SECTOR" COMMAND?
EE0D  D002 ^EE11        BNE     CKSTC
EE0F  A280              LDX     #PUTDAT     ;YES, SET "PUT DATA" COMMAND FOR SIO
EE11  C953      CKSTC:  CMP     #STATC      ;IS COMMAND A STATUS COMMAND?
EE13  D00C ^EE21        BNE     PUTCNT
EE15  A9EA              LDA     #STATVL
EE17  8D0403            STA     DBUFLO
EE1A  A902              LDA     #STATVH
EE1C  8D0503            STA     DBUFHI      ;SET BUFFER ADDR TO GLOBAL STATUS BUFFER
EE1F  A004              LDY     #4          ;YES, SET BYTE COUNT TO 4
EE21  8E0303    PUTCNT: STX     DSTATS      ;PUT STATUS COMMAND FOR SIO IN DCB
EE24  8C0803            STY     DBYTLO
EE27  A900              LDA     #0
EE29  8D0903            STA     DBYTHI      ;PUT BYTE COUNT IN DCB
EE2C  2059E4            JSR     SIOV        ;CALL SERIAL I/O.
EE2F  1001 ^EE32        BPL     GOODST      ;NO ERROR
EE31  60                RTS                 ;NO, GO BACK
EE32  AD0203    GOODST: LDA     DCOMND      ;READ THE COMMAND
EE35  C953              CMP     #STATC      ;WAS IT A STATUS COMMAND?
EE37  D00A ^EE43        BNE     PUTBC
EE39  206DEE            JSR     PUTADR      ;PUT BUFFER ADDR IN TEMP REG.
EE3C  A002              LDY     #2
EE3E  B115              LDA     (BUFADR),Y  ;READ DISK TIMEOUT VALUE BYTE OF STATUS
EE40  8D4602            STA     DSKTIM      ;PUT IT IN DISK TIMEOUT REG.
EE43  AD0203    PUTBC:  LDA     DCOMND
EE46  C921              CMP     #FOMAT      ;WAS COMMAND A FORMAT COMMAND?
EE48  D01F ^EE69        BNE     ENDDIF
EE4A  206DEE    FMTD:   JSR     PUTADR      ;YES, PUT BUFFER ADDR INTO TEMP REG
EE4D  A0FE              LDY     #$FE        ;SET BUFFER POINTER
EE4F  C8        TWICE:  INY
EE50  C8                INY                 ;INCR BUFFER POINTER BY 2
EE51  B115      RDBAD:  LDA     (BUFADR),Y  ;READ LO BYTE BAD SECTOR DATA
EE53  C9FF              CMP     #$FF
EE55  D0F8 ^EE4F        BNE     TWICE       ;IS IT "FF" ?
EE57  C8                INY                 ;YES,
EE58  B115              LDA     (BUFADR),Y  ;READ HI BYTE BAD SECTOR DATA
EE5A  C8                INY
```

```
EE5B  C9FF              CMP     #$FF
EE5D  D0F2 ^EE51        BNE     RDBAD       ;IS IT "FF" ?
EE5F  88                DEY
EE60  88                DEY                 ;YES,
EE61  8C0803            STY     DBYTLO      ;PUT BAD SECTOR BYTE COUNT INTO DCB
EE64  A900              LDA     #0
EE66  8D0903            STA     DBYTHI
EE69  AC0303      ENDDIF: LDY   DSTATS
EE6C  60                RTS
                  ;
                  ;
                  ;
                  ;
                  ;       S U B R O U T I N E S
                  ;
                  ;
                  ;       PUT BUFFER ADDR FROM DCB INTO TEMP REG
                  ;
EE6D  AD0403      PUTADR: LDA     DBUFLO
EE70  8515              STA     BUFADR
EE72  AD0503            LDA     DBUFHI
EE75  8516              STA     BUFADR+1    ;PUT BUFFER ADDR IN TEMP REG
EE77  60                RTS
                  ;*****************************************************************
                  ;
                  ;
                  ;       SPARE BYTE OR MODULE TOO LONG FLAG
                  ;
      = EE78      CRNTP4  =       *
                  ;
EE78  = 0014              ORG     $14
0014  00          DSKSPR: DB      PRNORG-CRNTP4 ;^GDISKP TOO LONG
                  ;
```

```
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;       DEVICE NUMBER OR CODE EQUATES
                        ;
= 0002          OPNOUT  =       $2          ;IOCB OPEN FOR OUTPUT COMMAND
= 0028          NBUFSZ  =       40          ;PRINT NORMAL BUFFER SIZE
= 0014          DBUFSZ  =       20          ;PRINT DOUBLE BUFFFER SIZE
= 001D          SBUFSZ  =       29          ;PRINT SIDEWAYS BUFFER SIZE
= 0040          PDEVN   =       $40         ;PRINTER DEVICE NUMBER
                ; STATC =       $53         ;DCB STATUS COMMAND CODE
= 0057          WRITEC  =       $57         ;DCB WRITE COMMAND
= 0020          SPACE   =       $20         ;ASCII SPACE CHAR.
= 004E          N       =       $4E         ;ASCII "N" CHAR.
= 0044          D       =       $44         ;ASCII "D" CHAR.
= 0053          S       =       $53         ;ASCII "S" CHAR.
                        ;
                        ;
                        ;       PRINTER HANDLER ENTRY POINTS
                        ;
                        ;
                        ;
                        ;
0015  = E430            ORG     $E430
                        ;
E430  9EEE              DW      PHOPEN-1    ;PRINTER HANDLER OPEN
E432  DBEE              DW      PHCLOS-1    ;PH CLOSE
E434  9DEE              DW      BADST-1     ;PH READ
E436  A6EE              DW      PHWRIT-1    ;PH WRITE
E438  80EE              DW      PHSTAT-1    ;PH STATUS
E43A  9DEE              DW      BADST-1     ;PH SPECIAL
E43C  4C78EE            JMP     PHINIT      ;PH INIT.
E43F  00                DB      0           ;ROM FILLER
                        ;
                        ;
                        ;
                        ;
                        ;
E440  = EE78            ORG     PRNORG
                        ;
                        ;
                        ;
                        ;
                        ;       PRINTER HANDLER INITIALIZATION ROUTINE
                        ;
EE78  A91E      PHINIT: LDA     #30
EE7A  851C              STA     PTIMOT      ;SET UP INITIAL PRINTER TIMEOUT OF 30 SEC.
EE7C  60                RTS
                        ;
```

```
                        ;
                        ;       PRINTER HANDLER CONSTANTS
                        ;
EE7D  EA02      PHSTLO: DW      DVSTAT      ;STATUS BUFFER POINTER
EE7F  C003      PHCHLO: DW      PRNBUF      ;CHAR. BUFFER POINTER
                        ;
                        ;
                        ;
                        ;       ************************
                        ;       PRINTER HANDLER ROUTINES
                        ;       ************************
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;       PRINTER HANDLER STATUS ROUTINE
                        ;
EE81  A904      PHSTAT: LDA     #4
EE83  851E              STA     PBUFSZ      ;SET BUFFER SIZE TO 4 BYTES
EE85  AE7DEE            LDX     PHSTLO
EE88  AC7EEE            LDY     PHSTLO+1    ;SET POINTER TO STATUS BUFFER
EE8B  A953              LDA     #STATC      ;SET COMMAND TO "STATUS"
EE8D  8D0203            STA     DCOMND      ;SET STATUS COMMAND
EE90  8D0A03            STA     DAUX1
EE93  20E6EE            JSR     SETDCB      ;GO SETUP DCB
EE96  2059E4            JSR     SIOV        ;SEND STATUS COMMAND
EE99  3003 ^EE9E        BMI     BADST       ;GO IF ERROR
EE9B  2014EF            JSR     PHPUT       ;YES,PUT STATUS INTO GLOBAL BUFFER.
EE9E  60        BADST:  RTS
                        ;
                        ;
                        ;
                        ;
                        ;       PRINTER HANDLER OPEN ROUTINE
                        ;
EE9F  2081EE    PHOPEN: JSR     PHSTAT      ;DO STATUS COMMAND TO SIO
EEA2  A900              LDA     #0
EEA4  851D              STA     PBPNT       ;CLEAR PRINT BUFFER POINTER
EEA6  60                RTS
                        ;
                        ;
                        ;
                        ;
                        ;       PRINTER HANDLER WRITE ROUTINE
                        ;
EEA7  851F      PHWRIT: STA     PTEMP       ;SAVE ACCUM
EEA9  201AEF            JSR     PRMODE      ;GO DETERMINE PRINT MODE
EEAC  A61D              LDX     PBPNT
EEAE  A51F              LDA     PTEMP       ;GET CHAR. SENT BY CIO
EEB0  9DC003            STA     PRNBUF,X    ;PUT CHAR. IN PRINT BUFFER
EEB3  E8                INX                 ;INCR. BUFFER POINTER
EEB4  E41E              CPX     PBUFSZ      ;BUFFER POINTER=BUFFER SIZE?
EEB6  F013 ^EECB        BEQ     BUFFUL
EEB8  861D              STX     PBPNT       ;SAVE BUFFER POINTER
EEBA  C99B              CMP     #CR         ;IS CHAR. = EOL ?
EEBC  F003 ^EEC1        BEQ     BLFILL      ;IF YES, GO DO BLANK FILL.
```

```
EEBE  A001              LDY     #SUCCES     ;PUT GOOD STATUS IN Y REG FOR CIO.
EEC0  60                RTS
EEC1  A920        BLFILL: LDA     #SPACE      ;PUT BLANK IN ACCUM.
EEC3  9DC003      FILLBF: STA     PRNBUF,X    ;STORE IT IN PRINT BUFFER.
EEC6  E8                INX
EEC7  E41E              CPX     PBUFSZ
EEC9  D0F8 ^EEC3        BNE     FILLBF      ;BUFFER BLANK FILLED?
EECB  A900        BUFFUL: LDA     #0
EECD  851D              STA     PBPNT       ;CLEAR PRINT BUFFER POINTER
EECF  AE7FEE            LDX     PHCHLO
EED2  AC80EE            LDY     PHCHLO+1    ;SET POINTER TO PRINT BUFFER
EED5  20E6EE            JSR     SETDCB      ;GO SETUP DCB
EED8  2059E4            JSR     SIOV        ;SEND PRINT COMMAND
EEDB  60                RTS                 ;YES.
                  ;
                  ;
                  ;
                  ;
                  ;       PRINTER HANDLER CLOSE ROUTINE
                  ;
EEDC  201AEF      PHCLOS: JSR     PRMODE      ;GO DETERMINE PRINT MODE
EEDF  A61D              LDX     PBPNT
EEE1  D0DE ^EEC1        BNE     BLFILL
EEE3  A001              LDY     #SUCCES
EEE5  60                RTS
                  ;
                  ;
                  ;
                  ;
                  ;
                  ;
                  ;
                  ;       S U B R O U T I N E S
                  ;
                  ;
                  ;
                  ;
                  ;
                  ;       SET UP DCB TO CALL SIO
                  ;
EEE6  8E0403      SETDCB: STX     DBUFLO
EEE9  8C0503              STY     DBUFHI      ;SET BUFFER POINTER
EEEC  A940                LDA     #PDEVN
EEEE  8D0003              STA     DDEVIC      ;SET PRINTER BUS I.D. FOR DCB
EEF1  A901                LDA     #1
EEF3  8D0103              STA     DUNIT       ;SET UNIT NUMBER TO 1
EEF6  A980                LDA     #$80        ;DEVICE WILL EXPECT DATA
EEF8  AE0203              LDX     DCOMND
EEFB  E053                CPX     #STATC      ;STATUS COMMAND?
EEFD  D002 ^EF01          BNE     PSIOC
EEFF  A940                LDA     #$40        ;EXPECT DATA FROM DEVICE
EF01  8D0303      PSIOC:  STA     DSTATS      ;SET SIO MODE COMMAND.
EF04  A51E                LDA     PBUFSZ
EF06  8D0803              STA     DBYTLO      ;SET LO BYTE COUNT
EF09  A900                LDA     #0
EF0B  8D0903              STA     DBYTHI      ;SET HI BYTE COUNT
```

```
EF0E  A51C              LDA     PTIMOT
EF10  8D0603            STA     DTIMLO      ;SET DEVICE TIMEOUT COUNT
EF13  60                RTS
                ;
                ;
                ;
                ;
                ; GET DEVICE TIMEOUT FROM STATUS & SAVE IT
                ;
EF14  ADEC02    PHPUT:  LDA     DVSTAT+2
EF17  851C              STA     PTIMOT      ;SAVE DEVICE TIMEOUT
EF19  60                RTS
                ;
                ;
                ;
                ;
                ; DETERMINE PRINT MODE & SETUP PRINT BUFFER SIZE, DCB PRINT
                ; COMMAND, & DCB AUX1 FOR PRINT MODE
                ;
EF1A  A057      PRMODE: LDY     #WRITEC     ;PUT WRITE COMMAND IN Y REG
EF1C  A52B              LDA     ICAX2Z      ;READ PRINT MODE
EF1E  C94E      CMODE:  CMP     #N
EF20  D004 ^EF26        BNE     CDUBL       ;PRINT NORMAL ?
EF22  A228              LDX     #NBUFSZ     ;YES, SET NORMAL CHAR. BUFFER SIZE

EF24  D00E ^EF34        BNE     SETBSZ
EF26  C944      CDUBL:  CMP     #D
EF28  D004 ^EF2E        BNE     CSIDE       ;PRINT DOUBLE?
EF2A  A214              LDX     #DBUFSZ     ;YES, SET DOUBLE CHAR. BUFFER SIZE
EF2C  D006 ^EF34        BNE     SETBSZ
EF2E  C953      CSIDE:  CMP     #S          ;PRINT SIDEWAYS ?
EF30  D00B ^EF3D        BNE     GOERR       ;IF NOT, GO TO ERROR ROUTINE
EF32  A21D              LDX     #SBUFSZ     ;YES, SET SIDEWAYS BUFFER SIZE
EF34  861E      SETBSZ: STX     PBUFSZ      ;STORE PRINT BUFFER SIZE
EF36  8C0203            STY     DCOMND      ;STORE DCB COMMAND
EF39  8D0A03            STA     DAUX1       ;STORE DCB AUX1 PRINT MODE
EF3C  60                RTS
EF3D  A94E      GOERR:  LDA     #N          ;SET DEFAULT PRINT MODE TO NORMAL
EF3F  D0DD ^EF1E        BNE     CMODE
                ;************************************************************
                ;
                ;
                ; SPARE BYTE OR MODULE TOO LONG FLAG
                ;
      = EF41    CRNTP5  =       *
                ;
EF41  = 0014            ORG     $14
                ;
0014  00        PRNSPR: DB      CASORG-CRNTP5 ;^GPRINTP TOO LONG
                ;
```

```
      = 0003      CBUFH   =       HIGH CASBUF
      = 00FD      CBUFL   =       LOW CASBUF
      = 0040      SRSTA   =       $40         ;SIO READ STATUS
      = 0080      SWSTA   =       $80         ;SIO WRITE STATUS
                  ;MOTRGO =       $34
                  ;MOTRST =       $3C
                  ;
                  ;
      = 00FC      DTA     =       $FC         ;DATA RECORD TYPE BYTE
      = 00FA      DT1     =       $FA         ;LAST DATA RECORD
      = 00FE      EOT     =       $FE         ;END OF TAPE
      = 00FB      HDR     =       $FB         ;HEADER
      = 0002      TONE1   =       2           ;CHANGE TO RECORD MODE TONE
      = 0001      TONE2   =       1           ;PRESS PLAY TONE
                  ;
                  ;
                  ;
0015  = E440              ORG     CASETV
E440  4BEF2AF0D5          DW      OPENC-1,CLOSEC-1,GBYTE-1,PBYTE-1,STATU-1,SPECIAL-1
E44C  4C41EF              JMP     INIT
E44F  00                  DB      0           ;ROM FILLER BYTE
                  ;
                  ;
                  ;
                  ; USED IN MONITP FOR CASSETTE BOOT
                  ;
E450  = E47A              ORG     RBLOKV
E47A  4CE9EF              JMP     RBLOK
                  ;
E47D  = E47D              ORG     CSOPIV
E47D  4C5DEF              JMP     OPINP
                  ;
                  ;
E480  = EF41              ORG     CASORG
                  ;
                  ;
                  ; INIT ROUTINE
                  ;
EF41  A9CC        INIT:   LDA     #$CC
EF43  8DEE02              STA     CBAUDL
EF46  A905                LDA     #$05
EF48  8DEF02              STA     CBAUDH      ;SET CASSET BAUD RATE TO 600
EF4B              SPECIAL:                    ;THATS ALL FOLKS
EF4B  60                  RTS
```

```
                        ;
                        ; OPEN FUNCTION - WITH NO TIMING ADJUST
                        ;
EF4C  A52B      OPENC:  LDA     ICAX2Z      ;GET AX2
EF4E  853E              STA     FTYPE       ;SAVE IT FOR FUTURE REFERENCE
EF50  A52A              LDA     ICAX1Z
EF52  290C              AND     #$0C        ;IN AND OUT BITS
EF54  C904              CMP     #$04
EF56  F005 ^EF5D        BEQ     OPINP
EF58  C908              CMP     #$08        ;SEE IF OPEN FOR OUTPUT
EF5A  F039 ^EF95        BEQ     OPOUT
EF5C  60                RTS                 ;IF ALREADY OPEN, RETURN LEAVING STATUS=$84
EF5D  A900      OPINP:  LDA     #0
EF5F  8D8902            STA     WMODE       ;SET READ MODE
EF62  853F              STA     FEOF        ;NO EOF YET
EF64  A901      SFH:    LDA     #TONE2      ;TONE FOR PRESS PLAY
EF66  2058F0            JSR     BEEP        ;GO BEEP
EF69  3024 ^EF8F        BMI     OPNRTN      ;IF ERROR DURING BEEP
EF6B  A934              LDA     #MOTRGO
EF6D  8D02D3            STA     PACTL       ;TURN MOTOR ON
      = 0000            IF      PALFLG
   -                    LDY     #$E0
   -                    LDX     #1
                        ENDIF
      = FFFF            IF      PALFLG-1
EF70  A040              LDY     #$40        ;5-31-79 9 SEC READ LEADER
EF72  A202              LDX     #2
                        ENDIF
EF74  A903              LDA     #3
EF76  8D2A02            STA     CDTMF3
EF79  205CE4            JSR     SETVBV      ;SET UP VBLANK TIMER
EF7C  AD2A02    WAITTM: LDA     CDTMF3
EF7F  D0FB ^EF7C        BNE     WAITTM      ;WAIT FOR MOTOR TO COME UP TO SPEED
EF81  A980              LDA     #$80        ;NEXT BYTE=NO BYTES IN BUFFER
EF83  853D              STA     BPTR
EF85  8D8A02            STA     BLIM
EF88  4CD3EF            JMP     OPOK        ;OPEN OK
                        ;
                        ; OPEN FOR OUTPUT
                        ;
EF8B  A080      PBRK:   LDY     #BRKABT     ;BREAK KEY ABORT STATUS
EF8D  C611              DEC     BRKKEY      ;RESET BREAK KEY
EF8F  A900      OPNRTN: LDA     #0          ;CLEAR WRITE MODE FLAG
EF91  8D8902            STA     WMODE
EF94  60                RTS                 ;AND EXIT.
                        ;
EF95  A980      OPOUT:  LDA     #$80
EF97  8D8902            STA     WMODE       ;SET WRITE MODE
EF9A  A902              LDA     #TONE1      ;TELL USER TO TURN ON RECORD MODE
EF9C  2058F0            JSR     BEEP
EF9F  30EE ^EF8F        BMI     OPNRTN      ;IF ERROR DURING BEEP
EFA1  A9CC              LDA     #$CC        ;SET BAUD RATE
EFA3  8D04D2            STA     AUDF3       ;WHICH SEEMS TO BE NESSECARY
EFA6  A905              LDA     #$05        ;FOR SOME OBSCURE REASON
EFA8  8D06D2            STA     AUDF4
EFAB  A960              LDA     #$60
EFAD  8D0003            STA     DDEVIC
```

```
EFB0  2068E4              JSR     SENDEV      ;TELL POKEY TO WRITE MARKS
EFB3  A934                LDA     #MOTRGO     ;WRITE 5 SEC BLANK TAPE
EFB5  8D02D3              STA     PACTL
EFB8  A903                LDA     #3
      = 0000              IF      PALFLG
   -                      LDX     #$3
   -                      LDY     #$C0
                          ENDIF
      = FFFF              IF      PALFLG-1
EFBA  A204                LDX     #4          ;5/30/79 20 SEC LEADER
EFBC  A080                LDY     #$80
                          ENDIF
EFBE  205CE4              JSR     SETVBV
EFC1  A9FF                LDA     #$FF
EFC3  8D2A02              STA     CDTMF3
EFC6  A511        WDLR:   LDA     BRKKEY
EFC8  F0C1 ^EF8B          BEQ     PBRK        ;IF BREAK DURING WRITE LEADER
EFCA  AD2A02              LDA     CDTMF3
EFCD  D0F7 ^EFC6          BNE     WDLR
EFCF  A900                LDA     #0          ;INIT BUFFER POINTER
EFD1  853D                STA     BPTR
EFD3  A001        OPOK:   LDY     #SUCCES
EFD5  60                  RTS
```

```
                        ;
                        ; GET BYTE
                        ;
EFD6  A53F      GBYTE:  LDA     FEOF        ;IF AT EOF ALREADY
EFD8  3033 ^F00D        BMI     ISEOF       ;RETURN EOF STATUS
EFDA  A63D              LDX     BPTR        ;BUFFER POINTER
EFDC  EC8A02            CPX     BLIM        ;IF END OF BUFFER
EFDF  F008 ^EFE9        BEQ     RBLOK       ;READ ANOTHER BLOCK
EFE1  BD0004            LDA     CASBUF+3,X  ;GET NEXT BYTE
EFE4  E63D              INC     BPTR        ;BUMP POINTER
EFE6  A001              LDY     #SUCCES     ;OK STATUS
EFE8  60        GBX:    RTS
EFE9  A952      RBLOK:  LDA     #'R'        ;READ OPCODE
EFEB  2095F0            JSR     SIOSB       ;SIO ON SYS BUF
EFEE  98                TYA
EFEF  30F7 ^EFE8        BMI     GBX         ;IF SIO ERRORS, RETURN
EFF1  A900              LDA     #0
EFF3  853D              STA     BPTR        ;RESET POINTER
EFF5  A280              LDX     #$80        ;DEFAULT # BYTES
EFF7  ADFF03            LDA     CASBUF+2
EFFA  C9FE              CMP     #EOT
EFFC  F00D ^F00B        BEQ     ATEOF       ;IF HEADER, GO READ AGAIN
EFFE  C9FA              CMP     #DT1        ;IF LAST DATA REC
F000  D003 ^F005        BNE     NLR
F002  AE7F04            LDX     CASBUF+130  ;LAST DATA RECORD, GET # BYTES
F005  8E8A02    NLR:    STX     BLIM
F008  4CD6EF            JMP     GBYTE       ;GET NEXT BYTE
F00B  C63F      ATEOF:  DEC     FEOF        ;SET FEOF
F00D  A088      ISEOF:  LDY     #EOFERR     ;ENDFILE STATUS
F00F  60                RTS
```

```
                        ;
                        ; PUT BYTE TO BUFFER
                        ;
F010  A63D      PBYTE:  LDX     BPTR        ;BUFFER POINTER
F012  9D0004            STA     CASBUF+3,X  ;STORE CHAR AWAY
F015  E63D              INC     BPTR        ;BUMP POINTER
F017  A001              LDY     #SUCCES     ;OK STATUS
F019  E07F              CPX     #127        ;IF BUFFER FULL
F01B  F001 ^F01E        BEQ     *+3
F01D  60                RTS
                        ; WRITE OUT THE BUFFER
F01E  A9FC              LDA     #DTA        ;RECORD TYPE = DATA
F020  20D2F0            JSR     WSIOSB      ;DO WRITE ON SYSTEM BUFFER
F023  A900              LDA     #0
F025  853D              STA     BPTR        ;RESET BUFFER POINTER
F027  60                RTS                 ;EXIT.
```

```
                        ;
                        ; STATUS - RETURN STATUS INFO THRU DVSTAT
                        ;
F028  A001              STATU:  LDY     #SUCCES
F02A  60                        RTS
```

```
                        ;
                        ; CLOSE
                        ;
F02B  AD8902    CLOSEC: LDA     WMODE       ;SEE IF WRITING
F02E  3008 ^F038        BMI     CLWRT       ;GO CLOSE FOR WRITE
                        ; CLOSE FOR READ - FLAG CLOSED
F030  A001              LDY     #SUCCES     ;SUCCESSFULL
F032  A93C      FCAX:   LDA     #MOTRST     ;STOP THE MOTOR IN CASE WAS SHORT IRG MODE
F034  8D02D3            STA     PACTL
F037  60                RTS
F038  A63D      CLWRT:  LDX     BPTR        ;BUFFER POINTER
F03A  F00A ^F046        BEQ     WTLR        ;IF NO DATA BYTES IN BUFFER, NO DT1 REC
F03C  8E7F04            STX     CASBUF+130  ;WRITE TO LAST RECORD
F03F  A9FA              LDA     #DT1        ;REC TYPE
F041  20D2F0            JSR     WSIOSB      ;WRITE OUT USER BUFFER
F044  30EC ^F032        BMI     FCAX        ;GO IF ERROR
F046  A27F      WTLR:   LDX     #127        ;ZERO BUFFER
F048  A900              LDA     #0
F04A  9D0004    ZTBUF:  STA     CASBUF+3,X
F04D  CA                DEX
F04E  10FA ^F04A        BPL     ZTBUF
F050  A9FE              LDA     #EOT        ;WRITE EOT RECORD
F052  20D2F0            JSR     WSIOSB
F055  4C32F0            JMP     FCAX        ;FLAG CLOSED AND EXIT
```

```
                        ;
                        ; SUBROUTINES
                        ;
                        ; BEEP - GENERATE TONE ON KEYBOARD SPEAKER
                        ; ON ENTRY A= FREQ
                        ;
F058  8540      BEEP:   STA     FREQ
F05A  A514      BEEP1:  LDA     RTCLOK+2        ;CURRENT CLOCK
F05C  18                CLC
      = 0000            IF      PALFLG
   -                    ADC     #25
                        ENDIF
      = FFFF            IF      PALFLG-1
F05D  691E              ADC     #30             ;1 SEC TONE
                        ENDIF
F05F  AA                TAX
F060  A9FF      WFL:    LDA     #$FF
F062  8D1FD0            STA     CONSOL          ;TURN ON SPEAKER
F065  A900              LDA     #0
F067  A0F0              LDY     #$F0
F069  88                DEY
F06A  D0FD ^F069        BNE     *-1
F06C  8D1FD0            STA     CONSOL          ;TURN OFF SPEAKER
F06F  A0F0              LDY     #$F0
F071  88                DEY
F072  D0FD ^F071        BNE     *-1
F074  E414              CPX     RTCLOK+2        ;SEE IF 1 SEC IS UP YET
F076  D0E8 ^F060        BNE     WFL
F078  C640              DEC     FREQ            ;COUNT BEEPS
F07A  F00B ^F087        BEQ     WFAK            ;IF ALL DONE GO WAIT FOR KEY
F07C  8A                TXA
F07D  18                CLC
      = 0000            IF      PALFLG
   -                    ADC     #8
                        ENDIF
      = FFFF            IF      PALFLG-1
F07E  690A              ADC     #10
                        ENDIF
F080  AA                TAX
F081  E414              CPX     RTCLOK+2
F083  D0FC ^F081        BNE     *-2
F085  F0D3 ^F05A        BEQ     BEEP1           ;UNCOND GO BEEP AGIN
F087  208CF0    WFAK:   JSR     WFAK1           ;USE SIMULATED "JMP (KGETCH)"
F08A  98                TYA
F08B  60                RTS
F08C  AD25E4    WFAK1:  LDA     KEYBDV+5
F08F  48                PHA
F090  AD24E4            LDA     KEYBDV+4        ;SIMULATE "JMP (KGETCH)"
F093  48                PHA
F094  60                RTS
                        ;
                        ; SIOSB - CALL SIO ON SYSTEM BUFFER
                        ;
F095  8D0203    SIOSB:  STA     DCOMND          ;SAVE COMMAND
F098  A900              LDA     #0
F09A  8D0903            STA     DBYTHI          ;SET BUFFER LENGTH
F09D  A983              LDA     #131
```

```
F09F  8D0803              STA     DBYTLO
F0A2  A903                LDA     #CBUFH
F0A4  8D0503              STA     DBUFHI      ;SET BUFFER ADDRESS
F0A7  A9FD                LDA     #CBUFL
F0A9  8D0403              STA     DBUFLO
F0AC  A960        CSIO:   LDA     #$60        ;CASSET PSEUDO DEVICE
F0AE  8D0003              STA     DDEVIC
F0B1  A900                LDA     #0
F0B3  8D0103              STA     DUNIT
F0B6  A923                LDA     #35         ;DEVICE TIMEOUT (5/30/79)
F0B8  8D0603              STA     DTIMLO
F0BB  AD0203              LDA     DCOMND      ;GET COMMAND BACK
F0BE  A040                LDY     #SRSTA      ;SIO READ STATUS COMMAND
F0C0  C952                CMP     #'R
F0C2  F002 ^F0C6          BEQ     *+4
F0C4  A080                LDY     #SWSTA      ;SIO WRITE STATUS COMMAND
F0C6  8C0303              STY     DSTATS      ;SET STATUS FOR SIO
F0C9  A53E                LDA     FTYPE
F0CB  8D0B03              STA     DAUX2       ;INDICATE IF SHORT IRG MODE
F0CE  2059E4              JSR     SIOV        ;GO CALL SIO
F0D1  60                  RTS
                  ;
                  ; WSIOSB - WRITE SIO SYSTEM BUFFER
                  ;
F0D2  8DFF03      WSIOSB: STA     CASBUF+2    ;STORE TYPE BYTE
F0D5  A955                LDA     #$55
F0D7  8DFD03              STA     CASBUF+0
F0DA  8DFE03              STA     CASBUF+1
F0DD  A957                LDA     #'W'        ;WRITE
F0DF  2095F0              JSR     SIOSB       ;CALL SIO ON SYSTEM BUFFER
F0E2  60                  RTS                 ;RETURN
      = F0E3      CRNTP6  =       *
F0E3  = 0014              ORG     $14
0014  00          CASSPR: DB      MONORG-CRNTP6 ;^GCASCV IS TOO LONG
```

```
                        ;
                        ;
                        ;
                        ;       CONSTANT EQUATES
                        ;
= 0009          PUTTXT  =       $9          ;"PUT TEXT RECORD" CIO COMMAND CODE
= 0007          GETCAR  =       $7          ;"GET CHARACTER" CIO COMMAND CODE
= 000B          PUTCAR  =       $B          ;"PUT CHARACTER" CIO COMMAND CODE
= 0000          INIMLL  =       $00         ;INITIAL MEM LO LOW BYTE
= 0007          INIMLH  =       $07         ;INITIAL MEM LO HIGH BYTE
                ; GOOD  =       $1          ;GOOD STATUS CODE
                ; WRITE =       $57         ;WRITE COMMAND
                ; READ  =       $52         ;READ COMMAND
                ; STATC =       $53         ;STATUS COMMAND
= 0000          SEX     =       $0          ;SCREEN EDITOR IOCB INDEX
= 007D          CLS     =       $7D         ;CLEAR SCREEN CODE
= 0092          CTRLC   =       $92         ;KEYBOARD CODE FOR 'CONTROL C'
= 0088          EOF     =       $88         ;CASSETTE END OF FILE CODE
= 0000          LIRG    =       $0          ;LONG IRG TYPE CODE
                ;
= 0004          BUFFH   =       HIGH [CASBUF+3]
= 0000          BUFFL   =       LOW [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)
                ;
0015  = BFFA            ORG     $BFFA
BFFA  = 0002    CARTCS: DS      2           ;CARTRIDGE COLD START ADDRESS.
BFFC  = 0001    CART:   DS      1           ;CARTRIDGE AVAILABLE FLAG BYTE.
BFFD  = 0001    CARTFG: DS      1           ;CARTRIDGE FLAG BYTE. BIT 0=FLAG1,
BFFE  = 0002    CARTAD: DS      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
                        ;
                        ;****************************
                        ;       ORG     $FFFC
                        ;
                        ; PVECT DW      PWRUP           POWER-UP VECTOR
                        ;*************************
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;       ENTRY POINT VECTOR
                        ;
C000  = E471            ;       ORG     BLKBDV
                        ;
E471  4C23F2            ;       JMP     SIGNON      ;BLACK BOARD VECTOR
                        ;
E474  = E474            ;       ORG     WARMSV
                        ;
E474  4C1BF1            ;       JMP     RESET       ;WARM START VECTOR
                        ;
E477  = E477            ;       ORG     COLDSV
                        ;
E477  4C25F1            ;       JMP     PWRUP       ;COLD START VECTOR (9000 FOR RAM VECTOR WRITING)
                        ;
E47A  = 9000            ;       ORG     $9000
9000  200C90            ;       JSR     $900C
9003  4C25F1            ;       JMP     PWRUP       ;(TO HANDLE RAM VECTOR WRITING)
9006  200C90            ;       JSR     $900C
9009  4C1BF1            ;       JMP     RESET
                        ;
                        ;
                        ;
900C  = F0E3            ;       ORG     MONORG
                        ;
                        ;
                        ;
                        ;
                        ;       HANDLER TABLE ENTRIES
                        ;
F0E3  50                TBLENT: DB      'P'
F0E4  30E4                      DW      PRINTV
F0E6  43                        DB      'C'
F0E7  40E4                      DW      CASETV
F0E9  45                        DB      'E'
```

```
F0EA  00E4            DW      EDITRV
F0EC  53              DB      'S'
F0ED  10E4            DW      SCRENV
F0EF  4B              DB      'K'
F0F0  20E4            DW      KEYBDV
                ;
                ;
                ;TBLLEN  =       IDENT-TBLENT-1  HANDLER TABLE LENGTH.  "MOVED TO LINE 885"
                ;
                ;       ***** PRINT MESSAGES *****
                ;
                ;
F0F2  7D41544152IDENT:  DB      CLS,'ATARI COMPUTER - MEMO PAD',CR
                ;
      = 00F0    IDENTH  =       HIGH IDENT
      = 00F2    IDENTL  =       LOW IDENT       ;SYSTEM I.D. MSG POINTER
                ;
      = 000E    TBLLEN  =       IDENT-TBLENT-1 ;HANDLER TABLE LENGTH
F10D  424F4F5420DERR5:  DB      'BOOT ERROR',CR
                ;
      = 00F1    DERRH   =       HIGH DERR5
      = 000D    DERRL   =       LOW DERR5       ;DISK ERROR MSG POINTER
                ;
                ;
                ;
                ;
                ;       DEVICE/FILENAME SPECIFICATIONS
                ;
F118  453A9B    OPNEDT: DB      'E:',CR ;"OPEN SCREEN EDITOR" DEVICE SPEC.
                ;
      = 00F1    OPNH    =       HIGH OPNEDT
      = 0018    OPNL    =       LOW OPNEDT ;SCREEN EDITOR OPEN POINTER

                ;
                ;
                ;
                ;
                ;*****************************************************************
                ;       RESET BUTTON ROUTINE STARTS HERE
                ;*****************************************************************
                ;
F11B  78        RESET:  SEI                     ;DISABLE IRQ INTERRUPTS
F11C  AD4402            LDA     COLDST          ;WERE WE IN MIDDLE OF COLDSTART?
F11F  D004 ^F125        BNE     PWRUP           ;YES, GO TRY IT AGAIN
F121  A9FF              LDA     #$FF
F123  D003 ^F128        BNE     PWRUP1          ;SET WARM START FLAG
                ;
                ;
                ;
                ;*************************************************************
                ;       POWER UP ROUTINES START HERE
                ;*************************************************************
                ;
F125  78        PWRUP:  SEI                     ;DISABLE IRQ INTERRUPTS
F126  A900              LDA     #0              ;CLEAR WARMSTART FLAG
F128  8508      PWRUP1: STA     WARMST
F12A  D8                CLD                    ;CLEAR DECIMAL FLAG.
```

```
F12B  A2FF              LDX     #$FF
F12D  9A                TXS                     ;SET STACK POINTER
F12E  203FF2            JSR     SPECL           ;CARTRIDGE SPECIAL CASE?
F131  2077F2            JSR     HARDI           ;DO HARDWARE INITIALIZATION
F134  A508              LDA     WARMST          ;IS IT WARMSTART?
F136  D028 ^F160        BNE     ZOSRAM          ;YES, ONLY ZERO OS RAM
                ;
F138  A900      ZERORM: LDA     #0
F13A  A008              LDY     #WARMST
F13C  8504              STA     RAMLO
F13E  8505              STA     RAMLO+1         ;INITIALIZE RAM POINTER
F140  9104      CLRRAM: STA     (RAMLO),Y       ;CLEAR MEMORY LOC.
F142  C8                INY
F143  C000              CPY     #0              ;AT END OF PAGE?
F145  D0F9 ^F140        BNE     CLRRAM
F147  E605              INC     RAMLO+1         ;YES, INCR PAGE POINTER
F149  A605              LDX     RAMLO+1
F14B  E406              CPX     TRAMSZ          ;AT END OF MEM?
F14D  D0F1 ^F140        BNE     CLRRAM          ;NO.
                ;
                ; INITIALIZE DOSVEC TO POINT TO SIGNON (BLACKBOARD)
F14F  AD72E4            LDA     BLKBDV+1
F152  850A              STA     DOSVEC          ;USE BLACKBOARD VECTOR
F154  AD73E4            LDA     BLKBDV+2        ;FOR DOSVEC
F157  850B              STA     DOSVEC+1
F159  A9FF              LDA     #$FF
F15B  8D4402            STA     COLDST          ;SET TO SHOW IN MIDDLE OF COLDSTART
F15E  D013 ^F173        BNE     ESTSCM          ;GO AROUND ZOSRAM
                ;
                ; CLEAR OS RAM (FOR WARMSTART)
F160  A200      ZOSRAM: LDX     #0
F162  8A                TXA
F163  9D0002    ZOSRM2: STA     $200,X          ;CLEAR PAGES 2 AND 3
F166  9D0003            STA     $300,X
F169  CA                DEX
F16A  D0F7 ^F163        BNE     ZOSRM2
F16C  A210              LDX     #INTZBS
F16E  9500      ZOSRM3: STA     0,X             ;CLEAR ZERO PAGE LOCATIONS INTZBS-7F
F170  E8                INX
F171  10FB ^F16E        BPL     ZOSRM3
                ;
                ; ESTABLISH SCREEN MARGINS
F173  A902      ESTSCM: LDA     #LEDGE
F175  8552              STA     LMARGN
F177  A927              LDA     #REDGE
F179  8553              STA     RMARGN
                ;
                ;
                ; MOVE VECTOR TABLE FROM ROM TO RAM
F17B  A225      OPSYS:  LDX     #$25
F17D  BD80E4    MOVVEC: LDA     VCTABL,X        ;ROM TABLE
F180  9D0002            STA     INTABS,X        ;TO RAM
F183  CA                DEX
F184  10F7 ^F17D        BPL     MOVVEC
F186  208AF2            JSR     OSRAM           ;DO O.S. RAM SETUP
F189  58                CLI                     ;ENABLE IRQ INTERRUPTS
                ;
```

```
                        ;
                        ;       LINK HANDLERS
                        ;
F18A  A20E              NXTENT: LDX     #TBLLEN
F18C  BDE3F0                    LDA     TBLENT,X    ;READ HANDLER TABLE ENTRY
F18F  9D1A03                    STA     HATABS,X    ;PUT IN TABLE
F192  CA                        DEX
F193  10F7 ^F18C                BPL     NXTENT      ;DONE WITH ALL ENTRIES?
                        ;
                        ;
                        ;
                        ;
                        ;
                        ; INTERROGATE CARTRIDGE ADDR. SPACE TO SEE WHICH CARTRIDGES THERE ARE
                        ;
F195  A200                      LDX     #0
F197  8607                      STX     TSTDAT      ;CLEAR "B" CART. FLAG
F199  8606                      STX     TRAMSZ      ;CLEAR "A" CART. FLAG
F19B  AEE402                    LDX     RAMSIZ
F19E  E090                      CPX     #$90        ;RAM IN "B" CART. SLOT?
F1A0  B00A ^F1AC                BCS     ENDBCK
F1A2  ADFC9F                    LDA     CART-$2000  ;NO,
F1A5  D005 ^F1AC                BNE     ENDBCK      ;CART. PLUGGED INTO "B" SLOT?
F1A7  E607                      INC     TSTDAT      ;YES, SET "B" CART. FLAG
F1A9  203CF2                    JSR     CBINI       ;INITIALIZE CARTRIDGE "B"
                        ;
F1AC  AEE402            ENDBCK: LDX     RAMSIZ
F1AF  E0B0                      CPX     #$B0        ;RAM IN "A" CART. SLOT?
F1B1  B00A ^F1BD                BCS     ENDACK
F1B3  AEFCBF                    LDX     CART        ;NO,
F1B6  D005 ^F1BD                BNE     ENDACK      ;CART. PLUGGED INTO "A" SLOT?
F1B8  E606                      INC     TRAMSZ      ;YES, SET "A" CART. FLAG
F1BA  2039F2                    JSR     CAINI       ;INITIALIZE CARTRIDGE "A"
                        ;
                        ;
                        ; OPEN SCREEN EDITOR
                        ;
F1BD  A903              ENDACK: LDA     #3
F1BF  A200                      LDX     #SEX
F1C1  9D4203                    STA     ICCOM,X     ;OPEN I/O COMMAND
F1C4  A918                      LDA     #OPNL
F1C6  9D4403                    STA     ICBAL,X
F1C9  A9F1                      LDA     #OPNH
F1CB  9D4503                    STA     ICBAH,X     ;SET BUFFER POINTER TO OPEN SCREEN EDITOR
F1CE  A90C                      LDA     #$C
F1D0  9D4A03                    STA     ICAX1,X     ;SET UP OPEN FOR INPUT/OUTPUT
F1D3  2056E4                    JSR     CIOV        ;GO TO CIO
                        ;
F1D6  1003 ^F1DB                BPL     SCRNOK      ;BR IF NO ERROR
F1D8  4C25F1                    JMP     PWRUP       ;RETRY PWRUP IF ERROR (SHOULD NEVER HAPPEN!)
F1DB  E8                SCRNOK: INX                 ;SCREEN OK, SO WAIT FOR VBLANK TO
F1DC  D0FD ^F1DB                BNE     SCRNOK      ;BRING UP THE DISPLAY
F1DE  C8                        INY
F1DF  10FA ^F1DB                BPL     SCRNOK
                        ;
                        ;
                        ; DO CASSETTE BOOT
```

```
F1E1  20B2F3              JSR     CSBOOT      ;CHECK, BOOT, AND INIT
                  ;
                  ; CHECK TO SEE IF EITHER CARTRIDGE WANTS DISK BOOT
F1E4  A506                LDA     TRAMSZ      ;CHECK BOTH CARTRIDGES
F1E6  0507                ORA     TSTDAT      ;
F1E8  F012 ^F1FC          BEQ     NOCART      ;NEITHER CARTRIDGE LIVES
F1EA  A506                LDA     TRAMSZ      ;"A" CART?
F1EC  F003 ^F1F1          BEQ     NOA1        ;NO
F1EE  ADFDBF              LDA     CARTFG      ;GET CARTRIDGE MODE FLAG
F1F1  A607        NOA1:   LDX     TSTDAT      ;"B" CART?
F1F3  F003 ^F1F8          BEQ     NOB1        ;NO
F1F5  0DFD9F              ORA     CARTFG-$2000 ;ADD OTHER FLAG
F1F8  2901        NOB1:   AND     #1          ;DOES EITHER CART WANT BOOT?
F1FA  F003 ^F1FF          BEQ     NOBOOT      ;NO
                  ;
                  ; DO DISK BOOT
F1FC  20CFF2      NOCART: JSR     BOOT        ;CHECK, BOOT, AND INIT
                  ;
                  ; GO TO ONE OF THE CARTRIDGES IF THEY SO DESIRE
F1FF  A900        NOBOOT: LDA     #0
F201  8D4402              STA     COLDST      ;RESET TO SHOW DONE WITH COLDSTART
F204  A506                LDA     TRAMSZ      ;"A" CART?
F206  F00A ^F212          BEQ     NOA2        ;NO
F208  ADFDBF              LDA     CARTFG      ;GET CARTRIDGE MODE FLAG
F20B  2904                AND     #4          ;DOES IT WANT TO RUN?
F20D  F003 ^F212          BEQ     NOA2        ;NO
F20F  6CFABF              JMP     (CARTCS)    ;RUN "A" CARTRIDGE
F212  A507        NOA2:   LDA     TSTDAT      ;"B" CART?
F214  F00A ^F220          BEQ     NOCAR2      ;NO
F216  ADFD9F              LDA     CARTFG-$2000 ;GET "B" MODE FLAG
F219  2904                AND     #4          ;DOES IT WANT TO RUN?
F21B  F0DF ^F1FC          BEQ     NOCART      ;NO
F21D  6CFA9F              JMP     (CARTCS-$2000) ;RUN "B" CARTRIDGE
                  ;
                  ; NO CARTRIDGES, OR NEITHER WANTS TO RUN,
                  ; SO GO TO DOSVEC (DOS, CASSETTE, OR BLACKBOARD)
F220  6C0A00      NOCAR2: JMP     (DOSVEC)
                  ;
                  ; PRINT SIGN-ON MESSAGE
F223  A2F2        SIGNON: LDX     #IDENTL
F225  A0F0                LDY     #IDENTH
F227  2085F3              JSR     PUTLIN      ;GO PUT SIGN-ON MSG ON SCREEN
                  ;
                  ;
                  ;
                  ;       BLACKBOARD ROUTINE
F22A  2030F2      BLACKB: JSR     BLKB2       ;"JSR EGETCH"
F22D  4C2AF2              JMP     BLACKB      ;FOREVER
F230  AD05E4      BLKB2:  LDA     EDITRV+5    ;HIGH BYTE
F233  48                  PHA
F234  AD04E4              LDA     EDITRV+4    ;LOW BYTE
F237  48                  PHA
F238  60                  RTS                 ;SIMULATES "JMP (EDITRV)"
                  ;
                  ;
                  ; CARTRIDGE INITIALIZATION INDIRECT JUMPS
F239  6CFEBF      CAINI:  JMP     (CARTAD)
```

```
F23C  6CFE9F      CBINI:  JMP     (CARTAD-$2000)
```

```
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;               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.
                        ;
F23F  ADFCBF      SPECL:  LDA     CART        ;CHECK FOR RAM OR CART
F242  D013 ^F257          BNE     ENSPE2      ;GO IF NOTHING OR MAYBE RAM
F244  EEFCBF              INC     CART        ;NOW DO RAM CHECK
F247  ADFCBF              LDA     CART        ;IS IT ROM?
F24A  D008 ^F254          BNE     ENSPEC      ;NO
F24C  ADFDBF              LDA     CARTFG      ;YES,
F24F  1003 ^F254          BPL     ENSPEC      ;BIT SET?
F251  6CFEBF              JMP     (CARTAD)    ;YES, GO RUN CARTRIDGE
                        ;
                        ; CHECK FOR AMOUNT OF RAM
                        ;
                        ;
F254  CEFCBF      ENSPEC: DEC     CART        ;RESTORE RAM IF NEEDED
F257  A000        ENSPE2: LDY     #0
F259  8405                STY     RAMLO+1
F25B  A910                LDA     #$10
F25D  8506                STA     TRAMSZ      ;SET RAM POINTER TO 4K.
F25F  B105        HOWMCH: LDA     (RAMLO+1),Y ;READ RAM LOCATION
F261  49FF                EOR     #$FF        ;INVERT IT.
F263  9105                STA     (RAMLO+1),Y ;WRITE INVERTED DATA.
F265  D105                CMP     (RAMLO+1),Y ;READ RAM AGAIN
F267  D00D ^F276          BNE     ENDRAM
F269  49FF                EOR     #$FF        ;CONVERT IT BACK
F26B  9105                STA     (RAMLO+1),Y ;RESTORE ORIGINAL RAM DATA
F26D  A506                LDA     TRAMSZ
F26F  18                  CLC
F270  6910                ADC     #$10
F272  8506                STA     TRAMSZ      ;INCR. RAM POINTER BY 4K.
F274  D0E9 ^F25F          BNE     HOWMCH      ;GO FIND HOW MUCH RAM.
```

```
F276  60          ENDRAM: RTS
                  ;
                  ;
                  ;
                  ;
                  ;       HARDWARE INITIALIZATION
                  ;
                  ;
F277  A900        HARDI:  LDA     #0
F279  AA                  TAX
F27A  9D00D0      CLRCHP: STA     $D000,X
F27D  9D00D4              STA     $D400,X
F280  9D00D2              STA     $D200,X
F283  9D00D3              STA     $D300,X
F286  E8                  INX
F287  D0F1 ^F27A          BNE     CLRCHP
F289  60                  RTS
                  ;
                  ;
                  ;       O.S. RAM SETUP
                  ;
F28A  C611        OSRAM:  DEC     BRKKEY      ;TURN OFF BREAK KEY FLAG
F28C  A954                LDA     #LOW BRKKY2
F28E  8D3602              STA     BRKKY
F291  A9E7                LDA     #HIGH BRKKY2
F293  8D3702              STA     BRKKY+1
F296  A506                LDA     TRAMSZ      ;READ RAM SIZE IN TEMP. REG.
F298  8DE402              STA     RAMSIZ      ;SAVE IT IN RAM SIZE.
F29B  8DE602              STA     MEMTOP+1    ; INIT. MEMTOP ADDR HI BYTE
F29E  A900                LDA     #0
F2A0  8DE502              STA     MEMTOP      ;INIT. MEMTOP ADDR LO BYTE
F2A3  A900                LDA     #INIMLL
F2A5  8DE702              STA     MEMLO
F2A8  A907                LDA     #INIMLH
F2AA  8DE802              STA     MEMLO+1     ;INITIALIZE MEMLO ADDR VECTOR
F2AD  200CE4              JSR     EDITRV+$C   ;EDITOR INIT.
F2B0  201CE4              JSR     SCRENV+$C   ;SCREEN INIT.
F2B3  202CE4              JSR     KEYBDV+$C   ;KEYBOARD INIT.
F2B6  203CE4              JSR     PRINTV+$C   ;PRINTER HANDLER INIT
F2B9  204CE4              JSR     CASETV+$C   ;CASSETTE HANDLER INIT
F2BC  206EE4              JSR     CIOINV      ;CIO INIT.
F2BF  2065E4              JSR     SIOINV      ;SIO INIT.
F2C2  206BE4              JSR     INTINV      ;INTERRUPT HANDLER INIT.
F2C5  AD1FD0              LDA     CONSOL
F2C8  2901                AND     #$1
F2CA  D002 ^F2CE          BNE     NOKEY       ;GAME START KEY DEPRESSED?
F2CC  E64A                INC     CKEY        ;YES, SET KEY FLAG.
F2CE  60          NOKEY:  RTS
                  ;
                  ;
                  ; DO BOOT OF DISK
                  ;
F2CF  A508        BOOT:   LDA     WARMST
F2D1  F00A ^F2DD          BEQ     NOWARM      ;WARM START?
F2D3  A509                LDA     BOOT?       ;YES,
F2D5  2901                AND     #1
F2D7  F003 ^F2DC          BEQ     NOINIT      ;VALID BOOT?
```

```
F2D9  207EF3              JSR     DINI        ;YES, RE-INIT. DOS SOFTWARE
F2DC  60          NOINIT: RTS
F2DD  A901        NOWARM: LDA     #1
F2DF  8D0103              STA     DUNIT       ;ASSIGN DISK DRIVE NO.
F2E2  A953                LDA     #STATC
F2E4  8D0203              STA     DCOMND      ;SET UP STATUS COMMAND
F2E7  2053E4              JSR     DSKINV      ;GO DO DISK STATUS
F2EA  1001 ^F2ED          BPL     DOBOOT      ;IS STATUS FROM SIO GOOD?
F2EC  60                  RTS                 ;NO, GO BACK WITH BAD BOOT STATUS
                  ;
F2ED  A900        DOBOOT: LDA     #0
F2EF  8D0B03              STA     DAUX2
F2F2  A901                LDA     #1
F2F4  8D0A03              STA     DAUX1       ;SET SECTOR # TO 1.
F2F7  A900                LDA     #BUFFL
F2F9  8D0403              STA     DBUFLO
F2FC  A904                LDA     #BUFFH
F2FE  8D0503              STA     DBUFHI      ;SET UP BUFFER ADDR
F301  209DF3      SECT1:  JSR     GETSEC      ;GET SECTOR
F304  1008 ^F30E          BPL     ALLSEC      ;STATUS O.K. ?
F306  2081F3      BADDSK: JSR     DSKRDE      ;NO, GO PRINT DISK READ ERROR
F309  A54B                LDA     CASSBT
F30B  F0E0 ^F2ED          BEQ     DOBOOT      ;CASSETTE BOOT?
F30D  60                  RTS                 ;YES, QUIT
F30E  A203        ALLSEC: LDX     #3
F310  BD0004      RDBYTE: LDA     CASBUF+3,X  ;READ A BUFFER BYTE
F313  9D4002              STA     DFLAGS,X    ;STORE IT
F316  CA                  DEX
F317  10F7 ^F310          BPL     RDBYTE      ;DONE WITH 4 BYTE TRANSFER ?
F319  AD4202              LDA     BOOTAD      ;YES,
F31C  8504                STA     RAMLO
F31E  AD4302              LDA     BOOTAD+1
F321  8505                STA     RAMLO+1     ;PUT BOOT ADDR INTO Z. PAGE RAM
F323  AD0404              LDA     CASBUF+7
F326  850C                STA     DOSINI      ;ESTABLISH DOS INIT ADDRESS
F328  AD0504              LDA     CASBUF+8
F32B  850D                STA     DOSINI+1
F32D  A07F        MVBUFF: LDY     #$7F        ;YES, SET BYTE COUNT
F32F  B90004      MVNXB:  LDA     CASBUF+3,Y
F332  9104                STA     (RAMLO),Y   ;MOVE A BYTE FROM SECTOR BUFFER TO BOOT ADDR.
F334  88                  DEY
F335  10F8 ^F32F          BPL     MVNXB       ;DONE ?
F337  18                  CLC                 ;YES,
F338  A504                LDA     RAMLO
F33A  6980                ADC     #$80
F33C  8504                STA     RAMLO
F33E  A505                LDA     RAMLO+1
F340  6900                ADC     #0
F342  8505                STA     RAMLO+1     ;INCR BOOT LOADER BUFFER POINTER.
F344  CE4102              DEC     DBSECT      ;DECR # OF SECTORS.
F347  F011 ^F35A          BEQ     ENBOOT      ;MORE SECTORS ?
F349  EE0A03              INC     DAUX1       ;YES, INCR SECTOR #
F34C  209DF3      SECTX:  JSR     GETSEC      ;GO GET SECTOR.
F34F  10DC ^F32D          BPL     MVBUFF      ;STATUS O.K. ?
F351  2081F3              JSR     DSKRDE      ;NO, GO PRINT DISK READ ERROR
F354  A54B                LDA     CASSBT
F356  D0AE ^F306          BNE     BADDSK      ;IF CASSETTE, QUIT.
```

```
F358  F0F2 ^F34C          BEQ     SECTX       ;IF DISK, TRY SECTOR AGAIN.
F35A  A54B        ENBOOT: LDA     CASSBT
F35C  F003 ^F361          BEQ     XBOOT       ;CASSETTE BOOT ?
F35E  209DF3              JSR     GETSEC      ;YES, GET EOF RECORD, BUT DON'T USE IT.
F361  206CF3      XBOOT:  JSR     BLOAD       ;GO EXECUTE BOOT LOADER
F364  B0A0 ^F306          BCS     BADDSK      ;IF BAD BOOT, DO IT OVER AGAIN
F366  207EF3              JSR     DINI        ;GO INIT. SOFTWARE
F369  E609                INC     BOOT?       ;SHOW BOOT SUCCESS
F36B  60                  RTS
F36C  18          BLOAD:  CLC
F36D  AD4202              LDA     BOOTAD
F370  6906                ADC     #6
F372  8504                STA     RAMLO
F374  AD4302              LDA     BOOTAD+1
F377  6900                ADC     #0
F379  8505                STA     RAMLO+1     ;PUT START ADDR OF BOOTLOADER INTO RAM
F37B  6C0400              JMP     (RAMLO)
F37E  6C0C00      DINI:   JMP     (DOSINI)
                  ;
                  ;
                  ;
                  ;
                  ; DISPLAY DISK READ ERROR MSG
                  ;
F381  A20D        DSKRDE: LDX     #DERRL
F383  A0F1                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
                  ;
F385  8A          PUTLIN: TXA
F386  A200                LDX     #SEX
F388  9D4403              STA     ICBAL,X
F38B  98                  TYA
F38C  9D4503              STA     ICBAH,X     ;SET UP ADDR OF BEGIN OF LINE
F38F  A909                LDA     #PUTTXT
F391  9D4203              STA     ICCOM,X     ;"PUT TEXT RECORD" COMMAND
F394  A9FF                LDA     #$FF
F396  9D4803              STA     ICBLL,X     ;SET BUFFER LENGTH
F399  2056E4              JSR     CIOV        ;PUT LINE ON SCREEN
F39C  60                  RTS
                  ;
                  ;
                  ;
                  ;
                  ; GET SECTOR FROM DISK 0
                  ;
F39D  A54B        GETSEC: LDA     CASSBT
F39F  F003 ^F3A4          BEQ     DISKM       ;CASSETTE BOOT ?
F3A1  4C7AE4              JMP     RBLOKV      ;YES, GO TO READ BLOCK ROUTINE
F3A4  A952        DISKM:  LDA     #READ
F3A6  8D0203              STA     DCOMND      ;SET READ SECTOR COMMAND
F3A9  A901                LDA     #1
```

```
F3AB  8D0103              STA     DUNIT       ;SET DRIVE NO. TO DRIVE 0
F3AE  2053E4              JSR     DSKINV      ;GET SECTOR
F3B1  60                  RTS
                ;
                ;
                ;
                ; DO CHECK FOR CASSETTE BOOT & IF SO, DO BOOT
                ;
F3B2  A508        CSBOOT: LDA     WARMST      ;WARMSTART?
F3B4  F00A ^F3C0          BEQ     CSBOT2      ;NO
F3B6  A509                LDA     BOOT?       ;GET BOOT FLAG
F3B8  2902                AND     #2          ;WAS CASSETTE BOOT SUCCESFULL?
F3BA  F003 ^F3BF          BEQ     NOCSB2      ;NO
F3BC  20E1F3              JSR     CINI        ;YES, INIT CASSETTE SOFTWARE
F3BF  60          NOCSB2: RTS
                ;
F3C0  A54A        CSBOT2: LDA     CKEY
F3C2  F01C ^F3E0          BEQ     NOCSBT      ;"C" KEY FLAG SET ?
F3C4  A980                LDA     #$80        ;YES,
F3C6  853E                STA     FTYPE       ;SET LONG IRG TYPE
F3C8  E64B                INC     CASSBT      ;SET CASSETTE BOOT FLAG
F3CA  207DE4              JSR     CSOPIV      ;OPEN CASSETTE FOR INPUT
F3CD  2001F3              JSR     SECT1       ;DO BOOT & INIT.
F3D0  A900                LDA     #0
F3D2  854B                STA     CASSBT      ;RESET CASSETTE BOOT FLAG
F3D4  854A                STA     CKEY        ;CLEAR KEY FLAG
F3D6  0609                ASL     BOOT?       ;SHIFT BOOT FLAG (NOW=2 IF SUCCESS)
F3D8  A50C                LDA     DOSINI
F3DA  8502                STA     CASINI      ;MOVE INIT ADDRESS FOR CASSETTE
F3DC  A50D                LDA     DOSINI+1
F3DE  8503                STA     CASINI+1
F3E0  60          NOCSBT: RTS
                ;
F3E1  6C0200      CINI:   JMP     (CASINI)    ;INIT CASSETTE
                ;*********************************************************************
                ;
                ;
                ; SPARE BYTE OR MODULE TOO LONG FLAG
                ;
      = F3E4      CRNTP7  =       *
                ;
F3E4  = 0014              ORG     $14
0014  00          MONSPR: DB      KBDORG-CRNTP7 ;^GMONITP TOO LONG
                ;
```

```
                ;
                ; HANDLER DEPENDENT EQUATES
                ;
= 007D      CLRCOD  =       $7D         ;CLEAR SCREEN ATASCI CODE
= 009F      CNTL1   =       $9F         ;POKEY KEY CODE FOR ^1
                ;
= 0068      FRMADR  =       SAVADR
= 0066      TOADR   =       MLTTMP
                ;
```

```
                        ;
                        ;
0015  = E400            ORG     EDITRV
                        ;
                        ; SCREEN EDITOR HANDLER ENTRY POINT
                        ;
E400  FBF3      EDITOR: DW      EOPEN-1
E402  33F6              DW      RETUR1-1    ;(CLOSE)
E404  3DF6              DW      EGETCH-1
E406  A3F6              DW      EOUTCH-1
E408  33F6              DW      RETUR1-1    ;(STATUS)
E40A  3CF6              DW      NOFUNC-1    ;(SPECIAL)
E40C  4CE4F3            JMP     PWRONA
E40F  00                DB      0           ;ROM FILLER BYTE
                        ;
E410  = E410            ORG     SCRENV
                        ;
                        ; DISPLAY HANDLER ENTRY POINT
                        ;
E410  F5F3      DISPLA: DW      DOPEN-1
E412  33F6              DW      RETUR1-1    ;(CLOSE)
E414  92F5              DW      GETCH-1
E416  B6F5              DW      OUTCH-1
E418  33F6              DW      RETUR1-1    ;(STATUS)
E41A  FBFC              DW      DRAW-1      ;(SPECIAL)
E41C  4CE4F3            JMP     PWRONA
E41F  00                DB      0           ;ROM FILLER BYTE
                        ;
E420  = E420            ORG     KEYBDV
                        ;
                        ;
                        ; KEYBOARD HANDLER ENTRY POINT
                        ;
E420  33F6      KBDHND: DW      RETUR1-1
E422  33F6              DW      RETUR1-1    ;(CLOSE)
E424  E1F6              DW      KGETCH-1
E426  3CF6              DW      NOFUNC-1    ;(OUTCH)
E428  33F6              DW      RETUR1-1    ;(STATUS)
E42A  3CF6              DW      NOFUNC-1    ;(SPECIAL)
E42C  4CE4F3            JMP     PWRONA
E42F  00                DB      0           ;ROM FILLER BYTE
                        ;
                        ;
                        ; INTERRUPT VECTOR TABLE ENTRY
E430  = E488            ORG     VCTABL-INTABS+VKEYBD
E488  BEFF              DW      PIRQ5       ;KEYBOARD IRQ INTERRUPT VECTOR
```

```
E48A  = F3E4              ORG     KBDORG
                  ;
F3E4  A9FF        PWRONA: LDA     #$FF
F3E6  8DFC02              STA     CH
F3E9  ADE602              LDA     MEMTOP+1
F3EC  29F0                AND     #$F0        ;INSURE 4K PAGE BOUNDARY
F3EE  856A                STA     RAMTOP
F3F0  A940                LDA     #$40        ;DEFAULT TO UPPER CASE ALPHA AT PWRON
F3F2  8DBE02              STA     SHFLOK
F3F5  60                  RTS                 ;POWER ON COMPLETED
```

```
                        ;
                        ;
                        ; BEGIN DISPLAY HANDLER OPEN PROCESSING
                        ;
F3F6  A52B      DOPEN:  LDA     ICAX2Z      ;GET AUX 2 BYTE
F3F8  290F              AND     #$F
F3FA  D008 ^F404        BNE     OPNCOM      ;IF MODE ZERO, CLEAR ICAX1Z
F3FC  A52A      EOPEN:  LDA     ICAX1Z      ;CLEAR "CLR INHIBIT" AND "MXD MODE" BITS
F3FE  290F              AND     #$F
F400  852A              STA     ICAX1Z
F402  A900              LDA     #0
F404  8557      OPNCOM: STA     DINDEX

F406  A9E0              LDA     #$E0        ;INITIALIZE GLOBAL VBLANK RAM
F408  8DF402            STA     CHBAS
F40B  A902              LDA     #2
F40D  8DF302            STA     CHACT
F410  8D2F02            STA     SDMCTL      ;TURN OFF DMA NEXT VBLANK
F413  A901              LDA     #SUCCES
F415  854C              STA     DSTAT       ;CLEAR STATUS
F417  A9C0              LDA     #$C0        ;DO IRQEN
F419  0510              ORA     POKMSK
F41B  8510              STA     POKMSK
F41D  8D0ED2            STA     IRQEN
F420  A900              LDA     #0
F422  8D9302            STA     TINDEX      ;TEXT INDEX MUST ALWAYS BE 0
F425  8564              STA     ADRESS
F427  857B              STA     SWPFLG
F429  8DF002            STA     CRSINH      ;TURN CURSOR ON AT OPEN
F42C  A00E              LDY     #14         ;CLEAR TAB STOPS
F42E  A901              LDA     #1          ;INIT TAB STOPS TO EVERY 8 CHARACTERS
F430  99A302    CLRTBS: STA     TABMAP,Y
F433  88                DEY
F434  10FA ^F430        BPL     CLRTBS
F436  A204              LDX     #4          ;LOAD COLOR REGISTERS
F438  BDC1FE    DOPEN8: LDA     COLRTB,X
F43B  9DC402            STA     COLOR0,X
F43E  CA                DEX
F43F  10F7 ^F438        BPL     DOPEN8
F441  A46A              LDY     RAMTOP      ;DO TXTMSC=$2C40 (IF MEMTOP=3000)
F443  88                DEY
F444  8C9502            STY     TXTMSC+1
F447  A960              LDA     #$60
F449  8D9402            STA     TXTMSC
F44C  A657              LDX     DINDEX
F44E  BD69FE            LDA     ANCONV,X    ;CONVERT IT TO ANTIC CODE
F451  D004 ^F457        BNE     DOPENA      ;IF ZERO, IT IS ILLEGAL
F453  A991      OPNERR: LDA     #BADMOD     ;SET ERROR STATUS
F455  854C              STA     DSTAT
F457  8551      DOPENA: STA     HOLD1
F459  A56A              LDA     RAMTOP      ;SET UP AN INDIRECT POINTER
F45B  8565              STA     ADRESS+1
F45D  BC45FE            LDY     ALOCAT,X    ;ALLOCATE N BLOCKS OF 40 BYTES
F460  A928      DOPEN1: LDA     #40
F462  2021F9            JSR     DBSUB
F465  88                DEY
F466  D0F8 ^F460        BNE     DOPEN1
```

```
F468  AD6F02              LDA     GPRIOR      ;CLEAR GTIA MODES
F46B  293F                AND     #$3F
F46D  8567                STA     OPNTMP+1
F46F  A8                  TAY
F470  E008                CPX     #8          ;TEST IF 320X1
F472  9017 ^F48B          BCC     NOT8
F474  8A                  TXA                 ;GET 2 LOW BITS
F475  6A                  ROR     A
F476  6A                  ROR     A
F477  6A                  ROR     A
F478  29C0                AND     #$C0        ;NOW 2 TOP BITS
F47A  0567                ORA     OPNTMP+1
F47C  A8                  TAY
F47D  A910                LDA     #16         ;SUBTRACT 16 MORE FOR PAGE BOUNDARY
F47F  2021F9              JSR     DBSUB
F482  E00B                CPX     #11         ;TEST MODE 11
F484  D005 ^F48B          BNE     NOT8        ;IF MODE = 11
F486  A906                LDA     #6          ;PUT GTIA LUM VALUE INTO BACKGROUND REGISTER
F488  8DC802              STA     COLOR4
F48B  8C6F02      NOT8:   STY     GPRIOR      ;STORE NEW PRIORITY
F48E  A564                LDA     ADRESS      ;SAVE MEMORY SCAN COUNTER ADDRESS
F490  8558                STA     SAVMSC
F492  A565                LDA     ADRESS+1
F494  8559                STA     SAVMSC+1
F496  AD0BD4      VBWAIT: LDA     VCOUNT      ;WAIT FOR NEXT VBLANK BEFORE MESSING
F499  C97A                CMP     #$7A        ;WITH THE DISPLAY LIST
F49B  D0F9 ^F496          BNE     VBWAIT
F49D  201FF9              JSR     DBDEC       ;START PUTTING DISPLAY LIST RIGHT UNDER RAM
F4A0  BD75FE              LDA     PAGETB,X    ;TEST IF DISPLAY LIST WILL BE IN TROUBLE
F4A3  F006 ^F4AB          BEQ     NOMOD       ;OF CROSSING A 256 BYTE PAGE BOUNDARY
F4A5  A9FF                LDA     #$FF        ;IF SO, DROP DOWN A PAGE
F4A7  8564                STA     ADRESS
F4A9  C665                DEC     ADRESS+1
F4AB  A564        NOMOD:  LDA     ADRESS      ;SAVE END OF DISPLAY LIST FOR LATER
F4AD  8568                STA     SAVADR
F4AF  A565                LDA     ADRESS+1
F4B1  8569                STA     SAVADR+1
F4B3  2013F9              JSR     DBDDEC      ;(DOUBLE BYTE DOUBLE DECREMENT)
F4B6  A941                LDA     #$41        ;(ANTIC) WAIT FOR VBLANK AND JMP TO TOP
F4B8  2017F9              JSR     STORE
F4BB  8666                STX     OPNTMP
F4BD  A918                LDA     #24         ;INITIALIZE BOTSCR
F4BF  8DBF02              STA     BOTSCR
F4C2  A557                LDA     DINDEX      ;DISALLOW MIXED MODE IF MODE.GE.9
F4C4  C909                CMP     #9
F4C6  B02D ^F4F5          BCS     NOTMXD
F4C8  A52A                LDA     ICAX1Z      ;TEST MIXED MODE
F4CA  2910                AND     #$10
F4CC  F027 ^F4F5          BEQ     NOTMXD
F4CE  A904                LDA     #4
F4D0  8DBF02              STA     BOTSCR
F4D3  A202                LDX     #2          ;ADD 4 LINES OF TEXT AT BOTTOM OF SCREEN
F4D5  A902        DOPEN2: LDA     #2
F4D7  2017F9              JSR     STORE
F4DA  CA                  DEX
F4DB  10F8 ^F4D5          BPL     DOPEN2
F4DD  A46A                LDY     RAMTOP      ;RELOAD MSC FOR TEXT
```

```
F4DF  88                DEY
F4E0  98                TYA
F4E1  2017F9            JSR     STORE
F4E4  A960              LDA     #$60
F4E6  2017F9            JSR     STORE
F4E9  A942              LDA     #$42
F4EB  2017F9            JSR     STORE
F4EE  18                CLC
F4EF  A90C              LDA     #MXDMDE-NUMDLE ;POINT X AT MIXED MODE TABLE
F4F1  6566              ADC     OPNTMP
F4F3  8566              STA     OPNTMP
F4F5  A466        NOTMXD: LDY   OPNTMP
F4F7  BE51FE            LDX     NUMDLE,Y    ;GET NUMBER OF DISPLAY LIST ENTRIES
F4FA  A551        DOPEN3: LDA   HOLD1       ;STORE N DLE'S
F4FC  2017F9            JSR     STORE
F4FF  CA                DEX
F500  D0F8 ^F4FA        BNE     DOPEN3
F502  A557              LDA     DINDEX      ;DO THE MESSY 320X1 PROBLEM
F504  C908              CMP     #8
F506  901C ^F524        BCC     DOPEN5
F508  A25D              LDX     #93         ;GET REMAINING NUMBER OF DLE'S
F50A  A56A              LDA     RAMTOP      ;RELOAD MEMORY SCAN COUNTER
F50C  38                SEC
F50D  E910              SBC     #$10
F50F  2017F9            JSR     STORE
F512  A900              LDA     #0
F514  2017F9            JSR     STORE
F517  A94F              LDA     #$4F        ;(ANTIC) RELOAD MSC CODE
F519  2017F9            JSR     STORE
F51C  A551        DOPEN4: LDA   HOLD1       ;DO REMAINING DLE'S
F51E  2017F9            JSR     STORE
F521  CA                DEX
F522  D0F8 ^F51C        BNE     DOPEN4
F524  A559        DOPEN5: LDA   SAVMSC+1    ;POLISH OFF DISPLAY LIST
F526  2017F9            JSR     STORE
F529  A558              LDA     SAVMSC
F52B  2017F9            JSR     STORE
F52E  A551              LDA     HOLD1
F530  0940              ORA     #$40
F532  2017F9            JSR     STORE
F535  A970              LDA     #$70        ;24 BLANK LINES
F537  2017F9            JSR     STORE
F53A  A970              LDA     #$70
F53C  2017F9            JSR     STORE
F53F  A564              LDA     ADRESS      ;SAVE DISPLAY LIST ADDRESS
F541  8D3002            STA     SDLSTL
F544  A565              LDA     ADRESS+1
F546  8D3102            STA     SDLSTL+1
F549  A970              LDA     #$70        ;ADD LAST BLANK LINE ENTRY
F54B  2017F9            JSR     STORE       ;POSITION ADRESS=SDLSTL-1
F54E  A564              LDA     ADRESS      ;STORE NEW MEMTOP
F550  8DE502            STA     MEMTOP
F553  A565              LDA     ADRESS+1
F555  8DE602            STA     MEMTOP+1
F558  A568              LDA     SAVADR
F55A  8564              STA     ADRESS
F55C  A569              LDA     SAVADR+1
```

```
F55E  8565                STA     ADRESS+1
F560  AD3102              LDA     SDLSTL+1
F563  2017F9              JSR     STORE
F566  AD3002              LDA     SDLSTL
F569  2017F9              JSR     STORE
F56C  A54C                LDA     DSTAT       ;IF ERROR OCURRED ON ALLOCATION, OPEN THE EDITOR
F56E  1007 ^F577          BPL     DOPEN9
F570  48                  PHA                 ;SAVE STATUS
F571  20FCF3              JSR     EOPEN       ;OPEN THE EDITOR
F574  68                  PLA                 ;RESTORE STATUS
F575  A8                  TAY                 ;AND RETURN IT TO CIO
F576  60                  RTS
F577  A52A        DOPEN9: LDA     ICAX1Z      ;TEST CLEAR INHIBIT BIT
F579  2920                AND     #$20
F57B  D00B ^F588          BNE     DOPEN7
F57D  20B9F7              JSR     CLRSCR      ;CLEAR SCREEN
F580  8D9002              STA     TXTROW      ;AND HOME TEXT CURSOR (AC IS ZERO)
F583  A552                LDA     LMARGN
F585  8D9102              STA     TXTCOL
F588  A922        DOPEN7: LDA     #$22        ;EVERYTHING ELSE IS SET UP
F58A  0D2F02              ORA     SDMCTL      ;SO TURN ON DMACTL
F58D  8D2F02              STA     SDMCTL
F590  4C21F6              JMP     RETUR2
                  ;
                  ;
F593  2096FA      GETCH:  JSR     RANGE       ;GETCH DOES INCRSR, GETPLT DOESN'T
F596  20A2F5              JSR     GETPLT
F599  2032FB              JSR     INATAC      ;CONVERT INTERNAL CODE TO ATASCII
F59C  20D4F9              JSR     INCRSB
F59F  4C34F6              JMP     RETUR1
F5A2  2047F9      GETPLT: JSR     CONVRT      ;CONVERT ROW/COLUMN TO ADRESS
F5A5  B164                LDA     (ADRESS),Y
F5A7  2DA002              AND     DMASK
F5AA  466F        SHIFTD: LSR     SHFAMT      ;SHIFT DATA DOWN TO LOW BITS
F5AC  B003 ^F5B1          BCS     SHIFT1
F5AE  4A                  LSR     A
F5AF  10F9 ^F5AA          BPL     SHIFTD      ;(UNCONDITIONAL)
F5B1  8DFA02      SHIFT1: STA     CHAR
F5B4  C900                CMP     #0          ;RESTORE FLAGS ALSO
F5B6  60                  RTS
                  ;
                  ;
                  ;
F5B7  8DFB02      OUTCH:  STA     ATACHR
F5BA  2096FA              JSR     RANGE
                  ;       JSR     OFFCRS
F5BD  ADFB02      OUTCHA: LDA     ATACHR      ;TEST FOR CLEAR SCREEN
F5C0  C97D                CMP     #CLRCOD
F5C2  D006 ^F5CA          BNE     OUTCHE
F5C4  20B9F7              JSR     CLRSCR
F5C7  4C21F6              JMP     RETUR2
F5CA  ADFB02      OUTCHE: LDA     ATACHR      ;TEST FOR CARRIAGE RETURN
F5CD  C99B                CMP     #CR
F5CF  D006 ^F5D7          BNE     OUTCHB
F5D1  2030FA              JSR     DOCRWS      ;DO CR
F5D4  4C21F6              JMP     RETUR2
F5D7  20E0F5      OUTCHB: JSR     OUTPLT
```

```
F5DA  20D8F9              JSR     INCRSR
F5DD  4C21F6              JMP     RETUR2
                  ;
                  ;
F5E0  ADFF02      OUTPLT: LDA     SSFLAG      ;*****LOOP HERE IF START/STOP FLAG IS NON-0
F5E3  D0FB ^F5E0          BNE     OUTPLT
F5E5  A202                LDX     #2
F5E7  B554        CRLOOP: LDA     ROWCRS,X    ;SAVE CURSOR LOCATION FOR DRAW LINE TO DRAW FROM
F5E9  955A                STA     OLDROW,X
F5EB  CA                  DEX
F5EC  10F9 ^F5E7          BPL     CRLOOP
F5EE  ADFB02              LDA     ATACHR      ;CONVERT ATASCII(ATACHR) TO INTERNAL(CHAR)
F5F1  A8                  TAY                 ;SAVE ATACHR
F5F2  2A                  ROL     A
F5F3  2A                  ROL     A
F5F4  2A                  ROL     A
F5F5  2A                  ROL     A
F5F6  2903                AND     #3
F5F8  AA                  TAX                 ;X HAS INDEX INTO ATAINT
F5F9  98                  TYA                 ;RESTORE ATACHR
F5FA  299F                AND     #$9F        ;STRIP OFF COLUMN ADDRESS
F5FC  1DF6FE              ORA     ATAINT,X    ;OR IN NEW COLUMN ADDRESS
F5FF  8DFA02      OUTCH2: STA     CHAR
F602  2047F9              JSR     CONVRT
F605  ADFA02              LDA     CHAR
F608  466F        SHIFTU: LSR     SHFAMT      ;SHIFT UP TO PROPER POSITION
F60A  B004 ^F610          BCS     SHIFT2
F60C  0A                  ASL     A
F60D  4C08F6              JMP     SHIFTU
F610  2DA002      SHIFT2: AND     DMASK
F613  8550                STA     TMPCHR      ;SAVE SHIFTED DATA
F615  ADA002              LDA     DMASK       ;INVERT MASK
F618  49FF                EOR     #$FF
F61A  3164                AND     (ADRESS),Y  ;MASK OFF OLD DATA
F61C  0550                ORA     TMPCHR      ;OR IN NEW DATA
F61E  9164                STA     (ADRESS),Y
F620  60                  RTS
                  ;
                  ;
F621  20A2F5      RETUR2: JSR     GETPLT      ;DO CURSOR ON THE WAY OUT
F624  855D                STA     OLDCHR
F626  A657                LDX     DINDEX      ;GRAPHICS HAVE INVISIBLE CURSOR
F628  D00A ^F634          BNE     RETUR1
F62A  AEF002              LDX     CRSINH      ;TEST CURSOR INHIBIT
F62D  D005 ^F634          BNE     RETUR1
F62F  4980                EOR     #$80        ;TOGGLE MSB
F631  20FFF5              JSR     OUTCH2      ;DISPLAY IT
F634  A44C        RETUR1: LDY     DSTAT       ;RETURN TO CIO WITH STATUS IN Y
F636  A901                LDA     #SUCCES
F638  854C                STA     DSTAT       ;SET STATUS= SUCCESSFUL COMPLETION
F63A  ADFB02              LDA     ATACHR      ;PUT ATACHR IN AC FOR RETURN TO CIO
F63D  60          NOFUNC: RTS                 ;(NON-EXISTENT FUNCTION RETURN POINT)
                  ;
                  ;
                  ;
                  ; END OF DISPLAY HANDLER
                  ;
```

```
                        ;
                        ;
                        ;
                        ;
F63E  20B3FC      EGETCH: JSR     SWAP
F641  2088FA              JSR     ERANGE
F644  A56B                LDA     BUFCNT      ;ANYTHING IN THE BUFFER?
F646  D034 ^F67C          BNE     EGETC3      ;YES
F648  A554                LDA     ROWCRS      ;NO, SO SAVE BUFFER START ADDRESS
F64A  856C                STA     BUFSTR
F64C  A555                LDA     COLCRS
F64E  856D                STA     BUFSTR+1
F650  20E2F6      EGETC1: JSR     KGETCH      ;LET'S FILL OUR BUFFER
F653  844C                STY     DSTAT       ;SAVE KEYBOARD STATUS
F655  ADFB02              LDA     ATACHR      ;TEST FOR CR
F658  C99B                CMP     #CR
F65A  F012 ^F66E          BEQ     EGETC2
F65C  20ADF6              JSR     DOSS        ;NO, SO PRINT IT
F65F  20B3FC              JSR     SWAP        ;JSR DOSS DID SWAP SO SWAP BACK
F662  A563                LDA     LOGCOL      ;BEEP IF NEARING LOGICAL COL 120
F664  C971                CMP     #113
F666  D003 ^F66B          BNE     EGETC6
F668  200AF9              JSR     BELL
F66B  4C50F6      EGETC6: JMP     EGETC1
F66E  20E4FA      EGETC2: JSR     OFFCRS      ;GET BUFFER COUNT
F671  2000FC              JSR     DOBUFC
F674  A56C                LDA     BUFSTR      ;RETURN A CHARACTER
F676  8554                STA     ROWCRS
F678  A56D                LDA     BUFSTR+1
F67A  8555                STA     COLCRS
F67C  A56B        EGETC3: LDA     BUFCNT
F67E  F011 ^F691          BEQ     EGETC5
F680  C66B        EGETC7: DEC     BUFCNT      ;AND RETURN TILL BUFCNT=0
F682  F00D ^F691          BEQ     EGETC5
F684  A54C                LDA     DSTAT       ;IF ERR, LOOP ON EGETC7 UNTIL BUFR IS EMPTIED
F686  30F8 ^F680          BMI     EGETC7
F688  2093F5              JSR     GETCH
F68B  8DFB02              STA     ATACHR
F68E  4CB3FC              JMP     SWAP        ;AND RETURN WITHOUT TURNING CURSOR BACK ON
F691  2030FA      EGETC5: JSR     DOCRWS      ;DO REAL CARRIAGE RETURN
F694  A99B                LDA     #CR         ;AND RETURN EOL
F696  8DFB02              STA     ATACHR
F699  2021F6              JSR     RETUR2      ;TURN ON CURSOR THEN SWAP
F69C  844C                STY     DSTAT       ;SAVE KEYBOARD STATUS
F69E  4CB3FC              JMP     SWAP        ;AND RETURN THROUGH RETUR1
                        ;
F6A1  6C6400      JSRIND: JMP     (ADRESS)    ;JSR TO THIS CAUSES JSR INDIRECT
                        ;
F6A4  8DFB02      EOUTCH: STA     ATACHR      ;SAVE ATASCII VALUE
F6A7  20B3FC              JSR     SWAP
F6AA  2088FA              JSR     ERANGE
F6AD  20E4FA      DOSS:   JSR     OFFCRS      ;TURN OFF CURSOR
F6B0  208DFC              JSR     TSTCTL      ;TEST FOR CONTROL CHARACTERS (Z=1 IF CTL)
F6B3  F009 ^F6BE          BEQ     EOUTC5
F6B5  0EA202      EOUTC6: ASL     ESCFLG      ;ESCFLG ONLY WORKS ONCE
F6B8  20CAF5              JSR     OUTCHE
F6BB  4CB3FC      ERETN:  JMP     SWAP        ;AND RETURN THROUGH RETUR1
```

```
F6BE  ADFE02      EOUTC5: LDA     DSPFLG      ;DO DSPFLG AND ESCFLG
F6C1  0DA202              ORA     ESCFLG
F6C4  D0EF ^F6B5          BNE     EOUTC6      ;IF NON-0 DISPLAY RATHER THAN EXECUTE IT
F6C6  0EA202              ASL     ESCFLG
F6C9  E8                  INX                 ;PROCESS CONTROL CHARACTERS
F6CA  BDC6FE              LDA     CNTRLS,X    ;GET DISPLACEMENT INTO ROUTINE
F6CD  8564                STA     ADRESS
F6CF  BDC7FE              LDA     CNTRLS+1,X  ;GET HIGH BYTE
F6D2  8565                STA     ADRESS+1
F6D4  20A1F6              JSR     JSRIND      ;DO COMPUTED JSR
F6D7  2021F6              JSR     RETUR2      ;DO CURSOR
F6DA  4CB3FC              JMP     SWAP        ;ALL DONE SO RETURN THROUGH RETUR1
                    ;
                    ;
                    ;
                    ;
                    ; END SCREEN EDITOR.
                    ;
                    ;
                    ; BEGIN KEYBOARD HANDLER
                    ;
                    ;
                    ;
F6DD  A9FF          KGETC2: LDA     #$FF
F6DF  8DFC02              STA     CH
F6E2  A52A          KGETCH: LDA     ICAX1Z      ;TEST LSB OF AUX1 FOR SPECIAL EDITOR READ MODE
F6E4  4A                  LSR     A
F6E5  B062 ^F749          BCS     GETOUT
F6E7  A980                LDA     #BRKABT
F6E9  A611                LDX     BRKKEY      ;TEST BREAK
F6EB  F058 ^F745          BEQ     K7          ;IF BREAK, PUT BRKABT IN DSTAT AND CR IN ATACHR
F6ED  ADFC02              LDA     CH
F6F0  C9FF                CMP     #$FF
F6F2  F0EE ^F6E2          BEQ     KGETCH
F6F4  857C                STA     HOLDCH      ;SAVE CH FOR SHIFT LOCK PROC
F6F6  A2FF                LDX     #$FF        ;"CLEAR" CH
F6F8  8EFC02              STX     CH
F6FB  20D8FC              JSR     CLICK       ;DO KEYBOARD AUDIO FEEDBACK (A IS OK)
F6FE  AA            KGETC3: TAX                 ;DO ASCCON
F6FF  E0C0                CPX     #$C0        ;TEST FOR CTL & SHIFT TOGETHER
F701  9002 ^F705          BCC     ASCCO1
F703  A203                LDX     #3          ;BAD CODE
F705  BDFEFE        ASCCO1: LDA     ATASCI,X
F708  8DFB02              STA     ATACHR      ;DONE
F70B  C980                CMP     #$80        ;DO NULLS
F70D  F0CE ^F6DD          BEQ     KGETC2
F70F  C981                CMP     #$81        ;CHECK ATARI KEY
F711  D00B ^F71E          BNE     KGETC1
F713  ADB602              LDA     INVFLG
F716  4980                EOR     #$80
F718  8DB602              STA     INVFLG
F71B  4CDDF6              JMP     KGETC2      ;DONT RETURN A VALUE
F71E  C982          KGETC1: CMP     #$82        ;CAPS/LOWER
F720  D007 ^F729          BNE     K1
F722  A900                LDA     #0          ;CLEAR SHFLOK
F724  8DBE02              STA     SHFLOK
```

```
F727  F0B4 ^F6DD          BEQ     KGETC2
F729  C983        K1:     CMP     #$83        ;SHIFT CAPS/LOWER
F72B  D007 ^F734          BNE     K2
F72D  A940                LDA     #$40
F72F  8DBE02              STA     SHFLOK      ;SHIFT BIT
F732  D0A9 ^F6DD          BNE     KGETC2
F734  C984        K2:     CMP     #$84        ;CNTL CAPS/LOWER
F736  D007 ^F73F          BNE     K3
F738  A980                LDA     #$80        ;CNTL BIT
F73A  8DBE02              STA     SHFLOK
F73D  D09E ^F6DD          BNE     KGETC2
F73F  C985        K3:     CMP     #$85        ;DO EOF
F741  D00A ^F74D          BNE     K6
F743  A988                LDA     #EOFERR
F745  854C        K7:     STA     DSTAT
F747  8511                STA     BRKKEY      ;RESTORE BREAK
F749  A99B        GETOUT: LDA     #CR         ;PUT CR IN ATACHR
F74B  D026 ^F773          BNE     K8          ;(UNCONDITIONAL)
F74D  A57C        K6:     LDA     HOLDCH      ;PROCESS SHIFT LOCKS
F74F  C940                CMP     #$40        ;REGULAR SHIFT AND CONTROL TAKE PRECEDENCE
F751  B015 ^F768          BCS     K5          ;OVER LOCK
F753  ADFB02              LDA     ATACHR      ;TEST FOR ALPHA
F756  C961                CMP     #$61        ;LOWER CASE A
F758  900E ^F768          BCC     K5          ;NOT ALPHA IF LT
F75A  C97B                CMP     #$7B        ;LOWER CASE Z+1
F75C  B00A ^F768          BCS     K5          ;NOT ALPHA IF GE
F75E  ADBE02              LDA     SHFLOK      ;DO SHIFT/CONTROL LOCK
F761  F005 ^F768          BEQ     K5          ;IF NO LOCK, DONT RE-DO IT
F763  057C                ORA     HOLDCH
F765  4CFEF6              JMP     KGETC3      ;DO RETRY
F768  208DFC      K5:     JSR     TSTCTL      ;DONT INVERT MSB OF CONTROL CHARACTERS
F76B  F009 ^F776          BEQ     K4
F76D  ADFB02              LDA     ATACHR
F770  4DB602              EOR     INVFLG
F773  8DFB02      K8:     STA     ATACHR
F776  4C34F6      K4:     JMP     RETUR1      ;ALL DONE
                  ;
                  ;
```

```
                        ;
                        ;
                        ; CONTROL CHARACTER PROCESSORS
                        ;
F779  A980      ESCAPE: LDA     #$80        ;SET ESCAPE FLAG
F77B  8DA202            STA     ESCFLG
F77E  60                RTS
F77F  C654      CRSRUP: DEC     ROWCRS
F781  1006 ^F789        BPL     COMRET
F783  AEBF02            LDX     BOTSCR      ;WRAPAROUND
F786  CA                DEX
F787  8654      UPDNCM: STX     ROWCRS
F789  4C5CFC    COMRET: JMP     STRBEG      ;COLVERT ROW AND COL TO LOGCOL AND RETURN
F78C  E654      CRSRDN: INC     ROWCRS
F78E  A554              LDA     ROWCRS
F790  CDBF02            CMP     BOTSCR
F793  90F4 ^F789        BCC     COMRET
F795  A200              LDX     #0
F797  F0EE ^F787        BEQ     UPDNCM      ;(UNCONDITIONAL)
F799  C655      CRSRLF: DEC     COLCRS
F79B  A555              LDA     COLCRS
F79D  3004 ^F7A3        BMI     CRSRL1      ;(IF LMARGN=0, THIS ELIMINATES PROBLEM CASE)
F79F  C552              CMP     LMARGN
F7A1  B004 ^F7A7        BCS     COMRE1
F7A3  A553      CRSRL1: LDA     RMARGN
F7A5  8555      LFRTCM: STA     COLCRS
F7A7  4CDDFB    COMRE1: JMP     DOLCOL      ;COLVERT ROW AND COL TO LOGCOL AND RETURN
F7AA  E655      CRSRRT: INC     COLCRS
F7AC  A555              LDA     COLCRS
F7AE  C553              CMP     RMARGN
F7B0  90F5 ^F7A7        BCC     COMRE1
F7B2  F0F3 ^F7A7        BEQ     COMRE1      ;(CAUSE BLE)
F7B4  A552              LDA     LMARGN
F7B6  4CA5F7            JMP     LFRTCM      ;UNCONDITIONAL TO COMMON STORE
F7B9  20F3FC    CLRSCR: JSR     PUTMSC
F7BC  A000              LDY     #0
F7BE  98                TYA                 ;PUT 0 IN THE AC
F7BF  9164      CLRSC2: STA     (ADRESS),Y  ;(AC IS ZERO)
F7C1  C8                INY
F7C2  D0FB ^F7BF        BNE     CLRSC2
F7C4  E665              INC     ADRESS+1
F7C6  A665              LDX     ADRESS+1
F7C8  E46A              CPX     RAMTOP
F7CA  90F3 ^F7BF        BCC     CLRSC2
F7CC  A9FF              LDA     #$FF        ;CLEAN UP LOGICAL LINE BIT MAP
F7CE  99B202    CLRSC3: STA     LOGMAP,Y    ;(Y IS ZERO AFTER CLRSC2 LOOP)
F7D1  C8                INY
F7D2  C004              CPY     #4
F7D4  90F8 ^F7CE        BCC     CLRSC3
F7D6  20E4FC    HOME:   JSR     COLCR       ;PLACE COLCRS AT LEFT EDGE
F7D9  8563              STA     LOGCOL
F7DB  856D              STA     BUFSTR+1
F7DD  A900              LDA     #0
F7DF  8554              STA     ROWCRS
F7E1  8556              STA     COLCRS+1
F7E3  856C              STA     BUFSTR
F7E5  60                RTS
```

```
                        ;
F7E6  A563        BS:   LDA   LOGCOL    ;BACKSPACE
F7E8  C552              CMP   LMARGN
F7EA  F021 ^F80D        BEQ   BS1
F7EC  A555        BSA:  LDA   COLCRS    ;LEFT EDGE?
F7EE  C552              CMP   LMARGN
F7F0  D003 ^F7F5        BNE   BS3       ;NO
F7F2  2073FC            JSR   DELTIM    ;YES, SEE IF LINE SHOULD BE DELETED
F7F5  2099F7      BS3:  JSR   CRSRLF
F7F8  A555              LDA   COLCRS
F7FA  C553              CMP   RMARGN
F7FC  D007 ^F805        BNE   BS2
F7FE  A554              LDA   ROWCRS
F800  F003 ^F805        BEQ   BS2
F802  207FF7            JSR   CRSRUP
F805  A920        BS2:  LDA   #$20      ;MAKE BACKSPACE DESTRUCTIVE
F807  8DFB02            STA   ATACHR
F80A  20E0F5            JSR   OUTPLT
F80D  4CDDFB      BS1:  JMP   DOLCOL    ;AND RETURN
F810  20AAF7      TAB:  JSR   CRSRRT    ;BEGIN SEARCH
F813  A555              LDA   COLCRS    ;TEST FOR NEW LINE
F815  C552              CMP   LMARGN
F817  D00A ^F823        BNE   TAB1      ;NO
F819  2034FA            JSR   DOCR      ;DO CARRIAGE RETURN
F81C  2020FB            JSR   LOGGET    ;CHECK IF END OF LOGICAL LINE
F81F  9002 ^F823        BCC   TAB1      ;NO, CONTINUE
F821  B007 ^F82A        BCS   TAB2      ;(UNCONDITIONAL)
F823  A563        TAB1: LDA   LOGCOL    ;CHECK FOR TAB STOP
F825  2025FB            JSR   BITGET
F828  90E6 ^F810        BCC   TAB       ;NO, SO KEEP LOOKING
F82A  4CDDFB      TAB2: JMP   DOLCOL    ;COLVERT ROW AND COL TO LOGCOL AND RETURN
F82D  A563        SETTAB: LDA LOGCOL
F82F  4C06FB            JMP   BITSET    ;SET BIT IN MAP AND RETURN
F832  A563        CLRTAB: LDA LOGCOL
F834  4C12FB            JMP   BITCLR    ;CLEAR " " " " "
F837  209DFC      INSCHR: JSR   PHACRS
F83A  20A2F5            JSR   GETPLT    ;GET CHARACTER UNDER CURSOR
F83D  857D              STA   INSDAT
F83F  A900              LDA   #0
F841  8DBB02            STA   SCRFLG
F844  20FFF5      INSCH4: JSR   OUTCH2    ;STORE DATA
F847  A563              LDA   LOGCOL    ;SAVE LOGCOL: IF AFTER INCRSA LOGCOL IS
F849  48                PHA             ;< THAN IT IS NOW, END LOOP
F84A  20DCF9            JSR   INCRSA    ;SPECIAL INCRSR ENTRY POINT
F84D  68                PLA
F84E  C563              CMP   LOGCOL
F850  B00C ^F85E        BCS   INSCH3    ;QUIT
F852  A57D        INSCH1: LDA   INSDAT    ;KEEP GOING
F854  48                PHA
F855  20A2F5            JSR   GETPLT
F858  857D              STA   INSDAT
F85A  68                PLA
F85B  4C44F8            JMP   INSCH4
F85E  20A8FC      INSCH3: JSR   PLACRS
F861  CEBB02      INSCH6: DEC   SCRFLG
F864  3004 ^F86A        BMI   INSCH5    ;IF SCROLL OCCURRED
F866  C654              DEC   ROWCRS    ;MOVE CURSOR UP
```

```
F868  D0F7 ^F861          BNE     INSCH6      ;(UNCOND) CONTINUE UNTIL SCRFLG IS MINUS
F86A  4CDDFB      INSCH5: JMP     DOLCOL      ;COLVERT ROW AND COL TO LOGCOL AND RETURN
                  ;
                  ;
F86D  209DFC      DELCHR: JSR     PHACRS
F870  2047F9      DELCH1: JSR     CONVRT      ;GET DATA TO THE RIGHT OF THE CURSOR
F873  A564                LDA     ADRESS
F875  8568                STA     SAVADR      ;SAVE ADRESS TO KNOW WHERE TO PUT DATA
F877  A565                LDA     ADRESS+1
F879  8569                STA     SAVADR+1
F87B  A563                LDA     LOGCOL
F87D  48                  PHA
F87E  20D4F9              JSR     INCRSB      ;PUT CURSOR OVER NEXT CHARACTER
F881  68                  PLA
F882  C563                CMP     LOGCOL      ;TEST NEW LOGCOL AGAINST OLD LOGCOL
F884  B010 ^F896          BCS     DELCH2      ;IF OLD.GE.NEW THEN QUIT
F886  A554                LDA     ROWCRS      ;IS ROW OFF SCREEN?
F888  CDBF02              CMP     BOTSCR
F88B  B009 ^F896          BCS     DELCH2      ;YES, SO QUIT
F88D  20A2F5              JSR     GETPLT      ;GET DATA UNDER CURSOR
F890  A000                LDY     #0
F892  9168                STA     (SAVADR),Y  ;PUT IT IN PREVIOUS POSITION
F894  F0DA ^F870          BEQ     DELCH1      ;AND LOOP (UNCONDITIONAL)
F896  A000        DELCH2: LDY     #0
F898  98                  TYA
F899  9168                STA     (SAVADR),Y  ;CLEAR THE LAST POSITION
F89B  2068FC              JSR     DELTIA      ;TRY TO DELETE A LINE
F89E  20A8FC              JSR     PLACRS
F8A1  4CDDFB              JMP     DOLCOL      ;AND RETURN
F8A4  38          INSLIN: SEC                 ;NORMAL INSLIN PUTS "1" INTO BIT MAP
F8A5  207BFB      INSLIA: JSR     EXTEND      ;ENTRY POINT FOR C=0
F8A8  A552                LDA     LMARGN      ;DO CARRIAGE RETURN (NO LF)
F8AA  8555                STA     COLCRS
F8AC  2047F9              JSR     CONVRT      ;GET ADDRESS
F8AF  A564                LDA     ADRESS      ;SET UP T0=40+FROM (FROM = CURSOR)
F8B1  8568                STA     FRMADR
F8B3  18                  CLC
F8B4  6928                ADC     #40
F8B6  8566                STA     TOADR
F8B8  A565                LDA     ADRESS+1
F8BA  8569                STA     FRMADR+1
F8BC  6900                ADC     #0
F8BE  8567                STA     TOADR+1
F8C0  A654                LDX     ROWCRS      ;SET UP LOOP COUNTER
F8C2  E017                CPX     #23
F8C4  F008 ^F8CE          BEQ     INSLI2
F8C6  204EFB      INSLI1: JSR     MOVLIN
F8C9  E8                  INX
F8CA  E017                CPX     #23
F8CC  D0F8 ^F8C6          BNE     INSLI1
F8CE  209BFB      INSLI2: JSR     CLRLIN      ;CLEAR CURRENT LINE
F8D1  4CDDFB              JMP     DOLCOL      ;COLVERT ROW AND COL TO LOGCOL AND RETURN
F8D4  20DDFB      DELLIN: JSR     DOLCOL      ;GET BEGINNING OF LOG LINE (HOLD1)
F8D7  A451        DELLIA: LDY     HOLD1       ;SQUEEZE BIT MAP
F8D9  8454                STY     ROWCRS      ;PUT CURSOR THERE
F8DB  A454        DELLIB: LDY     ROWCRS
F8DD  98          DELLI1: TYA
```

```
F8DE  38                SEC
F8DF  2023FB            JSR     LO2GET      ;GET NEXT BIT
F8E2  08                PHP
F8E3  98                TYA
F8E4  18                CLC
F8E5  6978              ADC     #120
F8E7  28                PLP
F8E8  2004FB            JSR     BITPUT      ;WRITE IT OVER PRESENT BIT
F8EB  C8                INY
F8EC  C018              CPY     #24
F8EE  D0ED ^F8DD        BNE     DELLI1      ;LOOP
F8F0  ADB402            LDA     LOGMAP+2    ;SET LSB
F8F3  0901              ORA     #1
F8F5  8DB402            STA     LOGMAP+2
F8F8  A552      DELLI2: LDA     LMARGN      ;DELETE LINE OF DATA USING PART OF SCROLL
F8FA  8555              STA     COLCRS      ;CR NO LF
F8FC  2047F9            JSR     CONVRT
F8FF  20B7FB            JSR     SCROL1
F902  2020FB            JSR     LOGGET      ;TEST NEXT LINE FOR CONTINUATION
                ; IS IT A NEW LOG LINE?
F905  90D4 ^F8DB        BCC     DELLIB      ;NO SO DELETE ANOTHER
F907  4CDDFB            JMP     DOLCOL      ;YES SO DOLCOL AND RETURN
F90A  A020      BELL:   LDY     #$20
F90C  20D8FC    BELL1:  JSR     CLICK
F90F  88                DEY
F910  10FA ^F90C        BPL     BELL1
F912  60                RTS
```

```
                        ;
                        ;
                        ; ROUTINES
                        ;
                        ;
                        ; DOUBLE BYTE DECREMENT OF INDIRECT POINTER
                        ; INCLUDING DB SUBTRACT AND DB DOUBLE DECREMENT
                        ;
F913  A902      DBDDEC: LDA     #2
F915  D00A ^F921        BNE     DBSUB       ;(UNCONDITIONAL)
                        ;
                        ; STORE DATA INDIRECT AND DECREMENT POINTER
                        ; (PLACED HERE TO SAVE JMP DBDEC AFTER STORE)
F917  A44C      STORE:  LDY     DSTAT       ;RETURN ON ERROR
F919  302B ^F946        BMI     STROK
F91B  A000              LDY     #0
F91D  9164      STORE1: STA     (ADRESS),Y
                        ;       JMP     DBDEC       DECREMENT AND RETURN
                        ;
F91F  A901      DBDEC:  LDA     #1
F921  8D9E02    DBSUB:  STA     SUBTMP
F924  A54C              LDA     DSTAT       ;RETURN ON ERROR
F926  301E ^F946        BMI     STROK
F928  A564              LDA     ADRESS
F92A  38                SEC
F92B  ED9E02            SBC     SUBTMP
F92E  8564              STA     ADRESS
F930  B002 ^F934        BCS     DBSUB1
F932  C665              DEC     ADRESS+1
F934  A50F      DBSUB1: LDA     APPMHI+1    ;MAKE SURE NOTHING EVER OVERWRITES APPMHI
F936  C565              CMP     ADRESS+1
F938  900C ^F946        BCC     STROK       ;OK
F93A  D006 ^F942        BNE     STRERR      ;ERROR
F93C  A50E              LDA     APPMHI
F93E  C564              CMP     ADRESS
F940  9004 ^F946        BCC     STROK
F942  A993      STRERR: LDA     #SCRMEM     ;SHOW MEM TOO SMALL FOR SCREEN ERROR
F944  854C              STA     DSTAT
F946  60        STROK:  RTS
                        ;
                        ;
                        ;
                        ; CONVERT ROW/COLUMN CURSOR INTO REAL ADDRESS (FROM SAVMSC ON UP)
                        ;
F947  A554      CONVRT: LDA     ROWCRS      ;SAVE CURSOR
F949  48                PHA
F94A  A555              LDA     COLCRS
F94C  48                PHA
F94D  A556              LDA     COLCRS+1
F94F  48                PHA
F950  20F3FC            JSR     PUTMSC
F953  A554              LDA     ROWCRS      ;PUT 10*ROWCRS INTO MLTTMP
F955  8566              STA     MLTTMP
F957  A900              LDA     #0
F959  8567              STA     MLTTMP+1
F95B  A566              LDA     MLTTMP      ;QUICK X8
F95D  0A                ASL     A
```

```
F95E  2667                ROL     MLTTMP+1
F960  8551                STA     HOLD1       ;(SAVE 2X VALUE)
F962  A467                LDY     MLTTMP+1    ;""
F964  8C9F02              STY     HOLD2       ;""
F967  0A                  ASL     A
F968  2667                ROL     MLTTMP+1
F96A  0A                  ASL     A
F96B  2667                ROL     MLTTMP+1
F96D  18                  CLC                 ;ADD IN 2X
F96E  6551                ADC     HOLD1
F970  8566                STA     MLTTMP
F972  A567                LDA     MLTTMP+1
F974  6D9F02              ADC     HOLD2
F977  8567                STA     MLTTMP+1
F979  A657                LDX     DINDEX      ;NOW SHIFT MLTTMP LEFT DHLINE TIMES TO FINISH
F97B  BC81FE              LDY     DHLINE,X    ;MULTIPLY
F97E  88          CONVR1: DEY                 ;LOOP N TIMES
F97F  3007 ^F988          BMI     CONVR2
F981  0666                ASL     MLTTMP
F983  2667                ROL     MLTTMP+1
F985  4C7EF9              JMP     CONVR1
F988  BCA5FE      CONVR2: LDY     DIV2TB,X    ;NOW DIVIDE HCRSR TO ACCOUNT FOR PARTIAL BYTES
F98B  A555                LDA     COLCRS
F98D  A207                LDX     #7          ;* TRICKY *
F98F  88          CONVR3: DEY
F990  300A ^F99C          BMI     CONVR4
F992  CA                  DEX
F993  4656                LSR     COLCRS+1
F995  6A                  ROR     A
F996  6EA102              ROR     TMPLBT      ;SAVE LOW BITS FOR MASK
F999  4C8FF9              JMP     CONVR3
F99C  C8          CONVR4: INY                 ;SO Y IS ZERO UPON RETURN FROM THIS ROUTINE
F99D  18                  CLC
F99E  6566                ADC     MLTTMP      ;ADD SHIFTED COLCRS TO MLTTMP
F9A0  8566                STA     MLTTMP
F9A2  9002 ^F9A6          BCC     CONVR5
F9A4  E667                INC     MLTTMP+1
F9A6  38          CONVR5: SEC                 ;* TRICKY *
F9A7  6EA102      CONVR6: ROR     TMPLBT      ;SLIDE A "1" UP AGAINST LOW BITS (CONTINUE TILL X=-1)
F9AA  18                  CLC
F9AB  CA                  DEX                 ;AND FINISH SHIFT SO LOW BITS ARE
F9AC  10F9 ^F9A7          BPL     CONVR6      ;RIGHT JUSTIFIED.
F9AE  AEA102              LDX     TMPLBT      ;TMPLBT IS NOW THE INDEX INTO DMASKTB
F9B1  A566                LDA     MLTTMP      ;PREPARE FOR RETURN
F9B3  18                  CLC
F9B4  6564                ADC     ADRESS
F9B6  8564                STA     ADRESS
F9B8  855E                STA     OLDADR      ;REMEMBER THIS ADDRESS FOR CURSOR
F9BA  A567                LDA     MLTTMP+1
F9BC  6565                ADC     ADRESS+1
F9BE  8565                STA     ADRESS+1
F9C0  855F                STA     OLDADR+1
F9C2  BDB1FE              LDA     DMASKT,X
F9C5  8DA002              STA     DMASK
F9C8  856F                STA     SHFAMT
F9CA  68                  PLA
F9CB  8556                STA     COLCRS+1
```

```
F9CD  68                PLA
F9CE  8555              STA     COLCRS
F9D0  68                PLA
F9D1  8554              STA     ROWCRS
F9D3  60                RTS
                ;
                ;
                ; INCREMENT CURSOR AND DETECT BOTH END OF LINE AND END OF SCREEN
                ;
F9D4  A900      INCRSB: LDA     #0          ;NON-EXTEND ENTRY POINT
F9D6  F002 ^F9DA        BEQ     INCRSC
F9D8  A99B      INCRSR: LDA     #$9B        ;SPECIAL CASE ELIMINATOR
F9DA  857D      INCRSC: STA     INSDAT
F9DC  E663      INCRSA: INC     LOGCOL      ;(INSCHR ENTRY POINT)
F9DE  E655              INC     COLCRS
F9E0  D002 ^F9E4        BNE     INCRS2      ;DO HIGH BYTE
F9E2  E656              INC     COLCRS+1
F9E4  A555      INCRS2: LDA     COLCRS      ;TEST END OF LINE
F9E6  A657              LDX     DINDEX
F9E8  DD8DFE            CMP     COLUMN,X    ;TEST TABLED VALUE FOR ALL SCREEN MODES
F9EB  F00B ^F9F8        BEQ     INC2A       ;DO CR IF EQUAL
F9ED  E000              CPX     #0          ;MODE 0?
F9EF  D006 ^F9F7        BNE     INCRS3      ;IF NOT, JUST RETURN
F9F1  C553              CMP     RMARGN      ;TEST AGAINST RMARGN
F9F3  F002 ^F9F7        BEQ     INCRS3      ;EQUAL IS OK
F9F5  B001 ^F9F8        BCS     INC2A       ;IF GREATER THAN, DO CR
F9F7  60        INCRS3: RTS
F9F8  E008      INC2A:  CPX     #8          ;CHECK MODE
F9FA  9004 ^FA00        BCC     DOCR1       ;NOT 320X1 SO DO IT
F9FC  A556              LDA     COLCRS+1    ;TEST MSD
F9FE  F0F7 ^F9F7        BEQ     INCRS3      ;ONLY AT 64 SO DON'T DO IT
FA00  A557      DOCR1:  LDA     DINDEX      ;DON'T MESS WITH LOGMAP IF NO MODE ZERO
FA02  D030 ^FA34        BNE     DOCR
FA04  A563              LDA     LOGCOL      ;TEST LINE OVERRUN
FA06  C951              CMP     #81
FA08  900A ^FA14        BCC     DOCR1B      ;IF LESS THAN 81 IT IS DEFINITELY NOT LINE 3
FA0A  A57D              LDA     INSDAT
FA0C  F026 ^FA34        BEQ     DOCR        ;ONLY DO LOG LINE OVERFLOW IF INSDAT <>0
FA0E  2030FA            JSR     DOCRWS      ;LOG LINE OVERFLOW IS SPECIAL CASE
FA11  4C77FA            JMP     INCRS1      ;RETURN
FA14  2034FA    DOCR1B: JSR     DOCR        ;GET IT OVER WITH
FA17  A554              LDA     ROWCRS
FA19  18                CLC                 ;TEST LOGICAL LINE BIT MAP
FA1A  6978              ADC     #120
FA1C  2025FB            JSR     BITGET
FA1F  9008 ^FA29        BCC     DOCR1A      ;DON'T EXTEND IF OVERRUN IS INTO MIDDLE OF LOG LINE
FA21  A57D              LDA     INSDAT      ;DON'T EXTEND IF INSDAT IS ZERO
FA23  F004 ^FA29        BEQ     DOCR1A      ;(INSCHR SPECIAL CASE)
FA25  18                CLC                 ;INSERT "0" INTO BIT MAP
FA26  20A5F8            JSR     INSLIA
FA29  4CDDFB    DOCR1A: JMP     DOLCOL      ;CONVERT ROW AND COL TO LOGCOL AND RETURN
FA2C  A900      NOSCRL: LDA     #0          ;DOCR WITHOUT SCROLL
FA2E  F002 ^FA32        BEQ     NOSCR1      ;(UNCONDITIONAL)
FA30  A99B      DOCRWS: LDA     #$9B        ;DOCR WITH SCROLLING (NORMAL MODE)
FA32  857D      NOSCR1: STA     INSDAT
FA34  20E4FC    DOCR:   JSR     COLCR       ;PLACE COLCRS AT LEFT EDGE
FA37  A900              LDA     #0
```

```
FA39  8556              STA     COLCRS+1
FA3B  E654              INC     ROWCRS
FA3D  A657        DOCR2:LDX     DINDEX
FA3F  A018              LDY     #24         ;SET UP SCROLL LOOP COUNTER
FA41  247B              BIT     SWPFLG
FA43  1005 ^FA4A        BPL     DOCR2A      ;BRANCH IF NORMAL
FA45  A004              LDY     #4
FA47  98                TYA
FA48  D003 ^FA4D        BNE     DOCR2B      ;(UNCONDITIONAL)
FA4A  BD99FE      DOCR2A:LDA    NOROWS,X    ;GET NO OF ROWS
FA4D  C554        DOCR2B:CMP    ROWCRS
FA4F  D026 ^FA77        BNE     INCRS1
FA51  8C9D02            STY     HOLD3
FA54  8A                TXA                 ;DON'T SCROLL IF MODE <> 0
FA55  D020 ^FA77        BNE     INCRS1
FA57  A57D              LDA     INSDAT      ;OR IF INSDAT = 0
FA59  F01C ^FA77        BEQ     INCRS1
                ;       LDA     INSDAT      IF INSDAT <> $9B THEN ROLL IN A 0
FA5B  C99B              CMP     #$9B        ;TO EXTEND BOTTOM LOGICAL LINE
FA5D  38                SEC
FA5E  F001 ^FA61        BEQ     DOCR4B
FA60  18                CLC
FA61  20ACFB      DOCR4B:JSR    SCROLL      ;LOOP BACK TO HERE IF >1 SCROLLS
FA64  EEBB02            INC     SCRFLG
FA67  C66C              DEC     BUFSTR      ;ROWS MOVE UP SO BUFSTR SHOULD TOO
FA69  CE9D02            DEC     HOLD3
FA6C  ADB202            LDA     LOGMAP
FA6F  38                SEC                 ;FOR PARTIAL LINES, ROLL IN A "1"
FA70  10EF ^FA61        BPL     DOCR4B      ;AGAIN IF PARTIAL LOGICAL LINE
FA72  AD9D02            LDA     HOLD3       ;PLACE CURSOR AT NEW LINE NEAR THE BOTTOM
FA75  8554              STA     ROWCRS
FA77  4CDDFB      INCRS1:JMP    DOLCOL      ;COLVERT ROW AND COL TO LOGCOL AND RETURN
                ;
                ;
                ; SUBEND: SUBTRACT ENDPT FROM ROWAC OR COLAC. (X=0 OR 2)
                ;
FA7A  38          SUBEND:SEC
FA7B  B570              LDA     ROWAC,X
FA7D  E574              SBC     ENDPT
FA7F  9570              STA     ROWAC,X
FA81  B571              LDA     ROWAC+1,X
FA83  E575              SBC     ENDPT+1
FA85  9571              STA     ROWAC+1,X
FA87  60                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)
                ;
FA88  ADBF02      ERANGE:LDA    BOTSCR      ;IF BOTSCR=4
FA8B  C904              CMP     #4
FA8D  F007 ^FA96        BEQ     RANGE       ;THEN IT IS IN MIXED MODE AND OK
FA8F  A557              LDA     DINDEX      ;IF MODE = 0
FA91  F003 ^FA96        BEQ     RANGE       ;THEN IT IS IN EDITOR MODE AND OK
FA93  20FCF3            JSR     EOPEN       ;IF NOT, OPEN EDITOR
FA96  A927        RANGE:LDA     #39         ;***** RANGE CHECK RMARGN ***** SET UP AC
```

```
FA98  C553              CMP     RMARGN      ;***** RANGE CHECK RMARGN ***** COMPARE
FA9A  B002 ^FA9E        BCS     RANGE3      ;***** RANGE CHECK RMARGN ***** BRANCH GE
FA9C  8553              STA     RMARGN      ;***** RANGE CHECK RMARGN ***** BAD SO STORE 39
FA9E  A657        RANGE3: LDX   DINDEX
FAA0  BD99FE            LDA     NOROWS,X    ;CHECK ROWS
FAA3  C554              CMP     ROWCRS
FAA5  902A ^FAD1        BCC     RNGERR      ;(ERROR IF TABLE.GE.ROWCRS)
FAA7  F028 ^FAD1        BEQ     RNGERR
FAA9  E008              CPX     #8          ;CHECK FOR 320X1
FAAB  D00A ^FAB7        BNE     RANGE1      ;SPECIAL CASE IT
FAAD  A556              LDA     COLCRS+1
FAAF  F013 ^FAC4        BEQ     RNGOK       ;IF HIGH BYTE IS 0, COL IS OK
FAB1  C901              CMP     #1
FAB3  D01C ^FAD1        BNE     RNGERR      ;IF >1, BAD
FAB5  F004 ^FABB        BEQ     RANGE2      ;IF 1, GO CHECK LOW BYTE
FAB7  A556        RANGE1: LDA   COLCRS+1    ;FOR OTHERS, NON-ZERO HIGH BYTE IS BAD
FAB9  D016 ^FAD1        BNE     RNGERR
FABB  BD8DFE      RANGE2: LDA   COLUMN,X    ;CHECK LOW BYTE
FABE  C555              CMP     COLCRS
FAC0  900F ^FAD1        BCC     RNGERR
FAC2  F00D ^FAD1        BEQ     RNGERR
FAC4  A901        RNGOK:  LDA   #SUCCES     ;SET STATUS OK
FAC6  854C              STA     DSTAT
FAC8  A980              LDA     #BRKABT     ;PREPARE BREAK ABORT STATUS
FACA  A611              LDX     BRKKEY      ;CHECK BREAK KEY FLAG
FACC  8511              STA     BRKKEY      ;'CLEAR' BREAK
FACE  F006 ^FAD6        BEQ     RNGER2      ;IF BREAK, QUIT IMMEDIATELY AND RETURN TO CIO
FAD0  60                RTS
FAD1  20D6F7      RNGERR: JSR   HOME        ;ON RANGE ERROR, BRING CURSOR BACK
FAD4  A98D              LDA     #CRSROR     ;SHOW CURSOR OVERRANGE ERROR
FAD6  854C        RNGER2: STA   DSTAT
FAD8  68          RNGER1: PLA               ;RESTORE STACK (THIS ROUTINE IS ALWAYS 1 LEVEL
FAD9  68                  PLA               ;AWAY FROM RETURN TO CIO)
FADA  A57B                LDA     SWPFLG    ;IF SWAPPED, SWAP BACK
FADC  1003 ^FAE1          BPL     RETUR3
FADE  20B9FC              JSR     SWAPA     ;AND DONT DO RETUR1
FAE1  4C34F6      RETUR3: JMP     RETUR1    ;RETURN TO CIO
                  ;
                  ;
                  ;
                  ; OFFCRS: RESTORE OLD DATA UNDER CURSOR SO IT CAN BE MOVED
                  ;
FAE4  A000        OFFCRS: LDY     #0
FAE6  A55D                LDA     OLDCHR
FAE8  915E                STA     (OLDADR),Y
FAEA  60                  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
                        ;
FAEB  48          BITCON: PHA
FAEC  2907                AND     #7
FAEE  AA                  TAX                     ;GET MASK
FAEF  BDB9FE              LDA     MASKTB,X
FAF2  856E                STA     BITMSK
FAF4  68                  PLA                     ;PROCESS INDEX
FAF5  4A                  LSR     A
FAF6  4A                  LSR     A
FAF7  4A                  LSR     A
FAF8  AA                  TAX
FAF9  60                  RTS
                        ;
                        ;
FAFA  2EB402      BITROL: ROL     LOGMAP+2
FAFD  2EB302              ROL     LOGMAP+1
FB00  2EB202              ROL     LOGMAP
FB03  60                  RTS
                        ;
                        ;
FB04  900C ^FB12  BITPUT: BCC     BITCLR          ;AND RETURN
                        ; OTHERWISE FALL THROUGH TO BITSET AND RETURN
FB06  20EBFA      BITSET: JSR     BITCON
FB09  BDA302              LDA     TABMAP,X
FB0C  056E                ORA     BITMSK
FB0E  9DA302              STA     TABMAP,X
FB11  60                  RTS
                        ;
FB12  20EBFA      BITCLR: JSR     BITCON
FB15  A56E                LDA     BITMSK
FB17  49FF                EOR     #$FF
FB19  3DA302              AND     TABMAP,X
FB1C  9DA302              STA     TABMAP,X
FB1F  60                  RTS
                        ;
FB20  A554        LOGGET: LDA     ROWCRS
FB22  18          LO1GET: CLC
FB23  6978        LO2GET: ADC     #120
FB25  20EBFA      BITGET: JSR     BITCON
FB28  18                  CLC
FB29  BDA302              LDA     TABMAP,X
FB2C  256E                AND     BITMSK
FB2E  F001 ^FB31          BEQ     BITGE1
FB30  38                  SEC
FB31  60          BITGE1: RTS
                        ;
                        ;
                        ;
                        ;
                        ; INATAC: INTERNAL(CHAR) TO ATASCII(ATACHR) CONVERSION
                        ;
FB32  ADFA02      INATAC: LDA     CHAR
FB35  A457                LDY     DINDEX          ;IF GRAPHICS MODES
FB37  C003                CPY     #3
FB39  B00F ^FB4A          BCS     INATA1          ;THEN DON'T CHANGE CHAR
FB3B  2A                  ROL     A
```

```
FB3C  2A                ROL     A
FB3D  2A                ROL     A
FB3E  2A                ROL     A
FB3F  2903              AND     #3
FB41  AA                TAX
FB42  ADFA02            LDA     CHAR
FB45  299F              AND     #$9F
FB47  1DFAFE            ORA     INTATA,X
FB4A  8DFB02    INATA1: STA     ATACHR
FB4D  60                RTS
                ;
                ;
                ;
                ; MOVLIN: MOVE 40 BYTES AT FRMADR TO TOADR SAVING OLD TOADR
                ;       DATA IN THE LINBUF. THEN MAKE NEXT FRMADR
                ;       BE AT LINBUF FOR NEXT TRANSFER & TOADR=TOADR+40
                ;
FB4E  A902      MOVLIN: LDA     #HIGH LINBUF;SET UP ADRESS=LINBUF=$247
FB50  8565              STA     ADRESS+1
FB52  A947              LDA     #LOW LINBUF
FB54  8564              STA     ADRESS
FB56  A027              LDY     #39
FB58  B166      MOVLI1: LDA     (TOADR),Y   ;SAVE TO DATA
FB5A  8550              STA     TMPCHR
FB5C  B168              LDA     (FRMADR),Y  ;STORE DATA
FB5E  9166              STA     (TOADR),Y
FB60  A550              LDA     TMPCHR
FB62  9164              STA     (ADRESS),Y
FB64  88                DEY
FB65  10F1 ^FB58        BPL     MOVLI1
FB67  A565              LDA     ADRESS+1    ;SET UP FRMADR=LAST LINE
FB69  8569              STA     FRMADR+1
FB6B  A564              LDA     ADRESS
FB6D  8568              STA     FRMADR
FB6F  18                CLC                 ;ADD 40 TO TOADR
FB70  A566              LDA     TOADR
FB72  6928              ADC     #40
FB74  8566              STA     TOADR
FB76  9002 ^FB7A        BCC     MOVLI2
FB78  E667              INC     TOADR+1
FB7A  60        MOVLI2: RTS
                ;
                ;
                ;
                ; EXTEND: EXTEND BIT MAP FROM ROWCRS (EXTEND LOGICAL LINE
                ;
FB7B  08        EXTEND: PHP                 ;SAVE CARRY
FB7C  A017              LDY     #23
FB7E  98        EXTEN1: TYA
FB7F  2022FB            JSR     LO1GET
FB82  08                PHP
FB83  98                TYA
FB84  18                CLC
FB85  6979              ADC     #121
FB87  28                PLP
FB88  2004FB            JSR     BITPUT
```

```
FB8B  88          EXTEN3: DEY
FB8C  3004 ^FB92          BMI     EXTEN4
FB8E  C454                CPY     ROWCRS
FB90  B0EC ^FB7E          BCS     EXTEN1
FB92  A554        EXTEN4: LDA     ROWCRS
FB94  18                  CLC
FB95  6978                ADC     #120
FB97  28                  PLP
FB98  4C04FB              JMP     BITPUT      ;STORE NEW LINE'S BIT AND RETURN
                  ;
                  ;
                  ;
                  ; CLRLIN: CLEAR LINE CURSOR IS ON
                  ;
FB9B  A552        CLRLIN: LDA     LMARGN
FB9D  8555                STA     COLCRS
FB9F  2047F9              JSR     CONVRT
FBA2  A027                LDY     #39
FBA4  A900                LDA     #0
FBA6  9164        CLRLI1: STA     (ADRESS),Y
FBA8  88                  DEY
FBA9  10FB ^FBA6          BPL     CLRLI1
FBAB  60                  RTS
                  ;
                  ;
                  ;
                  ; SCROLL: SCROLL SCREEN
                  ;
FBAC  20FAFA      SCROLL: JSR     BITROL      ;ROLL IN CARRY
FBAF  A558                LDA     SAVMSC      ;SET UP WORKING REGISTERS
FBB1  8564                STA     ADRESS
FBB3  A559                LDA     SAVMSC+1
FBB5  8565                STA     ADRESS+1
FBB7  A028        SCROL1: LDY     #40         ;LOOP
FBB9  B164                LDA     (ADRESS),Y
FBBB  A66A                LDX     RAMTOP      ;TEST FOR LAST LINE
FBBD  CA                  DEX
FBBE  E465                CPX     ADRESS+1
FBC0  D008 ^FBCA          BNE     SCROL2
FBC2  A2D7                LDX     #$D7
FBC4  E464                CPX     ADRESS
FBC6  B002 ^FBCA          BCS     SCROL2
FBC8  A900                LDA     #0          ;YES SO STORE ZERO DATA FOR THIS ENTIRE LINE
FBCA  A000        SCROL2: LDY     #0
FBCC  9164                STA     (ADRESS),Y
FBCE  E664                INC     ADRESS
FBD0  D0E5 ^FBB7          BNE     SCROL1
FBD2  E665                INC     ADRESS+1
FBD4  A565                LDA     ADRESS+1
FBD6  C56A                CMP     RAMTOP
FBD8  D0DD ^FBB7          BNE     SCROL1
FBDA  4CDDFB              JMP     DOLCOL      ;AND RETURN
                  ;
                  ;
                  ; DOLCOL: DO LOGICAL COLUMN FROM BITMAP AND COLCRS
                  ;
```

```
FBDD  A900        DOLCOL: LDA     #0          ;START WITH ZERO
FBDF  8563                STA     LOGCOL
FBE1  A554                LDA     ROWCRS
FBE3  8551                STA     HOLD1
FBE5  A551        DOLCO1: LDA     HOLD1       ;ADD IN ROW COMPONENT
FBE7  2022FB              JSR     LO1GET
FBEA  B00C ^FBF8          BCS     DOLCO2      ;FOUND BEGINNING OF LINE
FBEC  A563                LDA     LOGCOL      ;ADD 40 AND LOOK BACK ONE
FBEE  18                  CLC
FBEF  6928                ADC     #40
FBF1  8563                STA     LOGCOL
FBF3  C651                DEC     HOLD1       ;UP ONE LINE
FBF5  4CE5FB              JMP     DOLCO1
FBF8  18          DOLCO2: CLC                 ;ADD IN COLCRS
FBF9  A563                LDA     LOGCOL
FBFB  6555                ADC     COLCRS
FBFD  8563                STA     LOGCOL
FBFF  60                  RTS
                  ;
                  ;
                  ;
                  ; DOBUFC: COMPUTE BUFFER COUNT AS THE NUMBER OF BYTES FROM
                  ;         BUFSTR TO END OF LOGICAL LINE WITH TRAILING SPACES REMOVED
                  ;
FC00  209DFC      DOBUFC: JSR     PHACRS
FC03  A563                LDA     LOGCOL
FC05  48                  PHA
FC06  A56C                LDA     BUFSTR      ;START
FC08  8554                STA     ROWCRS
FC0A  A56D                LDA     BUFSTR+1
FC0C  8555                STA     COLCRS
FC0E  A901                LDA     #1
FC10  856B                STA     BUFCNT
FC12  A217        DOBUF1: LDX     #23         ;NORMAL
FC14  A57B                LDA     SWPFLG      ;IF SWAPPED, ROW 3 IS THE LAST LINE ON SCREEN
FC16  1002 ^FC1A          BPL     DOB1
FC18  A203                LDX     #3
FC1A  E454        DOB1:   CPX     ROWCRS      ;TEST IF CRSR IS AT LAST SCREEN POSITION
FC1C  D00B ^FC29          BNE     DOBU1A
FC1E  A555                LDA     COLCRS
FC20  C553                CMP     RMARGN
FC22  D005 ^FC29          BNE     DOBU1A
FC24  E66B                INC     BUFCNT      ;YES, SO FAKE INCRSR TO AVOID SCROLLING
FC26  4C39FC              JMP     DOBUF2
FC29  20D4F9      DOBU1A: JSR     INCRSB
FC2C  E66B                INC     BUFCNT
FC2E  A563                LDA     LOGCOL
FC30  C552                CMP     LMARGN
FC32  D0DE ^FC12          BNE     DOBUF1      ;NOT YET EOL
FC34  C654                DEC     ROWCRS      ;BACK UP ONE INCRSR
FC36  2099F7              JSR     CRSRLF
FC39  20A2F5      DOBUF2: JSR     GETPLT      ;TEST CURRENT COLUMN FOR NON-ZERO DATA
FC3C  D017 ^FC55          BNE     DOBUF4      ;QUIT IF NON-ZERO
FC3E  C66B                DEC     BUFCNT      ;DECREMENT COUNTER
FC40  A563                LDA     LOGCOL      ;BEGINNING OF LOGICAL LINE YET?
FC42  C552                CMP     LMARGN
FC44  F00F ^FC55          BEQ     DOBUF4      ;YES, SO QUIT
```

```
FC46  2099F7              JSR     CRSRLF      ;BACK UP CURSOR
FC49  A555                LDA     COLCRS      ;IF LOGCOL=RMARGN, GO UP 1 ROW
FC4B  C553                CMP     RMARGN
FC4D  D002 ^FC51          BNE     DOBUF3
FC4F  C654                DEC     ROWCRS
FC51  A56B        DOBUF3: LDA     BUFCNT
FC53  D0E4 ^FC39          BNE     DOBUF2      ;LOOP UNLESS BUFCNT JUST WENT TO ZERO
FC55  68          DOBUF4: PLA
FC56  8563                STA     LOGCOL
FC58  20A8FC              JSR     PLACRS
FC5B  60                  RTS
                  ;
                  ;
                  ;
                  ;
                  ; STRBEG: MOVE BUFSTR TO BEGINNING OF LOGICAL LINE.
                  ;
FC5C  20DDFB      STRBEG: JSR     DOLCOL      ;USE DOLCOL TO POINT HOLD1 AT BOL
FC5F  A551                LDA     HOLD1
FC61  856C                STA     BUFSTR
FC63  A552                LDA     LMARGN
FC65  856D                STA     BUFSTR+1
FC67  60                  RTS
                  ;
                  ;
                  ;
                  ;
                  ; DELTIM: TIME TO DELETE A LINE IF IT IS EMPTY AND AN EXTENSION
                  ;
FC68  A563        DELTIA: LDA     LOGCOL      ;IF LOGCOL<>LMARGN
FC6A  C552                CMP     LMARGN      ;THEN DONT MOVE UP ONE
FC6C  D002 ^FC70          BNE     DELTIB      ;LINE BEFORE TESTING DELTIM
FC6E  C654                DEC     ROWCRS
FC70  20DDFB      DELTIB: JSR     DOLCOL
FC73  A563        DELTIM: LDA     LOGCOL      ;TEST FOR EXTENSION
FC75  C552                CMP     LMARGN
FC77  F013 ^FC8C          BEQ     DELTI3      ;NO
FC79  2047F9              JSR     CONVRT
FC7C  A553                LDA     RMARGN      ;SET UP COUNT
FC7E  38                  SEC
FC7F  E552                SBC     LMARGN
FC81  A8                  TAY
FC82  B164        DELTI1: LDA     (ADRESS),Y
FC84  D006 ^FC8C          BNE     DELTI3      ;FOUND A NON-0 SO QUIT AND RETURN
FC86  88                  DEY
FC87  10F9 ^FC82          BPL     DELTI1
FC89  4CDBF8      DELTI2: JMP     DELLIB      ;DELETE A LINE AND RETURN
FC8C  60          DELTI3: RTS
                  ;
                  ;
                  ;
                  ; TSTCTL: SEARCH CNTRLS TABLE TO SEE IF ATACHR IS A CNTL CHAR
                  ;
FC8D  A22D        TSTCTL: LDX     #45         ;PREPARE TO SEARCH TABLE
FC8F  BDC6FE      TSTCT1: LDA     CNTRLS,X
FC92  CDFB02              CMP     ATACHR
```

```
FC95  F005 ^FC9C          BEQ     TSTCT2
FC97  CA                  DEX
FC98  CA                  DEX
FC99  CA                  DEX
FC9A  10F3 ^FC8F          BPL     TSTCT1
FC9C  60          TSTCT2: RTS
                  ;
                  ;
                  ;
                  ; PUSH ROWCRS,COLCRS AND COLCRS+1
                  ;
FC9D  A202        PHACRS: LDX     #2
FC9F  B554        PHACR1: LDA     ROWCRS,X
FCA1  9DB802              STA     TMPROW,X
FCA4  CA                  DEX
FCA5  10F8 ^FC9F          BPL     PHACR1
FCA7  60                  RTS
                  ;
                  ;
                  ; PULL COLCRS+1,COLCRS AND ROWCRS
                  ;
FCA8  A202        PLACRS: LDX     #2
FCAA  BDB802      PLACR1: LDA     TMPROW,X
FCAD  9554                STA     ROWCRS,X
FCAF  CA                  DEX
FCB0  10F8 ^FCAA          BPL     PLACR1
FCB2  60                  RTS
                  ;
                  ;
                  ;
                  ; SWAP: IF MIXED MODE, SWAP TEXT CURSORS WITH REGULAR CURSORS
                  ;
FCB3  20B9FC      SWAP:   JSR     SWAPA       ;THIS ENTRY POINT DOES RETUR1
FCB6  4C34F6              JMP     RETUR1
FCB9  ADBF02      SWAPA:  LDA     BOTSCR
FCBC  C918                CMP     #24
FCBE  F017 ^FCD7          BEQ     SWAP3
FCC0  A20B                LDX     #11
FCC2  B554        SWAP1:  LDA     ROWCRS,X
FCC4  48                  PHA
FCC5  BD9002              LDA     TXTROW,X
FCC8  9554                STA     ROWCRS,X
FCCA  68                  PLA
FCCB  9D9002              STA     TXTROW,X
FCCE  CA                  DEX
FCCF  10F1 ^FCC2          BPL     SWAP1
FCD1  A57B                LDA     SWPFLG
FCD3  49FF                EOR     #$FF
FCD5  857B                STA     SWPFLG
FCD7  60          SWAP3:  RTS
                  ;
                  ;
                  ; CLICK: MAKE CLICK THROUGH KEYBOARD SPEAKER
                  ;
FCD8  A27F        CLICK:  LDX     #$7F
FCDA  8E1FD0      CLICK1: STX     CONSOL
FCDD  8E0AD4              STX     WSYNC
```

```
FCE0  CA                DEX
FCE1  10F7 ^FCDA        BPL     CLICK1
FCE3  60                RTS
                ;
                ;
                ; COLCR: PUTS EITHER 0 OR LMARGN INTO COLCRS BASED ON MODE AND SWPFLG
                ;
FCE4  A900      COLCR:  LDA     #0
FCE6  A67B              LDX     SWPFLG
FCE8  D004 ^FCEE        BNE     COLCR1
FCEA  A657              LDX     DINDEX
FCEC  D002 ^FCF0        BNE     COLCR2
FCEE  A552      COLCR1: LDA     LMARGN
FCF0  8555      COLCR2: STA     COLCRS
FCF2  60                RTS
                ;
                ;
                ; PUTMSC: PUT SAVMSC INTO ADRESS
                ;
FCF3  A558      PUTMSC: LDA     SAVMSC      ;SET UP ADDRESS
FCF5  8564              STA     ADRESS
FCF7  A559              LDA     SAVMSC+1
FCF9  8565              STA     ADRESS+1
FCFB  60                RTS
                ;
```

```
                        ;
                        ;
                        ; DRAW -- DRAW A LINE FROM OLDROW,OLDCOL TO NEWROW,NEWCOL
                        ; (THE AL MILLER METHOD FROM BASKETBALL)
FCFC  A200      DRAW:   LDX     #0
FCFE  A522              LDA     ICCOMZ      ;TEST COMMAND: $11=DRAW $12=FILL
FD00  C911              CMP     #$11
FD02  F008 ^FD0C        BEQ     DRAWA
FD04  C912              CMP     #$12        ;TEST FILL
FD06  F003 ^FD0B        BEQ     DRAWB       ;YES
FD08  A084              LDY     #NVALID     ;NO, SO RETURN INVALID COMMAND
FD0A  60                RTS
FD0B  E8        DRAWB:  INX
FD0C  8EB702    DRAWA:  STX     FILFLG
FD0F  A554              LDA     ROWCRS      ;PUT CURSOR INTO NEWROW,NEWCOL
FD11  8560              STA     NEWROW
FD13  A555              LDA     COLCRS
FD15  8561              STA     NEWCOL
FD17  A556              LDA     COLCRS+1
FD19  8562              STA     NEWCOL+1
FD1B  A901              LDA     #1
FD1D  8579              STA     ROWINC      ;SET UP INITIAL DIRECTIONS
FD1F  857A              STA     COLINC
FD21  38                SEC
FD22  A560              LDA     NEWROW      ;DETERMINE DELTA ROW
FD24  E55A              SBC     OLDROW
FD26  8576              STA     DELTAR
FD28  B00D ^FD37        BCS     DRAW1       ;DO DIRECTION AND ABSOLUTE VALUE
FD2A  A9FF              LDA     #$FF        ;BORROW WAS ATTEMPTED
FD2C  8579              STA     ROWINC      ;SET DIRECTION=DOWN
FD2E  A576              LDA     DELTAR
FD30  49FF              EOR     #$FF        ;DELTAR = |DELTAR|
FD32  18                CLC
FD33  6901              ADC     #1
FD35  8576              STA     DELTAR
FD37  38        DRAW1:  SEC
FD38  A561              LDA     NEWCOL      ;NOW DELTA COLUMN
FD3A  E55B              SBC     OLDCOL
FD3C  8577              STA     DELTAC
FD3E  A562              LDA     NEWCOL+1    ;TWO-BYTE QUANTITY
FD40  E55C              SBC     OLDCOL+1
FD42  8578              STA     DELTAC+1
FD44  B016 ^FD5C        BCS     DRAW2       ;DIRECTION AND ABSOLUTE VALUE
FD46  A9FF              LDA     #$FF        ;BORROW WAS ATTEMPTED
FD48  857A              STA     COLINC      ;SET DIRECTION = LEFT
FD4A  A577              LDA     DELTAC
FD4C  49FF              EOR     #$FF        ;DELTAC = |DELTAC|
FD4E  8577              STA     DELTAC
FD50  A578              LDA     DELTAC+1
FD52  49FF              EOR     #$FF
FD54  8578              STA     DELTAC+1
FD56  E677              INC     DELTAC      ;ADD ONE FOR TWOS COMPLEMENT
FD58  D002 ^FD5C        BNE     DRAW2
FD5A  E678              INC     DELTAC+1
FD5C  A202      DRAW2:  LDX     #2          ;ZERO RAM FOR DRAW LOOP
FD5E  A000              LDY     #0
FD60  8473              STY     COLAC+1
```

```
FD62  98          DRAW3A: TYA
FD63  9570                STA     ROWAC,X
FD65  B55A                LDA     OLDROW,X
FD67  9554                STA     ROWCRS,X
FD69  CA                  DEX
FD6A  10F6 ^FD62          BPL     DRAW3A
FD6C  A577                LDA     DELTAC      ;FIND LARGER ONE (ROW OR COL)
                  ;       STA     COUNTR      (PREPARE COUNTR AND ENDPT)
                  ;       STA     ENDPT
FD6E  E8                  INX                 ;MAKE X 0
FD6F  A8                  TAY
FD70  A578                LDA     DELTAC+1
FD72  857F                STA     COUNTR+1
FD74  8575                STA     ENDPT+1
FD76  D00B ^FD83          BNE     DRAW3       ;AUTOMATICALLY LARGER IF MSD>0
FD78  A577                LDA     DELTAC
FD7A  C576                CMP     DELTAR      ;LOW COL >LOW ROW?
FD7C  B005 ^FD83          BCS     DRAW3       ;YES
FD7E  A576                LDA     DELTAR
FD80  A202                LDX     #2
FD82  A8                  TAY
FD83  98          DRAW3:  TYA                 ;PUT IN INITIAL CONDITIONS
FD84  857E                STA     COUNTR
FD86  8574                STA     ENDPT
FD88  48                  PHA                 ;SAVE AC
FD89  A575                LDA     ENDPT+1     ;PUT LSB OF HIGH BYTE
FD8B  4A                  LSR     A           ;INTO CARRY
FD8C  68                  PLA                 ;RESTORE AC
FD8D  6A                  ROR     A           ;ROR THE 9 BIT ACUMULATOR
FD8E  9570                STA     ROWAC,X
FD90  A57E        DRAW4A: LDA     COUNTR      ;TEST ZERO
FD92  057F                ORA     COUNTR+1
FD94  D003 ^FD99          BNE     DRAW11      ;IF COUNTER IS ZERO, LEAVE DRAW
FD96  4C42FE              JMP     DRAW10
FD99  18          DRAW11: CLC                 ;ADD ROW TO ROWAC (PLOT LOOP)
FD9A  A570                LDA     ROWAC
FD9C  6576                ADC     DELTAR
FD9E  8570                STA     ROWAC
FDA0  9002 ^FDA4          BCC     DRAW5
FDA2  E671                INC     ROWAC+1
FDA4  A571        DRAW5:  LDA     ROWAC+1     ;COMPARE ROW TO ENDPOINT
FDA6  C575                CMP     ENDPT+1     ;IF HIGH BYTE OF ROW IS .LT. HIGH
FDA8  9014 ^FDBE          BCC     DRAW6       ;BYTE OF ENDPT, BLT TO COLUMN
FDAA  D006 ^FDB2          BNE     DRAW5A
FDAC  A570                LDA     ROWAC
FDAE  C574                CMP     ENDPT       ;LOW BYTE
FDB0  900C ^FDBE          BCC     DRAW6       ;ALSO BLT
FDB2  18          DRAW5A: CLC                 ;GE SO MOVE POINT
FDB3  A554                LDA     ROWCRS
FDB5  6579                ADC     ROWINC
FDB7  8554                STA     ROWCRS
FDB9  A200                LDX     #0          ;AND SUBTRACT ENDPT FROM ROWAC
FDBB  207AFA              JSR     SUBEND
FDBE  18          DRAW6:  CLC                 ;DO SAME FOR COLUMN (DOUBLE BYTE ADD)
FDBF  A572                LDA     COLAC       ;ADD
FDC1  6577                ADC     DELTAC
FDC3  8572                STA     COLAC
```

```
FDC5  A573               LDA     COLAC+1
FDC7  6578               ADC     DELTAC+1
FDC9  8573               STA     COLAC+1
FDCB  C575               CMP     ENDPT+1     ;COMPARE HIGH BYTE
FDCD  9027 ^FDF6         BCC     DRAW8
FDCF  D006 ^FDD7         BNE     DRAW6A
FDD1  A572               LDA     COLAC       ;COMPARE LOW BYTE
FDD3  C574               CMP     ENDPT
FDD5  901F ^FDF6         BCC     DRAW8
FDD7  247A       DRAW6A: BIT     COLINC      ;+ OR - ?
FDD9  1010 ^FDEB         BPL     DRAW6B
FDDB  C655               DEC     COLCRS      ;DO DOUBLE BYTE DECREMENT
FDDD  A555               LDA     COLCRS
FDDF  C9FF               CMP     #$FF
FDE1  D00E ^FDF1         BNE     DRAW7
FDE3  A556               LDA     COLCRS+1
FDE5  F00A ^FDF1         BEQ     DRAW7       ;DON'T DEC IF ZERO
FDE7  C656               DEC     COLCRS+1
FDE9  1006 ^FDF1         BPL     DRAW7       ;(UNCONDITIONAL)
FDEB  E655       DRAW6B: INC     COLCRS      ;DO DOUBLE BYTE INCREMENT
FDED  D002 ^FDF1         BNE     DRAW7
FDEF  E656               INC     COLCRS+1
FDF1  A202       DRAW7:  LDX     #2          ;AND SUBTRACT ENDPT FROM COLAC
FDF3  207AFA             JSR     SUBEND
FDF6  2096FA     DRAW8:  JSR     RANGE
FDF9  20E0F5             JSR     OUTPLT      ;PLOT POINT
FDFC  ADB702             LDA     FILFLG      ;TEST RIGHT FILL
FDFF  F02F ^FE30         BEQ     DRAW9
FE01  209DFC             JSR     PHACRS
FE04  ADFB02             LDA     ATACHR
FE07  8DBC02             STA     HOLD4
FE0A  A554       DRAW8A: LDA     ROWCRS      ;SAVE ROW IN CASE OF CR
FE0C  48                 PHA
FE0D  20DCF9             JSR     INCRSA      ;POSITION CURSOR ONE PAST DOT
FE10  68                 PLA                 ;RESTORE ROWCRS
FE11  8554               STA     ROWCRS
FE13  2096FA     DRAW8C: JSR     RANGE
FE16  20A2F5             JSR     GETPLT      ;GET DATA
FE19  D00C ^FE27         BNE     DRAW8B      ;STOP IF NON-ZERO DATA IS ENCOUNTERED
FE1B  ADFD02             LDA     FILDAT      ;FILL DATA
FE1E  8DFB02             STA     ATACHR
FE21  20E0F5             JSR     OUTPLT      ;DRAW IT
FE24  4C0AFE             JMP     DRAW8A      ;LOOP
FE27  ADBC02     DRAW8B: LDA     HOLD4
FE2A  8DFB02             STA     ATACHR
FE2D  20A8FC             JSR     PLACRS
FE30  38         DRAW9:  SEC                 ;DO DOUBLE BYTE SUBTRACT
FE31  A57E               LDA     COUNTR
FE33  E901               SBC     #1
FE35  857E               STA     COUNTR
FE37  A57F               LDA     COUNTR+1
FE39  E900               SBC     #0
FE3B  857F               STA     COUNTR+1
FE3D  3003 ^FE42         BMI     DRAW10
FE3F  4C90FD             JMP     DRAW4A
FE42  4C34F6     DRAW10: JMP     RETUR1
```

```
                        ;
                        ;
                        ; TABLES
                        ;
                        ;
                        ; MEMORY ALLOCATION
                        ;
FE45  18100A0A10  ALOCAT: DB      24,16,10,10,16,28,52,100,196,196,196,196
                        ;
                        ;
                        ; NUMBER OF DISPLAY LIST ENTRIES
                        ;
FE51  17170B172F  NUMDLE: DB      23,23,11,23,47,47,95,95,97,97,97,97
FE5D  1313091327  MXDMDE: DB      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      ""
                        ;       6               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
                        ;
FE69  0206070809  ANCONV: DB      2,6,7,8,9,$A,$B,$D,$F,$F,$F,$F ;ZEROS FOR RANGE TEST IN PAGETB
                        ;
                        ;
                        ; PAGE TABLE TELLS WHICH DISPLAY LISTS ARE IN DANGER OF
                        ; CROSSING A 256 BYTE PAGE BOUNDARY
                        ;
FE75  0000000000  PAGETB: DB      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)
                        ;
FE81  0201010000  DHLINE: DB      2,1,1,0,0,1,1,2,2,2,2,2
                        ;
                        ;
                        ; COLUMN: NUMBER OF COLUMNS
                        ;
FE8D  2814142850  COLUMN: DB      40,20,20,40,80,80,160,160,64,80,80,80 ;MODE 8 IS SPECIAL CASE
                        ;
                        ;
                        ;
                        ; NOROWS: NUMBER OF ROWS
                        ;
FE99  18180C1830  NOROWS: DB      24,24,12,24,48,48,96,96,192,192,192,192
                        ;
```

```
                        ;
                        ;
                        ;
                        ; DIV2TB: HOW MANY RIGHT SHIFTS FOR HCRSR FOR PARTIAL BYTE MODES
                        ;
FEA5  0000000203        DIV2TB: DB      0,0,0,2,3,2,3,2,3,1,1,1
                        ;
                        ;
                        ; DMASKT: DISPLAY MASK TABLE
                        ;
FEB1  00FFF00F          DMASKT: DB      $00,$FF,$F0,$0F
FEB5  C0300C03                  DB      $C0,$30,$0C,$03
                        ;
                        ; MASKTB: BIT MASK.   (ALSO PART OF DMASKTB! DO NOT SEPARATE)
                        ;
FEB9  8040201008        MASKTB: DB      $80,$40,$20,$10,$08,$04,$02,$01
                        ;
                        ;
                        ;
                        ;
FEC1  28CA944600        COLRTB: DB      $28,$CA,$94,$46,$00
                        ;
                        ;
                        ;
                        ;
                        ; CNTRLS: CONTROL CODES AND THEIR DISPLACEMENTS INTO THE
                        ;         CONTROL CHARACTER PROCESSORS
                        ;
FEC6  1B                CNTRLS: DB      $1B
FEC7  79F7                      DW      ESCAPE
FEC9  1C                        DB      $1C
FECA  7FF7                      DW      CRSRUP
FECC  1D                        DB      $1D
FECD  8CF7                      DW      CRSRDN
FECF  1E                        DB      $1E
FED0  99F7                      DW      CRSRLF
FED2  1F                        DB      $1F
FED3  AAF7                      DW      CRSRRT
FED5  7D                        DB      $7D
FED6  B9F7                      DW      CLRSCR
FED8  7E                        DB      $7E
FED9  E6F7                      DW      BS
FEDB  7F                        DB      $7F
FEDC  10F8                      DW      TAB
FEDE  9B                        DB      $9B
FEDF  30FA                      DW      DOCRWS
FEE1  9C                        DB      $9C
FEE2  D4F8                      DW      DELLIN
FEE4  9D                        DB      $9D
FEE5  A4F8                      DW      INSLIN
FEE7  9E                        DB      $9E
FEE8  32F8                      DW      CLRTAB
FEEA  9F                        DB      $9F
FEEB  2DF8                      DW      SETTAB
FEED  FD                        DB      $FD
FEEE  0AF9                      DW      BELL
FEF0  FE                        DB      $FE
```

```
FEF1  6DF8              DW      DELCHR
FEF3  FF                DB      $FF
FEF4  37F8              DW      INSCHR
                ;
                ;
                ;
                ;
                ;
                ; ATAINT: ATASCI TO INTERNAL TABLE
                ;
FEF6  40002060  ATAINT: DB      $40,$00,$20,$60
                ;
                ;
                ; INTATA: INTERNAL TO ATASCI TABLE
                ;
FEFA  20400060  INTATA: DB      $20,$40,$00,$60
                ;
                ;
                ; ATASCI: ATASCII CONVERSION TABLE
                ;
FEFE  6C6A3B8080 ATASCI: DB     $6C,$6A,$3B,$80,$80,$6B,$2B,$2A ;LOWER CASE
FF06  6F8070759B         DB     $6F,$80,$70,$75,$9B,$69,$2D,$3D

FF0E  7680638080         DB     $76,$80,$63,$80,$80,$62,$78,$7A
FF16  348033361B         DB     $34,$80,$33,$36,$1B,$35,$32,$31

FF1E  2C202E6E80         DB     $2C,$20,$2E,$6E,$80,$6D,$2F,$81
FF26  728065797F         DB     $72,$80,$65,$79,$7F,$74,$77,$71

FF2E  398030377E         DB     $39,$80,$30,$37,$7E,$38,$3C,$3E
FF36  6668648082         DB     $66,$68,$64,$80,$82,$67,$73,$61


FF3E  4C4A3A8080         DB     $4C,$4A,$3A,$80,$80,$4B,$5C,$5E ;UPPER CASE
FF46  4F8050559B         DB     $4F,$80,$50,$55,$9B,$49,$5F,$7C

FF4E  5680438080         DB     $56,$80,$43,$80,$80,$42,$58,$5A
FF56  248023261B         DB     $24,$80,$23,$26,$1B,$25,$22,$21

FF5E  5B205D4E80         DB     $5B,$20,$5D,$4E,$80,$4D,$3F,$81
FF66  528045599F         DB     $52,$80,$45,$59,$9F,$54,$57,$51

FF6E  288029279C         DB     $28,$80,$29,$27,$9C,$40,$7D,$9D
FF76  4648448083         DB     $46,$48,$44,$80,$83,$47,$53,$41


FF7E  0C0A7B8080         DB     $0C,$0A,$7B,$80,$80,$0B,$1E,$1F ;CONTROL
FF86  0F8010159B         DB     $0F,$80,$10,$15,$9B,$09,$1C,$1D

FF8E  1680038080         DB     $16,$80,$03,$80,$80,$02,$18,$1A
FF96  808085801B         DB     $80,$80,$85,$80,$1B,$80,$FD,$80

FF9E  0020600E80         DB     $00,$20,$60,$0E,$80,$0D,$80,$81
FFA6  128005199E         DB     $12,$80,$05,$19,$9E,$14,$17,$11

FFAE  80808080FE         DB     $80,$80,$80,$80,$FE,$80,$7D,$FF
FFB6  0608048084         DB     $06,$08,$04,$80,$84,$07,$13,$01
```

```
                        ;
                        ;
                        ;
                        ;
                        ;
FFBE  AD09D2      PIRQ5:  LDA     KBCODE
FFC1  CDF202              CMP     CH1         ;TEST AGAINST LAST KEY PRESSED
FFC4  D005 ^FFCB          BNE     PIRQ3       ;IF NOT, GO PROCESS KEY
FFC6  ADF102              LDA     KEYDEL      ;IF KEY DELAY BYTE > 0
FFC9  D020 ^FFEB          BNE     PIRQ4       ;IGNORE KEY AS BOUNCE
FFCB  AD09D2      PIRQ3:  LDA     KBCODE      ;RESTORE AC
FFCE  C99F                CMP     #CNTL1      ;TEST CONTROL 1 (SSFLAG)
FFD0  D00A ^FFDC          BNE     PIRQ1
FFD2  ADFF02              LDA     SSFLAG
FFD5  49FF                EOR     #$FF
FFD7  8DFF02              STA     SSFLAG
FFDA  B00F ^FFEB          BCS     PIRQ4       ;(UNCONDITIONAL) MAKE ^1 INVISIBLE
FFDC  8DFC02      PIRQ1:  STA     CH
FFDF  8DF202              STA     CH1
FFE2  A903                LDA     #3
FFE4  8DF102              STA     KEYDEL      ;INITIALIZE KEY DELAY FOR DEBOUNCE
FFE7  A900                LDA     #0          ;CLEAR COLOR SHIFT BYTE
FFE9  854D                STA     ATRACT
FFEB  A930        PIRQ4:  LDA     #$30
FFED  8D2B02              STA     SRTIMR
FFF0  68          PIRQ2:  PLA
FFF1  40                  RTI
                        ;
                        ;
FFF2  FFFFFFFFFF          DB      $FF,$FF,$FF,$FF,$FF,$FF
                        ;
      = FFF8      CRNTPC  =       *
FFF8  = 0014              ORG     $14
0014  00          KBDSPR: DB      $FFF8-CRNTPC ;^GDISPLC IS TOO LONG


                        ;*********************************************************************
                        ; ADDITIONS TO REV B TO GENERATE EXACT ROM IMAGE
                        ; BY M.B. & V.W. 8/21/84
                        ;

                                LIST    I

                                INCLUDE D:CHARSET.ASM
0015  = E000                    ORG     $E000

E000  0000000000    DB $00,$00,$00,$00,$00,$00,$00,$00,$00,$18,$18,$18,$18,$00,$18,$00
E010  0066666600    DB $00,$66,$66,$66,$00,$00,$00,$00,$00,$66,$FF,$66,$66,$FF,$66,$00
E020  183E603C06    DB $18,$3E,$60,$3C,$06,$7C,$18,$00,$00,$66,$6C,$18,$30,$66,$46,$00
E030  1C361C386F    DB $1C,$36,$1C,$38,$6F,$66,$3B,$00,$00,$18,$18,$18,$00,$00,$00,$00
E040  000E1C1818    DB $00,$0E,$1C,$18,$18,$1C,$0E,$00,$00,$70,$38,$18,$18,$38,$70,$00
E050  00663CFF3C    DB $00,$66,$3C,$FF,$3C,$66,$00,$00,$00,$18,$18,$7E,$18,$18,$00,$00
E060  0000000000    DB $00,$00,$00,$00,$00,$18,$18,$30,$00,$00,$00,$7E,$00,$00,$00,$00
E070  0000000000    DB $00,$00,$00,$00,$00,$18,$18,$00,$00,$06,$0C,$18,$30,$60,$40,$00
E080  003C666E76    DB $00,$3C,$66,$6E,$76,$66,$3C,$00,$00,$18,$38,$18,$18,$18,$7E,$00
E090  003C660C18    DB $00,$3C,$66,$0C,$18,$30,$7E,$00,$00,$7E,$0C,$18,$0C,$66,$3C,$00
E0A0  000C1C3C6C    DB $00,$0C,$1C,$3C,$6C,$7E,$0C,$00,$00,$7E,$60,$7C,$06,$66,$3C,$00
```

```
E0B0  003C607C66   DB $00,$3C,$60,$7C,$66,$66,$3C,$00,$00,$7E,$06,$0C,$18,$30,$30,$00
E0C0  003C663C66   DB $00,$3C,$66,$3C,$66,$66,$3C,$00,$00,$3C,$66,$3E,$06,$0C,$38,$00
E0D0  0000181800   DB $00,$00,$18,$18,$00,$18,$18,$00,$00,$00,$18,$18,$00,$18,$18,$30
E0E0  060C183018   DB $06,$0C,$18,$30,$18,$0C,$06,$00,$00,$00,$7E,$00,$00,$7E,$00,$00
E0F0  6030180C18   DB $60,$30,$18,$0C,$18,$30,$60,$00,$00,$3C,$66,$0C,$18,$00,$18,$00
E100  003C666E6E   DB $00,$3C,$66,$6E,$6E,$60,$3E,$00,$00,$18,$3C,$66,$66,$7E,$66,$00
E110  007C667C66   DB $00,$7C,$66,$7C,$66,$66,$7C,$00,$00,$3C,$66,$60,$60,$66,$3C,$00
E120  00786C6666   DB $00,$78,$6C,$66,$66,$6C,$78,$00,$00,$7E,$60,$7C,$60,$60,$7E,$00
E130  007E607C60   DB $00,$7E,$60,$7C,$60,$60,$60,$00,$00,$3E,$60,$60,$6E,$66,$3E,$00
E140  0066667E66   DB $00,$66,$66,$7E,$66,$66,$66,$00,$00,$7E,$18,$18,$18,$18,$7E,$00
E150  0006060606   DB $00,$06,$06,$06,$06,$66,$3C,$00,$00,$66,$6C,$78,$78,$6C,$66,$00
E160  0060606060   DB $00,$60,$60,$60,$60,$60,$7E,$00,$00,$63,$77,$7F,$6B,$63,$63,$00
E170  0066767E7E   DB $00,$66,$76,$7E,$7E,$6E,$66,$00,$00,$3C,$66,$66,$66,$66,$3C,$00
E180  007C66667C   DB $00,$7C,$66,$66,$7C,$60,$60,$00,$00,$3C,$66,$66,$66,$6C,$36,$00
E190  007C66667C   DB $00,$7C,$66,$66,$7C,$6C,$66,$00,$00,$3C,$60,$3C,$06,$06,$3C,$00
E1A0  007E181818   DB $00,$7E,$18,$18,$18,$18,$18,$00,$00,$66,$66,$66,$66,$66,$7E,$00
E1B0  0066666666   DB $00,$66,$66,$66,$66,$66,$3C,$18,$00,$00,$63,$63,$6B,$7F,$77,$63,$00
E1C0  0066663C3C   DB $00,$66,$66,$3C,$3C,$66,$66,$00,$00,$66,$66,$3C,$18,$18,$18,$00
E1D0  007E0C1830   DB $00,$7E,$0C,$18,$30,$60,$7E,$00,$00,$1E,$18,$18,$18,$18,$1E,$00
E1E0  0040603018   DB $00,$40,$60,$30,$18,$0C,$06,$00,$00,$78,$18,$18,$18,$18,$78,$00
E1F0  00081C3663   DB $00,$08,$1C,$36,$63,$00,$00,$00,$00,$00,$00,$00,$00,$00,$FF,$00
E200  00367F7F3E   DB $00,$36,$7F,$7F,$3E,$1C,$08,$00,$18,$18,$18,$1F,$1F,$18,$18,$18
E210  0303030303   DB $03,$03,$03,$03,$03,$03,$03,$03,$18,$18,$18,$F8,$F8,$00,$00,$00
E220  181818F8F8   DB $18,$18,$18,$F8,$F8,$18,$18,$18,$00,$00,$00,$F8,$F8,$18,$18,$18
E230  03070E1C38   DB $03,$07,$0E,$1C,$38,$70,$E0,$C0,$C0,$E0,$70,$38,$1C,$0E,$07,$03
E240  0103070F1F   DB $01,$03,$07,$0F,$1F,$3F,$7F,$FF,$00,$00,$00,$00,$0F,$0F,$0F,$0F
E250  80C0E0F0F8   DB $80,$C0,$E0,$F0,$F8,$FC,$FE,$FF,$0F,$0F,$0F,$0F,$00,$00,$00,$00
E260  F0F0F0F000   DB $F0,$F0,$F0,$F0,$00,$00,$00,$00,$FF,$FF,$00,$00,$00,$00,$00,$00
E270  0000000000   DB $00,$00,$00,$00,$00,$00,$FF,$FF,$00,$00,$00,$00,$F0,$F0,$F0,$F0
E280  001C1C7777   DB $00,$1C,$1C,$77,$77,$08,$1C,$00,$00,$00,$00,$1F,$1F,$18,$18,$18
E290  000000FFFF   DB $00,$00,$00,$FF,$FF,$00,$00,$00,$18,$18,$18,$FF,$FF,$18,$18,$18
E2A0  00003C7E7E   DB $00,$00,$3C,$7E,$7E,$7E,$3C,$00,$00,$00,$00,$00,$FF,$FF,$FF,$FF
E2B0  C0C0C0C0C0   DB $C0,$C0,$C0,$C0,$C0,$C0,$C0,$C0,$00,$00,$00,$FF,$FF,$18,$18,$18
E2C0  181818FFFF   DB $18,$18,$18,$FF,$FF,$00,$00,$00,$F0,$F0,$F0,$F0,$F0,$F0,$F0,$F0
E2D0  1818181F1F   DB $18,$18,$18,$1F,$1F,$00,$00,$00,$78,$60,$78,$60,$7E,$18,$1E,$00
E2E0  00183C7E18   DB $00,$18,$3C,$7E,$18,$18,$18,$00,$00,$18,$18,$18,$7E,$3C,$18,$00
E2F0  0018307E30   DB $00,$18,$30,$7E,$30,$18,$00,$00,$00,$18,$0C,$7E,$0C,$18,$00,$00
E300  00183C7E7E   DB $00,$18,$3C,$7E,$7E,$3C,$18,$00,$00,$00,$3C,$06,$3E,$66,$3E,$00
E310  0060607C66   DB $00,$60,$60,$7C,$66,$66,$7C,$00,$00,$00,$3C,$60,$60,$60,$3C,$00
E320  0006063E66   DB $00,$06,$06,$3E,$66,$66,$3E,$00,$00,$00,$3C,$66,$7E,$60,$3C,$00
E330  000E183E18   DB $00,$0E,$18,$3E,$18,$18,$18,$00,$00,$00,$3E,$66,$66,$3E,$06,$7C
E340  0060607C66   DB $00,$60,$60,$7C,$66,$66,$66,$00,$00,$18,$00,$38,$18,$18,$3C,$00
E350  0006000606   DB $00,$06,$00,$06,$06,$06,$06,$3C,$00,$60,$60,$6C,$78,$6C,$66,$00
E360  0038181818   DB $00,$38,$18,$18,$18,$18,$3C,$00,$00,$00,$66,$7F,$7F,$6B,$63,$00
E370  00007C6666   DB $00,$00,$7C,$66,$66,$66,$66,$00,$00,$00,$3C,$66,$66,$66,$3C,$00
E380  00007C6666   DB $00,$00,$7C,$66,$66,$7C,$60,$60,$00,$00,$3E,$66,$66,$3E,$06,$06
E390  00007C6660   DB $00,$00,$7C,$66,$60,$60,$60,$00,$00,$00,$3E,$60,$3C,$06,$7C,$00
E3A0  00187E1818   DB $00,$18,$7E,$18,$18,$18,$0E,$00,$00,$00,$66,$66,$66,$66,$3E,$00
E3B0  0000666666   DB $00,$00,$66,$66,$66,$3C,$18,$00,$00,$00,$63,$6B,$7F,$3E,$36,$00
E3C0  0000663C18   DB $00,$00,$66,$3C,$18,$3C,$66,$00,$00,$00,$66,$66,$66,$3E,$0C,$78
E3D0  00007E0C18   DB $00,$00,$7E,$0C,$18,$30,$7E,$00,$00,$18,$3C,$7E,$7E,$18,$3C,$00
E3E0  1818181818   DB $18,$18,$18,$18,$18,$18,$18,$18,$00,$7E,$78,$7C,$6E,$66,$06,$00
E3F0  0818387838   DB $08,$18,$38,$78,$38,$18,$08,$00,$10,$18,$1C,$1E,$1C,$18,$10,$00

                   ;****
                   ; HOLES WITH RANDOM DATA BYTES
```

```
E400  = EDE8              ORG     $EDE8
EDE8  2485                DB      $24,$85

EDEA  = E90B              ORG     $E90B
E90B  66667E6600          DB      $66,$66,$7E,$66,$00,$00,$7C,$4C,$ED,$E8,$66,$7C,$00
E918  003C666060          DB      $00,$3C,$66,$60,$60,$66,$3C,$00,$00,$78,$6C,$66,$66
E925  6C7800007E          DB      $6C,$78,$00,$00,$7E,$60,$7C,$60,$60,$7E,$00,$00,$7E
E932  607C606060          DB      $60,$7C,$60,$60,$60,$00,$00,$3E,$60,$60,$6E,$66,$3E
E93F  000066667E          DB      $00,$00,$66,$66,$7E


                  ;****
                  ;

E944  = FFF8              ORG     $FFF8   ;CHECKSUM & HARDWARE VECTORS
FFF8  F3E691E725          DB      $F3,$E6,$91,$E7,$25,$F1,$F3,$E6
0000                      END
```

```
  no ERRORs, 1215 Labels, $3AD2 free.


     ACK    0041     39#32   44/54
     ACKREC E9C6     42/51   42#60
     ADDCOR 030E     10#36   55/41   55/45   55/49
     ADJ1   ED0C     55/59   56# 6
     ADJUST ED04     54/53   54/56   55#58
     ADRESS 0064      6#35   91/31   91/56   92/25   92/27   92/36   92/37   92/38   92/40   93/49   93/51
93/55   93/57
                     93/60   94/ 5   94/36   95/38   95/40   96/51   97/11   97/13   99/42   99/45   99/46
101/11  101/13
                    101/39  101/44  103/21  103/28  103/31  103/33  103/35  103/39  104/50  104/51  104/54
104/55  109/23
                    109/25  109/32  109/35  109/37  110/24  110/36  110/38  110/40  110/43  110/46  110/50
110/51  110/53
                    110/54  112/48  114/25  114/27
     ADRTAB E6FE     31#35   32/13
   n AFP    D800     11#55
   n ALLPOT D208     13#26
     ALLSEC F30E     85/24   85#29
     ALOCAT FE45     91/57  118#12
     ANCONV FE69     91/50  118#37
     ANTIC  D400     14#45   14/46   14/47   14/48   14/49   14/50   14/51   14/52   14/53   14/54   14/55
14/56   14/57
                     14/58   14/59   14/60
   n APPEND 0001      3#48
     APPMHI 000E      4#59  103/34  103/38
     ASCCO1 F705     97/46   97#48
     ASCZER 0030     16# 7   28/31
     ATACHR 02FB     10# 7   94/48   94/51   94/56   95/16   95/55   96/19   96/42   96/46   96/53   97/49
98/26   98/37
                     98/39  100/21  109/13  112/61  117/34  117/45  117/49
     ATAINT FEF6     95/26  120#15
   n ATAN   BE43     12#26
     ATASCI FEFE     97/48  120#25
     ATEOF  F00B     70/26   70#32
     ATRACT 004D      6#14   32/26   34/13   34/19   34/21  121/27
     AUDC1  D201     13#34   52/40   52/50   53/18
     AUDC2  D203     13#36   52/51
     AUDC3  D205     13#38   52/46
```

```
   n AUDC4  D207     13#40
     AUDCTL D208     13#41   52/33
     AUDF1  D200     13#33   51/51
     AUDF2  D202     13#35   51/49
     AUDF3  D204     13#37   42/15   50/14   57/19   68/57
     AUDF4  D206     13#39   42/17   50/16   57/21   68/59
     B192HI 0000     39#41   42/16
     B192LO 0028     39#40   42/14
     B600HI 0005     39#43   50/15
     B600LO 00CC     39#42   50/13
     BAD    EA63     45/14   45#20
     BADCOM E9BF     42/48   42#54   43/18
     BADDSK F306     85#25   85/61   86/10
     BADIOC 0086      4#22   18/23
     BADMOD 0091      4#34   91/52
     BADST  EE9E     63/40   63/43   64/32   64#34
     BEEP   F058     68/21   68/54   74#11
     BEEP1  F05A     74#12   74/46
     BEGIN  ED10     51/ 9   56#27   56/39
     BELL   F90A     96/27  102#27  119/60
     BELL1  F90C    102#28  102/30
     BFENHI 0035      5#47   42/40   44/38   46/23   49/15   49/52
     BFENLO 0034      5#46   42/37   44/35   46/21   49/13   49/47
     BITCLR FB12    100/39  108/26  108#34
     BITCON FAEB    108# 7  108/28  108/34  108/44
     BITGE1 FB31    108/48  108#50
     BITGET FB25    100/33  105/49  108#44
     BITMSK 006E      6#42  108/11  108/30  108/35  108/47
     BITPUT FB04    102/12  108#26  109/60  110/13
     BITROL FAFA    108#20  110/34
     BITSET FB06    100/37  108#28
     BLACKB F22A     81#51   81/52
     BLFILL EEC1     64/61   65# 7   65/27
     BLIM   028A      8#50   68/40   70/11   70/30
     BLKB2  F230     81/51   81#53
     BLKBDV E471      3#14   77/30   79/26   79/28
     BLOAD  F36C     86/ 9   86#14
     BOOT   F2CF     81/21   84#57
     BOOT?  0009      4#56   84/59   86/12   87/15   87/31
     BOOTAD 0242      8#12   85/34   85/36   86/15   86/18
     BOTSCR 02BF      9#24   92/47   92/55   99/14   99/20  101/22  106/55  113/39
     BPTR   003D      5#57   68/39   69/25   70/10   70/14   70/22   71/ 8   71/10   71/19   73/15
     BRKABT 0080      4#16   57/49   68/45   97/34  107/28
     BRKKEY 0011      5# 7   32/23   45/52   47/46   56/27   56/49   57/55   68/46   69/20   84/26   97/35
98/20  107/29
                     107/30
     BRKKY  0236      7#54   31/35   84/28   84/30
     BRKKY2 E754     32#22   84/27   84/29
     BROKE  EDA0     45/54   47/48   56/29   56/50   57#44
     BS     F7E6    100# 6  119/46
     BS1    F80D    100/ 8  100#23
     BS2    F805    100/16  100/18  100#20
     BS3    F7F5    100/11  100#13
   n BSA    F7EC    100# 9
     BUFADR 0015      5#10   61/47   61/56   61/60   62/24   62/26
     BUFCNT 006B      6#40   96/11   96/35   96/37  111/37  111/47  111/50  111/58  112/10
     BUFFH  0004     76#25   85/21
     BUFFL  0000     76#26   85/19
```

```
     BUFFUL EECB     64/58   65#12
     BUFRFL 0038      5#50   47/36   48/36   49/28
     BUFRHI 0033      5#45   42/39   44/37   46/18   46/22   49/10   49/14   49/50   57/36   57/38
     BUFRLO 0032      5#44   42/35   44/33   45/46   46/16   46/20   46/47   48/58   49/ 8   49/12   49/45
57/27   57/29
                      57/33   57/35
     BUFSTR 006C      6#41   96/14   96/16   96/31   96/33   99/56   99/60  106/29  111/32  111/34  112/24
112/26
     CAINI  F239     80/37   81#61
     CART   BFFC     76#40   80/26   80/34   83/33   83/35   83/36   83/45
     CARTAD BFFE     76#42   81/61   82/ 5   83/40
     CARTCS BFFA     76#39   81/31   81/37
     CARTFG BFFD     76#41   81/13   81/16   81/28   81/34   83/38
     CAS31  EC5E     52/49   52#54
     CASBUF 03FD     11#26   67/ 5   67/ 6   70/13   70/24   70/29   71/ 9   73/17   73/23   75/29   75/31
75/32   76/25
                     76/26   85/30   85/38   85/40   85/43
     CASENT EB80     41/57   50# 8
     CASET  0060     39#11   41/55   51/43   52/48
     CASETV E440      2#53   67/22   77/60   84/44
     CASFLG 030F     10#37   42/ 5   47/31   50/45
     CASINI 0002      4#49   87/33   87/35   87/38
     CASORG EF41      2#24   66/53   67/38
     CASRED EBB3     50/ 9   50#44
     CASSBT 004B      6#11   85/26   85/60   86/ 6   86/56   87/25   87/29
   n CASSET 0000      3#61
   n CASSPR 0014     75#38
     CAUX1  023C      7#60   42/29
     CAUX2  023D      7#61   42/31
     CBAUDH 02EF      9#52   57/14   57/20   67/46
     CBAUDL 02EE      9#51   57/13   57/18   67/44
     CBINI  F23C     80/29   82# 5
     CBUFH  0003     67# 5   75/ 6
     CBUFHI 0002     40#12   42/38
     CBUFL  00FD     67# 6   75/ 8
     CBUFLO 003A     40#13   42/34
     CCOMND 023B      7#59   42/26
     CDEVIC 023A      7#58   40/12   40/13   42/23
     CDTMA1 0226      7#39   36/32   58/ 8   58/10
     CDTMA2 0228      7#40   36/33
     CDTMF3 022A      7#41   35/16   68/34   68/36   69/19   69/22
   n CDTMF4 022C      7#43
   n CDTMF5 022E      7#45
     CDTMV1 0218      7#32   35/12   35/13   36/39   36/41   36/43   36/44   36/46   37/20   37/22
   n CDTMV2 021A      7#33
   n CDTMV3 021C      7#34
   n CDTMV4 021E      7#35
   n CDTMV5 0220      7#36
     CDUBL  EF26     66/27   66#31
     CH     02FC     10# 8   35/39   90/ 8   97/30   97/37   97/42  121/22
     CH1    02F2      9#56  121/11  121/23
     CHACT  02F3      9#58   34/60   91/21
     CHACTL D401     14#47   34/61
     CHAR   02FA     10# 6   94/42   95/27   95/29  108/57  109/10
     CHBAS  02F4      9#59   34/58   91/19
     CHBASE D409     14#53   34/59
     CHKDON EABA     46/33   46#40   46/55
     CHKERR 008F      4#31   48/43
     CHKSNT 003B      5#53   45/42   46/26   46/32   47/ 7
     CHKSUM 0031      5#43   45/41   45/50   46/29   46/51   46/53   47/35   48/40   48/61   49/ 6   57/31
```

```
     CHKTIM EAF9     47#46   47/53
   n CHRORG E000      2#16
     CICLO2 E53F     20/11   20#13
     CICLOS E533     18/50   20# 8
     CIERR1 E4D1     18/18   18#23
     CIERR2 E6B0     28/11   28#21
   n CIERR3 E50F     19#14
     CIERR4 E511     18/39   19#15   19/19   19/25   20/30
     CIJUMP E693     27/12   27#18
     CINI   F3E1     87/18   87#38
     CIO    E4C4     17/ 8   18#12
     CIOCHR 002F      5#40   18/12   21/24   21/29   21/39   22/14   22/18   23/23   23/31   23/42   25/26
     CIOI1  E4A8     17#24   17/35
     CIOINT E4A6     17/12   17#23   29/12
     CIOINV E46E      3#13   17/11   84/45
     CIOORG E4A6      2#19   17/20
     CIOPEN E509     18/48   19# 9
   n CIOSPR 0014     29#15
     CIOV   E456      3# 5   17/ 7   80/51   86/48
     CIREAD E569     18/54   21# 7
     CIRT3  E62B     25#18   25/23
     CIRTN1 E61B     18/24   19/15   21/13   23/13   25# 8
     CIRTN2 E61D     19/38   20/19   20/40   21/25   22/28   24/18   25#12
     CIST1  E559     20/26   20#33
     CISTSP E54E     18/52   20#24
     CIWRIT E5C9     18/55   23# 7
   n CIX    00F2     12#40
     CKEY   004A      6#10   84/51   87/21   87/30
     CKSTC  EE11     61/26   61#28
     CLICK  FCD8     97/43  102/28  113#59
     CLICK1 FCDA    113#60  114/ 6
   n CLOSE  000C      3#30
     CLOSEC F02B     67/23   73# 8
     CLRCHP F27A     84#15   84/20
     CLRCOD 007D     88# 8   94/52
     CLRLI1 FBA6    110#24  110/26
     CLRLIN FB9B    101/55  110#19
     CLRRAM F140     79#16   79/19   79/23
     CLRSC2 F7BF     99#42   99/44   99/48
     CLRSC3 F7CE     99#50   99/53
     CLRSCR F7B9     94/20   94/54   99#39  119/44
     CLRTAB F832    100#38  119/56
     CLRTBS F430     91#36   91/38
     CLS    007D     76#20   78/17
     CLWRT  F038     73/ 9   73#15
     CMODE  EF1E     66#26   66/43
     CMPTAB E6F6     31#25   31/56
     CNTL1  009F     88# 9  121/16
     CNTRLS FEC6     97/10   97/12  112/60  119#33
     COLAC  0072      6#45  115/61  116/59  116/61  117/ 5  117/ 7  117/11
   n COLBK  D01A     14#18
     COLCR  FCE4     99/54  105/60  114#12
     COLCR1 FCEE    114/14  114#17
     COLCR2 FCF0    114/16  114#18
     COLCRS 0055      6#25   96/15   96/34   99/24   99/25   99/30   99/32   99/33   99/59  100/ 9  100/14
100/25  101/37
                    102/20  103/51  103/53  104/27  104/32  104/61  105/ 6  105/19  105/21  105/22  105/34
106/ 5  107/15
                    107/20  107/23  110/20  111/20  111/35  111/44  112/ 6  114/18  115/21  115/23  117/16
117/17  117/20
```

```
                117/22  117/24  117/26
  COLDST 0244     8#13   78/47   79/31   81/25
  COLDSV E477     3#16   77/38
  COLINC 007A     6#50  115/27  115/49  117/14
n COLON  003A    16# 8
  COLOR0 02C4     9#31   91/41
n COLOR1 02C5     9#32
n COLOR2 02C6     9#33
n COLOR3 02C7     9#34
  COLOR4 02C8     9#35   92/23
n COLPF0 D016    14#14
n COLPF1 D017    14#15
n COLPF2 D018    14#16
n COLPF3 D019    14#17
  COLPM0 D012    14#10   34/55
n COLPM1 D013    14#11
n COLPM2 D014    14#12
n COLPM3 D015    14#13
  COLRSH 004F     6#16   34/25   34/53
  COLRTB FEC1    91/40  119#25
  COLUMN FE8D   105/24  107/22  118#54
  COM1   E647    26/12   26#19
  COM2   E662    26/16   26#33
  COMENT E63D    19/24   19/33   20/10   20/33   21/16   23/16   26#10
  COMFRM E978    42#14   42/55
  COMMND E974    42# 9   44/ 8
  COMPLT 0043    39#34   44/56
  COMPUT ECA3    54#51   57/12
  COMRE1 F7A7    99/28   99#31   99/35   99/36
  COMRET F789    99/13   99#17   99/21
  COMTAB E6C9    18/47   26/24   29#11
  CONSOL D01F    14#23   34/50   74/22   74/27   84/48  113/60
  CONTIN EC31    51/61   52#28
  CONVR1 F97E   104#21  104/25
  CONVR2 F988   104/22  104#26
  CONVR3 F98F   104#29  104/35
  CONVR4 F99C   104/30  104#36
  CONVR5 F9A6   104/40  104#42
  CONVR6 F9A7   104#43  104/46
  CONVRT F947    94/35   95/28  101/10  101/38  102/21  103#49  110/21  112/43
n COS    BD73    12#25
  COUNT  ED3B    56#49   56/60   57/ 6   57/16
  COUNTR 007E     6#54  116/17  116/27  116/35  116/36  117/52  117/54  117/55  117/57
  CR     009B     4# 9   64/60   78/17   78/23   78/33   94/57   96/20   96/45   98/21
  CRETRI 000D    40#15   42/ 9   43/11
  CRETRN EBDF    50/38   51#13
  CRETRY 0036     5#48   42/10   42/54   43/12
  CRITIC 0042     6# 5   34/30   41/52   44/15
  CRLOOP F5E7    95#12   95/15
  CRNTP1 E6D5    29#13   29/15
  CRNTP2 E90B    37#38   37/40
  CRNTP3 EDE8    59#23   59/25
  CRNTP4 EE78    62#33   62/36
  CRNTP5 EF41    66#49   66/53
  CRNTP6 F0E3    75#36   75/38
  CRNTP7 F3E4    87#44   87/47
  CRNTPC FFF8   121#36  121/38
```

```
     CRSINH 02F0      9#54   32/25   91/33   95/48
     CRSRDN F78C     99#18  119/38
     CRSRL1 F7A3     99/26   99#29
     CRSRLF F799     99#24  100/13  111/55  112/ 5  119/40
     CRSROR 008D      4#29  107/34
     CRSRRT F7AA     99#32  100/24  119/42
     CRSRUP F77F     99#12  100/19  119/36
     CSBOOT F3B2     81/ 5   87#13
     CSBOT2 F3C0     87/14   87#21
     CSIDE  EF2E     66/32   66#35
   n CSIO   F0AC     75#10
     CSOPIV E47D      3#18   67/34   87/26
   n CSTAT  0288      8#48
     CTIA   D000     13#48   13/49   13/50   13/51   13/52   13/53   13/54   13/55   13/56   13/57   13/58
13/59   13/60
                    13/61   14/ 5   14/ 6   14/ 7   14/ 8   14/ 9   14/10   14/11   14/12   14/13   14/14
14/15   14/16
                    14/17   14/18   14/19   14/20   14/21   14/22   14/23   14/24   14/25   14/26   14/27
14/28   14/29
                    14/30   14/31   14/32   14/33   14/34   14/35   14/36   14/37   14/38   14/39   14/40
14/41   14/42
                    14/43
     CTIMHI 0000     40#18   54/29
     CTIMLO 0002     40#17   54/28
   n CTRLC  0092     76#21
     D      0044     63#27   66/31
     DAUX1  030A     10#32   42/28   64/29   66/40   85/18   85/56
     DAUX2  030B     10#33   42/30   50/21   50/48   51/13   75/23   85/16
     DBDDEC F913     92/42  103#13
     DBDEC  F91F     92/32  103#24
     DBSECT 0241      8#11   85/54
     DBSUB  F921     91/59   92/19  103/14  103#25
     DBSUB1 F934    103/32  103#34
     DBUFHI 0305     10#27   49/49   61/33   62/25   65/47   75/ 7   85/22
     DBUFLO 0304     10#26   49/44   61/31   62/23   65/46   75/ 9   85/20
     DBUFSZ 0014     63#20   66/33
     DBYTHI 0309     10#31   49/51   61/38   62/11   65/61   74/60
     DBYTLO 0308     10#30   49/46   61/36   62/ 9   65/59   75/ 5
   n DCB    0300     10#21
     DCOMND 0302     10#24   42/25   61/17   61/24   61/42   61/49   64/28   65/53   66/39   74/58   75/16
85/10   86/60
     DCTIM1 E8DD     36/40   36#44
     DCTIMR E8D0     34/27   35/ 6   35/15   36#39
     DCTXF  E8EA     36/42   36/45   36/47   36#50
     DDEVIC 0300     10#22   41/54   42/20   51/42   52/47   61/15   65/49   68/61   75/11
     DECBF1 E66D     26/41   26#43
     DECBFL E663     21/42   21/46   23/45   23/49   26#38
   n DEGFLG 00FB     12#45
   n DEGON  0006     12#48
     DELAY0 EC8C     54#20   54/24
     DELAY1 EC8E     54#21   54/22
     DELCH1 F870    101#10  101/27
     DELCH2 F896    101/20  101/23  101#28
     DELCHR F86D    101# 9  120/ 5
   n DELETE 0021      3#38
     DELLI1 F8DD    101#61  102/15
   n DELLI2 F8F8    102#19
   n DELLIA F8D7    101#58
     DELLIB F8DB    101#60  102/25  112/52
     DELLIN F8D4    101#57  119/52
     DELTAC 0077      6#48  115/43  115/46  115/50  115/52  115/53  115/55  115/56  115/58  116/11  116/16
116/20  116/60
                    117/ 6
```

```
     DELTAR 0076      6#47  115/31  115/35  115/39  116/21  116/23  116/41
     DELTI1 FC82    112#48  112/51
   n DELTI2 FC89    112#52
     DELTI3 FC8C    112/42  112/49  112#53
     DELTIA FC68    101/31  112#35
     DELTIB FC70    112/37  112#39
     DELTIM FC73    100/12  112#40
     DERR   E9F6     43/30   43#51
     DERR1  EA06     42/57   43/39   44# 5
     DERR5  F10D     78#23   78/25   78/26
     DERRH  00F1     78#25   86/30
     DERRL  000D     78#26   86/29
     DERROR 0090      4#33   45/ 5
     DEVS1  E6A6     28#13   28/18
     DEVS2  E6B5     28/14   28#26
     DEVS3  E6C5     28/33   28#35
     DEVS4  E6C8     28/23   28#39
     DEVSRC E69E     19/18   20/29   28# 9
     DFLAGS 0240      8#10   85/31
     DHLINE FE81    104/20  118#49
   n DIGRT  00F1     12#39
     DINDEX 0057      6#26   91/16   91/49   92/48   93/22   95/46  104/19  105/23  105/36  106/ 7  106/58
107/ 8  108/58
                    114/15
     DINI   F37E     85/ 5   86/11   86#22
     DINIT  EDEA     60/34   61# 6
   n DIRECT 0002      3#49
   n DISK   0000      4# 6
     DISKID 0031     60#19   61/14
   n DISKIV E450      2#60
     DISKM  F3A4     86/57   86#59
   n DISPLA E410     89#24
   n DISPLY 0000      3#59
     DIV2TB FEA5    104/26  119#10
     DLISTH D403     14#49   34/42
     DLISTL D402     14#48   34/44
     DMACTL D400     14#46   34/46
     DMASK  02A0      9#11   94/37   95/34   95/36  104/58
     DMASKT FEB1    104/57  119#15
     DNACK  008B      4#27   45/ 9
     DOB1   FC1A    111/40  111#42
     DOBOOT F2ED     85/12   85#15   85/27
     DOBU1A FC29    111/43  111/46  111#49
     DOBUF1 FC12    111#38  111/53
     DOBUF2 FC39    111/48  111#56  112/11
     DOBUF3 FC51    112/ 8  112#10
     DOBUF4 FC55    111/57  111/61  112#12
     DOBUFC FC00     96/30  111#29
     DOCR   FA34    100/28  105/37  105/42  105/45  105#60
     DOCR1  FA00    105/33  105#36
     DOCR1A FA29    105/50  105/52  105#55
     DOCR1B FA14    105/40  105#45
   n DOCR2  FA3D    106# 7
     DOCR2A FA4A    106/10  106#14
     DOCR2B FA4D    106/13  106#15
     DOCR4B FA61    106/25  106#27  106/33
     DOCRWS FA30     94/59   96/44  105/43  105#58  119/50
     DOINTP ECE5     55#35   55/38
```

```
     DOLCO1 FBE5    111# 9  111/17
     DOLCO2 FBF8    111/11  111#18
     DOLCOL FBDD     99/31  100/23  100/35  101/ 6  101/33  101/56  101/57  102/26  105/55  106/36  110/57
111# 5  112/22
                    112/39
     DOPEN  F3F6     89/24   91# 9
     DOPEN1 F460     91#58   91/61
     DOPEN2 F4D5     92#57   92/60
     DOPEN3 F4FA     93#18   93/21
     DOPEN4 F51C     93#34   93/37
     DOPEN5 F524     93/24   93#38
     DOPEN7 F588     94/19   94#24
     DOPEN8 F438     91#40   91/43
     DOPEN9 F577     94/11   94#17
     DOPENA F457     91/51   91#54
     DOSINI 000C      4#58   85/39   85/41   86/22   87/32   87/34
     DOSS   F6AD     96/22   96#56
     DOSVEC 000A      4#57   79/27   79/29   81/41
   n DOUBLE 0044     39#26
     DRAW   FCFC     89/29  115# 9
     DRAW1  FD37    115/32  115#40
     DRAW10 FE42    116/38  117/58  117#60
     DRAW11 FD99    116/37  116#39
     DRAW2  FD5C    115/47  115/57  115#59
     DRAW3  FD83    116/19  116/22  116#26
     DRAW3A FD62    116# 5  116/10
     DRAW4A FD90    116#35  117/59
     DRAW5  FDA4    116/43  116#45
     DRAW5A FDB2    116/48  116#52
     DRAW6  FDBE    116/47  116/51  116#58
     DRAW6A FDD7    117/10  117#14
     DRAW6B FDEB    117/15  117#24
     DRAW7  FDF1    117/19  117/21  117/23  117/25  117#27
     DRAW8  FDF6    117/ 9  117/13  117#29
     DRAW8A FE0A    117#36  117/47
     DRAW8B FE27    117/43  117#48
   n DRAW8C FE13    117#41
     DRAW9  FE30    117/32  117#51
     DRAWA  FD0C    115/12  115#18
     DRAWB  FD0B    115/14  115#17
   n DRAWLN 0011      3#35
     DRETRI 0001     40#16   42/ 7
     DRETRY 0037      5#49   42/ 8   44/ 5
     DRKMSK 004E      6#15   34/24   34/54
   n DSKFMS 0018      5#14
     DSKIF  EDF0     60/35   61#14
     DSKINV E453      2#61   85/11   87/ 6
     DSKORG EDEA      2#22   59/25   60/44
     DSKRDE F381     85/25   85/59   86#29
   n DSKSPR 0014     62#36
     DSKTIM 0246      8#17   61/ 7   61/16   61/48
   n DSKUTL 001A      5#15
     DSPFLG 02FE     10#10   97/ 5
     DSTAT  004C      6#12   91/24   91/53   94/10   95/52   95/54   96/18   96/39   96/48   98/19  103/18
103/26  103/42
                    107/27  107/35
     DSTATS 0303     10#25   42/60   43/35   44/17   50/ 8   61/35   62/12   65/57   75/21
     DT1    00FA     67#14   70/27   73/18
     DTA    00FC     67#13   71/16
```

```
  DTIMLO 0306     10#28   53/36   61/21   66/ 6   75/15
  DUNIT  0301     10#23   42/21   65/51   75/13   85/ 8   87/ 5
n DUNUSE 0307     10#29
  DVSTAT 02EA      9#50   60/11   60/12   64/ 8   66/14
n EDITOR E400     89#11
  EDITRV E400      2#49   78/ 5   81/53   81/55   84/40   89/ 7
n EEXP   00ED     12#35
  EGETC1 F650     96#17   96/28
  EGETC2 F66E     96/21   96#29
  EGETC3 F67C     96/12   96#35
  EGETC5 F691     96/36   96/38   96#44
  EGETC6 F66B     96/26   96#28
  EGETC7 F680     96#37   96/40
  EGETCH F63E     89/13   96# 9
  ENBOOT F35A     85/55   86# 6
  ENDACK F1BD     80/33   80/35   80#42
  ENDBCK F1AC     80/25   80/27   80#31
  ENDDIF EE69     61/51   62#12
  ENDPT  0074      6#46  106/43  106/46  116/18  116/28  116/30  116/46  116/50  117/ 8  117/12
  ENDRAM F276     83/54   84# 5
n ENINTP ECEB     55#40
  ENSPE2 F257     83/34   83#46
  ENSPEC F254     83/37   83/39   83#45
n ENTVEC 002C     18# 7
n EOF    0088     76#22
  EOFERR 0088      4#24   70/33   98/18
  EOL    009B     16# 9   21/40   22/19   23/43   24/13
  EOPEN  F3FC     89/11   91#12   94/13  106/60
  EOT    00FE     67#15   70/25   73/26
  EOUTC5 F6BE     96/58   97# 5
  EOUTC6 F6B5     96#59   97/ 7
  EOUTCH F6A4     89/14   96#53
  ERANGE FA88     96/10   96/55  106#55
n ERETN  F6BB     96#61
  ERRFLG 023F      8# 8   42/47   43/27   43/38   43/51   44/29   45/17
  ERROR  0045     39#35   44/59
  ERRTN  E4C0     17#39   17/40   17/41
  ERRTNH 00E4     17/28   17#40   20/15
  ERRTNL 00C0     17/26   17#41   20/17
  ESCAPE F779     99# 9  119/34
  ESCFLG 02A2      9#13   96/59   97/ 6   97/ 8   99/10
n ESIGN  00EF     12#37
  ESTSCM F173     79/32   79#47
n EXP    DDC0     12#19
n EXP10  DDCC     12#20
  EXTEN1 FB7E    109#53  110/ 8
n EXTEN3 FB8B    110# 5
  EXTEN4 FB92    110/ 6  110# 9
  EXTEND FB7B    101/35  109#51
n FADD   DA66     12# 5
n FASC   D8E6     11#57
  FCAX   F032     73#12   73/20   73/28
n FCHRFL 00F0     12#38
n FDIV   DB28     12# 7
  FEOF   003F      5#59   68/19   70/ 8   70/32
  FILDAT 02FD     10# 9  117/44
  FILFLG 02B7      9#17  115/18  117/31
```

```
  FILLBF EEC3     65# 8   65/11
n FILLIN 0012      3#36
n FINDX  ECD6     55#23
n FLD0P  DD8D     12# 9
n FLD0R  DD89     12# 8
n FLD1P  DD9C     12#11
n FLD1R  DD98     12#10
n FLOPPY 0030     39# 8
n FLPTR  00FC     12#49
n FMOVE  DDB6     12#14
n FMSZPG 0043      6# 7
n FMTD   EE4A     61#52
n FMUL   DADB     12# 6
  FNCNOT 0092      4#35   27/11
  FOMAT  0021     60#24   61/18   61/50
  FOOEY  EADE     47/ 8   47#17
n FORMAT 0022      3#39
n FPI    D9D2     11#60
  FPREC  0006     11#53   12/30   12/31   12/32   12/33   12/58   12/59
  FPSCR  05E6     12#58   12/59   12/60
  FPSCR1 05EC     12#59   12/61
n FPTR2  00FE     12#50
n FR0    00D4     12#30
n FR1    00E0     12#32
n FR2    00E6     12#33
n FRE    00DA     12#31
  FREQ   0040      5#60   74/11   74/33
  FRMADR 0068     88#11  101/40  101/45  109/29  109/36  109/38
  FRMERR 008C      4#28   48/27
n FRX    00EC     12#34
n FSCR   05E6     12#60
n FSCR1  05EC     12#61
n FST0P  DDAB     12#13
n FST0R  DDA7     12#12
n FSUB   DA60     11#61
  FTYPE  003E      5#58   68/ 9   75/22   87/24
  GBX    EFE8     70#16   70/20
  GBYTE  EFD6     67/23   70# 8   70/31
n GETCAR 0007     76#11
  GETCH  F593     89/26   94#30   96/41
n GETCHR 0007      3#27
  GETDAT 0040     60#26   61/22
  GETOUT F749     97/33   98#21
  GETPLT F5A2     94/31   94#35   95/44  100/41  100/54  101/24  111/56  117/42
n GETREC 0005      3#26
  GETSEC F39D     85/23   85/57   86/ 8   86#56
n GLBABS 02E0      9#42
n GOBACK EB09     47#54
  GOERR  EF3D     66/36   66#42
  GOHAND E689     19/28   20/12   20/34   21/23   21/28   22/13   23/32   24/14   27#11
  GOOD   EA65     44/55   44/57   45/18   45#22
  GOODST EE32     61/40   61#42
  GOON   EB64     49/19   49#27
  GOREAD ED6F     57/ 9   57#18
  GPRIOR 026F      8#21   34/47   92/ 5   92/24
n GRACTL D01D     14#21
n GRAFM  D011     14# 9
```

```
   n GRAFP0 D00D     14# 5
   n GRAFP1 D00E     14# 6
   n GRAFP2 D00F     14# 7
   n GRAFP3 D010     14# 8
     HARDI  F277     79/ 8   84#13
     HATABS 031A     10#50   10/51   26/19   26/21   28/13   80/10
   n HDR    00FB     67#16
   n HITCLR D01E     14#22
     HITIMR ECCC     55#16   55/20
     HITONE 0005     39#44   51/50
     HOLD1  0051      6#21   91/54   93/18   93/34   93/42  101/58  104/ 6  104/14  111/ 8  111/ 9  111/16
112/23
     HOLD2  029F      9#10  104/ 8  104/17
     HOLD3  029D      9# 8  106/17  106/30  106/34
     HOLD4  02BC      9#21  117/35  117/48
   n HOLD5  02BD      9#22
     HOLDCH 007C      6#52   97/40   98/23   98/33
     HOME   F7D6     99#54  107/33
     HOWMCH F25F     83#50   83/61
   n HPOSM0 D004     13#53
   n HPOSM1 D005     13#54
   n HPOSM2 D006     13#55
   n HPOSM3 D007     13#56
   n HPOSP0 D000     13#49
   n HPOSP1 D001     13#50
   n HPOSP2 D002     13#51
   n HPOSP3 D003     13#52
   n HSCROL D404     14#50
     ICAX1  034A     11# 9   80/50
     ICAX1Z 002A      5#36   21/ 8   23/ 8   68/10   91/12   91/14   92/51   94/17   97/31
   n ICAX2  034B     11#10
     ICAX2Z 002B      5#37   66/25   68/ 8   91/ 9
     ICBAH  0345     10#61   25/15   80/48   86/43
     ICBAHZ 0025      5#31   25/16
     ICBAL  0344     10#60   25/13   80/46   86/41
     ICBALZ 0024      5#30   21/32   23/30   25/14   26/48   26/50   28/10   28/30
     ICBLH  0349     11# 8   26/60
     ICBLHZ 0029      5#35   27/ 5
     ICBLL  0348     11# 7   26/57   86/47
     ICBLLZ 0028      5#34   21/20   21/21   23/20   23/21   23/24   26/38   26/39   26/42   26/43   26/58
26/59   26/61
     ICCOM  0342     10#58   80/44   86/45
     ICCOMT 0017      5#12   18/46   19/32   26/23
     ICCOMZ 0022      5#28   18/37   21/ 7   21/34   22/ 8   23/ 7   23/37   24/ 8  115/10
   n ICDNO  0341     10#57
     ICDNOZ 0021      5#27   28/35
     ICHID  0340     10#56   17/25   20/38
     ICHIDZ 0020      5#26   19/ 9   20/14   20/24   20/39   26/10   28/27
     ICIDNO 002E      5#39   18/13   20/37   25/12   25/27   26/55   27/24
     ICPTH  0347     11# 6   17/29
     ICPTHZ 0027      5#33   19/37   20/16
     ICPTL  0346     11# 5   17/27
     ICPTLZ 0026      5#32   19/35   20/18
   n ICSPR  034C     11#11
     ICSPRZ 002C      5#38    5/39    5/40   18/ 7   19/34   19/36   26/20   26/22   26/26   26/29   26/30
26/31   27/19
                      27/21
   n ICSTA  0343     10#59
     ICSTAZ 0023      5#29   20/ 9   22/24   25/ 8   25/28   27/13
     IDENT  F0F2     78#17   78/19   78/20   78/22
```

```
     IDENTH 00F0     78#19   81/45
     IDENTL 00F2     78#20   81/44
   n IFP    D9AA     11#58
     IHINIT E6D5     30/17   31#12
   n IMASK  028B      8#51
     INATA1 FB4A    108/60  109#13
     INATAC FB32     94/32  108#57
   n INBUFF 00F3     12#41
     INC2A  F9F8    105/25  105/30  105#32
     INCBF1 E676     26/49   26#51
     INCBFP E670     21/33   23/34   26#48
     INCRS1 FA77    105/44  106/16  106/19  106/21  106#36
     INCRS2 F9E4    105/20  105#22
     INCRS3 F9F7    105/27  105/29  105#31  105/35
     INCRSA F9DC    100/48  105#18  117/38
     INCRSB F9D4     94/33  101/17  105#14  111/49
     INCRSC F9DA    105/15  105#17
     INCRSR F9D8     95/ 5  105#16
     INIMLH 0007     76#14   84/38
     INIMLL 0000     76#13   84/36
     INIT   EF41     67/24   67#43
   n INSCH1 F852    100#52
     INSCH3 F85E    100/51  100#58
     INSCH4 F844    100#45  100/57
     INSCH5 F86A    100/60  101# 6
     INSCH6 F861    100#59  101/ 5
     INSCHR F837    100#40  120/ 7
   n INSCLR 0020      3#54
     INSDAT 007D      6#53  100/42  100/52  100/55  105/17  105/41  105/51  105/59  106/20
     INSLI1 F8C6    101#51  101/54
     INSLI2 F8CE    101/50  101#55
     INSLIA F8A5    101#35  105/54
     INSLIN F8A4    101#34  119/54
     INTABS 0200      7#19   30/19   31/35   31/36   31/37   31/38   31/39   31/40   31/41   31/42   32/15
32/17   41/17
                     79/56   89/49
     INTATA FEFA    109/12  120#20
     INTEMP 022D      7#44   37/13   37/19
     INTINV E46B      3#12   30/16   84/47
     INTORG E6D5      2#20   29/15   31/11
   n INTSPR 0014     37#40
   n INTTBL EC84     53#59
     INTZBS 0010      5# 5   79/41
     INVFLG 02B6      9#16   97/54   97/56   98/38
     IOC1   E4D6     18/20   18#27
     IOC1A  E4D8     18#28   18/33
     IOC2   E4F3     18/44   18#46
     IOC6   E514     19/11   19#18
   n IOC7   E519     19#24
     IOCB   0340     10#55   18/28   25/19
     IOCBAS 0020      5#25   18/29   25/18
     IOCBSZ 0010      5#23    5/24   11/12   17/32
     IOCFRE 00FF      3#44   17/24   19/10   20/13   20/25
     IRQEN  D20E     13#45   31/49   31/51   32/10   32/12   46/38   47/15   52/29   53/14   91/28
     IRQST  D20E     13#31   31/45   31/61
     ISEOF  F00D     70/ 9   70#33
     ISRODN EA90     41/20   46#13   53/60   54/ 8   54/ 9
     ISRSIR EB0F     41/19   48#16   53/59   54/ 6   54/ 7
```

```
     ISRTD  EACF     41/21   47# 7   53/61   54/10   54/11
     JMPP   E735     32/ 5   32# 9
     JSRIND F6A1     96#51   97/14
     JTADRH 00EB     51#26   58/ 9
     JTADRL 00EC     51#27   58/ 7
     JTIMER EBEC     51#25   51/26   51/27
     JTIMR1 E8CA     34/29   36#32
     JTIMR2 E8CD     35/ 8   36#33
     JVECK  028C      8#52   32/16   32/18   32/21   32/40   32/45   32/48
     K1     F729     97/59   98# 6
     K2     F734     98/ 7   98#11
     K3     F73F     98/12   98#16
     K4     F776     98/36   98#40
     K5     F768     98/25   98/28   98/30   98/32   98#35
     K6     F74D     98/17   98#23
     K7     F745     97/36   98#19
     K8     F773     98/22   98#39
     KBCODE D209     13#27   35/38  121/10  121/15
   n KBD    0000      3#58
   n KBDHND E420     89#38
     KBDORG F3E4      2#26   87/47   90/ 5
   n KBDSPR 0014    121#38
     KEYBDV E420      2#51   74/50   74/52   78/ 9   84/42   89/33
     KEYDEL 02F1      9#55   35/24   35/26  121/13  121/25
     KGETC1 F71E     97/53   97#58
     KGETC2 F6DD     97#29   97/51   97/57   98/ 5   98/10   98/15
     KGETC3 F6FE     97#44   98/34
     KGETCH F6E2     89/40   96/17   97#31   97/39
   n LBFEND 05FF     13# 5
   n LBPR1  057E     12#54
   n LBPR2  057F     12#55
     LBUFF  0580     12#56   12/57
     LDPNTR EB6A     43/14   43/47   49#43   50/34   50/61
     LEDGE  0002      6#18   79/47
   n LENGTH 022F     29#12
     LFRTCM F7A5     99#30   99/38
     LINBUF 0247      8#19  109/22  109/24
   n LINZBS 0000      4#46
   n LIRG   0000     76#23
     LL     E72F     31/60   32# 6
     LMARGN 0052      6#22   79/48   94/22   99/27   99/37  100/ 7  100/10  100/26  101/36  102/19  110/19
111/52  111/60
                     112/25  112/36  112/41  112/46  114/17
     LO1GET FB22    108#42  109/54  111/10
     LO2GET FB23    102/ 6  108#43
   n LOCKFL 0023      3#40
   n LOG    DECD     12#21
   n LOG10  DED1     12#22
     LOGCOL 0063      6#34   96/24   99/55  100/ 6  100/32  100/36  100/38  100/46  100/50  101/15  101/19
105/18  105/38
                     111/ 6  111/12  111/15  111/19  111/21  111/30  111/51  111/59  112/13  112/35  112/40
     LOGGET FB20    100/29  102/23  108#41
     LOGMAP 02B2      9#15   99/50  102/16  102/18  106/31  108/20  108/21  108/22
     LOOPM  E71F     31#56   32/ 7
     LOOPM2 E72A     31/58   31#61
     LOTONE 0007     39#45   51/48
     LPENH  0234      7#52   34/40
     LPENV  0235      7#53   34/38
   n M0PF   D000     14#24
```

```
   n M0PL   D008     14#32
   n M1PF   D001     14#25
   n M1PL   D009     14#33
   n M2PF   D002     14#26
   n M2PL   D00A     14#34
   n M3PF   D003     14#27
   n M3PL   D00B     14#35
     MASKTB FEB9    108/10  119#20
     MAXDEV 0021     10#51   26/11   28/12
     MAXIOC 0080      5#24   11/12   17/34   18/19
     MEMLO  02E7      9#48   84/37   84/39
     MEMTOP 02E5      9#47   84/33   84/35   90/ 9   93/56   93/58
     MLTTMP 0066      6#36    6/37   88/12  103/57  103/59  103/60  104/ 5  104/ 7  104/10  104/12  104/15
104/16  104/18
                     104/23  104/24  104/38  104/39  104/41  104/48  104/53
     MODATA E9F0     43/36   43#47
   n MODEM  0000      4# 5
     MONORG F0E3      2#25   75/38   77/50
   n MONSPR 0014     87#47
     MOTRGO 0034     40# 7   50/28   50/55   68/23   69/ 6
     MOTRST 003C     40# 8   41/29   51/16   57/45   73/12
     MOVLI1 FB58    109#27  109/34
     MOVLI2 FB7A    109/43  109#45
     MOVLIN FB4E    101/51  109#22
     MOVVEC F17D     79#55   79/58
     MVBUFF F32D     85#42   85/58
     MVNXB  F32F     85#43   85/46
     MXDMDE FE5D     93/13  118#18
   n MXDMOD 0010      3#53
     N      004E     63#26   66/26   66/42
   n NACK   004E     39#33
     NBUFSZ 0028     63#19   66/28
     NCOMHI 003C     40# 6   41/32   47/44
     NCOMLO 0034     40# 5   42/42
     NEWCOL 0061      6#33  115/22  115/24  115/41  115/44
     NEWROW 0060      6#32  115/20  115/29
     NLR    F005     70/28   70#30
     NMIEN  D40E     14#58   31/13
     NMIRES D40F     14#59   33/22
     NMIST  D40F     14#60   33/10   33/14
     NOA1   F1F1     81/12   81#14
     NOA2   F212     81/27   81/30   81#32
     NOB1   F1F8     81/15   81#17
     NOBOOT F1FF     81/18   81#24
     NOCAR2 F220     81/33   81#41
     NOCART F1FC     81/10   81#21   81/36
     NOCKSM 003C      5#54   44/41   49/18   49/22
     NOCLR  EAE9     47/33   47#36
     NOCSB2 F3BF     87/17   87#19
     NOCSBT F3E0     87/22   87#36
   n NODAT  0000     60#25
     NOFUNC F63D     89/16   89/41   89/43   95#56
     NOINIT F2DC     84/61   85# 6
     NOISE1 EC45     52/38   52#40   52/43
     NOKEY  F2CE     84/50   84#52
     NOMOD  F4AB     92/34   92#38
     NONDEV 0082      4#18   28/21
   n NORMAL 004E     39#25
```

```
  NOROWS FE99    106/14  107/ 9  118#60
  NOSCR1 FA32    105/57  105#59
n NOSCRL FA2C    105#56
  NOT8   F48B     92/10   92/21   92#24
  NOTCAS EC0C     51/44   51#53
  NOTCST E96B     41/56   41#61
  NOTDER EA52     44/60   45# 9
  NOTDON EA81     45#52   45/57
n NOTE   0026      3#43
  NOTEND EABE     46/24   46#46
  NOTERR EA00     43/52   43#58
  NOTMXD F4F5     92/50   92/53   93#16
  NOTOPN 0085      4#21   17/42   26/15
  NOTYET EB3C     48/37   48#56
  NOWARM F2DD     84/58   85# 7
  NOWRP0 EA98     46/17   46#20
n NSIGN  00EE     12#36
  NTBRK0 EA88     45/53   45#56
  NTBRK1 EB00     47/47   47#50
  NTBRK2 ED17     56/28   56#31
  NTFRAM EB1D     48/25   48#30
  NTOVRN EB25     48/31   48#36
  NTWRP1 EB50     49/ 9   49#12
  NUMDLE FE51     93/13   93/17  118#17
  NVALID 0084      4#20   18/36  115/15
  NWOK   EA56     44/48   45/ 7   45#12
  NXTENT F18C     80# 9   80/12
n ODNHI  00EA     54# 8
n ODNLO  0090     54# 9
  OFFCRS FAE4     96/29   96/56  107#47
  OKTIM1 ED1F     56/34   56#37
  OKTIMR ED48     56/53   56#57
  OLDADR 005E      6#31  104/52  104/56  107/49
  OLDCHR 005D      6#30   95/45  107/48
  OLDCOL 005B      6#29  115/42  115/45
  OLDROW 005A      6#28   95/13  115/30  116/ 7
  OPEN   0003      3#25   18/38
  OPENC  EF4C     67/23   68# 8
  OPINP  EF5D     67/35   68/13   68#17
  OPNCOM F404     91/11   91#16
  OPNEDT F118     78#33   78/35   78/36
n OPNERR F453     91#52
  OPNH   00F1     78#35   80/47
  OPNIN  0004      3#50    3/52
n OPNINO 000C      3#52
  OPNL   0018     78#36   80/45
  OPNOT  0008      3#51    3/52
n OPNOUT 0002     63#18
  OPNRTN EF8F     68/22   68#47   68/55
  OPNTMP 0066      6#37   92/ 7   92/16   92/45   93/14   93/15   93/16
  OPOK   EFD3     68/41   69#26
  OPOUT  EF95     68/15   68#51
n OPSYS  F17B     79#54
  OSRAM  F28A     79/59   84#26
  OUTCH  F5B7     89/27   94#48
  OUTCH2 F5FF     95#27   95/51  100/45
n OUTCHA F5BD     94#51
```

```
     OUTCHB F5D7     94/58   94#61
     OUTCHE F5CA     94/53   94#56   96/60
     OUTPLT F5E0     94/61   95# 9   95/10  100/22  117/30  117/46
     OVRRUN 008E      4#30   48/33
   n P0PF   D004     14#28
   n P0PL   D00C     14#36
   n P1PF   D005     14#29
   n P1PL   D00D     14#37
   n P2PF   D006     14#30
   n P2PL   D00E     14#38
   n P3PF   D007     14#31
   n P3PL   D00F     14#39
     PACTL  D302     15# 7   31/15   31/21   32/31   41/30   50/29   50/56   51/17   57/46   68/24   69/ 7
73/13
     PADDL0 0270      8#23   35/61
   n PADDL1 0271      8#24
   n PADDL2 0272      8#25
   n PADDL3 0273      8#26
     PADDL4 0274      8#27   36/ 6
   n PADDL5 0275      8#28
   n PADDL6 0276      8#29
   n PADDL7 0277      8#30
     PAGETB FE75     92/33  118#43
     PALFLG 0000      2#11   39/47   39/53   55/ 9   55/14   56/ 7   56/10   68/25   68/29   69/ 9   69/13
74/14   74/17
                     74/37   74/40
     PBCTL  D303     15# 8   31/16   31/22   32/35   41/33   42/43   47/45   57/47
     PBPNT  001D      5#18   64/43   64/53   64/59   65/13   65/26
     PBRK   EF8B     68#45   69/21
     PBUFSZ 001E      5#19   64/24   64/57   65/10   65/58   66/38
     PBYTE  F010     67/23   71# 8
     PCOLR0 02C0      9#27   34/52
   n PCOLR1 02C1      9#28
   n PCOLR2 02C2      9#29
   n PCOLR3 02C3      9#30
     PDEVN  0040     63#22   65/48
     PENH   D40C     14#56   34/39
     PENV   D40D     14#57   34/37
     PHACR1 FC9F    113#17  113/20
     PHACRS FC9D    100/40  101/ 9  111/29  113#16  117/33
     PHCHLO EE7F     64# 9   65/14   65/15
     PHCLOS EEDC     63/39   65#25
     PHINIT EE78     63/44   63#58
     PHOPEN EE9F     63/38   64#41
     PHPUT  EF14     64/33   66#14
     PHSTAT EE81     63/42   64#23   64/41
     PHSTLO EE7D     64# 8   64/25   64/26
     PHWRIT EEA7     63/41   64#51
     PIA    D300     14#61   15/ 5   15/ 6   15/ 7   15/ 8
     PIRQ   E6F3     31#24   37/33   37/34
     PIRQ1  FFDC    121/17  121#22
   n PIRQ2  FFF0    121#30
     PIRQ3  FFCB    121/12  121#15
     PIRQ4  FFEB    121/14  121/21  121#28
     PIRQ5  FFBE     89/50  121#10
     PIRQH  00E6     30/37   37#33
     PIRQL  00F3     30/39   37#34
     PLACR1 FCAA    113#27  113/30
     PLACRS FCA8    100/58  101/32  112/14  113#26  117/50
```

```
   n PLOT   0050     39#27
     PLUS   ECF8     55/44   55#46
     PLYARG 05E0     12#57   12/58
   n PLYEVL DD40     12#15
   n PMBASE D407     14#52
     PNMI   E791     33#10   37/35   37/36
     PNMI1  E799     33/11   33#13
     PNMIH  00E7     30/41   37#35
     PNMIL  0091     30/43   37#36
   n POINT  0025      3#42
     POKEY  D200     13#17   13/18   13/19   13/20   13/21   13/22   13/23   13/24   13/25   13/26   13/27
13/28   13/29
                     13/30   13/31   13/32   13/33   13/34   13/35   13/36   13/37   13/38   13/39   13/40
13/41   13/42
                     13/43   13/44   13/45   13/46
     POKMSK 0010      5# 6   31/50   31/59   32/11   46/35   46/37   47/12   47/14   51/57   52/26   52/28
53/12   53/13
                     91/26   91/27
     POKTAB EDD0     55/47   55/50   58#54
     PORTA  D300     15# 5   31/18   32/33   35/43   35/50
     PORTB  D301     15# 6   31/19   32/37
     POT0   D200     13#18   35/60
   n POT1   D201     13#19
   n POT2   D202     13#20
   n POT3   D203     13#21
     POT4   D204     13#22   36/ 5
   n POT5   D205     13#23
   n POT6   D206     13#24
   n POT7   D207     13#25
     POTGO  D20B     13#29   36/ 9
   n PRINTR 0000      3#60
     PRINTV E430      2#52   77/58   84/43
     PRIOR  D01B     14#19   34/48
     PRMODE EF1A     64/52   65/25   66#24
     PRNBUF 03C0     11#14   64/ 9   64/55   65/ 8
     PRNORG EE78      2#23   62/36   63/51
   n PRNSPR 0014     66#53
     PRVOPN 0081      4#17   19/14
     PSIOC  EF01     65/55   65#57
     PTEMP  001F      5#20   64/51   64/54
     PTIMOT 001C      5#17   63/59   66/ 5   66/15
     PTRIG0 027C      8#35   36/20
     PTRIG1 027D      8#36   36/17
   n PTRIG2 027E      8#37
   n PTRIG3 027F      8#38
   n PTRIG4 0280      8#39
   n PTRIG5 0281      8#40
   n PTRIG6 0282      8#41
   n PTRIG7 0283      8#42
     PTRLP  E8AC     36#13   36/24
     PUTADR EE6D     61/45   61/52   62#23
     PUTBC  EE43     61/44   61#49
   n PUTCAR 000B     76#12
     PUTCHR 000B      3#29   19/31
     PUTCNT EE21     61/29   61#35
     PUTDAT 0080     60#27   61/27
     PUTDTO EE01     61/19   61#21
     PUTLIN F385     81/46   86#39
     PUTMSC FCF3     99/39  103/55  114#24
   n PUTREC 0009      3#28
```

```
   n PUTSEC 0050     60#20
     PUTTXT 0009     76#10   86/44
     PWRONA F3E4     89/17   89/30   89/44   90# 7
     PWRUP  F125     77/40   77/44   78/48   78#58   80/54
     PWRUP1 F128     78/50   78#60
   n RADFLG 00FB     12#46
   n RADON  0000     12#47
     RAMLO  0004      4#50   79/14   79/15   79/16   79/20   79/21   83/47   83/50   83/52   83/53   83/56
85/35   85/37
                      85/44   85/48   85/50   85/51   85/53   86/17   86/20   86/21
     RAMSIZ 02E4      9#46   80/23   80/31   84/32
     RAMTOP 006A      6#39   90/11   91/44   91/55   92/61   93/26   99/47  110/41  110/55
   n RANDOM D20A     13#28
     RANGE  FA96     94/30   94/49  106/57  106/59  106#61  117/29  117/41
     RANGE1 FAB7    107/14  107#20
     RANGE2 FABB    107/19  107#22
     RANGE3 FA9E    107/ 6  107# 8
     RBLOK  EFE9     67/32   70/12   70#17
     RBLOKV E47A      3#17   67/31   86/58
     RCI1   E5A7     21/36   21/41   21#46
   n RCI11  E5BF     22#23
     RCI1A  E574     21/ 9   21#16
     RCI1B  E571     21#13   21/17
   n RCI2   E5AC     22# 8
     RCI3   E587     21/22   21#28   21/47
     RCI4   E5C3     21/30   21/43   22/10   22/15   22#27
     RCI6   E5B2     22#13   22/20
     RDBAD  EE51     61#56   62/ 6
     RDBYTE F310     85#30   85/33
     RDONLY 0087      4#23   23/12
     READ   0052     39#16   86/59
     RECEIV EAE0     43/49   44/43   47#29   51/11
     RECVDN 0039      5#51   47/37   47/52   48/47
   n RECVDS EC60     53#11
     RECVEN EC1B     47/43   52#17
     REDGE  0027      6#19   79/49
     RELONE EAB1     46/27   46#35
   n RENAME 0020      3#37
     RESET  F11B     77/36   77/46   78#46
     RETUR1 F634     89/12   89/15   89/25   89/28   89/38   89/39   89/42   94/34   95/47   95/49   95#52
98/40  107/41
                     113/38  117/60
     RETUR2 F621     94/27   94/55   94/60   95/ 6   95#44   96/47   97/15
     RETUR3 FAE1    107/39  107#41
     RETURN EA0D     43/40   43/60   44/ 6   44#13   44/18   51/19   57/58
     RIRGHI 0000     39#60   50/52
     RIRGLO 0078     39#55   50/51
     RMARGN 0053      6#23   79/50   99/29   99/34  100/15  105/28  107/ 5  107/ 7  111/45  112/ 7  112/44
   n RNGER1 FAD8    107#36
     RNGER2 FAD6    107/31  107#35
     RNGERR FAD1    107/11  107/12  107/18  107/21  107/24  107/25  107#33
     RNGOK  FAC4    107/16  107#26
     ROWAC  0070      6#44  106/42  106/44  106/45  106/47  116/ 6  116/34  116/40  116/42  116/44  116/45
116/49
     ROWCRS 0054      6#24   95/12   96/13   96/32   99/12   99/16   99/18   99/19   99/58  100/17  100/61
101/21  101/48
                     101/59  101/60  103/49  103/56  105/ 8  105/46  106/ 6  106/15  106/35  107/10  108/41
110/ 7  110/ 9
                     111/ 7  111/33  111/42  111/54  112/ 9  112/38  113/17  113/28  113/43  113/46  115/19
116/ 8  116/53
                     116/55  117/36  117/40
     ROWINC 0079      6#49  115/26  115/34  116/54
   n RRETRN EB0E     48# 6
```

```
  RSIRG  000A     39#57   50/47
  RTCLOK 0012      5# 8   34/11   34/14   34/16   34/22   56/42   57/11   74/12   74/31   74/44
  S      0053     63#28   66/35
  SAVADR 0068      6#38   88/11   92/39   92/41   93/59   93/61  101/12  101/14  101/26  101/30
  SAVIO  0316     10#43   56/40   56/59   56/61
  SAVMSC 0058      6#27   92/26   92/28   93/38   93/40  110/35  110/37  114/24  114/26
  SBUFSZ 001D     63#21   66/37
  SCOLLP E80E     34#51   34/57
n SCREDT 0000      3#57
  SCRENV E410      2#50   78/ 7   84/41   89/20
  SCRFLG 02BB      9#20  100/44  100/59  106/28
  SCRMEM 0093      4#36  103/41
  SCRNOK F1DB     80/53   80#55   80/56   80/58
  SCROL1 FBB7    102/22  110#39  110/52  110/56
  SCROL2 FBCA    110/44  110/47  110#49
  SCROLL FBAC    106/27  110#34
  SDLSTH 0231      7#48   34/41
  SDLSTL 0230      7#47   34/43   93/50   93/52   94/ 6   94/ 8
  SDMCTL 022F      7#46   34/45   91/22   94/25   94/26
  SECT1  F301     85#23   87/27
  SECTX  F34C     85#57   86/ 5
  SEND   EA6B     45#35   50/36   54/26
  SENDDS EC5F     44/13   45/59   53#10   57/44
  SENDEN EBF2     41/15   45/38   50/18   51#38
  SENDEV E468      3#11   41/14   69/ 5
  SENDIN EC8A     42/45   43/16   54#19
  SERIN  D20D     13#30   48/39   48/56
  SEROUT D20D     13#44   45/47   46/30   46/48
  SETBSZ EF34     66/30   66/34   66#38
  SETDCB EEE6     64/30   65/16   65#46
  SETLOP E8F7     37#17   37/18
  SETTAB F82D    100#36  119/58
  SETVBL E8ED     30/11   30/13   37#12
  SETVBV E45C      3# 7   30/12   58/15   68/35   69/17
  SETVBX EDB9     50/26   50/53   51/ 7   54/30   58# 7
  SEX    0000     76#19   80/43   86/40
n SFH    EF64     68#20
  SHFAMT 006F      6#43   94/38   95/30  104/59
  SHFLOK 02BE      9#23   90/13   97/61   98/ 9   98/14   98/31
  SHIFT1 F5B1     94/39   94#42
  SHIFT2 F610     95/31   95#34
  SHIFTD F5AA     94#38   94/41
  SHIFTU F608     95#30   95/33
n SIDWAY 0053     39#24
  SIGNON F223     77/32   81#44
n SIN    BD81     12#24
  SIO    E959     41/ 9   41#49
  SIOINT E944     41/12   41#29
  SIOINV E465      3#10   41/11   84/46
  SIOORG E944      2#21   37/40   41/25
  SIOSB  F095     70/18   74#58   75/34
n SIOSPR 0014     59#25
  SIOV   E459      3# 6   41/ 8   61/39   64/31   65/17   75/24
n SIRHI  00EB     54# 6
n SIRLO  000F     54# 7
n SIZEM  D00C     13#61
n SIZEP0 D008     13#57
```

```
   n SIZEP1 D009     13#58
   n SIZEP2 D00A     13#59
   n SIZEP3 D00B     13#60
     SKCTL  D20F     13#46   41/39   51/54   52/21
     SKRES  D20A     13#43   48/22   52/23
     SKSTAT D20F     13#32   35/20   35/30   48/21   56/37   56/57   57/23   57/25
     SOUNDR 0041      5#61   41/38   52/37
     SPACE  0020     63#25   65/ 7
     SPECIA EF4B     67/23   67#47
     SPECIL 000E      3#32   18/43   18/45
     SPECL  F23F     79/ 7   83#33
   n SQR    BEB1     12#27
     SRETRN EB34     48/41   48#46   49/24
     SRSTA  0040     67# 7   75/17
     SRTIM2 0006     30# 6   35/36
     SRTIMR 022B      7#42   35/28   35/33   35/37   36/30  121/29
     SRTIR0 EB9B     50/22   50#25
     SRTIR1 EBC1     50/49   50#52
     SRTIR2 EBE9     51/14   51#19
     SSFLAG 02FF     10#11   32/24   95/ 9  121/18  121/20
     SSKCTL 0232      7#49   41/37   51/39   51/53   52/18   52/20   57/24
     STACKP 0318     10#45   41/50   57/52
     STATC  0053     60#23   61/28   61/43   64/27   65/54   85/ 9
   n STATIS 000D      3#31
     STATU  F028     67/23   72# 8
     STATUS 0030      5#42   43/55   43/58   44/16   44/46   45/ 6   45/10   45/12   45/22   45/36   47/42
47/56   48/28
                     48/34   48/44   57/50
     STATVH 0002     60#11   61/32
     STATVL 00EA     60#12   61/30
     STICK0 0278      8#31   35/48   35/52   36/13
   n STICK1 0279      8#32
   n STICK2 027A      8#33
   n STICK3 027B      8#34
   n STIMER D209     13#42
     STLOOP E877     35#43   35/55
     STORE  F917     92/44   92/58   93/ 7   93/ 9   93/11   93/19   93/29   93/31   93/33   93/35   93/39
93/41   93/44
                     93/46   93/48   93/54   94/ 7   94/ 9  103#18
   n STORE1 F91D    103#21
     STRBEG FC5C     99/17  112#22
     STRERR F942    103/37  103#41
     STRIG0 0284      8#43   35/59
   n STRIG1 0285      8#44
   n STRIG2 0286      8#45
   n STRIG3 0287      8#46
     STRL   E890     35#58   36/ 8
     STROK  F946    103/19  103/27  103/36  103/40  103#43
     STTMOT EC75     43/24   51/ 6   53#36
     SUBBFL E677     22/27   24/17   26#55
     SUBEND FA7A    106#41  116/57  117/28
     SUBTMP 029E      9# 9  103/25  103/30
     SUCCES 0001      4#14   20/ 8   43/59   44/47   45/35   47/41   65/ 5   65/28   69/26   70/15   71/11
72/ 8   73/11
                     91/23   95/53  107/26
     SUSUAL EB38     48#49   49/16   49/30
   n SV7H   00E8     36#27
   n SV7L   0073     36#28
     SWAP   FCB3     96/ 9   96/23   96/43   96/49   96/54   96/61   97/16  113#37
     SWAP1  FCC2    113#43  113/50
```

```
  SWAP3  FCD7    113/41  113#54
  SWAPA  FCB9    107/40  113/37  113#39
  SWPFLG 007B      6#51   91/32  106/ 9  107/38  111/39  113/51  113/53  114/13
  SWSTA  0080     67# 8   75/20
  SYIRQ  E706     30/30   31#44
  SYIRQ2 E71B     31/47   31#53
  SYIRQ8 E762     32/ 8   32#29
  SYIRQ9 E76F     32/32   32#35
  SYIRQA E77A     32/36   32#39
  SYIRQB E78F     30/22   30/23   30/24   30/27   30/28   30/29   32#50
  SYRTI  E790     30/21   32#51
  SYRTI2 E78B     32/44   32#48
  SYSVB1 E7BA     34/12   34/15   34#17
  SYSVB2 E7D6     34/28   34#30
  SYSVB3 E7E5     34/35   34#37
  SYSVB4 E832     35/ 7   35# 9
  SYSVB6 E8C3     35/32   36#29
  SYSVB7 E873     35/29   35/34   35#41   36/27   36/28   36/31
  SYSVBA E844     35/14   35#17
  SYSVBB E834     35#10   35/18
  SYSVBL E7AE     30/14   30/32   34#11
n SYSVBV E45F      3# 8
  SYVB6A E857     35/22   35/25   35#28
  TAB    F810    100#24  100/34  119/48
  TAB1   F823    100/27  100/30  100#32
  TAB2   F82A    100/31  100#35
  TABMAP 02A3      9#14   91/36  108/29  108/31  108/37  108/38  108/46
  TBLENT F0E3     77#57   78/22   80/ 9
  TBLLEN 000E     78#22   80/ 8
n TDHI   00EA     54#10
n TDLO   00CF     54#11
  TEMP   023E      8# 6   40/10   40/11   44/53
  TEMP1  0312     10#40   54/61   55/22
n TEMP2  0314     10#41
  TEMP3  0315     10#42   56/47   57/ 8
  TEMPHI 0002     40#10   44/36
  TEMPLO 003E     40#11   44/32
  TIMER1 030C     10#35   54/55   54/57   54/60   55/ 7   56/43   56/44
  TIMER2 0310     10#38   54/51   54/52   54/54   54/58   55/ 5
  TIMFLG 0317     10#44   47/50   50/31   50/58   51/28   56/33   56/52   58/17
  TIMIT  EBA5     50#31   50/32
  TIMIT1 EBCB     50#58   50/59
  TIMOUT 008A      4#26   45/13   47/55
  TINDEX 0293      8#61   91/30
  TMPCHR 0050      6#20   95/35   95/39  109/28  109/31
n TMPCOL 02B9      9#19
  TMPLBT 02A1      9#12  104/34  104/43  104/47
  TMPROW 02B8      9#18  113/18  113/27
n TMPX1  029C      9# 7
  TOADR  0066     88#12  101/43  101/47  109/27  109/30  109/40  109/42  109/44
  TONE1  0002     67#17   68/53
  TONE2  0001     67#18   68/20
  TOUT   EB0A     47/51   47#55   56/55
  TOUT1  ED44     56/35   56#54
  TRAMSZ 0006      4#51   79/22   80/22   80/36   81/ 8   81/11   81/26   83/49   83/57   83/60   84/31
  TRIG0  D010     14#40   35/58
n TRIG1  D011     14#41
```

```
n TRIG2  D012     14#42
n TRIG3  D013     14#43
  TRNRCD 0089      4#25   22/23
  TSTAT  0319     10#46   43/54   45/23
  TSTCT1 FC8F    112#60  113/ 9
  TSTCT2 FC9C    113/ 5  113#10
  TSTCTL FC8D     96/57   98/35  112#59
  TSTDAT 0007      4#52   80/21   80/28   81/ 9   81/14   81/32
  TWICE  EE4F     61#54   61/58
  TXTCOL 0291      8#60   94/23
  TXTMSC 0294      9# 5   91/46   91/48
n TXTOLD 0296      9# 6
  TXTROW 0290      8#59   94/21  113/45  113/48
n UNLOCK 0024      3#41
  UPDNCM F787     99#16   99/23
n USAREA 0480     11#29
  VBATRA E7C8     34/20   34#24
  VBREAK 0206      7#23   32/47
  VBWAIT F496     92#29   92/31
  VCOUNT D40B     14#55   56/41   57/10   92/29
  VCTABL E480      2#18   30/19   41/17   79/55   89/49
n VDELAY D01C     14#20
  VDSLST 0200      7#20   30/19   33/12
n VECTBL E400      2#17
  VIMIRQ 0216      7#31   31/24
  VINTER 0204      7#22   32/38
  VKEYBD 0208      7#24   31/36   89/49
  VPRCED 0202      7#21   32/34
n VSCROL D405     14#51
  VSERIN 020A      7#25   31/42   31/52   41/17
  VSEROC 020E      7#27   31/40
  VSEROR 020C      7#26   31/41
  VTIMR1 0210      7#28   31/39
  VTIMR2 0212      7#29   31/38
  VTIMR4 0214      7#30   31/37
  VVBLKD 0224      7#38   36/26
  VVBLKI 0222      7#37   33/23
  WAIT   EA1A     44#28   54/32
  WAITER EC9B     43/29   54#30
  WAITTM EF7C     68#36   68/37
  WARMST 0008      4#55   78/60   79/ 9   79/13   84/57   87/13
  WARMSV E474      3#15   33/17   77/34
  WATCOM E9D7     42/61   43#24
  WCI1   E605     23/39   23/44   23#49
  WCI1A  E5D4     23/ 9   23#16
  WCI1B  E5D1     23#13   23/17
n WCI2   E60A     24# 8
  WCI3   E5E5     23/22   23#29   23/50
  WCI4   E5EB     23/25   23#32
  WCI5   E615     23/33   23/46   24/10   24#17
  WDLR   EFC6     69#20   69/23
  WFAK   F087     74/34   74#47
  WFAK1  F08C     74/47   74#50
  WFL    F060     74#21   74/32
  WIRGHI 0000     39#59   50/25
  WIRGLO 00B4     39#54   50/24
  WMODE  0289      8#49   68/18   68/48   68/52   73/ 8
```

```
n WOK    EA3D     44#53
  WRITE  0057     39#17   61/25
  WRITEC 0057     63#24   66/24
  WRONLY 0083      4#19   21/12
  WSIOSB F0D2     71/17   73/19   73/27   75#29
  WSIRG  000F     39#56   50/20
  WSYNC  D40A     14#54   37/16  113/61
  WTLR   F046     73/16   73#21
  XBOOT  F361     86/ 7   86# 9
  XITVBL E905     30/15   30/33   34/36   37#27
n XITVBV E462      3# 9
  XMTDON 003A      5#52   45/43   45/56   47/10
  XXIT   E7E2     34/31   34#36
  ZERIT  EC6D     53#18   53/21
n ZERORM F138     79#12
n ZIOCB  0020      5#22
  ZOSRAM F160     79/10   79#35
  ZOSRM2 F163     79#37   79/40
  ZOSRM3 F16E     79#42   79/44
  ZTBUF  F04A     73#23   73/25
n ZTEMP1 00F5     12#42
n ZTEMP3 00F9     12#44
n ZTEMP4 00F7     12#43
```